

COM303: Digital Signal Processing

Lecture 1: Introduction to Signal Processing

- ▶ practical information
- ▶ signal processing: history and philosophy
- ▶ transoceanic signal transmission

<http://com303.learndsp.org/>

No paper handouts, go to the website for:

- ▶ syllabus
- ▶ weekly announcements
- ▶ weekly homework
- ▶ handouts, extra chapters, slides
- ▶ login with your SCIPER number, email me if access doesn't work

<https://www.coursera.org/learn/dsp/>

- ▶ all lectures available on video
- ▶ auto-graded exercises
- ▶ forum
- ▶ more than 120K students worldwide



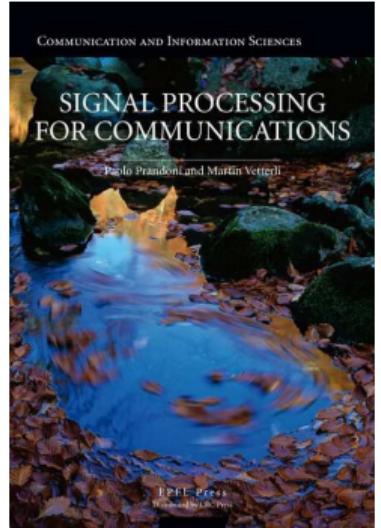
► lectures:

- Mondays 2pm-4pm, INF1
- Tuesdays 10am-12pm, CE2

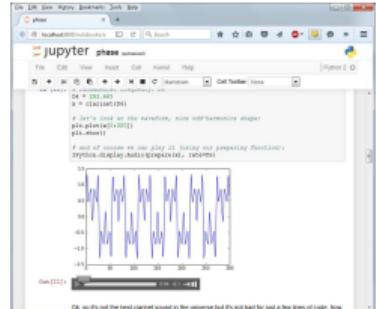
► exercise sessions:

- Thursdays 2pm-4pm, INM201
- [details on the website](#)

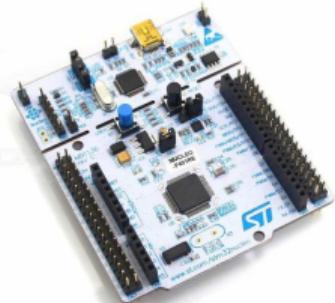
- ▶ Textbook: *Signal Processing for Communications*, EPFL Press, 2008, by P. Prandoni and M. Vetterli. Paper version available at PPUR, Amazon.
- ▶ Free PDF and HTML versions available online:
<http://www.sp4comm.org/>
- ▶ Weekly homework sets
- ▶ Occasional handouts
- ▶ Jupyter notebooks in Python



- ▶ It's important to implement what we learn to get a good feeling for what DSP can do
- ▶ We will be using Python:
 - real programming language
 - great numerical libraries (numpy, matplotlib)
 - **Jupyter notebooks!!**
- ▶ if in doubt: just download and install Anaconda



- ▶ five optional applied DSP labs
- ▶ get a feeling for what it's like to implement DSP algorithms on "real" hardware
- ▶ we will be using the ST Nucleo board
- ▶ Adrien Hoffet will be your guide



► Lectures:

- communication should be **two-way**
- ask questions!

► Exercise sessions:

- go to the exercise sessions but **do the exercises first!**
- tell us what you want to see explained
- come to office hours

► Exam:

- exam is not necessarily hard but cannot be done with pattern matching
- learn to “think” and solve problems like an engineer!

- ▶ final exam only (perfect final = 6)
- ▶ mock midterm on April 16
- ▶ homework is not graded

- ▶ download the math self-test from the website
- ▶ do the test on your own and without the internet!
- ▶ review the topics in which you don't feel confident, you'll be happy you did

Questions?

(digital) signal processing for communications

Description of the evolution of a physical phenomenon

- ▶ temperature (weather)
- ▶ pressure (sound)
- ▶ magnetic deviation (recorded sound)
- ▶ gray level on paper (photograph)
- ▶ ...

Description of the evolution of a physical phenomenon

- ▶ temperature (weather)
- ▶ pressure (sound)
- ▶ magnetic deviation (recorded sound)
- ▶ gray level on paper (photograph)
- ▶ ...

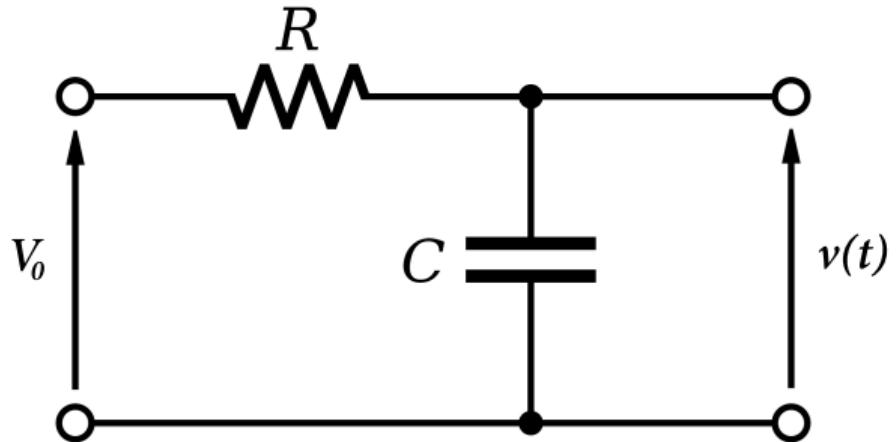
Analysis: *understanding* the information carried by the signal

Synthesis: *creating* a signal to contain the given information

Reception: *analysis* of an incoming signal

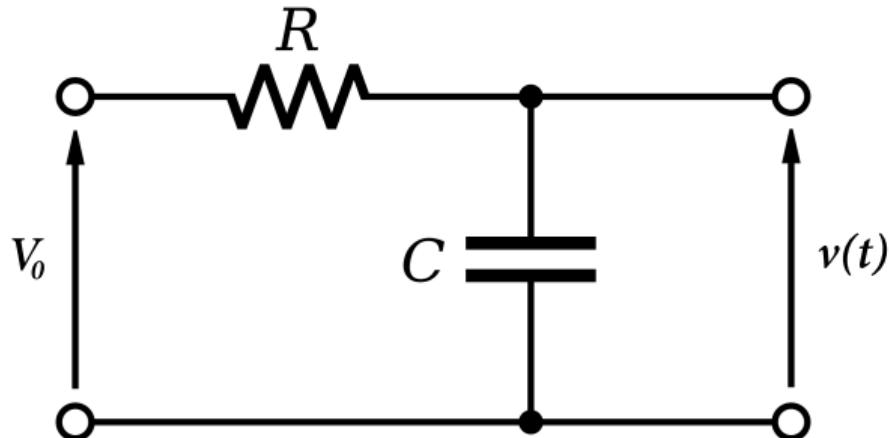
Transmission: *synthesis* of an outgoing signal

Description of the evolution of a physical phenomenon



$$v(t) = V_0(1 - e^{-\frac{t}{RC}})$$

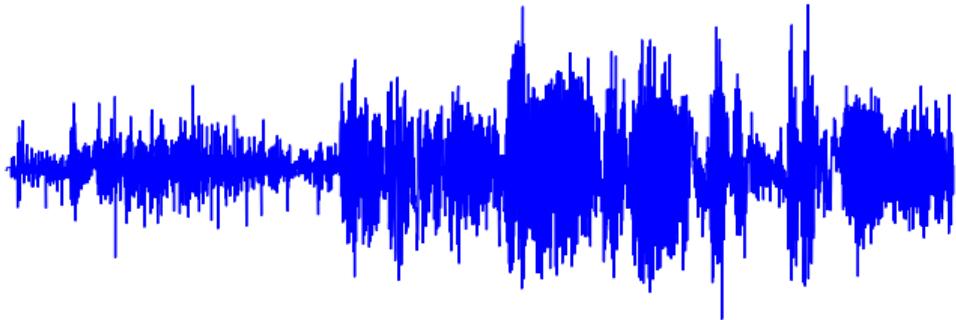
$$f : \mathbb{R} \rightarrow \mathbb{R}$$



$$v(t) = V_0 \left(1 - e^{-\frac{t}{RC}}\right)$$

only 2 degrees of freedom: R, C

What about “interesting” signals?

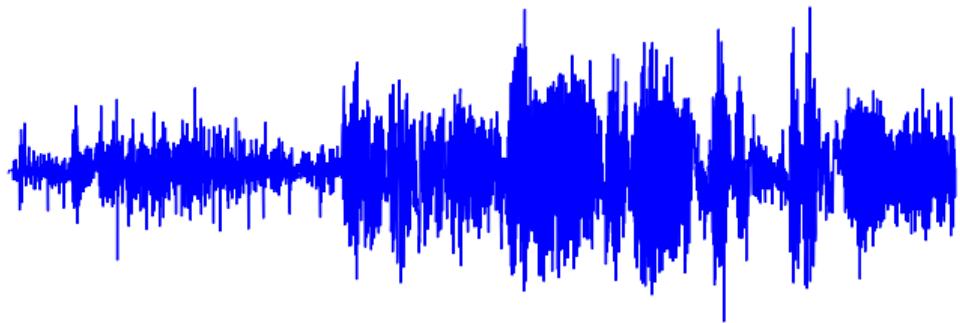


$$f(t) = ?$$

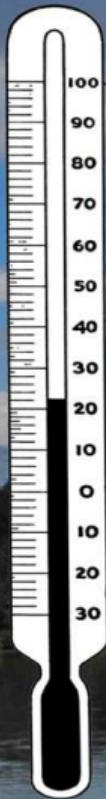
Key ingredients:

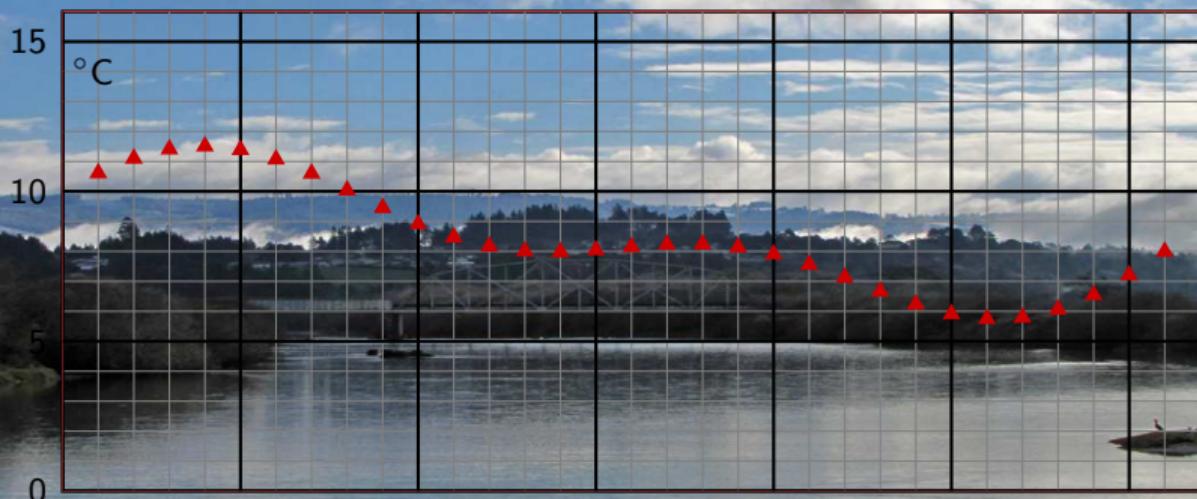
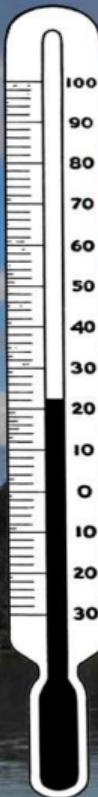
- ▶ discrete time
- ▶ discrete amplitude

From analog...



```
... 74 31 -66 9 -123 33 159 -26 102 148 86  
-136 -179 70 72 -84 -113 -42 -88 88 8 -180 -7  
-133 8 164 -4 108 35 -82 74 -49 52 32 -31 ...
```





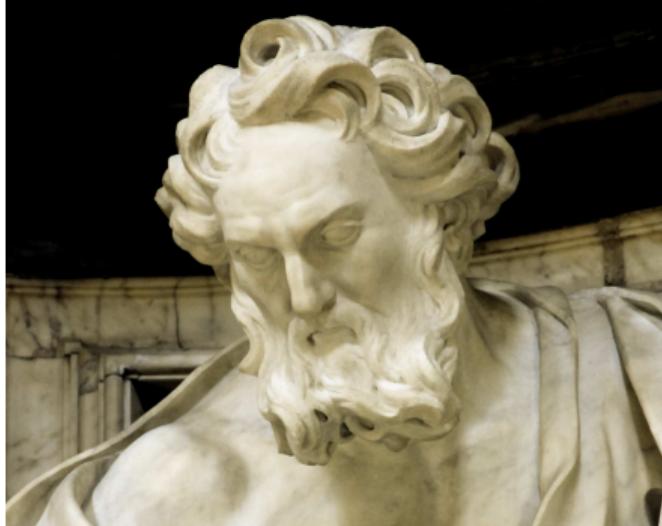
What is time?

What is time?



Immanuel Kant

A very old phylosophical problem



Zeno of Elea

The Dichotomy Paradox



The Dichotomy Paradox



The Dichotomy Paradox



The Dichotomy Paradox



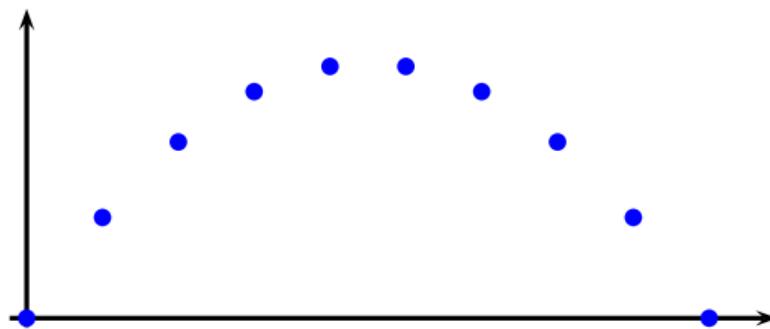
The Dichotomy Paradox



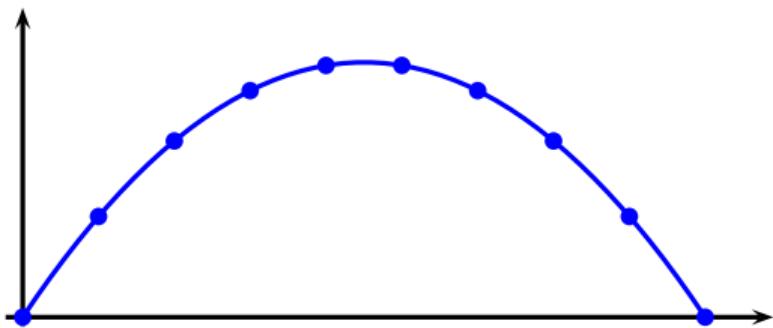
The Dichotomy Paradox



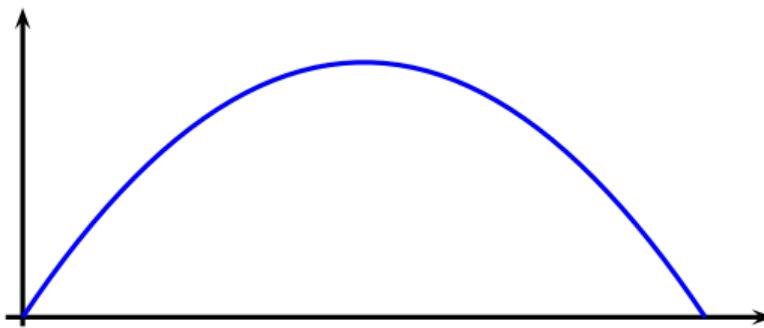
$$\sum_{n=1}^{\infty} \frac{1}{2^n} = 1$$



$$\vec{x}(t) = \vec{v}_0 t + (1/2)\vec{g} t^2$$



$$\vec{x}(t) = \vec{v}_0 t + (1/2)\vec{g} t^2$$



$$\vec{x}(t) = \vec{v}_0 t + (1/2)\vec{g} t^2$$

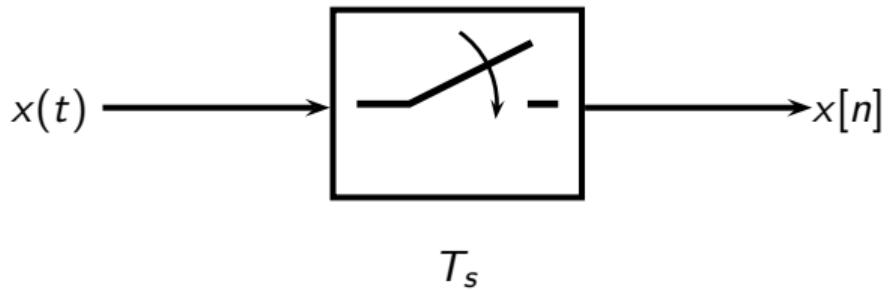
$$x : \mathbb{R} \rightarrow \mathbb{V}$$

$$\cancel{f : \mathbb{R} \rightarrow \mathbb{V}}$$

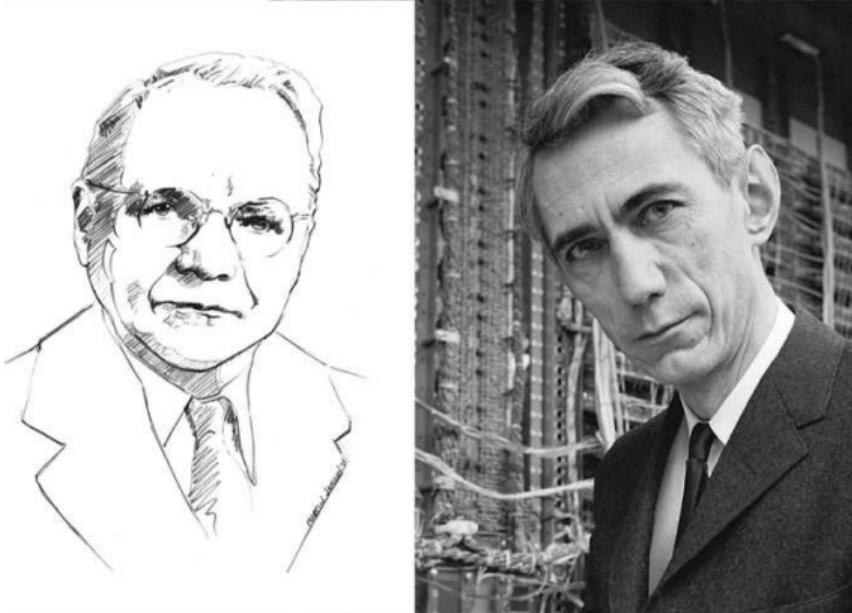
$$x : \mathbb{Z} \rightarrow \mathbb{V}$$

$$x[n] = \dots, 1.2390, -0.7372, 0.8987, 0.1798, -1.1501, -0.2642 \dots$$

Are we losing information?



Translating between languages: the founding fathers

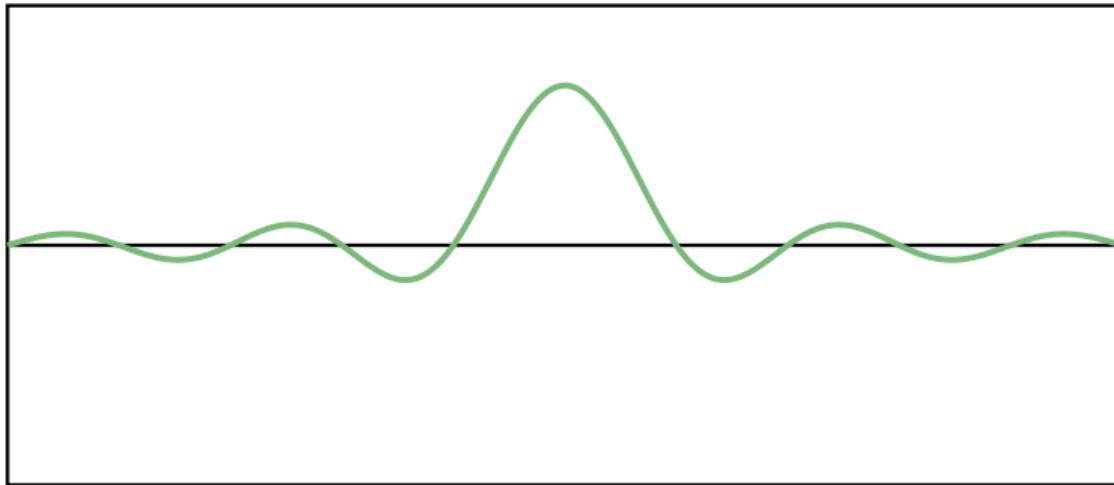


The Sampling Theorem (1920)

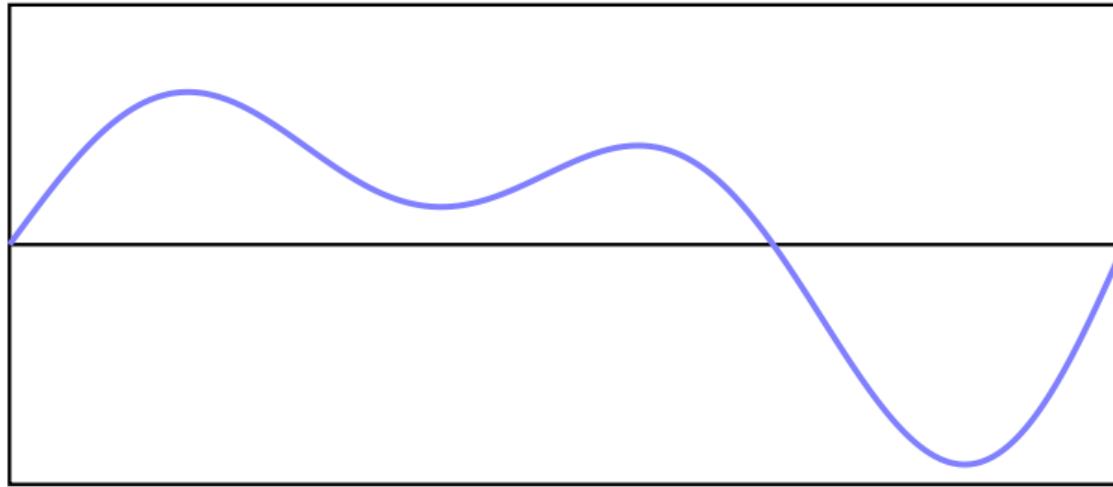
Under appropriate “slowness” conditions for $x(t)$ we have:

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right)$$

The sinc function

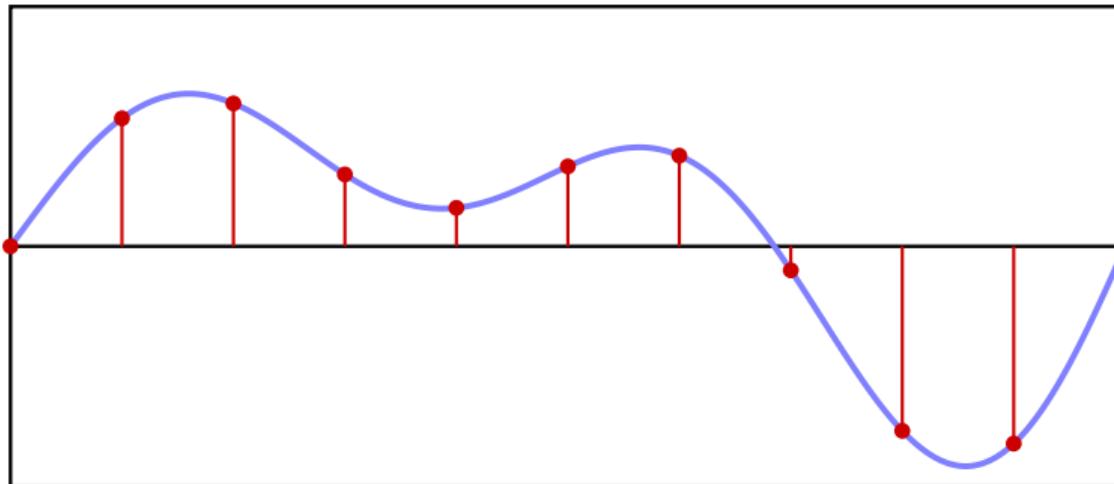


From continuous to discrete time and back!

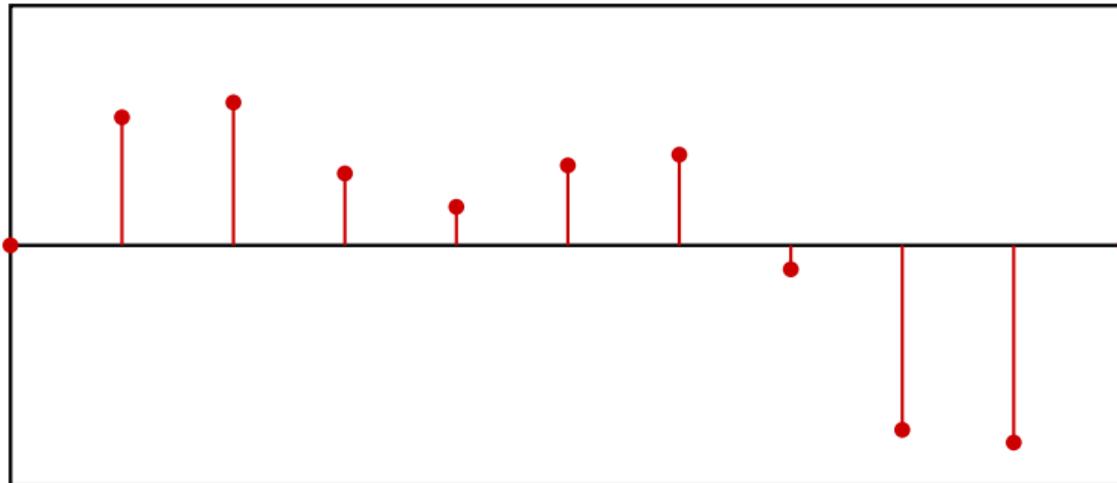


$$x(t)$$

From continuous to discrete time and back!

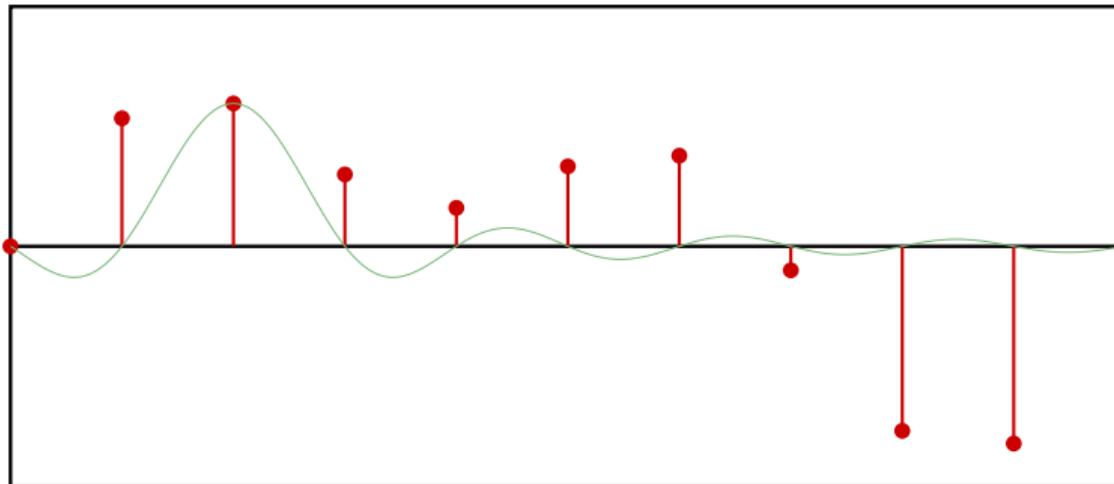


From continuous to discrete time and back!

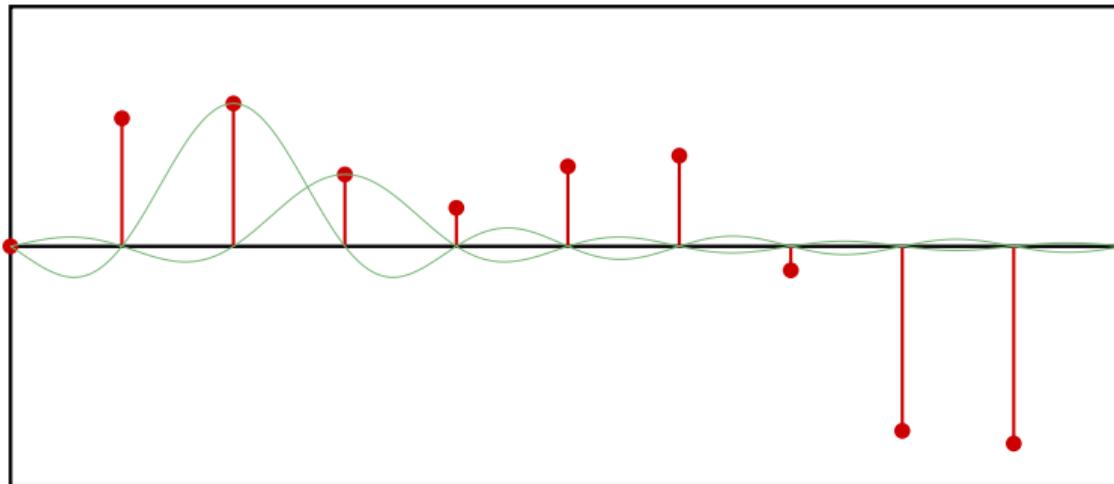


$x[n]$

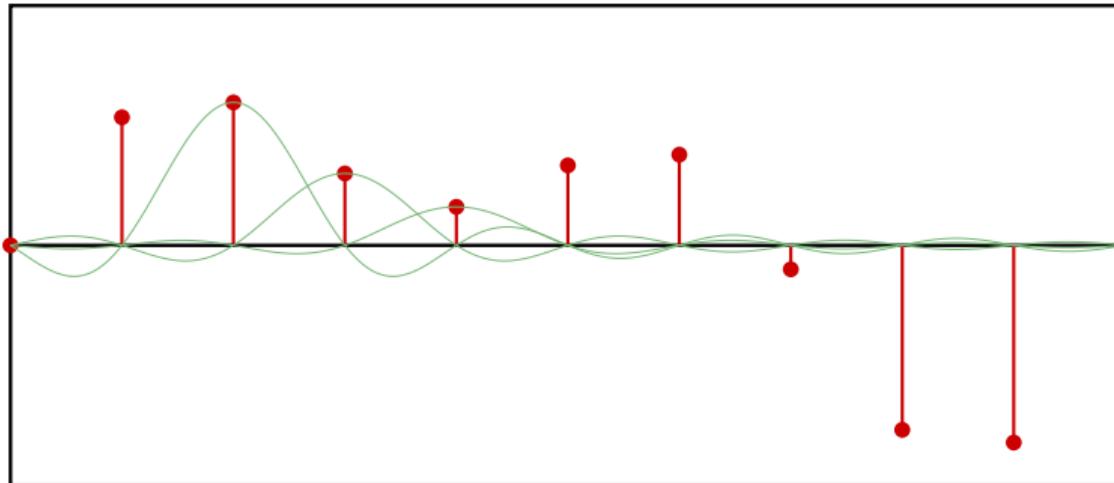
From continuous to discrete time and back!



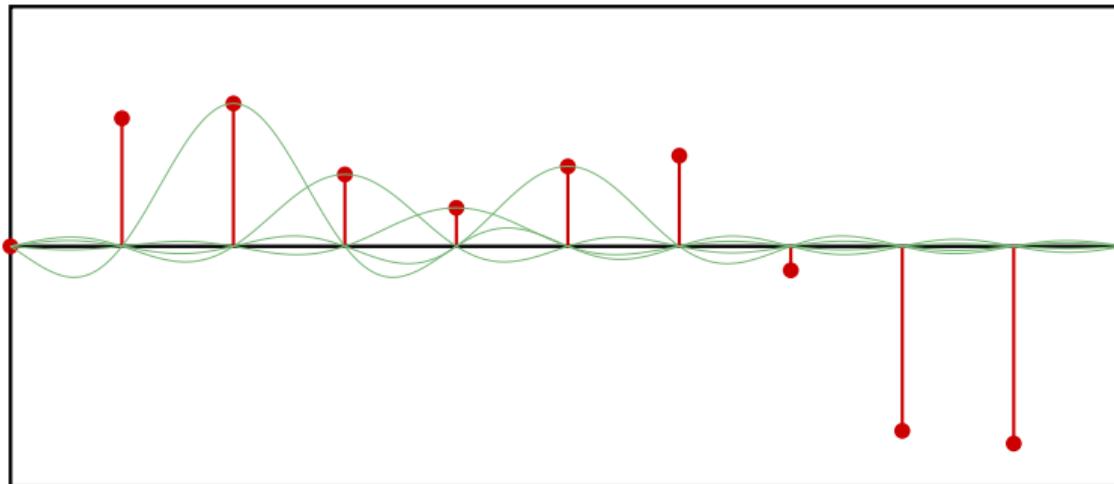
From continuous to discrete time and back!



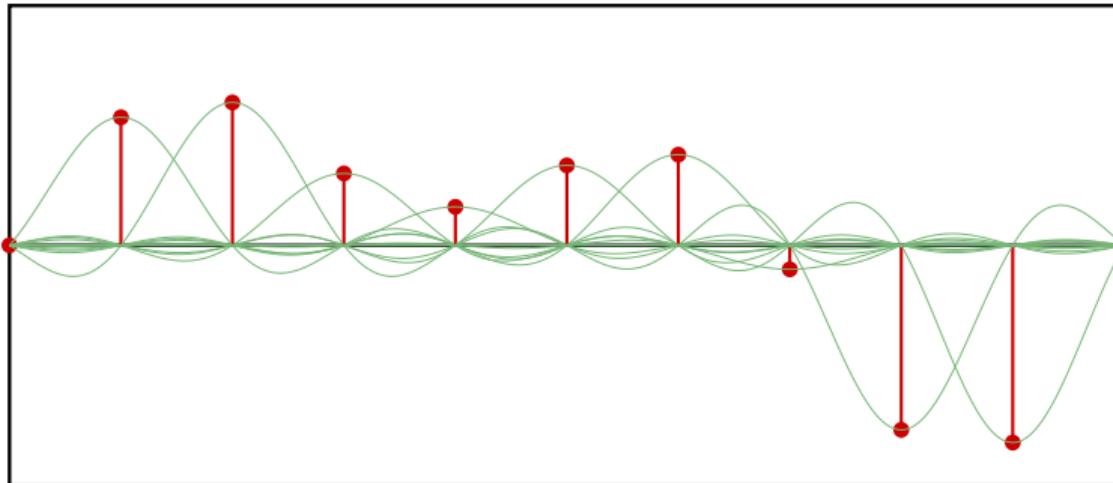
From continuous to discrete time and back!



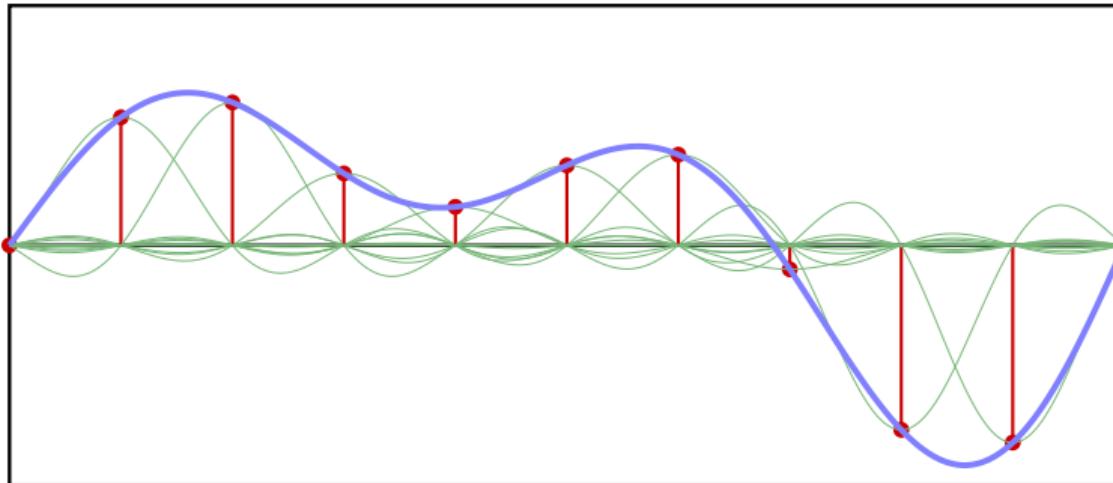
From continuous to discrete time and back!



From continuous to discrete time and back!



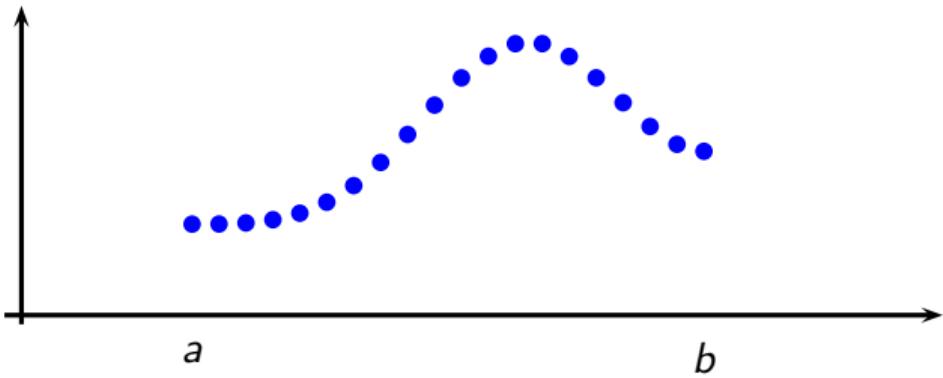
From continuous to discrete time and back!



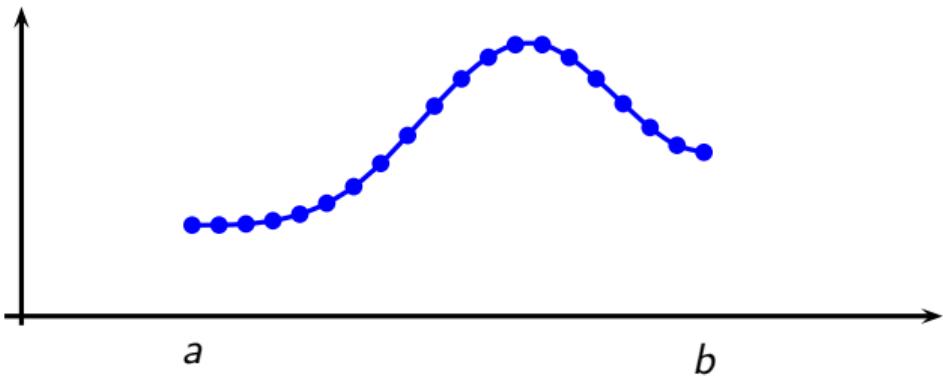
When can we do all this? Ask Fourier!



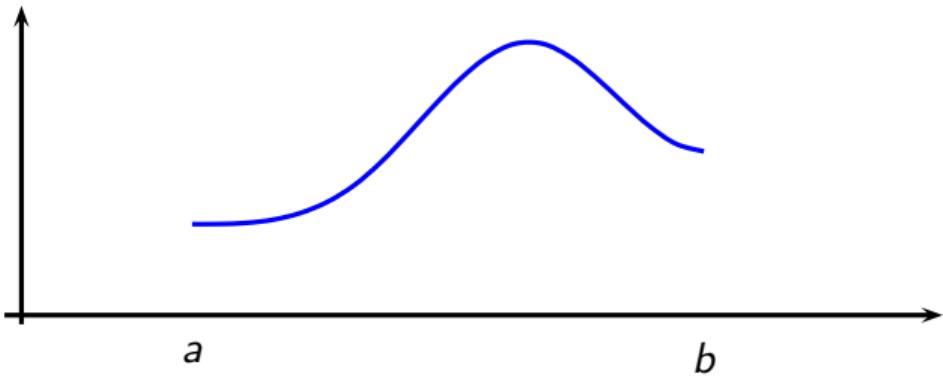
By the way, discrete time is easy!



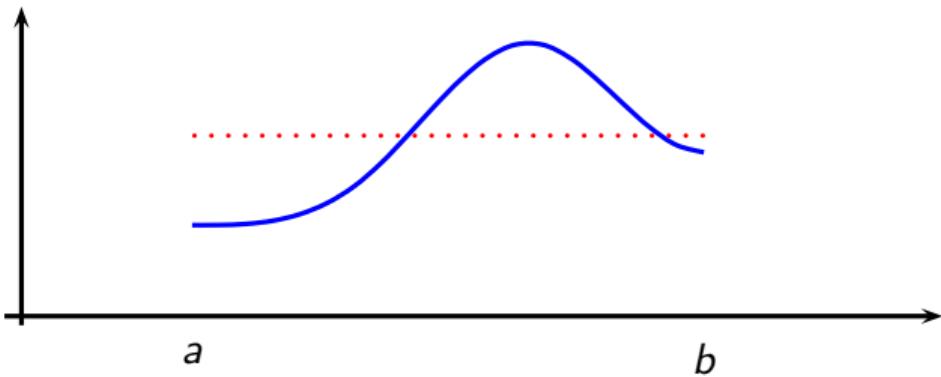
By the way, discrete time is easy!



By the way, discrete time is easy!

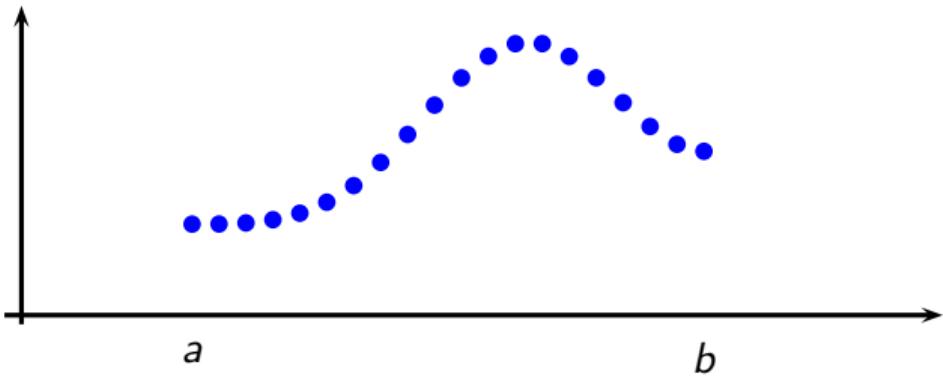


By the way, discrete time is easy!

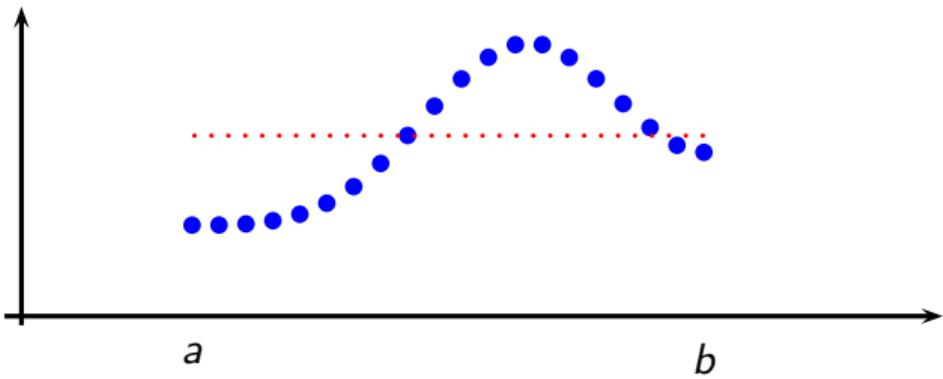


$$\bar{x} = \frac{1}{b-a} \int_a^b f(t) dt$$

By the way, discrete time is easy!



By the way, discrete time is easy!



$$\bar{x} = \frac{1}{N} \sum_{n=0}^{N-1} x[n]$$

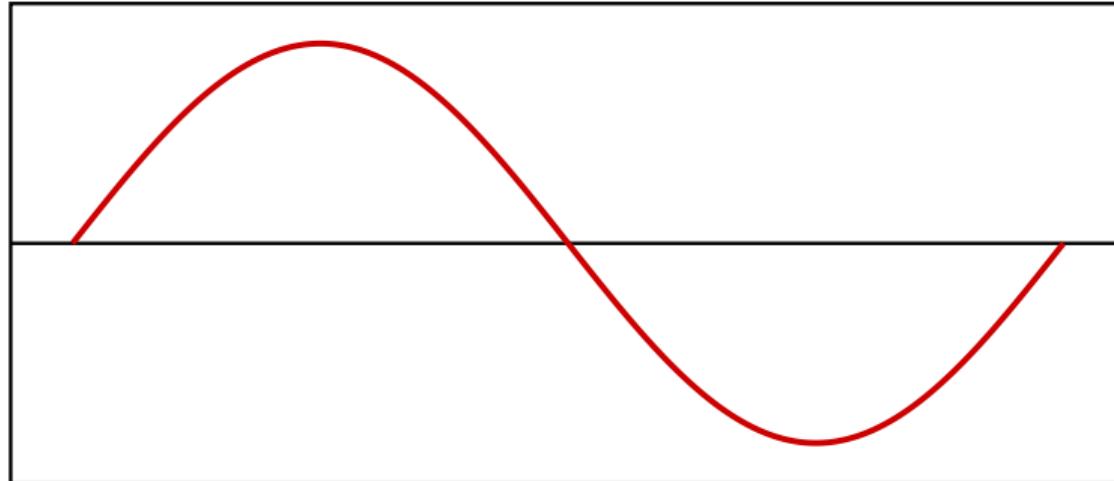
(digital) signal processing for communications

Key ingredients:

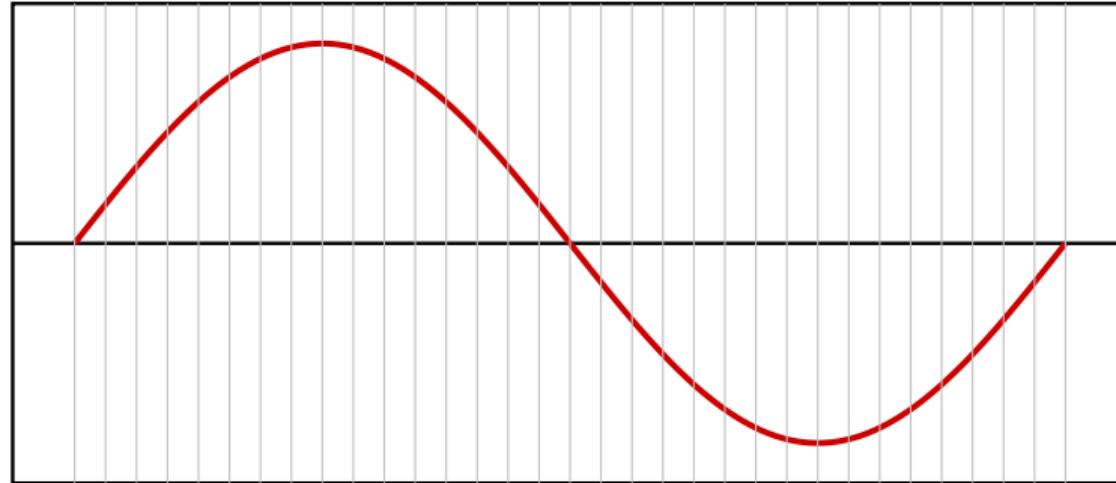
- ▶ discrete time
- ▶ **discrete amplitude**

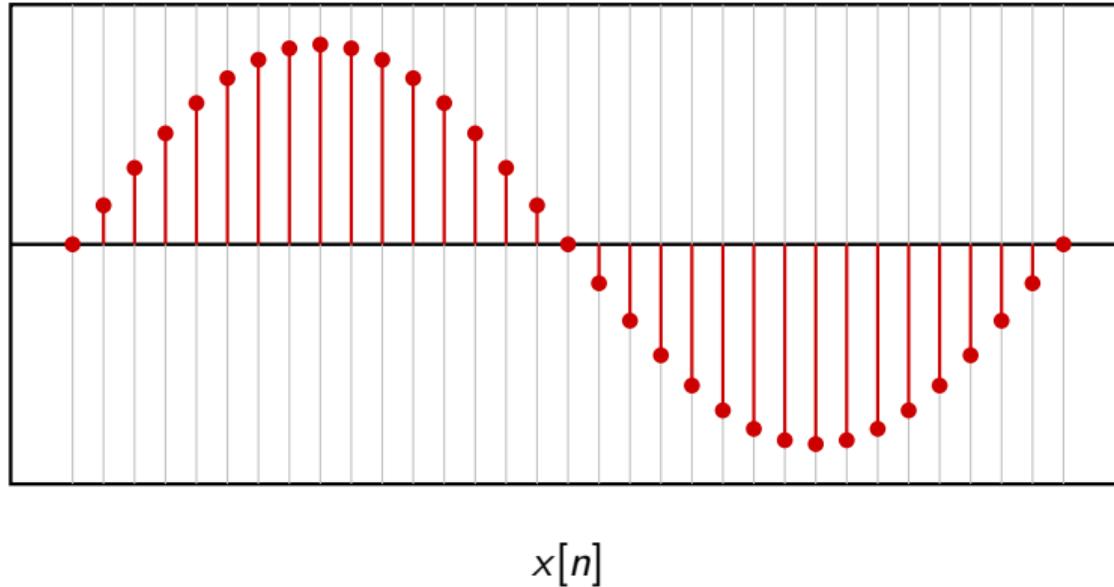
$$x : \mathbb{Z} \rightarrow \mathbb{Z}$$

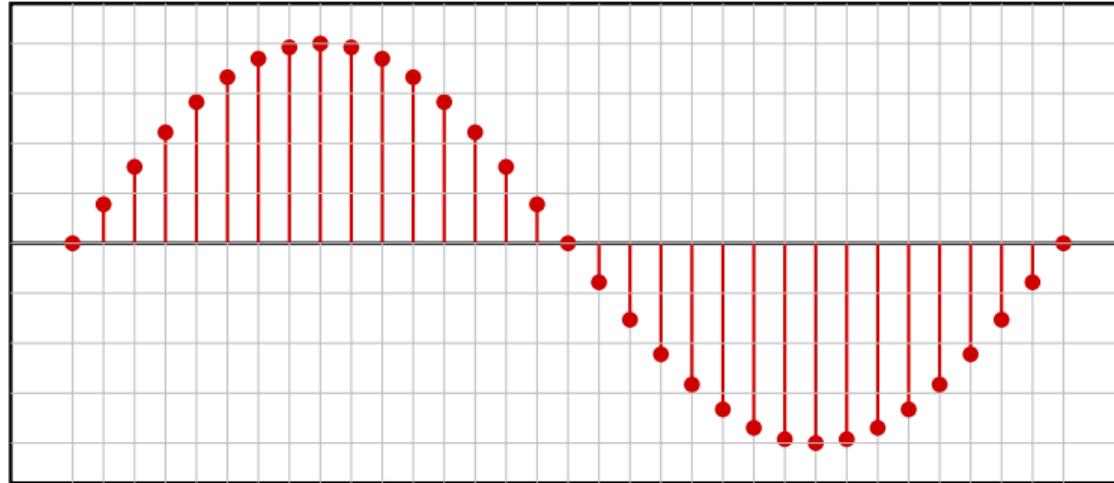
$$x[n] = \dots, 123, -73, 89, 17, -11, -26, \dots$$

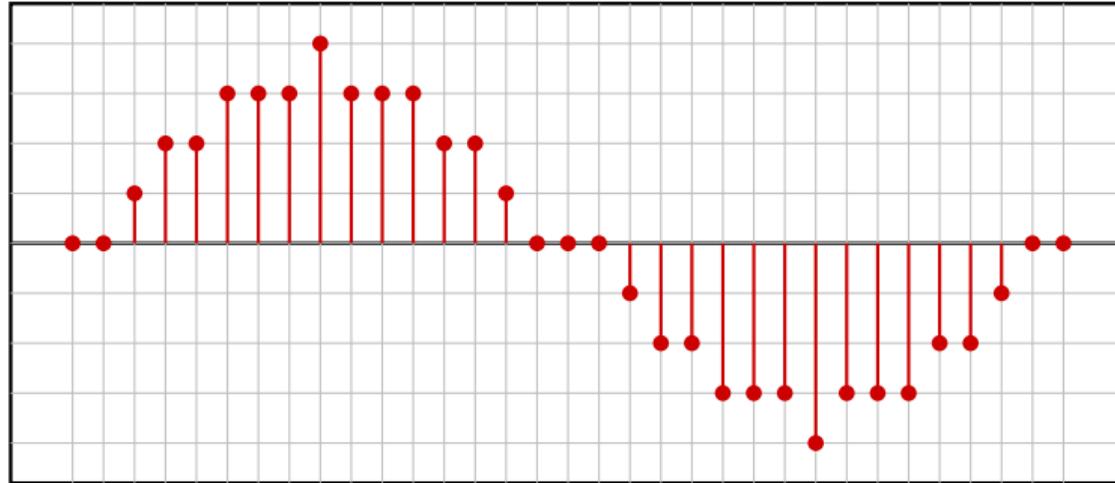


$$x(t)$$


$$x(t)$$



 $x[n]$


$$\hat{x}[n]$$

Why it is important:

- ▶ storage
- ▶ processing
- ▶ transmission

Analog storage:

paper, wax cylinders, reel-to-reel, vinyl, compact cassette, VHS, Betamax, silver plates, Kodachrome, Super8, 8-Track, microfilm, ...

Digital storage:

$\{0, 1\}$

Analog storage:

paper, wax cylinders, reel-to-reel, vinyl, compact cassette, VHS, Betamax, silver plates,
Kodachrome, Super8, 8-Track, microfilm, ...

Digital storage:

$\{0, 1\}$

Analog storage:

paper, wax cylinders, reel-to-reel, vinyl, compact cassette, VHS, Betamax, silver plates, Kodachrome, Super8, 8-Track, microfilm, ...

Digital storage:

$\{0, 1\}$

Analog storage:

paper, wax cylinders, reel-to-reel, vinyl, compact cassette, VHS, Betamax, silver plates,
Kodachrome, Super8, 8-Track, microfilm, ...

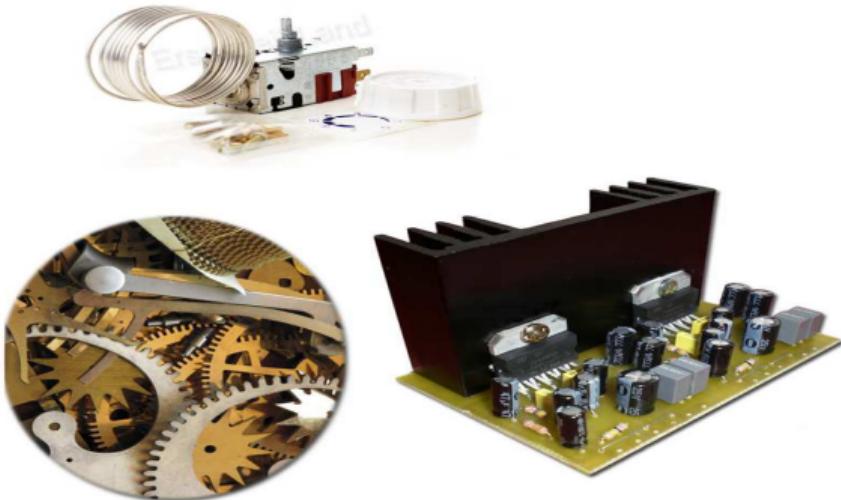
Digital storage:

$\{0, 1\}$



Just one
MicroSD card
stores more than
the rest combined...

25 years
of storage



```
extern double a[N];      // The a's coefficients
extern double b[M];      // The b's coefficients

static double x[M];      // Delay line for x
static double y[N];      // Delay line for y

double GetOutput(double input)
{
    int k;

    // Shift delay line for x:
    for (k = N-1; k > 0; k--)
        x[k] = x[k-1];

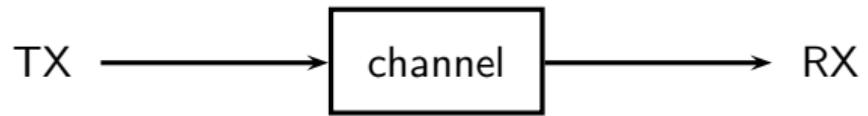
    // new input value x[n]:
    x[0] = input;

    // Shift delay line for y:
    for (k = M-1; k > 0; k--)
        y[k] = y[k-1];

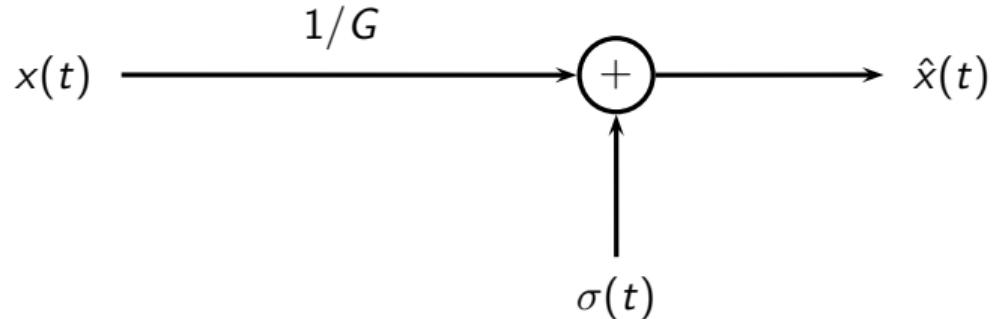
    double y = 0;
    for (k = 0; k < M; k++)
        y += b[k] * x[k];
    for (k = 1; k < N; k++)
        y -= a[k] * y[k];

    // New value for y[n]; store in delay line
    return (y[0] = y);
}
```

digital vs analog communications

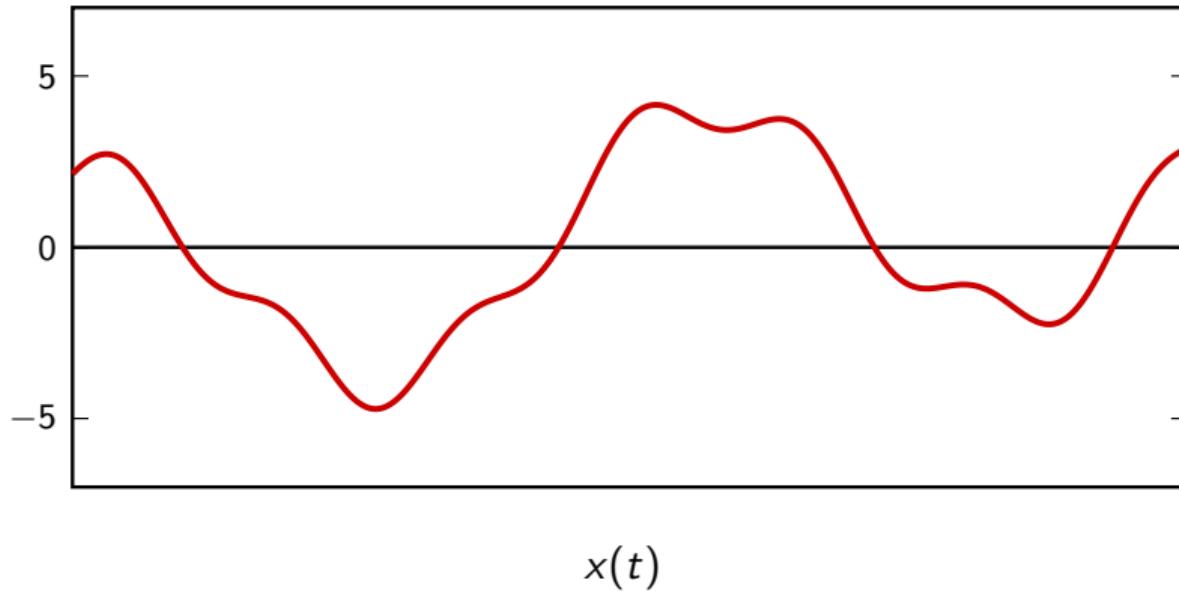


What happens to analog signals

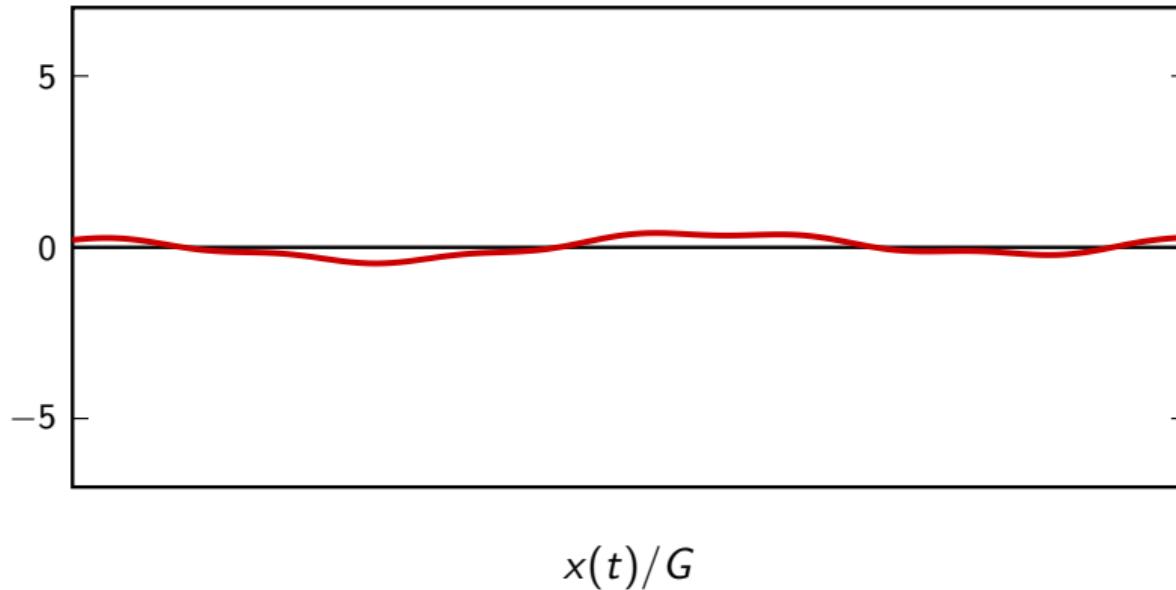


$$\hat{x}(t) = x(t)/G + \sigma(t)$$

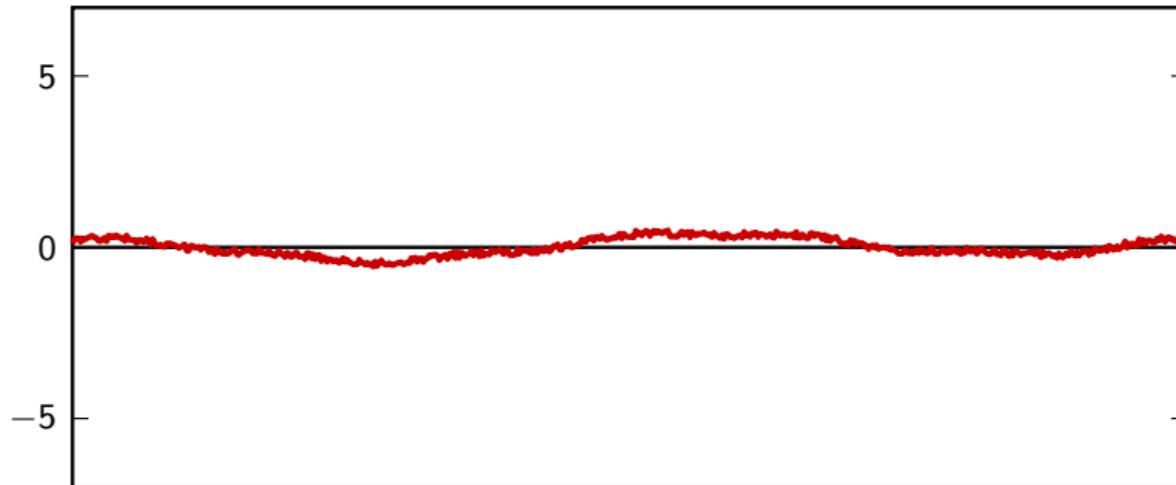
What happens to analog signals



What happens to analog signals

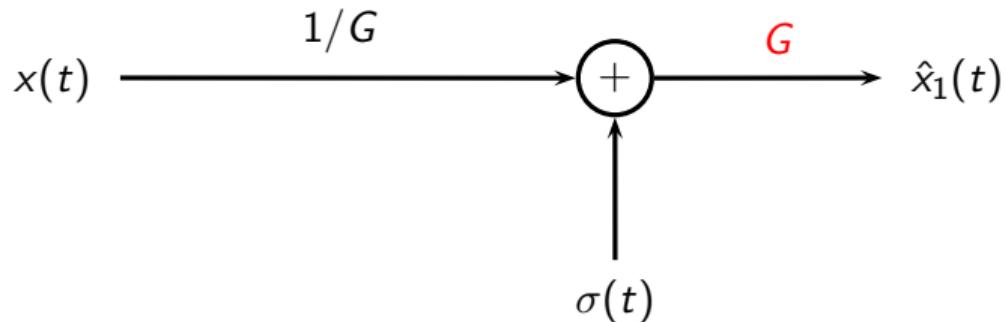


What happens to analog signals



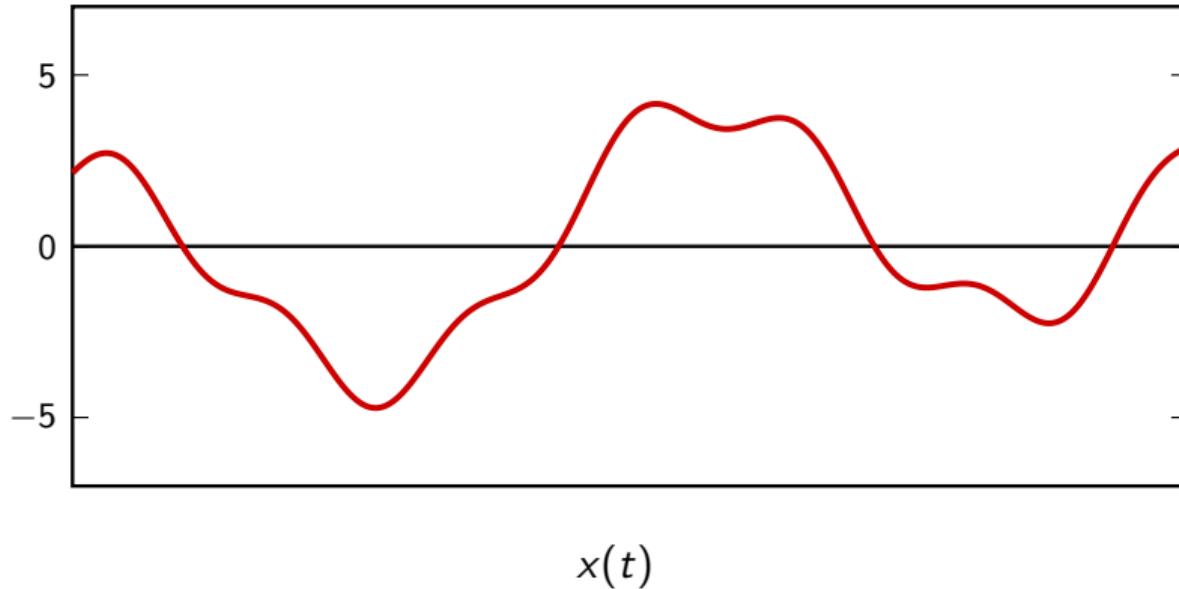
$$x(t)/G + \sigma(t)$$

We can amplify to compensate attenuation

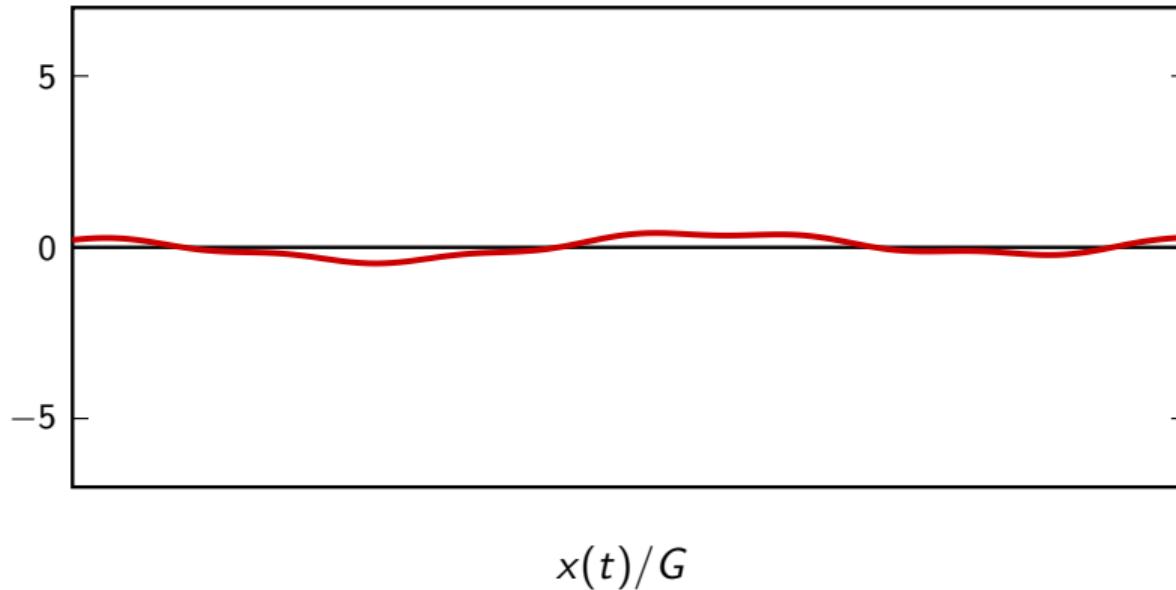


but: $\hat{x}_1(t) = x(t) + G\sigma(t)$

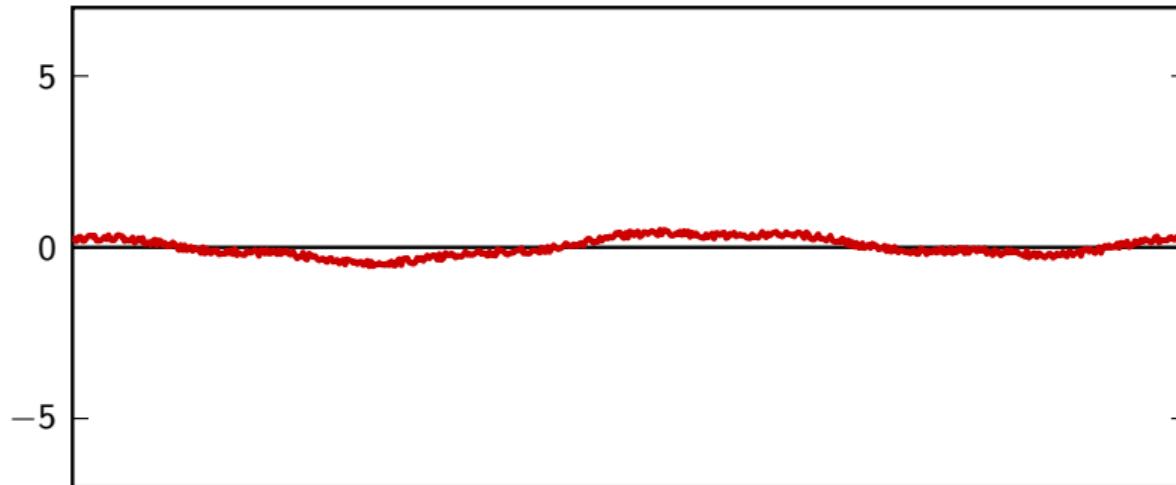
Transmission of analog signals



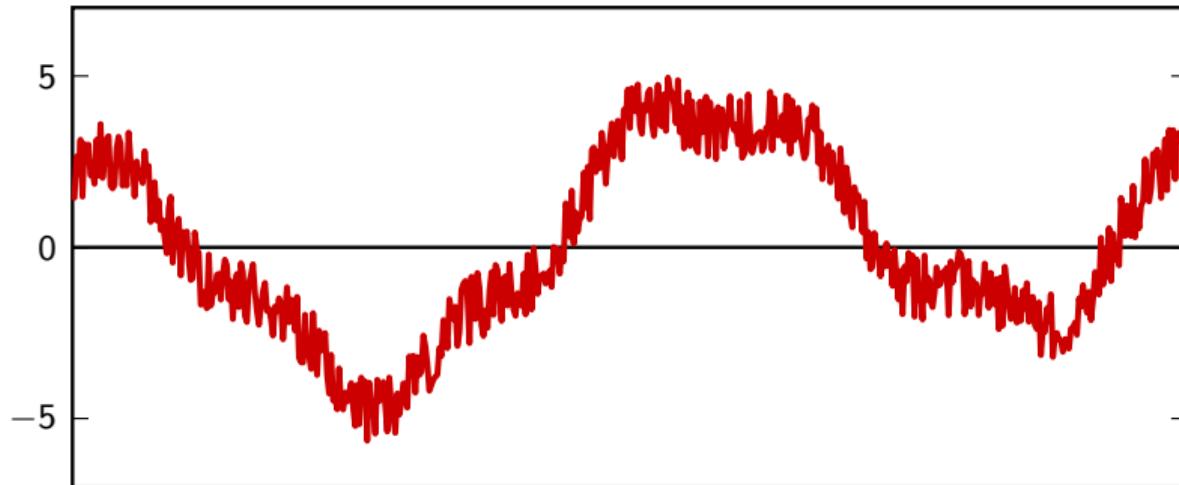
Transmission of analog signals



Transmission of analog signals



$$x(t)/G + \sigma(t)$$

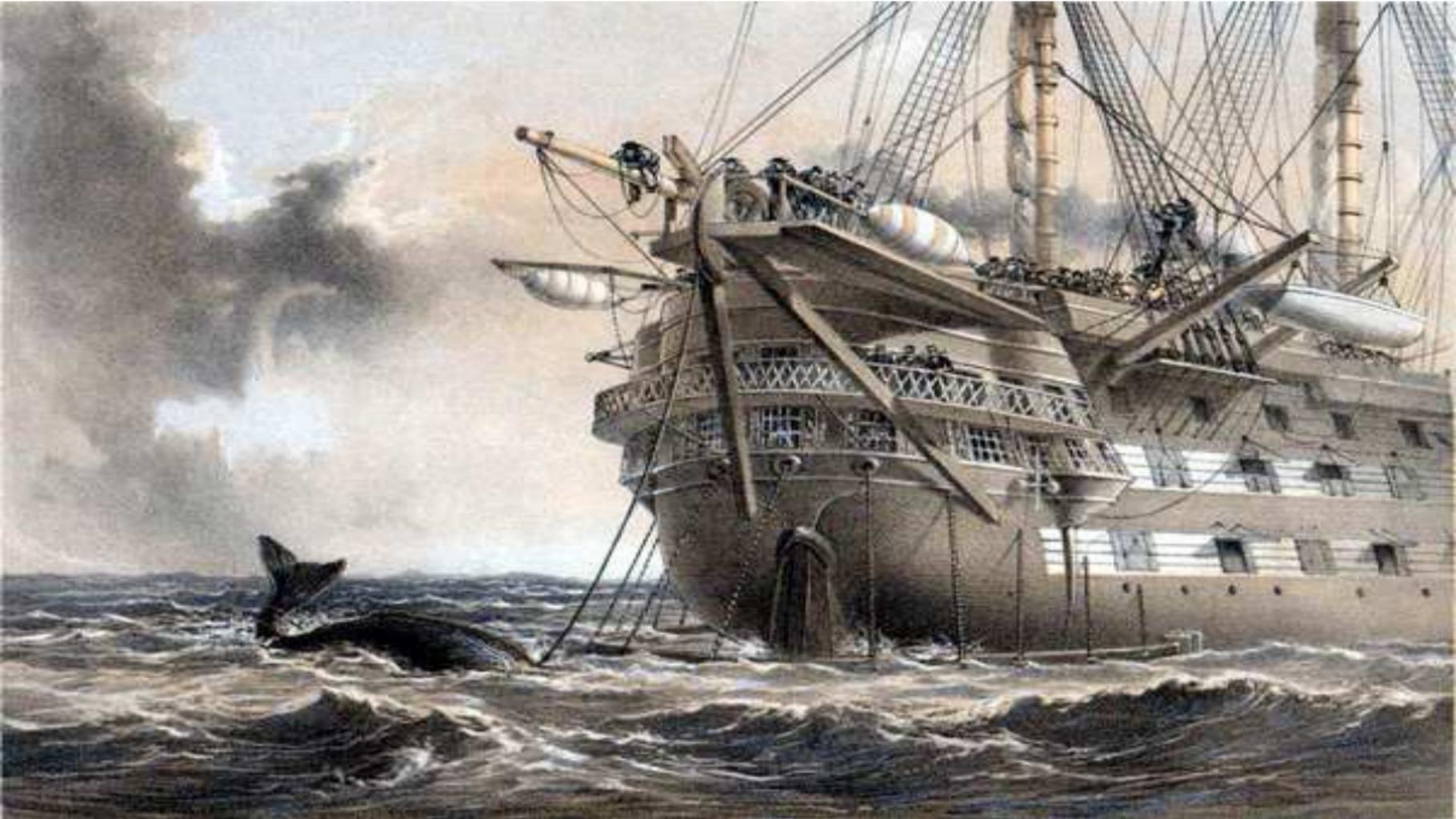


$$\hat{x}_1(t) = G[x(t)/G + \sigma(t)] = x(t) + G\sigma(t)$$

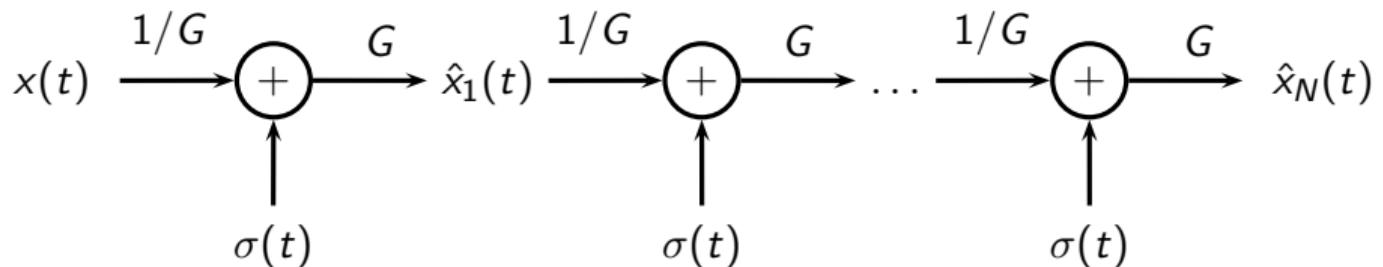
MAP
OF THE
SUBMARINE TELEGRAPH

BETWEEN AMERICA & EUROPE.

WITH ITS BRANCHES, ESTABLISHED
IN 1858 AND 1866.

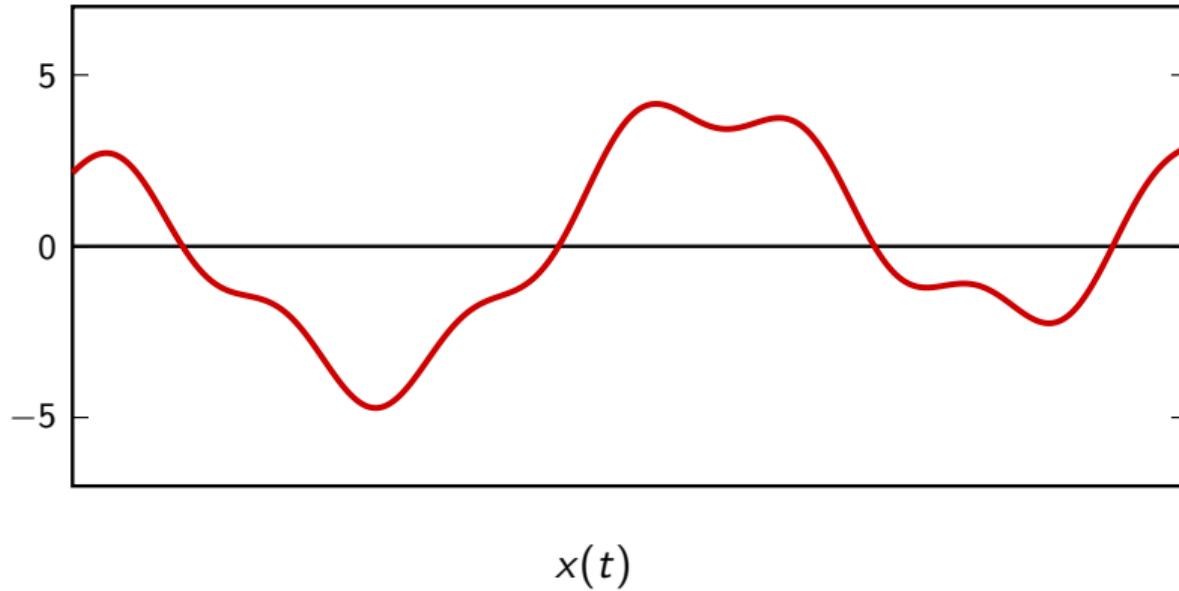


For a long, long channel we need repeaters

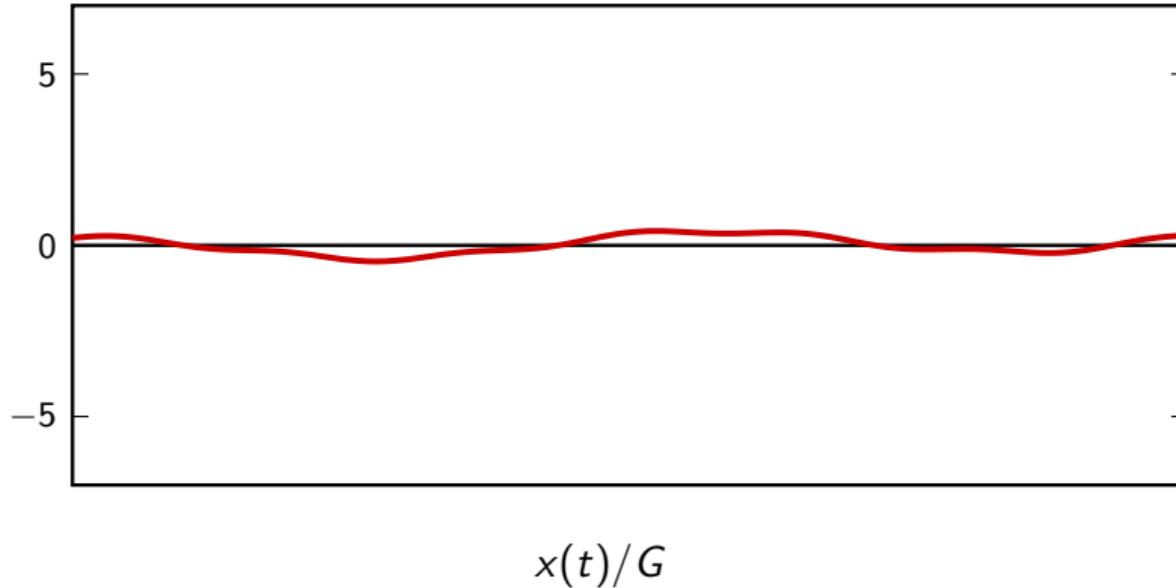


$$\hat{x}_N(t) = x(t) + NG\sigma(t)$$

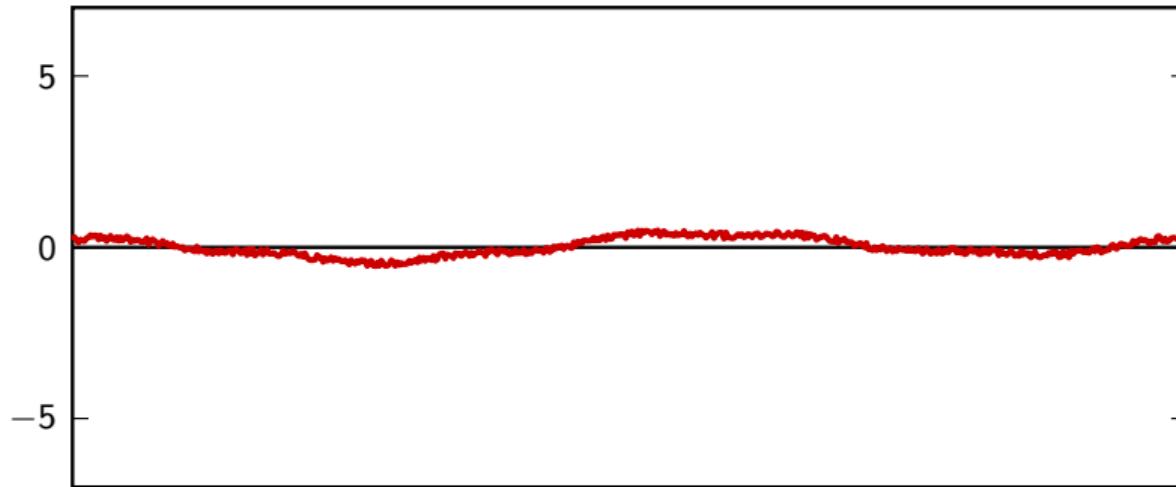
Transmission of analog signals



Transmission of analog signals

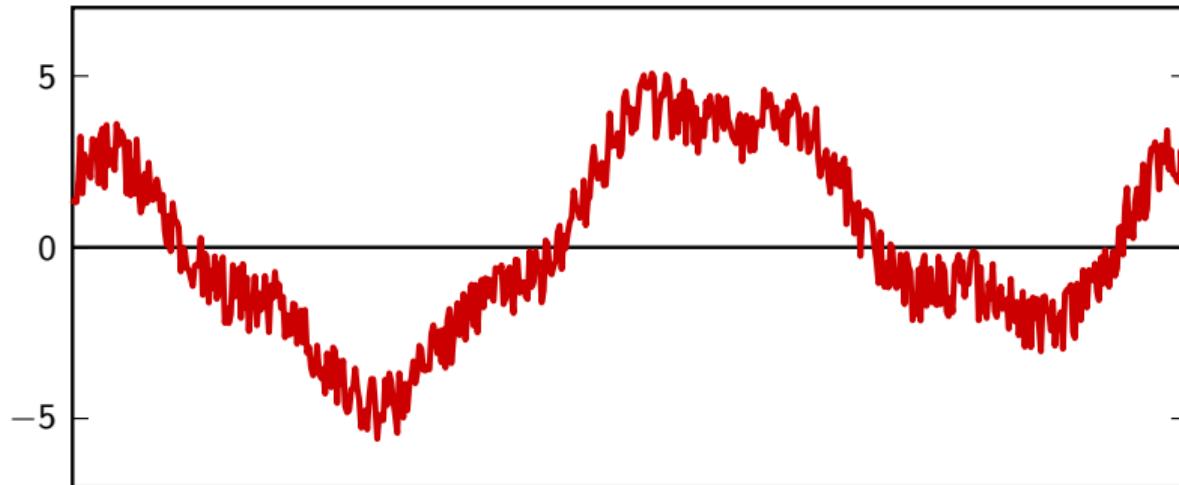


Transmission of analog signals



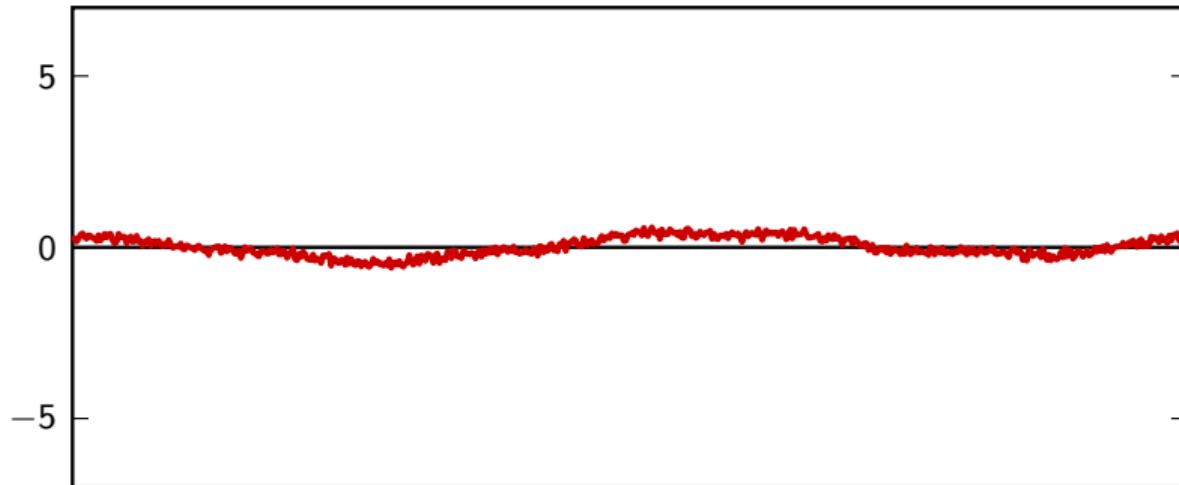
$$x(t)/G + \sigma(t)$$

Transmission of analog signals



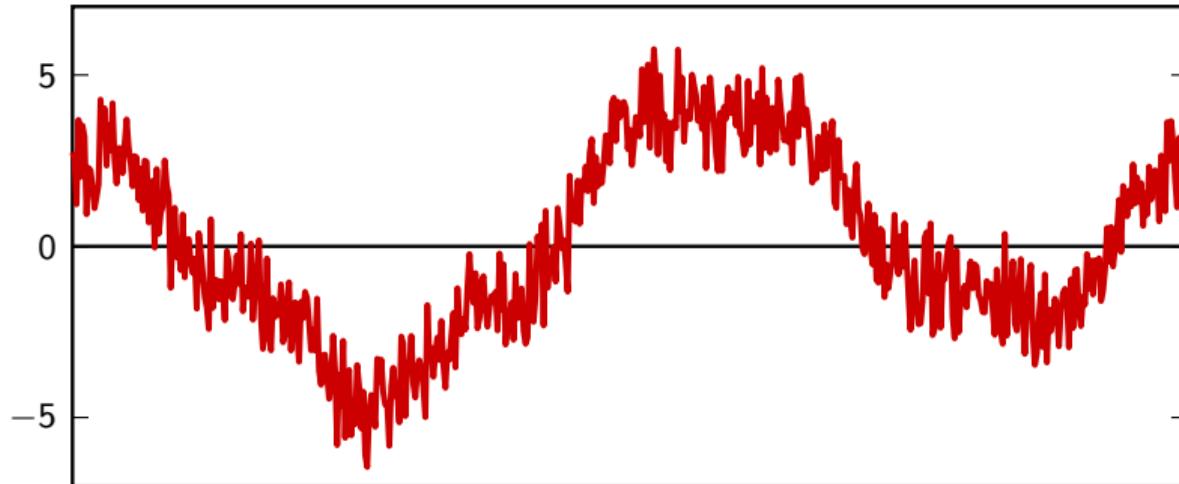
$$\hat{x}_1(t) = G[x(t)/G + \sigma(t)] = x(t) + G\sigma(t)$$

Transmission of analog signals



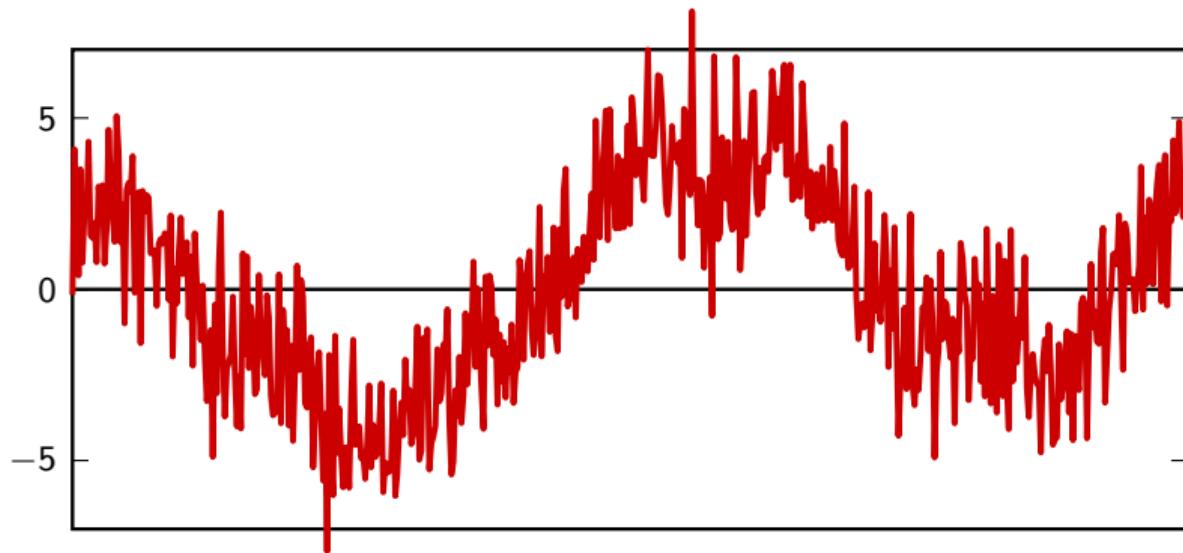
$$\hat{x}_1(t)/G + \sigma(t)$$

Transmission of analog signals



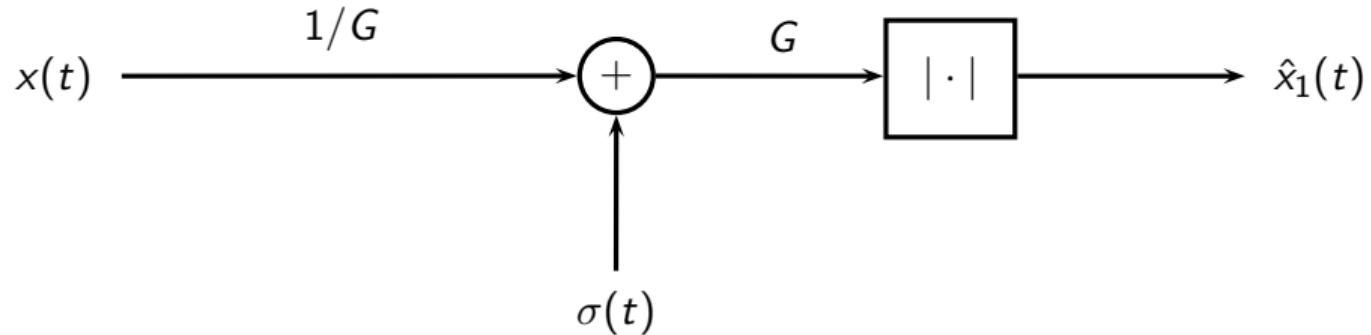
$$\hat{x}_2(t) = G[\hat{x}_1(t)/G + \sigma(t)] = x(t) + 2G\sigma(t)$$

Transmission of analog signals



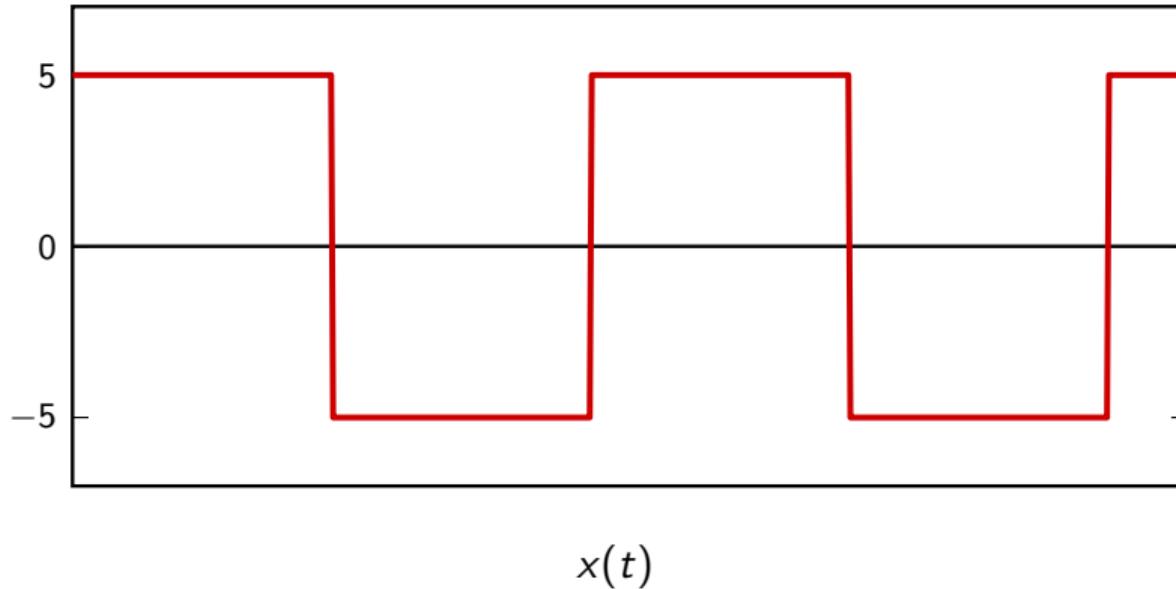
$$\hat{x}_N(t) = x(t) + NG\sigma(t)$$

In digital signals we can threshold

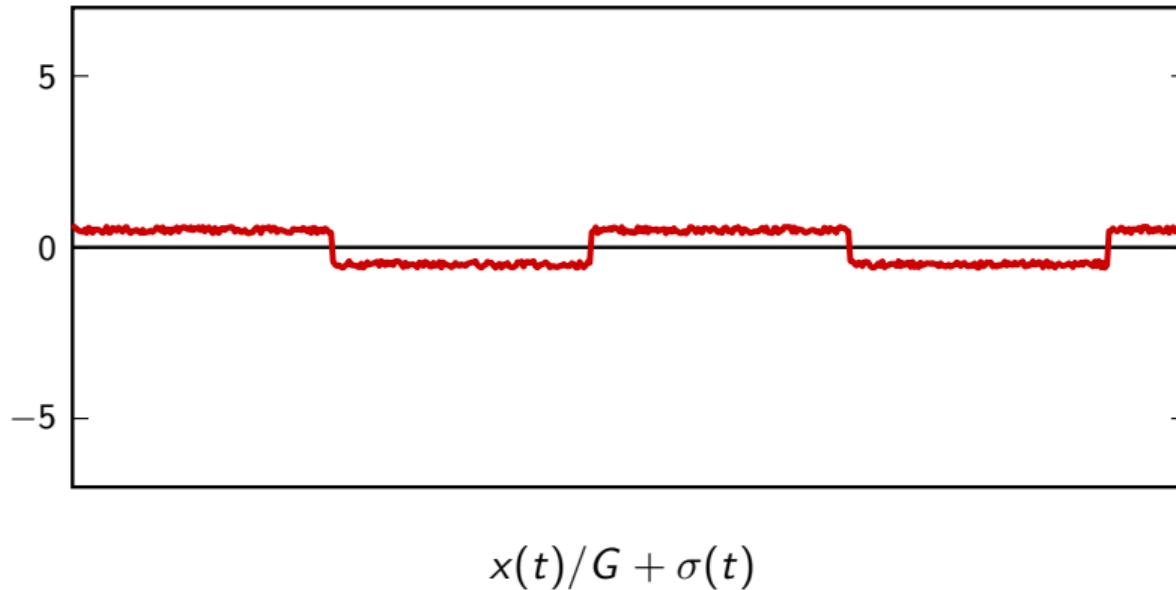


$$\hat{x}_1(t) = \text{sgn}[x(t) + G\sigma(t)]$$

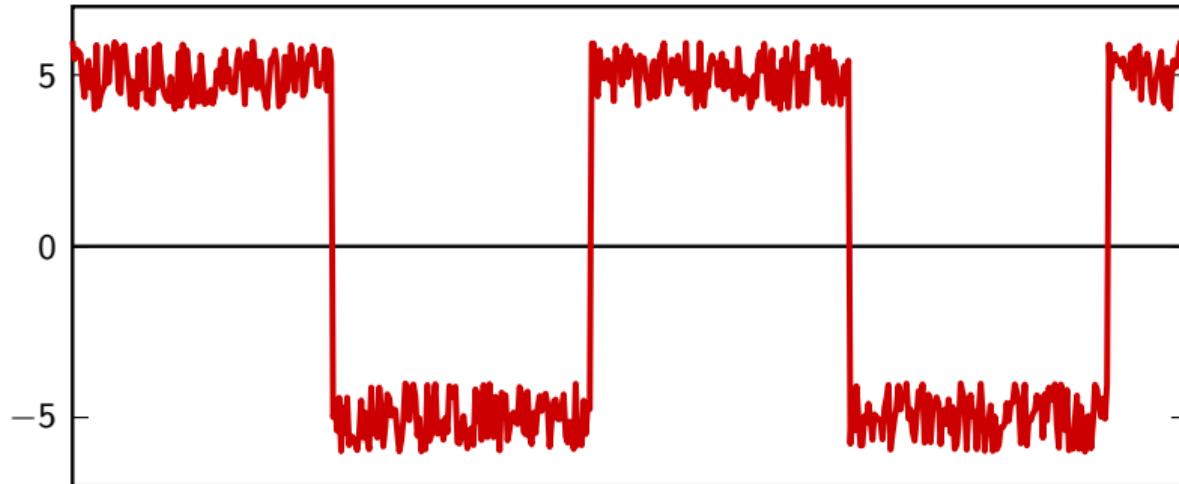
Transmission of quantized signals



Transmission of quantized signals

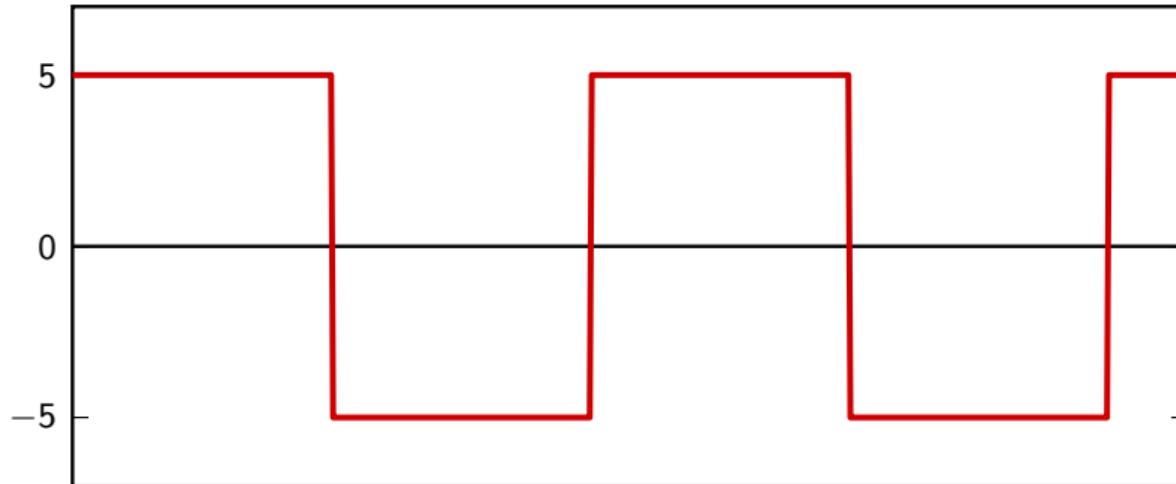


Transmission of quantized signals



$$G[x(t)/G + \sigma(t)] = x(t) + G\sigma(t)$$

Transmission of quantized signals



$$\hat{x}_1(t) = G \operatorname{sgn}[x(t) + G\sigma(t)]$$

► Transatlantic cable:

- 1866: 8 words per minute (≈ 5 bps)
- 1956: AT&T, coax, 48 voice channels (≈ 3 Mbps)
- 2005: Alcatel Tera10, fiber, 8.4 Tbps (8.4×10^{12} bps)
- 2012: fiber, 60 Tbps

► Voiceband modems

- 1950s: Bell 202, 1200 bps
- 1990s: V90, 56Kbps
- 2008: ADSL2+, 24Mbps

► Transatlantic cable:

- 1866: 8 words per minute (≈ 5 bps)
- 1956: AT&T, coax, 48 voice channels (≈ 3 Mbps)
- 2005: Alcatel Tera10, fiber, 8.4 Tbps (8.4×10^{12} bps)
- 2012: fiber, 60 Tbps

► Voiceband modems

- 1950s: Bell 202, 1200 bps
- 1990s: V90, 56Kbps
- 2008: ADSL2+, 24Mbps

numerical example

COM303: Digital Signal Processing

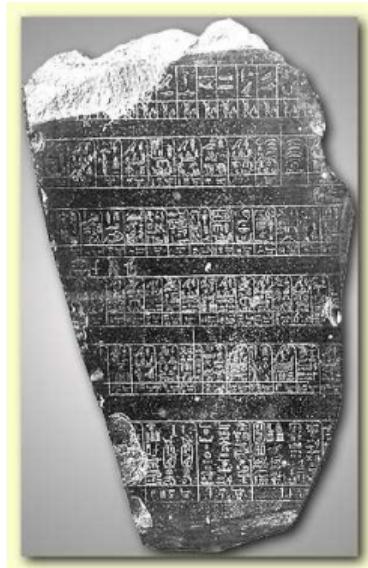
Lecture 2: Discrete-Time Signals

Module Overview:

- ▶ discrete-time signals
- ▶ elementary signal operations
- ▶ the Karplus-Strong algorithm

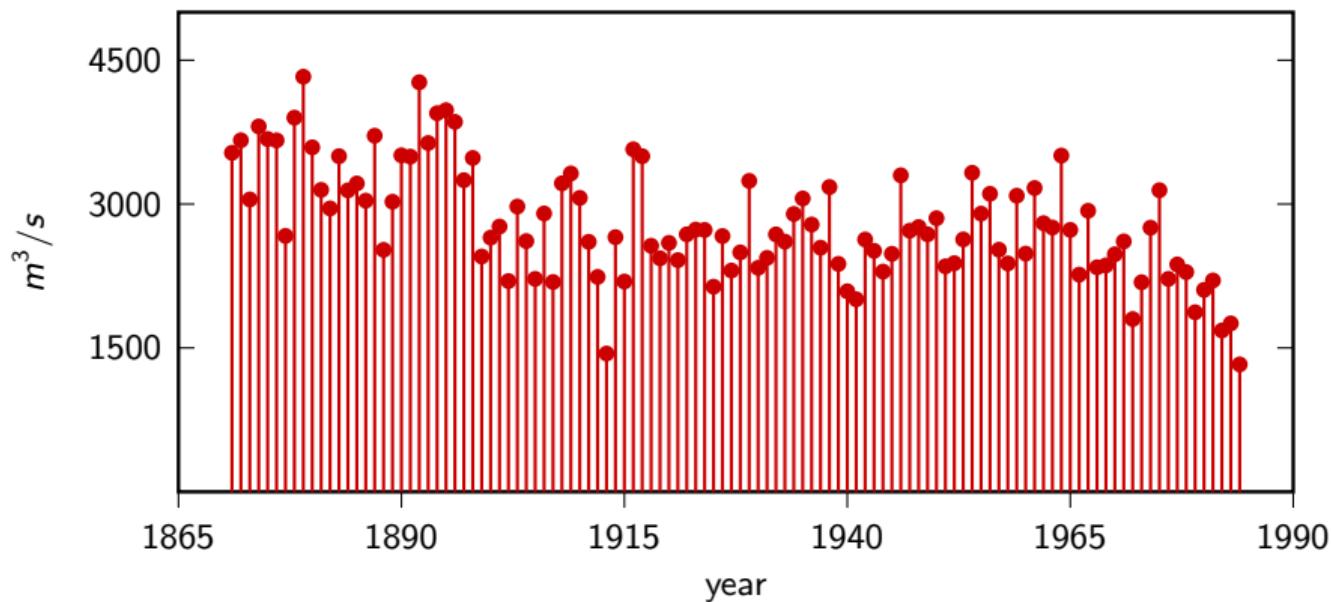
Discrete-time signals have a long tradition...

Meteorology (limnology): the floods of the Nile



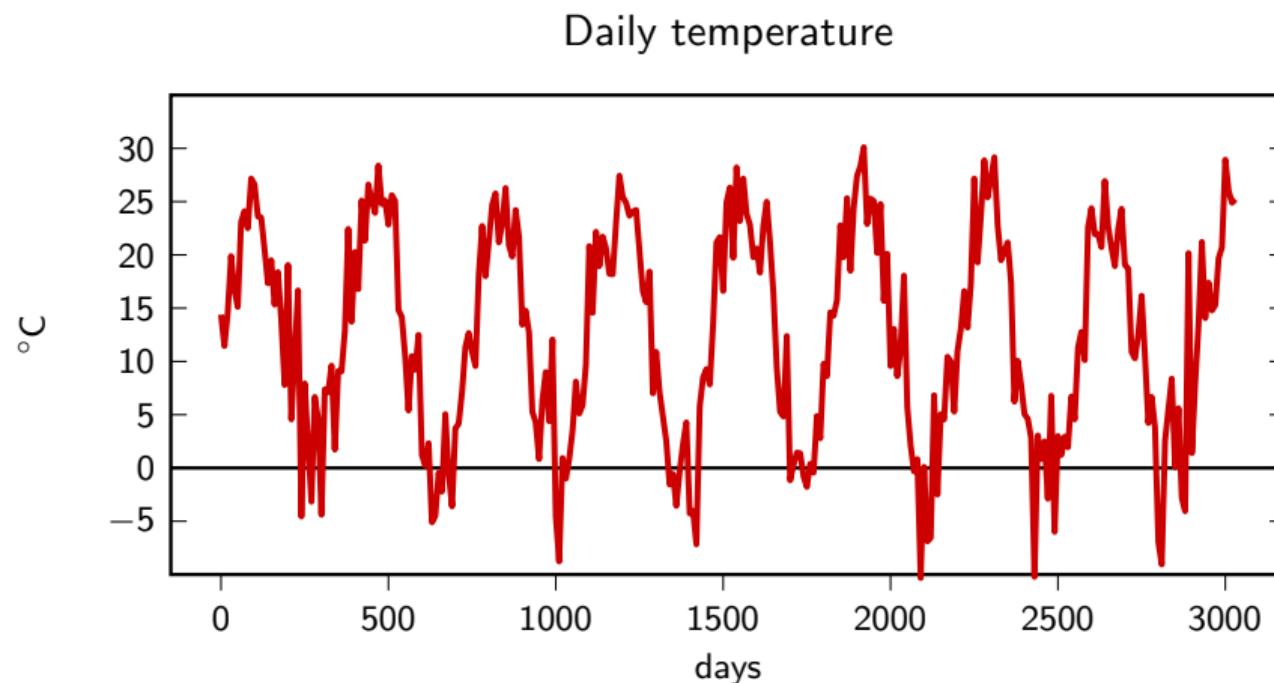
Representations of flood data: circa 2500 BC

Discrete-time signals have a long tradition...

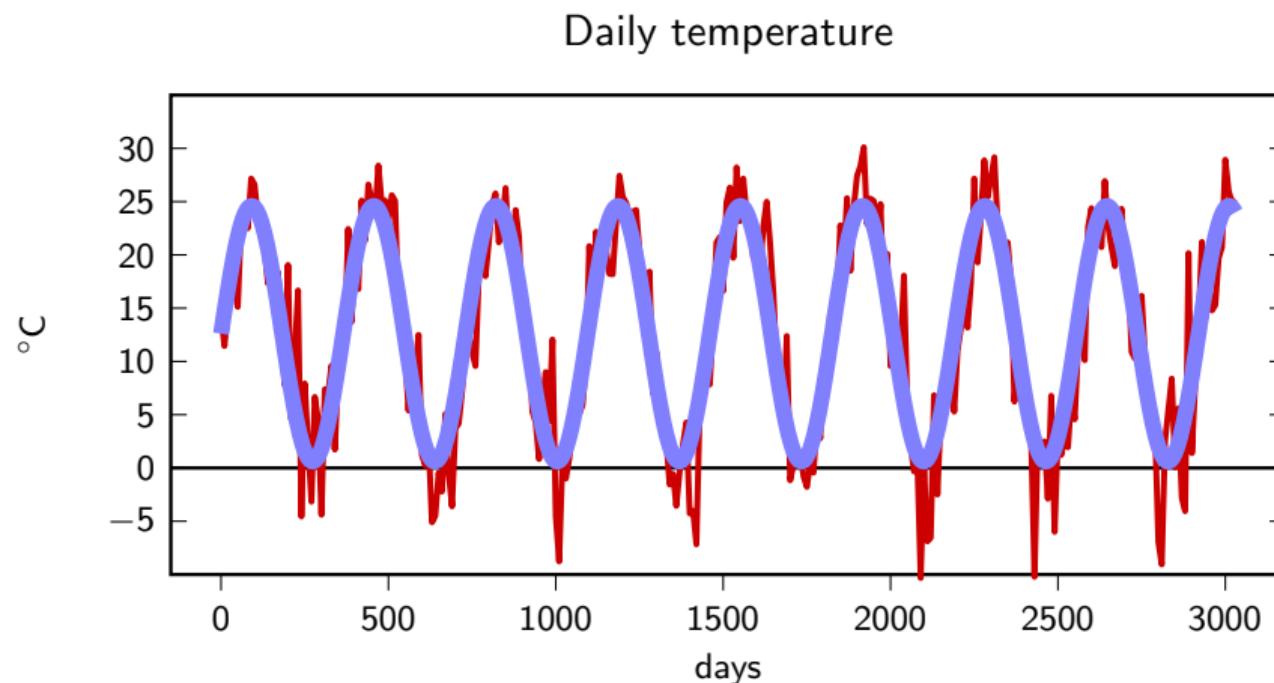


Representations of flood data: circa AD 2000

Probably your first scientific experiment...

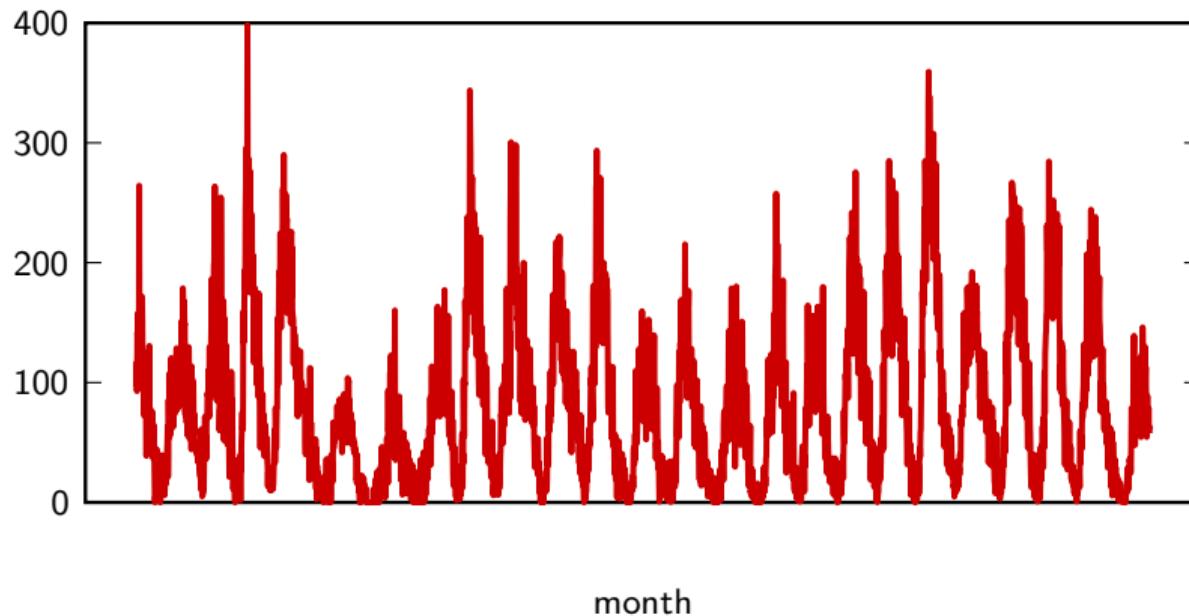


Probably your first scientific experiment...

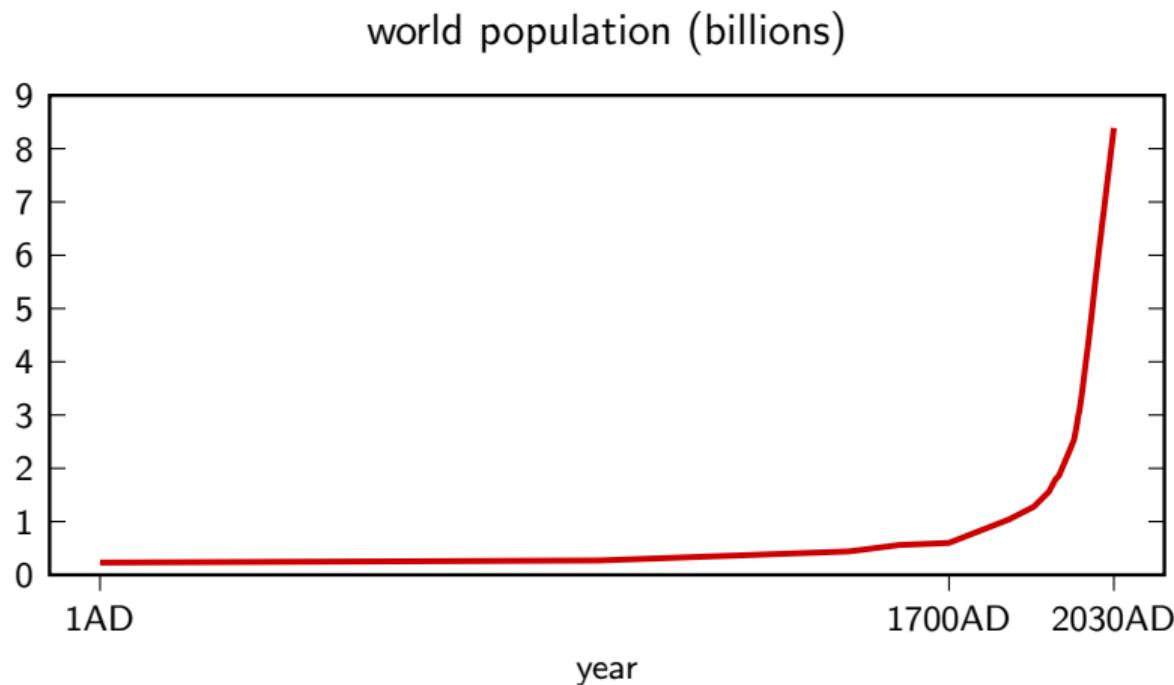


Astronomy

monthly solar spot activity, 1749 to 2015

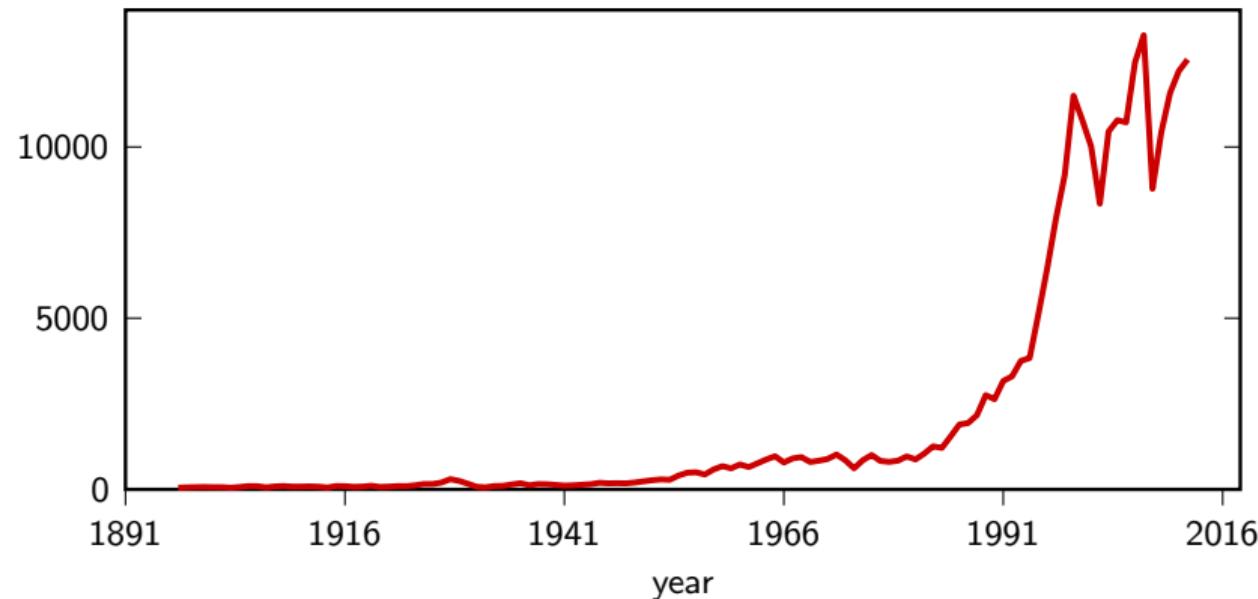


History and sociology



Economics

a purely man-made signal: the Dow Jones industrial average



More formally...

discrete-time signal: a sequence of **complex** numbers

- ▶ one dimension (for now)
- ▶ notation: $x[n]$
- ▶ two-sided sequences: $x : \mathbb{Z} \rightarrow \mathbb{C}$
- ▶ n is *a-dimensional "time"*
- ▶ analysis: periodic measurement
- ▶ synthesis: stream of generated samples

More formally...

discrete-time signal: a sequence of **complex** numbers

- ▶ one dimension (for now)
- ▶ notation: $x[n]$
- ▶ two-sided sequences: $x : \mathbb{Z} \rightarrow \mathbb{C}$
- ▶ n is *a-dimensional* “time”
- ▶ analysis: periodic measurement
- ▶ synthesis: stream of generated samples

More formally...

discrete-time signal: a sequence of **complex** numbers

- ▶ one dimension (for now)
- ▶ notation: $x[n]$
- ▶ two-sided sequences: $x : \mathbb{Z} \rightarrow \mathbb{C}$
- ▶ n is *a-dimensional* “time”
- ▶ analysis: periodic measurement
- ▶ synthesis: stream of generated samples

More formally...

discrete-time signal: a sequence of **complex** numbers

- ▶ one dimension (for now)
- ▶ notation: $x[n]$
- ▶ two-sided sequences: $x : \mathbb{Z} \rightarrow \mathbb{C}$
- ▶ n is *a-dimensional “time”*
- ▶ analysis: periodic measurement
- ▶ synthesis: stream of generated samples

More formally...

discrete-time signal: a sequence of **complex** numbers

- ▶ one dimension (for now)
- ▶ notation: $x[n]$
- ▶ two-sided sequences: $x : \mathbb{Z} \rightarrow \mathbb{C}$
- ▶ n is *a-dimensional* “time”
- ▶ analysis: periodic measurement
- ▶ synthesis: stream of generated samples

More formally...

discrete-time signal: a sequence of **complex** numbers

- ▶ one dimension (for now)
- ▶ notation: $x[n]$
- ▶ two-sided sequences: $x : \mathbb{Z} \rightarrow \mathbb{C}$
- ▶ n is *a-dimensional* “time”
- ▶ analysis: periodic measurement
- ▶ synthesis: stream of generated samples

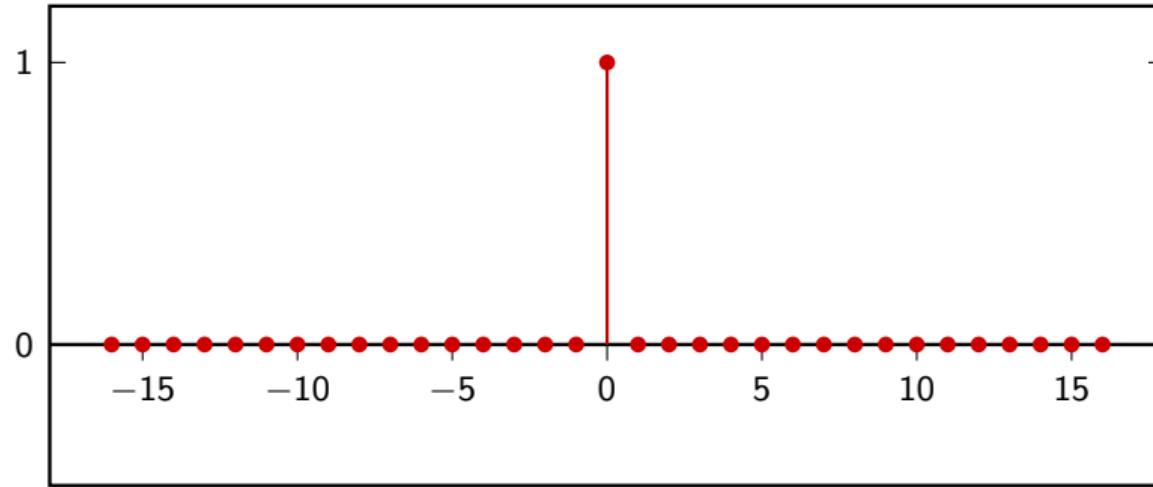
More formally...

discrete-time signal: a sequence of **complex** numbers

- ▶ one dimension (for now)
- ▶ notation: $x[n]$
- ▶ two-sided sequences: $x : \mathbb{Z} \rightarrow \mathbb{C}$
- ▶ n is *a-dimensional* “time”
- ▶ analysis: periodic measurement
- ▶ synthesis: stream of generated samples

The delta signal

$$x[n] = \delta[n]$$



How do you synchronize audio and video...

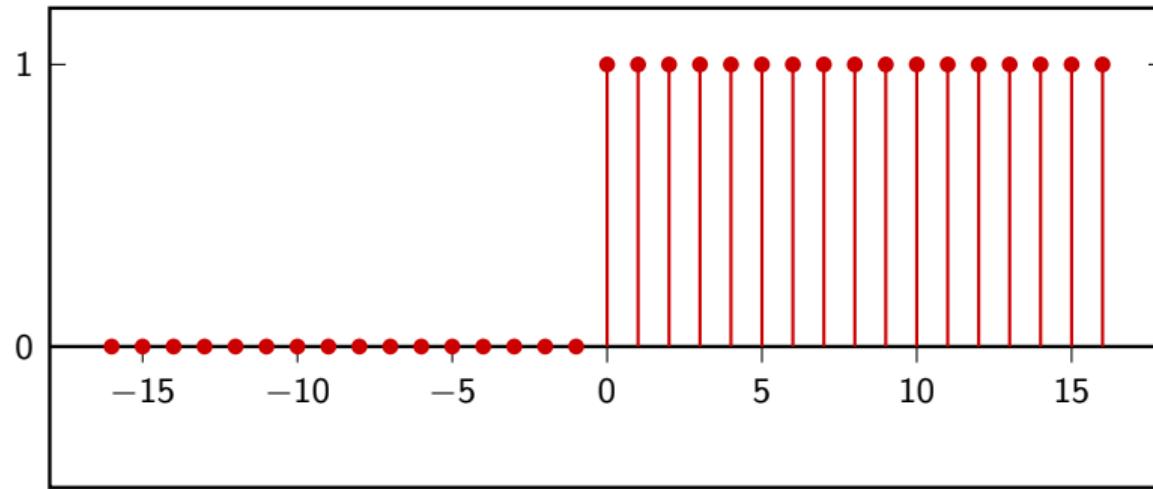


How do you synchronize audio and video...



The unit step

$$x[n] = u[n]$$

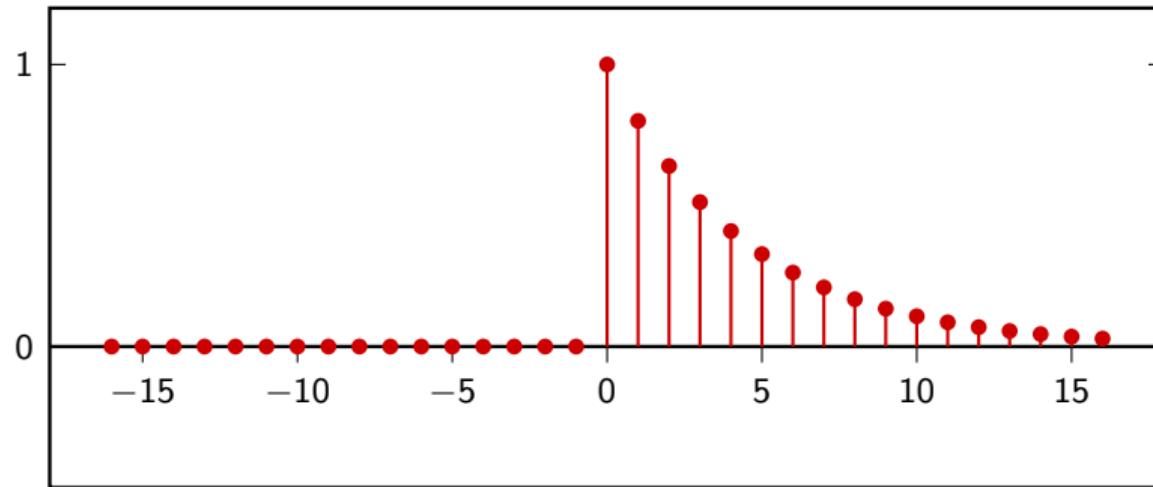


The Frankenstein switch...



The exponential decay

$$x[n] = |a|^n u[n], \quad |a| < 1$$



How fast does your coffee get cold...



How fast does your coffee get cold...

Newton's law of cooling:

$$\frac{dT}{dt} = -c(T - T_{\text{env}})$$

$$T(t) = T_{\text{env}} + (T_0 - T_{\text{env}})e^{-ct}$$

In practice:

- ▶ must have convection only
- ▶ must have large conductivity

How fast does your coffee get cold...

Newton's law of cooling:

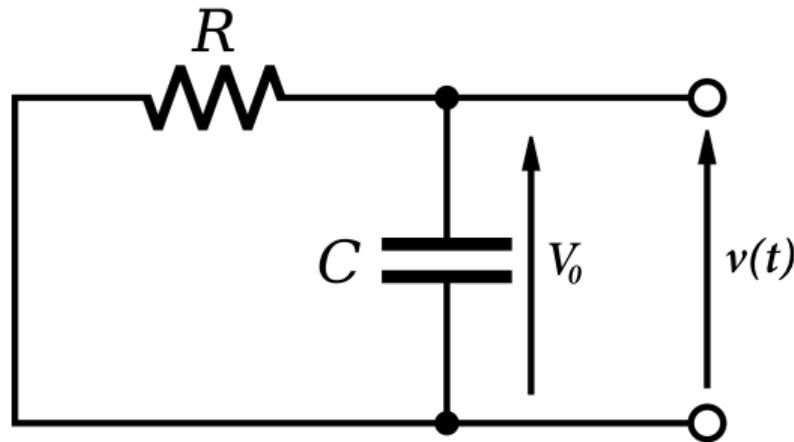
$$\frac{dT}{dt} = -c(T - T_{\text{env}})$$

$$T(t) = T_{\text{env}} + (T_0 - T_{\text{env}})e^{-ct}$$

In practice:

- ▶ must have convection only
- ▶ must have large conductivity

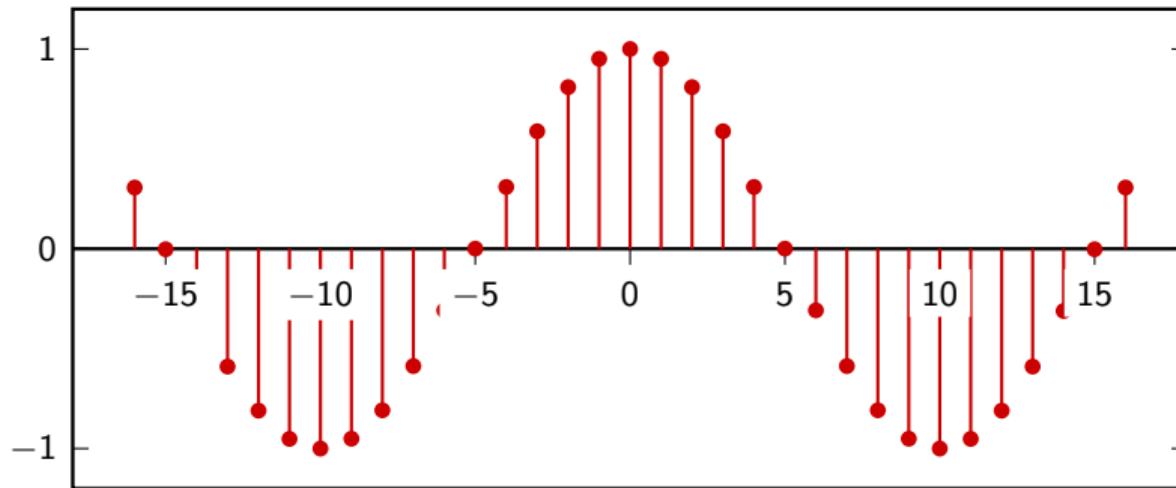
Also, how fast your capacitor discharges



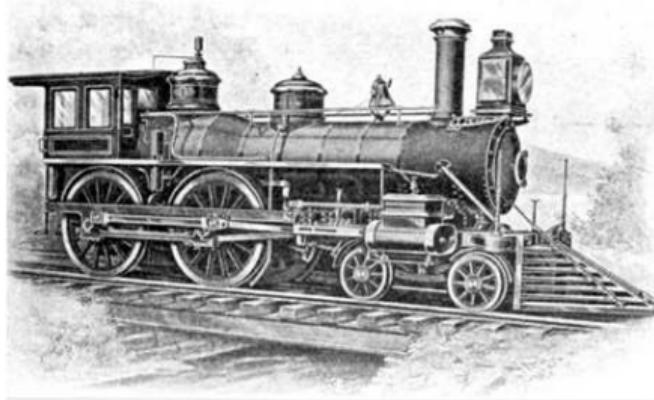
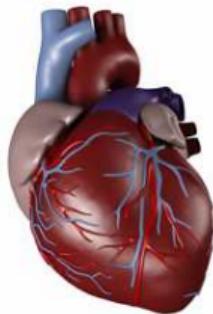
$$v(t) = V_0 e^{-\frac{t}{RC}}$$

The sinusoid

$$x[n] = \sin(\omega_0 n + \theta)$$



Oscillations are everywhere!



Four signal classes

- ▶ finite-length
- ▶ infinite-length
- ▶ periodic
- ▶ finite-support

Four signal classes

- ▶ finite-length
- ▶ infinite-length
- ▶ periodic
- ▶ finite-support

Four signal classes

- ▶ finite-length
- ▶ infinite-length
- ▶ periodic
- ▶ finite-support

Four signal classes

- ▶ finite-length
- ▶ infinite-length
- ▶ periodic
- ▶ finite-support

Finite-length signals

- ▶ sequence notation: $x[n], \quad n = 0, 1, \dots, N - 1$
- ▶ vector notation: $\mathbf{x} = [x_0 \quad x_1 \quad \dots \quad x_{N-1}]^T$
- ▶ practical entities, good for numerical packages (e.g.numpy)

Finite-length signals

- ▶ sequence notation: $x[n], \quad n = 0, 1, \dots, N - 1$
- ▶ vector notation: $\mathbf{x} = [x_0 \quad x_1 \quad \dots \quad x_{N-1}]^T$
- ▶ practical entities, good for numerical packages (e.g.numpy)

Finite-length signals

- ▶ sequence notation: $x[n], \quad n = 0, 1, \dots, N - 1$
- ▶ vector notation: $\mathbf{x} = [x_0 \quad x_1 \quad \dots \quad x_{N-1}]^T$
- ▶ practical entities, good for numerical packages (e.g.numpy)

Infinite-length signals

- ▶ sequence notation: $x[n], n \in \mathbb{Z}$
- ▶ vector notation: $\mathbf{x} = [\dots \ x_{-2} \ x_{-1} \ x_0 \ x_1 \ x_2 \ \dots]^T$
- ▶ abstraction, good for theorems

Infinite-length signals

- ▶ sequence notation: $x[n], \quad n \in \mathbb{Z}$
- ▶ vector notation: $\mathbf{x} = [\dots \quad x_{-2} \quad x_{-1} \quad x_0 \quad x_1 \quad x_2 \quad \dots]^T$
- ▶ abstraction, good for theorems

Infinite-length signals

- ▶ sequence notation: $x[n], \quad n \in \mathbb{Z}$
- ▶ vector notation: $\mathbf{x} = [\dots \quad x_{-2} \quad x_{-1} \quad x_0 \quad x_1 \quad x_2 \quad \dots]^T$
- ▶ abstraction, good for theorems

Periodic signals

- ▶ N -periodic sequence: $\tilde{x}[n] = \tilde{x}[n + kN]$, $n, k, N \in \mathbb{Z}$
- ▶ $\tilde{\mathbf{x}} = [\dots \quad x_{N-2} \quad x_{N-1} \quad x_0 \quad x_1 \quad \dots \quad x_{N-1} \quad x_0 \quad x_1 \quad \dots]^T$
- ▶ infinite length but same amount of information as length- N
- ▶ “natural” bridge between finite and infinite lengths

Periodic signals

- ▶ N -periodic sequence: $\tilde{x}[n] = \tilde{x}[n + kN]$, $n, k, N \in \mathbb{Z}$
- ▶ $\tilde{\mathbf{x}} = [\dots \ x_{N-2} \ x_{N-1} \ x_0 \ x_1 \ \dots \ x_{N-1} \ x_0 \ x_1 \ \dots]^T$
- ▶ infinite length but same amount of information as length- N
- ▶ “natural” bridge between finite and infinite lengths

Periodic signals

- ▶ N -periodic sequence: $\tilde{x}[n] = \tilde{x}[n + kN]$, $n, k, N \in \mathbb{Z}$
- ▶ $\tilde{\mathbf{x}} = [\dots \quad x_{N-2} \quad x_{N-1} \quad x_0 \quad x_1 \quad \dots \quad x_{N-1} \quad x_0 \quad x_1 \quad \dots]^T$
- ▶ infinite length but same amount of information as length- N
- ▶ “natural” bridge between finite and infinite lengths

Periodic signals

- ▶ N -periodic sequence: $\tilde{x}[n] = \tilde{x}[n + kN]$, $n, k, N \in \mathbb{Z}$
- ▶ $\tilde{\mathbf{x}} = [\dots \quad x_{N-2} \quad x_{N-1} \quad x_0 \quad x_1 \quad \dots \quad x_{N-1} \quad x_0 \quad x_1 \quad \dots]^T$
- ▶ infinite length but same amount of information as length- N
- ▶ “natural” bridge between finite and infinite lengths

Finite-support signals

- ▶ Finite-support sequence:

$$\bar{x}[n] = \begin{cases} x[n] & \text{if } 0 \leq n < N \\ 0 & \text{otherwise} \end{cases} \quad n \in \mathbb{Z}$$

- ▶ $\bar{\mathbf{x}} = [\dots \ 0 \ 0 \ x_0 \ x_1 \ \dots \ x_{N-1} \ 0 \ 0 \ \dots]^T$
- ▶ infinite length but same amount of information as length- N
- ▶ same information as finite-length of length N
- ▶ another bridge between finite and infinite lengths

Finite-support signals

- ▶ Finite-support sequence:

$$\bar{x}[n] = \begin{cases} x[n] & \text{if } 0 \leq n < N \\ 0 & \text{otherwise} \end{cases} \quad n \in \mathbb{Z}$$

- ▶ $\bar{\mathbf{x}} = [\dots \ 0 \ 0 \ x_0 \ x_1 \ \dots \ x_{N-1} \ 0 \ 0 \ \dots]^T$

- ▶ infinite length but same amount of information as length- N
- ▶ same information as finite-length of length N
- ▶ another bridge between finite and infinite lengths

Finite-support signals

- ▶ Finite-support sequence:

$$\bar{x}[n] = \begin{cases} x[n] & \text{if } 0 \leq n < N \\ 0 & \text{otherwise} \end{cases} \quad n \in \mathbb{Z}$$

- ▶ $\bar{\mathbf{x}} = [\dots \ 0 \ 0 \ x_0 \ x_1 \ \dots \ x_{N-1} \ 0 \ 0 \ \dots]^T$
- ▶ infinite length but same amount of information as length- N
- ▶ same information as finite-length of length N
- ▶ another bridge between finite and infinite lengths

Finite-support signals

- ▶ Finite-support sequence:

$$\bar{x}[n] = \begin{cases} x[n] & \text{if } 0 \leq n < N \\ 0 & \text{otherwise} \end{cases} \quad n \in \mathbb{Z}$$

- ▶ $\bar{\mathbf{x}} = [\dots \ 0 \ 0 \ x_0 \ x_1 \ \dots \ x_{N-1} \ 0 \ 0 \ \dots]^T$
- ▶ infinite length but same amount of information as length- N
- ▶ same information as finite-length of length N
- ▶ another bridge between finite and infinite lengths

Finite-support signals

- ▶ Finite-support sequence:

$$\bar{x}[n] = \begin{cases} x[n] & \text{if } 0 \leq n < N \\ 0 & \text{otherwise} \end{cases} \quad n \in \mathbb{Z}$$

- ▶ $\bar{\mathbf{x}} = [\dots \ 0 \ 0 \ x_0 \ x_1 \ \dots \ x_{N-1} \ 0 \ 0 \ \dots]^T$
- ▶ infinite length but same amount of information as length- N
- ▶ same information as finite-length of length N
- ▶ another bridge between finite and infinite lengths

Elementary operators

- ▶ scaling:

$$y[n] = \alpha x[n]$$

- ▶ sum:

$$y[n] = x[n] + z[n]$$

- ▶ product:

$$y[n] = x[n] \cdot z[n]$$

- ▶ shift by k (delay):

$$y[n] = x[n - k]$$

Elementary operators

- ▶ scaling:

$$y[n] = \alpha x[n]$$

- ▶ sum:

$$y[n] = x[n] + z[n]$$

- ▶ product:

$$y[n] = x[n] \cdot z[n]$$

- ▶ shift by k (delay):

$$y[n] = x[n - k]$$

Elementary operators

- ▶ scaling:

$$y[n] = \alpha x[n]$$

- ▶ sum:

$$y[n] = x[n] + z[n]$$

- ▶ product:

$$y[n] = x[n] \cdot z[n]$$

- ▶ shift by k (delay):

$$y[n] = x[n - k]$$

Elementary operators

- ▶ scaling:

$$y[n] = \alpha x[n]$$

- ▶ sum:

$$y[n] = x[n] + z[n]$$

- ▶ product:

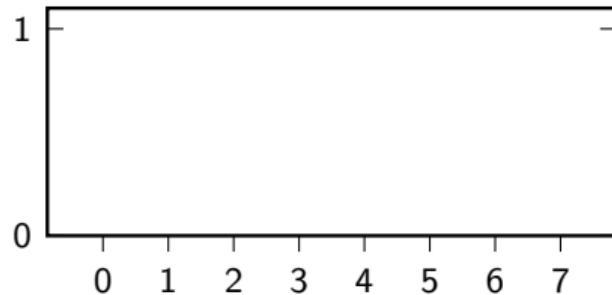
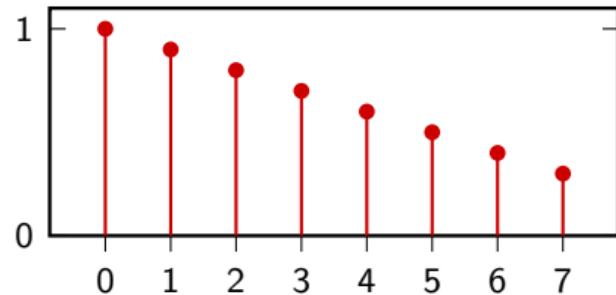
$$y[n] = x[n] \cdot z[n]$$

- ▶ shift by k (delay):

$$y[n] = x[n - k]$$

Shift of a finite-length: finite-support

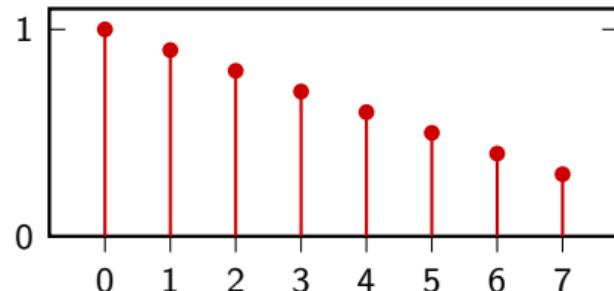
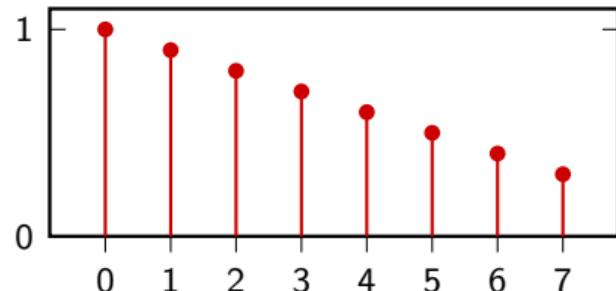
$$[x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7]$$



Shift of a finite-length: finite-support

$x[n]$

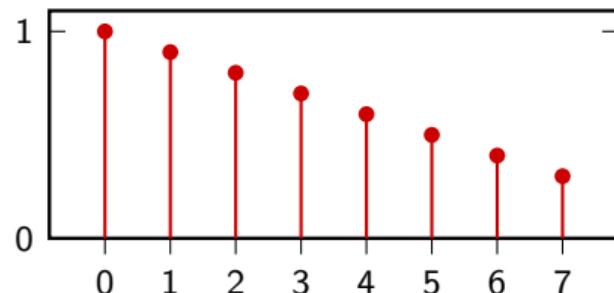
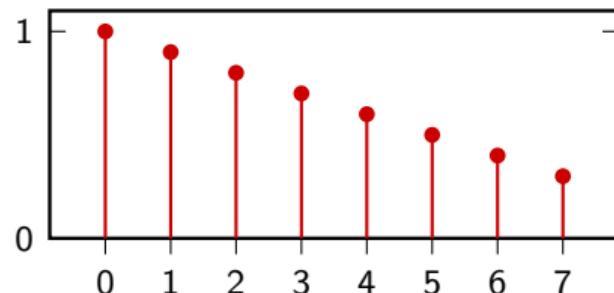
$\dots \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad \dots$



Shift of a finite-length: finite-support

$\bar{x}[n]$

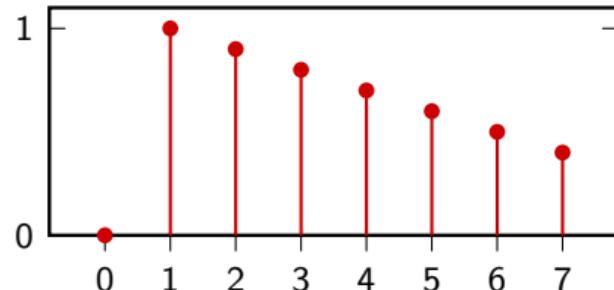
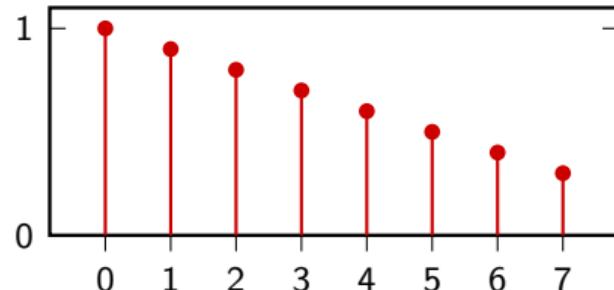
$\dots \ 0 \ 0 \ 0 \ x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ 0 \ 0 \ 0 \ \dots$



Shift of a finite-length: finite-support

$$\bar{x}[n - 1]$$

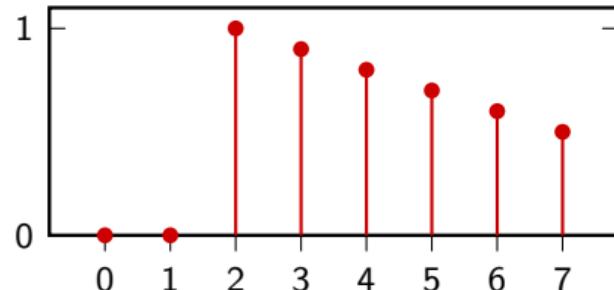
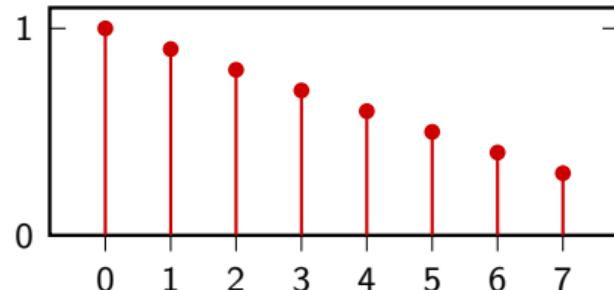
$\dots \ 0 \ 0 \ 0 \ 0 \ x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ 0 \ 0 \ \dots$



Shift of a finite-length: finite-support

$$\bar{x}[n - 2]$$

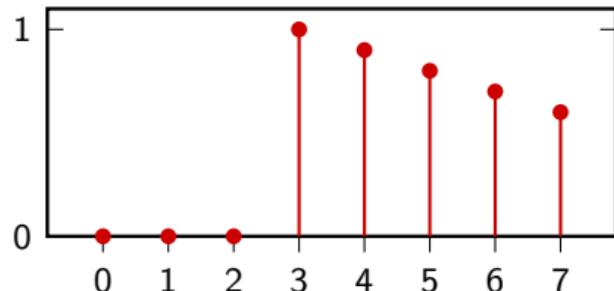
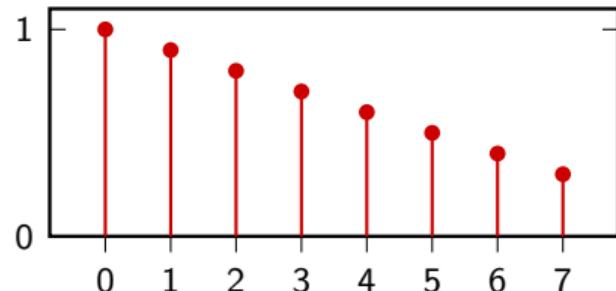
$\dots \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ 0 \ \dots$



Shift of a finite-length: finite-support

$$\bar{x}[n - 3]$$

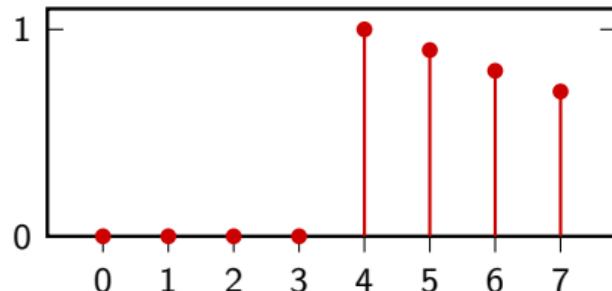
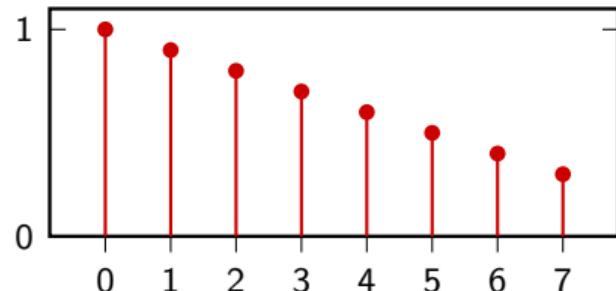
$\dots \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ \dots$



Shift of a finite-length: finite-support

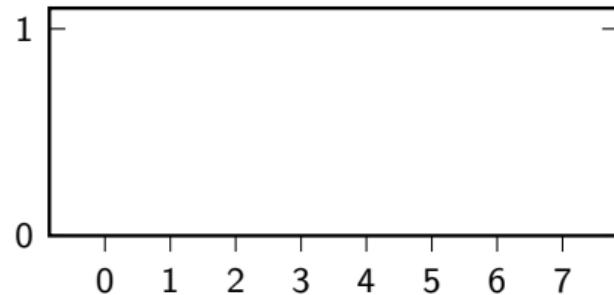
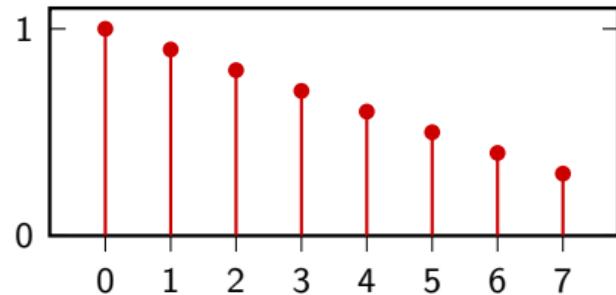
$$\bar{x}[n - 4]$$

$\dots \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ \dots$



Shift of a finite-length: periodic extension

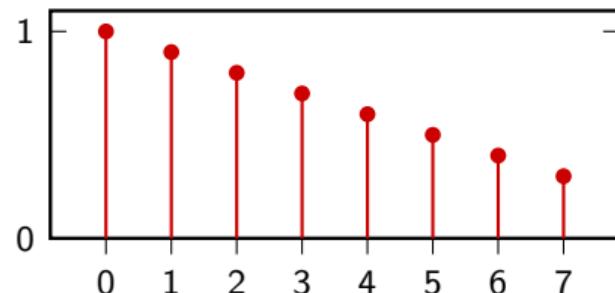
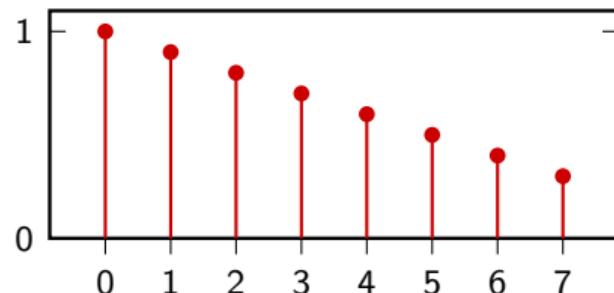
$$[x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7]$$



Shift of a finite-length: periodic extension

$x[n]$

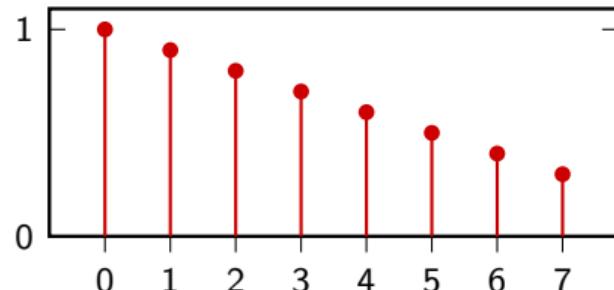
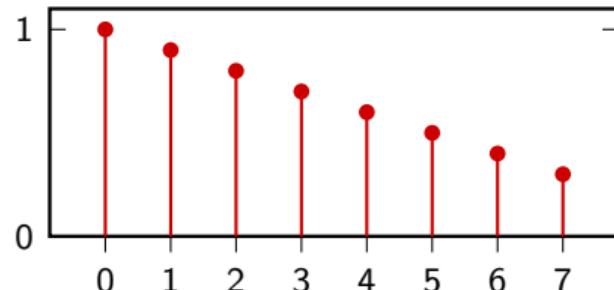
$\dots \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad \dots$



Shift of a finite-length: periodic extension

$\tilde{x}[n]$

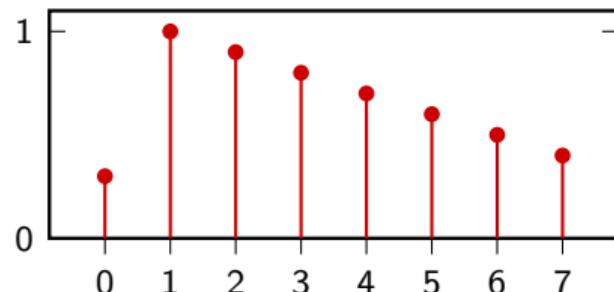
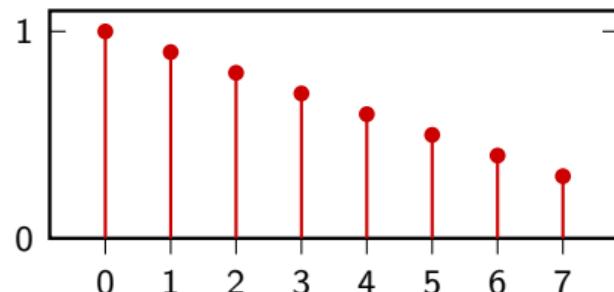
$\dots \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad \dots$



Shift of a finite-length: periodic extension

$$\tilde{x}[n - 1]$$

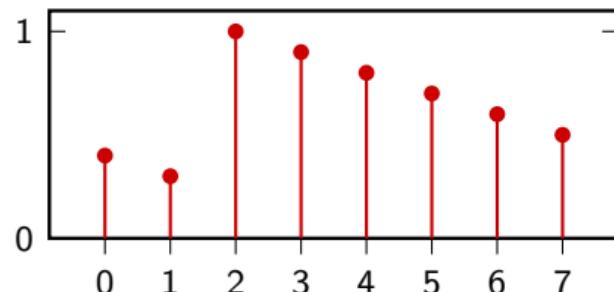
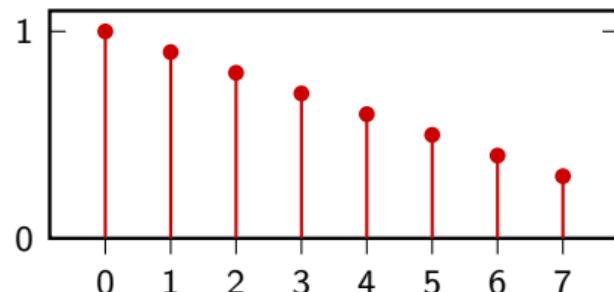
$\dots \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad \dots$



Shift of a finite-length: periodic extension

$$\tilde{x}[n - 2]$$

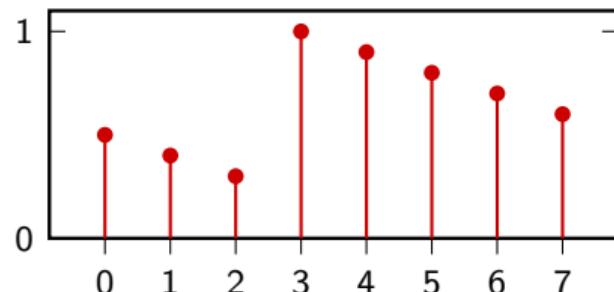
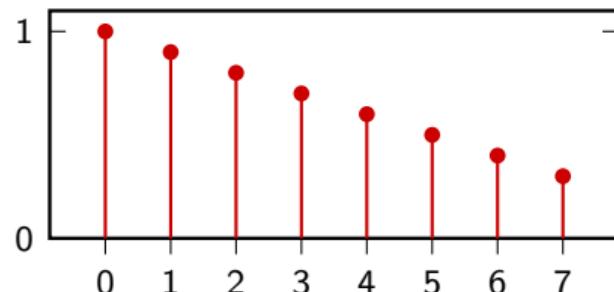
$\dots \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad \dots$



Shift of a finite-length: periodic extension

$$\tilde{x}[n - 3]$$

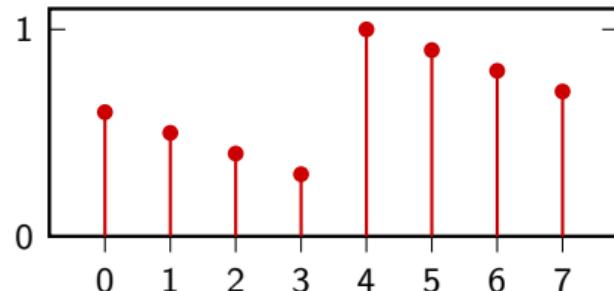
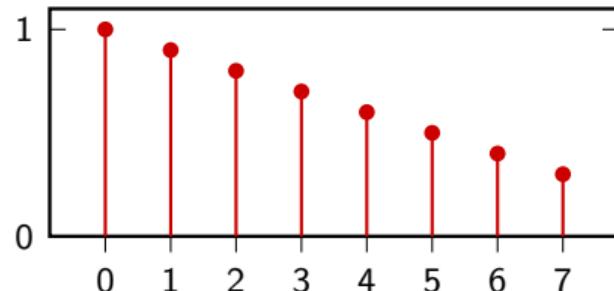
$\dots \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad \dots$



Shift of a finite-length: periodic extension

$$\tilde{x}[n - 4]$$

$\dots \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad \dots$



Energy and power

$$E_x = \sum_{n=-\infty}^{\infty} |x[n]|^2$$

$$P_x = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N |x[n]|^2$$

Energy and power

$$E_x = \sum_{n=-\infty}^{\infty} |x[n]|^2$$

$$P_x = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N |x[n]|^2$$

Energy and power: periodic signals

$$E_{\tilde{x}} = \infty$$

$$P_{\tilde{x}} \equiv \frac{1}{N} \sum_{n=0}^{N-1} |\tilde{x}[n]|^2$$

Energy and power: periodic signals

$$E_{\tilde{x}} = \infty$$

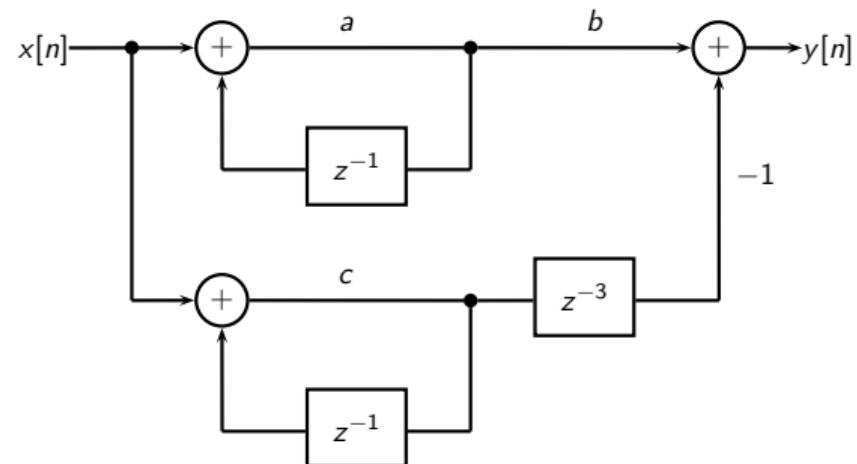
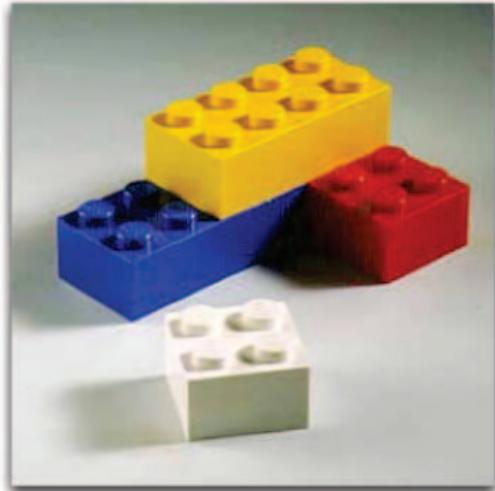
$$P_{\tilde{x}} \equiv \frac{1}{N} \sum_{n=0}^{N-1} |\tilde{x}[n]|^2$$

Let's play with LEGO!

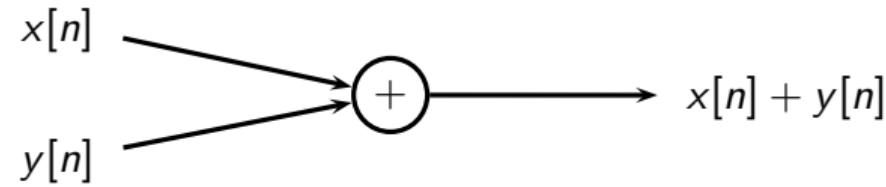
Overview:

- ▶ DSP as Lego: The fundamental building blocks
- ▶ Averages and moving averages
- ▶ Recursion: Revisiting your bank account
- ▶ Building a simple recursive synthesizer
- ▶ Examples of sounds

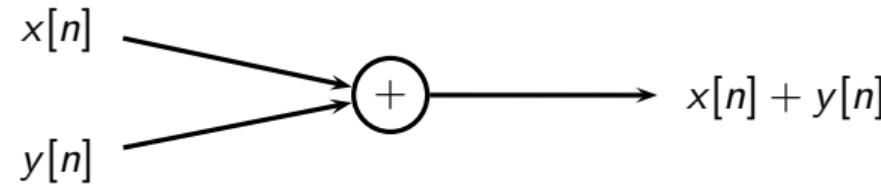
DSP as Lego



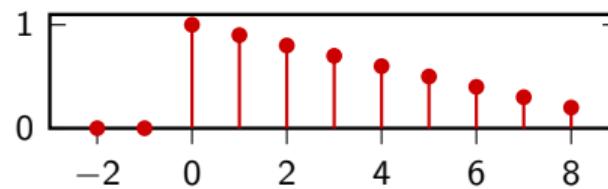
Building Blocks: Adder



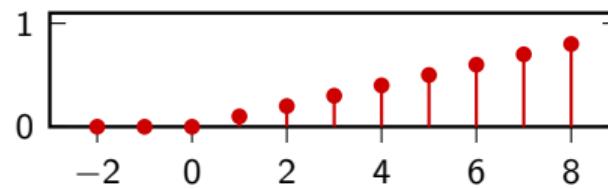
Building Blocks: Adder



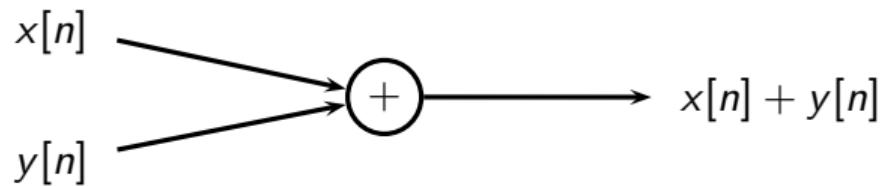
$$x[n] = 1 - n/10$$



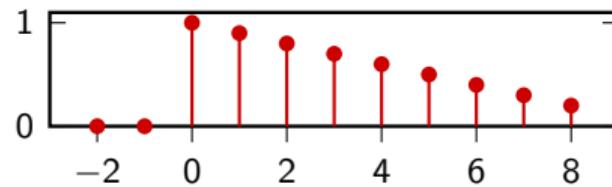
$$x[n] = n/10$$



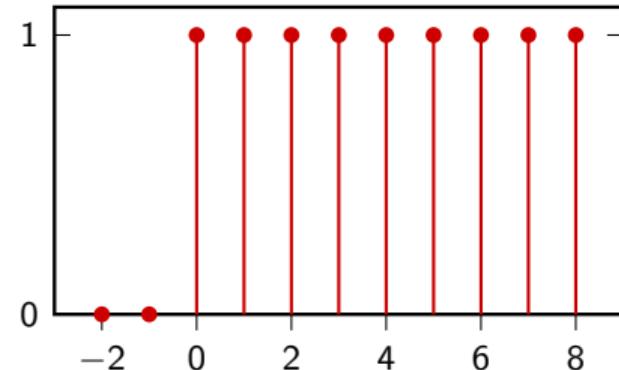
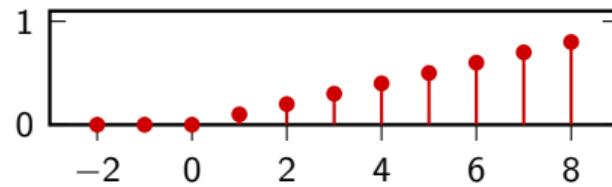
Building Blocks: Adder



$$x[n] = 1 - n/10$$



$$x[n] = n/10$$

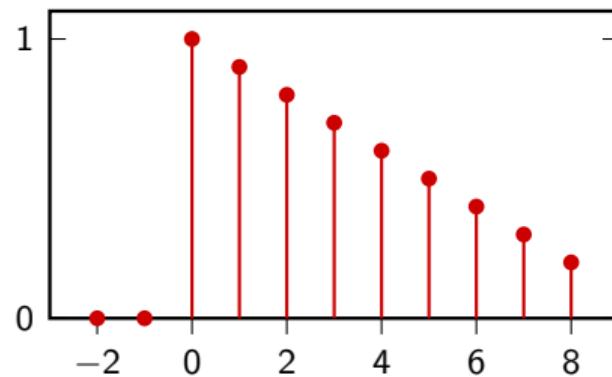


Building Blocks: Multiplier

$$x[n] \xrightarrow{\alpha} \alpha x[n]$$

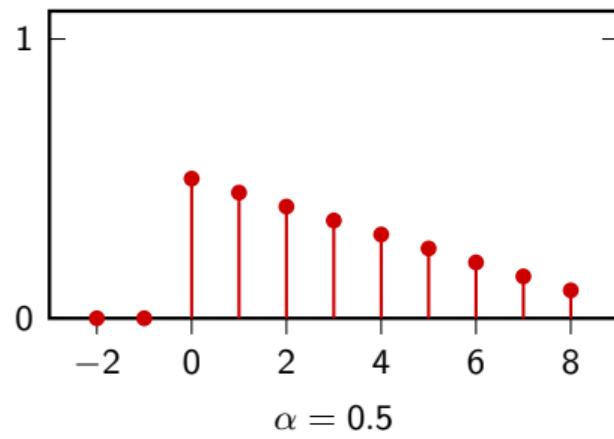
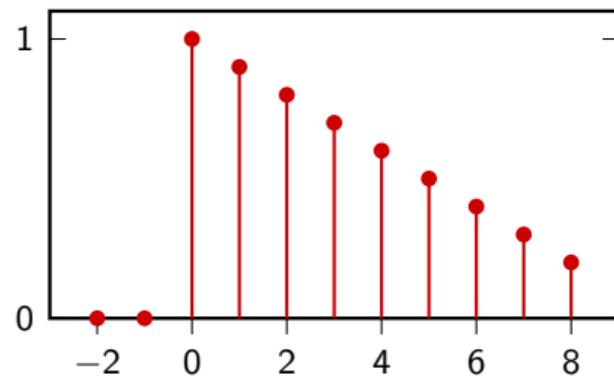
Building Blocks: Multiplier

$$x[n] \xrightarrow{\alpha} \alpha x[n]$$

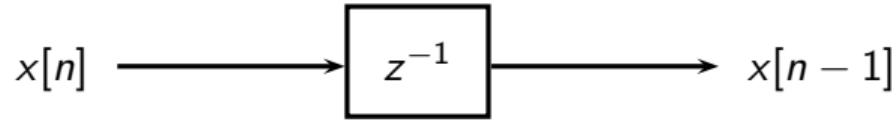


Building Blocks: Multiplier

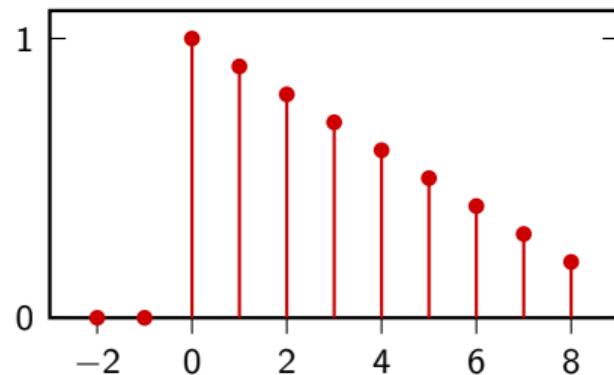
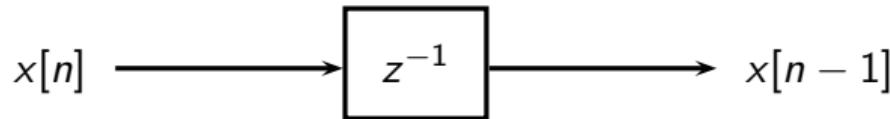
$$x[n] \xrightarrow{\alpha} \alpha x[n]$$



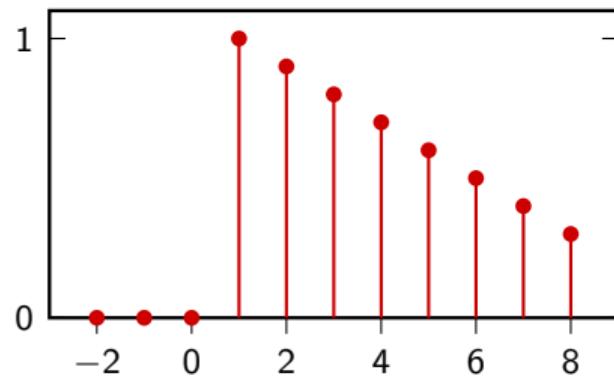
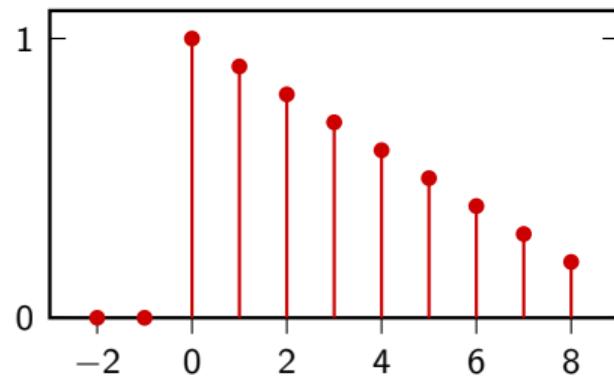
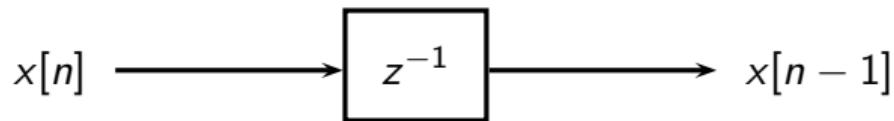
Building Blocks: Unit Delay



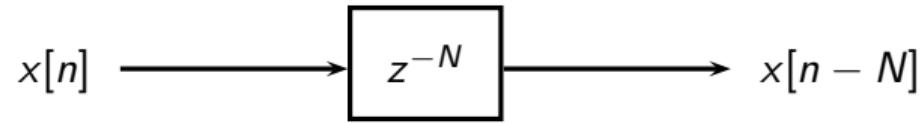
Building Blocks: Unit Delay



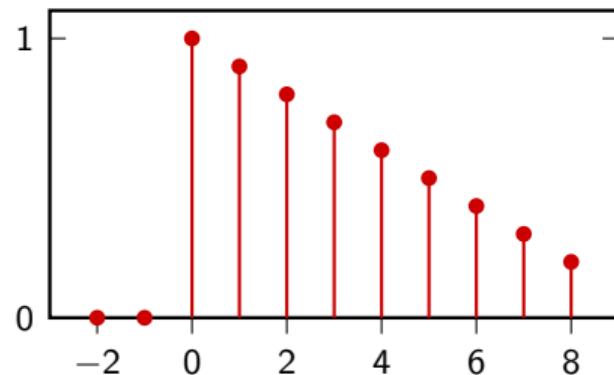
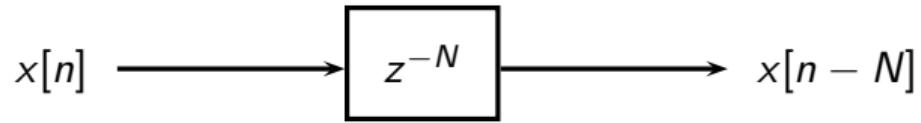
Building Blocks: Unit Delay



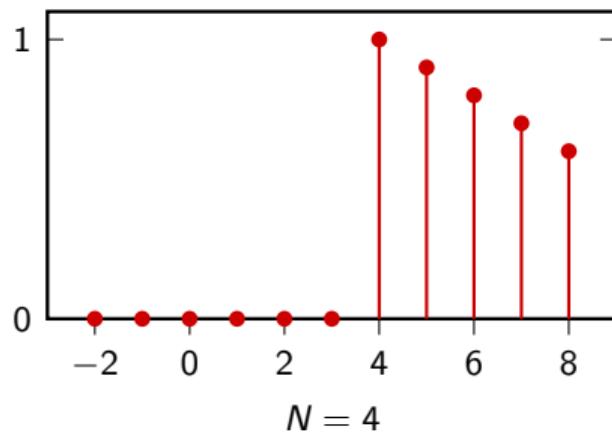
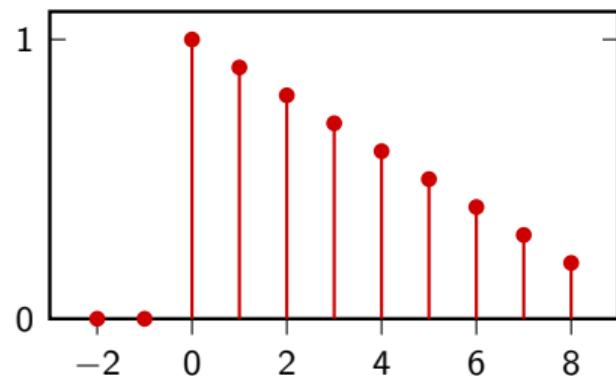
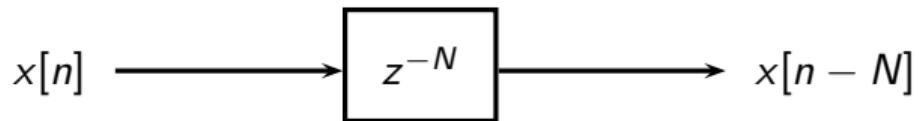
Building Blocks: Arbitrary Delay



Building Blocks: Arbitrary Delay



Building Blocks: Arbitrary Delay



The 2-point Moving Average

- ▶ simple average:

$$m = \frac{a + b}{2}$$

- ▶ moving average: take a “local” average

$$y[n] = \frac{x[n] + x[n - 1]}{2}$$

The 2-point Moving Average

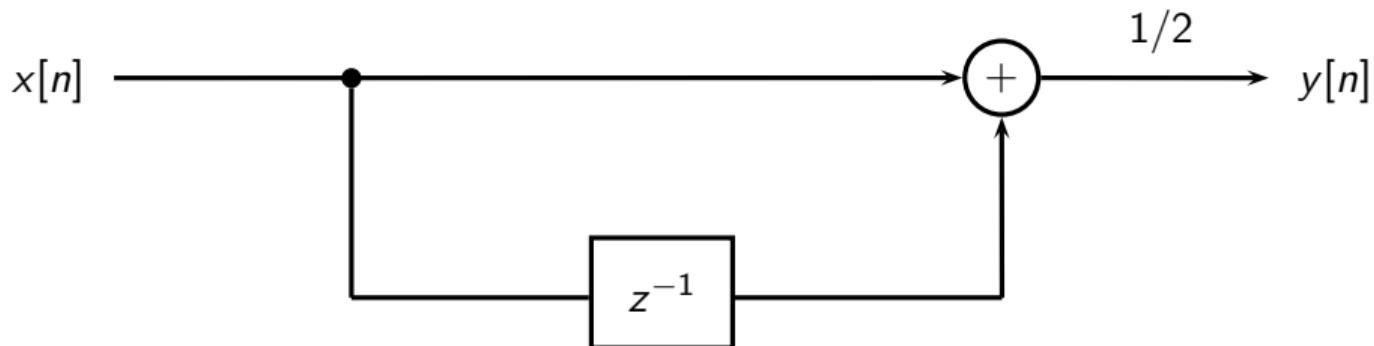
- ▶ simple average:

$$m = \frac{a + b}{2}$$

- ▶ moving average: take a “local” average

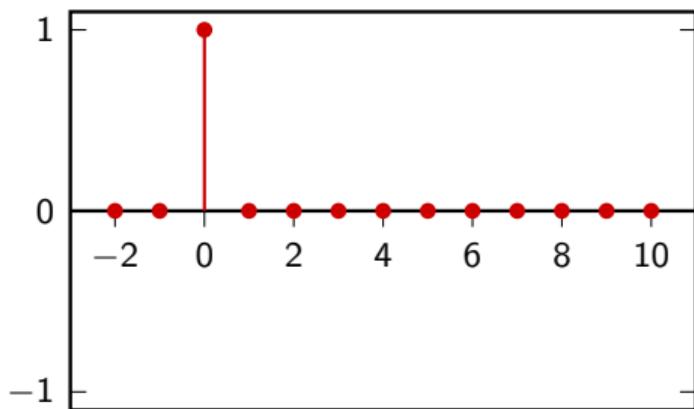
$$y[n] = \frac{x[n] + x[n - 1]}{2}$$

The 2-point Moving Average Using Lego



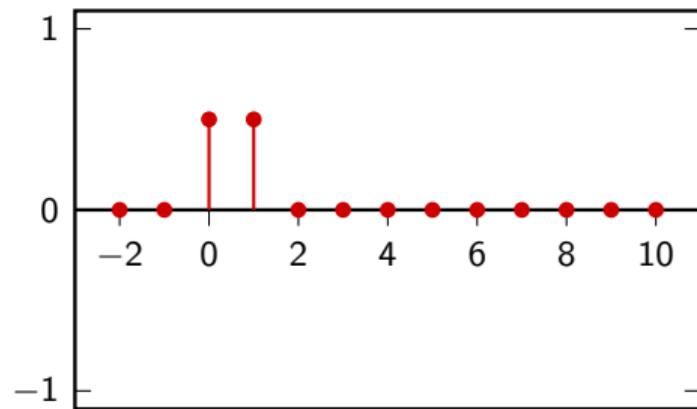
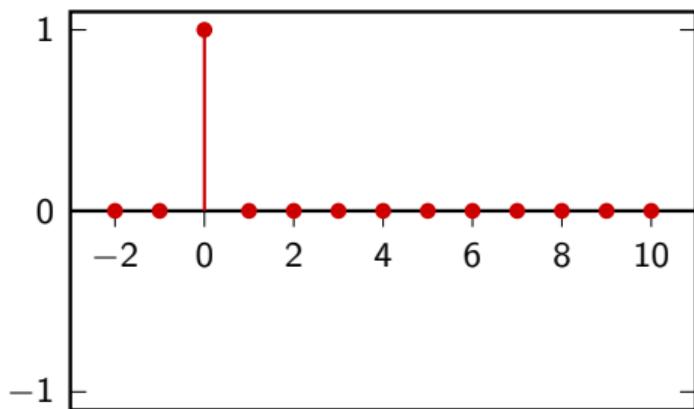
Let's average...

$$x[n] = \delta[n]$$



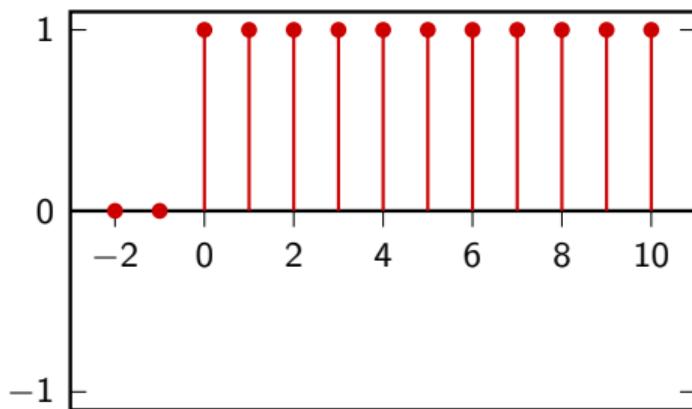
Let's average...

$$x[n] = \delta[n]$$



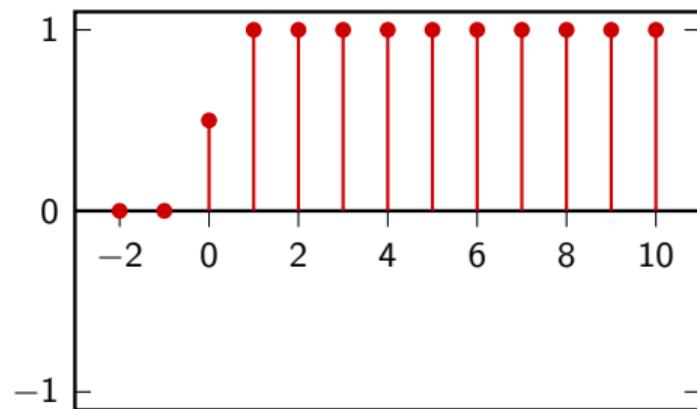
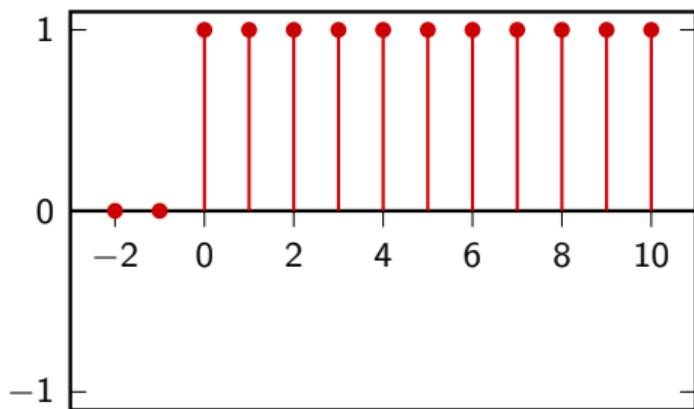
Let's average...

$$x[n] = u[n]$$



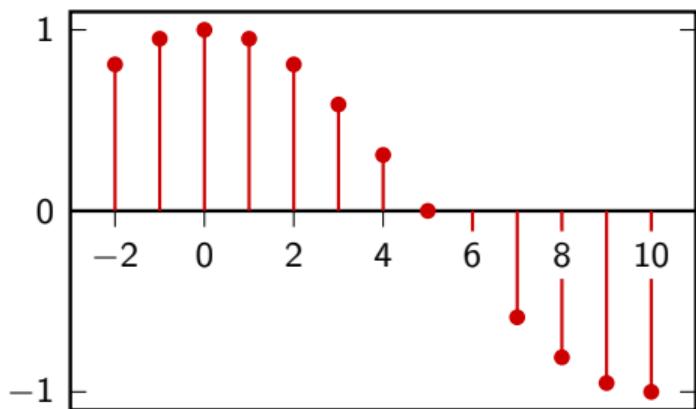
Let's average...

$$x[n] = u[n]$$



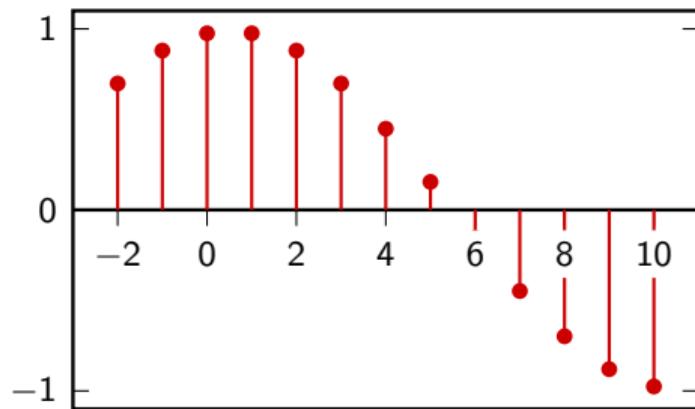
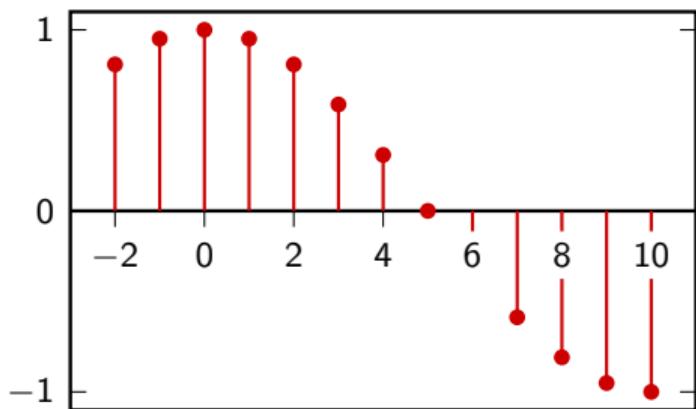
Let's average...

$$x[n] = \cos(\omega n), \quad \omega = \pi/10$$



Let's average...

$$x[n] = \cos(\omega n), \quad \omega = \pi/10$$



Let's average...

$$\cos(A) + \cos(B) = 2 \cos\left(\frac{A+B}{2}\right) \cos\left(\frac{A-B}{2}\right)$$

$$\begin{aligned}\frac{\cos(\omega n) + \cos(\omega(n-1))}{2} &= \cos\frac{\omega n + \omega n - \omega}{2} \cos\frac{\omega n - \omega n + \omega}{2} \\ &= \cos(\omega/2) \cos(\omega n - \omega/2)\end{aligned}$$

Don't remember the trigonometry? No problem

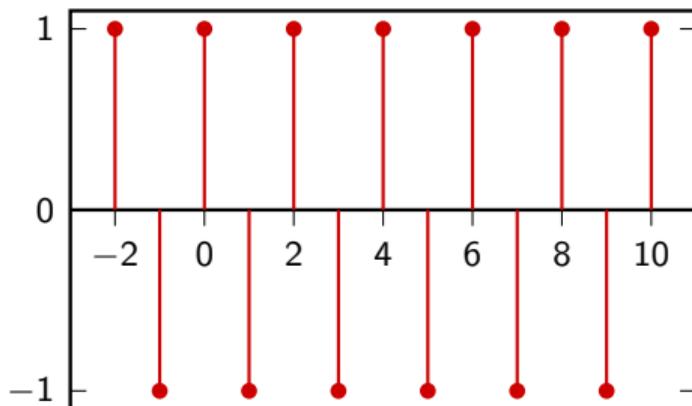
$$\begin{aligned}\frac{\cos(\omega n) + \cos(\omega(n-1))}{2} &= \frac{1}{2} \operatorname{Re}\{e^{j\omega n} + e^{j\omega(n+1)}\} \\ &= \frac{1}{2} \operatorname{Re}\{e^{j\omega(n+\frac{1}{2})}(e^{-j\omega/2} + e^{j\omega/2})\} \\ &= \cos(\omega/2) \operatorname{Re}\{e^{j\omega(n+\frac{1}{2})}\} \\ &= \cos(\omega/2) \cos(\omega n - \omega/2)\end{aligned}$$

Useful thing to remember

any linear combination of trigonometric functions at frequency ω_0 produces a sinusoid at the same frequency; linear combinations only alter the magnitude and phase

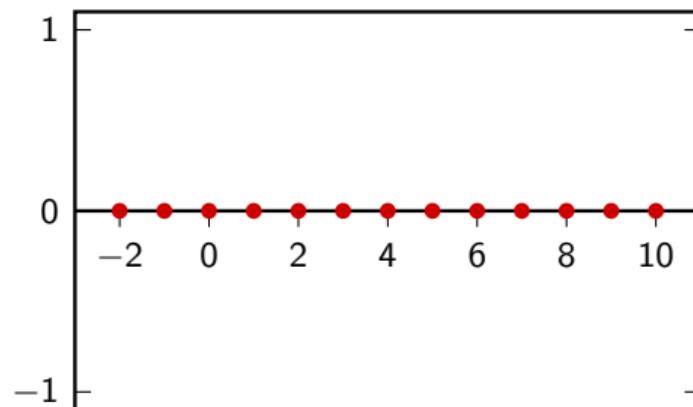
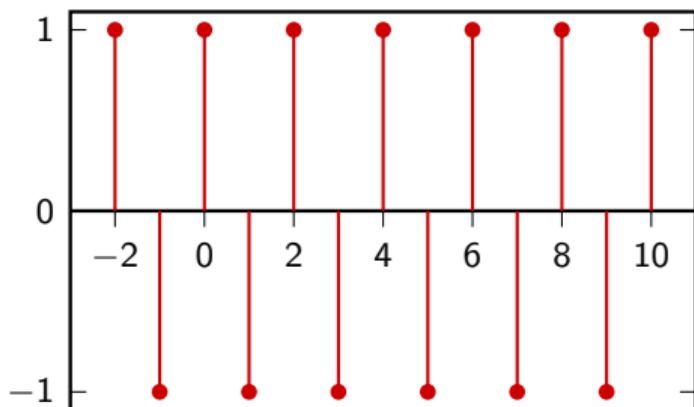
Let's average...

$$x[n] = (-1)^n$$

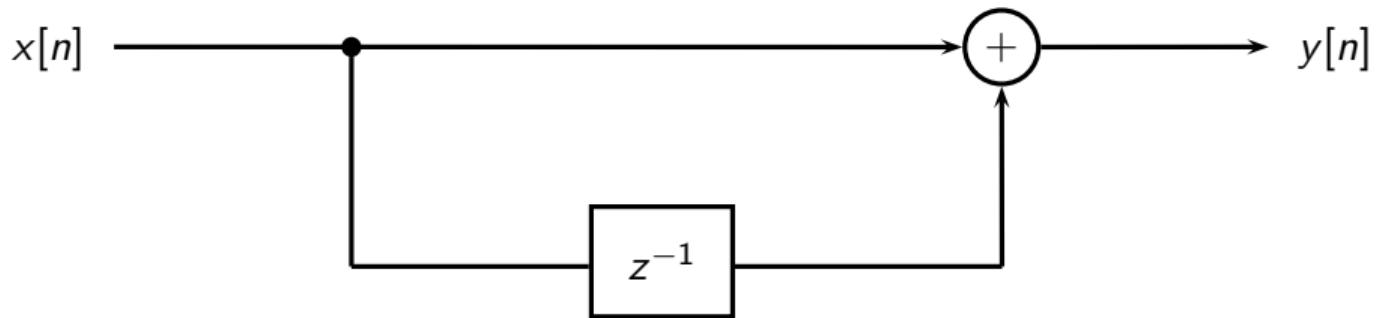


Let's average...

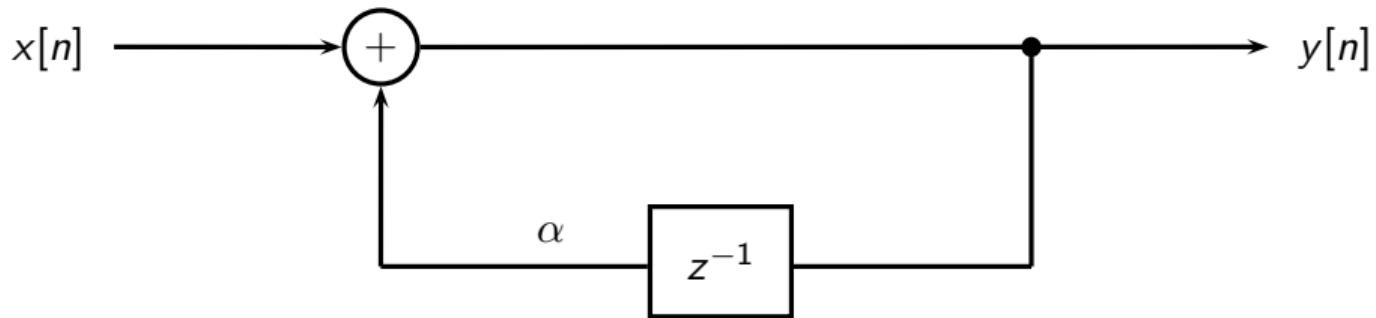
$$x[n] = (-1)^n$$



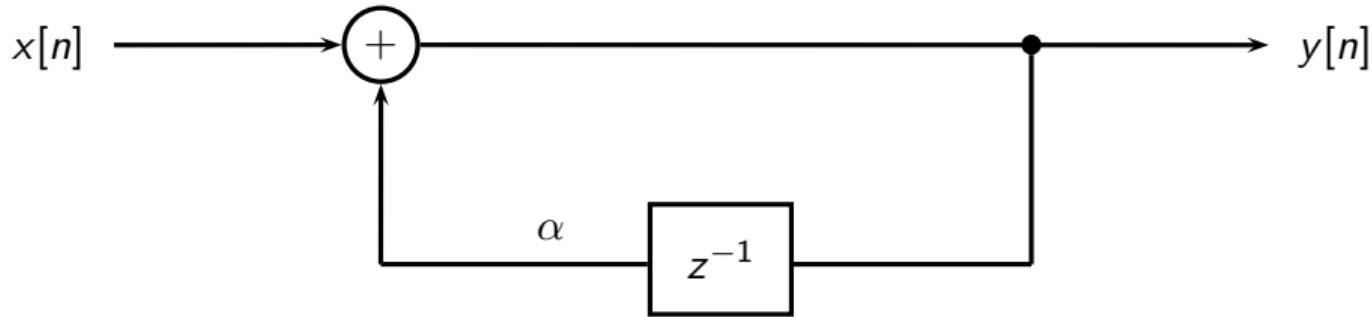
What if we reverse the loop?



What if we reverse the loop?

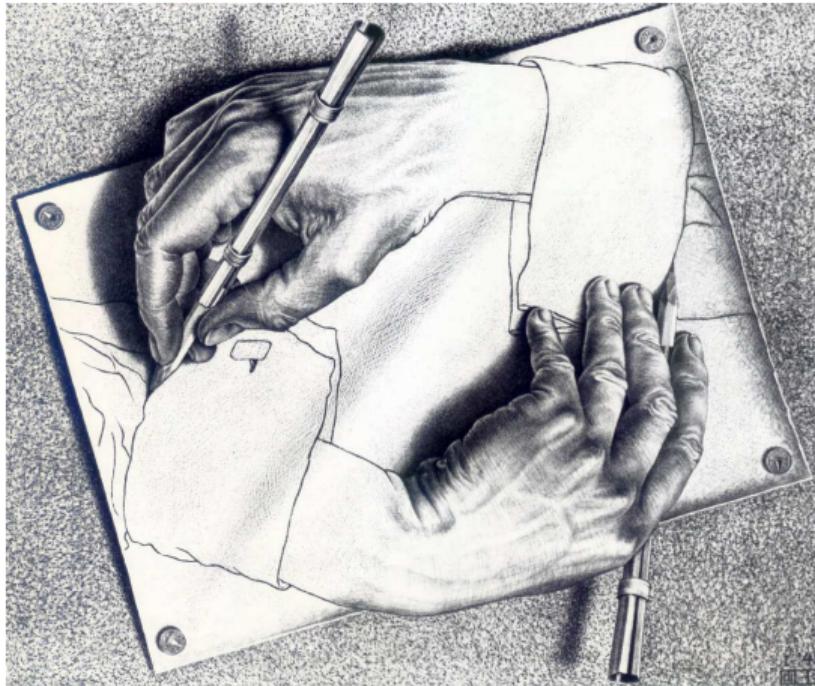


What if we reverse the loop?



$$y[n] = x[n] + \alpha y[n - 1]$$

A powerful concept: recursion



How we solve the chicken-and-egg problem

Zero Initial Conditions

- ▶ set a start time (usually $n_0 = 0$)
- ▶ assume input and output are zero for *all time* before n_0

A simple model for banking

A simple equation to describe compound interest:

- ▶ constant interest/borrowing rate of 5% per year
- ▶ interest accrues on Dec 31
- ▶ deposits/withdrawals during year n : $x[n]$
- ▶ balance at year n :

$$y[n] = 1.05 y[n - 1] + x[n]$$

A simple model for banking

A simple equation to describe compound interest:

- ▶ constant interest/borrowing rate of 5% per year
- ▶ interest accrues on Dec 31
- ▶ deposits/withdrawals during year n : $x[n]$
- ▶ balance at year n :

$$y[n] = 1.05 y[n - 1] + x[n]$$

A simple model for banking

A simple equation to describe compound interest:

- ▶ constant interest/borrowing rate of 5% per year
- ▶ interest accrues on Dec 31
- ▶ deposits/withdrawals during year n : $x[n]$
- ▶ balance at year n :

$$y[n] = 1.05 y[n - 1] + x[n]$$

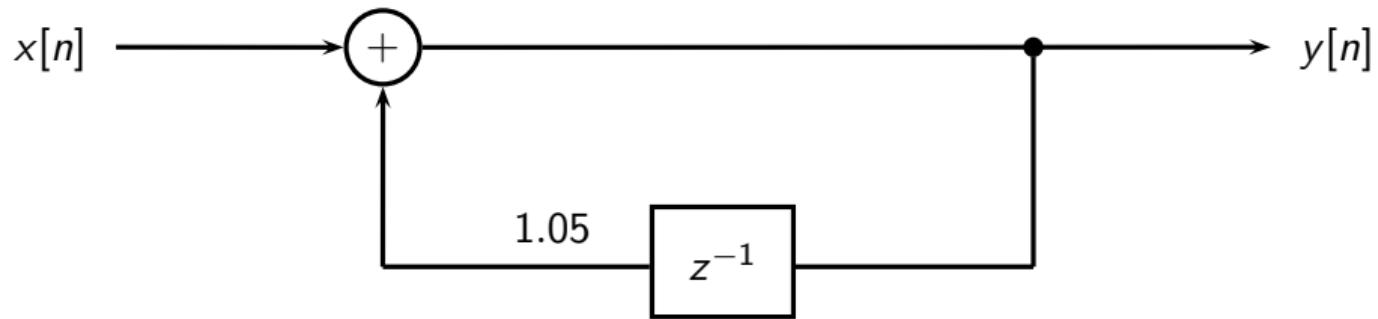
A simple model for banking

A simple equation to describe compound interest:

- ▶ constant interest/borrowing rate of 5% per year
- ▶ interest accrues on Dec 31
- ▶ deposits/withdrawals during year n : $x[n]$
- ▶ balance at year n :

$$y[n] = 1.05 y[n - 1] + x[n]$$

Accumulation of interest: first-order recursion

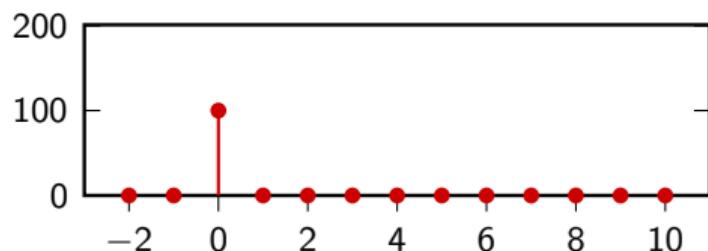


$$y[n] = 1.05 y[n - 1] + x[n]$$

Example: the one-time investment

$$x[n] = 100 \delta[n]$$

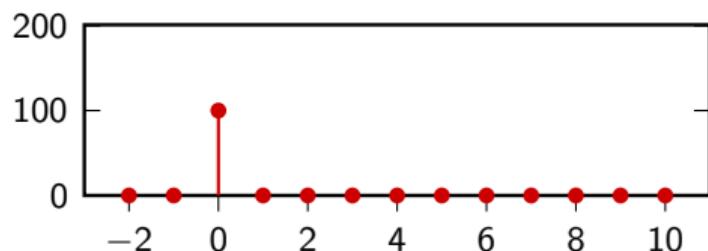
- ▶ $y[0] = 100$
- ▶ $y[1] = 105$
- ▶ $y[2] = 110.25, y[3] = 115.7625$ etc.
- ▶ In general: $y[n] = (1.05)^n 100 u[n]$



Example: the one-time investment

$$x[n] = 100 \delta[n]$$

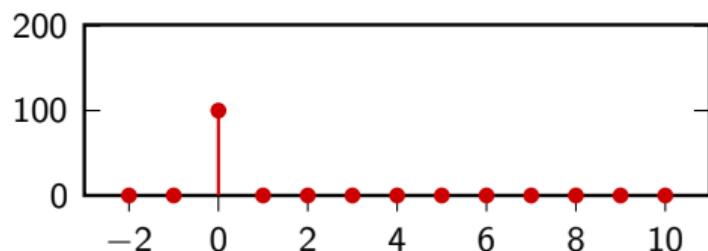
- ▶ $y[0] = 100$
- ▶ $y[1] = 105$
- ▶ $y[2] = 110.25, y[3] = 115.7625$ etc.
- ▶ In general: $y[n] = (1.05)^n 100 u[n]$



Example: the one-time investment

$$x[n] = 100 \delta[n]$$

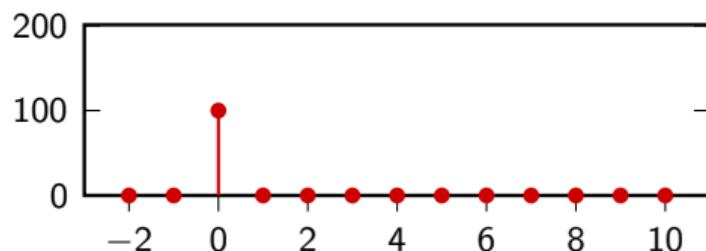
- ▶ $y[0] = 100$
- ▶ $y[1] = 105$
- ▶ $y[2] = 110.25, y[3] = 115.7625$ etc.
- ▶ In general: $y[n] = (1.05)^n 100 u[n]$



Example: the one-time investment

$$x[n] = 100 \delta[n]$$

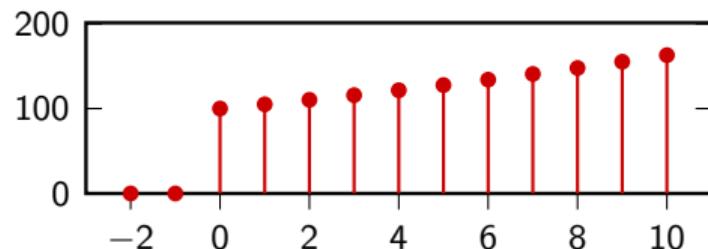
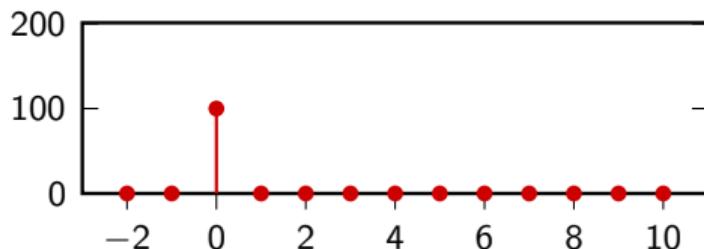
- ▶ $y[0] = 100$
- ▶ $y[1] = 105$
- ▶ $y[2] = 110.25, y[3] = 115.7625$ etc.
- ▶ In general: $y[n] = (1.05)^n 100 u[n]$



Example: the one-time investment

$$x[n] = 100 \delta[n]$$

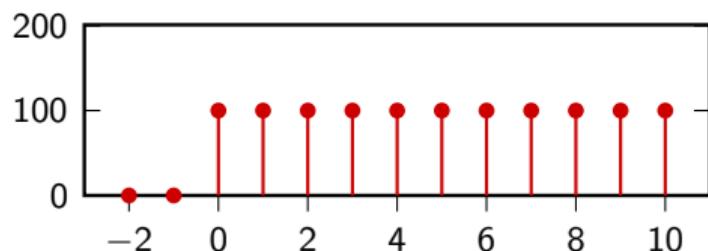
- ▶ $y[0] = 100$
- ▶ $y[1] = 105$
- ▶ $y[2] = 110.25, y[3] = 115.7625$ etc.
- ▶ In general: $y[n] = (1.05)^n 100 u[n]$



Example: the saver

$$x[n] = 100 u[n]$$

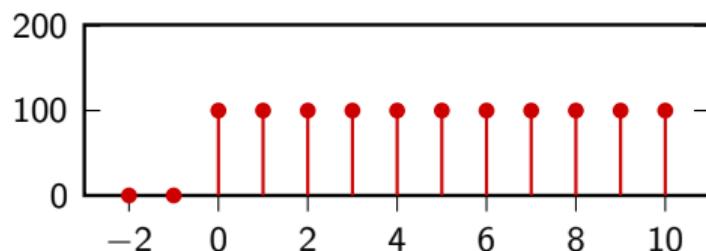
- ▶ $y[0] = 100$
- ▶ $y[1] = 205$
- ▶ $y[2] = 315.25, y[3] = 431.0125$ etc.
- ▶ In general: $y[n] = 2000 ((1.05)^{n+1} - 1) u[n]$



Example: the saver

$$x[n] = 100 u[n]$$

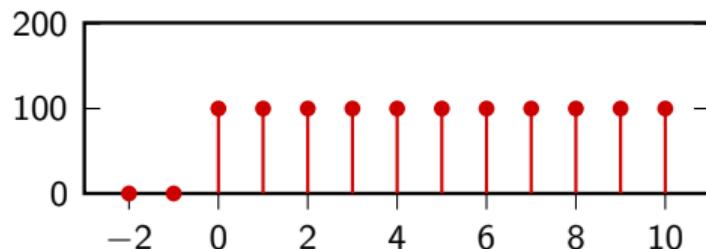
- ▶ $y[0] = 100$
- ▶ $y[1] = 205$
- ▶ $y[2] = 315.25, y[3] = 431.0125$ etc.
- ▶ In general: $y[n] = 2000 ((1.05)^{n+1} - 1) u[n]$



Example: the saver

$$x[n] = 100 u[n]$$

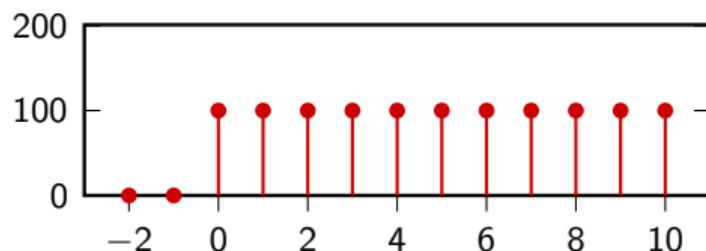
- ▶ $y[0] = 100$
- ▶ $y[1] = 205$
- ▶ $y[2] = 315.25, y[3] = 431.0125$ etc.
- ▶ In general: $y[n] = 2000 ((1.05)^{n+1} - 1) u[n]$



Example: the saver

$$x[n] = 100 u[n]$$

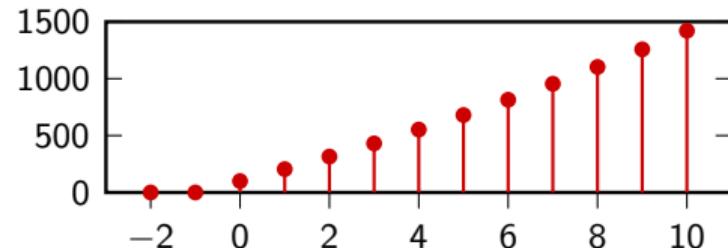
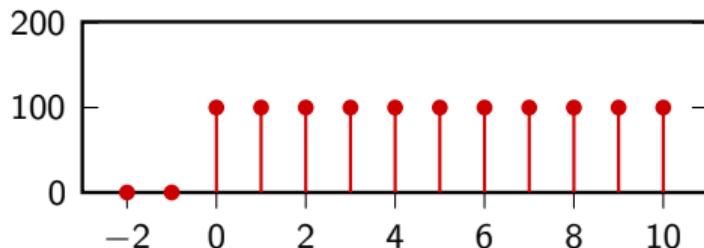
- ▶ $y[0] = 100$
- ▶ $y[1] = 205$
- ▶ $y[2] = 315.25, y[3] = 431.0125$ etc.
- ▶ In general: $y[n] = 2000 ((1.05)^{n+1} - 1) u[n]$



Example: the saver

$$x[n] = 100 u[n]$$

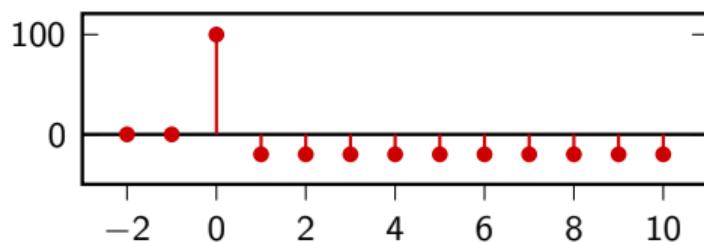
- ▶ $y[0] = 100$
- ▶ $y[1] = 205$
- ▶ $y[2] = 315.25, y[3] = 431.0125$ etc.
- ▶ In general: $y[n] = 2000 ((1.05)^{n+1} - 1) u[n]$



Example: living off the interest

$$x[n] = 100 \delta[n] - 5 u[n - 1]$$

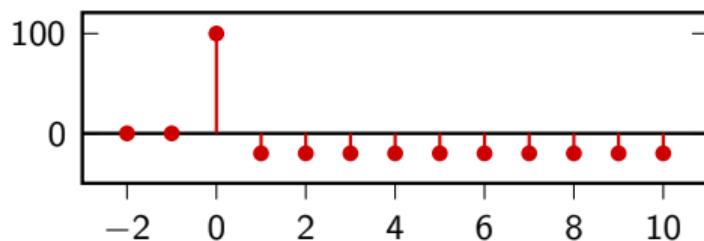
- ▶ $y[0] = 100$
- ▶ $y[1] = 100$
- ▶ $y[2] = 100, y[3] = 100$ etc.
- ▶ In general: $y[n] = 100 u[n]$



Example: living off the interest

$$x[n] = 100 \delta[n] - 5 u[n - 1]$$

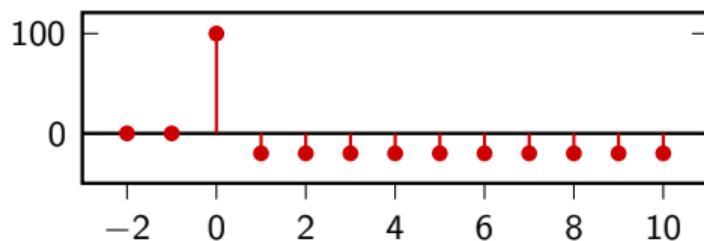
- ▶ $y[0] = 100$
- ▶ $y[1] = 100$
- ▶ $y[2] = 100, y[3] = 100$ etc.
- ▶ In general: $y[n] = 100 u[n]$



Example: living off the interest

$$x[n] = 100 \delta[n] - 5 u[n - 1]$$

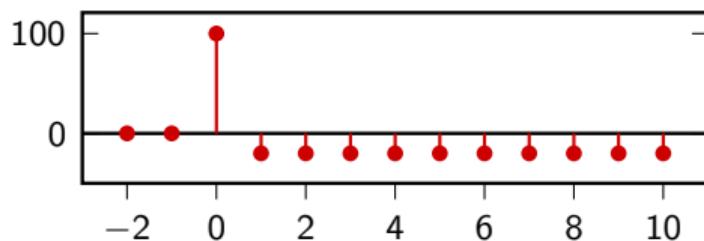
- ▶ $y[0] = 100$
- ▶ $y[1] = 100$
- ▶ $y[2] = 100, y[3] = 100$ etc.
- ▶ In general: $y[n] = 100 u[n]$



Example: living off the interest

$$x[n] = 100 \delta[n] - 5 u[n - 1]$$

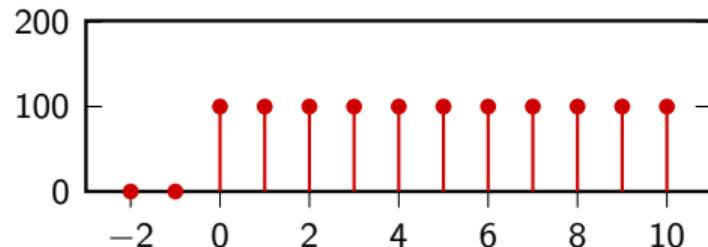
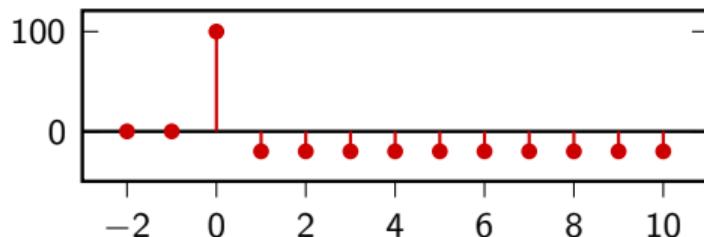
- ▶ $y[0] = 100$
- ▶ $y[1] = 100$
- ▶ $y[2] = 100, y[3] = 100$ etc.
- ▶ In general: $y[n] = 100 u[n]$



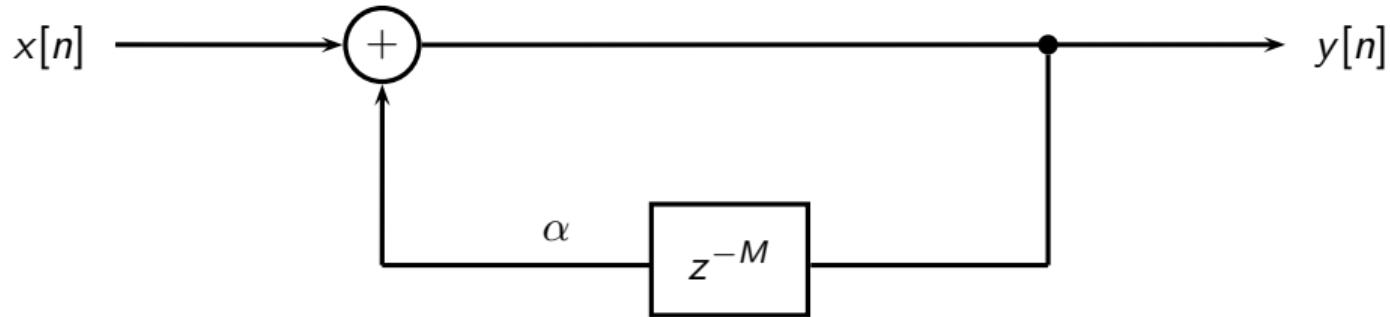
Example: living off the interest

$$x[n] = 100 \delta[n] - 5 u[n - 1]$$

- ▶ $y[0] = 100$
- ▶ $y[1] = 100$
- ▶ $y[2] = 100, y[3] = 100$ etc.
- ▶ In general: $y[n] = 100 u[n]$

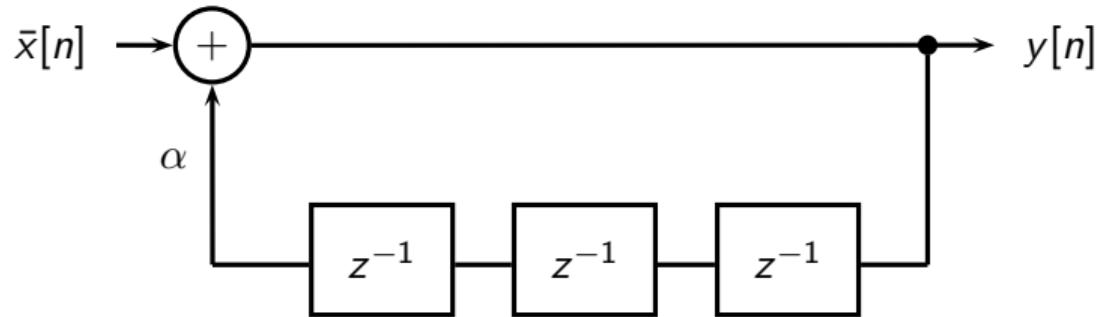


An interesting generalization



$$y[n] = \alpha y[n - M] + x[n]$$

Creating loops

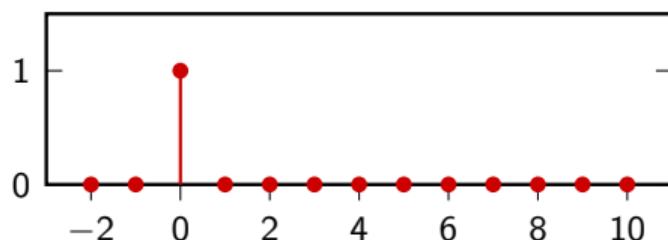


$$y[n] = \alpha y[n - 3] + \bar{x}[n]$$

Example

$$M = 3, \alpha = 0.7, x[n] = \delta[n]$$

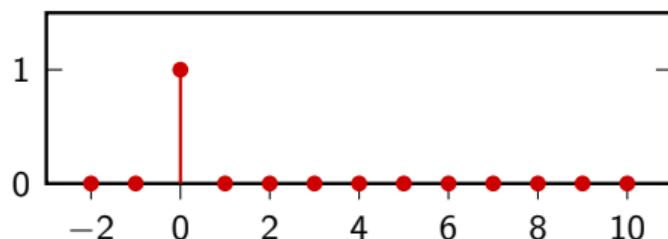
- ▶ $y[0] = 1, y[1] = 0, y[2] = 0$
- ▶ $y[3] = 0.7, y[4] = 0, y[5] = 0$
- ▶ $y[6] = 0.7^2, y[7] = 0, y[8] = 0$, etc.



Example

$$M = 3, \alpha = 0.7, x[n] = \delta[n]$$

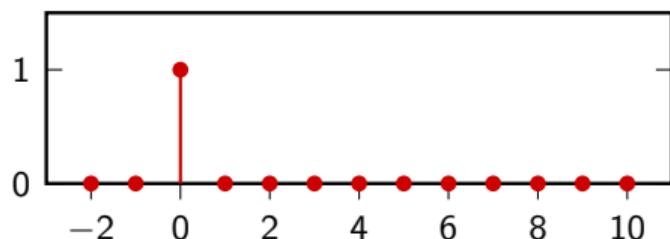
- ▶ $y[0] = 1, y[1] = 0, y[2] = 0$
- ▶ $y[3] = 0.7, y[4] = 0, y[5] = 0$
- ▶ $y[6] = 0.7^2, y[7] = 0, y[8] = 0$, etc.



Example

$$M = 3, \alpha = 0.7, x[n] = \delta[n]$$

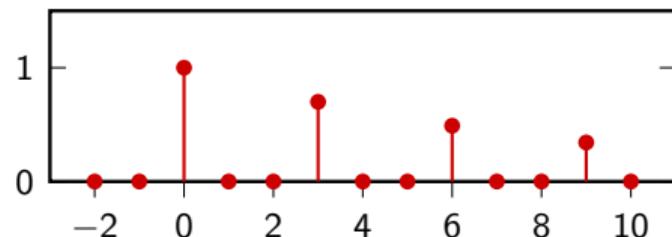
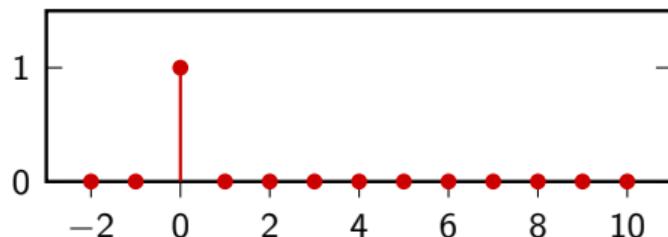
- ▶ $y[0] = 1, y[1] = 0, y[2] = 0$
- ▶ $y[3] = 0.7, y[4] = 0, y[5] = 0$
- ▶ $y[6] = 0.7^2, y[7] = 0, y[8] = 0$, etc.



Example

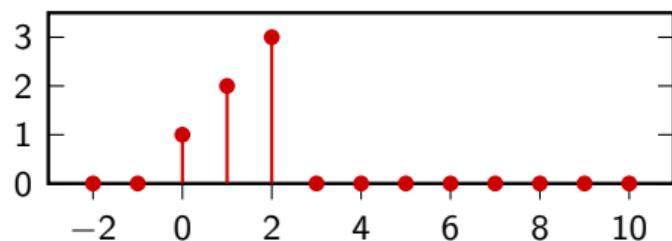
$$M = 3, \alpha = 0.7, x[n] = \delta[n]$$

- ▶ $y[0] = 1, y[1] = 0, y[2] = 0$
- ▶ $y[3] = 0.7, y[4] = 0, y[5] = 0$
- ▶ $y[6] = 0.7^2, y[7] = 0, y[8] = 0$, etc.



Example

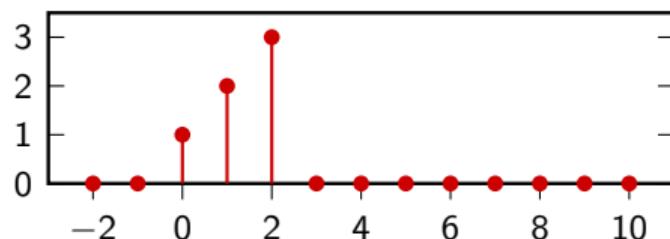
$$M = 3, \alpha = 1, x[n] = \delta[n] + 2\delta[n - 1] + 3\delta[n - 2]$$



Example

$$M = 3, \alpha = 1, x[n] = \delta[n] + 2\delta[n - 1] + 3\delta[n - 2]$$

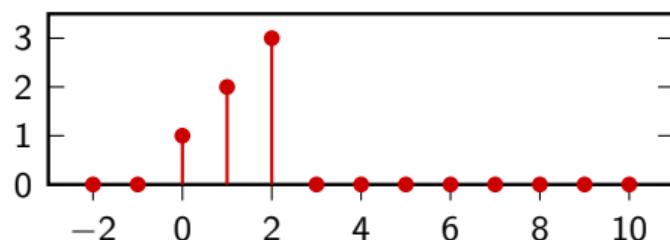
► $y[0] = 1, y[1] = 2, y[2] = 3$



Example

$$M = 3, \alpha = 1, x[n] = \delta[n] + 2\delta[n - 1] + 3\delta[n - 2]$$

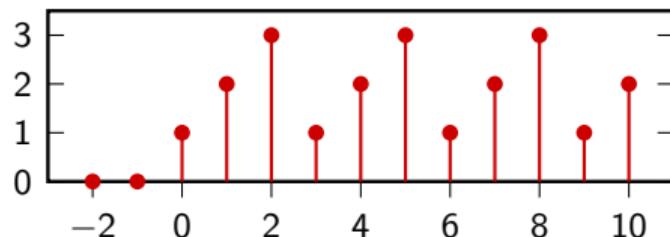
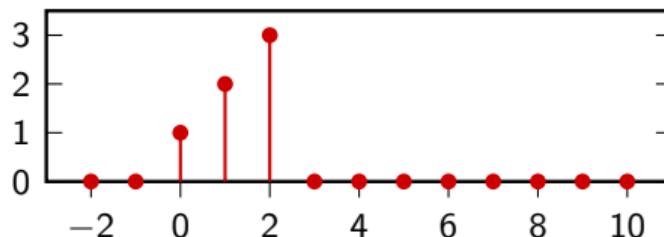
- ▶ $y[0] = 1, y[1] = 2, y[2] = 3$
- ▶ $y[3] = 1, y[4] = 2, y[5] = 3$



Example

$$M = 3, \alpha = 1, x[n] = \delta[n] + 2\delta[n - 1] + 3\delta[n - 2]$$

- ▶ $y[0] = 1, y[1] = 2, y[2] = 3$
- ▶ $y[3] = 1, y[4] = 2, y[5] = 3$
- ▶ $y[6] = 1, y[7] = 2, y[8] = 3, \text{ etc.}$



We can make music with that!

- ▶ build a recursion loop with a delay of M
- ▶ choose a signal $\bar{x}[n]$ that is nonzero only for $0 \leq n < M$
- ▶ choose a decay factor
- ▶ input $\bar{x}[n]$ to the system
- ▶ play the output

We can make music with that!

- ▶ build a recursion loop with a delay of M
- ▶ choose a signal $\bar{x}[n]$ that is nonzero only for $0 \leq n < M$
- ▶ choose a decay factor
- ▶ input $\bar{x}[n]$ to the system
- ▶ play the output

We can make music with that!

- ▶ build a recursion loop with a delay of M
- ▶ choose a signal $\bar{x}[n]$ that is nonzero only for $0 \leq n < M$
- ▶ choose a decay factor
- ▶ input $\bar{x}[n]$ to the system
- ▶ play the output

We can make music with that!

- ▶ build a recursion loop with a delay of M
- ▶ choose a signal $\bar{x}[n]$ that is nonzero only for $0 \leq n < M$
- ▶ choose a decay factor
- ▶ input $\bar{x}[n]$ to the system
- ▶ play the output

We can make music with that!

- ▶ build a recursion loop with a delay of M
- ▶ choose a signal $\bar{x}[n]$ that is nonzero only for $0 \leq n < M$
- ▶ choose a decay factor
- ▶ input $\bar{x}[n]$ to the system
- ▶ play the output

How do we “play” it, really?

- ▶ M -tap delay → M -sample “periodicity”

- ▶ associate time T to sample interval
- ▶ periodic signal of frequency

$$f = \frac{1}{MT} \text{Hz}$$

- ▶ example: $T = 22.7\mu\text{s}$, $M = 100$

$$f \approx 440\text{Hz}$$

How do we “play” it, really?

- ▶ M -tap delay $\longrightarrow M$ -sample “periodicity”
- ▶ associate time T to sample interval
- ▶ periodic signal of frequency

$$f = \frac{1}{MT} \text{Hz}$$

- ▶ example: $T = 22.7\mu\text{s}$, $M = 100$

$$f \approx 440\text{Hz}$$

How do we “play” it, really?

- ▶ M -tap delay $\longrightarrow M$ -sample “periodicity”
- ▶ associate time T to sample interval
- ▶ periodic signal of frequency

$$f = \frac{1}{MT} \text{Hz}$$

- ▶ example: $T = 22.7\mu\text{s}$, $M = 100$

$$f \approx 440\text{Hz}$$

How do we “play” it, really?

- ▶ M -tap delay → M -sample “periodicity”
- ▶ associate time T to sample interval
- ▶ periodic signal of frequency

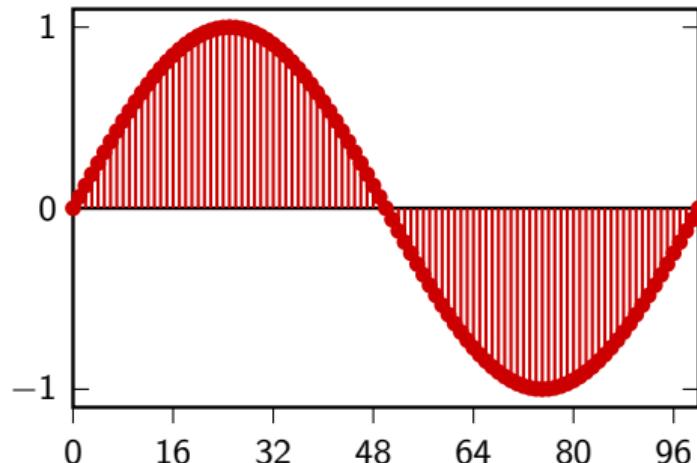
$$f = \frac{1}{MT} \text{Hz}$$

- ▶ example: $T = 22.7\mu s$, $M = 100$

$$f \approx 440 \text{Hz}$$

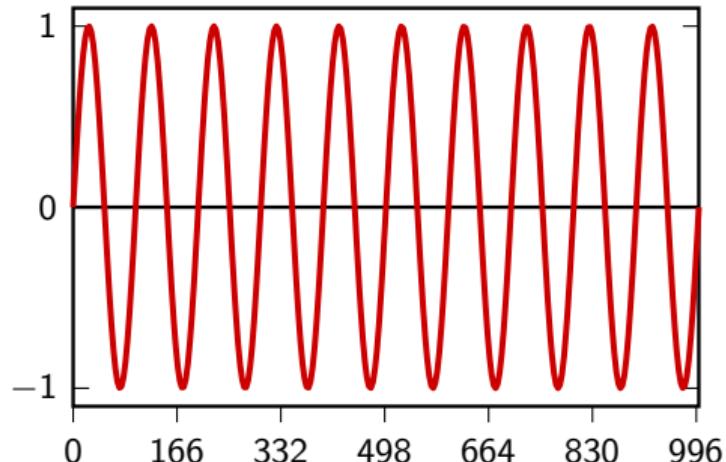
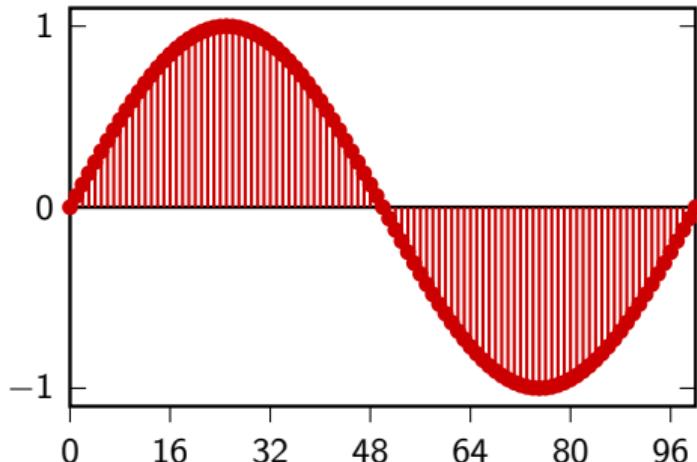
Playing a sine wave

$M = 100, \alpha = 1, \bar{x}[n] = \sin(2\pi n/100)$ for $0 \leq n < 100$ and zero elsewhere



Playing a sine wave

$M = 100, \alpha = 1, \bar{x}[n] = \sin(2\pi n/100)$ for $0 \leq n < 100$ and zero elsewhere



Play

Introducing some realism

- ▶ M controls frequency (pitch)
- ▶ α controls envelope (decay)
- ▶ $\bar{x}[n]$ controls color (timbre)

Introducing some realism

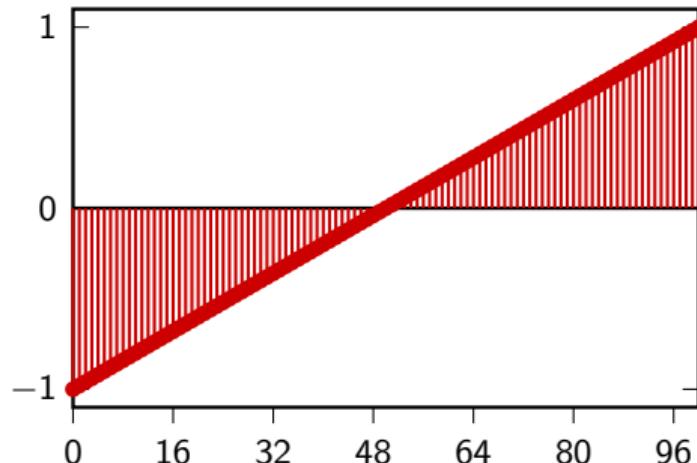
- ▶ M controls frequency (pitch)
- ▶ α controls envelope (decay)
- ▶ $\bar{x}[n]$ controls color (timbre)

Introducing some realism

- ▶ M controls frequency (pitch)
- ▶ α controls envelope (decay)
- ▶ $\bar{x}[n]$ controls color (timbre)

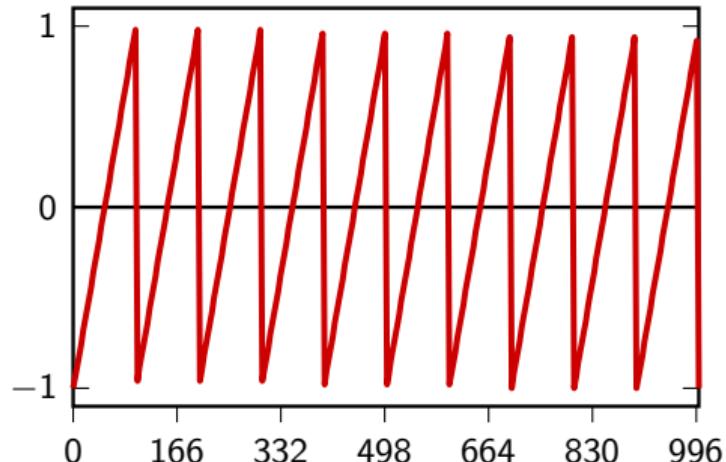
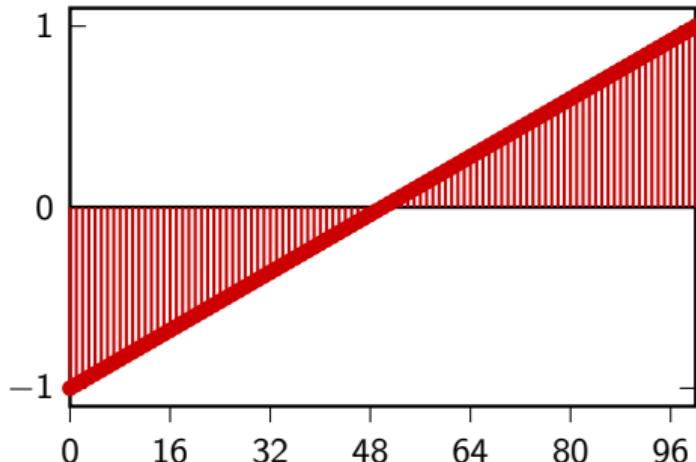
A proto-violin

$M = 100, \alpha = 0.95, \bar{x}[n]$: zero-mean sawtooth wave between 0 and 99, zero elsewhere



A proto-violin

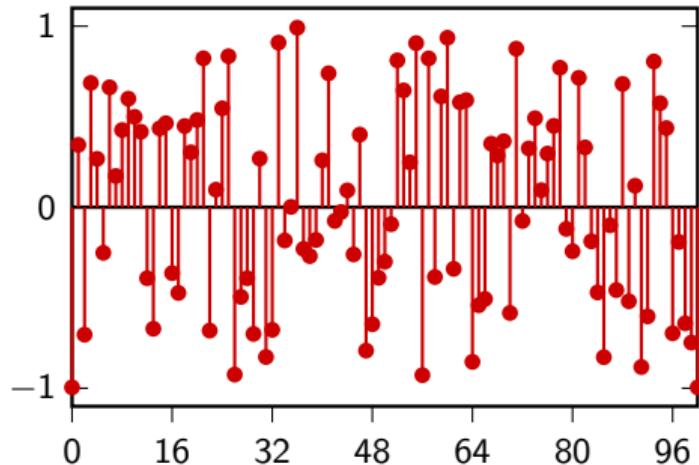
$M = 100, \alpha = 0.95, \bar{x}[n]$: zero-mean sawtooth wave between 0 and 99, zero elsewhere



Play

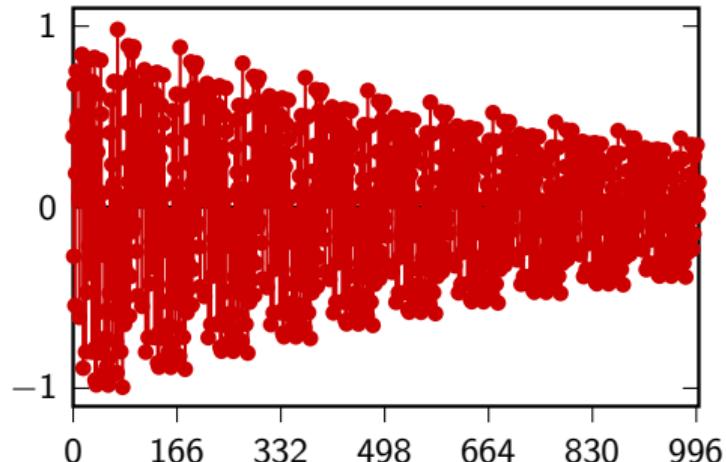
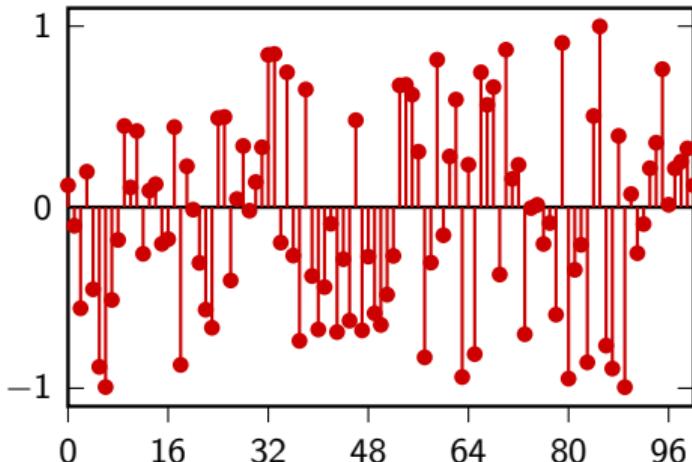
The Karplus-Strong Algorithm

$M = 100, \alpha = 0.9, \bar{x}[n]$: 100 random values between 0 and 99, zero elsewhere



The Karplus-Strong Algorithm

$M = 100, \alpha = 0.9, \bar{x}[n]$: 100 random values between 0 and 99, zero elsewhere



Play

Recap

- ▶ We have seen basic elements:
 - adders
 - multipliers
 - delays
- ▶ We have seen two systems
 - moving averages
 - recursive systems
- ▶ We were able to build simple systems with interesting properties
- ▶ to understand all of this in more details we need a mathematical framework!

COM303: Digital Signal Processing

Lecture 3: Signal Processing and Vector Spaces

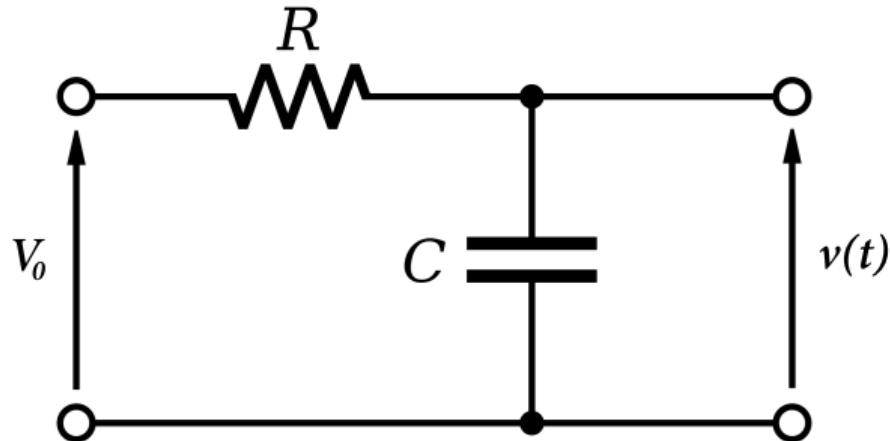
Module Overview:

- ▶ signal processing as geometry
- ▶ vectors and vector spaces
- ▶ Hilbert space and basis

Signal Models (in Physics)

Description of the evolution of a physical phenomenon

Signal Models (in Physics)



$$v(t) = V_0(1 - e^{-\frac{t}{RC}})$$

Signal Models (in Physics)

$$f : \mathbb{R} \rightarrow \mathbb{R}$$

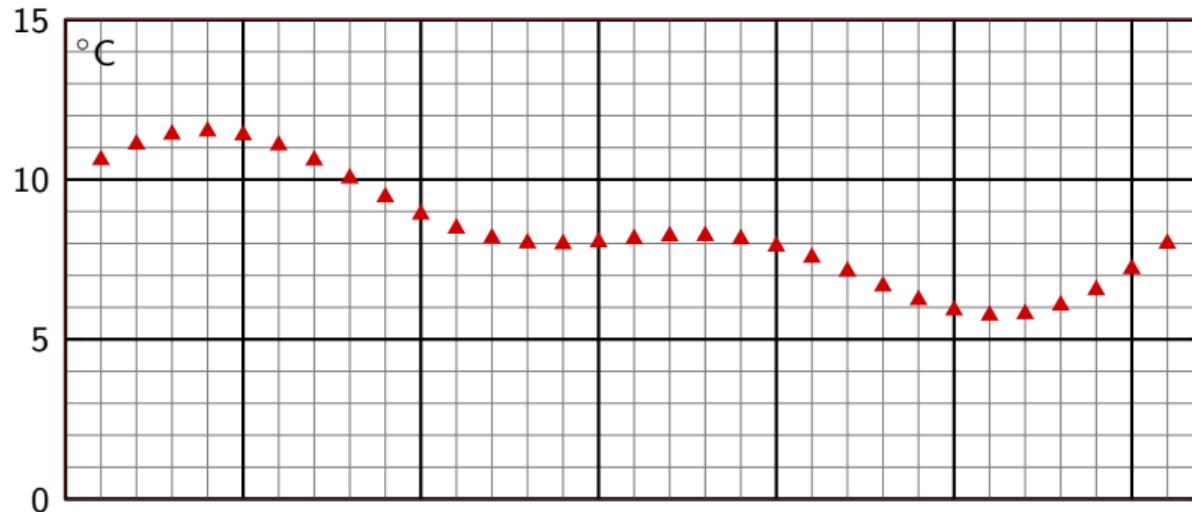
Signal Models (in DSP)

$$\cancel{f : \mathbb{R} \rightarrow \mathbb{R}}$$

Signal Models (in DSP)

$$x[n] = \dots, 1.2390, -0.7372, 0.8987, 0.1798, -1.1501, -0.2642 \dots$$

Signal Models (in DSP)



Discrete-Time Signal Model

$$\mathbb{C}^N$$

Discrete-Time Signal Model

\mathbb{C}^N : vector space of ordered tuples of N complex values

- ▶ complex values, because we can
- ▶ N can be ∞
- ▶ we will need more than just a vector space (Hilbert space)

Discrete-Time Signal Model

\mathbb{C}^N : vector space of ordered tuples of N complex values

- ▶ complex values, because we can
- ▶ N can be ∞
- ▶ we will need more than just a vector space (Hilbert space)

Let's talk about Vector Spaces...

Some spaces should be very familiar:

- ▶ $\mathbb{R}^2, \mathbb{R}^3$: Euclidean space, geometry
- ▶ $\mathbb{R}^N, \mathbb{C}^N$: linear algebra

Others perhaps not so much...

- ▶ $\ell_2(\mathbb{Z})$: space of square-summable infinite sequences
- ▶ $L_2([a, b])$: space of square-integrable *functions* over an interval

yes, vectors can be functions!

Let's talk about Vector Spaces...

Some spaces should be very familiar:

- ▶ $\mathbb{R}^2, \mathbb{R}^3$: Euclidean space, geometry
- ▶ $\mathbb{R}^N, \mathbb{C}^N$: linear algebra

Others perhaps not so much...

- ▶ $\ell_2(\mathbb{Z})$: space of square-summable infinite sequences
- ▶ $L_2([a, b])$: space of square-integrable *functions* over an interval

yes, vectors can be functions!

Let's talk about Vector Spaces...

Some spaces should be very familiar:

- ▶ $\mathbb{R}^2, \mathbb{R}^3$: Euclidean space, geometry
- ▶ $\mathbb{R}^N, \mathbb{C}^N$: linear algebra

Others perhaps not so much...

- ▶ $\ell_2(\mathbb{Z})$: space of square-summable infinite sequences
- ▶ $L_2([a, b])$: space of square-integrable *functions* over an interval

yes, vectors can be functions!

Why using vector vpaces in DSP?

Easier math and unified framework for signal processing:

- ▶ same object for different classes of signals (finite-length, finite-support, infinite, periodic)
- ▶ easy explanation of the Fourier Transform
- ▶ easy explanation of sampling and interpolation
- ▶ useful in approximation and compression
- ▶ fundamental in communication system design

The three take-home lessons today

- ▶ vector spaces are very general objects
- ▶ vector spaces are defined by their properties
- ▶ once you know the properties are satisfied, you can use all the tools for the space

Analogy #1: OOP

```
class Polygon(object):
    def __init__(self, num_sides, side_len=1, x=0, y=0):
        self.num_sides = num_sides
        self.side_len = side_len
        self.center = [x, y]

    def resize(self, factor):
        self.side_len *= factor

    def translate(self, x, y):
        self.center[0] += x
        self.center[1] += y

    def plot(self):
        ...
```

Analogy #1: OOP

```
class Triangle(Polygon):  
    def __init__(self):  
        super(Triangle, self).__init__(3)
```

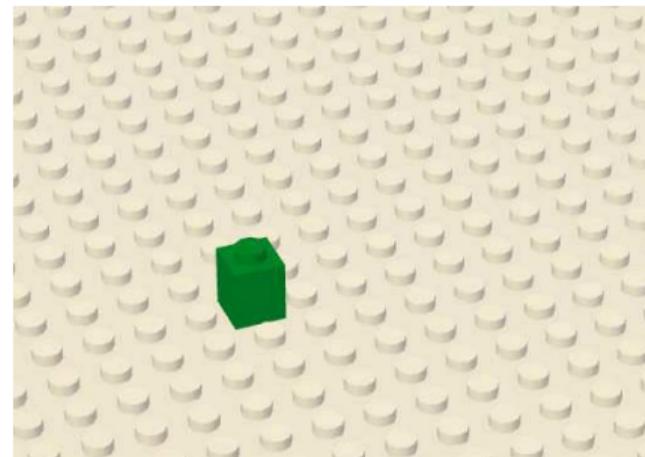
...

```
class Square(Polygon):  
    def __init__(self):  
        super(Square, self).__init__(4)
```

...

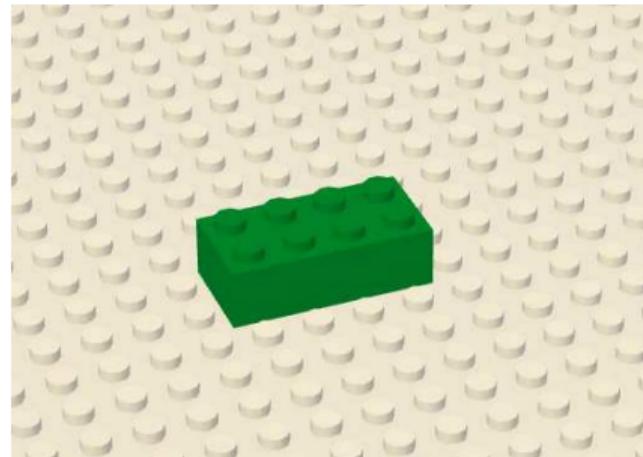
Analogy #2: LEGO

basic building block:



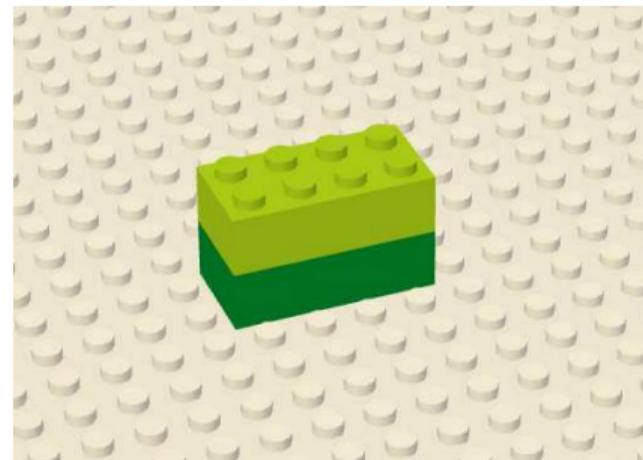
Analogy #2: LEGO

scaling (4x2):



Analogy #2: LEGO

adding:



vector spaces

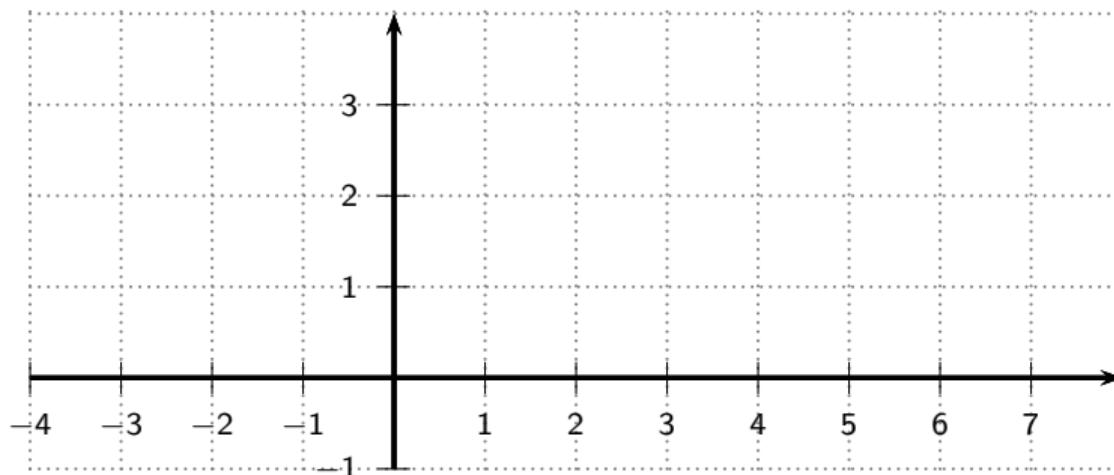
Graphical representation of a vector

Sometimes we can

$$\mathbb{R}^2 : \quad \mathbf{x} = [x_0 \quad x_1]^T$$

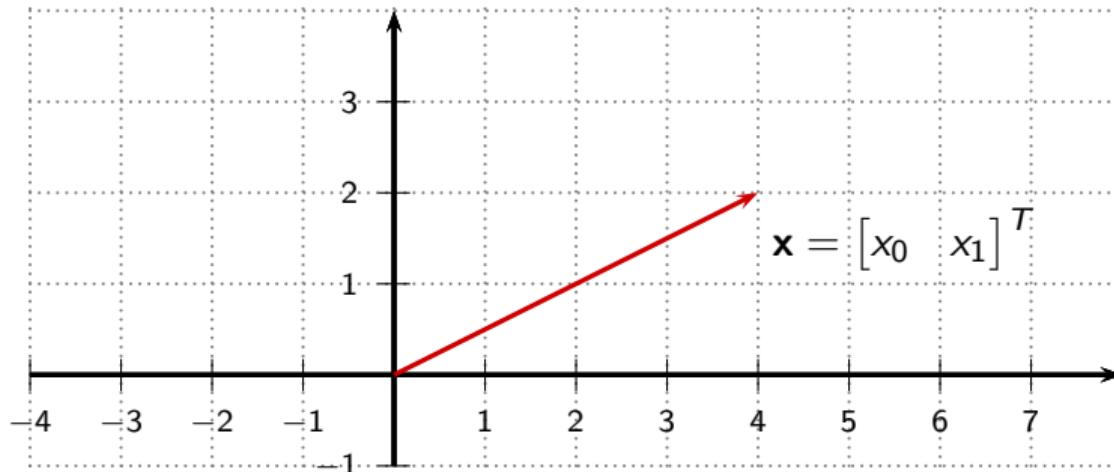
Sometimes we can

$$\mathbb{R}^2 : \quad \mathbf{x} = [x_0 \quad x_1]^T$$



Sometimes we can

$$\mathbb{R}^2 : \quad \mathbf{x} = [x_0 \quad x_1]^T$$

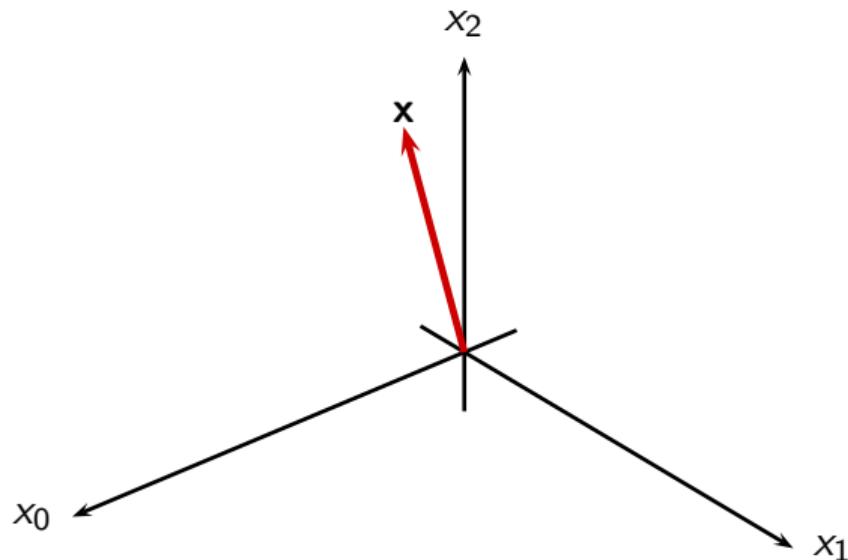


Sometimes we can

$$\mathbb{R}^3 : \quad \mathbf{x} = [x_0 \quad x_1 \quad x_2]^T$$

Sometimes we can

$$\mathbb{R}^3 : \quad \mathbf{x} = [x_0 \quad x_1 \quad x_2]^T$$



but most of the time we can't

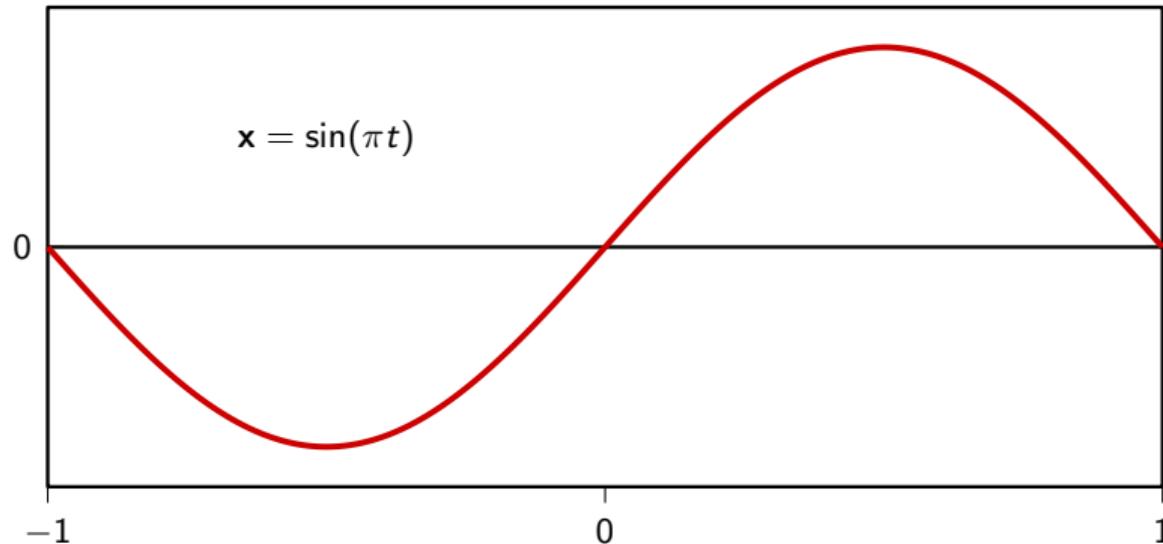
$$\mathbb{R}^N \text{ for } N > 3 : \quad \mathbf{x} = [x_0 \quad x_1 \quad \dots \quad x_{N-1}]^T$$

Sometimes we can

$$L_2([-1, 1]) : \quad \mathbf{x} = x(t) \in \mathbb{R}, \quad t \in [-1, 1]$$

Sometimes we can

$$L_2([-1, 1]) : \quad \mathbf{x} = x(t) \in \mathbb{R}, \quad t \in [-1, 1]$$



other times we can't

$$f : \mathbb{C} \rightarrow \mathbb{C}, \text{ analytic}$$

Vector spaces: operational definition

Ingredients:

- ▶ the set of vectors V
- ▶ a set of scalars (say \mathbb{C})

We need *at least* to be able to:

- ▶ resize vectors, i.e. multiply a vector by a scalar
- ▶ combine vectors together, i.e. sum them together

Vector spaces: operational definition

Ingredients:

- ▶ the set of vectors V
- ▶ a set of scalars (say \mathbb{C})

We need *at least* to be able to:

- ▶ resize vectors, i.e. multiply a vector by a scalar
- ▶ combine vectors together, i.e. sum them together

Formal properties of a vector space:

For $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and $\alpha, \beta \in \mathbb{C}$:

- ▶ $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$
- ▶ $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$
- ▶ $\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{y} + \alpha\mathbf{x}$
- ▶ $(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}$
- ▶ $\alpha(\beta\mathbf{x}) = (\alpha\beta)\mathbf{x}$
- ▶ $\exists \mathbf{0} \in V \quad | \quad \mathbf{x} + \mathbf{0} = \mathbf{0} + \mathbf{x} = \mathbf{x}$
- ▶ $\forall \mathbf{x} \in V \exists (-\mathbf{x}) \quad | \quad \mathbf{x} + (-\mathbf{x}) = 0$

Formal properties of a vector space:

For $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and $\alpha, \beta \in \mathbb{C}$:

- ▶ $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$
- ▶ $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$
- ▶ $\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{y} + \alpha\mathbf{x}$
- ▶ $(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}$
- ▶ $\alpha(\beta\mathbf{x}) = (\alpha\beta)\mathbf{x}$
- ▶ $\exists \mathbf{0} \in V \quad | \quad \mathbf{x} + \mathbf{0} = \mathbf{0} + \mathbf{x} = \mathbf{x}$
- ▶ $\forall \mathbf{x} \in V \exists (-\mathbf{x}) \quad | \quad \mathbf{x} + (-\mathbf{x}) = \mathbf{0}$

Formal properties of a vector space:

For $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and $\alpha, \beta \in \mathbb{C}$:

- ▶ $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$
- ▶ $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$
- ▶ $\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{y} + \alpha\mathbf{x}$
- ▶ $(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}$
- ▶ $\alpha(\beta\mathbf{x}) = (\alpha\beta)\mathbf{x}$
- ▶ $\exists \mathbf{0} \in V \quad | \quad \mathbf{x} + \mathbf{0} = \mathbf{0} + \mathbf{x} = \mathbf{x}$
- ▶ $\forall \mathbf{x} \in V \exists (-\mathbf{x}) \quad | \quad \mathbf{x} + (-\mathbf{x}) = \mathbf{0}$

Formal properties of a vector space:

For $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and $\alpha, \beta \in \mathbb{C}$:

- ▶ $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$
- ▶ $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$
- ▶ $\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{y} + \alpha\mathbf{x}$
- ▶ $(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}$
- ▶ $\alpha(\beta\mathbf{x}) = (\alpha\beta)\mathbf{x}$
- ▶ $\exists \mathbf{0} \in V \quad | \quad \mathbf{x} + \mathbf{0} = \mathbf{0} + \mathbf{x} = \mathbf{x}$
- ▶ $\forall \mathbf{x} \in V \exists (-\mathbf{x}) \quad | \quad \mathbf{x} + (-\mathbf{x}) = \mathbf{0}$

Formal properties of a vector space:

For $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and $\alpha, \beta \in \mathbb{C}$:

- ▶ $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$
- ▶ $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$
- ▶ $\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{y} + \alpha\mathbf{x}$
- ▶ $(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}$
- ▶ $\alpha(\beta\mathbf{x}) = (\alpha\beta)\mathbf{x}$
- ▶ $\exists \mathbf{0} \in V \quad | \quad \mathbf{x} + \mathbf{0} = \mathbf{0} + \mathbf{x} = \mathbf{x}$
- ▶ $\forall \mathbf{x} \in V \exists (-\mathbf{x}) \quad | \quad \mathbf{x} + (-\mathbf{x}) = \mathbf{0}$

Formal properties of a vector space:

For $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and $\alpha, \beta \in \mathbb{C}$:

- ▶ $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$
- ▶ $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$
- ▶ $\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{y} + \alpha\mathbf{x}$
- ▶ $(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}$
- ▶ $\alpha(\beta\mathbf{x}) = (\alpha\beta)\mathbf{x}$
- ▶ $\exists \mathbf{0} \in V \quad | \quad \mathbf{x} + \mathbf{0} = \mathbf{0} + \mathbf{x} = \mathbf{x}$
- ▶ $\forall \mathbf{x} \in V \exists (-\mathbf{x}) \quad | \quad \mathbf{x} + (-\mathbf{x}) = 0$

Formal properties of a vector space:

For $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and $\alpha, \beta \in \mathbb{C}$:

- ▶ $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$
- ▶ $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$
- ▶ $\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{y} + \alpha\mathbf{x}$
- ▶ $(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}$
- ▶ $\alpha(\beta\mathbf{x}) = (\alpha\beta)\mathbf{x}$
- ▶ $\exists \mathbf{0} \in V \quad | \quad \mathbf{x} + \mathbf{0} = \mathbf{0} + \mathbf{x} = \mathbf{x}$
- ▶ $\forall \mathbf{x} \in V \exists (-\mathbf{x}) \quad | \quad \mathbf{x} + (-\mathbf{x}) = \mathbf{0}$

Vector space example: \mathbb{R}^N

$$\mathbf{x} = [x_0 \quad x_1 \quad \dots \quad x_{N-1}]^T$$

$$\mathbf{y} = [y_0 \quad y_1 \quad \dots \quad y_{N-1}]^T$$

$$\alpha\mathbf{x} = [\alpha x_0 \quad \alpha x_1 \quad \dots \quad \alpha x_{N-1}]^T$$

$$\mathbf{x} + \mathbf{y} = [x_0 + y_0 \quad x_1 + y_1 \quad \dots \quad x_{N-1} + y_{N-1}]^T$$

Vector space example: \mathbb{R}^N

$$\mathbf{x} = [x_0 \quad x_1 \quad \dots \quad x_{N-1}]^T$$

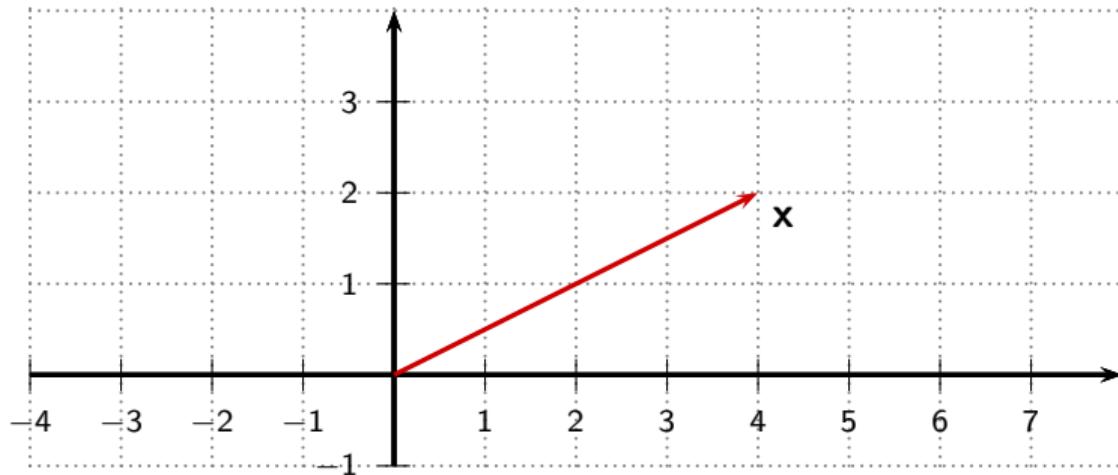
$$\mathbf{y} = [y_0 \quad y_1 \quad \dots \quad y_{N-1}]^T$$

$$\alpha\mathbf{x} = [\alpha x_0 \quad \alpha x_1 \quad \dots \quad \alpha x_{N-1}]^T$$

$$\mathbf{x} + \mathbf{y} = [x_0 + y_0 \quad x_1 + y_1 \quad \dots \quad x_{N-1} + y_{N-1}]^T$$

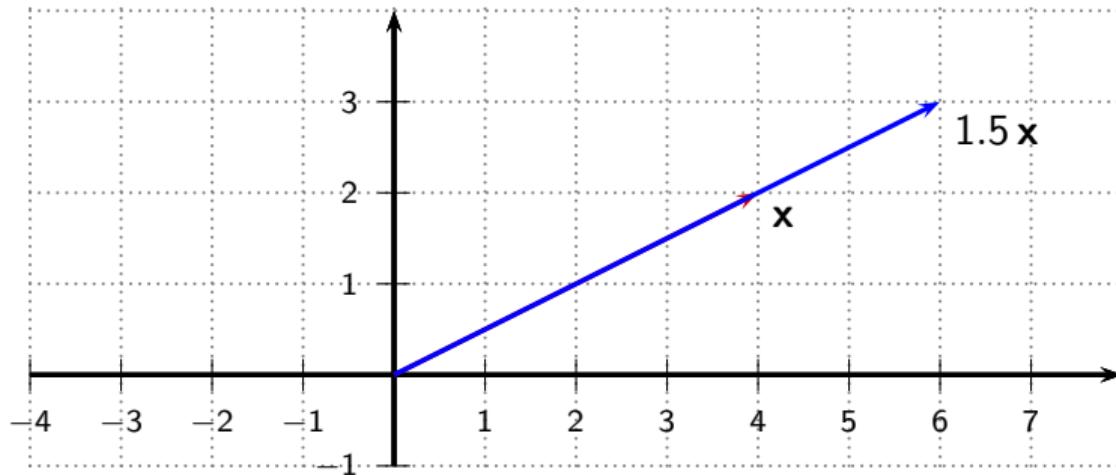
Scalar multiplication in \mathbb{R}^2

$$\alpha \mathbf{x} = [\alpha x_0 \quad \alpha x_1]^T$$



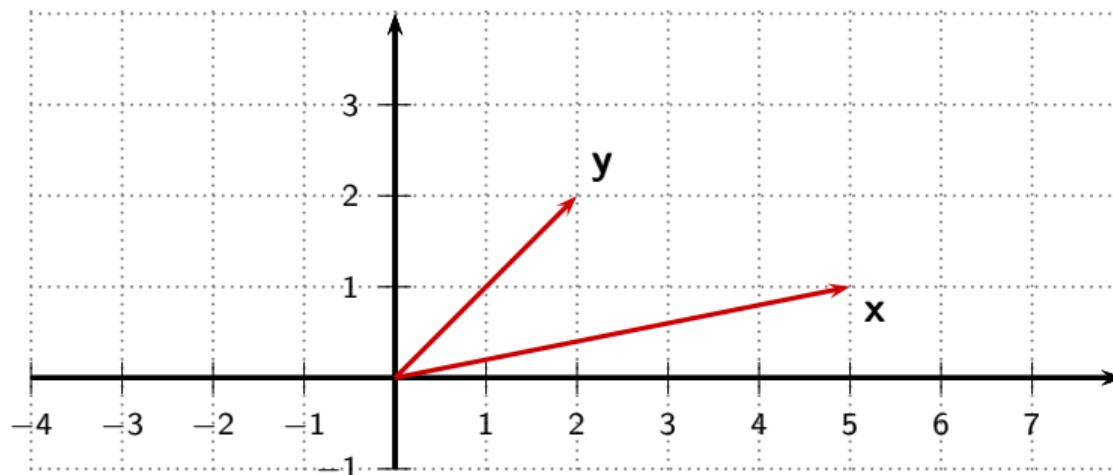
Scalar multiplication in \mathbb{R}^2

$$\alpha \mathbf{x} = [\alpha x_0 \quad \alpha x_1]^T$$



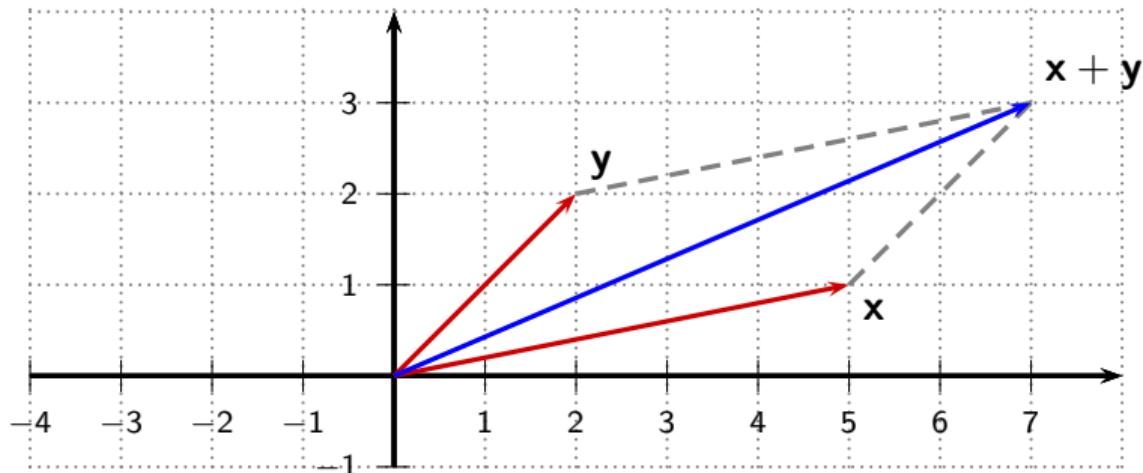
Addition in \mathbb{R}^2

$$\mathbf{x} + \mathbf{y} = [x_0 + y_0 \quad x_1 + y_1]^T$$



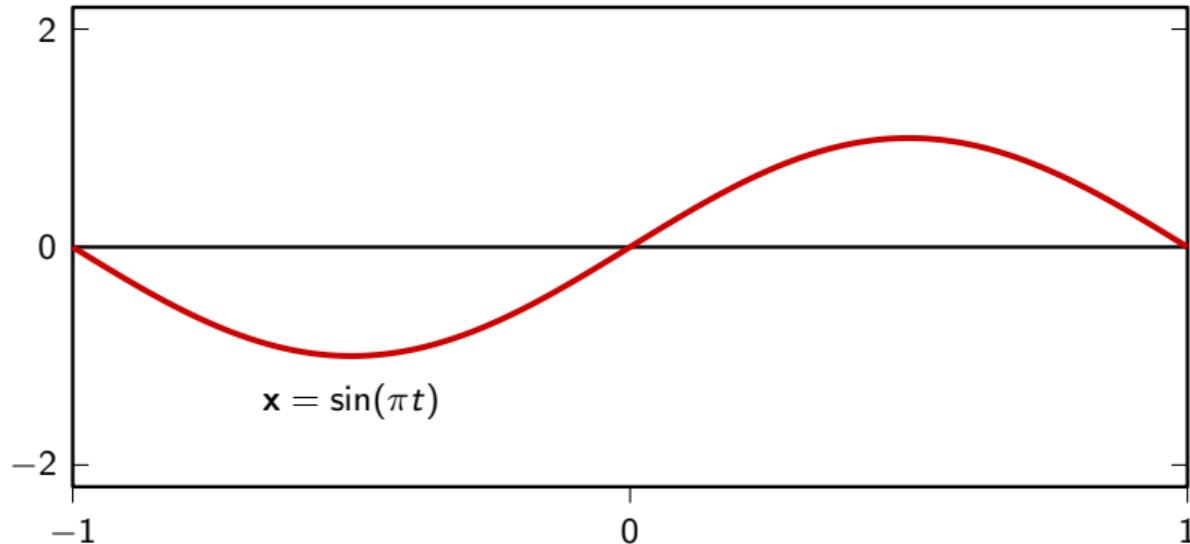
Addition in \mathbb{R}^2

$$\mathbf{x} + \mathbf{y} = [x_0 + y_0 \quad x_1 + y_1]^T$$



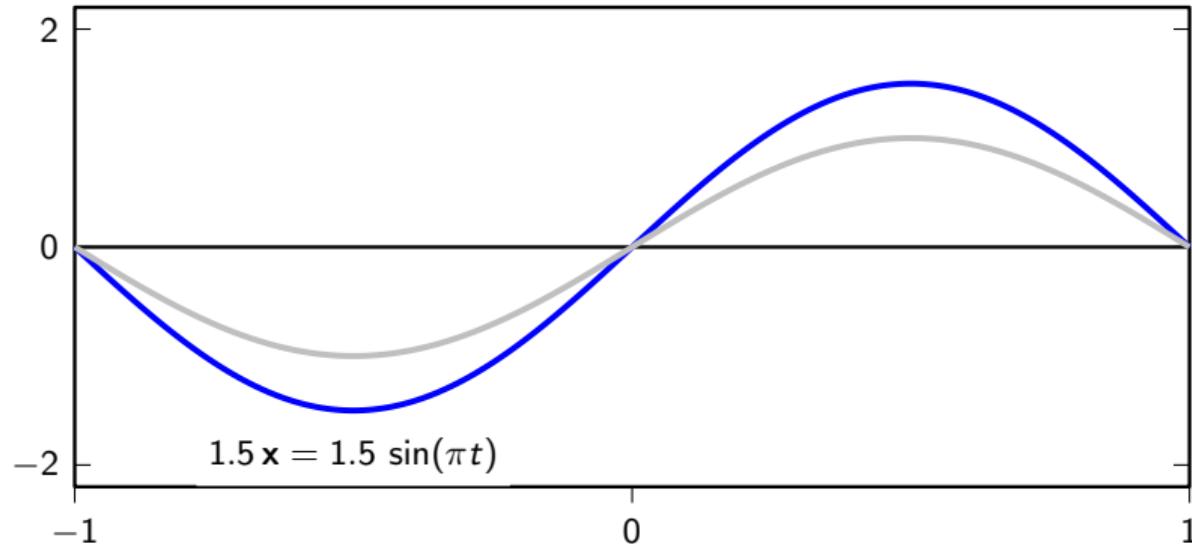
Scalar multiplication in $L_2[-1, 1]$

$$\alpha \mathbf{x} = \alpha x(t)$$



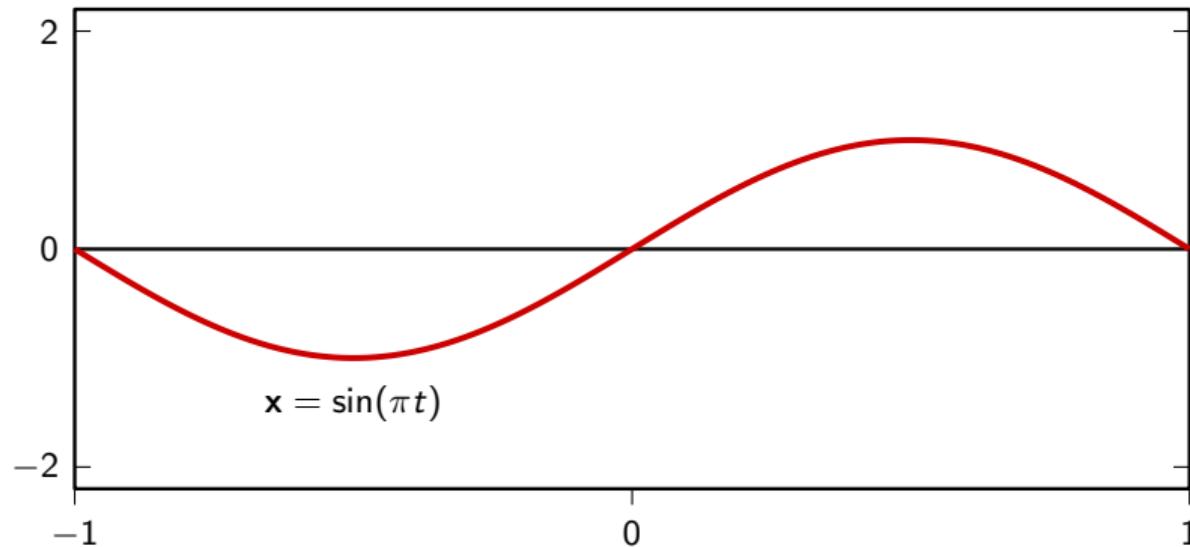
Scalar multiplication in $L_2[-1, 1]$

$$\alpha \mathbf{x} = \alpha x(t)$$



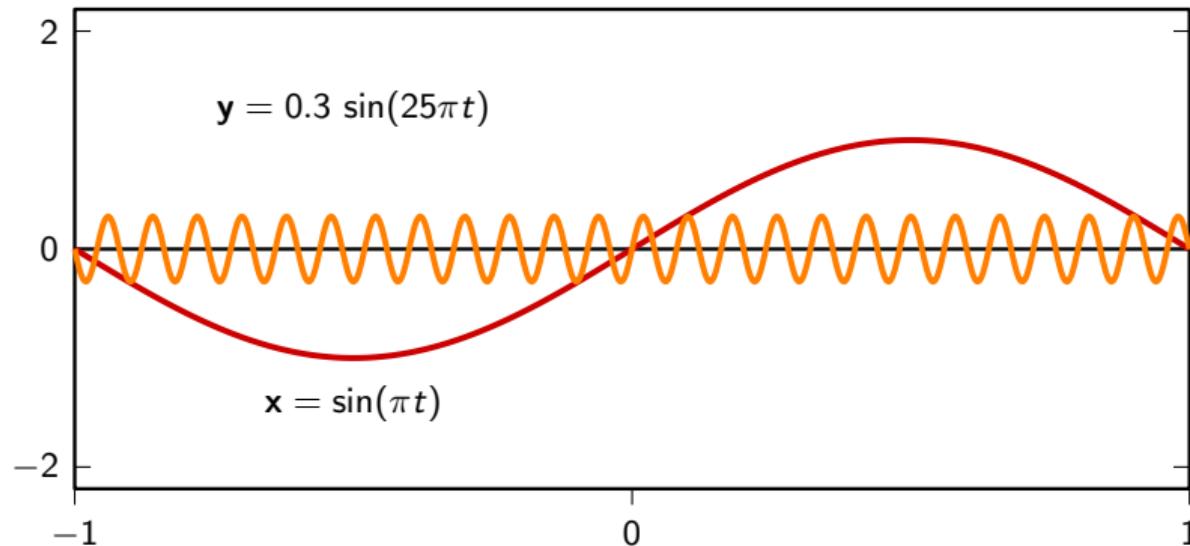
Addition in $L_2[-1, 1]$

$$\mathbf{x} + \mathbf{y} = x(t) + y(t)$$



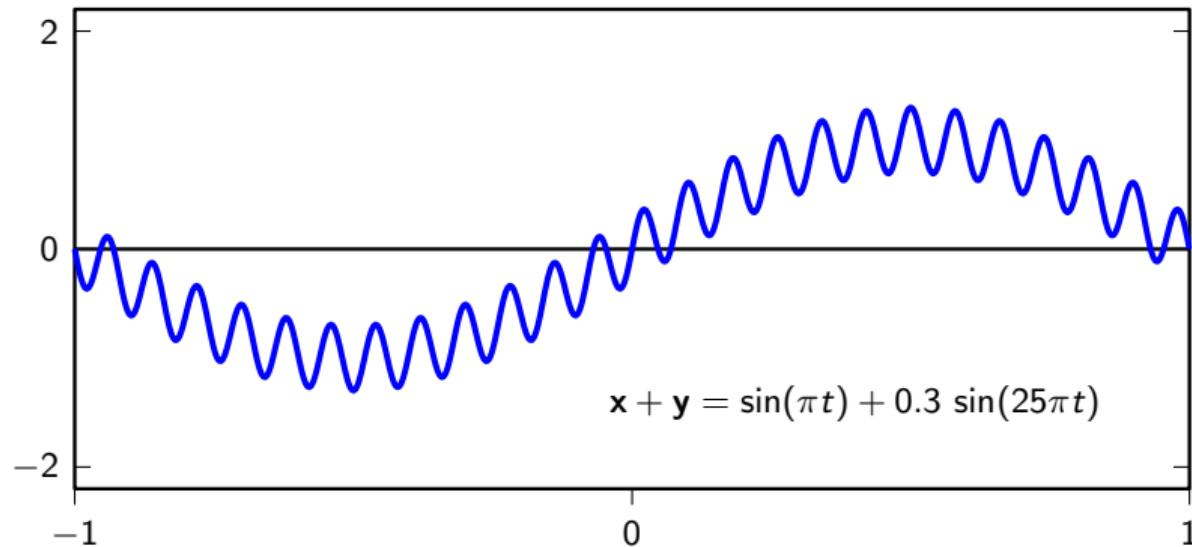
Addition in $L_2[-1, 1]$

$$\mathbf{x} + \mathbf{y} = x(t) + y(t)$$



Addition in $L_2[-1, 1]$

$$\mathbf{x} + \mathbf{y} = x(t) + y(t)$$



Vector spaces: we need something more

- ▶ the set of vectors V
- ▶ a set of scalars (say \mathbb{C})
- ▶ scalar multiplication
- ▶ addition

We need something to measure and compare:
inner product (aka dot product)

Vector spaces: we need something more

- ▶ the set of vectors V
- ▶ a set of scalars (say \mathbb{C})
- ▶ scalar multiplication
- ▶ addition

We need something to measure and compare:
inner product (aka dot product)

inner product spaces

Inner product

$$\langle \cdot, \cdot \rangle : V \times V \rightarrow \mathbb{C}$$

- ▶ measure of similarity between vectors
- ▶ inner product is zero? vectors are *orthogonal* (maximally different)

Formal properties of the inner product

For $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and $\alpha \in \mathbb{C}$:

- ▶ $\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle$
- ▶ $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle^*$
- ▶ $\langle \alpha \mathbf{x}, \mathbf{y} \rangle = \alpha^* \langle \mathbf{x}, \mathbf{y} \rangle$
 $\langle \mathbf{x}, \alpha \mathbf{y} \rangle = \alpha \langle \mathbf{x}, \mathbf{y} \rangle$
- ▶ $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$
- ▶ $\langle \mathbf{x}, \mathbf{x} \rangle = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$

Formal properties of the inner product

For $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and $\alpha \in \mathbb{C}$:

- ▶ $\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle$
- ▶ $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle^*$
- ▶ $\langle \alpha \mathbf{x}, \mathbf{y} \rangle = \alpha^* \langle \mathbf{x}, \mathbf{y} \rangle$
 $\langle \mathbf{x}, \alpha \mathbf{y} \rangle = \alpha \langle \mathbf{x}, \mathbf{y} \rangle$
- ▶ $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$
- ▶ $\langle \mathbf{x}, \mathbf{x} \rangle = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$

Formal properties of the inner product

For $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and $\alpha \in \mathbb{C}$:

- ▶ $\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle$
- ▶ $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle^*$
- ▶ $\langle \alpha \mathbf{x}, \mathbf{y} \rangle = \alpha^* \langle \mathbf{x}, \mathbf{y} \rangle$
 $\langle \mathbf{x}, \alpha \mathbf{y} \rangle = \alpha \langle \mathbf{x}, \mathbf{y} \rangle$
- ▶ $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$
- ▶ $\langle \mathbf{x}, \mathbf{x} \rangle = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$

Formal properties of the inner product

For $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and $\alpha \in \mathbb{C}$:

- ▶ $\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle$
- ▶ $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle^*$
- ▶ $\langle \alpha \mathbf{x}, \mathbf{y} \rangle = \alpha^* \langle \mathbf{x}, \mathbf{y} \rangle$
 $\langle \mathbf{x}, \alpha \mathbf{y} \rangle = \alpha \langle \mathbf{x}, \mathbf{y} \rangle$
- ▶ $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$
- ▶ $\langle \mathbf{x}, \mathbf{x} \rangle = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$

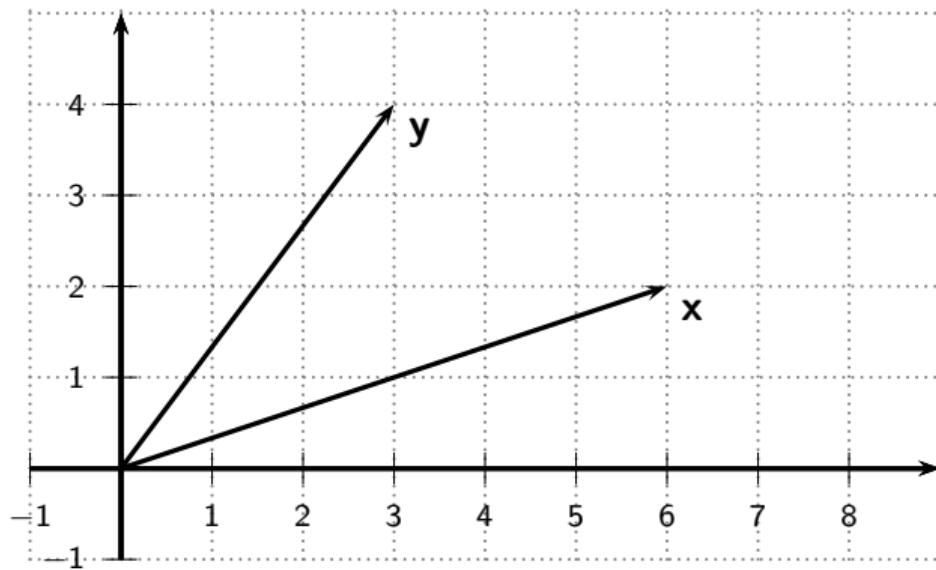
Formal properties of the inner product

For $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ and $\alpha \in \mathbb{C}$:

- ▶ $\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle$
- ▶ $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle^*$
- ▶ $\langle \alpha \mathbf{x}, \mathbf{y} \rangle = \alpha^* \langle \mathbf{x}, \mathbf{y} \rangle$
 $\langle \mathbf{x}, \alpha \mathbf{y} \rangle = \alpha \langle \mathbf{x}, \mathbf{y} \rangle$
- ▶ $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$
- ▶ $\langle \mathbf{x}, \mathbf{x} \rangle = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$

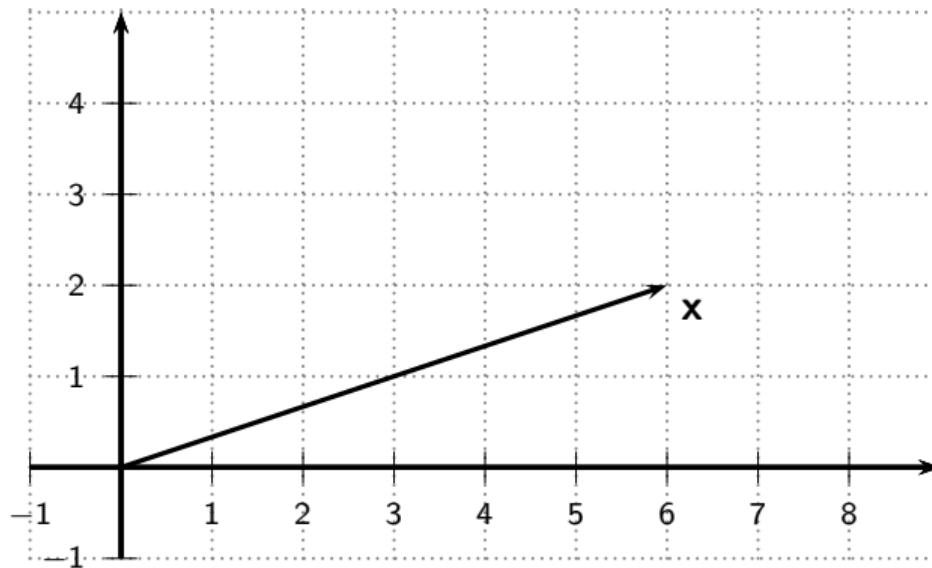
Inner product in \mathbb{R}^2

$$\langle \mathbf{x}, \mathbf{y} \rangle = x_0 y_0 + x_1 y_1$$



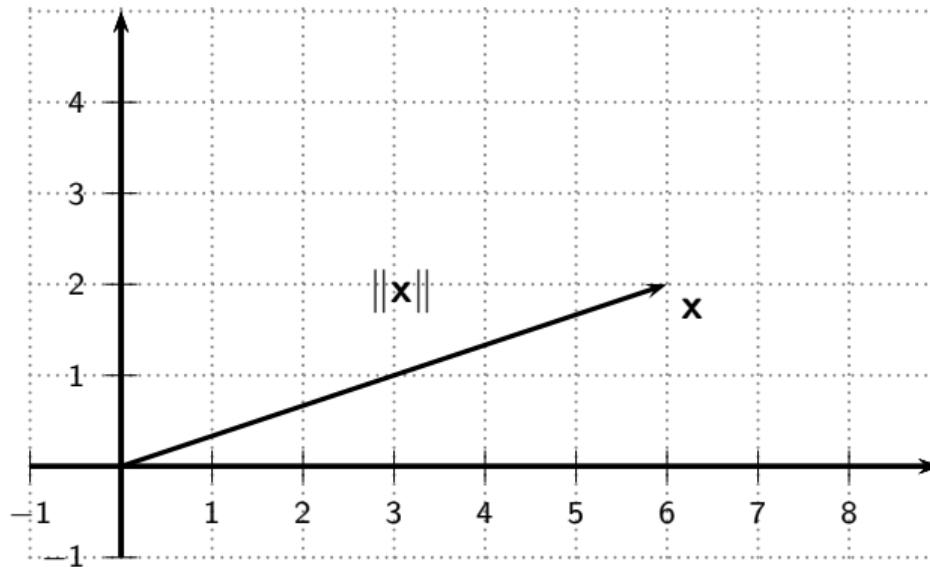
Inner product in \mathbb{R}^2 : the norm

$$\langle \mathbf{x}, \mathbf{x} \rangle = x_0^2 + x_1^2$$



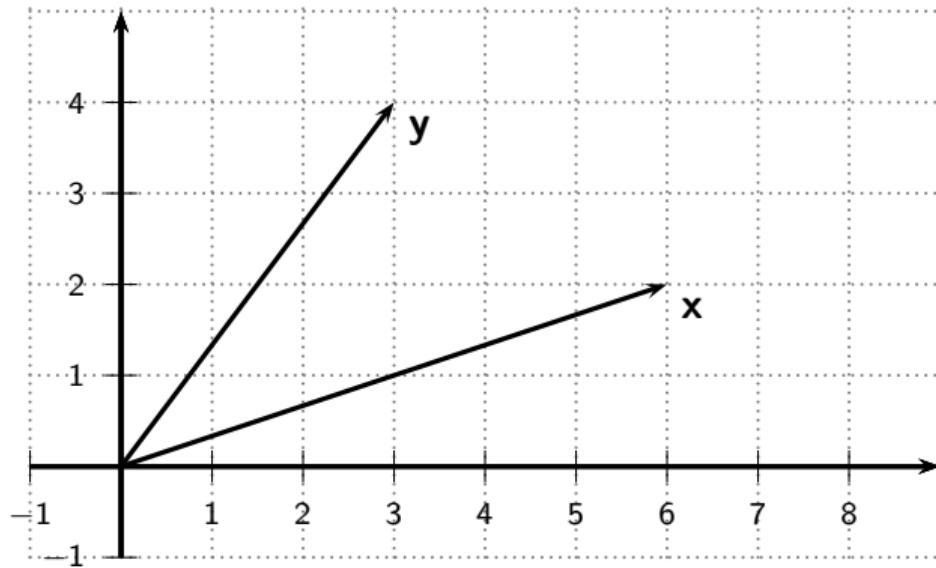
Inner product in \mathbb{R}^2 : the norm

$$\langle \mathbf{x}, \mathbf{x} \rangle = x_0^2 + x_1^2 = \|\mathbf{x}\|^2$$



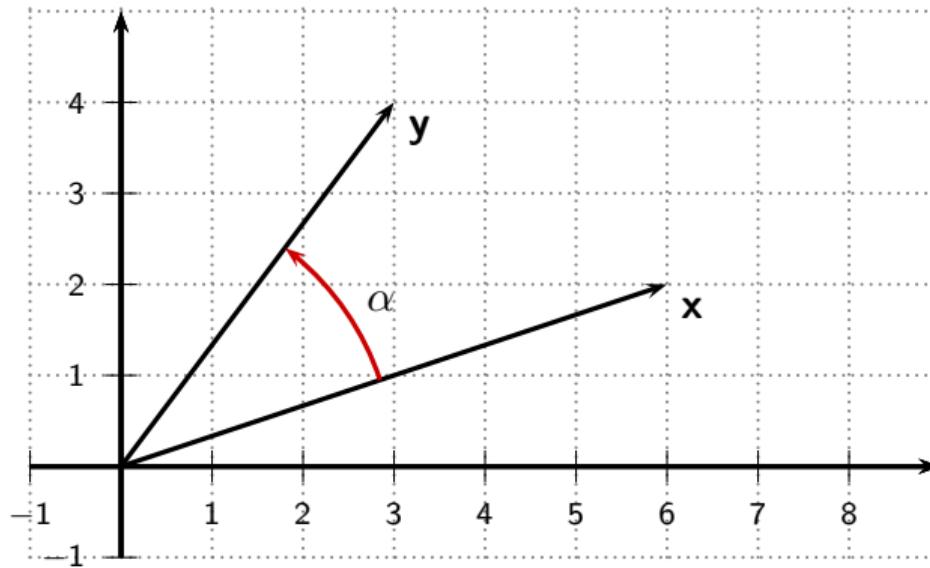
Inner product in \mathbb{R}^2

$$\langle \mathbf{x}, \mathbf{y} \rangle = x_0 y_0 + x_1 y_1$$



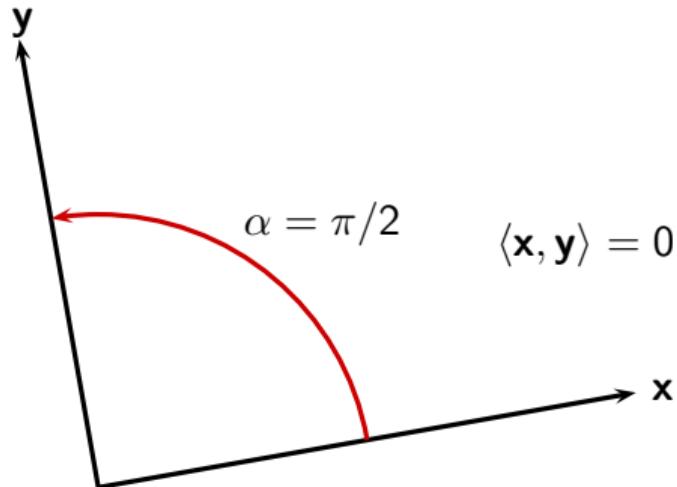
Inner product in \mathbb{R}^2

$$\langle \mathbf{x}, \mathbf{y} \rangle = x_0 y_0 + x_1 y_1 = \|\mathbf{x}\| \|\mathbf{y}\| \cos \alpha$$



Inner product in \mathbb{R}^2 : orthogonality

$$\langle \mathbf{x}, \mathbf{y} \rangle = x_0 y_0 + x_1 y_1 = \|\mathbf{x}\| \|\mathbf{y}\| \cos \alpha$$

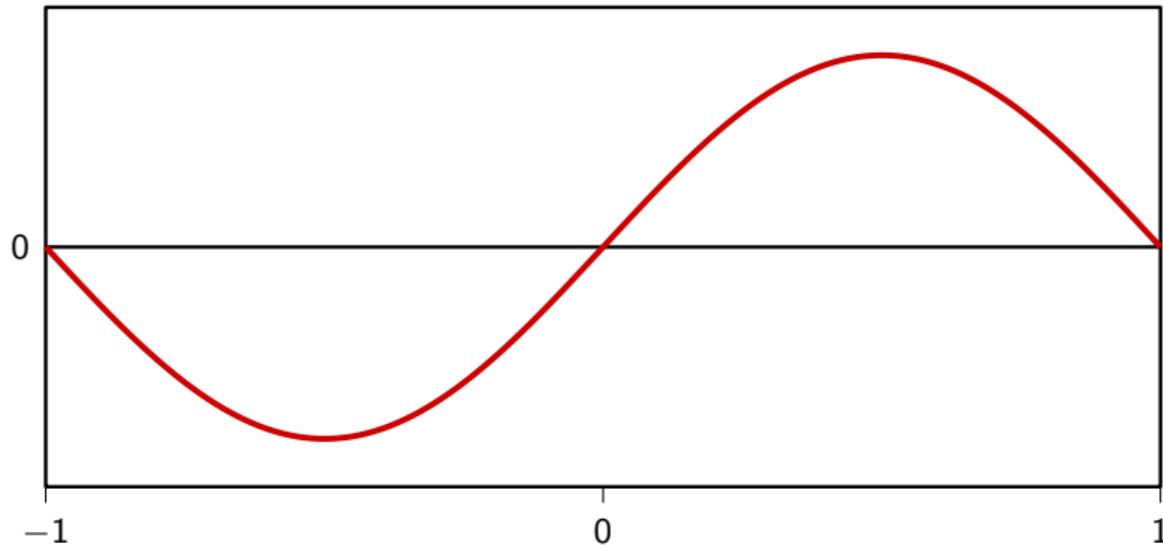


Inner product in $L_2[-1, 1]$

$$\langle \mathbf{x}, \mathbf{y} \rangle = \int_{-1}^1 x(t)y(t) dt$$

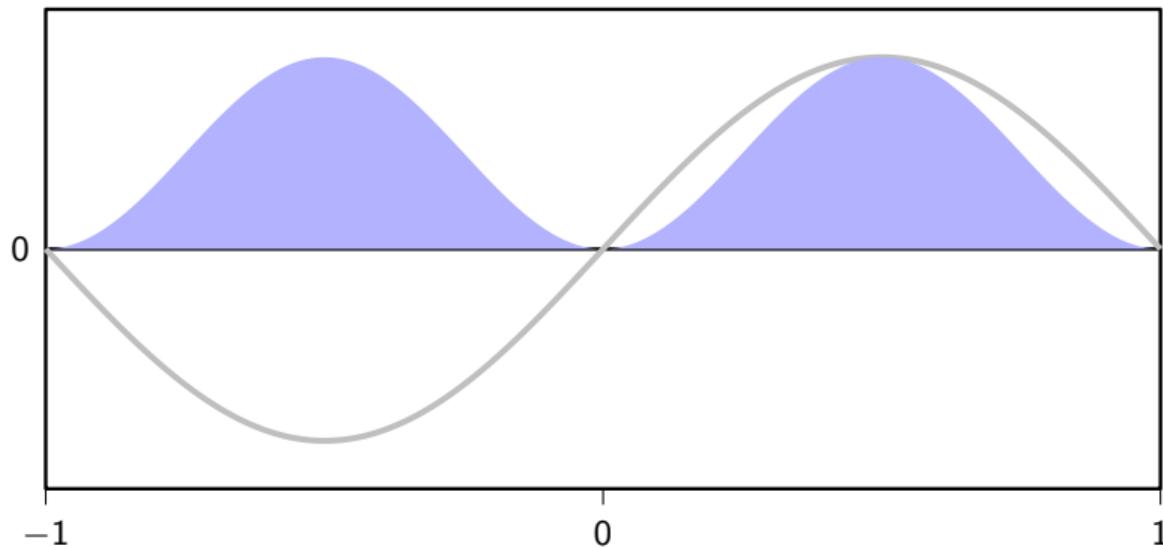
Inner product in $L_2[-1, 1]$: the norm

$$\langle \mathbf{x}, \mathbf{x} \rangle = \|\mathbf{x}\|^2 = \int_{-1}^1 \sin^2(\pi t) dt = 1$$



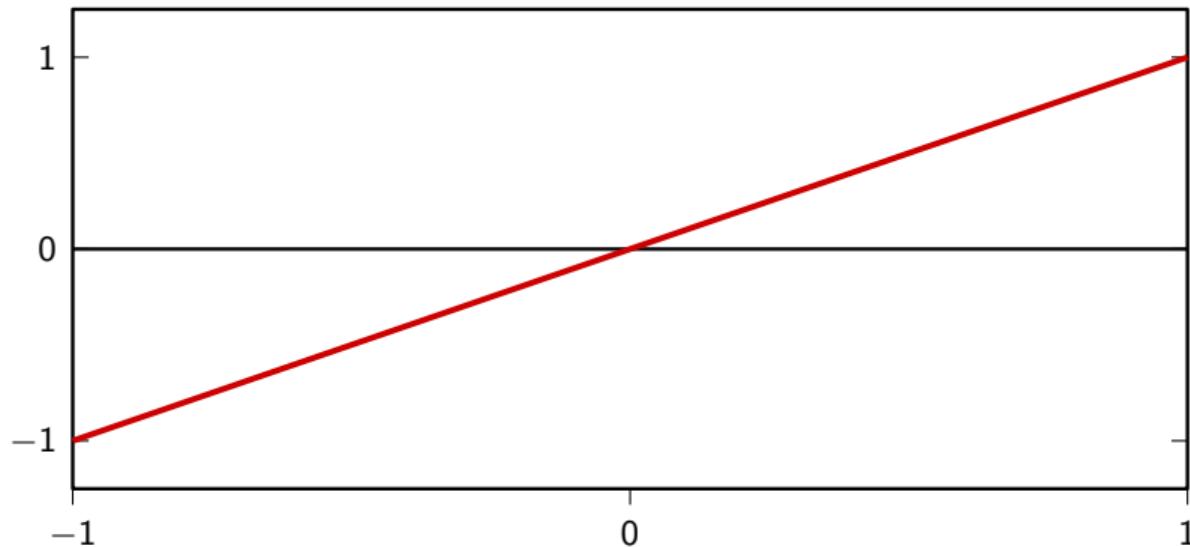
Inner product in $L_2[-1, 1]$: the norm

$$\langle \mathbf{x}, \mathbf{x} \rangle = \|\mathbf{x}\|^2 = \int_{-1}^1 \sin^2(\pi t) dt = 1$$



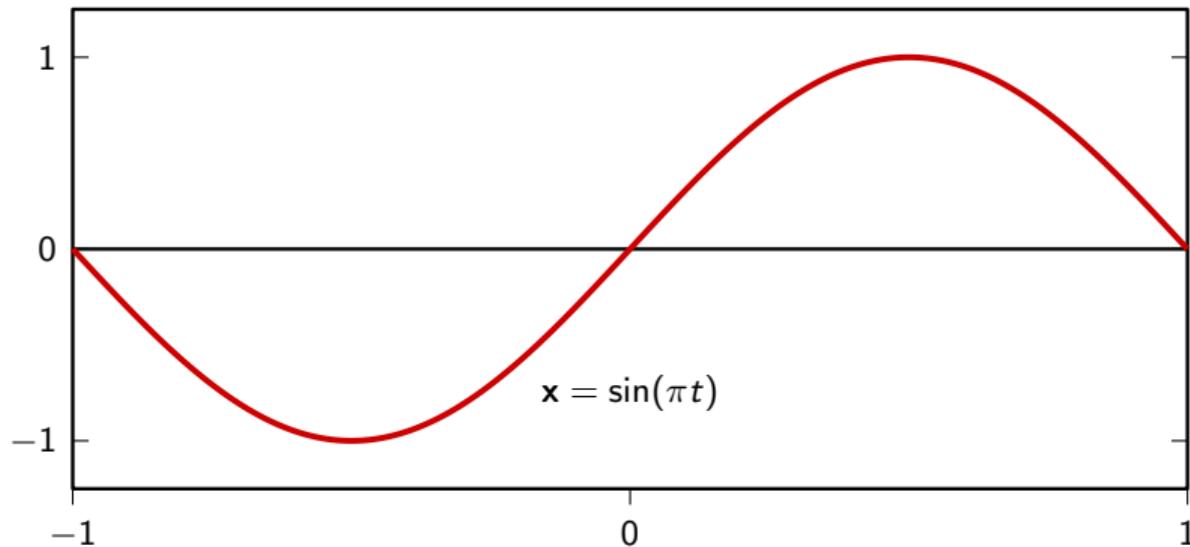
Inner product in $L_2[-1, 1]$: the norm

$$\|\mathbf{y}\|^2 = \int_{-1}^1 t^2 dt = 2/3$$



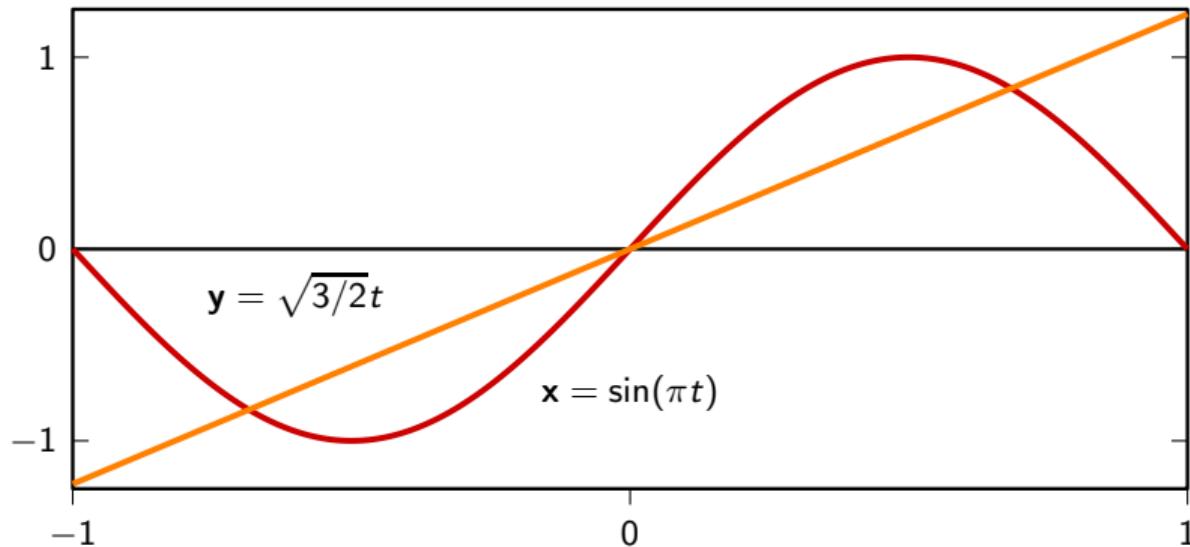
Inner product in $L_2[-1, 1]$

$$\langle \mathbf{x}, \mathbf{y} \rangle = \int_{-1}^1 x(t)y(t)dt$$



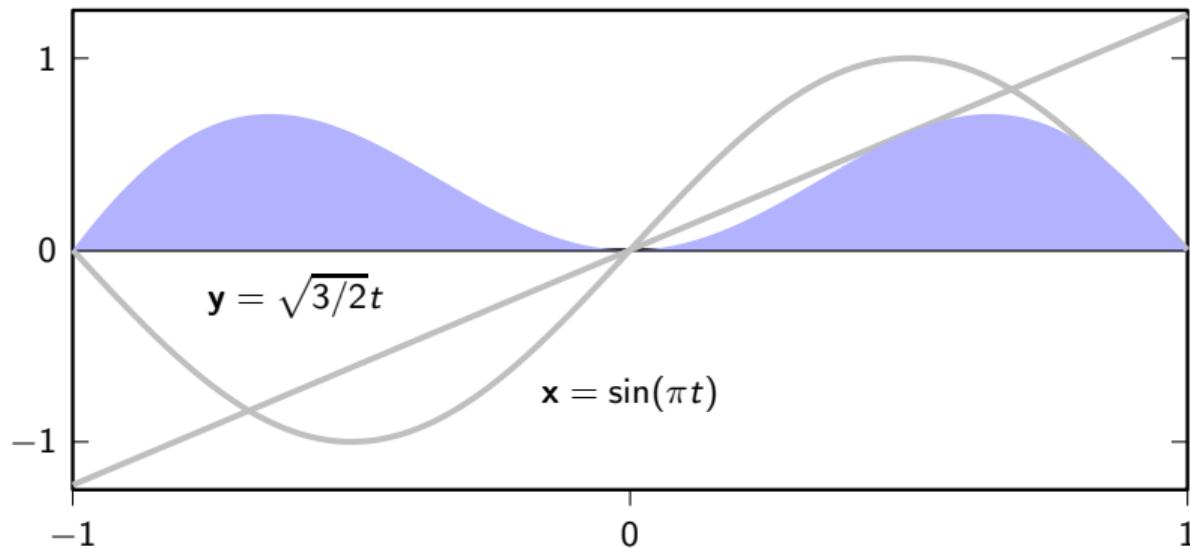
Inner product in $L_2[-1, 1]$

$$\langle \mathbf{x}, \mathbf{y} \rangle = \int_{-1}^1 x(t)y(t)dt$$



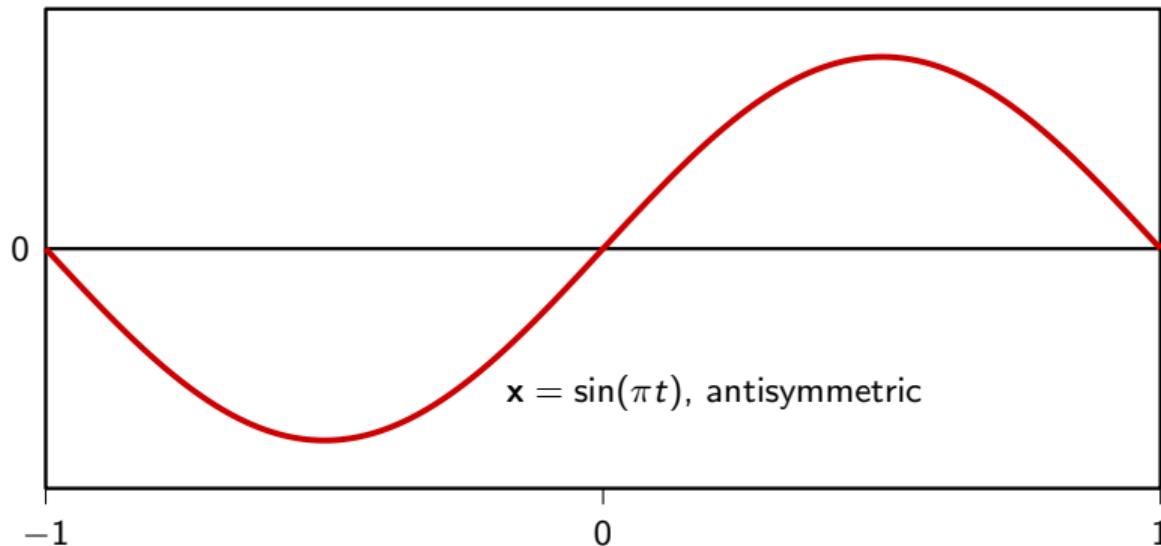
Inner product in $L_2[-1, 1]$

$$\langle \mathbf{x}, \mathbf{y} \rangle = \int_{-1}^1 \sqrt{3/2} t \sin(\pi t) dt = (2/\pi) \sqrt{3/2} \approx 0.78 \approx \cos(38.7^\circ)$$



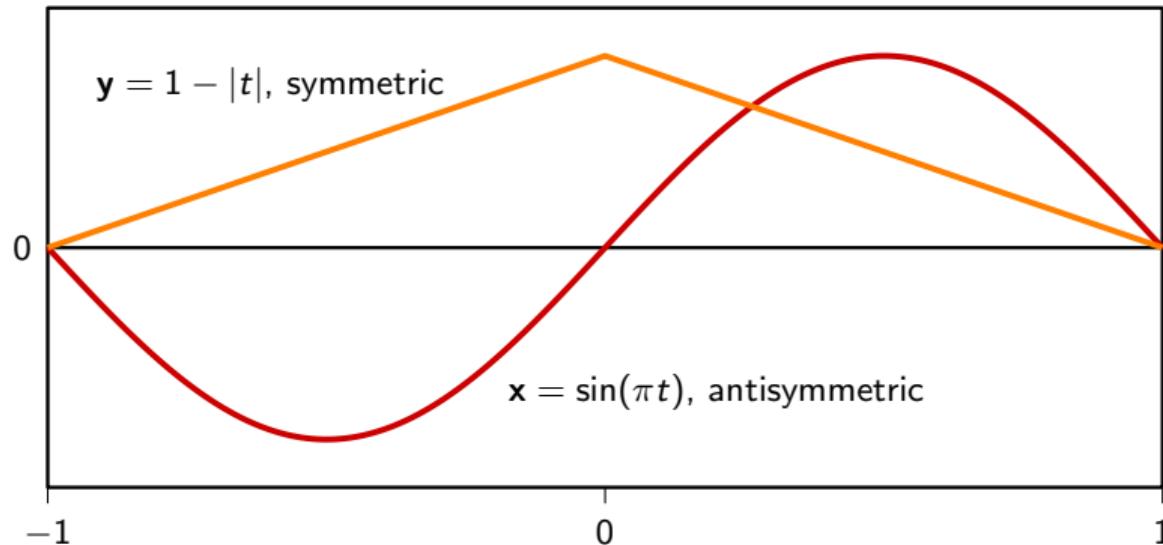
Inner product in $L_2[-1, 1]$

\mathbf{x}, \mathbf{y} from orthogonal subspaces:



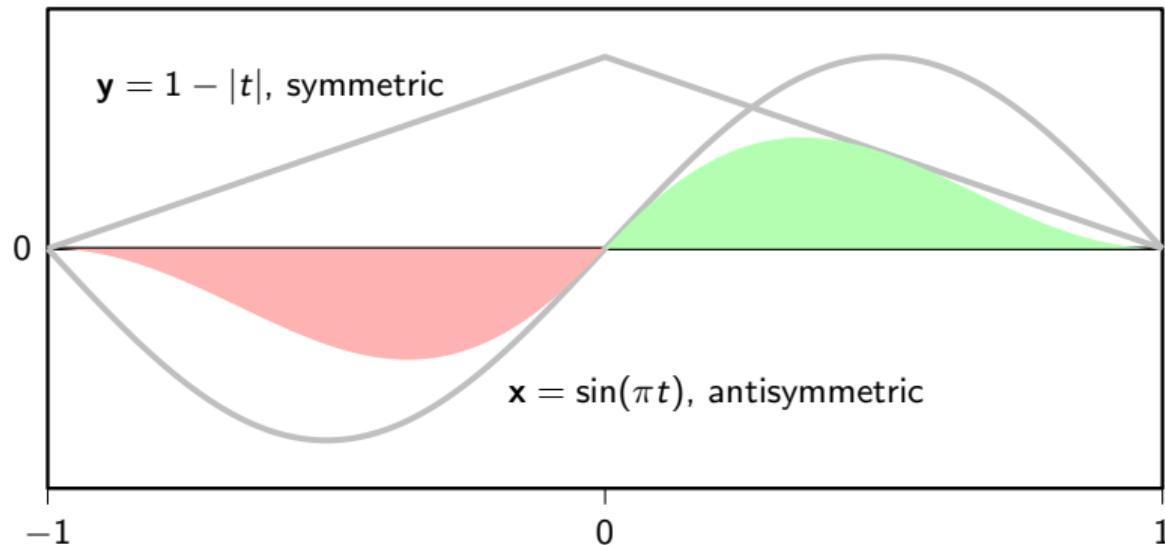
Inner product in $L_2[-1, 1]$

\mathbf{x}, \mathbf{y} from orthogonal subspaces:



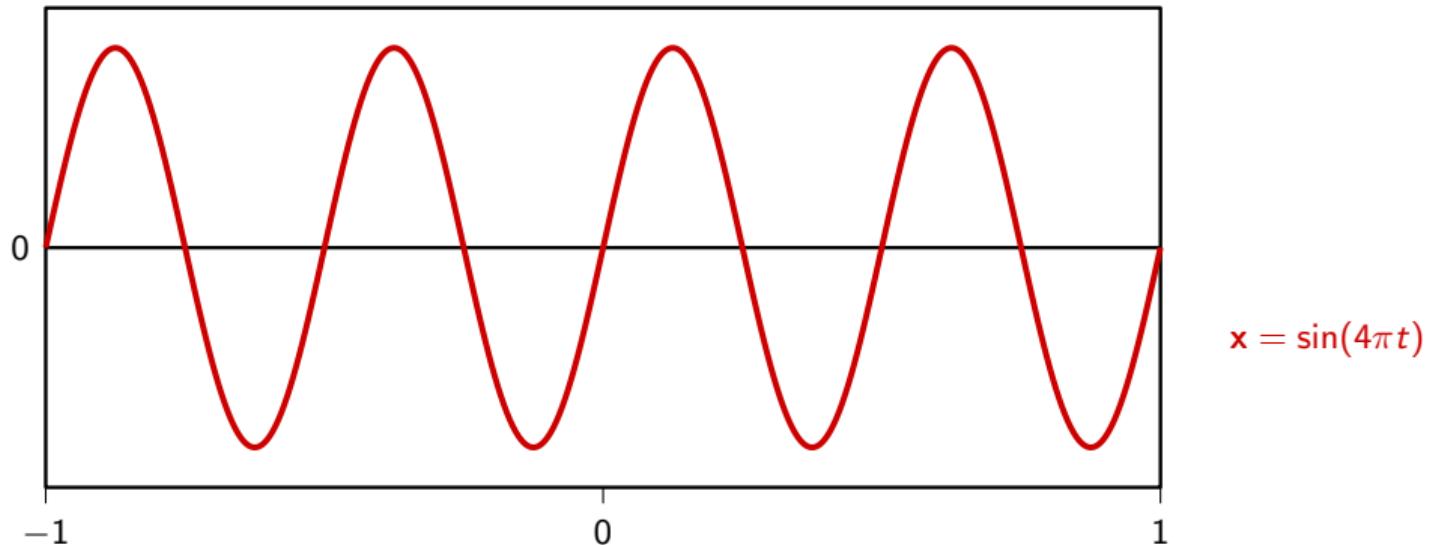
Inner product in $L_2[-1, 1]$

\mathbf{x}, \mathbf{y} from orthogonal subspaces: $\langle \mathbf{x}, \mathbf{y} \rangle = 0$



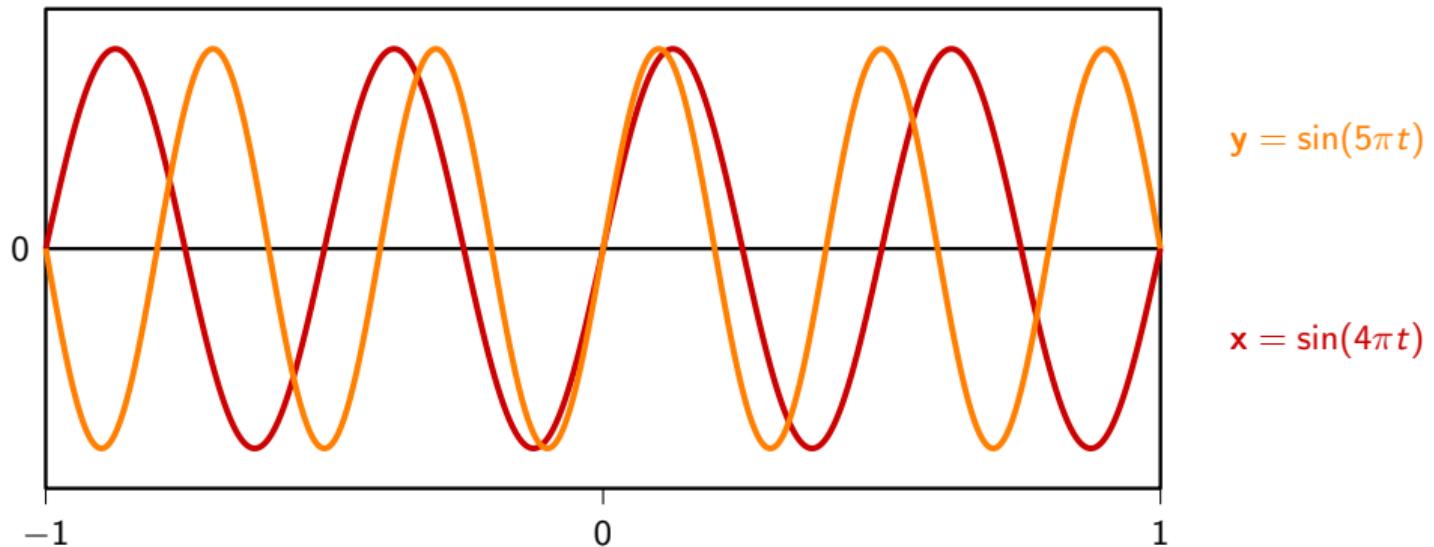
Inner product in $L_2[-1, 1]$

sinusoids with frequencies integer multiples of a fundamental



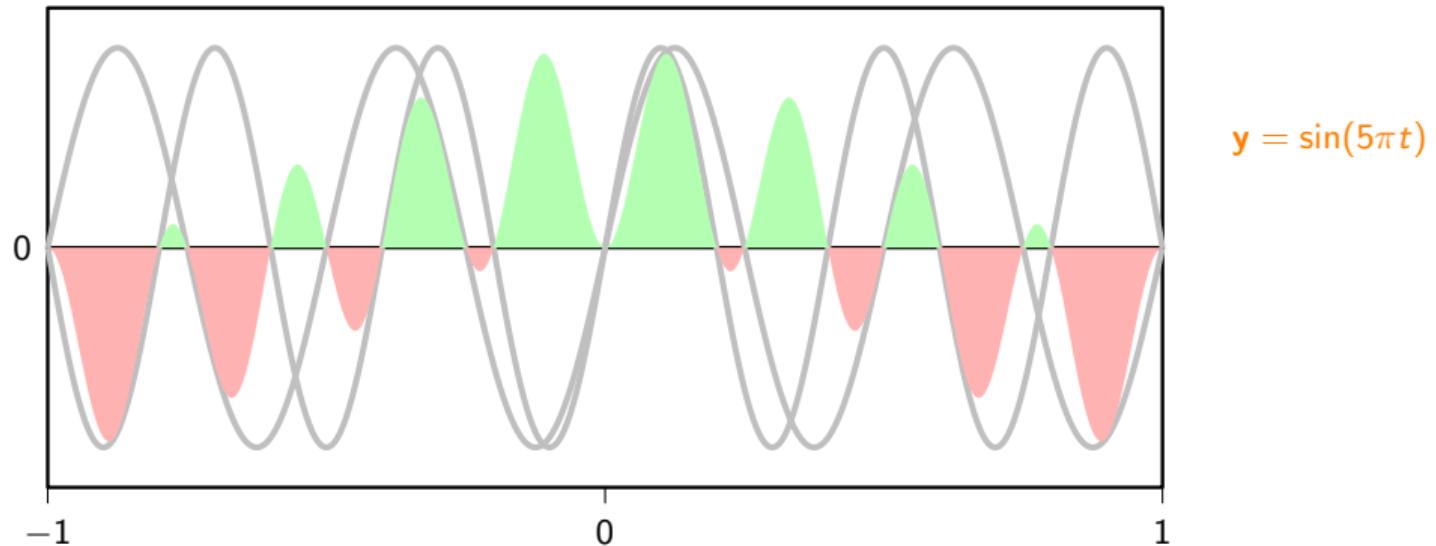
Inner product in $L_2[-1, 1]$

sinusoids with frequencies integer multiples of a fundamental



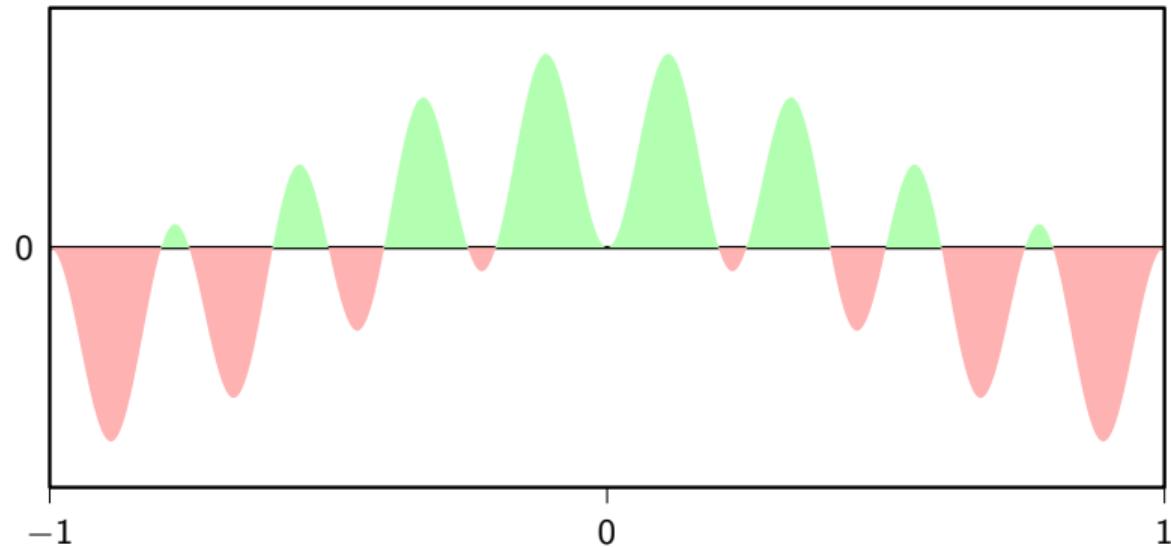
Inner product in $L_2[-1, 1]$

sinusoids with frequencies integer multiples of a fundamental



Inner product in $L_2[-1, 1]$

sinusoids with frequencies integer multiples of a fundamental



Norm vs Distance

- ▶ inner product defines a norm: $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$
- ▶ norm defines a distance: $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$

Norm vs Distance

- ▶ inner product defines a norm: $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$
- ▶ norm defines a distance: $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$

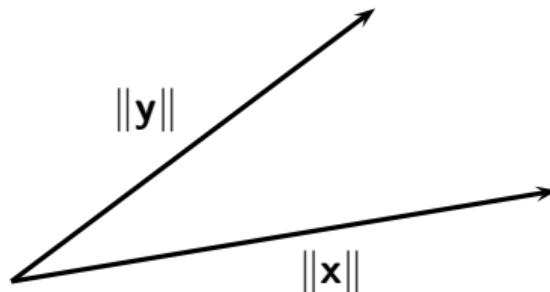
Norm and distance in \mathbb{R}^2

$$\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = \sqrt{x_0^2 + x_1^2}$$



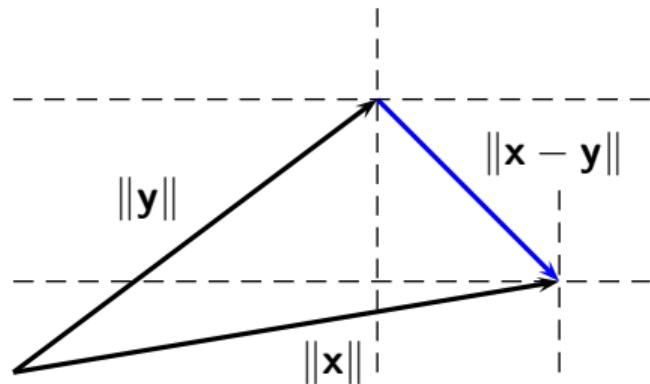
Norm and distance in \mathbb{R}^2

$$\|\mathbf{y}\| = \sqrt{\langle \mathbf{y}, \mathbf{y} \rangle} = \sqrt{y_0^2 + y_1^2}$$



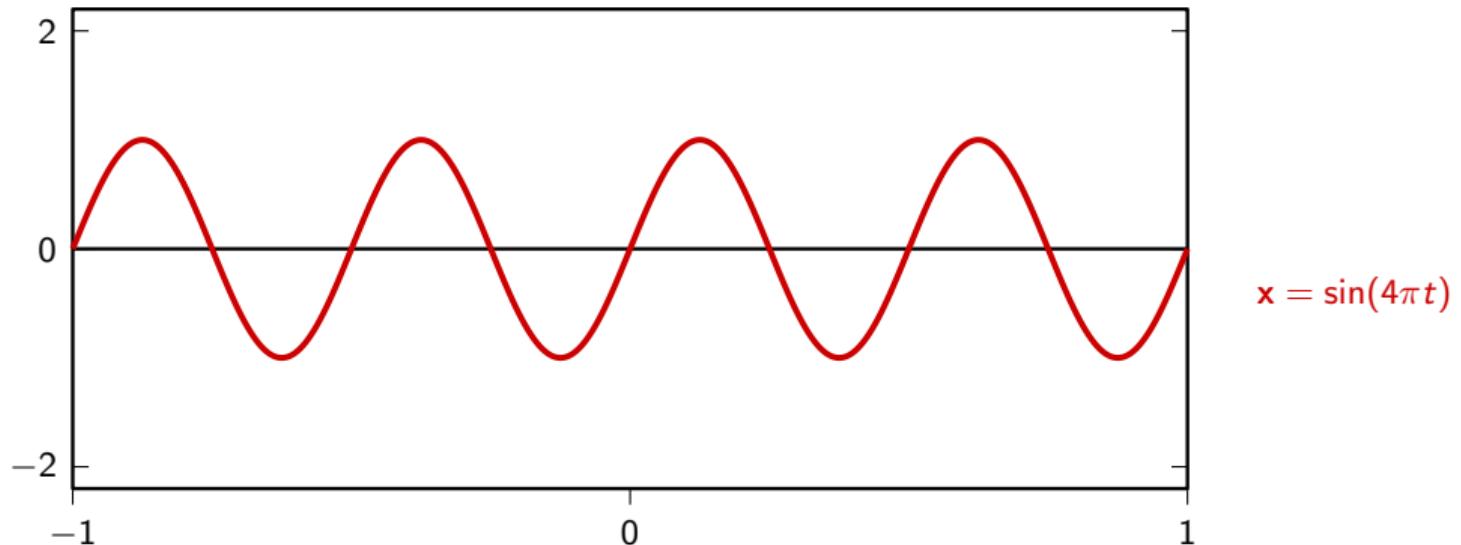
Norm and distance in \mathbb{R}^2

$$\|\mathbf{x} - \mathbf{y}\| = \sqrt{(x_0 - y_0)^2 + (x_1 - y_1)^2}$$



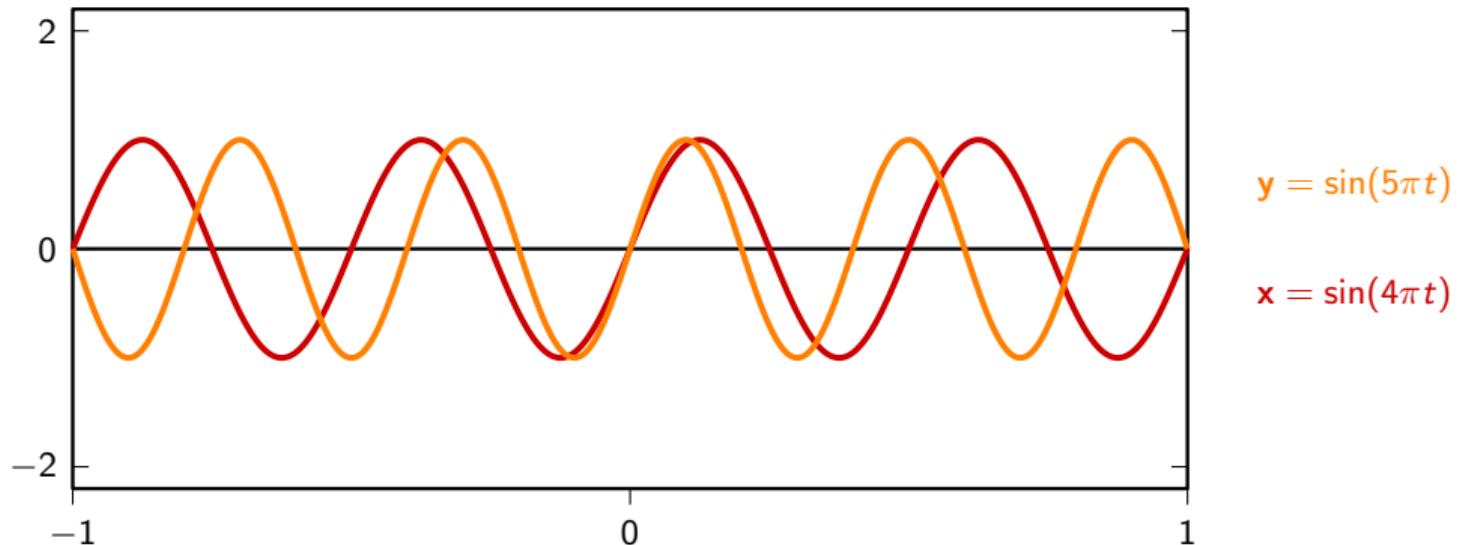
Distance in $L_2[-1, 1]$: the Mean Square Error

$$\|\mathbf{x} - \mathbf{y}\|^2 = \int_{-1}^1 |x(t) - y(t)|^2 dt$$



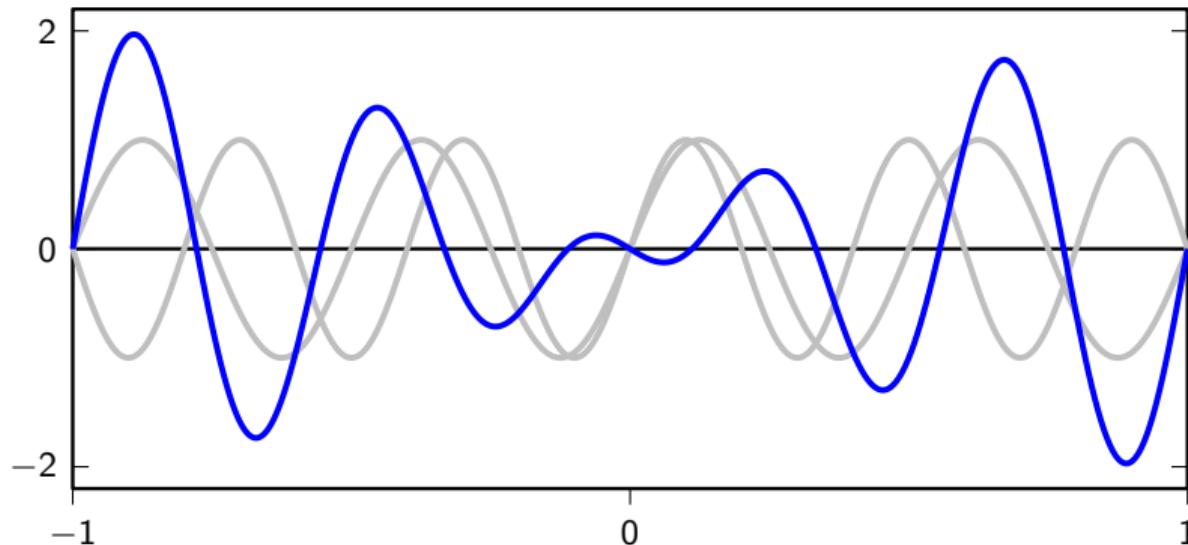
Distance in $L_2[-1, 1]$: the Mean Square Error

$$\|\mathbf{x} - \mathbf{y}\|^2 = \int_{-1}^1 |x(t) - y(t)|^2 dt$$



Distance in $L_2[-1, 1]$: the Mean Square Error

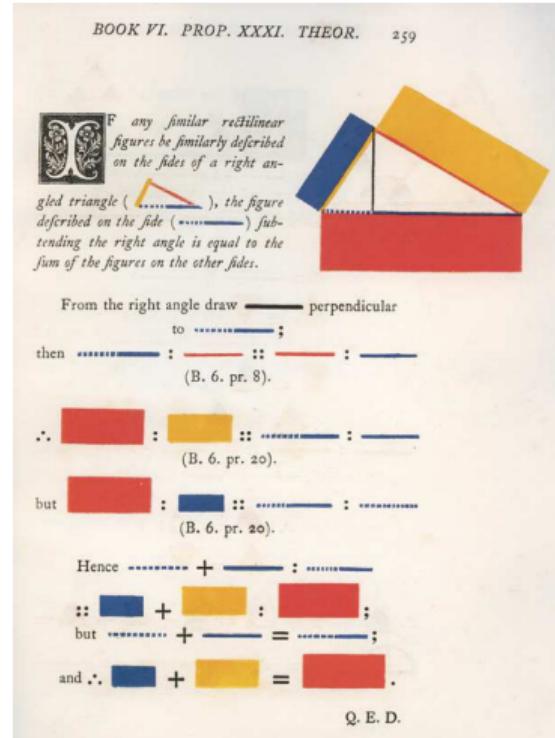
$$\|\mathbf{x} - \mathbf{y}\|^2 = \int_{-1}^1 |x(t) - y(t)|^2 dt = 2$$



A familiar result

Pythagorean theorem:

$$\begin{aligned}\|\mathbf{x} + \mathbf{y}\|^2 &= \langle \mathbf{x} + \mathbf{y}, \mathbf{x} + \mathbf{y} \rangle \\ &= \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 \text{ for } \mathbf{x} \perp \mathbf{y}\end{aligned}$$



From Euclid's elements by Oliver
Byrne (1810 - 1880)

Inner product for signals

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{n=0}^{N-1} x^*[n]y[n]$$

well defined for all finite-length vectors (i.e. vectors in \mathbb{C}^N)

Inner product for signals

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{n=-\infty}^{\infty} x^*[n]y[n]$$

careful: sum may explode!

Inner product for signals

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{n=-\infty}^{\infty} x^*[n]y[n]$$

We require sequences to be *square-summable*: $\sum |x[n]|^2 < \infty$

Space of square-summable sequences: $\ell_2(\mathbb{Z})$

bases

Bases

linear combination is the basic operation in vector spaces:

$$\mathbf{g} = \alpha \mathbf{x} + \beta \mathbf{y}$$

can we find a set of vectors $\{\mathbf{w}^{(k)}\}$ so that we can write *any* vector as a linear combination of the $\{\mathbf{w}^{(k)}\}$?

Bases

linear combination is the basic operation in vector spaces:

$$\mathbf{g} = \alpha \mathbf{x} + \beta \mathbf{y}$$

can we find a set of vectors $\{\mathbf{w}^{(k)}\}$ so that we can write *any* vector as a linear combination of the $\{\mathbf{w}^{(k)}\}$?

The canonical \mathbb{R}^2 basis

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathbf{e}^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{e}^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The canonical \mathbb{R}^2 basis

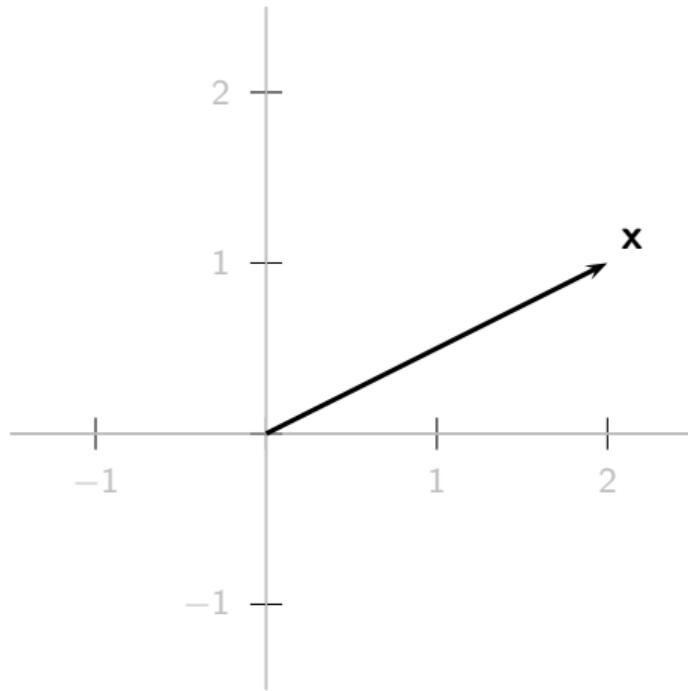
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathbf{e}^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{e}^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The canonical \mathbb{R}^2 basis

$$\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

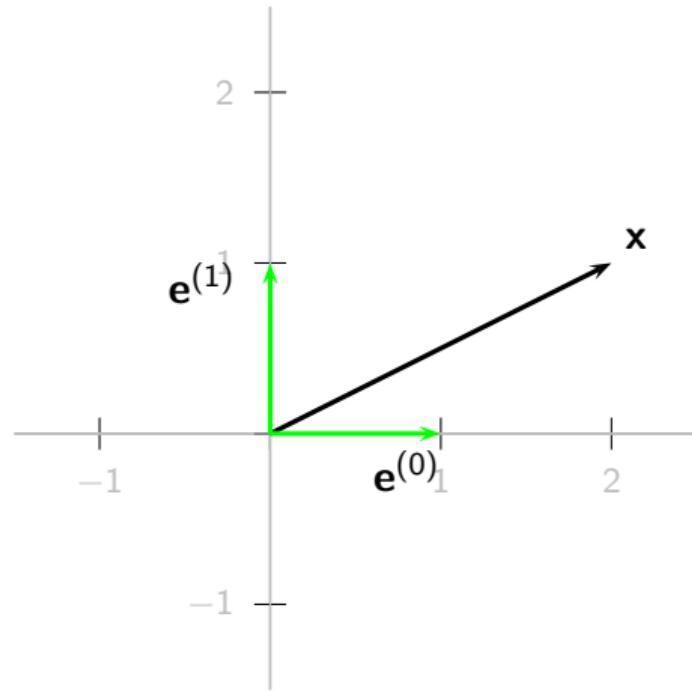
$$\mathbf{x} = 2\mathbf{e}^{(0)} + \mathbf{e}^{(1)}$$



The canonical \mathbb{R}^2 basis

$$\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\mathbf{x} = 2\mathbf{e}^{(0)} + \mathbf{e}^{(1)}$$



Another \mathbb{R}^2 basis

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \alpha_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\mathbf{v}^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{v}^{(1)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\alpha_1 = x_1 - x_2, \quad \alpha_2 = x_2$$

Another \mathbb{R}^2 basis

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \alpha_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

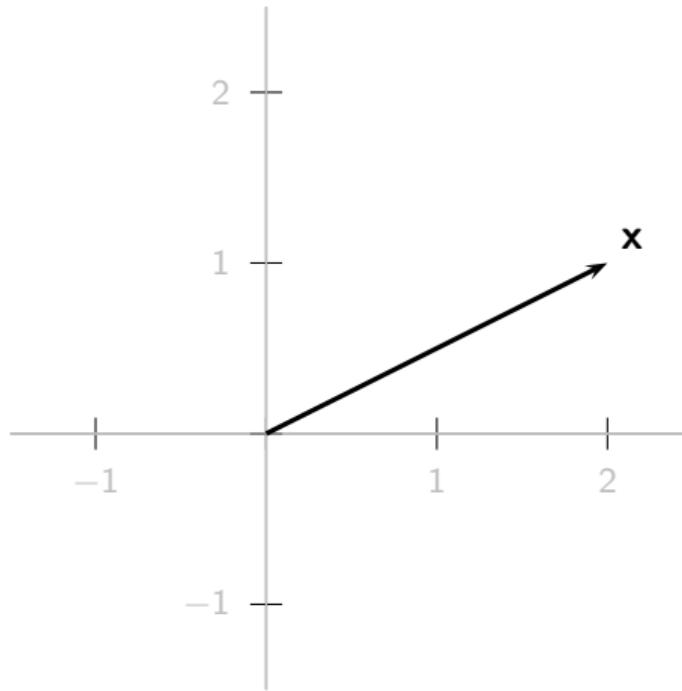
$$\mathbf{v}^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{v}^{(1)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\alpha_1 = x_1 - x_2, \quad \alpha_2 = x_2$$

Another \mathbb{R}^2 basis

$$\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

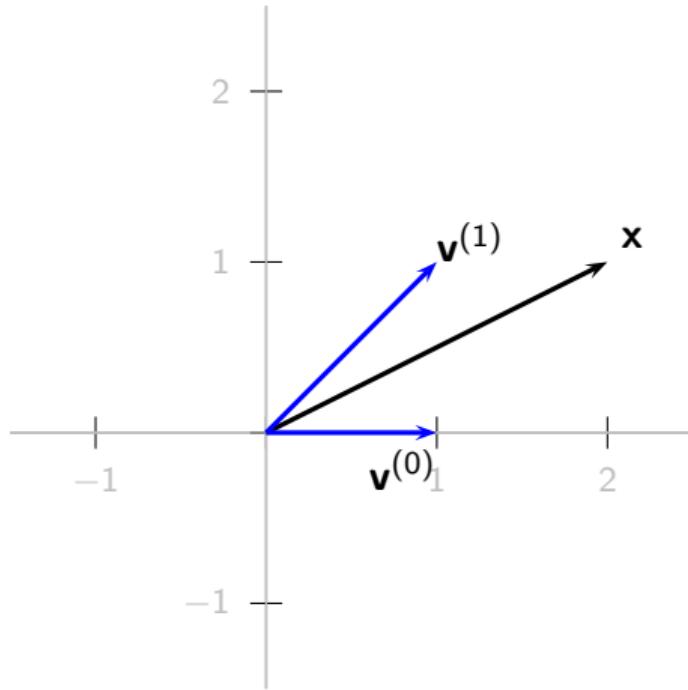
$$\mathbf{x} = \mathbf{v}^{(0)} + \mathbf{v}^{(1)}$$



Another \mathbb{R}^2 basis

$$\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{v}^{(0)} + \mathbf{v}^{(1)}$$



But this is not a basis for \mathbb{R}^2 ...

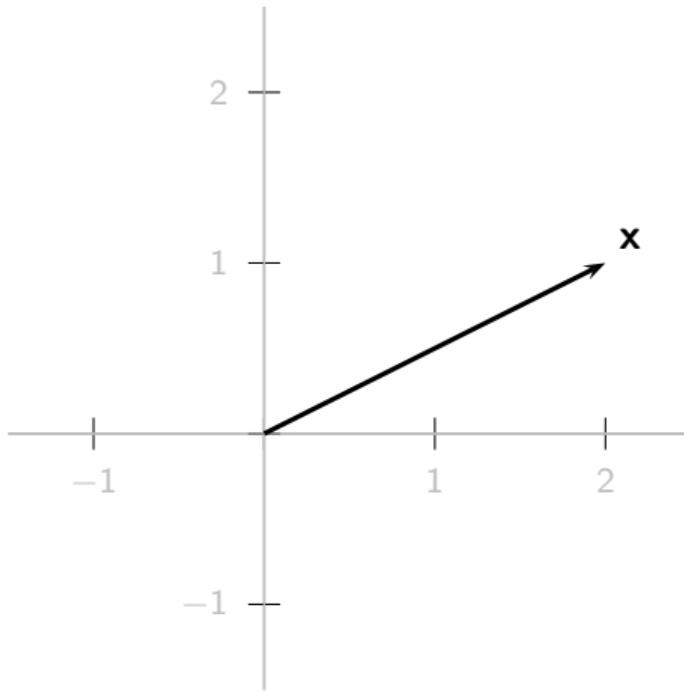
$$\mathbf{g}^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{g}^{(1)} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \neq \alpha_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} -1 \\ 0 \end{bmatrix} \text{ whenever } x_2 \neq 0$$

Another \mathbb{R}^2 basis

$$\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

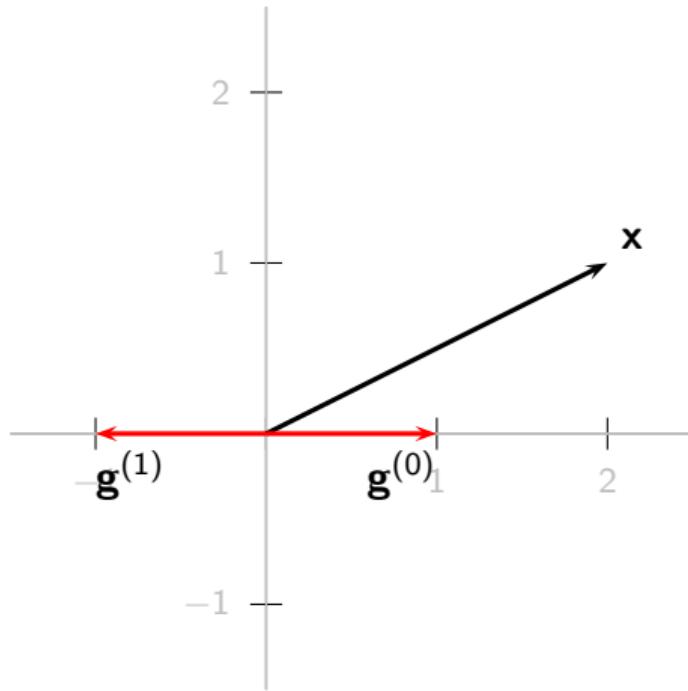
$$\mathbf{x} \neq \alpha_1 \mathbf{g}^{(0)} + \alpha_2 \mathbf{g}^{(1)}$$



Another \mathbb{R}^2 basis

$$\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\mathbf{x} \neq \alpha_1 \mathbf{g}^{(0)} + \alpha_2 \mathbf{g}^{(1)}$$



What about infinite-dimensional spaces?

$$\mathbf{x} = \sum_{k=-\infty}^{\infty} \alpha_k \mathbf{w}^{(k)}$$

A basis for $\ell_2(\mathbb{Z})$

$$\mathbf{e}^{(k)} = \begin{bmatrix} \vdots \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

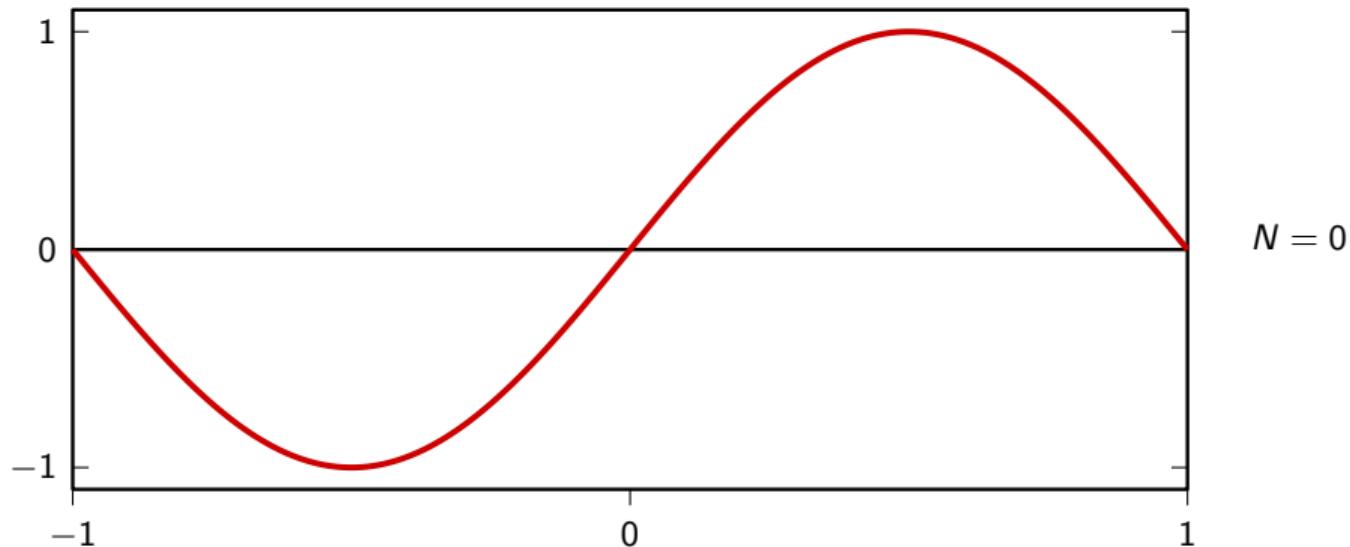
k-th position, $k \in \mathbb{Z}$

What about functional vector spaces?

$$f(t) = \sum_k \alpha_k h^{(k)}(t)$$

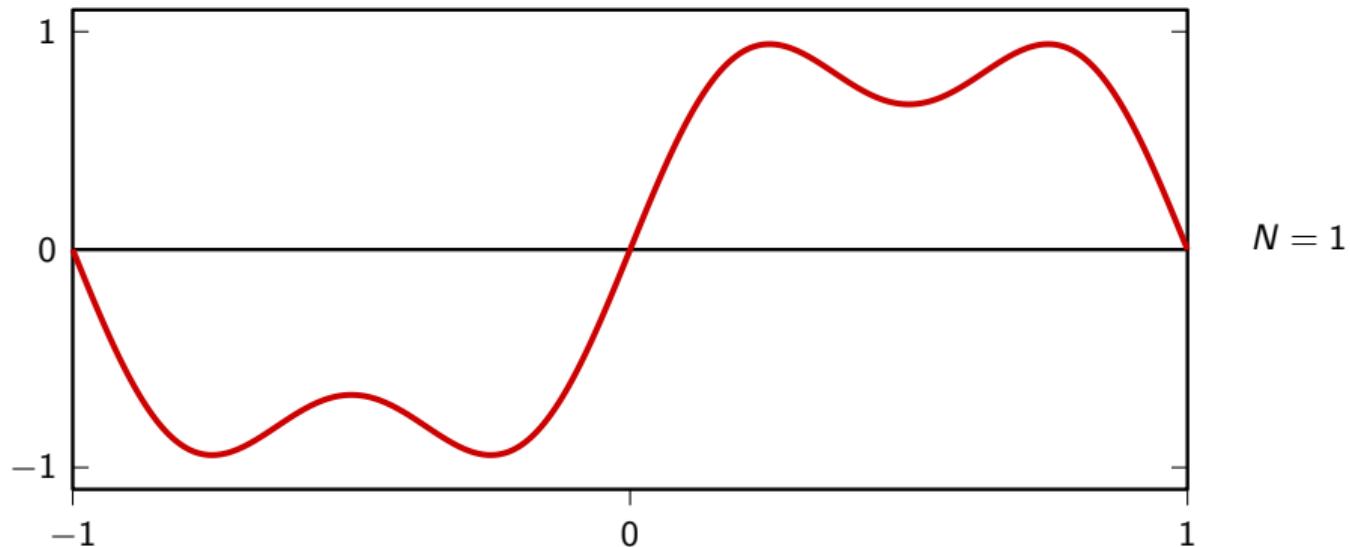
A basis for the functions over an interval?

$$\sum_{k=0}^N (1/k) \mathbf{v}^{(k)}, \quad \mathbf{v}^{(k)} = \sin(\pi(2k+1)t), \quad t \in [-1, 1]$$



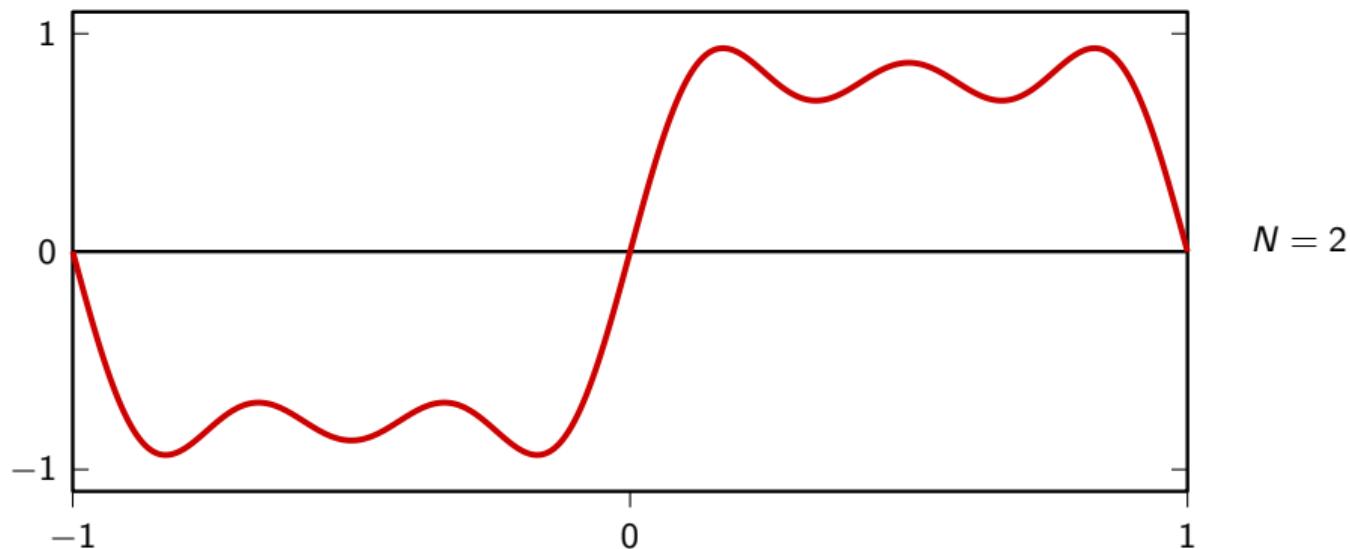
A basis for the functions over an interval?

$$\sum_{k=0}^N (1/k) \mathbf{v}^{(k)}, \quad \mathbf{v}^{(k)} = \sin(\pi(2k+1)t), \quad t \in [-1, 1]$$



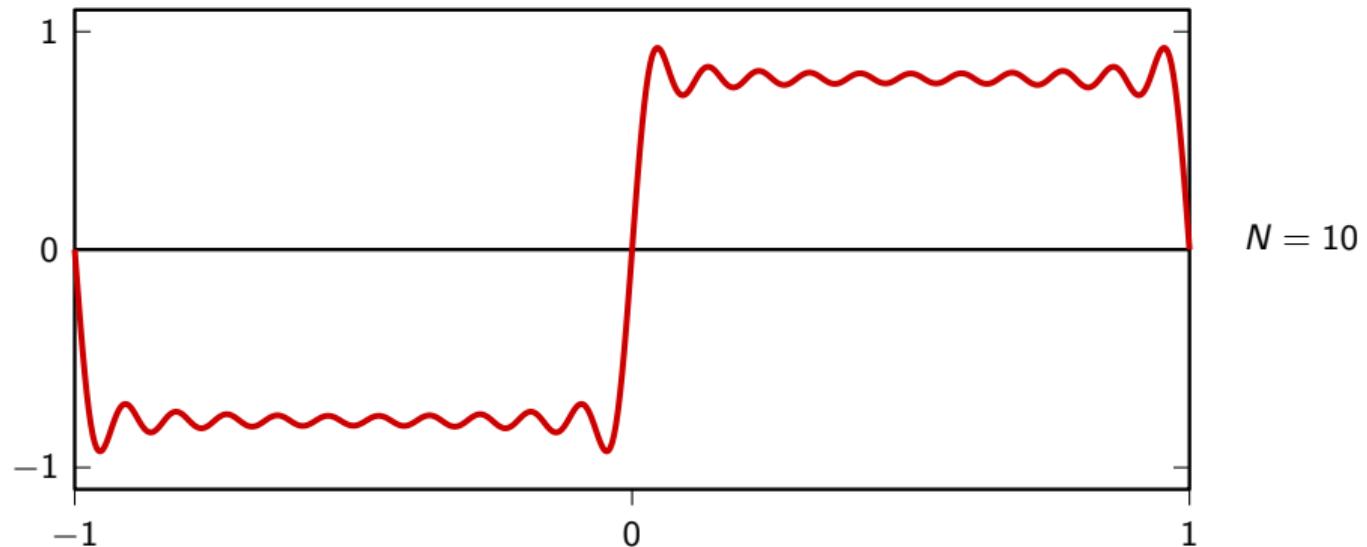
A basis for the functions over an interval?

$$\sum_{k=0}^N (1/k) \mathbf{v}^{(k)}, \quad \mathbf{v}^{(k)} = \sin(\pi(2k+1)t), \quad t \in [-1, 1]$$



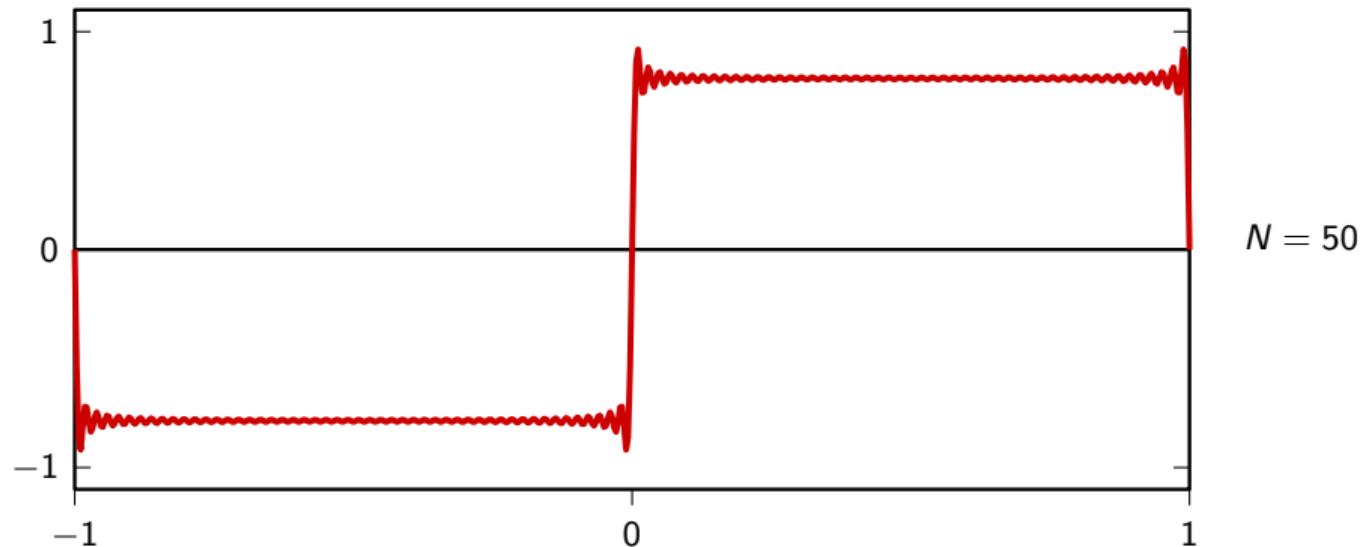
A basis for the functions over an interval?

$$\sum_{k=0}^N (1/k) \mathbf{v}^{(k)}, \quad \mathbf{v}^{(k)} = \sin(\pi(2k+1)t), \quad t \in [-1, 1]$$



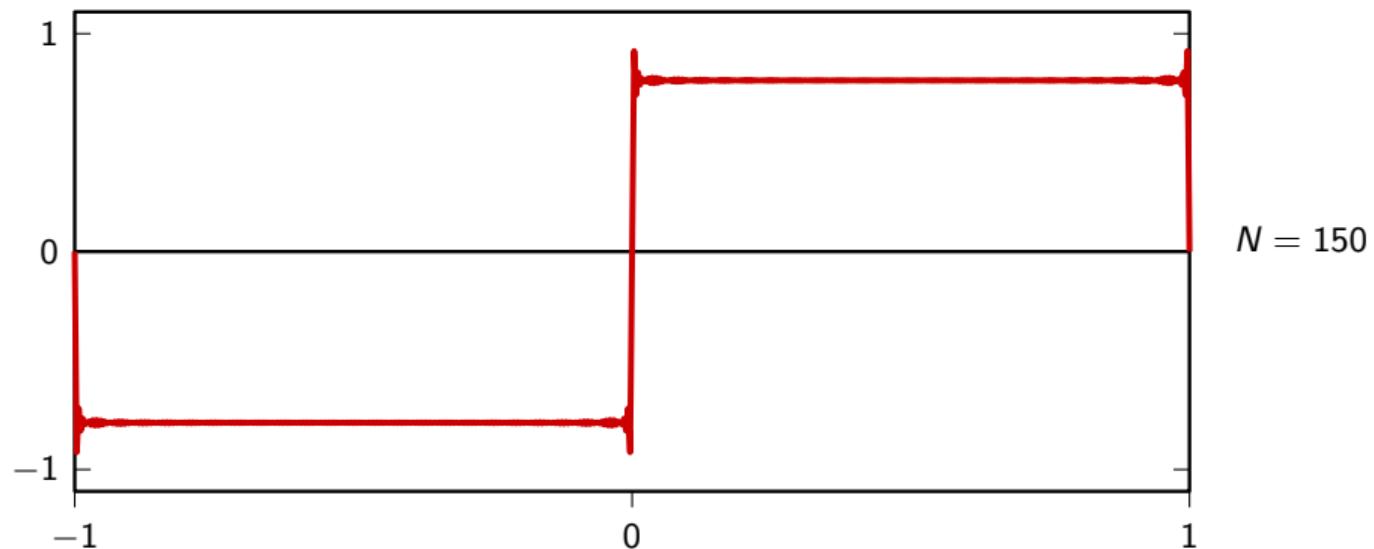
A basis for the functions over an interval?

$$\sum_{k=0}^N (1/k) \mathbf{v}^{(k)}, \quad \mathbf{v}^{(k)} = \sin(\pi(2k+1)t), \quad t \in [-1, 1]$$



A basis for the functions over an interval?

$$\sum_{k=0}^N (1/k) \mathbf{v}^{(k)}, \quad \mathbf{v}^{(k)} = \sin(\pi(2k+1)t), \quad t \in [-1, 1]$$



Bases: formal definition

Given:

- ▶ a vector space H
- ▶ a set of K vectors from H : $W = \{\mathbf{w}^{(k)}\}_{k=0,1,\dots,K-1}$

W is a basis for H if:

1. we can write for all $\mathbf{x} \in H$:

$$\mathbf{x} = \sum_{k=0}^{K-1} \alpha_k \mathbf{w}^{(k)}, \quad \alpha_k \in \mathbb{C}$$

2. the coefficients α_k are unique

Bases: formal definition

Given:

- ▶ a vector space H
- ▶ a set of K vectors from H : $W = \{\mathbf{w}^{(k)}\}_{k=0,1,\dots,K-1}$

W is a basis for H if:

1. we can write for all $\mathbf{x} \in H$:

$$\mathbf{x} = \sum_{k=0}^{K-1} \alpha_k \mathbf{w}^{(k)}, \quad \alpha_k \in \mathbb{C}$$

2. the coefficients α_k are unique

Bases: formal definition

Unique representation implies linear independence:

$$\sum_{k=0}^{K-1} \alpha_k \mathbf{w}^{(k)} = 0 \iff \alpha_k = 0, \quad k = 0, 1, \dots, K-1$$

Special bases

Orthogonal basis:

$$\langle \mathbf{w}^{(k)}, \mathbf{w}^{(n)} \rangle = 0 \text{ for } k \neq n$$

Orthonormal basis:

$$\langle \mathbf{w}^{(k)}, \mathbf{w}^{(n)} \rangle = \delta[n - k]$$

(we can always orthonormalize a basis via the Gram-Schmidt algorithm)

Special bases

Orthogonal basis:

$$\langle \mathbf{w}^{(k)}, \mathbf{w}^{(n)} \rangle = 0 \text{ for } k \neq n$$

Orthonormal basis:

$$\langle \mathbf{w}^{(k)}, \mathbf{w}^{(n)} \rangle = \delta[n - k]$$

(we can always orthonormalize a basis via the Gram-Schmidt algorithm)

Special bases

Orthogonal basis:

$$\langle \mathbf{w}^{(k)}, \mathbf{w}^{(n)} \rangle = 0 \text{ for } k \neq n$$

Orthonormal basis:

$$\langle \mathbf{w}^{(k)}, \mathbf{w}^{(n)} \rangle = \delta[n - k]$$

(we can always orthonormalize a basis via the Gram-Schmidt algorithm)

Basis expansion

$$\mathbf{x} = \sum_{k=0}^{K-1} \alpha_k \mathbf{w}^{(k)}$$

how do we find the α 's ?

Orthonormal bases are the best:

$$\alpha_k = \langle \mathbf{w}^{(k)}, \mathbf{x} \rangle$$

Basis expansion

$$\mathbf{x} = \sum_{k=0}^{K-1} \alpha_k \mathbf{w}^{(k)}$$

how do we find the α 's ?

Orthonormal bases are the best:

$$\alpha_k = \langle \mathbf{w}^{(k)}, \mathbf{x} \rangle$$

Change of basis

$$\mathbf{x} = \sum_{k=0}^{K-1} \alpha_k \mathbf{w}^{(k)} = \sum_{k=0}^{K-1} \beta_k \mathbf{v}^{(k)}$$

if $\{\mathbf{v}^{(k)}\}$ is orthonormal:

$$\beta_h = \langle \mathbf{v}^{(h)}, \mathbf{x} \rangle$$

$$= \langle \mathbf{v}^{(h)}, \sum_{k=0}^{K-1} \alpha_k \mathbf{w}^{(k)} \rangle$$

$$= \sum_{k=0}^{K-1} \alpha_k \langle \mathbf{v}^{(h)}, \mathbf{w}^{(k)} \rangle$$

Change of basis

$$\mathbf{x} = \sum_{k=0}^{K-1} \alpha_k \mathbf{w}^{(k)} = \sum_{k=0}^{K-1} \beta_k \mathbf{v}^{(k)}$$

if $\{\mathbf{v}^{(k)}\}$ is orthonormal:

$$\beta_h = \langle \mathbf{v}^{(h)}, \mathbf{x} \rangle$$

$$= \langle \mathbf{v}^{(h)}, \sum_{k=0}^{K-1} \alpha_k \mathbf{w}^{(k)} \rangle$$

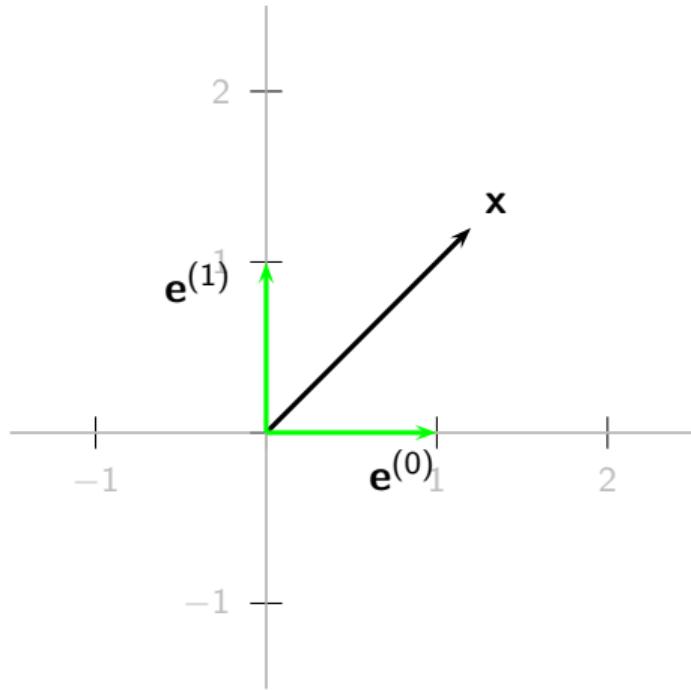
$$= \sum_{k=0}^{K-1} \alpha_k \langle \mathbf{v}^{(h)}, \mathbf{w}^{(k)} \rangle$$

Change of basis

$$\begin{aligned}\beta_h &= \sum_{k=0}^{K-1} \alpha_k \langle \mathbf{v}^{(h)}, \mathbf{w}^{(k)} \rangle \\ &= \sum_{k=0}^{K-1} \alpha_k c_{hk} \\ &= \begin{bmatrix} c_{00} & c_{01} & \dots & c_{0(K-1)} \\ \vdots & & & \\ c_{(K-1)0} & c_{(K-1)1} & \dots & c_{(K-1)(K-1)} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{K-1} \end{bmatrix}\end{aligned}$$

Change of basis: example

- ▶ canonical basis $E = \{e^{(0)}, e^{(1)}\}$
- ▶ $x = \alpha_0 e^{(0)} + \alpha_1 e^{(1)}$



Change of basis: example

- ▶ canonical basis $E = \{\mathbf{e}^{(0)}, \mathbf{e}^{(1)}\}$

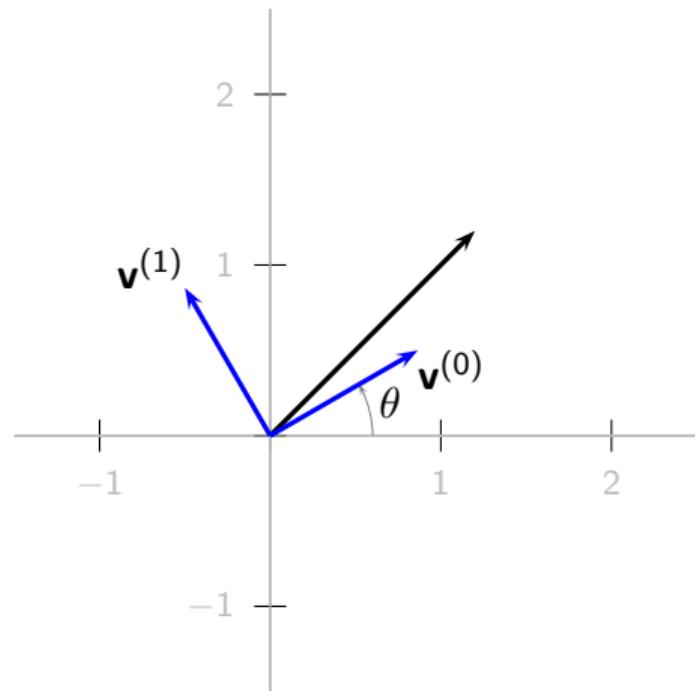
- ▶ $\mathbf{x} = \alpha_0 \mathbf{e}^{(0)} + \alpha_1 \mathbf{e}^{(1)}$

- ▶ new basis $V = \{\mathbf{v}^{(0)}, \mathbf{v}^{(1)}\}$ with

$$\mathbf{v}^{(0)} = [\cos \theta \quad \sin \theta]^T$$

$$\mathbf{v}^{(1)} = [-\sin \theta \quad \cos \theta]^T$$

- ▶ $\mathbf{x} = \beta_0 \mathbf{v}^{(0)} + \beta_1 \mathbf{v}^{(1)}$



Change of basis: example

- ▶ new basis is orthonormal:

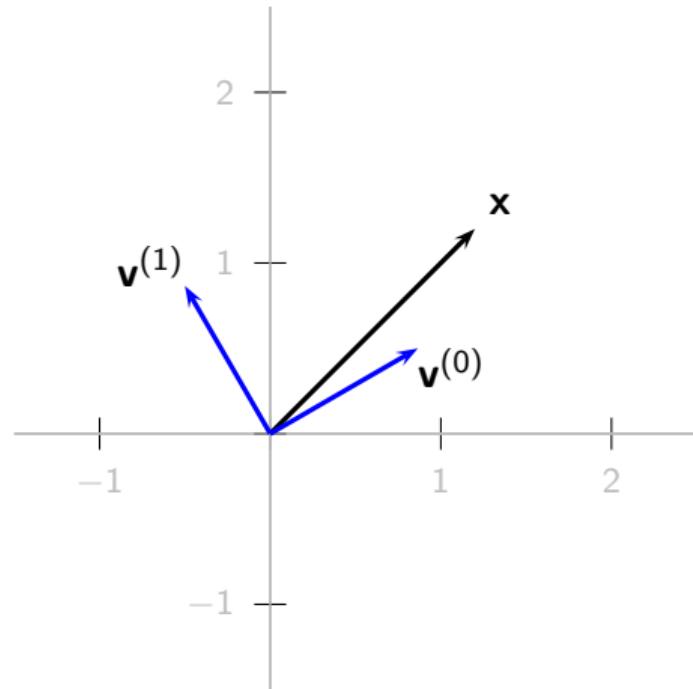
$$c_{hk} = \langle \mathbf{v}^{(h)}, \mathbf{e}^{(k)} \rangle$$

- ▶ in compact form:

$$\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \mathbf{R}\boldsymbol{\alpha}$$

- ▶ \mathbf{R} : rotation matrix

- ▶ key fact: $\mathbf{R}^T \mathbf{R} = \mathbf{I}$



Change of basis: example

- ▶ new basis is orthonormal:

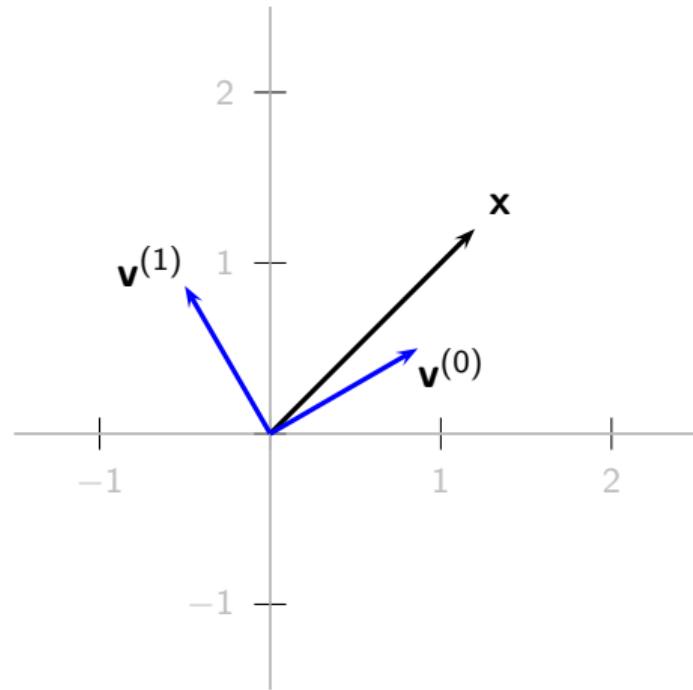
$$c_{hk} = \langle \mathbf{v}^{(h)}, \mathbf{e}^{(k)} \rangle$$

- ▶ in compact form:

$$\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \mathbf{R}\boldsymbol{\alpha}$$

- ▶ \mathbf{R} : rotation matrix

- ▶ key fact: $\mathbf{R}^T \mathbf{R} = \mathbf{I}$



Change of basis: example

- ▶ new basis is orthonormal:

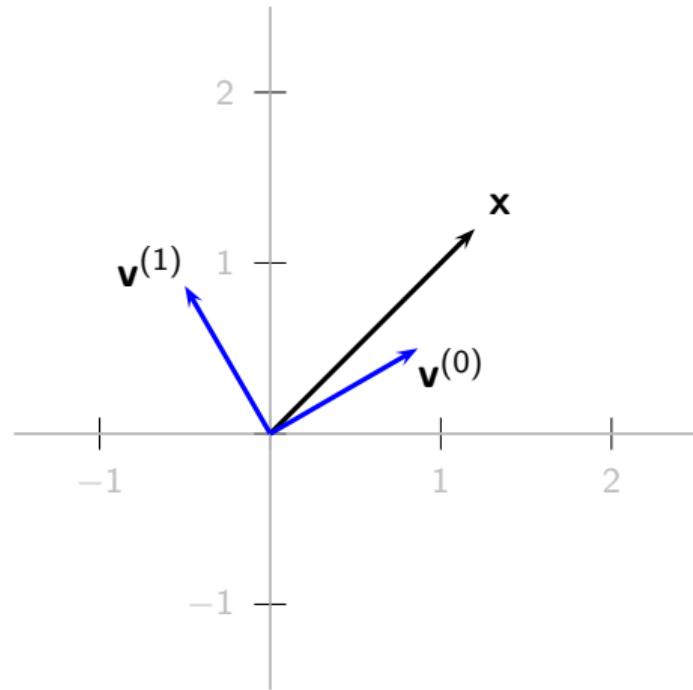
$$c_{hk} = \langle \mathbf{v}^{(h)}, \mathbf{e}^{(k)} \rangle$$

- ▶ in compact form:

$$\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \mathbf{R}\boldsymbol{\alpha}$$

- ▶ \mathbf{R} : rotation matrix

- ▶ key fact: $\mathbf{R}^T \mathbf{R} = \mathbf{I}$



Change of basis: example

- ▶ new basis is orthonormal:

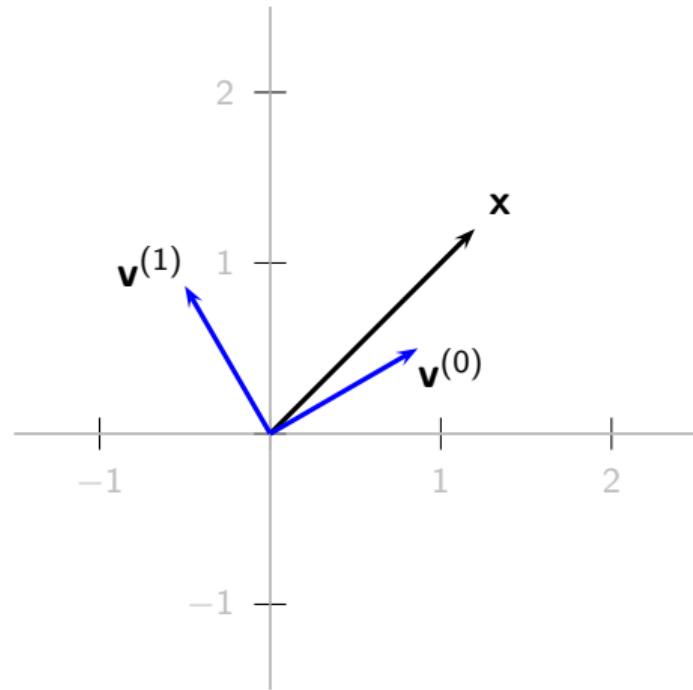
$$c_{hk} = \langle \mathbf{v}^{(h)}, \mathbf{e}^{(k)} \rangle$$

- ▶ in compact form:

$$\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \mathbf{R}\boldsymbol{\alpha}$$

- ▶ \mathbf{R} : rotation matrix

- ▶ key fact: $\mathbf{R}^T \mathbf{R} = \mathbf{I}$



Norm and energy

In \mathbb{C}^N

$$\|\mathbf{x}\|^2 = \langle \mathbf{x}, \mathbf{x} \rangle = \sum_{k=0}^{K-1} |x_k|^2$$

(remember the definition of energy for discrete-time signals)

Parseval's Theorem (conservation of energy)

If $\{\mathbf{w}^{(k)}\}$ is orthonormal and $\mathbf{x} = \sum_{k=0}^{K-1} \alpha_k \mathbf{w}^{(k)}$

$$\|\mathbf{x}\|^2 = \langle \mathbf{x}, \mathbf{x} \rangle$$

$$= \sum_{k=0}^{K-1} |\alpha_k|^2$$

energy is conserved across a change of basis

Conservation of energy: example

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \mathbf{R}\alpha$$

► square norm in canonical basis: $\|\mathbf{x}\|^2 = \alpha_0^2 + \alpha_1^2$

► square norm in rotated basis: $\|\mathbf{x}\|^2 = \beta_0^2 + \beta_1^2$

► let's verify Parseval:

$$\begin{aligned} \beta_0^2 + \beta_1^2 &= \beta^T \beta \\ &= (\mathbf{R}\alpha)^T (\mathbf{R}\alpha) \\ &= \alpha^T (\mathbf{R}^T \mathbf{R}) \alpha \\ &= \alpha^T \alpha = \alpha_0^2 + \alpha_1^2 \end{aligned}$$

Conservation of energy: example

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \mathbf{R}\alpha$$

- ▶ square norm in canonical basis: $\|\mathbf{x}\|^2 = \alpha_0^2 + \alpha_1^2$
- ▶ square norm in rotated basis: $\|\mathbf{x}\|^2 = \beta_0^2 + \beta_1^2$
- ▶ let's verify Parseval:

$$\begin{aligned}\beta_0^2 + \beta_1^2 &= \beta^T \beta \\ &= (\mathbf{R}\alpha)^T (\mathbf{R}\alpha) \\ &= \alpha^T (\mathbf{R}^T \mathbf{R}) \alpha \\ &= \alpha^T \alpha = \alpha_0^2 + \alpha_1^2\end{aligned}$$

Conservation of energy: example

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \mathbf{R}\alpha$$

- ▶ square norm in canonical basis: $\|\mathbf{x}\|^2 = \alpha_0^2 + \alpha_1^2$
- ▶ square norm in rotated basis: $\|\mathbf{x}\|^2 = \beta_0^2 + \beta_1^2$
- ▶ let's verify Parseval:

$$\begin{aligned}\beta_0^2 + \beta_1^2 &= \beta^T \beta \\ &= (\mathbf{R}\alpha)^T (\mathbf{R}\alpha) \\ &= \alpha^T (\mathbf{R}^T \mathbf{R}) \alpha \\ &= \alpha^T \alpha = \alpha_0^2 + \alpha_1^2\end{aligned}$$

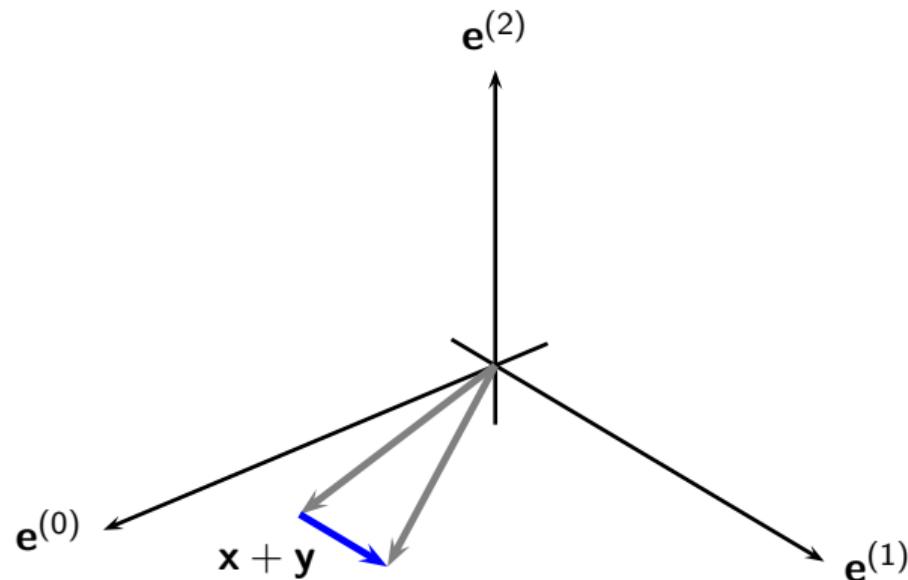
subspaces and approximations

Vector subspace

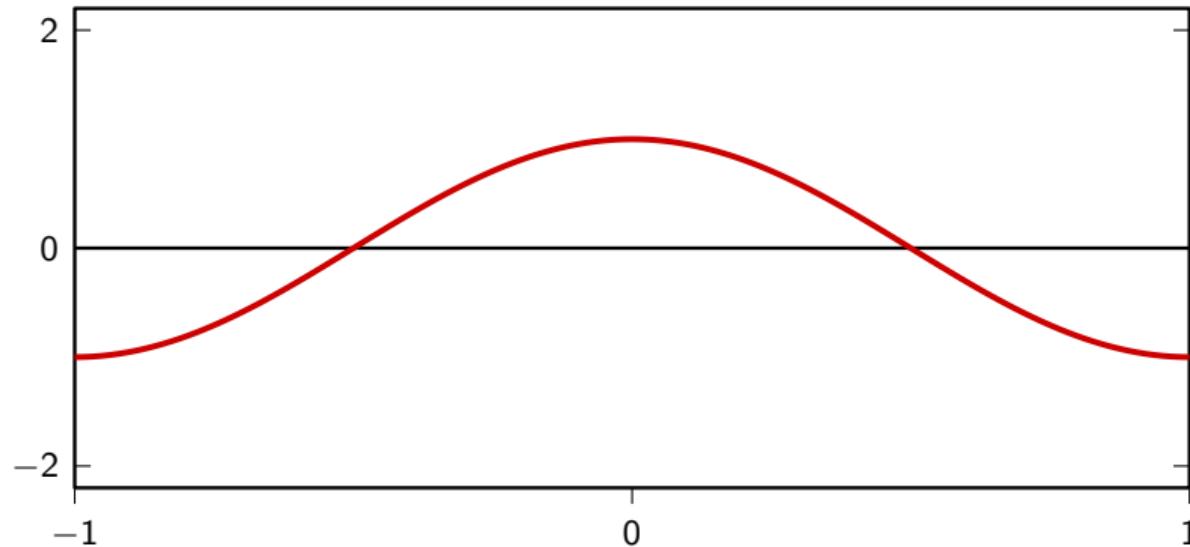
a subset of vectors *closed* under addition and scalar multiplication

Example in Euclidean Space

intuition: $\mathbb{R}^2 \subset \mathbb{R}^3$

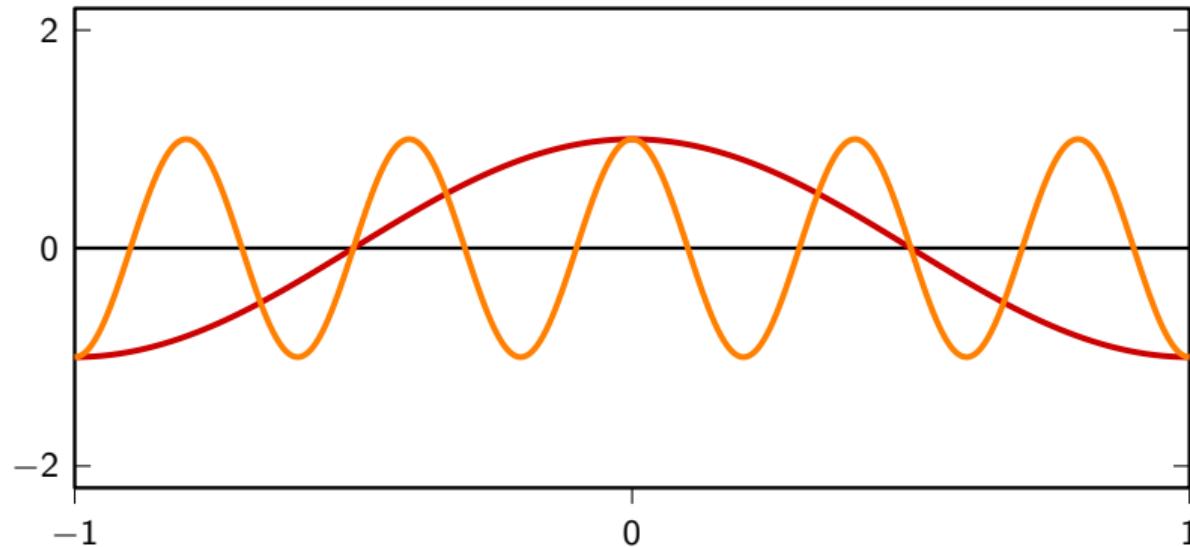


Subspace of symmetric functions over $L_2[-1, 1]$



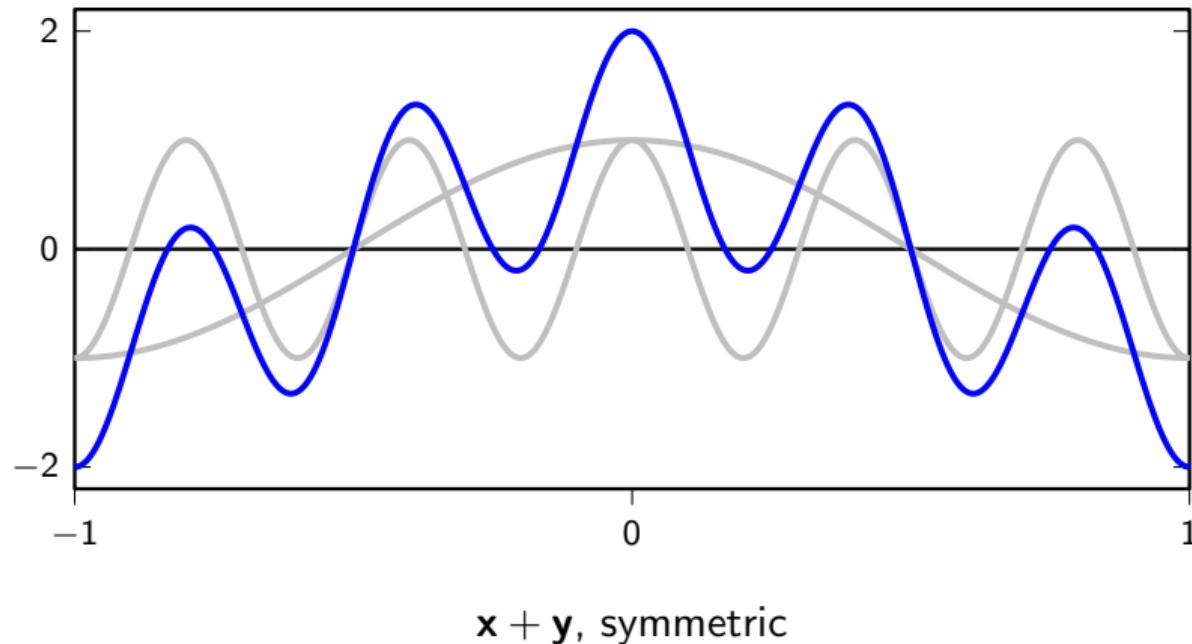
$$x = \cos(\pi t)$$

Subspace of symmetric functions over $L_2[-1, 1]$



$$y = \cos(5\pi t)$$

Subspace of symmetric functions over $L_2[-1, 1]$



Subspaces have their own basis

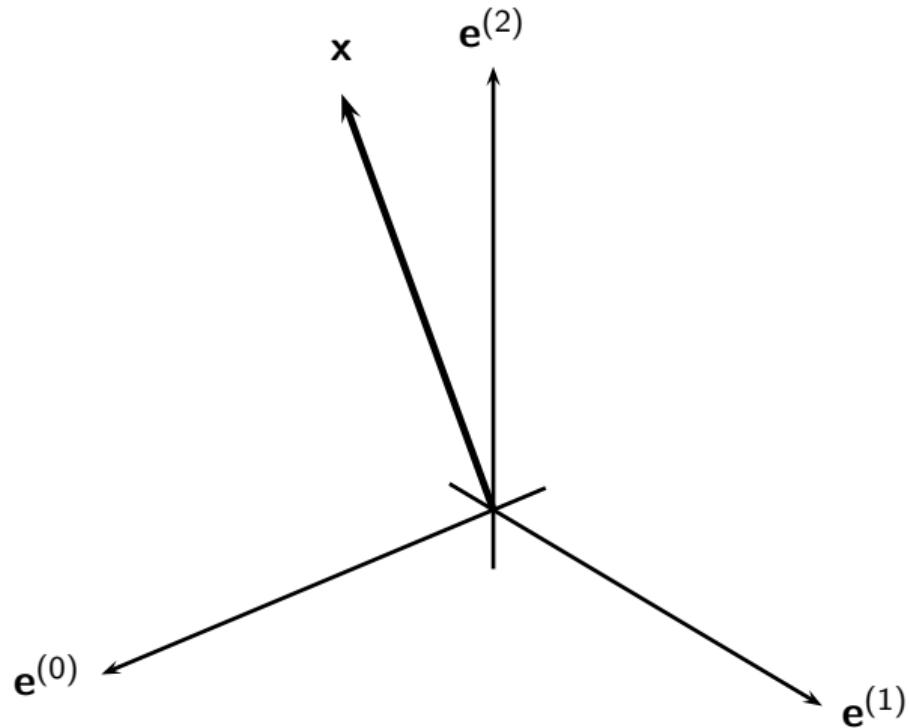
$$\mathbf{e}^{(0)} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{e}^{(1)} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

basis vector for the plane in \mathbb{R}^3

Approximation

Problem:

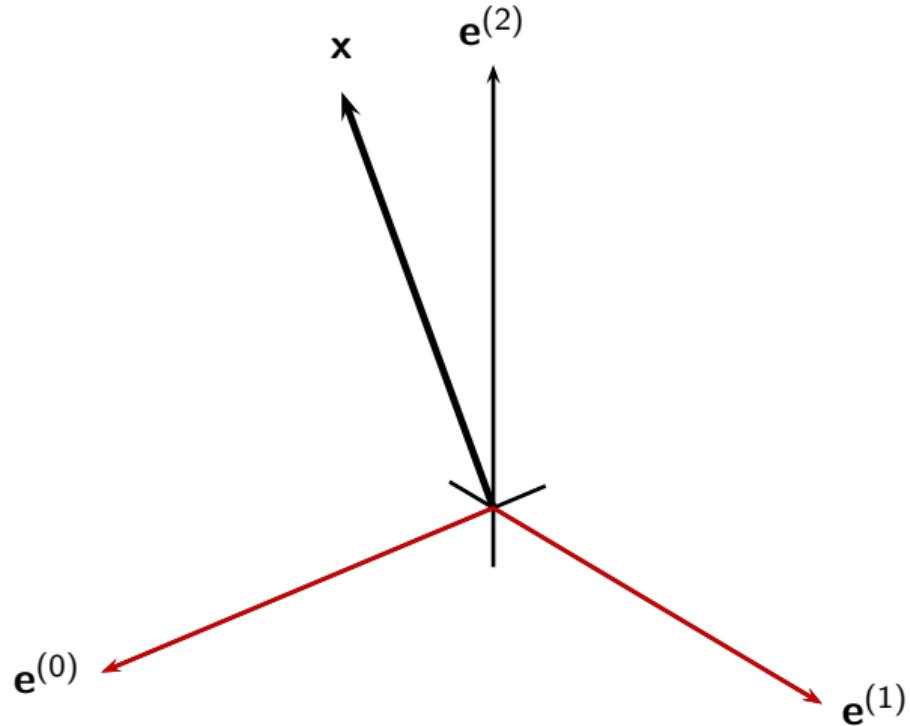
- ▶ vector $\mathbf{x} \in V$
- ▶ subspace $S \subseteq V$
- ▶ approximate \mathbf{x} with $\hat{\mathbf{x}} \in S$



Approximation

Problem:

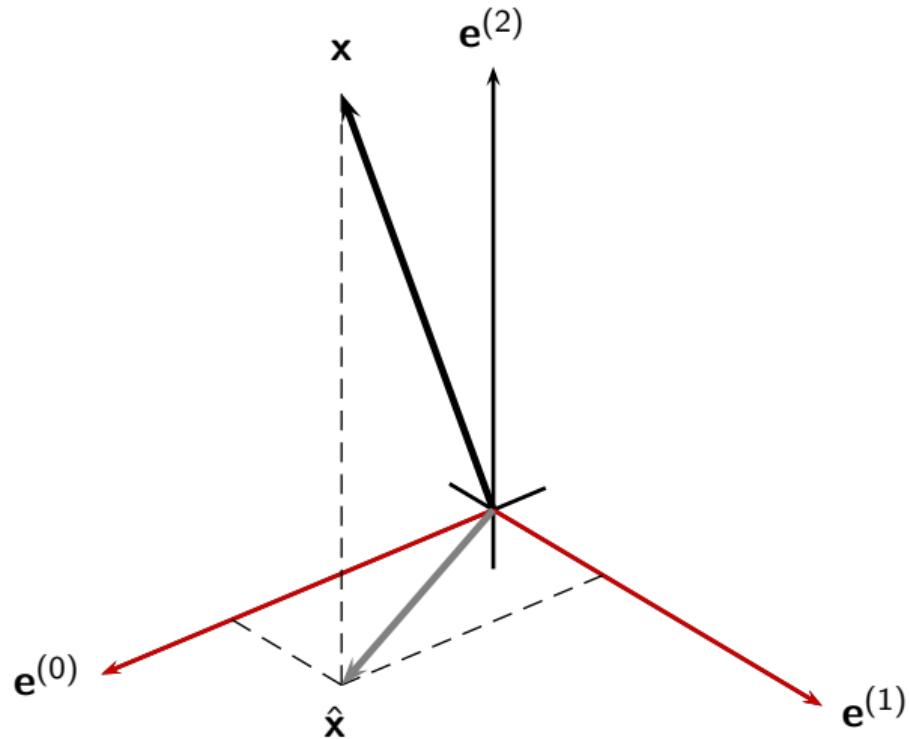
- ▶ vector $\mathbf{x} \in V$
- ▶ subspace $S \subseteq V$
- ▶ approximate \mathbf{x} with $\hat{\mathbf{x}} \in S$



Approximation

Problem:

- ▶ vector $\mathbf{x} \in V$
- ▶ subspace $S \subseteq V$
- ▶ approximate \mathbf{x} with $\hat{\mathbf{x}} \in S$



Least-Squares Approximation

- ▶ $\{\mathbf{s}^{(k)}\}_{k=0,1,\dots,K-1}$ orthonormal basis for S

- ▶ orthogonal projection:

$$\hat{\mathbf{x}} = \sum_{k=0}^{K-1} \langle \mathbf{s}^{(k)}, \mathbf{x} \rangle \mathbf{s}^{(k)}$$

orthogonal projection is the “best” approximation over S

Least-Squares Approximation

- ▶ $\{\mathbf{s}^{(k)}\}_{k=0,1,\dots,K-1}$ orthonormal basis for S
- ▶ orthogonal projection:

$$\hat{\mathbf{x}} = \sum_{k=0}^{K-1} \langle \mathbf{s}^{(k)}, \mathbf{x} \rangle \mathbf{s}^{(k)}$$

orthogonal projection is the “best” approximation over S

Least-Squares Approximation

- ▶ $\{\mathbf{s}^{(k)}\}_{k=0,1,\dots,K-1}$ orthonormal basis for S
- ▶ orthogonal projection:

$$\hat{\mathbf{x}} = \sum_{k=0}^{K-1} \langle \mathbf{s}^{(k)}, \mathbf{x} \rangle \mathbf{s}^{(k)}$$

orthogonal projection is the “best” approximation over S

Least-Squares Approximation

- ▶ orthogonal projection has minimum-norm error:

$$\arg \min_{y \in S} \|x - y\| = \hat{x}$$

- ▶ error is orthogonal to approximation:

$$\langle x - \hat{x}, \hat{x} \rangle = 0$$

Least-Squares Approximation

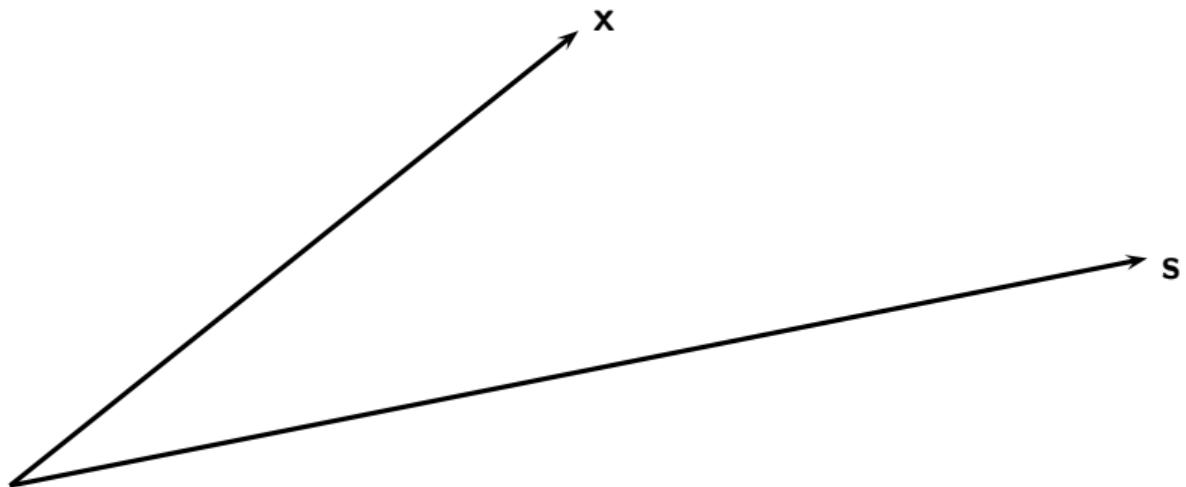
- ▶ orthogonal projection has minimum-norm error:

$$\arg \min_{y \in S} \|x - y\| = \hat{x}$$

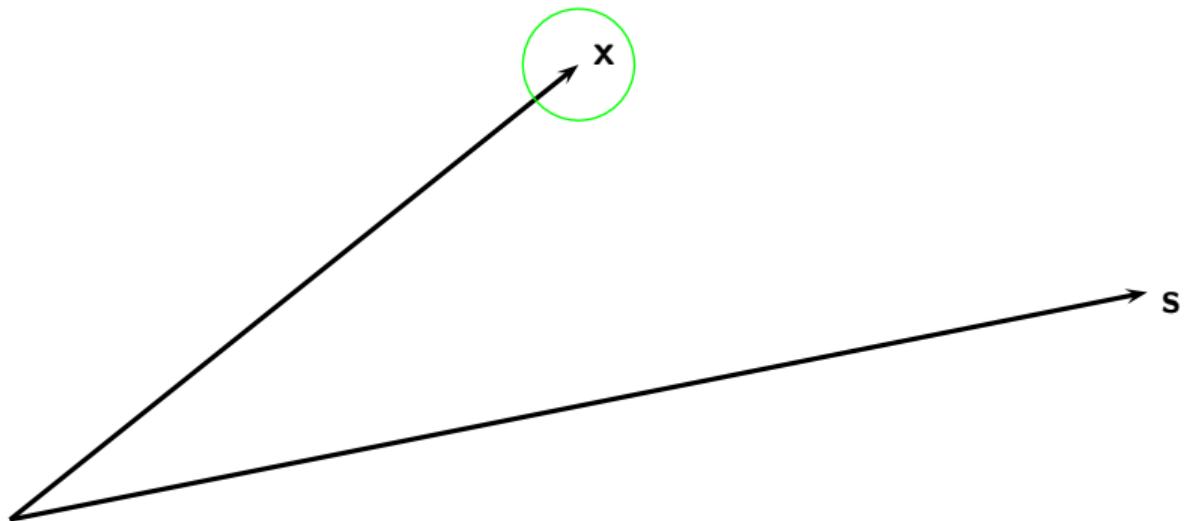
- ▶ error is orthogonal to approximation:

$$\langle x - \hat{x}, \hat{x} \rangle = 0$$

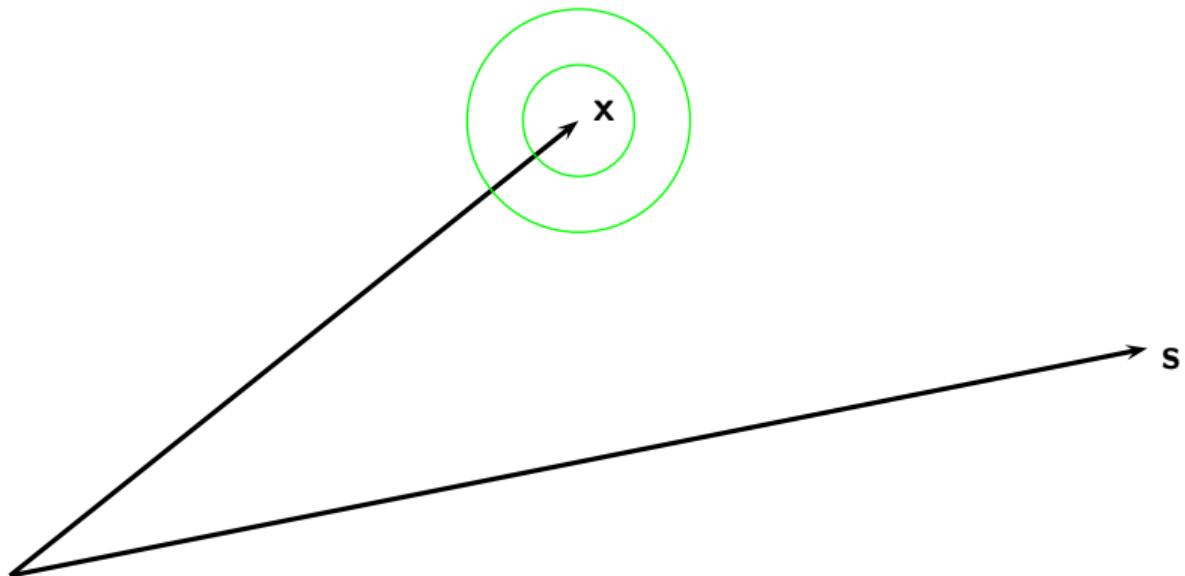
Least Squares Approximation



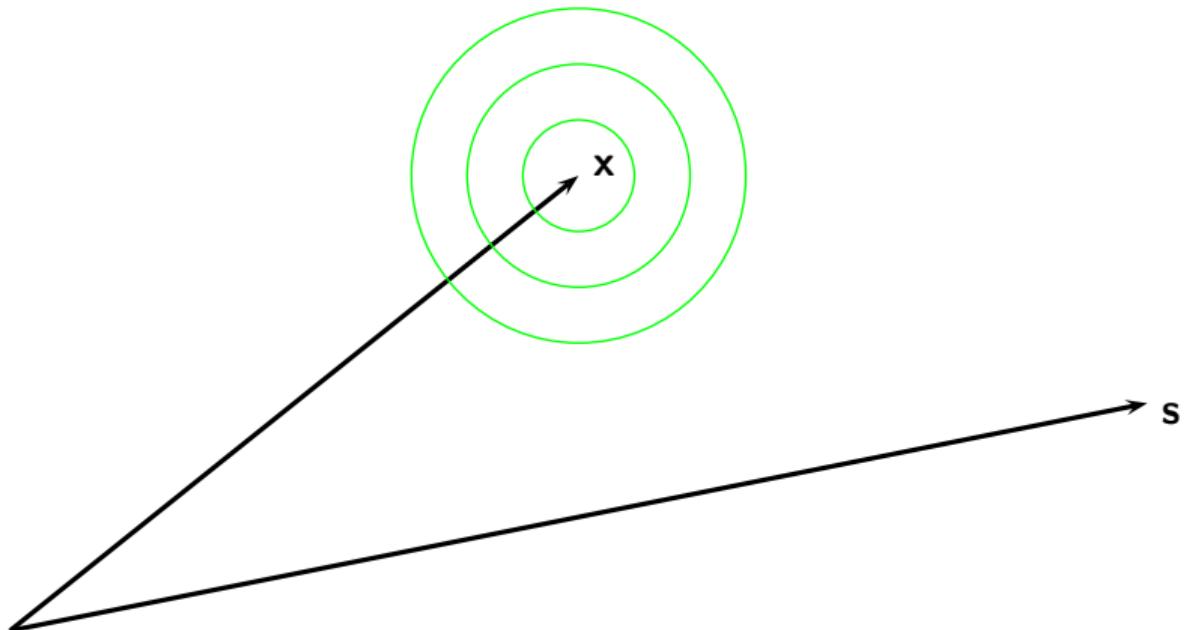
Least Squares Approximation



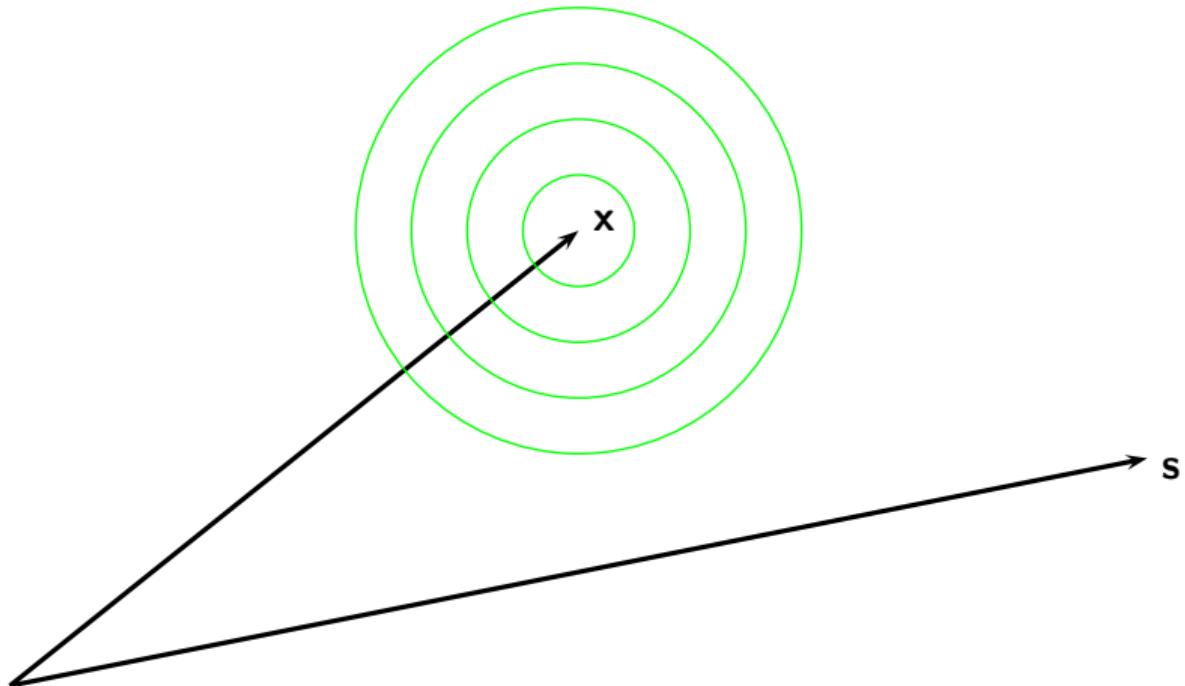
Least Squares Approximation



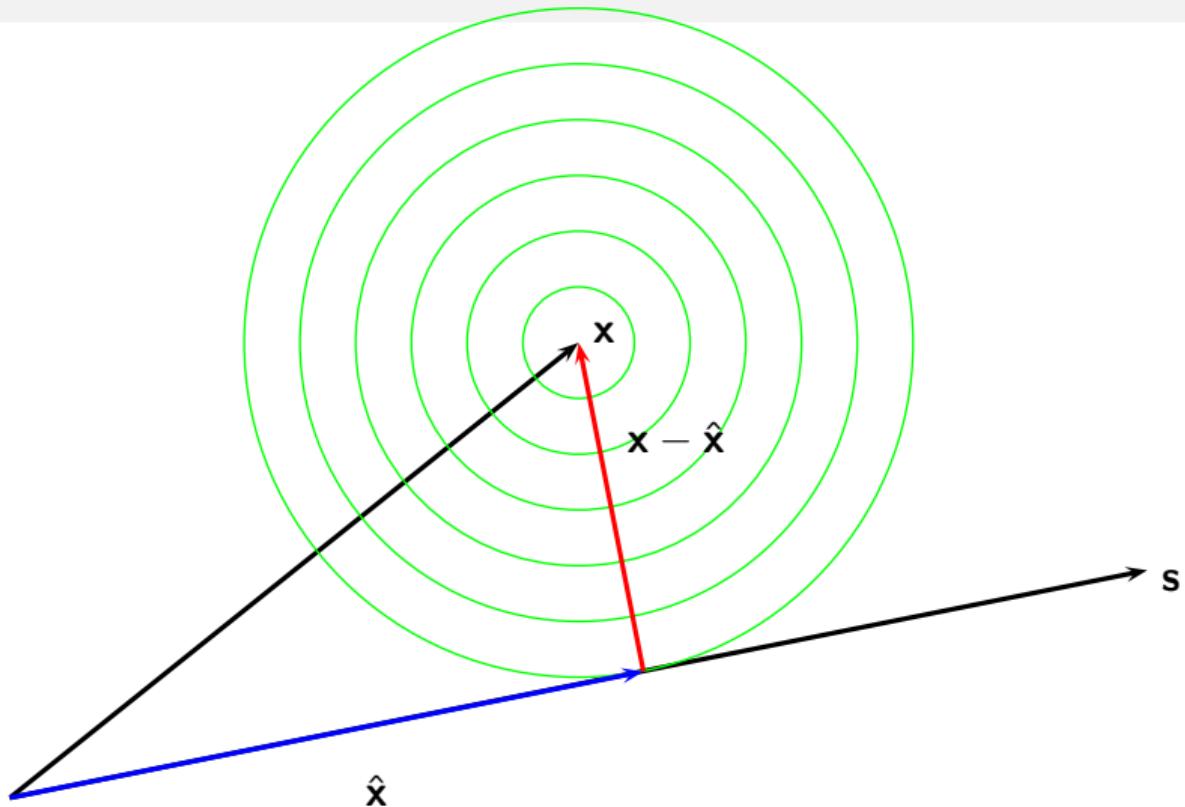
Least Squares Approximation



Least Squares Approximation



Least Squares Approximation



Example: polynomial approximation

- ▶ vector space $P_N[-1, 1] \subset L_2[-1, 1]$
- ▶ $\mathbf{p} = a_0 + a_1 t + \dots + a_N t^N$
- ▶ a self-evident, naive basis: $\mathbf{s}^{(k)} = t^k, \quad k = 0, 1, \dots, N$
- ▶ naive basis is not orthonormal

Example: polynomial approximation

- ▶ vector space $P_N[-1, 1] \subset L_2[-1, 1]$
- ▶ $\mathbf{p} = a_0 + a_1 t + \dots + a_N t^N$
- ▶ a self-evident, naive basis: $\mathbf{s}^{(k)} = t^k, \quad k = 0, 1, \dots, N$
- ▶ naive basis is not orthonormal

Example: polynomial approximation

- ▶ vector space $P_N[-1, 1] \subset L_2[-1, 1]$
- ▶ $\mathbf{p} = a_0 + a_1 t + \dots + a_N t^N$
- ▶ a self-evident, naive basis: $\mathbf{s}^{(k)} = t^k, \quad k = 0, 1, \dots, N$
- ▶ naive basis is not orthonormal

Example: polynomial approximation

- ▶ vector space $P_N[-1, 1] \subset L_2[-1, 1]$
- ▶ $\mathbf{p} = a_0 + a_1 t + \dots + a_N t^N$
- ▶ a self-evident, naive basis: $\mathbf{s}^{(k)} = t^k, \quad k = 0, 1, \dots, N$
- ▶ naive basis is not orthonormal

Example: polynomial approximation

goal: approximate $\mathbf{x} = \sin t \in L_2[-1, 1]$ over $P_2[-1, 1]$

- ▶ build orthonormal basis from naive basis
- ▶ project \mathbf{x} over the orthonormal basis
- ▶ compute approximation error
- ▶ compare error to Taylor approximation (well known but not optimal over the interval)

Example: polynomial approximation

goal: approximate $\mathbf{x} = \sin t \in L_2[-1, 1]$ over $P_2[-1, 1]$

- ▶ build orthonormal basis from naive basis
- ▶ project \mathbf{x} over the orthonormal basis
- ▶ compute approximation error
- ▶ compare error to Taylor approximation (well known but not optimal over the interval)

Example: polynomial approximation

goal: approximate $\mathbf{x} = \sin t \in L_2[-1, 1]$ over $P_2[-1, 1]$

- ▶ build orthonormal basis from naive basis
- ▶ project \mathbf{x} over the orthonormal basis
- ▶ compute approximation error
- ▶ compare error to Taylor approximation (well known but not optimal over the interval)

Example: polynomial approximation

goal: approximate $\mathbf{x} = \sin t \in L_2[-1, 1]$ over $P_2[-1, 1]$

- ▶ build orthonormal basis from naive basis
- ▶ project \mathbf{x} over the orthonormal basis
- ▶ compute approximation error
- ▶ compare error to Taylor approximation (well known but not optimal over the interval)

Example: polynomial approximation

goal: approximate $\mathbf{x} = \sin t \in L_2[-1, 1]$ over $P_2[-1, 1]$

- ▶ build orthonormal basis from naive basis
- ▶ project \mathbf{x} over the orthonormal basis
- ▶ compute approximation error
- ▶ compare error to Taylor approximation (well known but not optimal over the interval)

Building an orthonormal basis

Gram-Schmidt orthonormalization procedure:

$$\{\mathbf{s}^{(k)}\} \longrightarrow \{\mathbf{u}^{(k)}\}$$

original set orthonormal set

Algorithmic procedure: at each step k

$$1. \mathbf{p}^{(k)} = \mathbf{s}^{(k)} - \sum_{n=0}^{k-1} \langle \mathbf{u}^{(n)}, \mathbf{s}^{(k)} \rangle \mathbf{u}^{(n)}$$

$$2. \mathbf{u}^{(k)} = \mathbf{p}^{(k)} / \|\mathbf{p}^{(k)}\|$$

Building an orthonormal basis

Gram-Schmidt orthonormalization procedure:

$$\{\mathbf{s}^{(k)}\} \xrightarrow{\hspace{1cm}} \{\mathbf{u}^{(k)}\}$$

original set orthonormal set

Algorithmic procedure: at each step k

1. $\mathbf{p}^{(k)} = \mathbf{s}^{(k)} - \sum_{n=0}^{k-1} \langle \mathbf{u}^{(n)}, \mathbf{s}^{(k)} \rangle \mathbf{u}^{(n)}$

2. $\mathbf{u}^{(k)} = \mathbf{p}^{(k)} / \|\mathbf{p}^{(k)}\|$

Building an orthonormal basis

Gram-Schmidt orthonormalization procedure:

$$\{\mathbf{s}^{(k)}\} \xrightarrow{\hspace{1cm}} \{\mathbf{u}^{(k)}\}$$

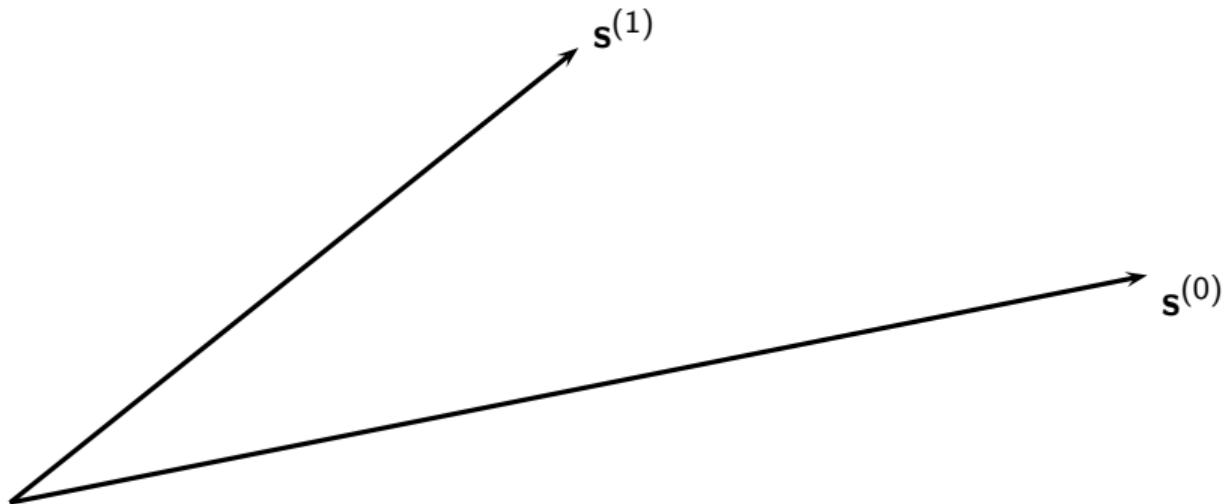
original set orthonormal set

Algorithmic procedure: at each step k

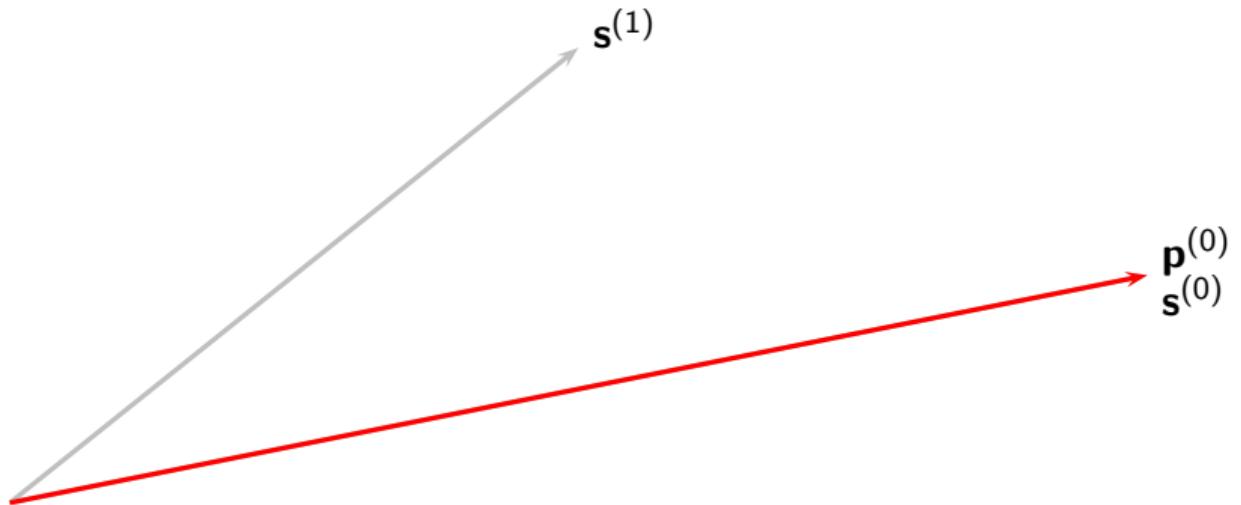
1. $\mathbf{p}^{(k)} = \mathbf{s}^{(k)} - \sum_{n=0}^{k-1} \langle \mathbf{u}^{(n)}, \mathbf{s}^{(k)} \rangle \mathbf{u}^{(n)}$

2. $\mathbf{u}^{(k)} = \mathbf{p}^{(k)} / \|\mathbf{p}^{(k)}\|$

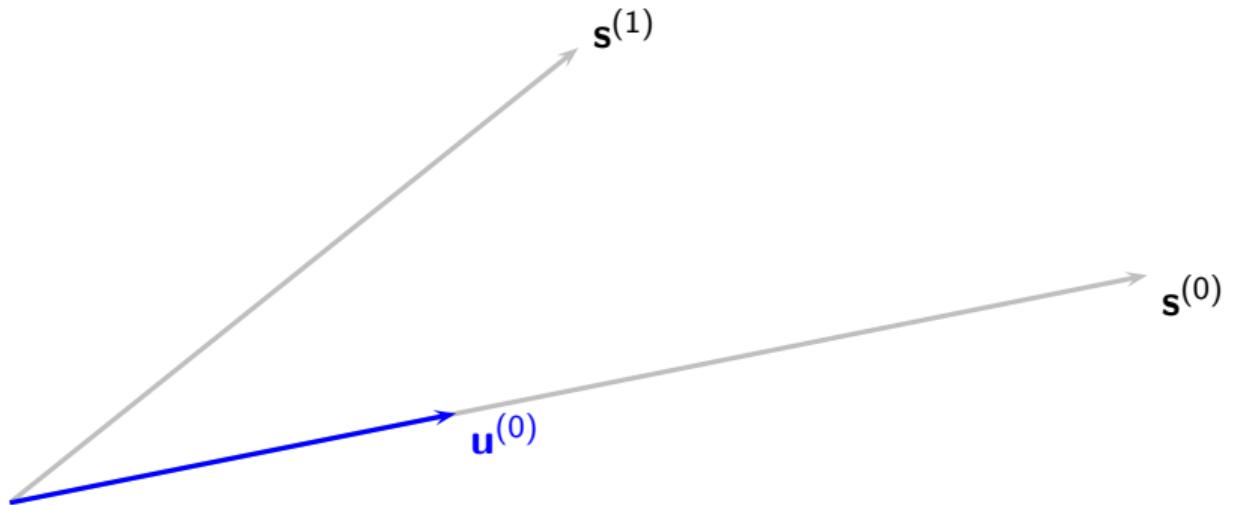
Building an orthonormal basis



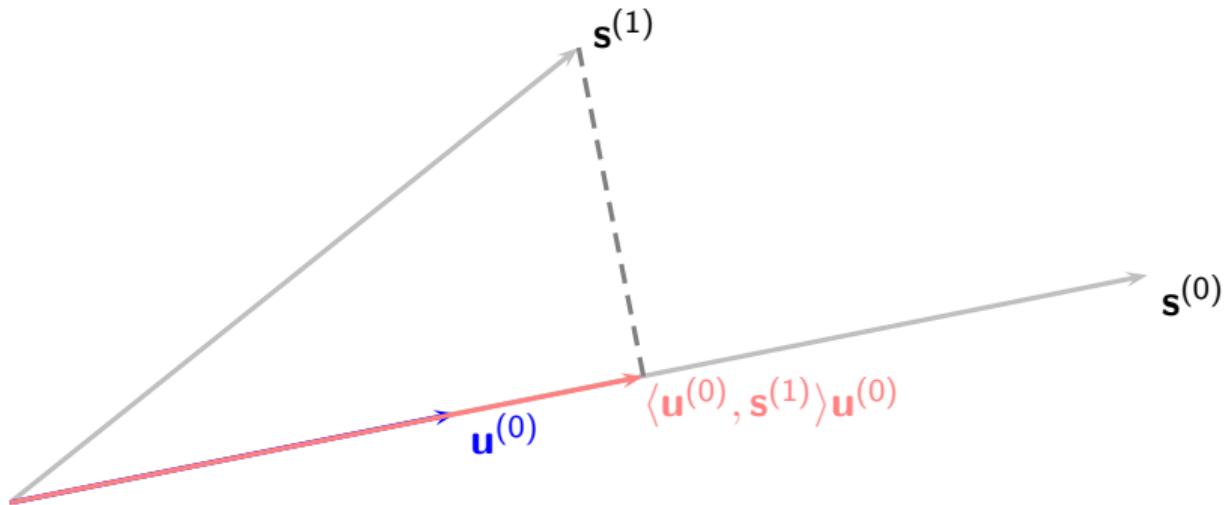
Building an orthonormal basis



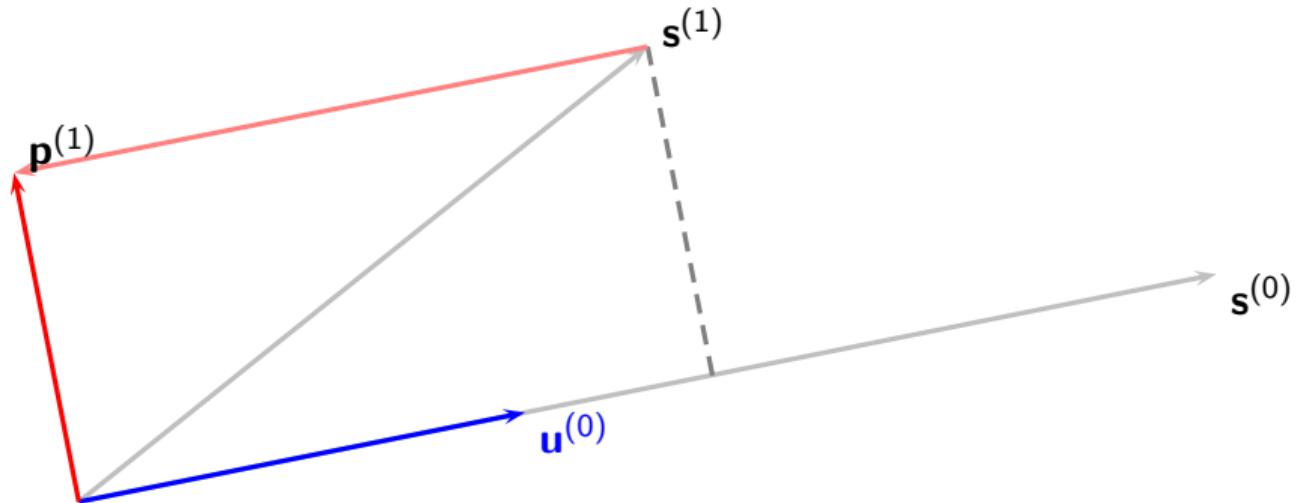
Building an orthonormal basis



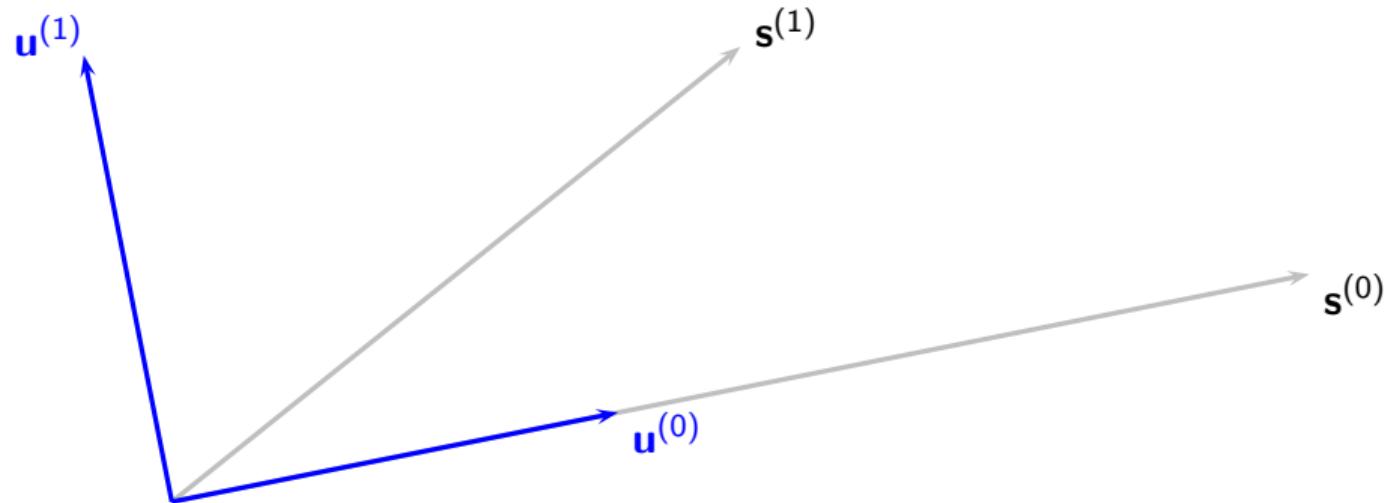
Building an orthonormal basis



Building an orthonormal basis



Building an orthonormal basis



Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2}t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Building an orthonormal basis

Gram-Schmidt orthonormalization of the naive basis: $\{\mathbf{s}^{(k)}\} \rightarrow \{\mathbf{u}^{(k)}\}$

► $\mathbf{s}^{(0)} = 1$

- $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = 1$
- $\|\mathbf{p}^{(0)}\|^2 = 2$
- $\mathbf{u}^{(0)} = \mathbf{p}^{(0)} / \|\mathbf{p}^{(0)}\| = \sqrt{1/2}$

► $\mathbf{s}^{(1)} = t$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(1)} \rangle = \int_{-1}^1 t / \sqrt{2} = 0$
- $\mathbf{p}^{(1)} = \mathbf{s}^{(1)} = t$
- $\|\mathbf{p}^{(1)}\|^2 = 2/3$
- $\mathbf{u}^{(1)} = \sqrt{3/2} t$

► $\mathbf{s}^{(2)} = t^2$

- $\langle \mathbf{u}^{(0)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^2 / \sqrt{2} = 2/3\sqrt{2}$
- $\langle \mathbf{u}^{(1)}, \mathbf{s}^{(2)} \rangle = \int_{-1}^1 t^3 / \sqrt{2} = 0$
- $\mathbf{p}^{(2)} = \mathbf{s}^{(2)} - (2/3\sqrt{2})\mathbf{u}^{(0)} = t^2 - 1/3$
- $\|\mathbf{p}^{(2)}\|^2 = 8/45$
- $\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$

Legendre polynomials

The Gram-Schmidt algorithm leads to an orthonormal basis for $P_N([-1, 1])$

$$\mathbf{u}^{(0)} = \sqrt{1/2}$$

$$\mathbf{u}^{(1)} = \sqrt{3/2} t$$

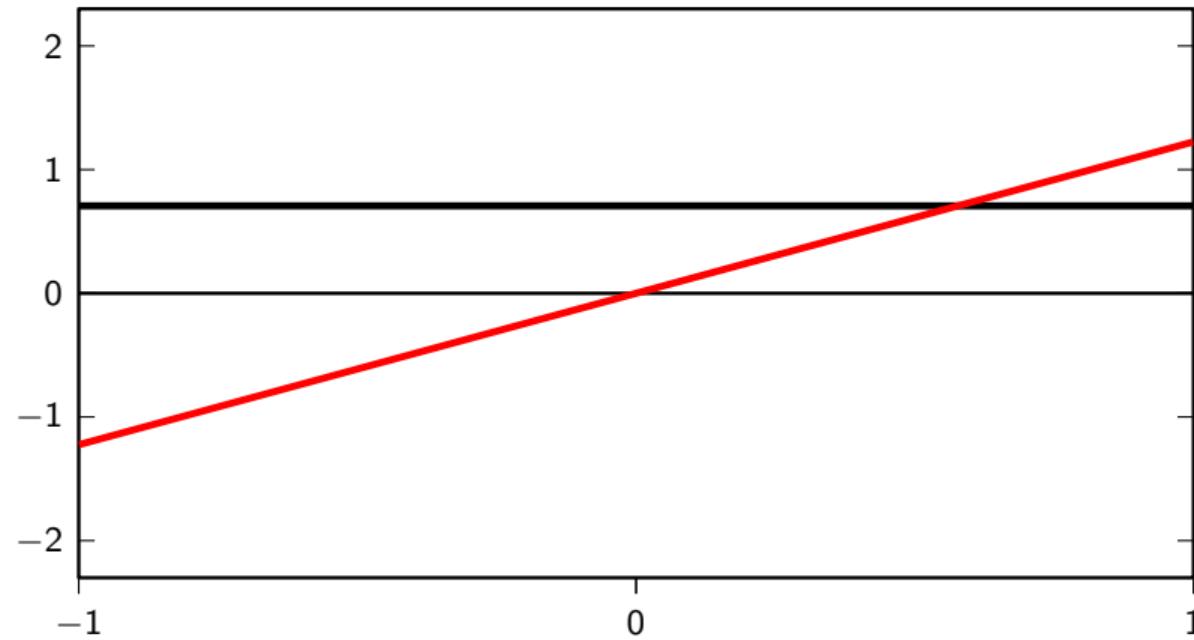
$$\mathbf{u}^{(2)} = \sqrt{5/8}(3t^2 - 1)$$

$$\mathbf{u}^{(3)} = \dots$$

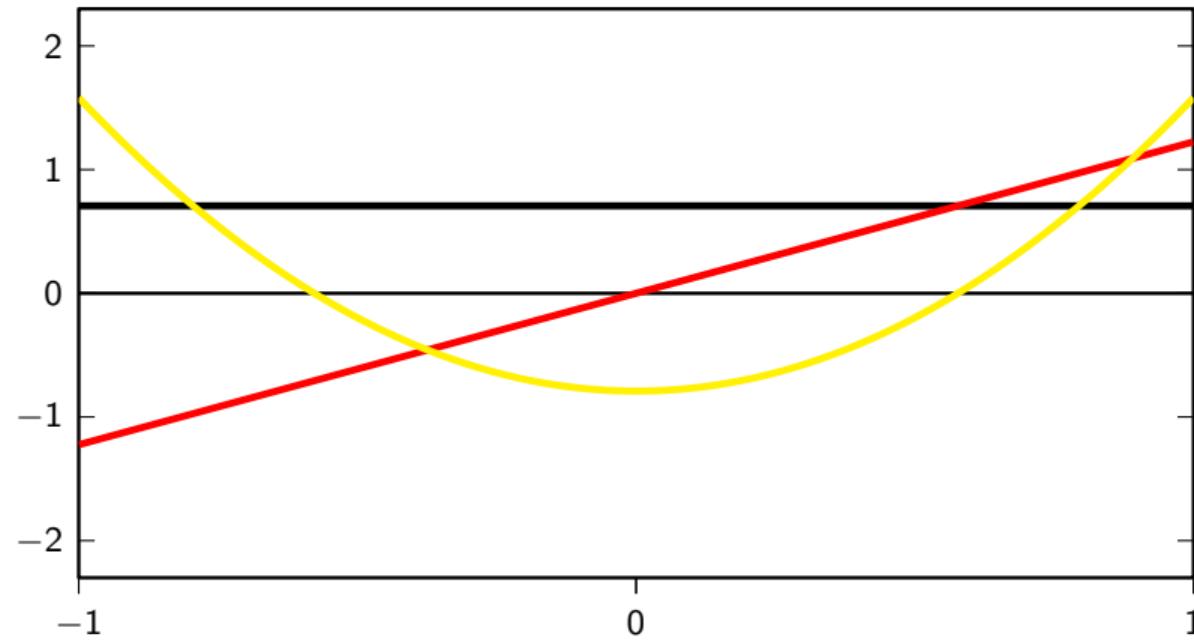
Legendre Polynomials



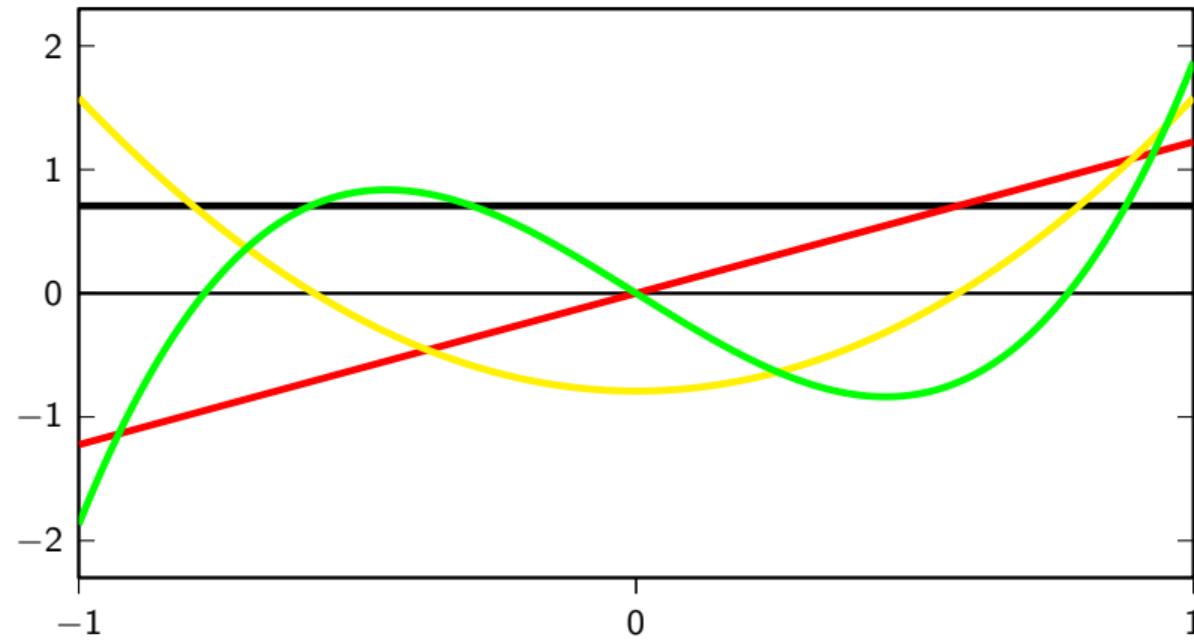
Legendre Polynomials



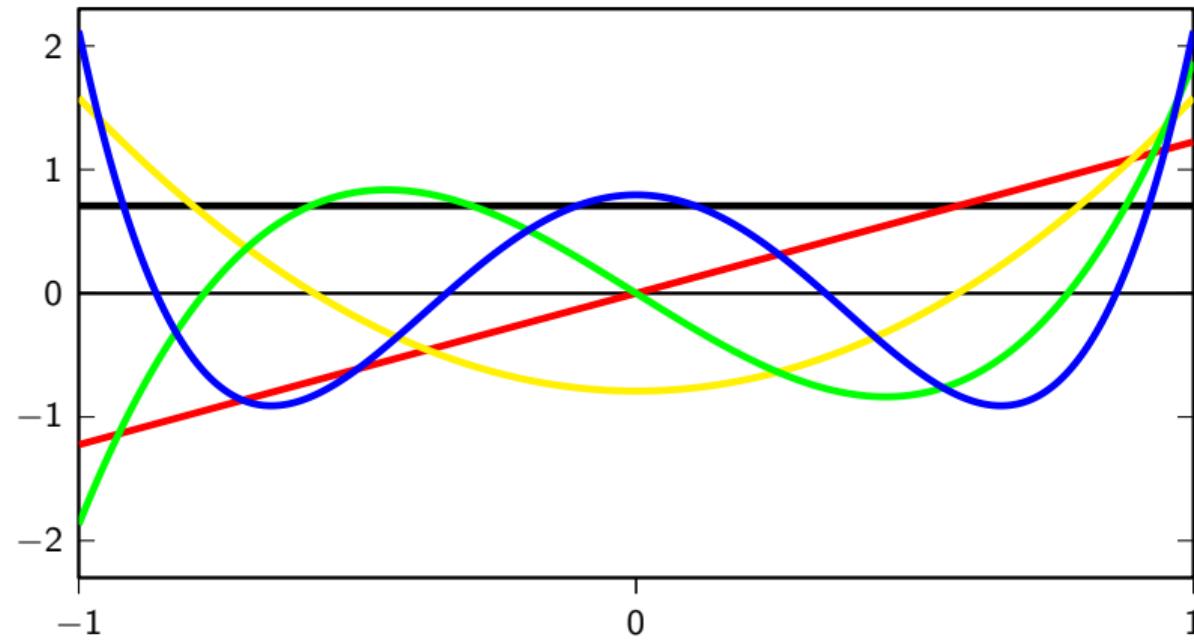
Legendre Polynomials



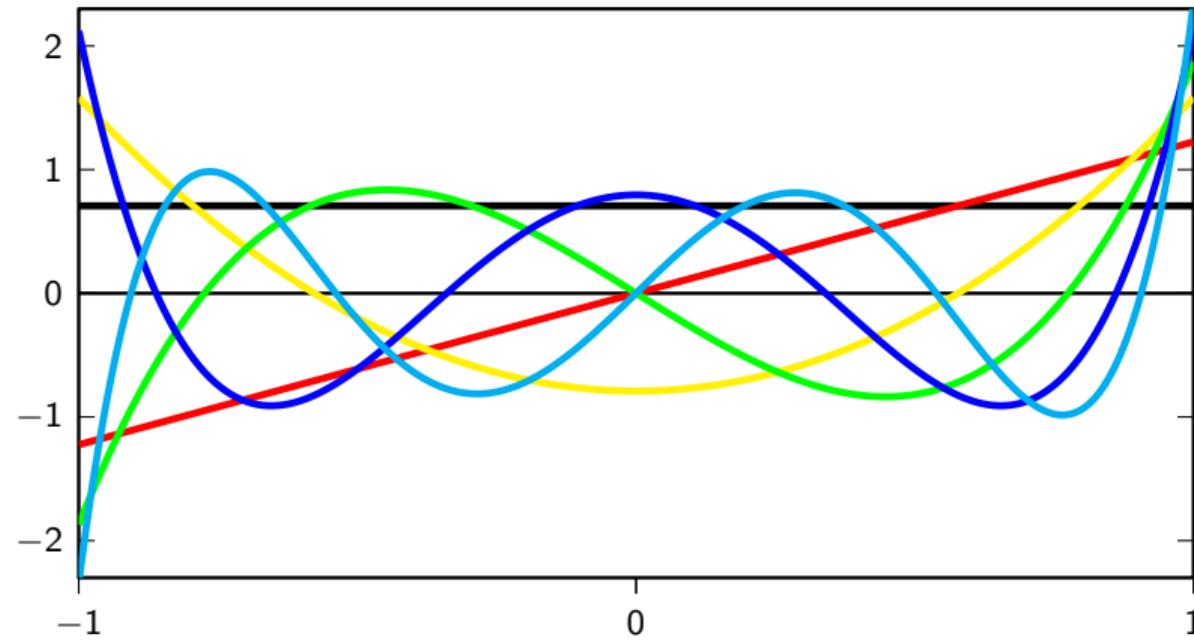
Legendre Polynomials



Legendre Polynomials



Legendre Polynomials



Orthogonal projection over $P_2[-1, 1]$

$$\alpha_k = \langle \mathbf{u}^{(k)}, \mathbf{x} \rangle = \int_{-1}^1 u_k(t) \sin t \, dt$$

- ▶ $\alpha_0 = \langle \sqrt{1/2}, \sin t \rangle = 0$
- ▶ $\alpha_1 = \langle \sqrt{3/2} t, \sin t \rangle \approx 0.7377$
- ▶ $\alpha_2 = \langle \sqrt{5/8}(3t^2 - 1), \sin t \rangle = 0$

Orthogonal projection over $P_2[-1, 1]$

$$\alpha_k = \langle \mathbf{u}^{(k)}, \mathbf{x} \rangle = \int_{-1}^1 u_k(t) \sin t \, dt$$

- ▶ $\alpha_0 = \langle \sqrt{1/2}, \sin t \rangle = 0$
- ▶ $\alpha_1 = \langle \sqrt{3/2} t, \sin t \rangle \approx 0.7377$
- ▶ $\alpha_2 = \langle \sqrt{5/8}(3t^2 - 1), \sin t \rangle = 0$

Orthogonal projection over $P_2[-1, 1]$

$$\alpha_k = \langle \mathbf{u}^{(k)}, \mathbf{x} \rangle = \int_{-1}^1 u_k(t) \sin t \, dt$$

- ▶ $\alpha_0 = \langle \sqrt{1/2}, \sin t \rangle = 0$
- ▶ $\alpha_1 = \langle \sqrt{3/2} t, \sin t \rangle \approx 0.7377$
- ▶ $\alpha_2 = \langle \sqrt{5/8}(3t^2 - 1), \sin t \rangle = 0$

Approximation

Using the orthogonal projection over $P_2[-1, 1]$:

$$\sin t \rightarrow \alpha_1 \mathbf{u}^{(1)} \approx 0.9035 t$$

Using Taylor's series:

$$\sin t \approx t$$

Approximation

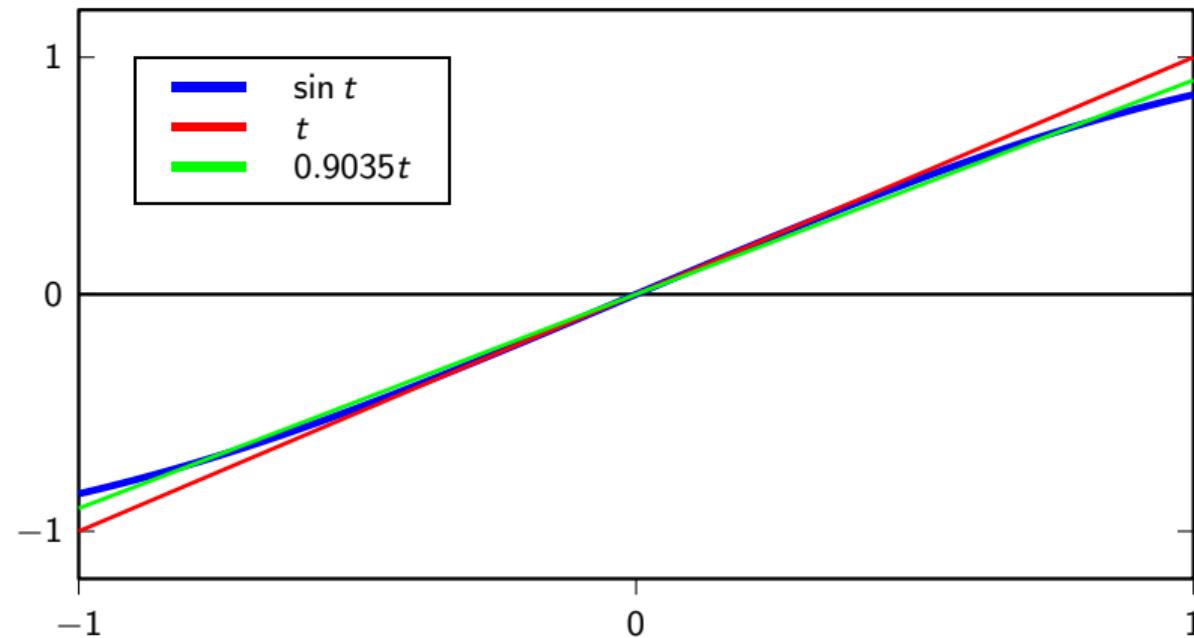
Using the orthogonal projection over $P_2[-1, 1]$:

$$\sin t \rightarrow \alpha_1 \mathbf{u}^{(1)} \approx 0.9035 t$$

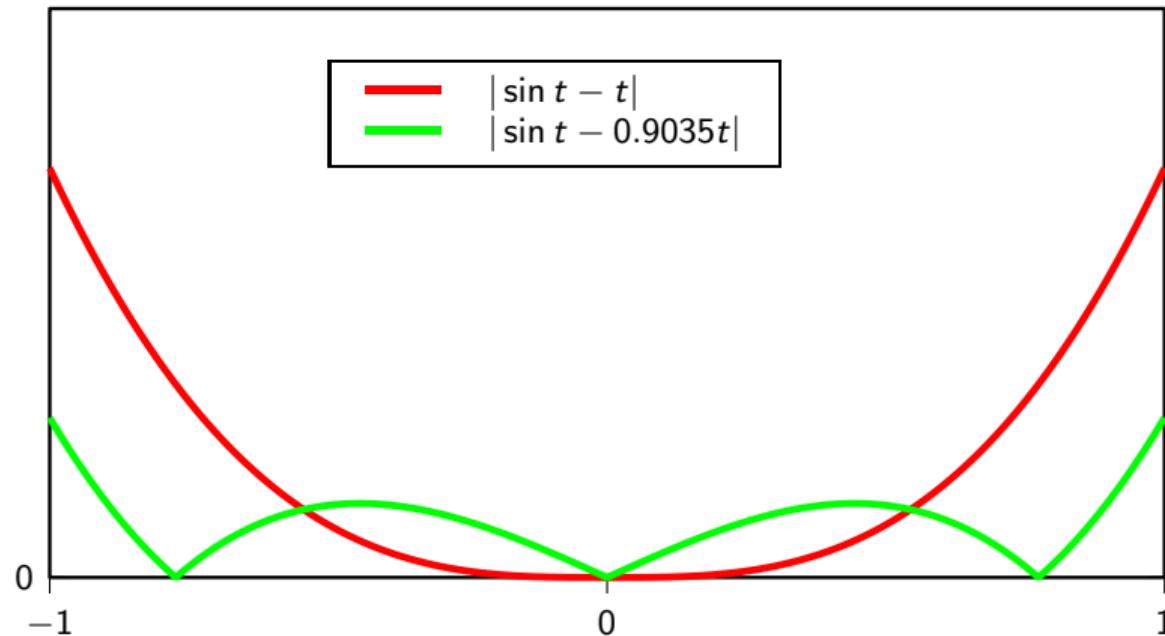
Using Taylor's series:

$$\sin t \approx t$$

Sine approximation



Approximation error



Error norm

Orthogonal projection over $P_2[-1, 1]$:

$$\|\sin t - \alpha_1 \mathbf{u}^{(1)}\| \approx 0.0337$$

Taylor series:

$$\|\sin t - t\| \approx 0.0857$$

Error norm

Orthogonal projection over $P_2[-1, 1]$:

$$\|\sin t - \alpha_1 \mathbf{u}^{(1)}\| \approx 0.0337$$

Taylor series:

$$\|\sin t - t\| \approx 0.0857$$

Hilbert space

Hilbert Space – the ingredients:

1. a vector space: $H(V, \mathbb{C})$
2. an inner product: $\langle \cdot, \cdot \rangle : V \times V \rightarrow \mathbb{C}$
3. completeness

Hilbert Space – the ingredients:

1. a vector space: $H(V, \mathbb{C})$
2. an inner product: $\langle \cdot, \cdot \rangle : V \times V \rightarrow \mathbb{C}$
3. completeness

Hilbert Space – the ingredients:

1. a vector space: $H(V, \mathbb{C})$
2. an inner product: $\langle \cdot, \cdot \rangle : V \times V \rightarrow \mathbb{C}$
3. completeness

Completeness

limiting operations must yield vector space elements

Example of an *incomplete* space: the set of rational numbers

$$x_n = \sum_{k=0}^n \frac{1}{k!} \in \mathbb{Q} \quad \text{but} \quad \lim_{n \rightarrow \infty} x_n = e \notin \mathbb{Q}$$

Completeness

limiting operations must yield vector space elements

Example of an *incomplete* space: the set of rational numbers

$$x_n = \sum_{k=0}^n \frac{1}{k!} \in \mathbb{Q} \quad \text{but} \quad \lim_{n \rightarrow \infty} x_n = e \notin \mathbb{Q}$$

Completeness

limiting operations must yield vector space elements

Example of an *incomplete* space: the set of rational numbers

$$x_n = \sum_{k=0}^n \frac{1}{k!} \in \mathbb{Q} \quad \text{but} \quad \lim_{n \rightarrow \infty} x_n = e \notin \mathbb{Q}$$



Signals in Hilbert Space

Why did we do all this?

- ▶ finite-length and periodic signals live in \mathbb{C}^N
- ▶ infinite-length signals live in $\ell_2(\mathbb{Z})$
- ▶ different bases are different observation tools for signals
- ▶ subspace projections are useful in filtering and compression

Signals in Hilbert Space

Why did we do all this?

- ▶ finite-length and periodic signals live in \mathbb{C}^N
- ▶ infinite-length signals live in $\ell_2(\mathbb{Z})$

- ▶ different bases are different observation tools for signals
- ▶ subspace projections are useful in filtering and compression

Signals in Hilbert Space

Why did we do all this?

- ▶ finite-length and periodic signals live in \mathbb{C}^N
- ▶ infinite-length signals live in $\ell_2(\mathbb{Z})$

- ▶ different bases are different observation tools for signals
- ▶ subspace projections are useful in filtering and compression

Signals in Hilbert Space

Why did we do all this?

- ▶ finite-length and periodic signals live in \mathbb{C}^N
- ▶ infinite-length signals live in $\ell_2(\mathbb{Z})$

- ▶ different bases are different observation tools for signals
- ▶ subspace projections are useful in filtering and compression

COM303: Digital Signal Processing

Lecture 4: Introduction to Fourier Analysis

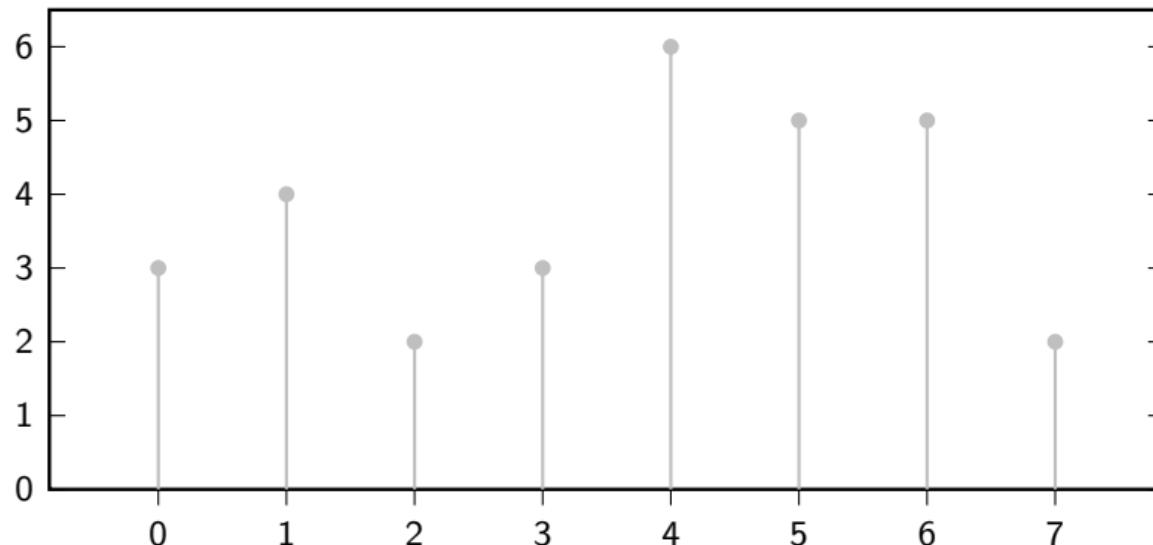
Overview

- ▶ Fourier analysis: concept and motivation
- ▶ the complex exponential
- ▶ the Fourier basis
- ▶ the DFT

The time domain

signals are often expressed as a linear combination of “atomic” time units:

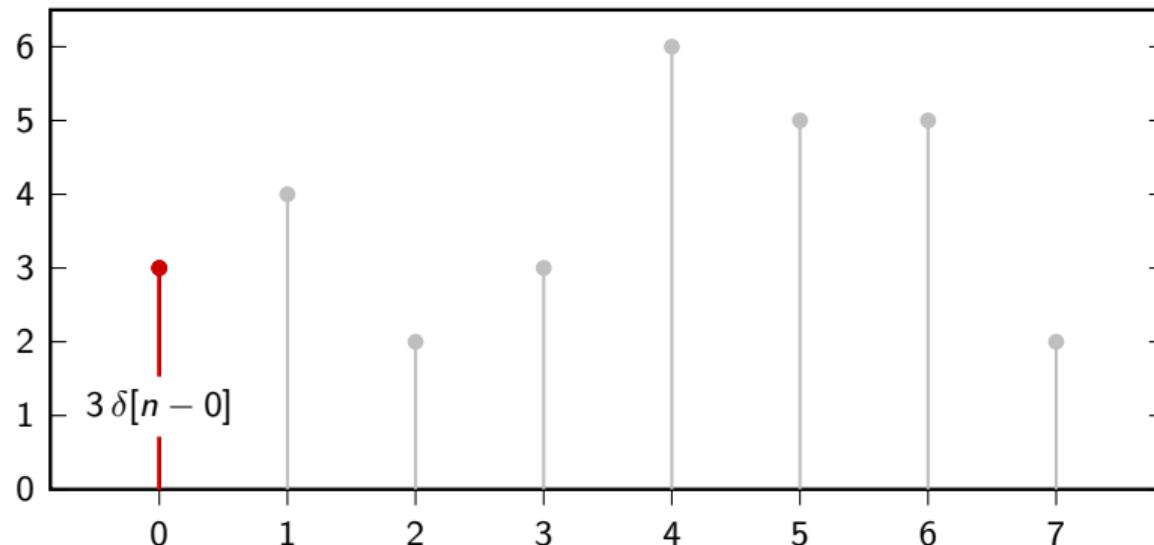
$$x[n] = \sum_{k=0}^{N-1} x[k]\delta[n - k]$$



The time domain

signals are often expressed as a linear combination of “atomic” time units:

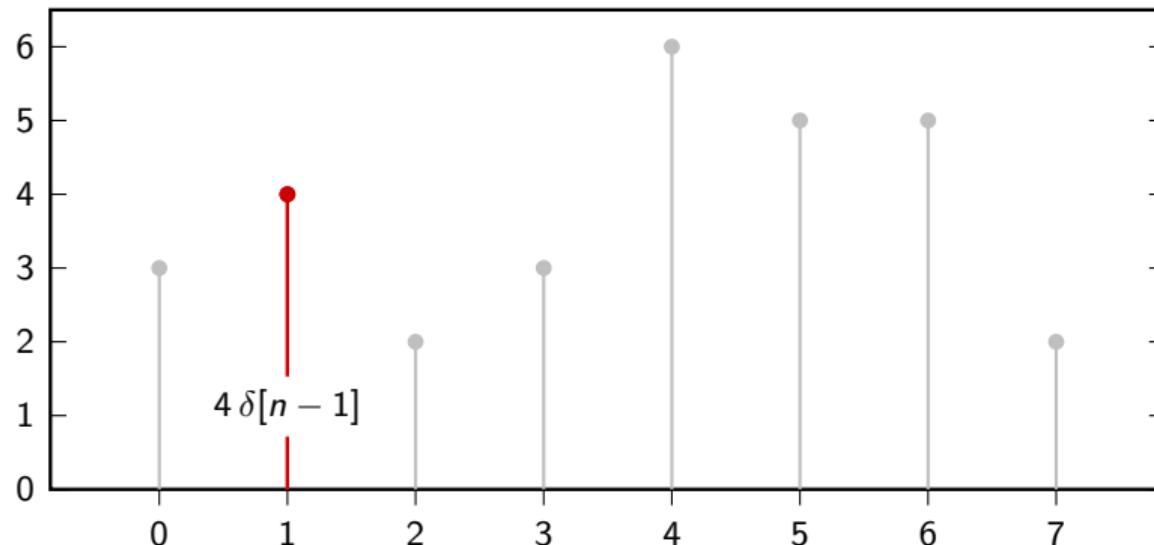
$$x[n] = \sum_{k=0}^{N-1} x[k]\delta[n - k]$$



The time domain

signals are often expressed as a linear combination of “atomic” time units:

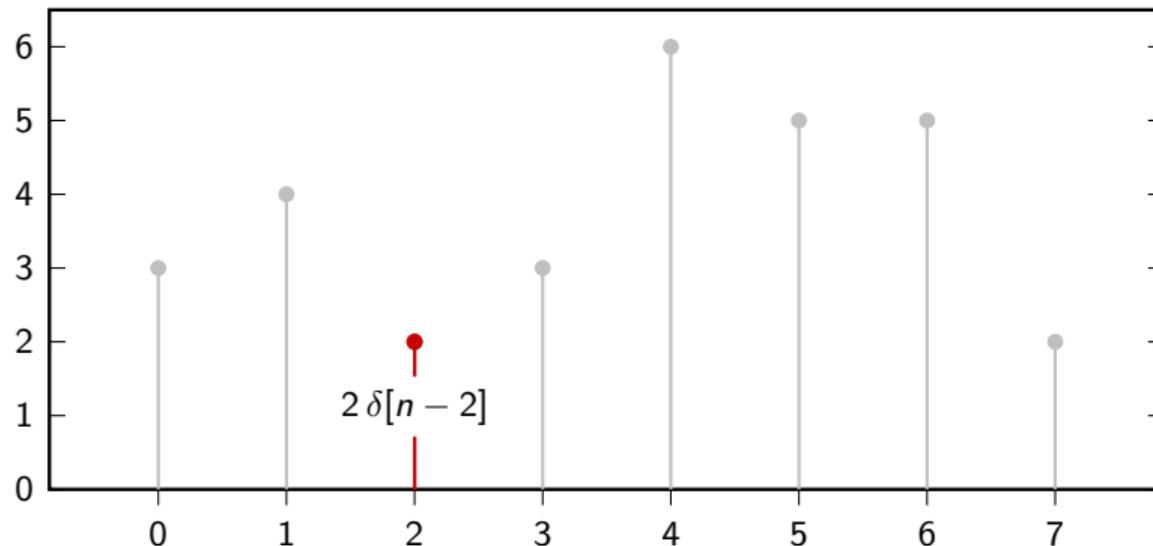
$$x[n] = \sum_{k=0}^{N-1} x[k]\delta[n - k]$$



The time domain

signals are often expressed as a linear combination of “atomic” time units:

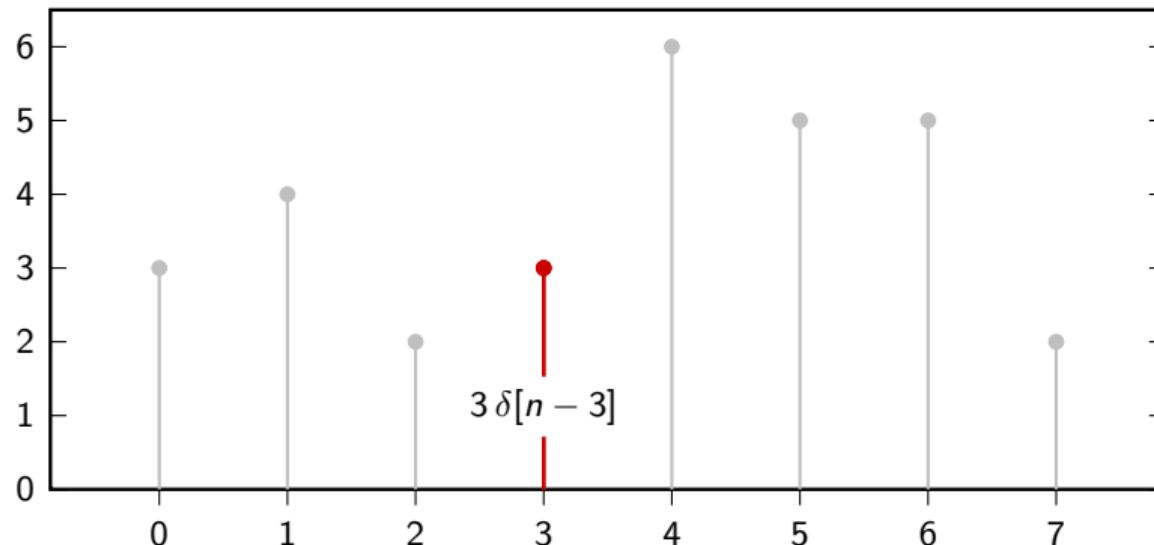
$$x[n] = \sum_{k=0}^{N-1} x[k]\delta[n - k]$$



The time domain

signals are often expressed as a linear combination of “atomic” time units:

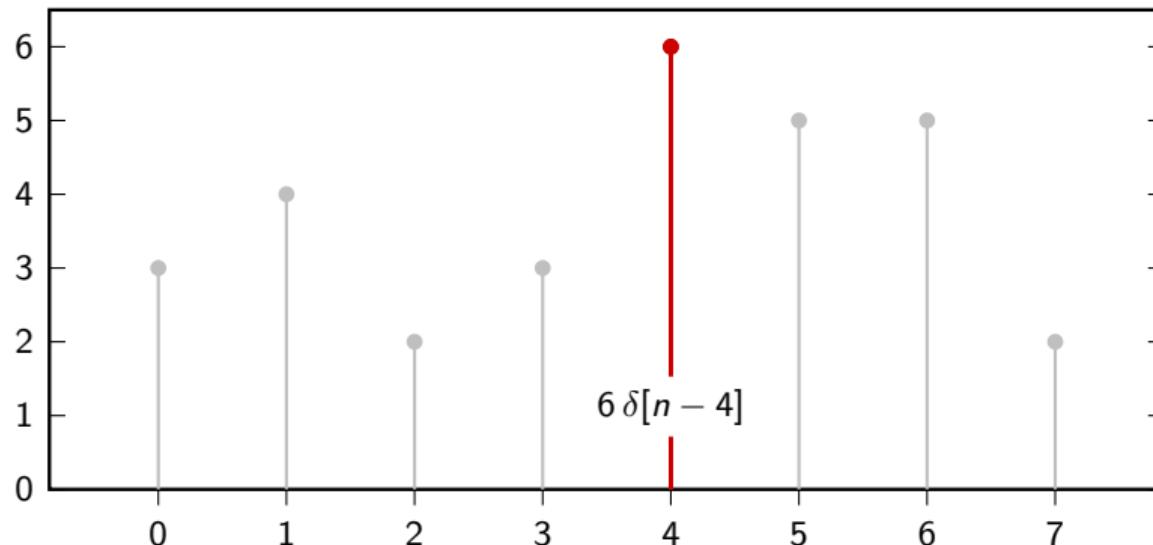
$$x[n] = \sum_{k=0}^{N-1} x[k]\delta[n - k]$$



The time domain

signals are often expressed as a linear combination of “atomic” time units:

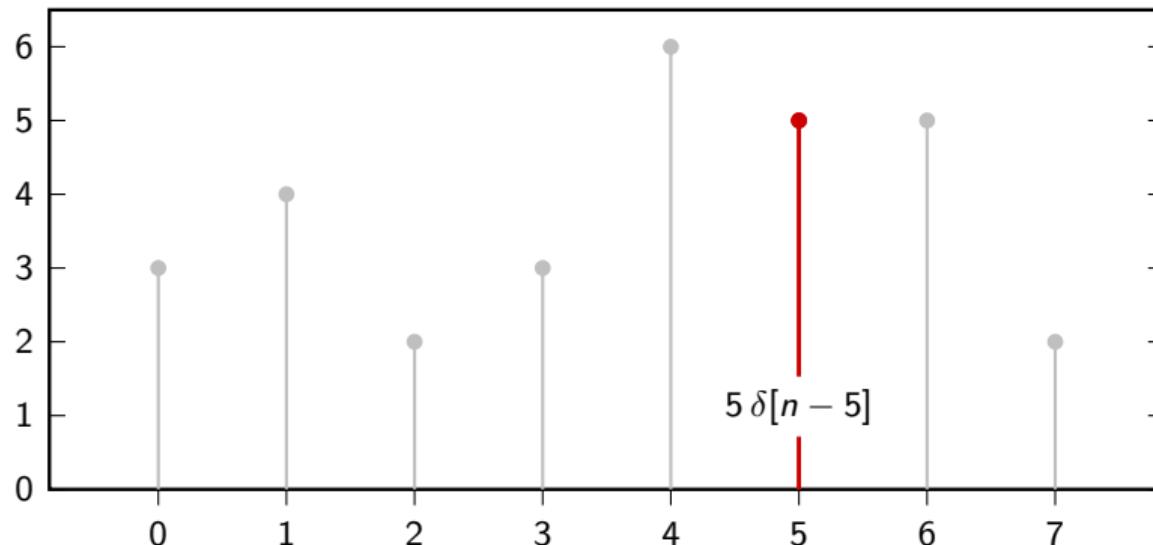
$$x[n] = \sum_{k=0}^{N-1} x[k]\delta[n - k]$$



The time domain

signals are often expressed as a linear combination of “atomic” time units:

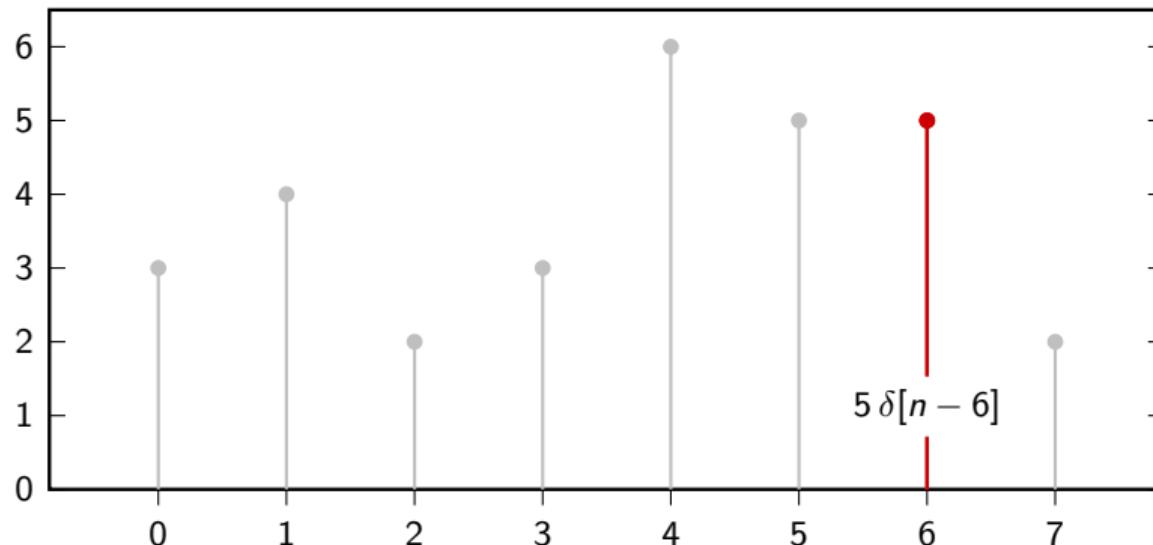
$$x[n] = \sum_{k=0}^{N-1} x[k]\delta[n - k]$$



The time domain

signals are often expressed as a linear combination of “atomic” time units:

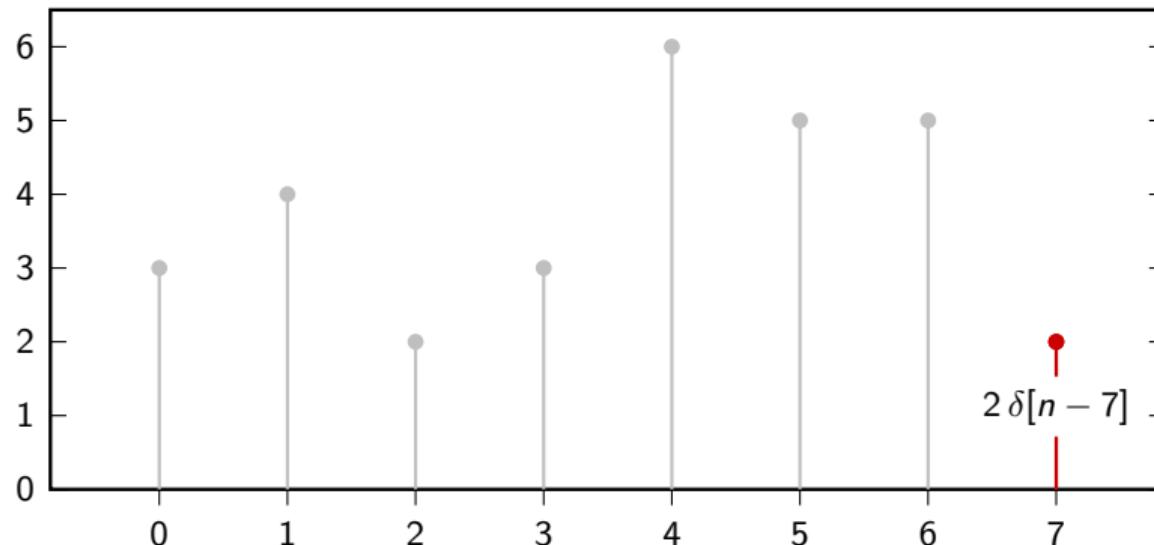
$$x[n] = \sum_{k=0}^{N-1} x[k]\delta[n - k]$$



The time domain

signals are often expressed as a linear combination of “atomic” time units:

$$x[n] = \sum_{k=0}^{N-1} x[k]\delta[n - k]$$



The time domain

in vector notation:

$$\mathbf{x} = \sum_{k=0}^{N-1} x_k \boldsymbol{\delta}^{(k)}$$

where $\{\boldsymbol{\delta}^{(k)}\}$ is the canonical basis for \mathbb{C}^N ; e.g.:

$$\boldsymbol{\delta}^{(2)} = [0 \ 0 \ 1 \ 0 \ \dots \ 0]^T$$

The frequency domain

Fourier analysis: express a signal as a combination of periodic oscillations:

$$\mathbf{x} = \sum_{k=0}^{N-1} X_k \mathbf{w}^{(k)}$$

where $\{\mathbf{w}^{(k)}\}$ is the Fourier basis.

Fourier transform: a change of basis in the space of discrete-time signals

The frequency domain

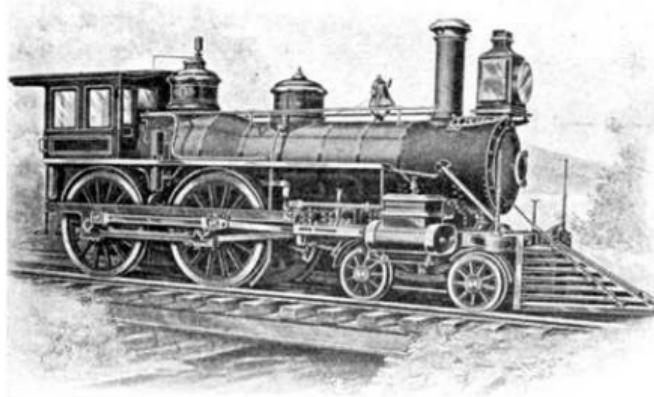
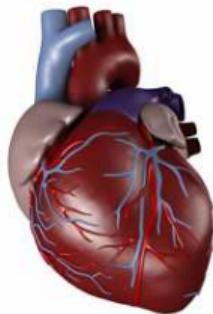
Fourier analysis: express a signal as a combination of periodic oscillations:

$$\mathbf{x} = \sum_{k=0}^{N-1} X_k \mathbf{w}^{(k)}$$

where $\{\mathbf{w}^{(k)}\}$ is the Fourier basis.

Fourier transform: a change of basis in the space of discrete-time signals

Oscillations are everywhere!



Oscillations are everywhere

- ▶ sustainable dynamic systems exhibit oscillatory behavior
- ▶ intuitively: things that don't move in circles can't last:
 - bombs
 - rockets
 - human beings...

Oscillations are everywhere

- ▶ sustainable dynamic systems exhibit oscillatory behavior
- ▶ intuitively: things that don't move in circles can't last:
 - bombs
 - rockets
 - human beings...

Oscillations are everywhere

- ▶ sustainable dynamic systems exhibit oscillatory behavior
- ▶ intuitively: things that don't move in circles can't last:
 - bombs
 - rockets
 - human beings...

Oscillations are everywhere

- ▶ sustainable dynamic systems exhibit oscillatory behavior
- ▶ intuitively: things that don't move in circles can't last:
 - bombs
 - rockets
 - human beings...

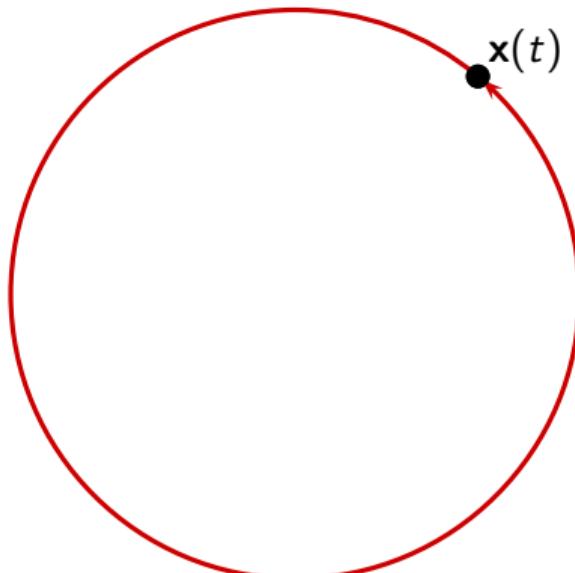
Oscillations are everywhere

- ▶ sustainable dynamic systems exhibit oscillatory behavior
- ▶ intuitively: things that don't move in circles can't last:
 - bombs
 - rockets
 - human beings...

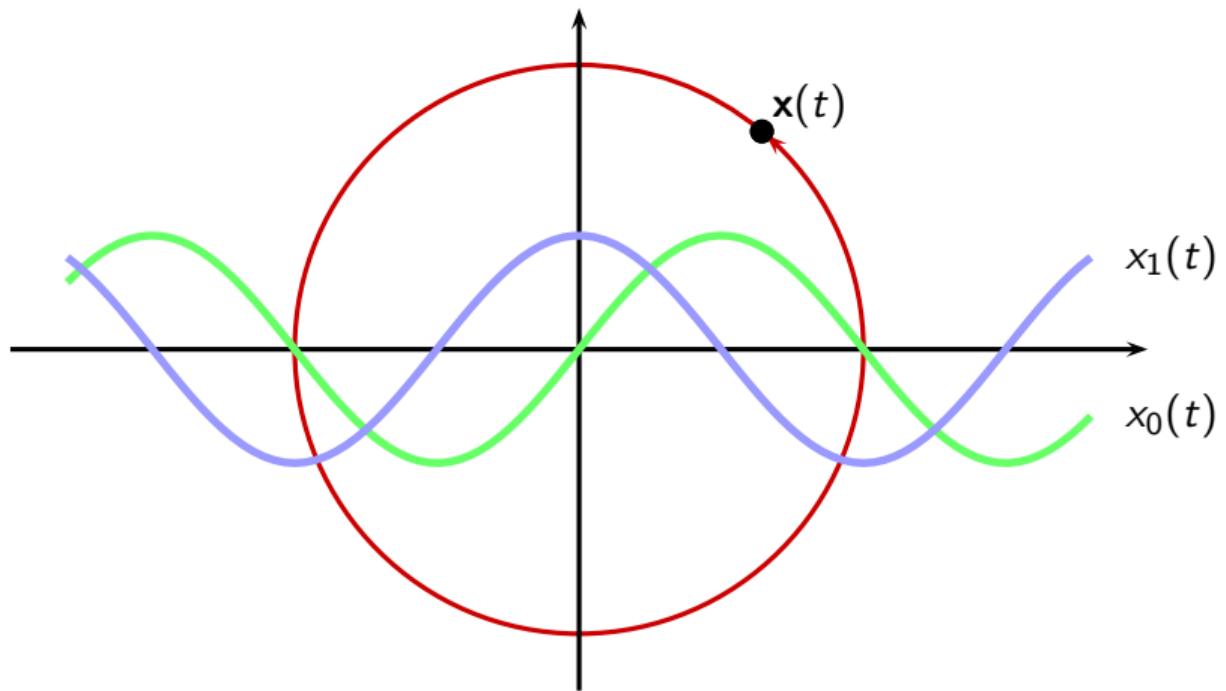
Oscillations are everywhere



Oscillations are everywhere



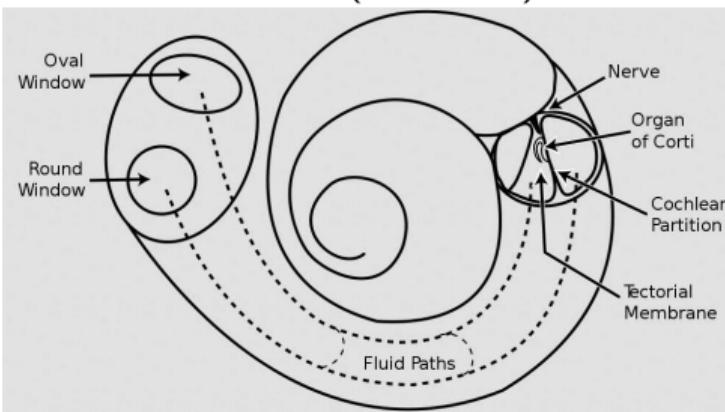
Oscillations are everywhere



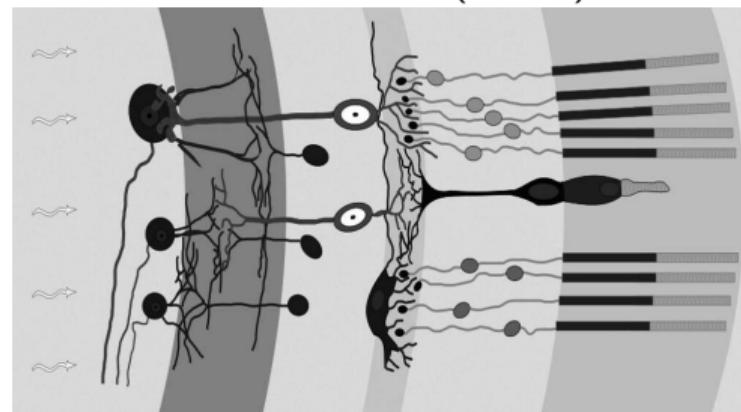
You too can detect sinusoids!

the human body has two receptors for sinusoidal signals:

cochlea (inner ear)



rods and cones (retina)



- ▶ air pressure sinusoids
- ▶ frequencies from 20Hz to 20KHz
- ▶ electromagnetic sinusoids
- ▶ frequencies from 430THz to 790THz

The intuition

- ▶ humans analyze complex signals (audio, images) in terms of their sinusoidal components
- ▶ we can build instruments that “resonate” at one or multiple frequencies (tuning fork vs piano)
- ▶ the “frequency domain” seems to be as important as the time domain

The intuition

- ▶ humans analyze complex signals (audio, images) in terms of their sinusoidal components
- ▶ we can build instruments that “resonate” at one or multiple frequencies (tuning fork vs piano)
- ▶ the “frequency domain” seems to be as important as the time domain

The intuition

- ▶ humans analyze complex signals (audio, images) in terms of their sinusoidal components
- ▶ we can build instruments that “resonate” at one or multiple frequencies (tuning fork vs piano)
- ▶ the “frequency domain” seems to be as important as the time domain

Fundamental question

can we decompose any signal into sinusoidal elements?

yes, and Fourier showed us how to do it *exactly!*

analysis

- ▶ from time domain to frequency domain
- ▶ find the contribution of different frequencies
- ▶ discover “hidden” signal properties

synthesis

- ▶ from frequency domain to time domain
- ▶ create signals with known frequency content
- ▶ fit signals to specific frequency regions

Fundamental question

can we decompose any signal into sinusoidal elements?

yes, and Fourier showed us how to do it *exactly!*

analysis

- ▶ from time domain to frequency domain
- ▶ find the contribution of different frequencies
- ▶ discover “hidden” signal properties

synthesis

- ▶ from frequency domain to time domain
- ▶ create signals with known frequency content
- ▶ fit signals to specific frequency regions

Fundamental question

can we decompose any signal into sinusoidal elements?

yes, and Fourier showed us how to do it *exactly!*

analysis

- ▶ from time domain to frequency domain
- ▶ find the contribution of different frequencies
- ▶ discover “hidden” signal properties

synthesis

- ▶ from frequency domain to time domain
- ▶ create signals with known frequency content
- ▶ fit signals to specific frequency regions

Fundamental question

can we decompose any signal into sinusoidal elements?

yes, and Fourier showed us how to do it *exactly!*

analysis

- ▶ from time domain to frequency domain
- ▶ find the contribution of different frequencies
- ▶ discover “hidden” signal properties

synthesis

- ▶ from frequency domain to time domain
- ▶ create signals with known frequency content
- ▶ fit signals to specific frequency regions

The mathematical setup

- ▶ let's start with finite-length signals (i.e. vectors in \mathbb{C}^N)
- ▶ Fourier analysis is a simple change of basis
- ▶ a change of basis is a change of perspective
- ▶ a change of perspective can reveal things (if the basis is good)

The mathematical setup

- ▶ let's start with finite-length signals (i.e. vectors in \mathbb{C}^N)
- ▶ Fourier analysis is a simple change of basis
- ▶ a change of basis is a change of perspective
- ▶ a change of perspective can reveal things (if the basis is good)

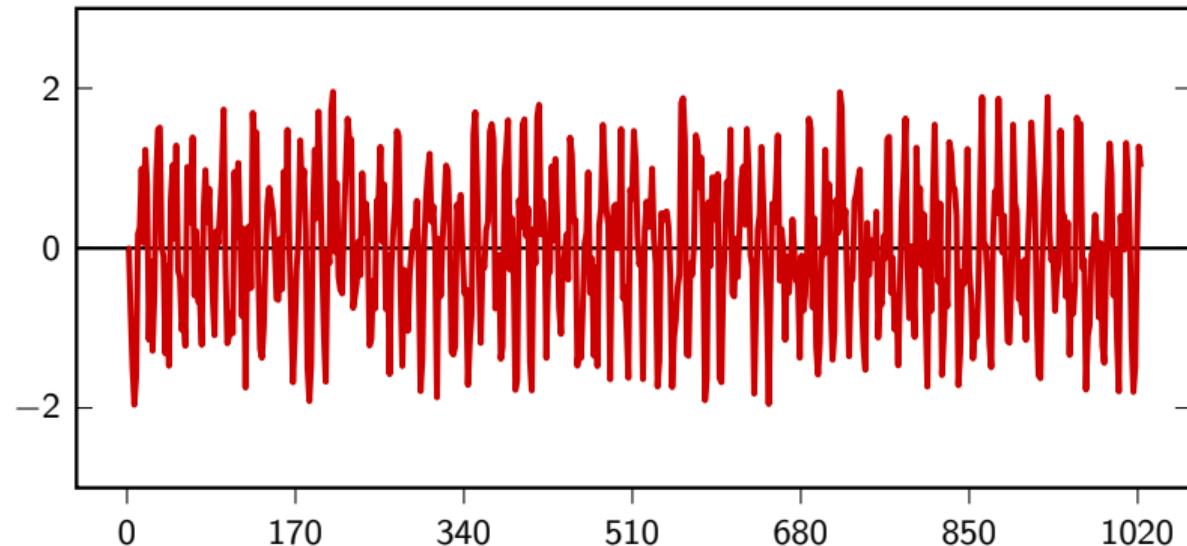
The mathematical setup

- ▶ let's start with finite-length signals (i.e. vectors in \mathbb{C}^N)
- ▶ Fourier analysis is a simple change of basis
- ▶ a change of basis is a change of perspective
- ▶ a change of perspective can reveal things (if the basis is good)

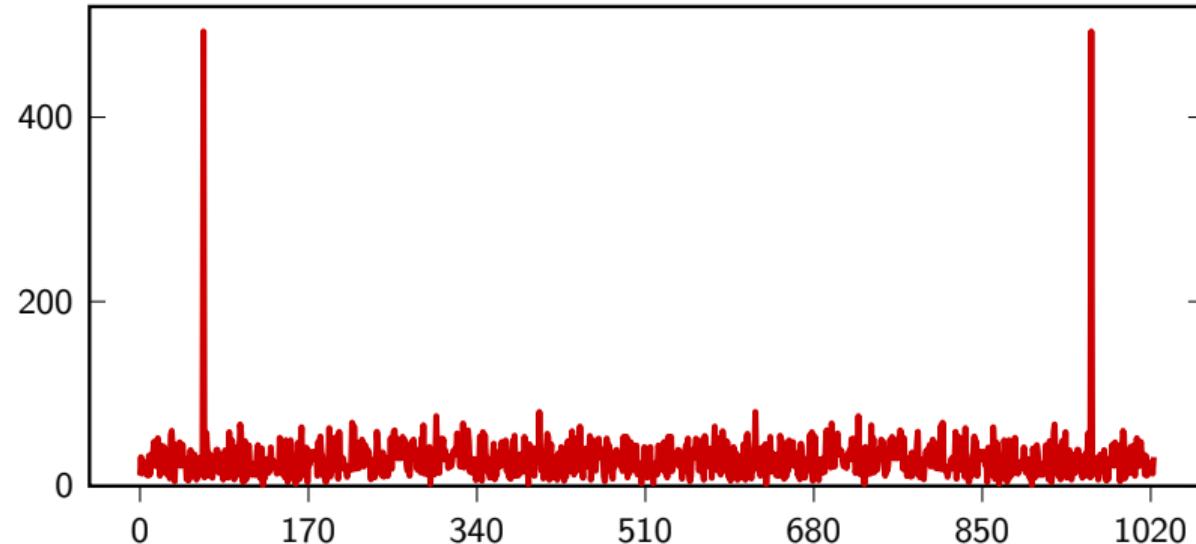
The mathematical setup

- ▶ let's start with finite-length signals (i.e. vectors in \mathbb{C}^N)
- ▶ Fourier analysis is a simple change of basis
- ▶ a change of basis is a change of perspective
- ▶ a change of perspective can reveal things (if the basis is good)

Mystery signal



Mystery signal in the Fourier basis



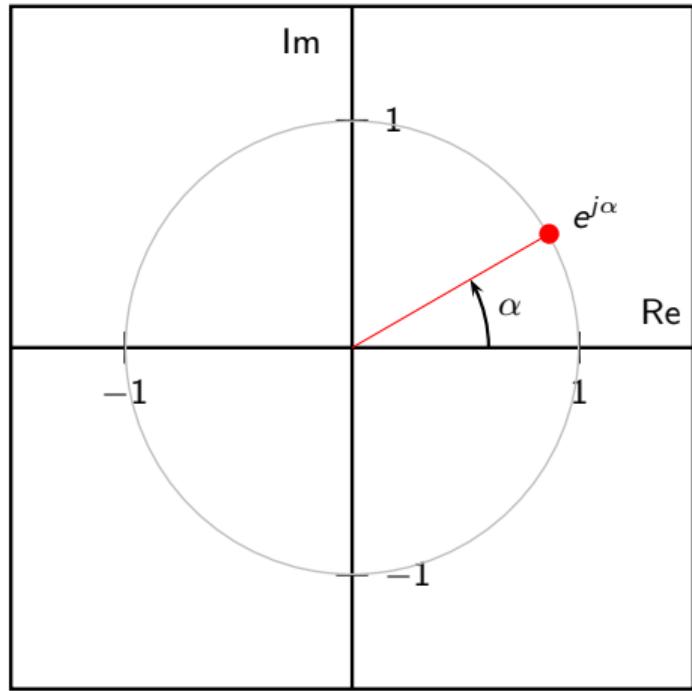
the complex exponential

Prerequisite Warning!

$$e^{j\alpha}$$

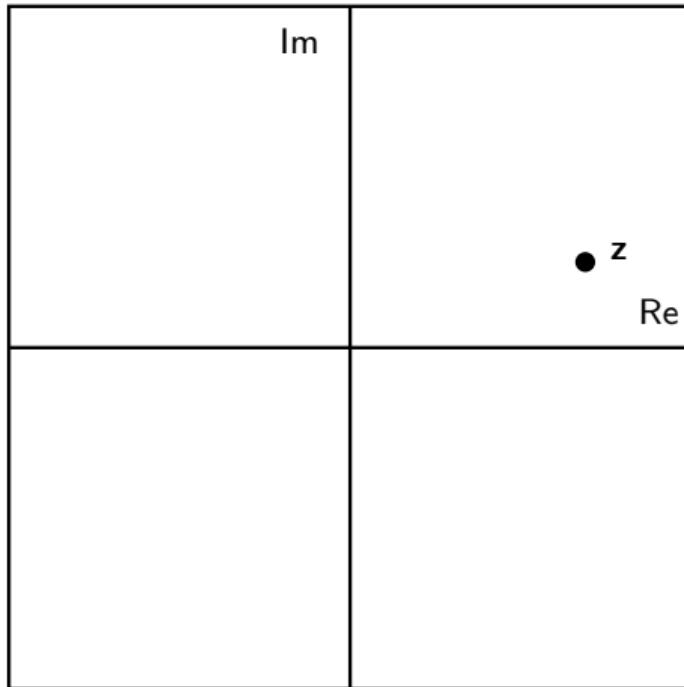
The complex exponential

$$e^{j\alpha} = \cos \alpha + j \sin \alpha$$



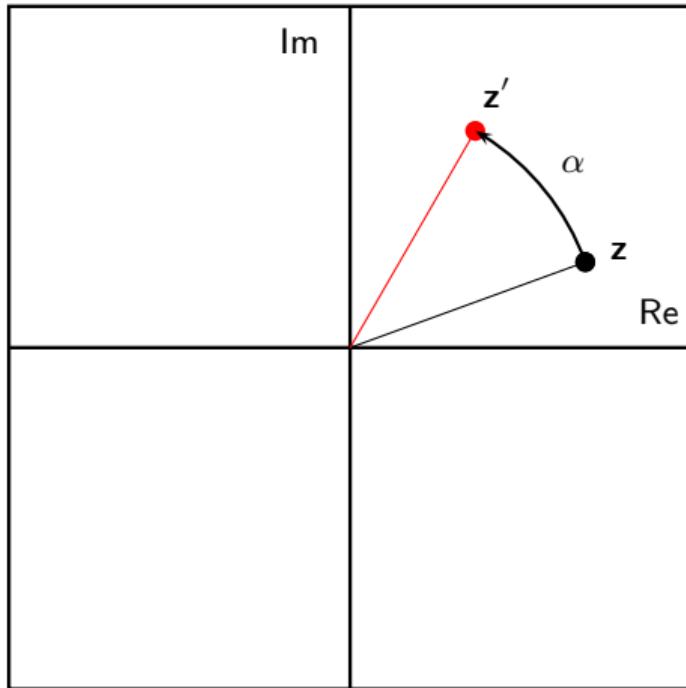
The complex exponential

z : point on the complex plane



The complex exponential

$$\text{rotation: } z' = z e^{j\alpha}$$



The discrete-time oscillatory heartbeat

Ingredients:

- ▶ a frequency ω (units: radians)
- ▶ an initial phase ϕ (units: radians)
- ▶ an amplitude A

$$\begin{aligned}x[n] &= Ae^{j(\omega n + \phi)} \\&= A[\cos(\omega n + \phi) + j \sin(\omega n + \phi)]\end{aligned}$$

The discrete-time oscillatory heartbeat

Ingredients:

- ▶ a frequency ω (units: radians)
- ▶ an initial phase ϕ (units: radians)
- ▶ an amplitude A

$$\begin{aligned}x[n] &= Ae^{j(\omega n + \phi)} \\&= A[\cos(\omega n + \phi) + j \sin(\omega n + \phi)]\end{aligned}$$

Why complex exponentials?

- ▶ we can use complex numbers in digital systems, so why not?
- ▶ it makes sense: every sinusoid can always be written as a sum of sine and cosine
- ▶ math is simpler: trigonometry becomes algebra

Why complex exponentials?

- ▶ we can use complex numbers in digital systems, so why not?
- ▶ it makes sense: every sinusoid can always be written as a sum of sine and cosine
- ▶ math is simpler: trigonometry becomes algebra

Why complex exponentials?

- ▶ we can use complex numbers in digital systems, so why not?
- ▶ it makes sense: every sinusoid can always be written as a sum of sine and cosine
- ▶ math is simpler: trigonometry becomes algebra

The advantages of complex exponentials

Example: change the phase of a cosine the “old-school” way

$$\cos(\omega n + \phi) = a \cos(\omega n) - b \sin(\omega n), \quad a = \cos \phi, \quad b = \sin \phi$$

- ▶ we have to remember complex trigonometric formulas
- ▶ we have to carry more terms in our equations

The advantages of complex exponentials

Example: change the phase of a cosine the “old-school” way

$$\cos(\omega n + \phi) = a \cos(\omega n) - b \sin(\omega n), \quad a = \cos \phi, \quad b = \sin \phi$$

- ▶ we have to remember complex trigonometric formulas
- ▶ we have to carry more terms in our equations

The advantages of complex exponentials

Example: change the phase of a cosine the “old-school” way

$$\cos(\omega n + \phi) = a \cos(\omega n) - b \sin(\omega n), \quad a = \cos \phi, \quad b = \sin \phi$$

- ▶ we have to remember complex trigonometric formulas
- ▶ we have to carry more terms in our equations

The advantages of complex exponentials

Example: change the phase of a pure cosine with complex exponentials

$$\cos(\omega n + \phi) = \operatorname{Re}\{e^{j(\omega n + \phi)}\} = \operatorname{Re}\{e^{j\omega n} e^{j\phi}\}$$

- ▶ sine and cosine “live” together
- ▶ phase shift is simple multiplication
- ▶ notation is simpler

The advantages of complex exponentials

Example: change the phase of a pure cosine with complex exponentials

$$\cos(\omega n + \phi) = \operatorname{Re}\{e^{j(\omega n + \phi)}\} = \operatorname{Re}\{e^{j\omega n} e^{j\phi}\}$$

- ▶ sine and cosine “live” together
- ▶ phase shift is simple multiplication
- ▶ notation is simpler

The advantages of complex exponentials

Example: change the phase of a pure cosine with complex exponentials

$$\cos(\omega n + \phi) = \operatorname{Re}\{e^{j(\omega n + \phi)}\} = \operatorname{Re}\{e^{j\omega n} e^{j\phi}\}$$

- ▶ sine and cosine “live” together
- ▶ phase shift is simple multiplication
- ▶ notation is simpler

The advantages of complex exponentials

Example: change the phase of a pure cosine with complex exponentials

$$\cos(\omega n + \phi) = \operatorname{Re}\{e^{j(\omega n + \phi)}\} = \operatorname{Re}\{e^{j\omega n} e^{j\phi}\}$$

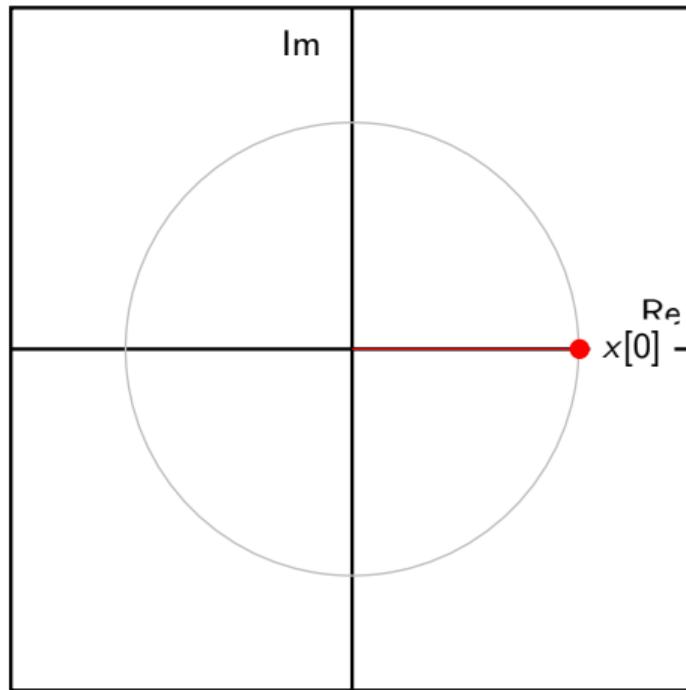
- ▶ sine and cosine “live” together
- ▶ phase shift is simple multiplication
- ▶ notation is simpler

The discrete-time oscillatory heartbeat

$$x[n] = A e^{j(\omega n + \phi)}$$

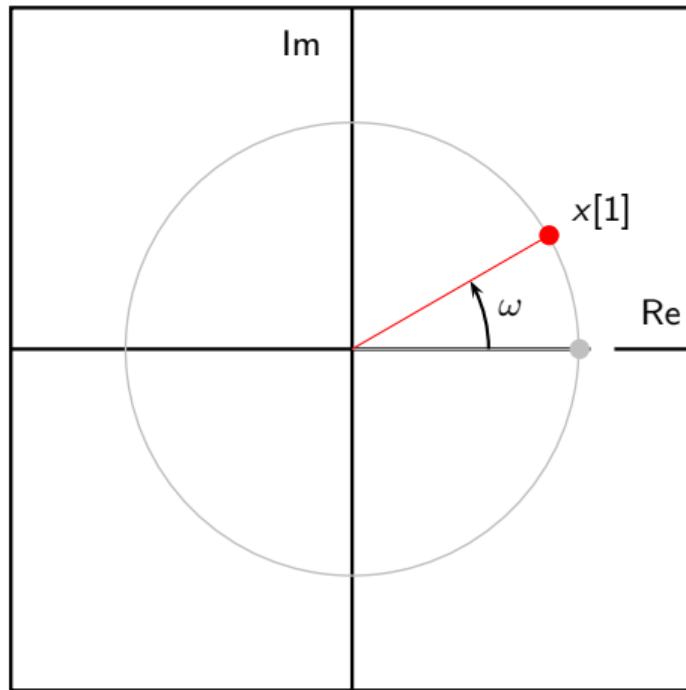
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



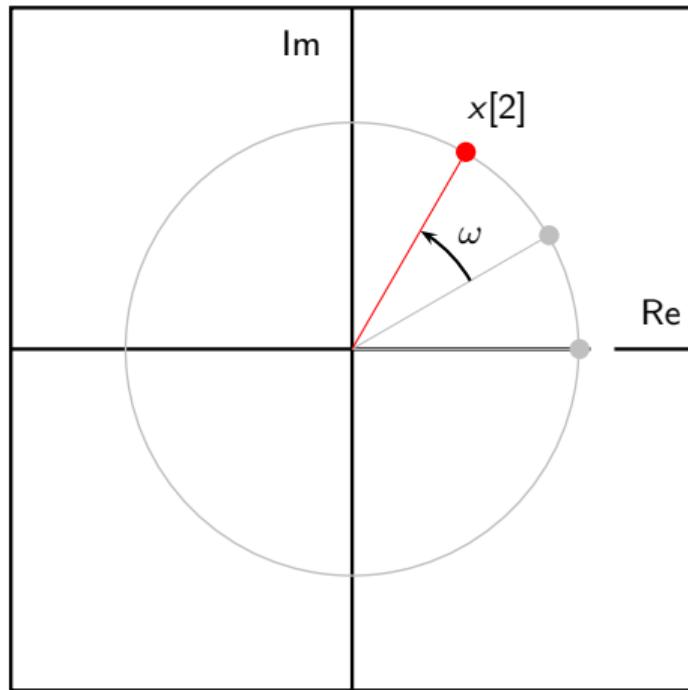
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



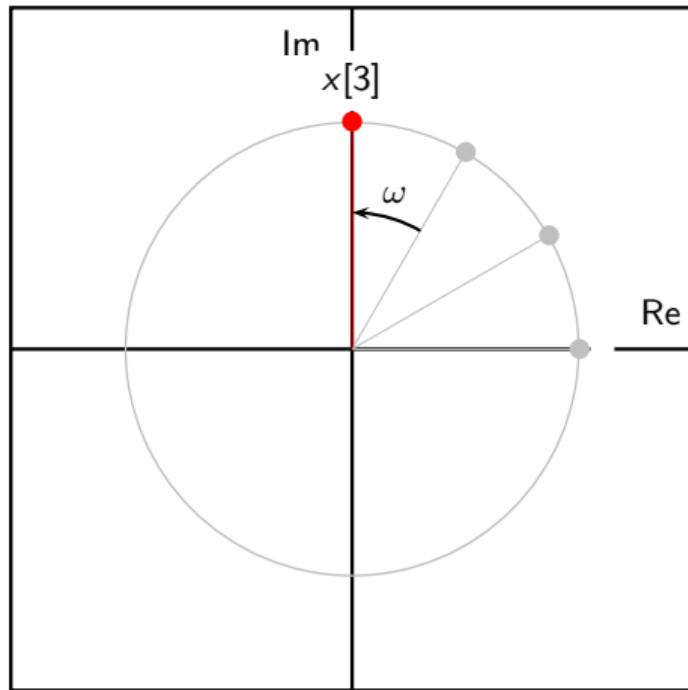
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



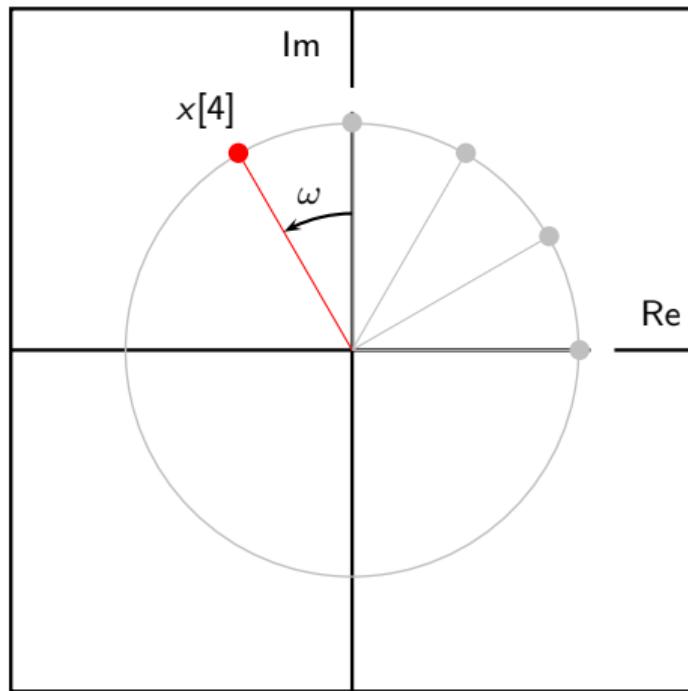
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



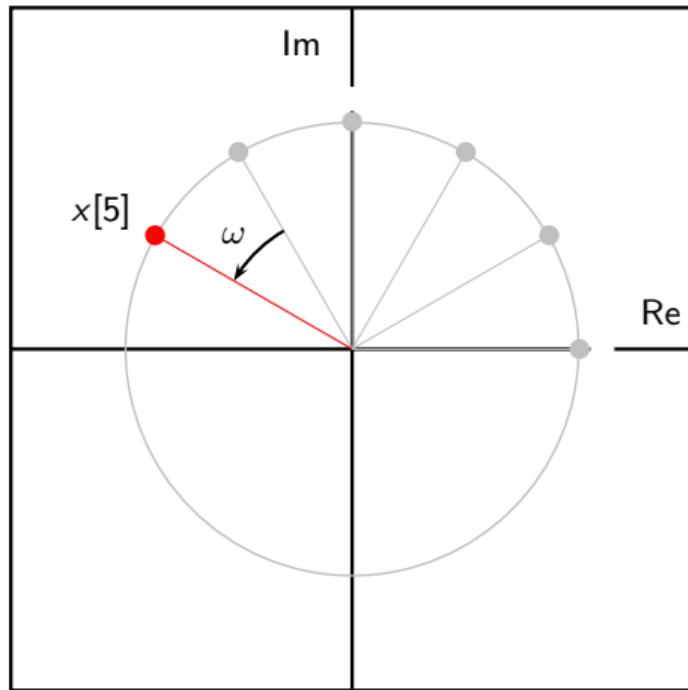
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



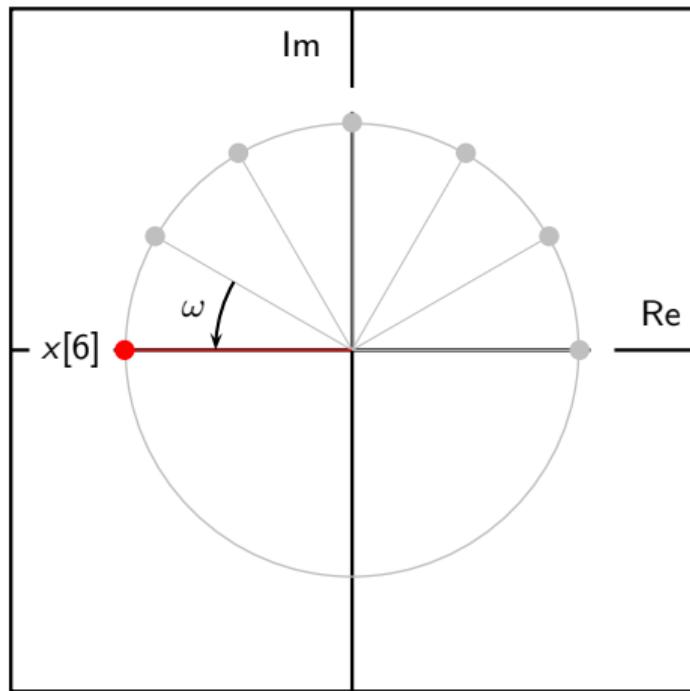
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



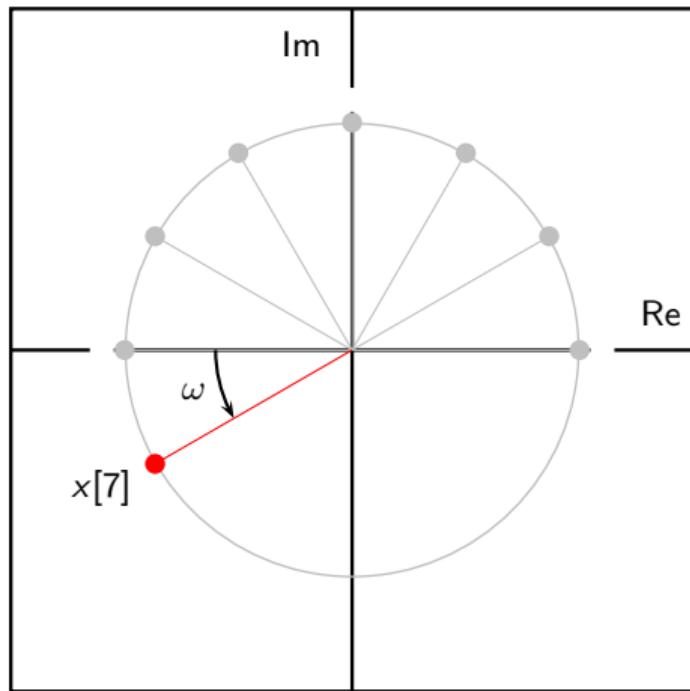
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



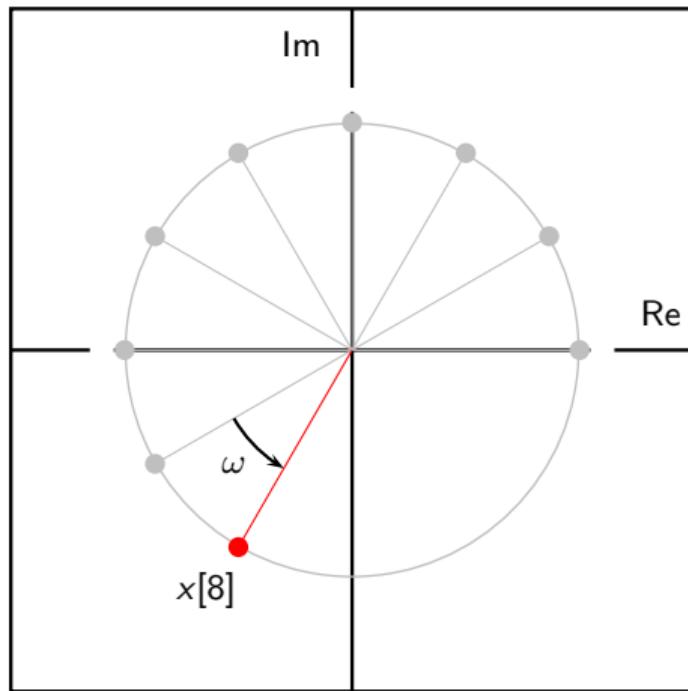
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



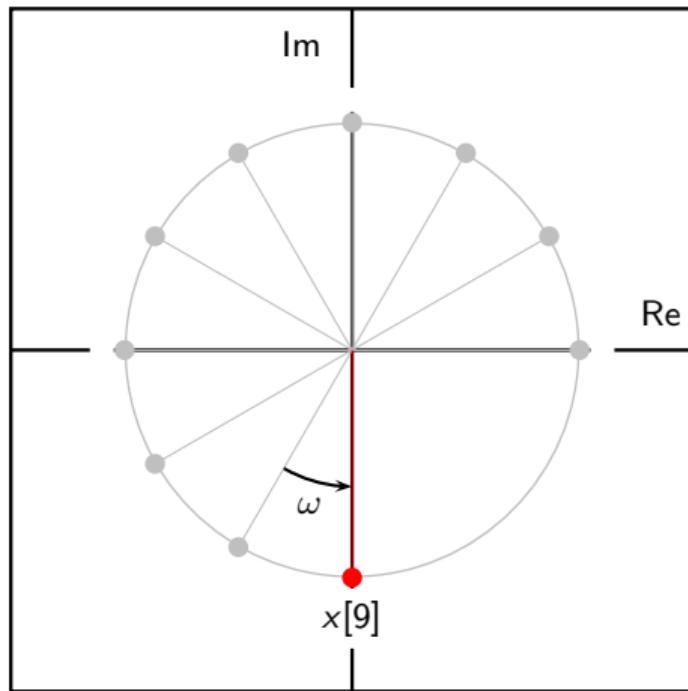
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



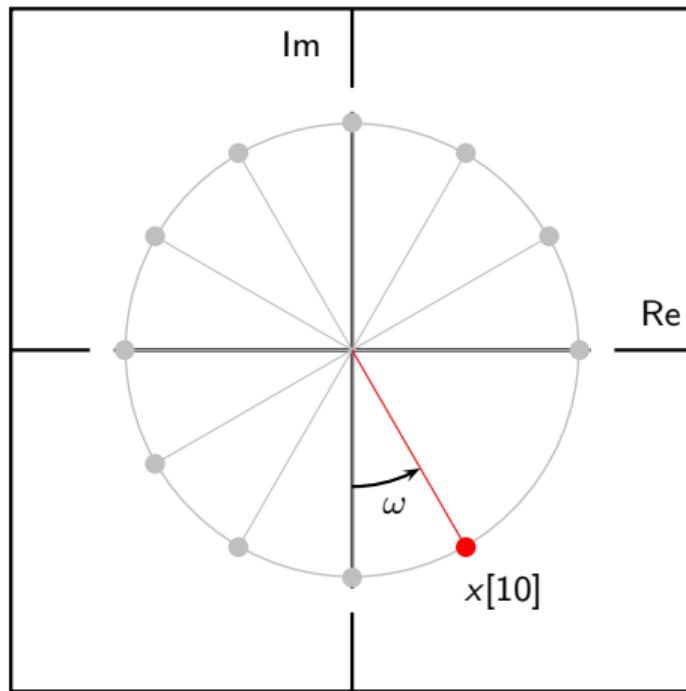
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



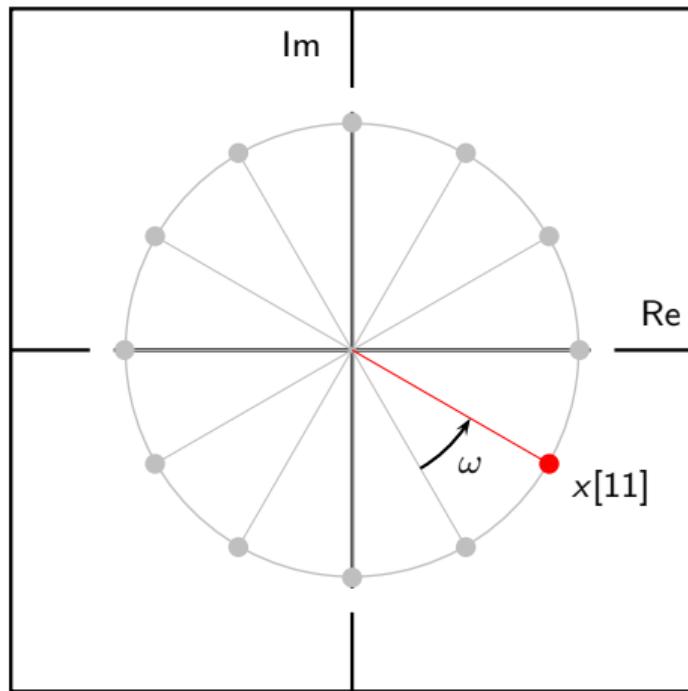
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



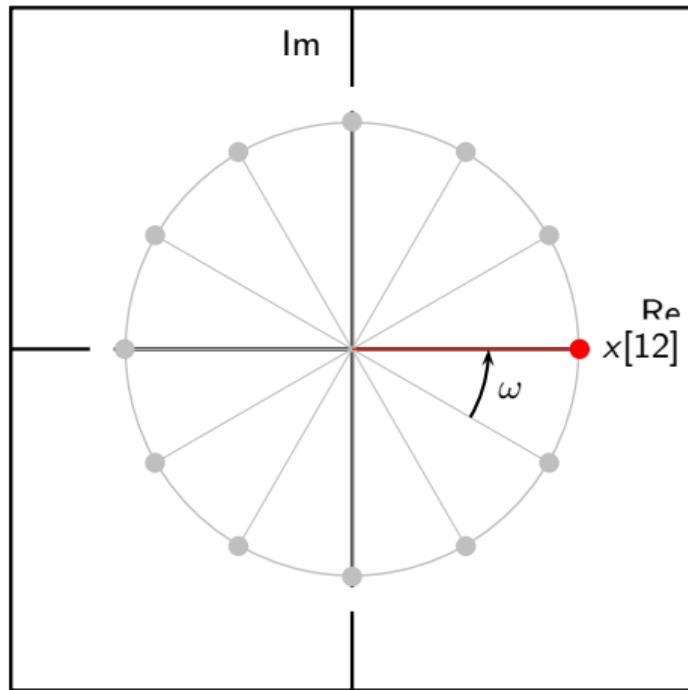
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



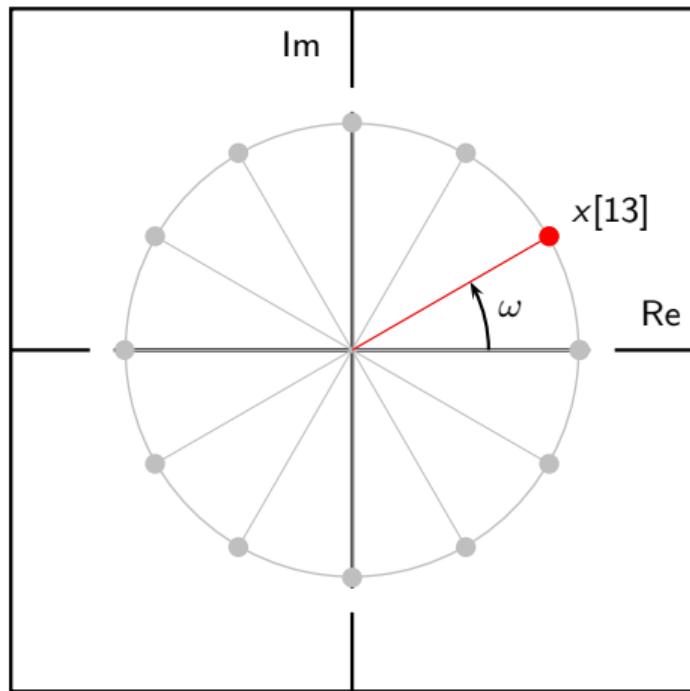
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



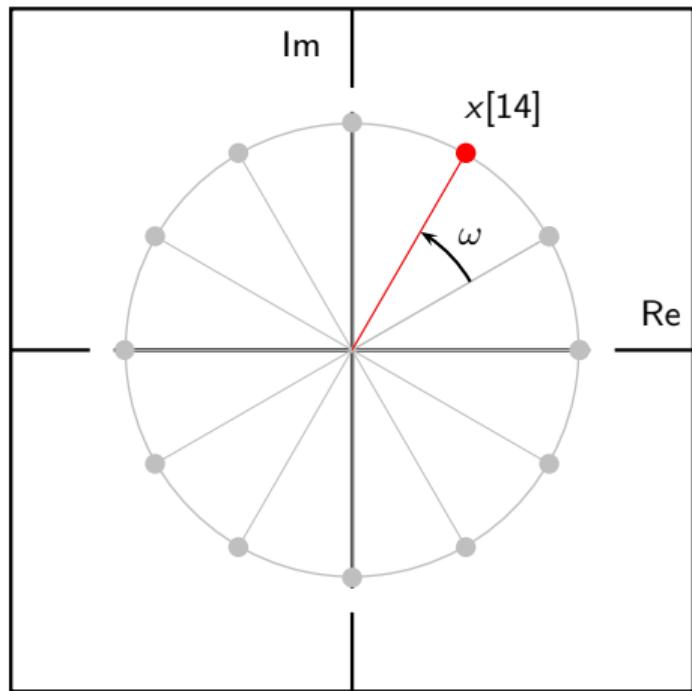
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



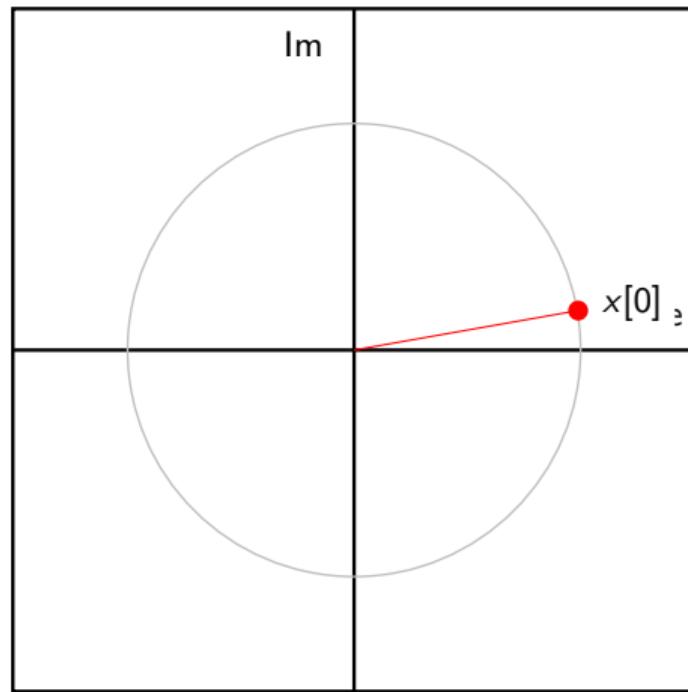
The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



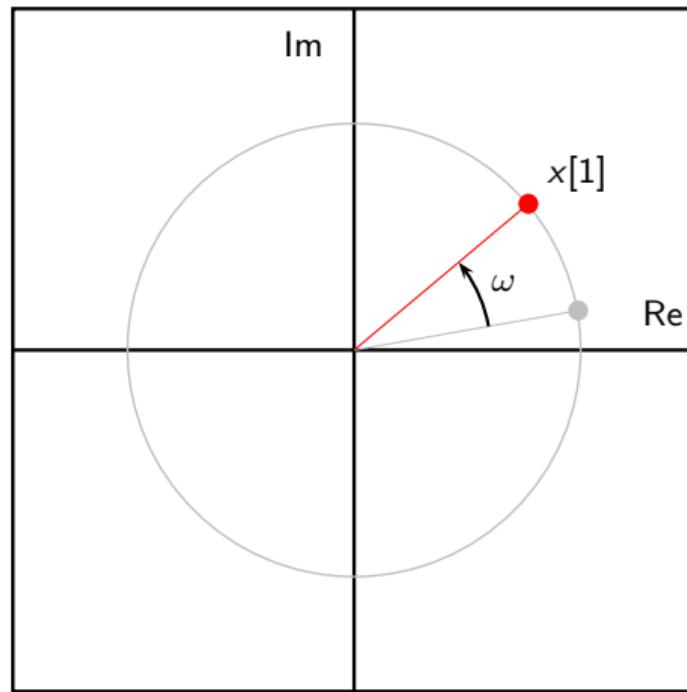
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



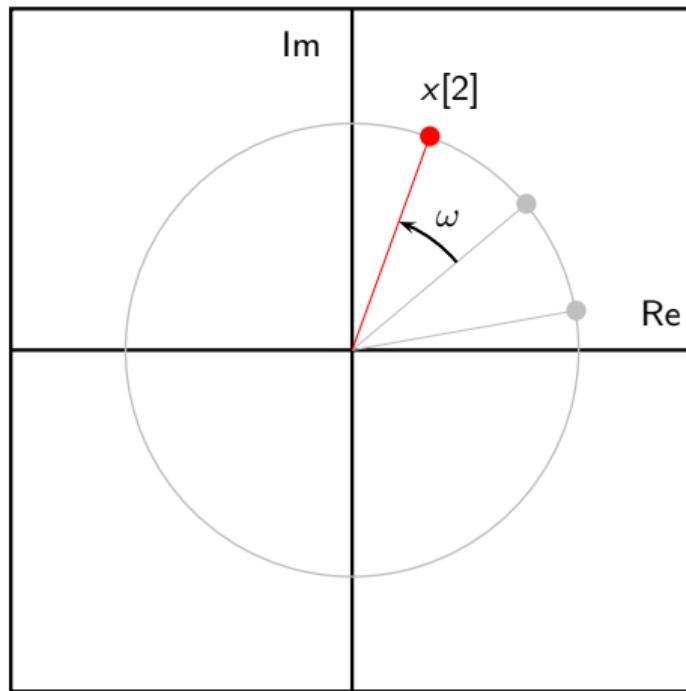
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



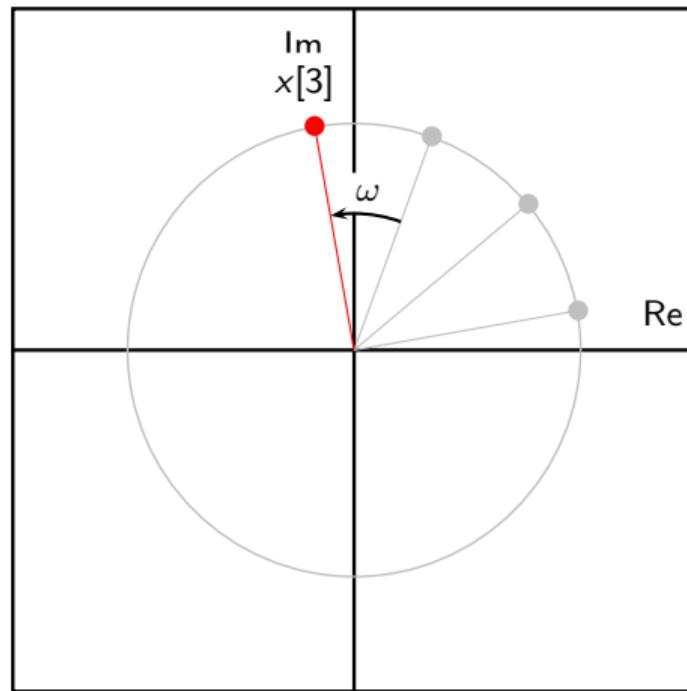
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



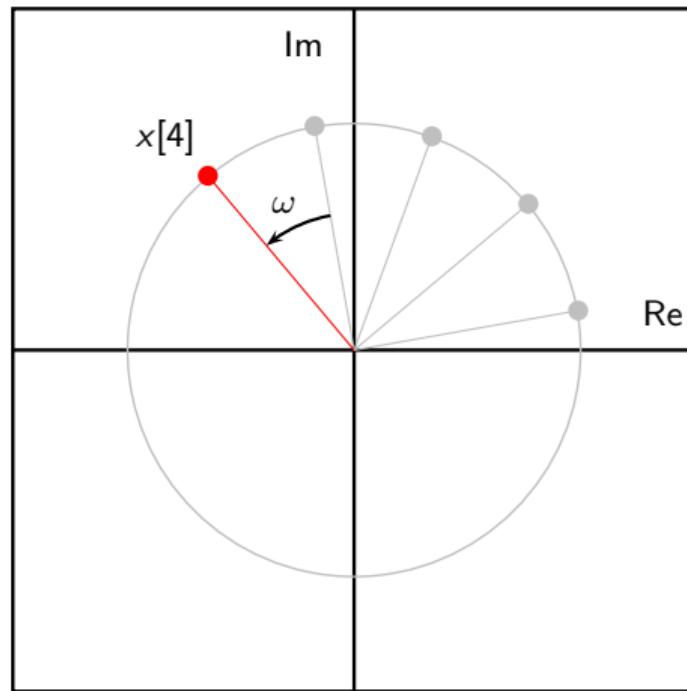
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



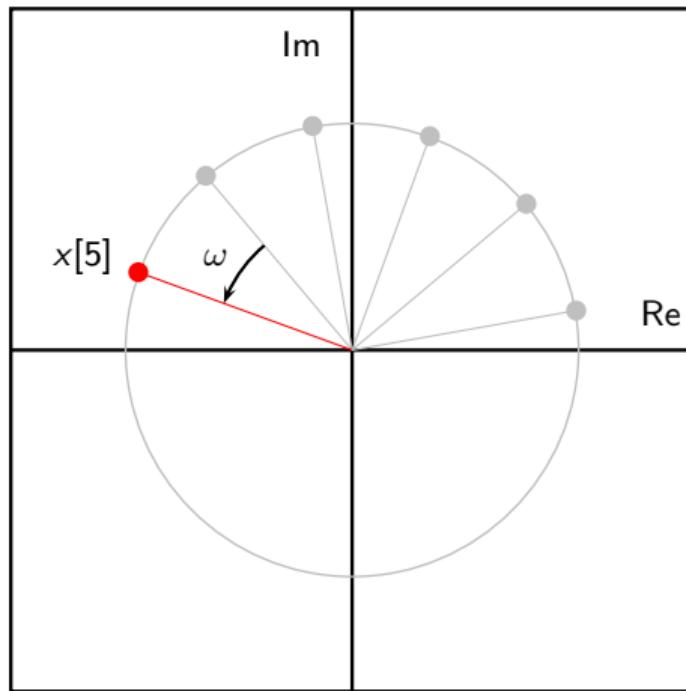
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



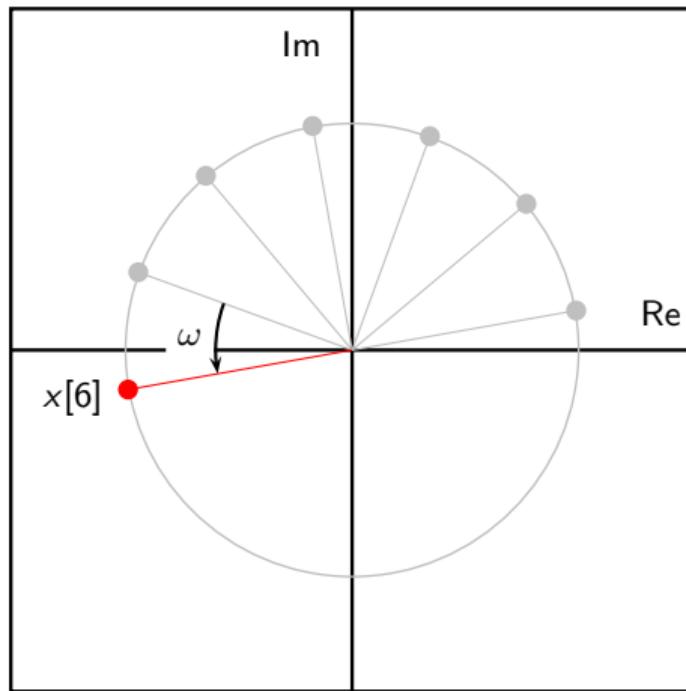
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



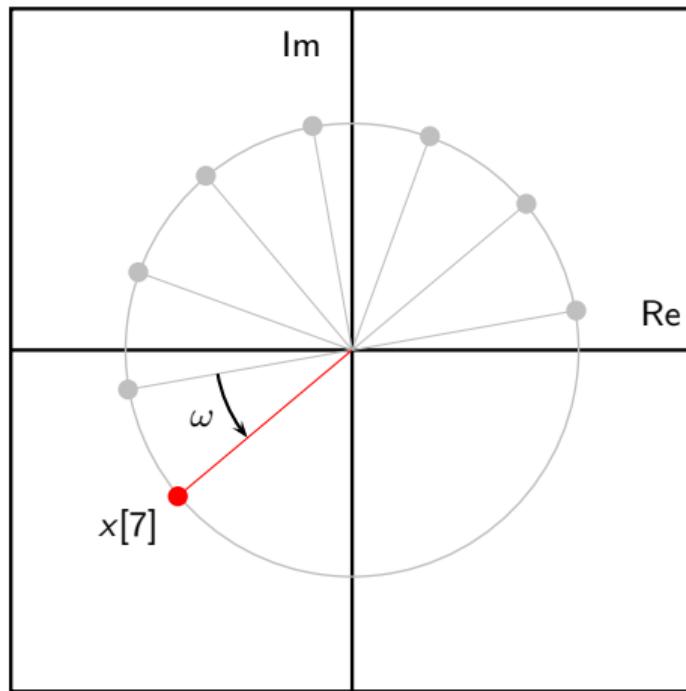
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



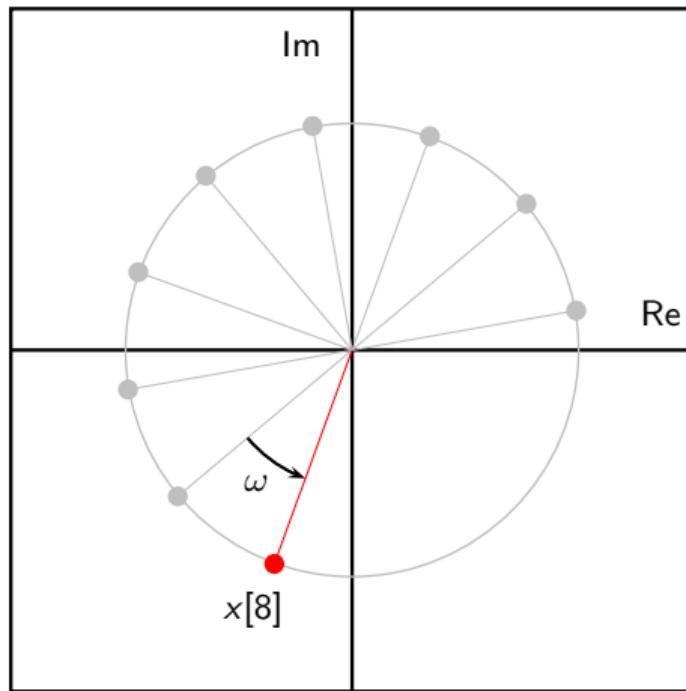
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



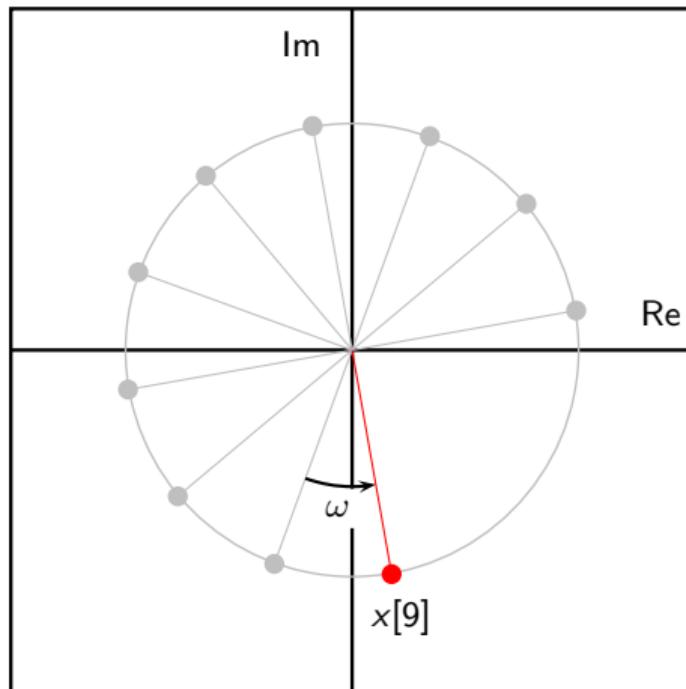
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



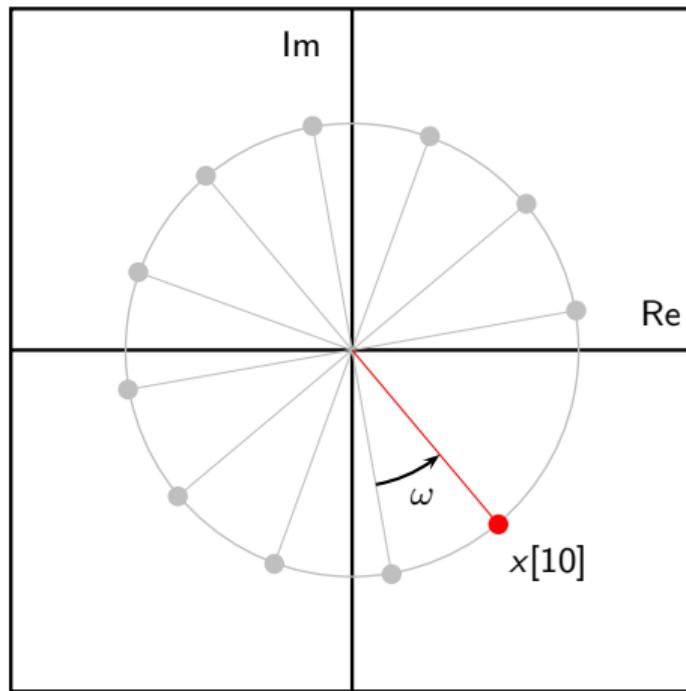
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



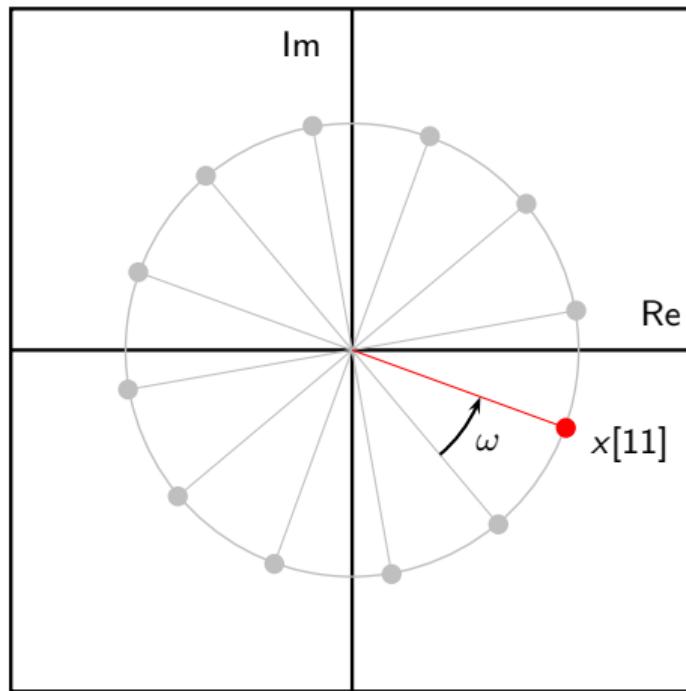
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



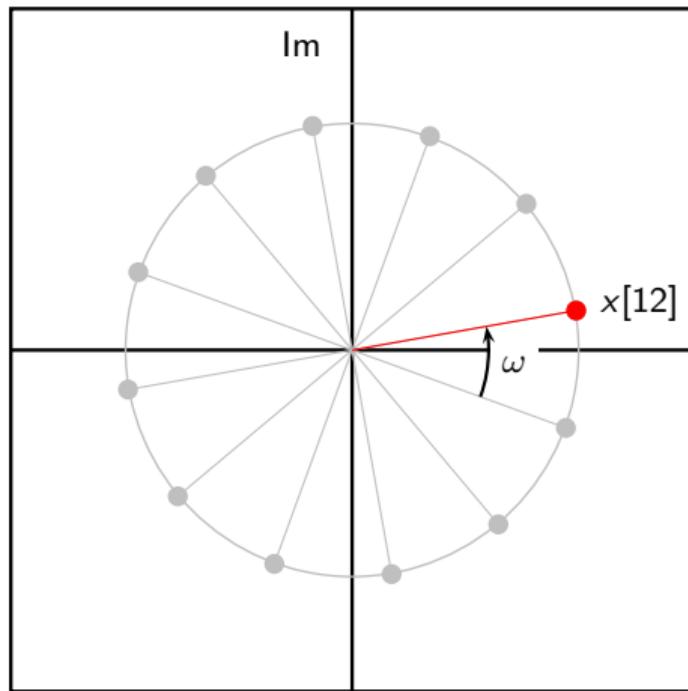
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



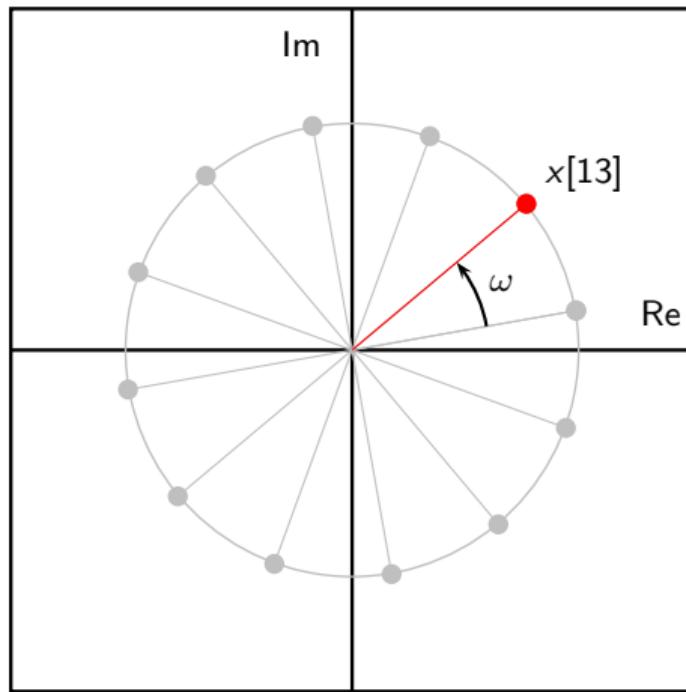
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



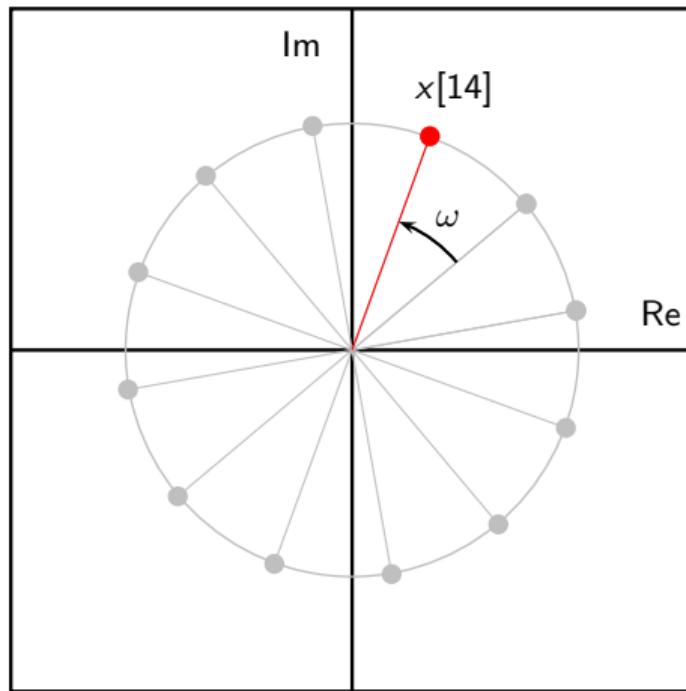
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



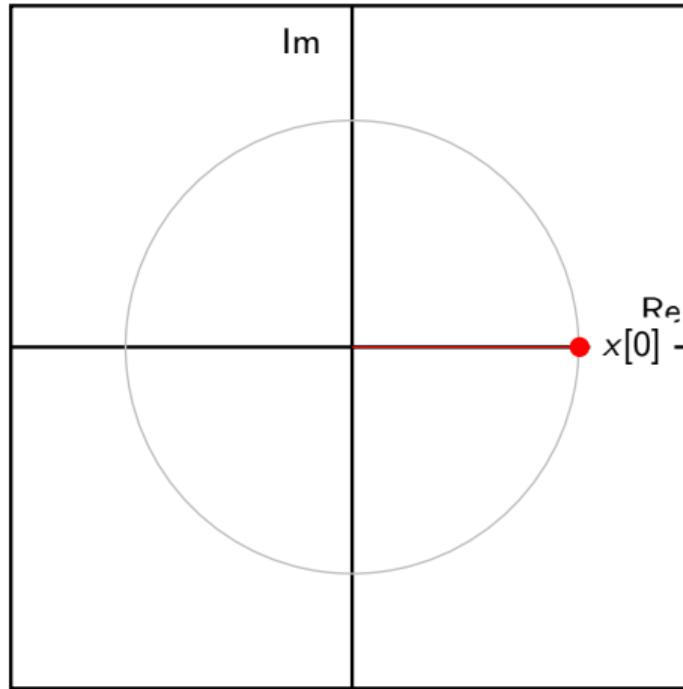
Initial phase

$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



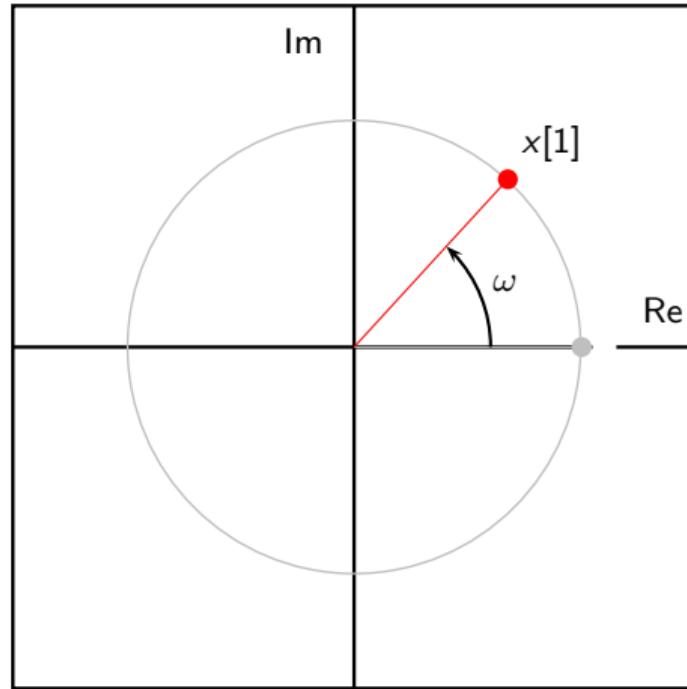
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



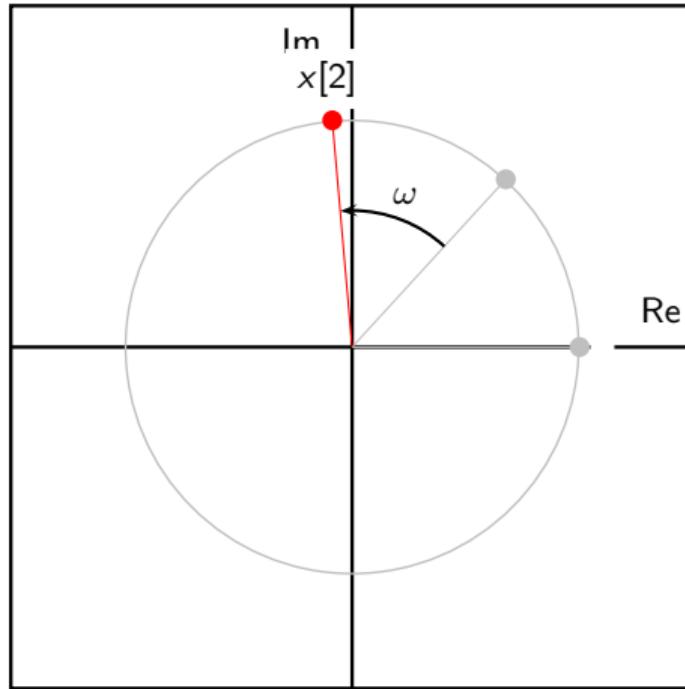
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



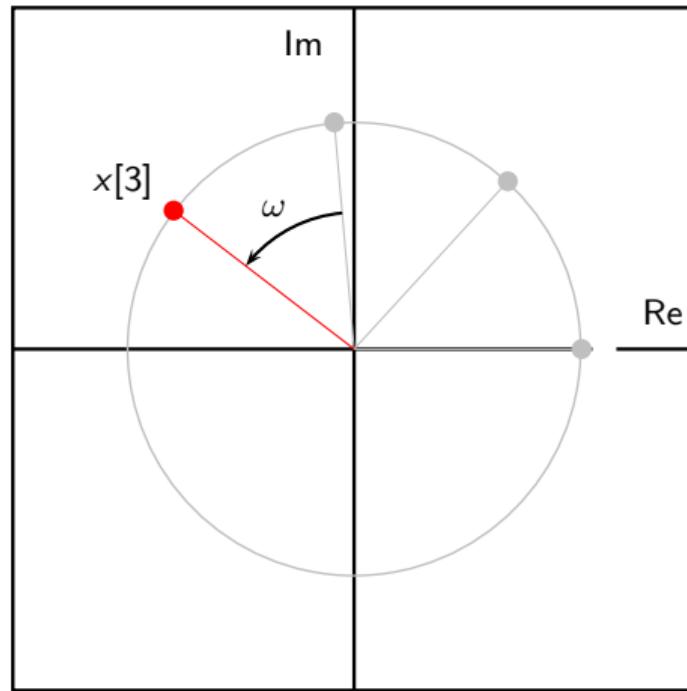
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



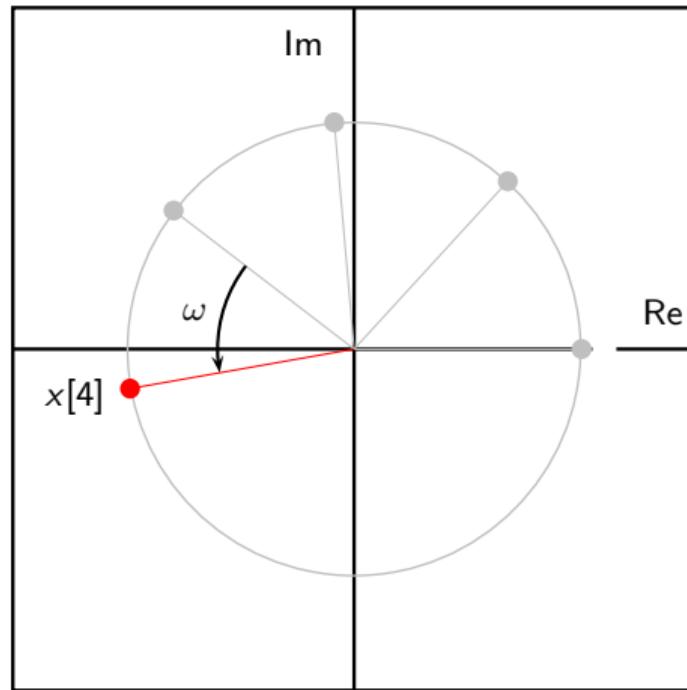
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



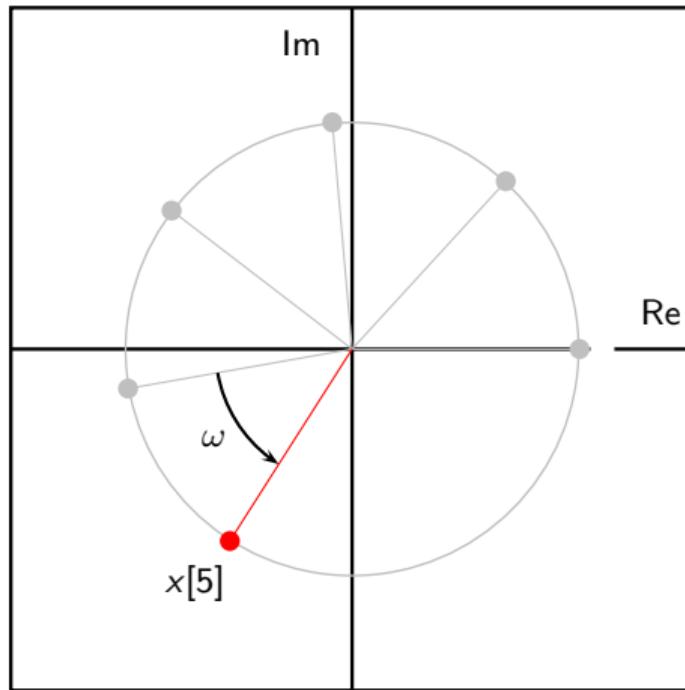
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



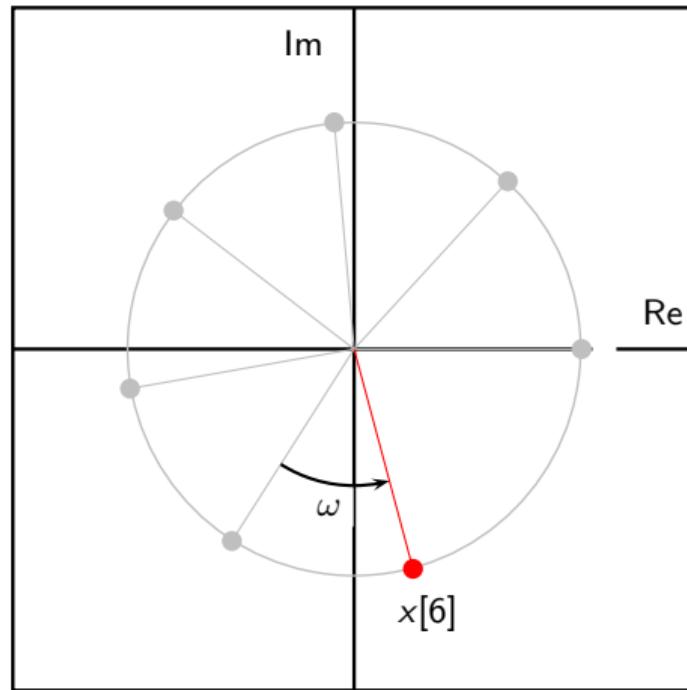
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



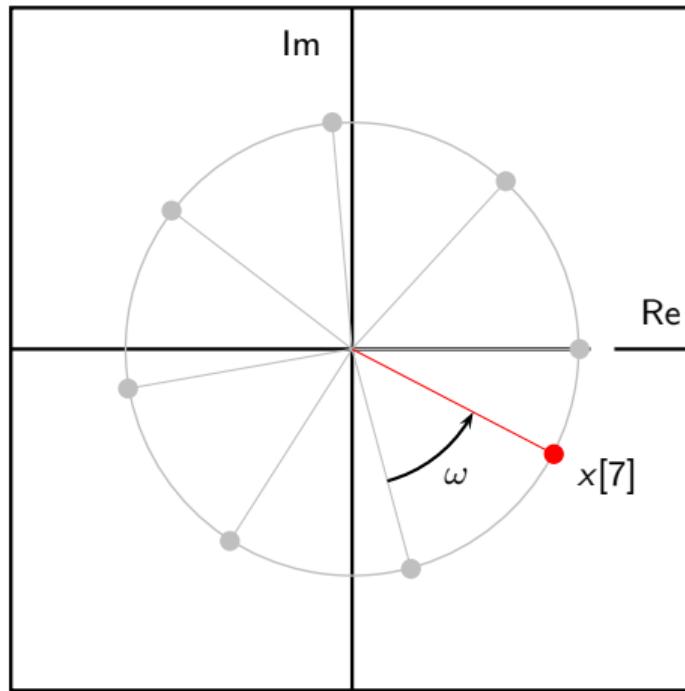
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



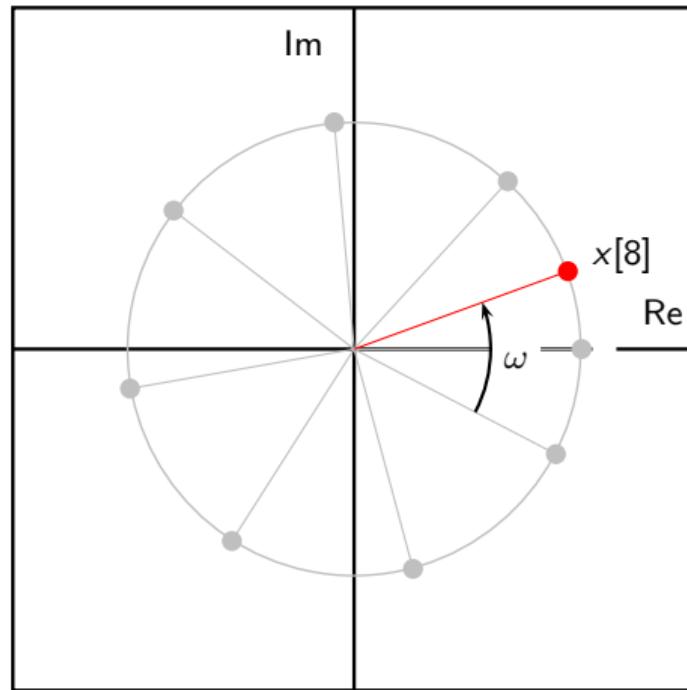
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



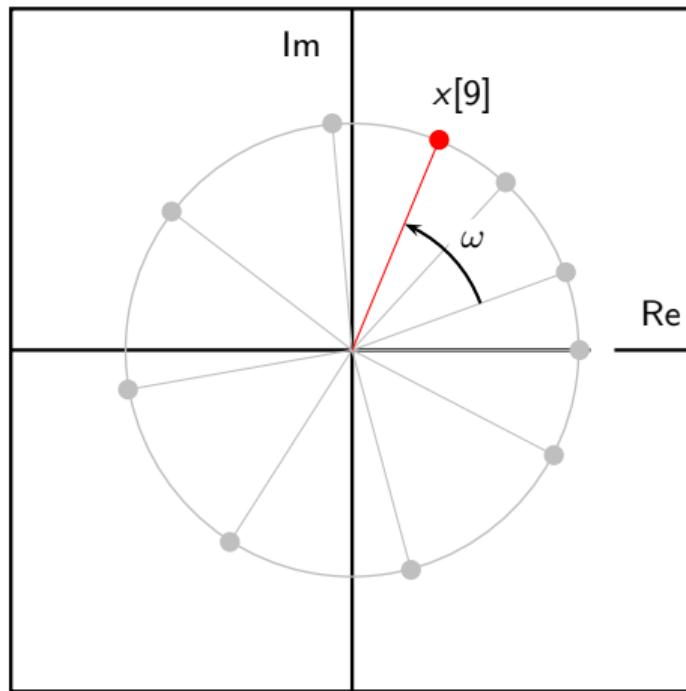
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



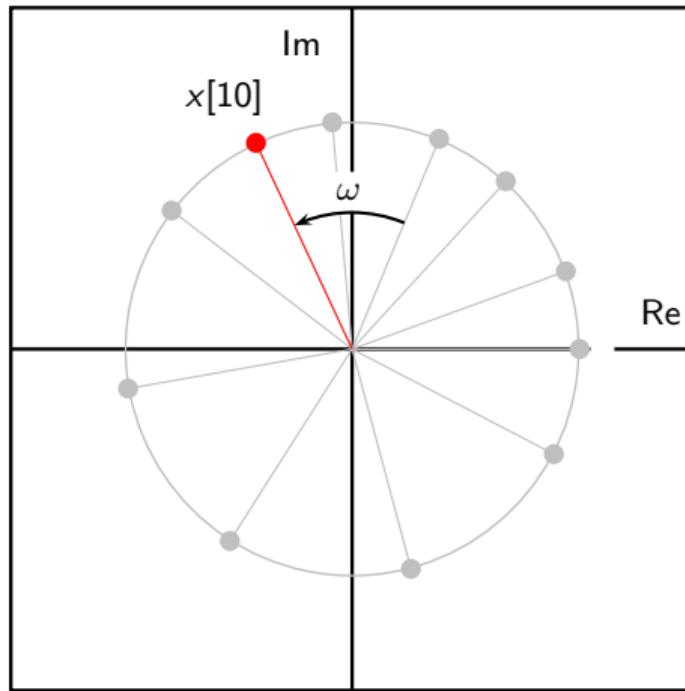
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



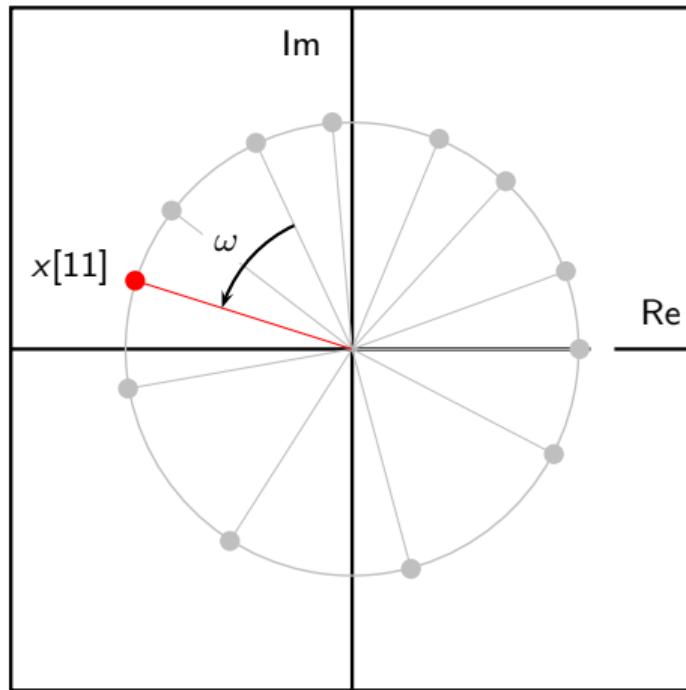
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



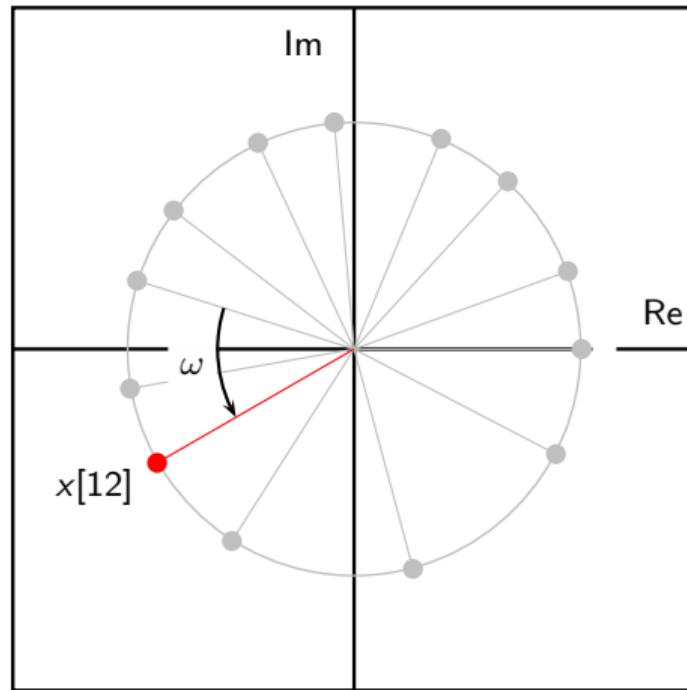
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



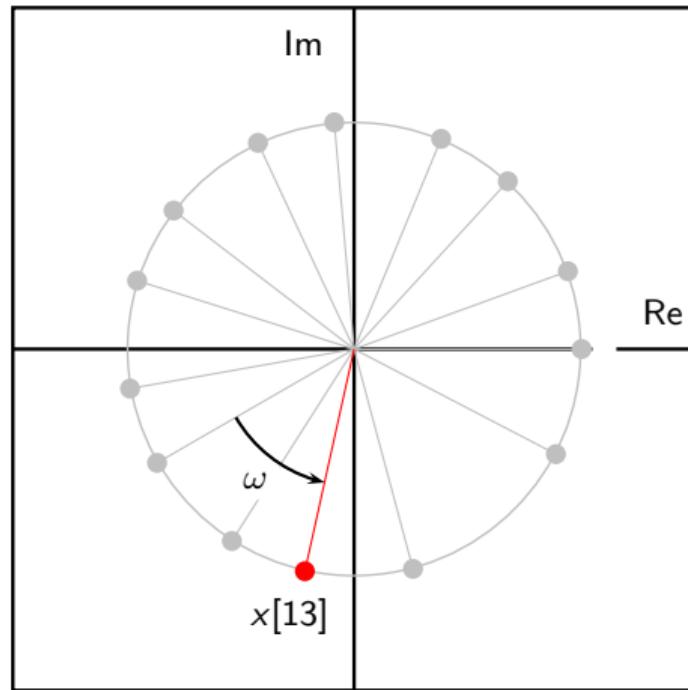
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



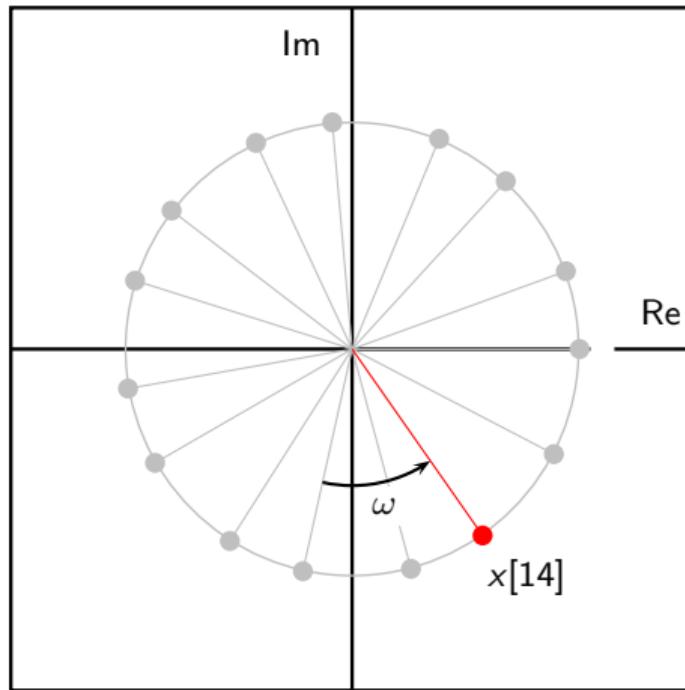
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



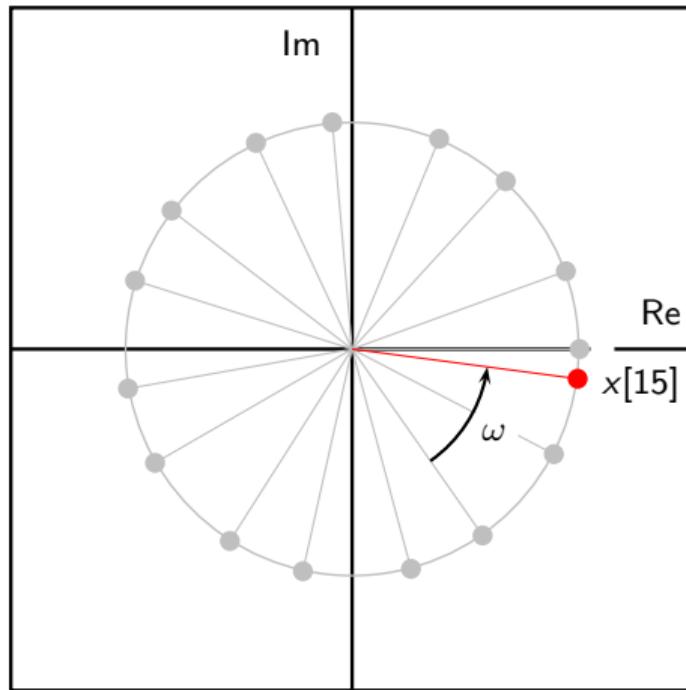
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



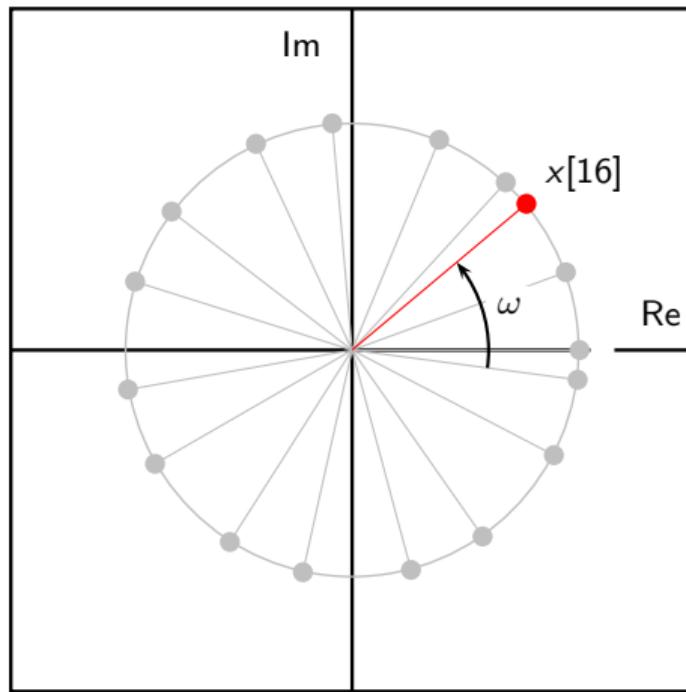
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



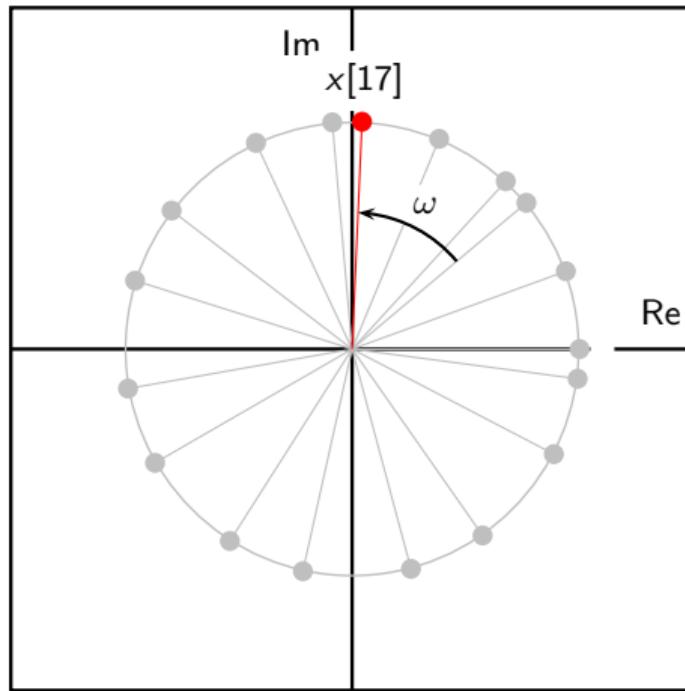
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



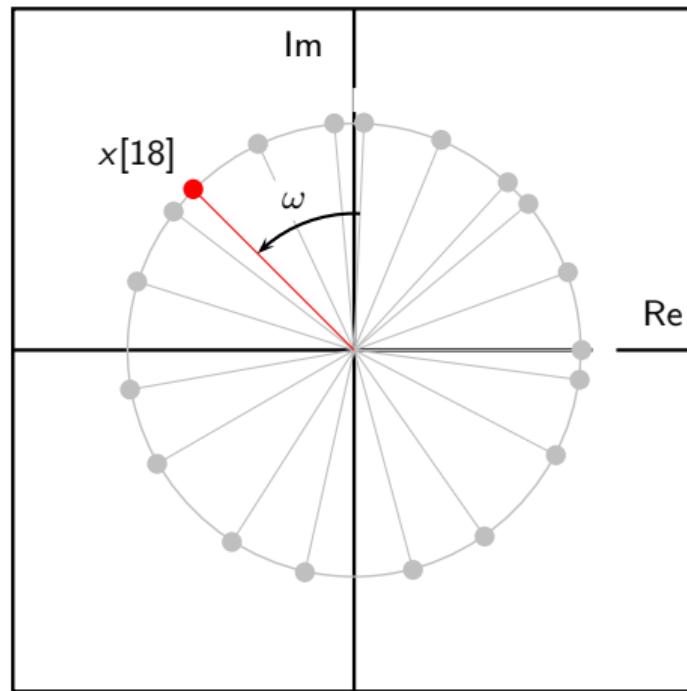
Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



Careful: not every discrete-time sinusoid is periodic!

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



Condition for periodicity in discrete time

$$e^{j\omega n} \text{ periodic in } n \iff \omega = \frac{M}{N}2\pi, \quad M, N \in \mathbb{Z}$$

Condition for periodicity in discrete time

$$x[n] = x[n + N]$$

$$e^{j(\omega n + \phi)} = e^{j(\omega(n+N) + \phi)}$$

$$e^{j\omega n} e^{j\phi} = e^{j\omega n} e^{j\omega N} e^{j\phi}$$

$$e^{j\omega N} = 1$$

$$\omega N = 2M\pi, \quad M \in \mathbb{Z}$$

$$\omega = \frac{M}{N} 2\pi$$

Condition for periodicity in discrete time

$$x[n] = x[n + N]$$

$$e^{j(\omega n + \phi)} = e^{j(\omega(n+N) + \phi)}$$

$$e^{j\omega n} e^{j\phi} = e^{j\omega n} e^{j\omega N} e^{j\phi}$$

$$e^{j\omega N} = 1$$

$$\omega N = 2M\pi, \quad M \in \mathbb{Z}$$

$$\omega = \frac{M}{N} 2\pi$$

Condition for periodicity in discrete time

$$x[n] = x[n + N]$$

$$e^{j(\omega n + \phi)} = e^{j(\omega(n+N) + \phi)}$$

$$e^{j\omega n} e^{j\phi} = e^{j\omega n} e^{j\omega N} e^{j\phi}$$

$$e^{j\omega N} = 1$$

$$\omega N = 2M\pi, \quad M \in \mathbb{Z}$$

$$\omega = \frac{M}{N} 2\pi$$

Condition for periodicity in discrete time

$$x[n] = x[n + N]$$

$$e^{j(\omega n + \phi)} = e^{j(\omega(n+N) + \phi)}$$

$$e^{j\omega n} e^{j\phi} = e^{j\omega n} e^{j\omega N} e^{j\phi}$$

$$e^{j\omega N} = 1$$

$$\omega N = 2M\pi, \quad M \in \mathbb{Z}$$

$$\omega = \frac{M}{N} 2\pi$$

Condition for periodicity in discrete time

$$x[n] = x[n + N]$$

$$e^{j(\omega n + \phi)} = e^{j(\omega(n+N) + \phi)}$$

$$e^{j\omega n} e^{j\phi} = e^{j\omega n} e^{j\omega N} e^{j\phi}$$

$$e^{j\omega N} = 1$$

$$\omega N = 2M\pi, \quad M \in \mathbb{Z}$$

$$\omega = \frac{M}{N} 2\pi$$

Condition for periodicity in discrete time

$$x[n] = x[n + N]$$

$$e^{j(\omega n + \phi)} = e^{j(\omega(n+N) + \phi)}$$

$$e^{j\omega n} e^{j\phi} = e^{j\omega n} e^{j\omega N} e^{j\phi}$$

$$e^{j\omega N} = 1$$

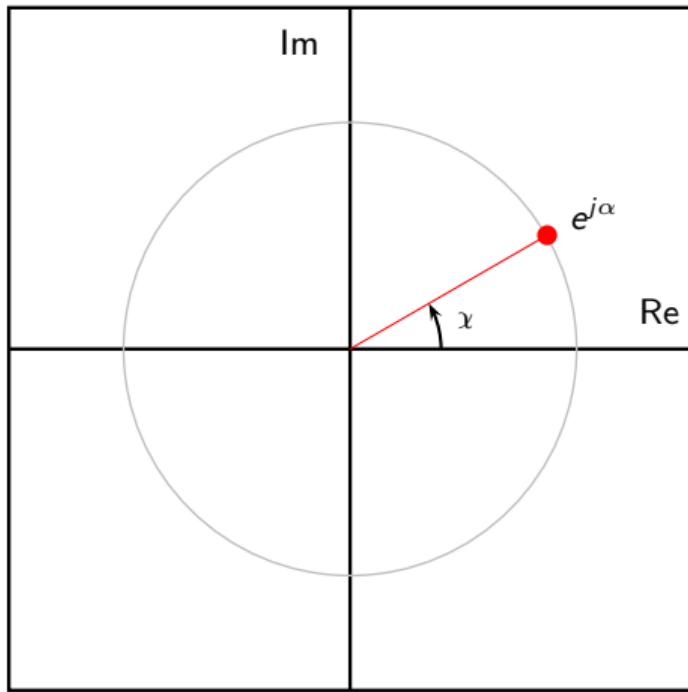
$$\omega N = 2M\pi, \quad M \in \mathbb{Z}$$

$$\omega = \frac{M}{N} 2\pi$$

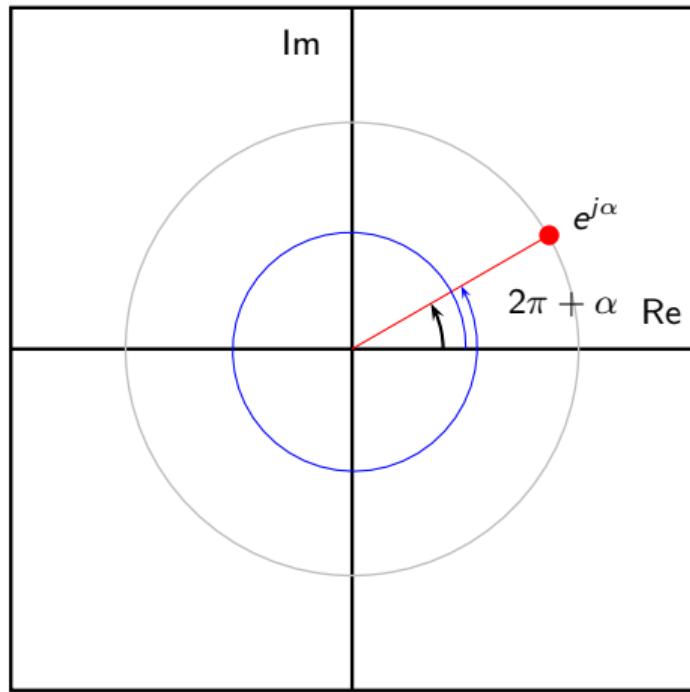
2π phase periodicity of complex exponentials

$$e^{j\alpha} = e^{j(\alpha+2k\pi)} \quad \forall k \in \mathbb{Z}$$

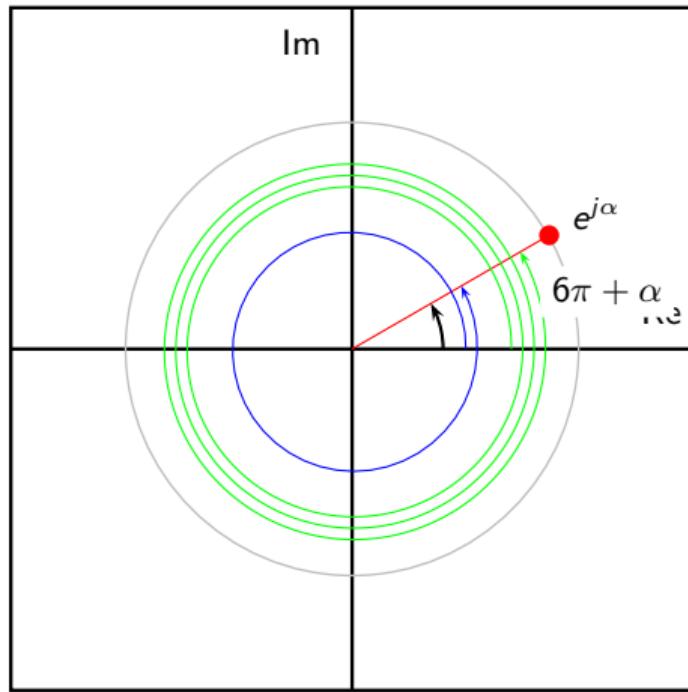
2π -periodicity: one point, many names



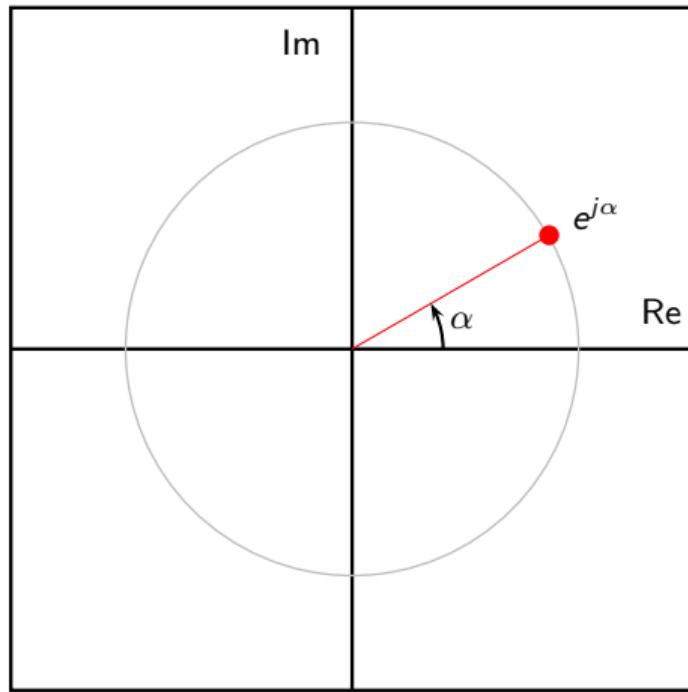
2π -periodicity: one point, many names



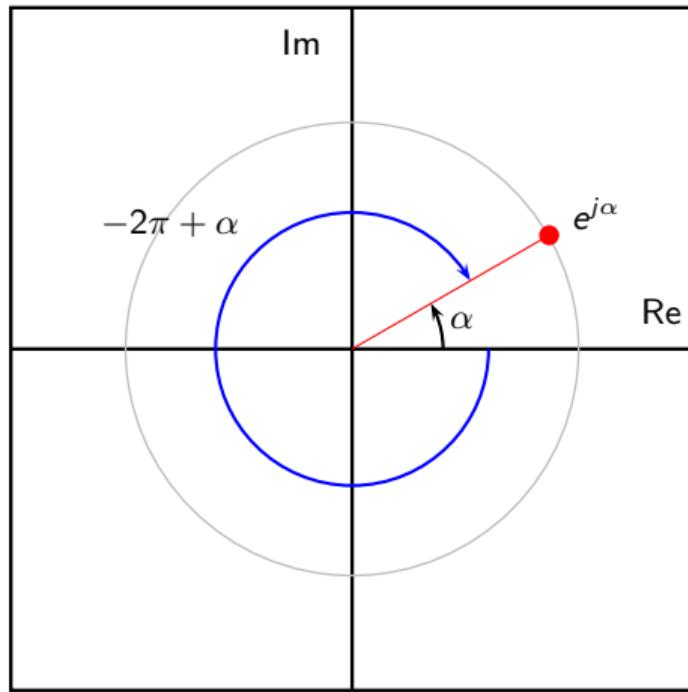
2π -periodicity: one point, many names



One point, many names



One point, many names

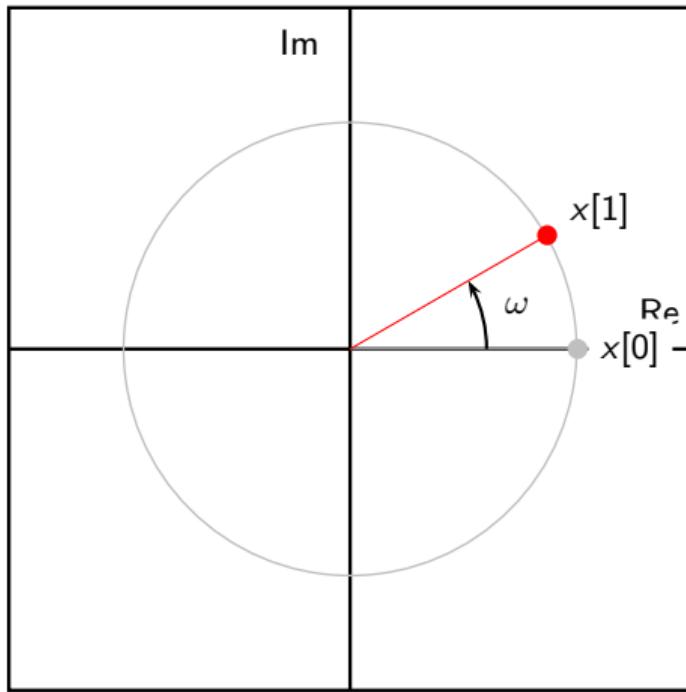


How “fast” can we go?

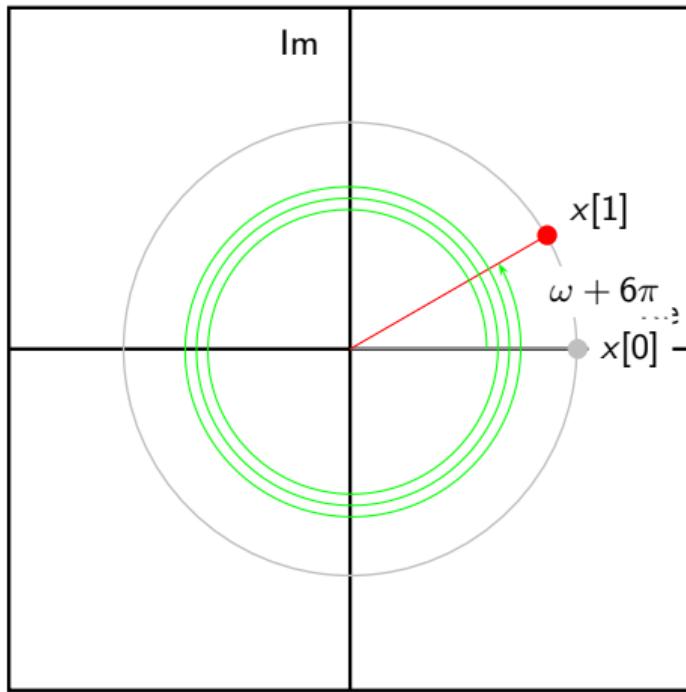
How “fast” can we go?

$$0 \leq \omega < 2\pi$$

Remember the complex exponential generating machine

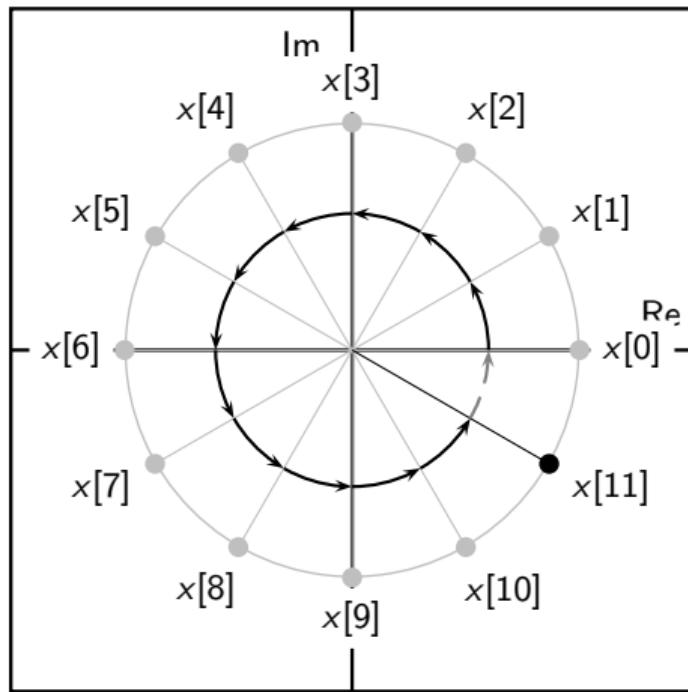


Remember the complex exponential generating machine



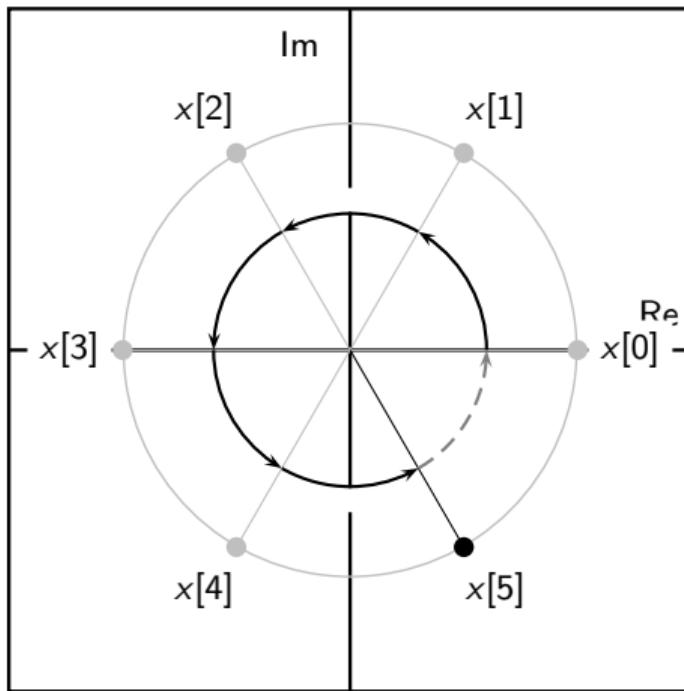
How “fast” can we go?

$$\omega = 2\pi/12$$



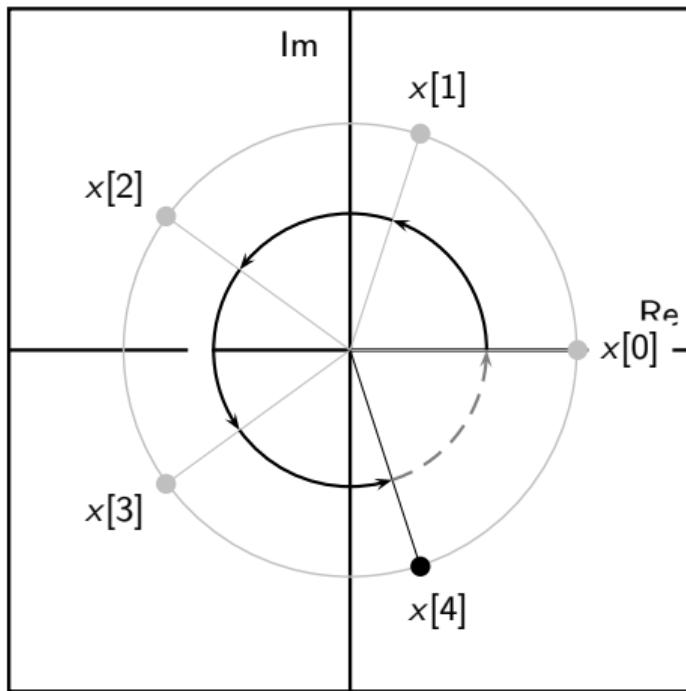
How “fast” can we go?

$$\omega = 2\pi/6$$



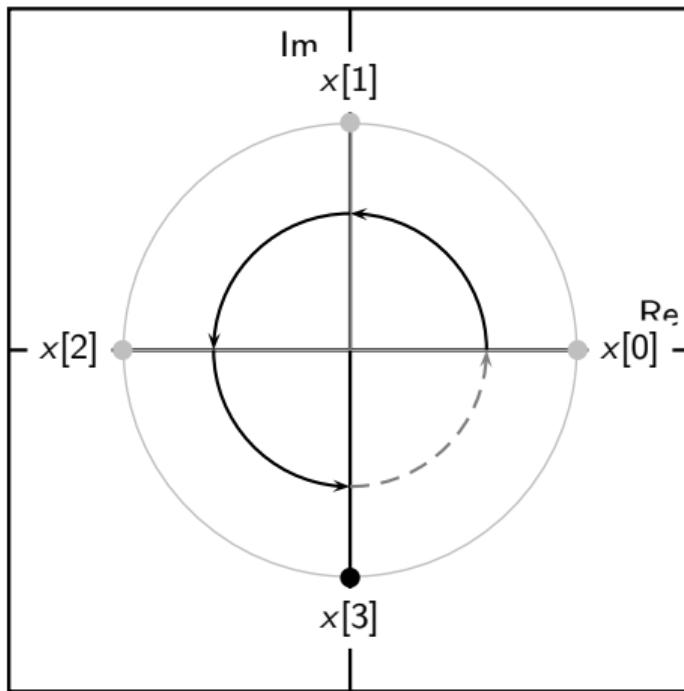
How “fast” can we go?

$$\omega = 2\pi/5$$



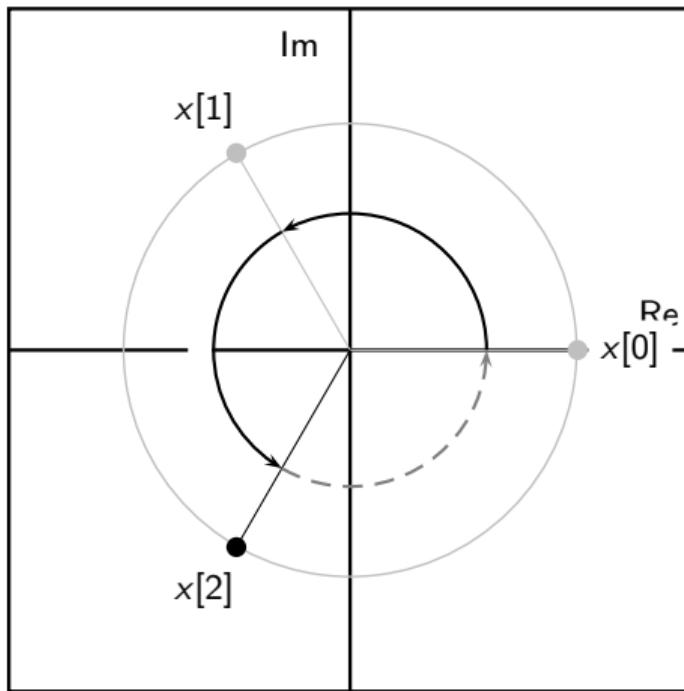
How “fast” can we go?

$$\omega = 2\pi/4$$



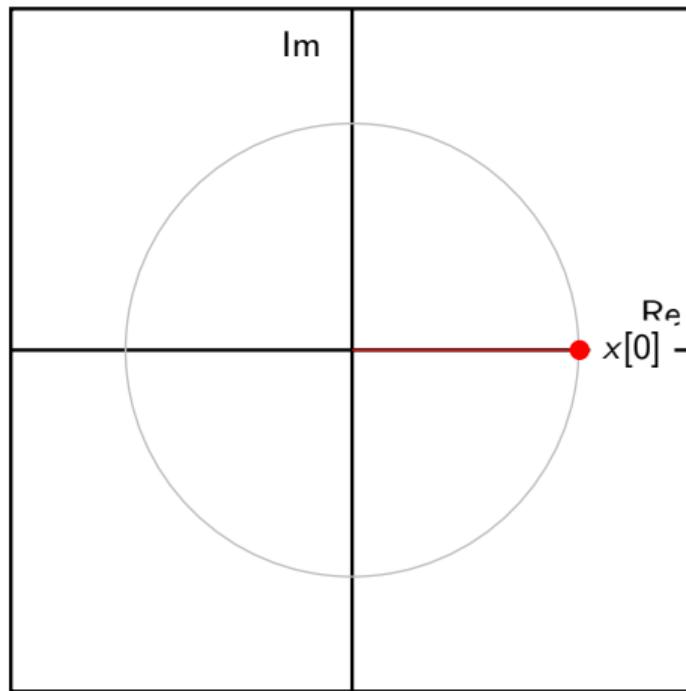
How “fast” can we go?

$$\omega = 2\pi/3$$



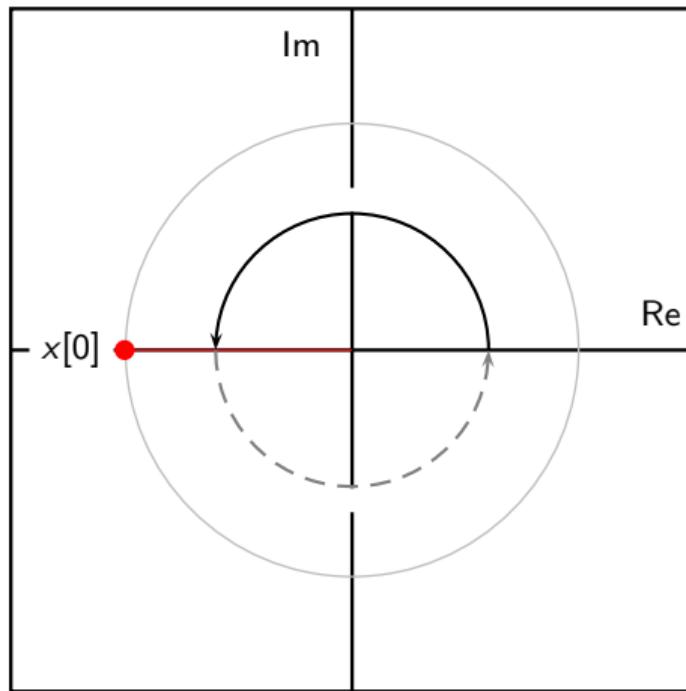
How “fast” can we go?

$$\omega = 2\pi/2 = \pi$$



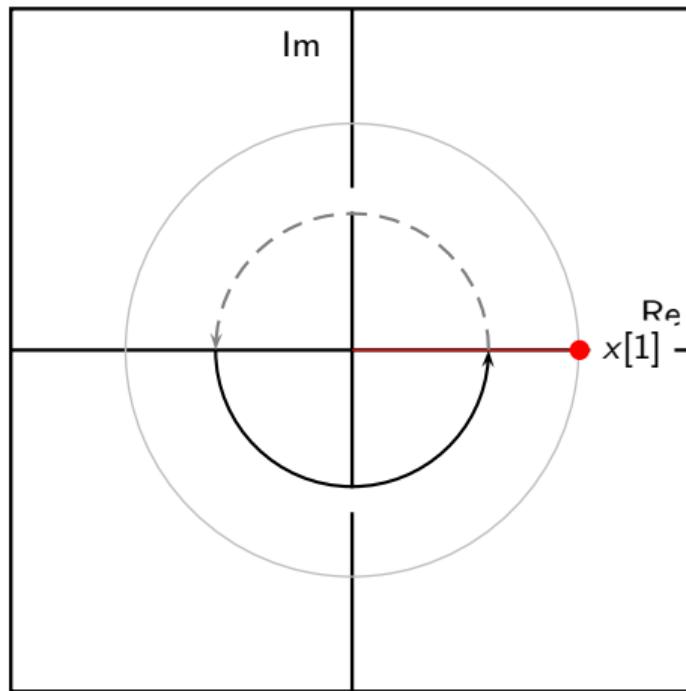
How “fast” can we go?

$$\omega = 2\pi/2 = \pi$$



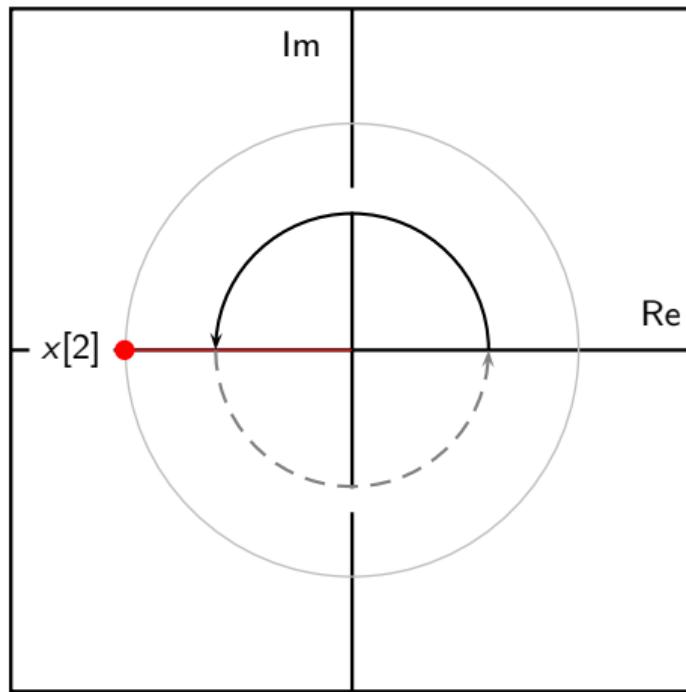
How “fast” can we go?

$$\omega = 2\pi/2 = \pi$$



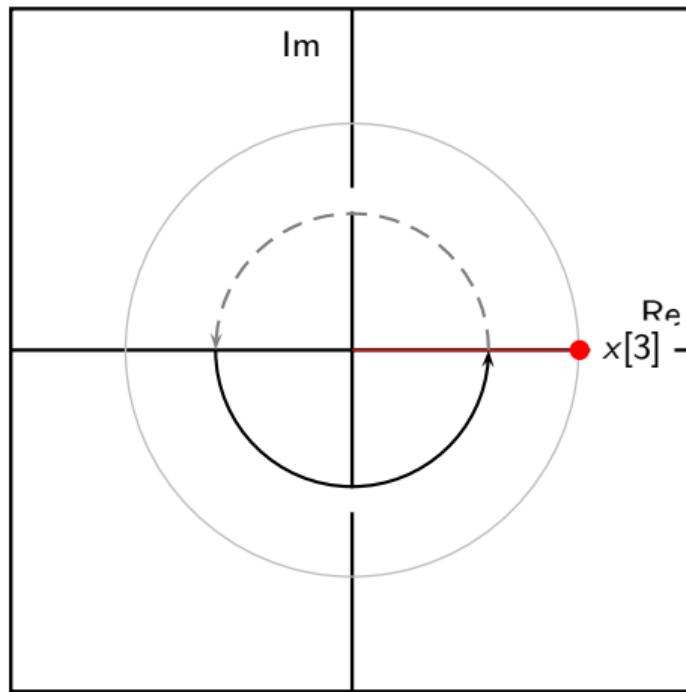
How “fast” can we go?

$$\omega = 2\pi/2 = \pi$$



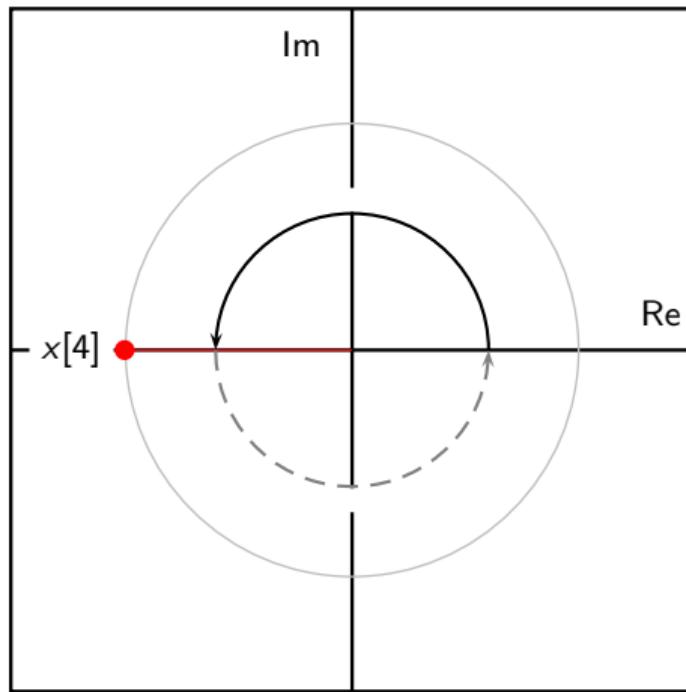
How “fast” can we go?

$$\omega = 2\pi/2 = \pi$$



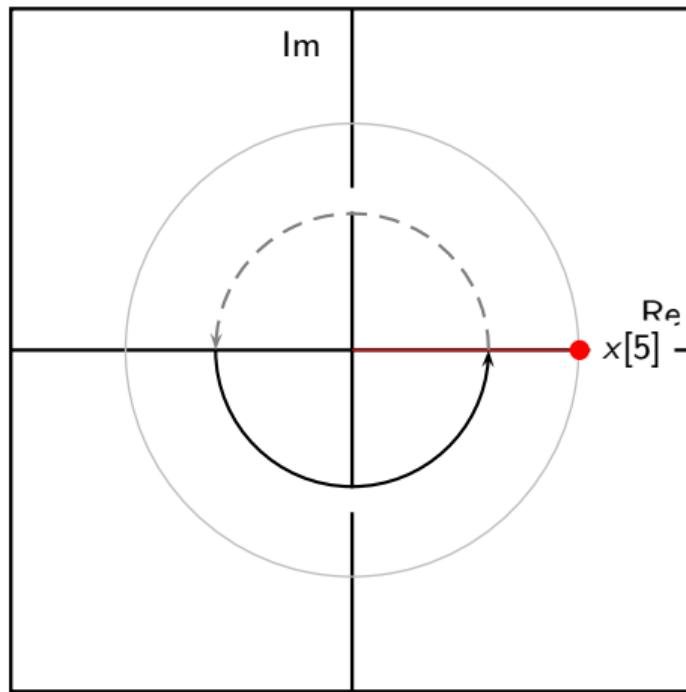
How “fast” can we go?

$$\omega = 2\pi/2 = \pi$$



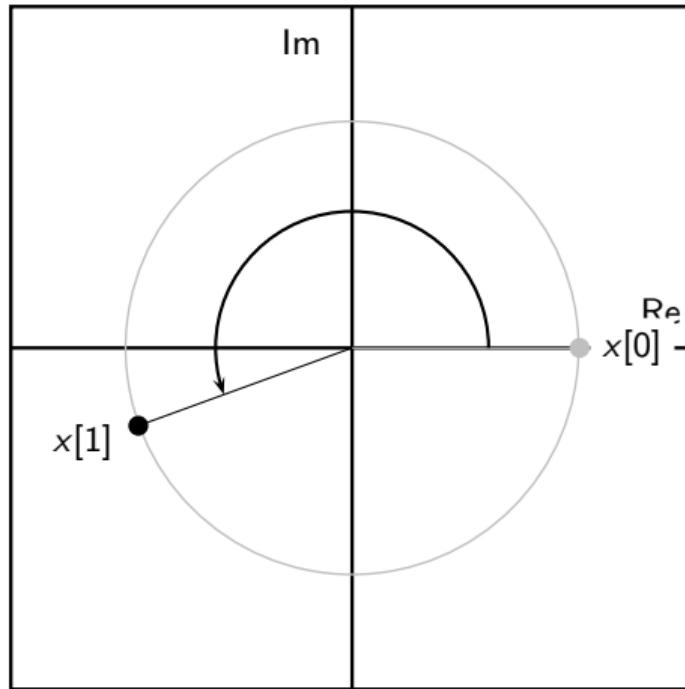
How “fast” can we go?

$$\omega = 2\pi/2 = \pi$$



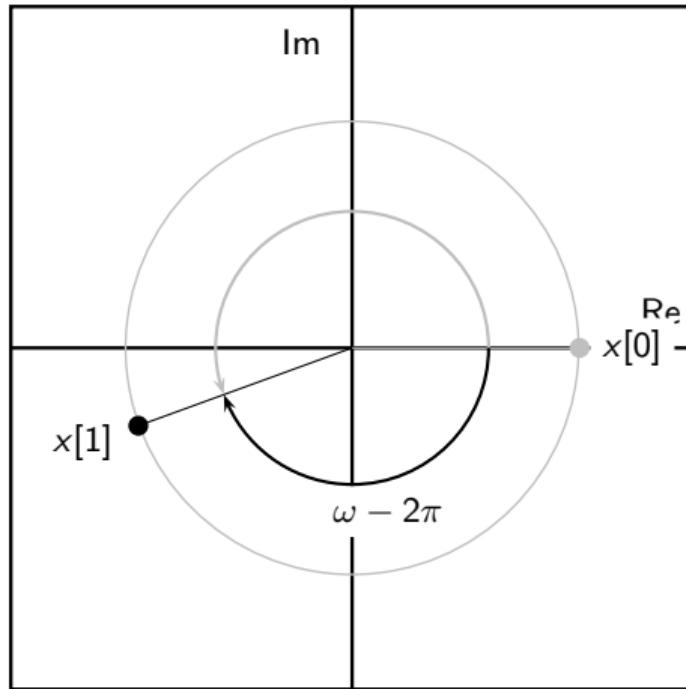
What if we go “faster”?

$$\pi < \omega < 2\pi$$



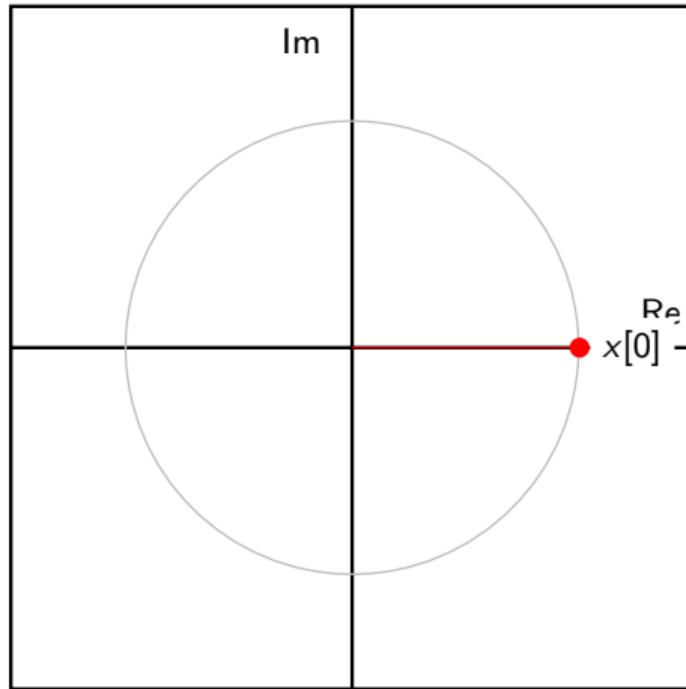
What if we go “faster”?

$$\pi < \omega < 2\pi$$



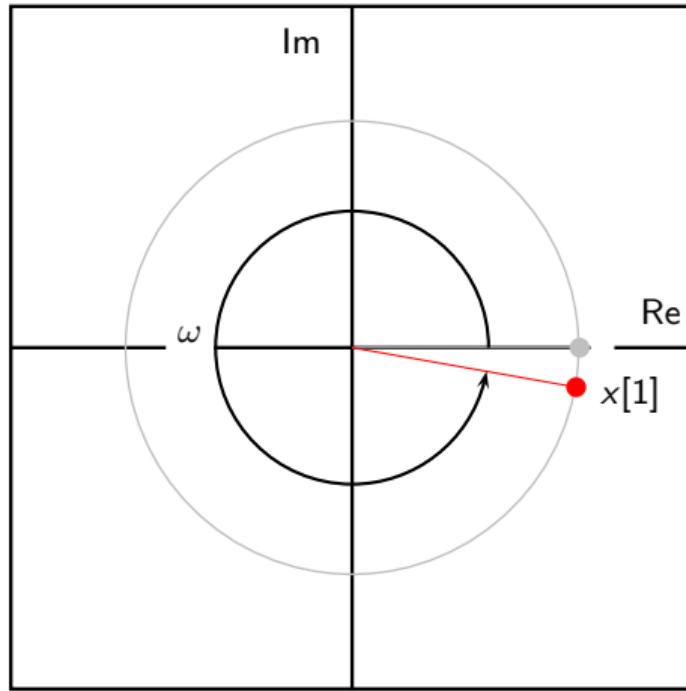
Let's go really too fast

$$\omega = 2\pi - \alpha, \quad \alpha \text{ small}$$



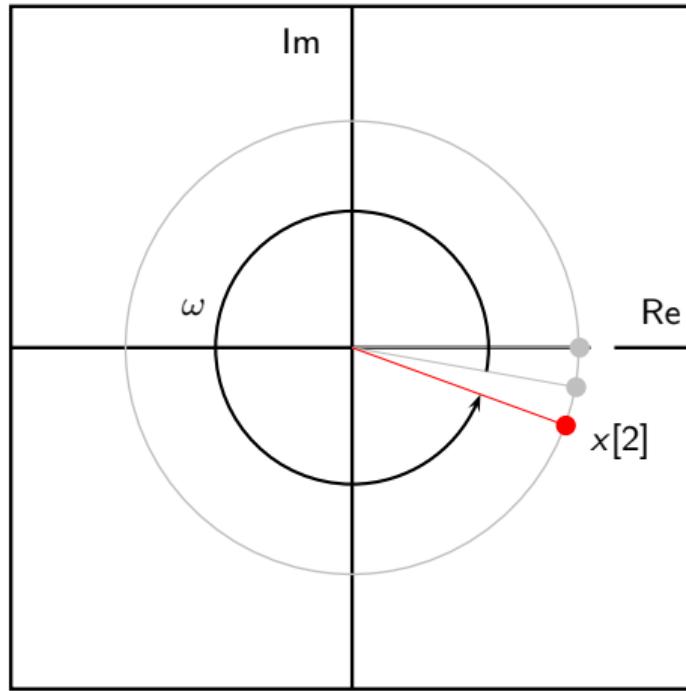
Let's go really too fast

$$\omega = 2\pi - \alpha, \quad \alpha \text{ small}$$



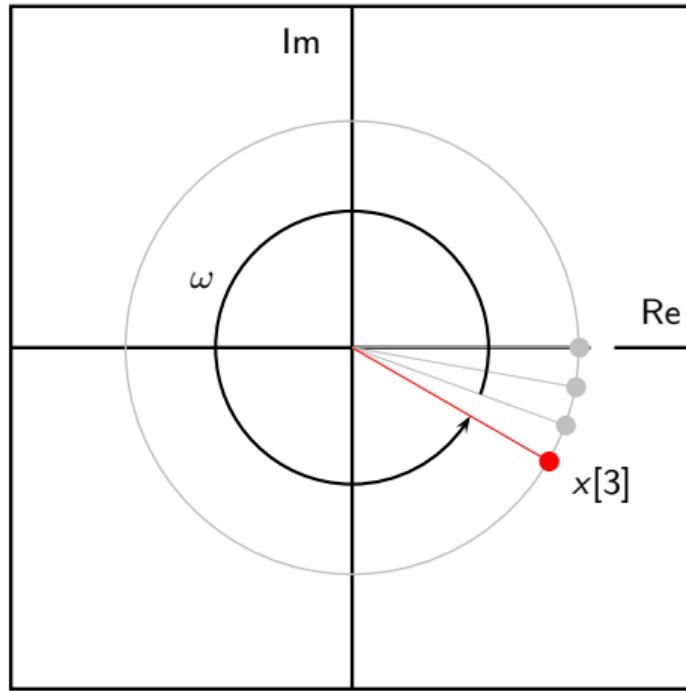
Let's go really too fast

$$\omega = 2\pi - \alpha, \quad \alpha \text{ small}$$



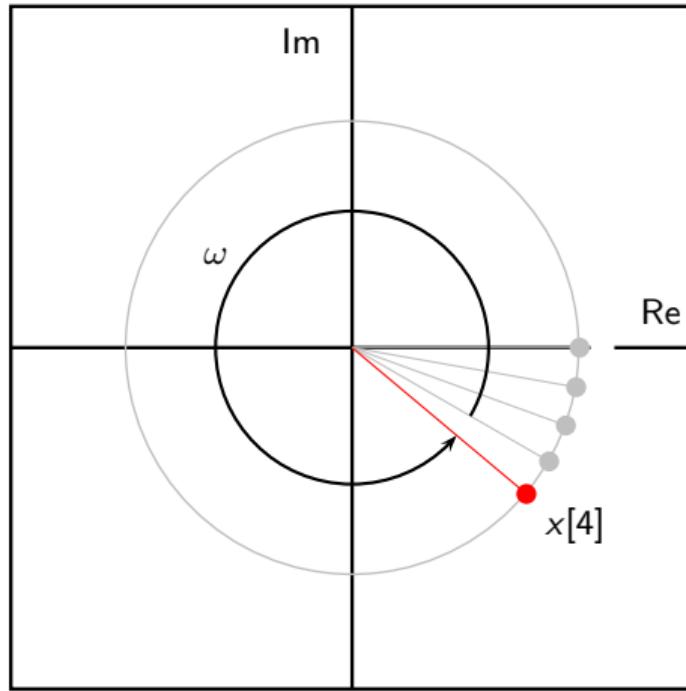
Let's go really too fast

$$\omega = 2\pi - \alpha, \quad \alpha \text{ small}$$



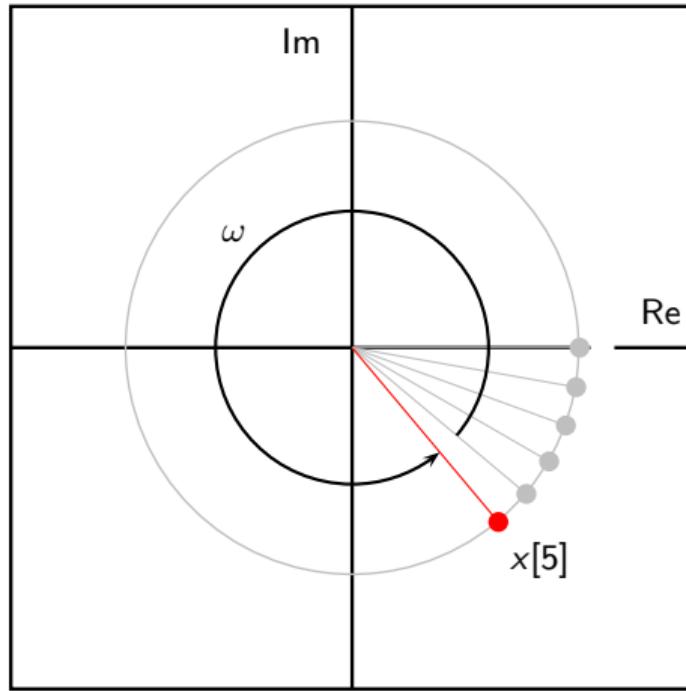
Let's go really too fast

$$\omega = 2\pi - \alpha, \quad \alpha \text{ small}$$



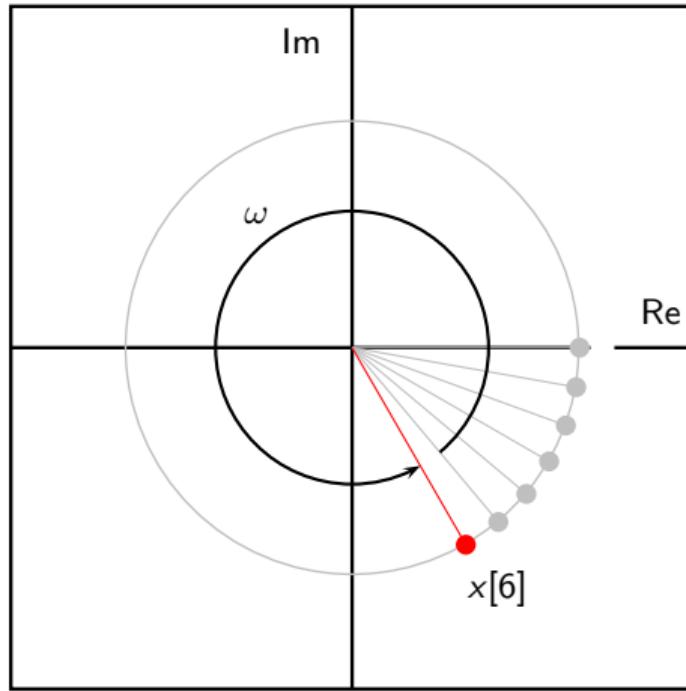
Let's go really too fast

$$\omega = 2\pi - \alpha, \quad \alpha \text{ small}$$



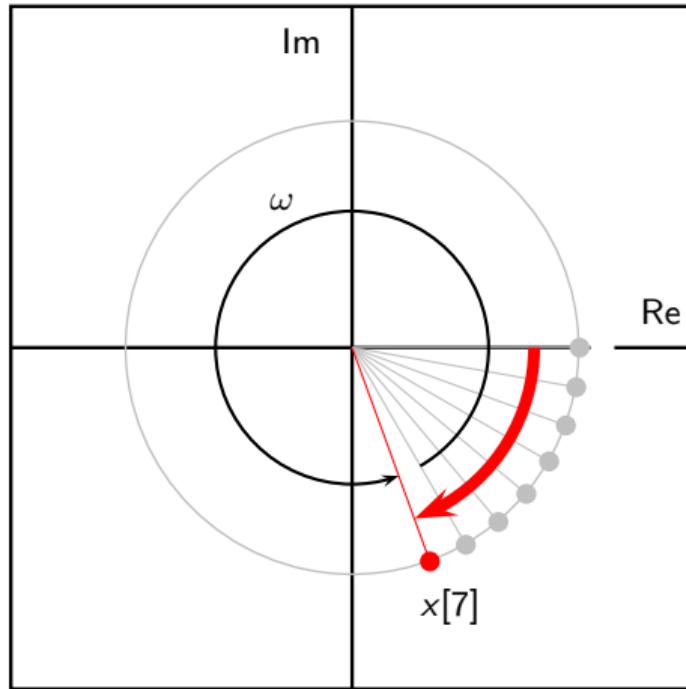
Let's go really too fast

$$\omega = 2\pi - \alpha, \quad \alpha \text{ small}$$



Let's go really too fast

$$\omega = 2\pi - \alpha, \quad \alpha \text{ small}$$



The wagonwheel effect

Summary

- ▶ $x[n] = e^{j(\omega n + \phi)}$ is the prototypical DSP oscillation
- ▶ discrete-time oscillations are periodic ONLY if frequency a rational multiple of π
- ▶ in discrete time, ω and $\omega + 2k\pi$ are indistinguishable frequencies

Digital vs physical frequency

- ▶ Discrete time:
 - n : no physical dimension (just a counter)
 - periodicity: how many samples before pattern repeats
- ▶ “Real world”:
 - periodicity: how many *seconds* before pattern repeats
 - frequency measured in Hz (s^{-1})

Digital vs physical frequency

- ▶ Discrete time:
 - n : no physical dimension (just a counter)
 - periodicity: how many samples before pattern repeats
- ▶ “Real world”:
 - periodicity: how many *seconds* before pattern repeats
 - frequency measured in Hz (s^{-1})

Digital vs physical frequency

- ▶ Discrete time:
 - n : no physical dimension (just a counter)
 - periodicity: how many samples before pattern repeats
- ▶ “Real world”:
 - periodicity: how many *seconds* before pattern repeats
 - frequency measured in Hz (s^{-1})

Digital vs physical frequency

- ▶ Discrete time:
 - n : no physical dimension (just a counter)
 - periodicity: how many samples before pattern repeats
- ▶ “Real world”:
 - periodicity: how many *seconds* before pattern repeats
 - frequency measured in Hz (s^{-1})

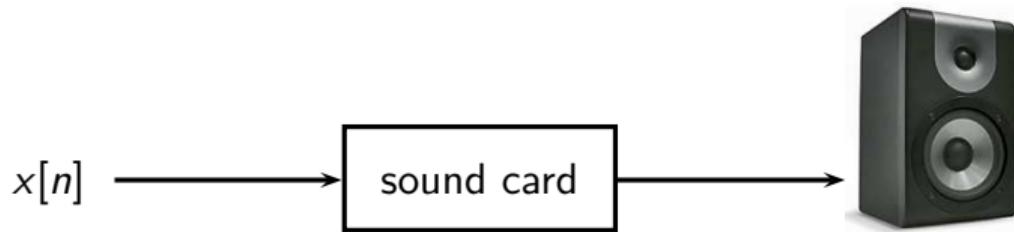
Digital vs physical frequency

- ▶ Discrete time:
 - n : no physical dimension (just a counter)
 - periodicity: how many samples before pattern repeats
- ▶ “Real world”:
 - periodicity: how many *seconds* before pattern repeats
 - frequency measured in Hz (s^{-1})

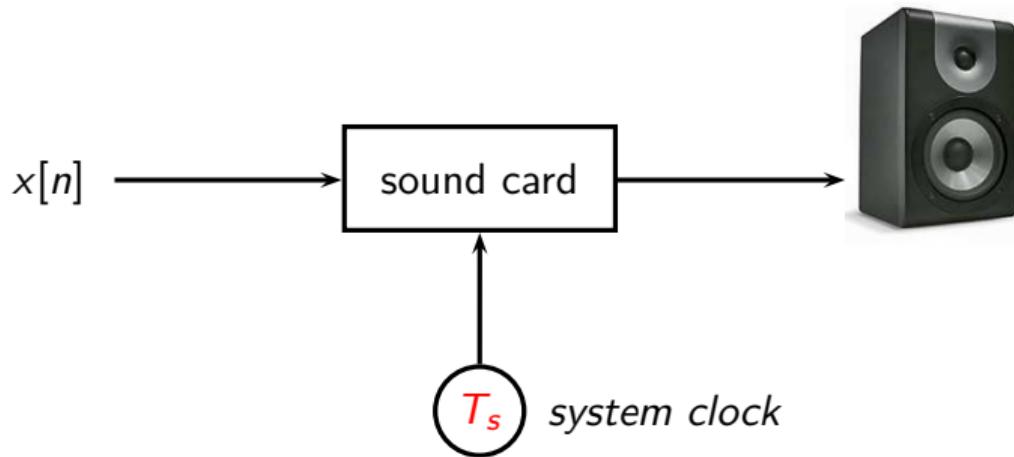
Digital vs physical frequency

- ▶ Discrete time:
 - n : no physical dimension (just a counter)
 - periodicity: how many samples before pattern repeats
- ▶ “Real world”:
 - periodicity: how many *seconds* before pattern repeats
 - frequency measured in Hz (s^{-1})

How your PC plays sounds



How your PC plays sounds



Digital vs physical frequency

- ▶ set T_s , time in seconds between samples
- ▶ periodicity of M samples \longrightarrow periodicity of MT_s seconds
- ▶ real world frequency:

$$f = \frac{1}{MT_s}$$

Digital vs physical frequency

- ▶ set T_s , time in seconds between samples
- ▶ periodicity of M samples \rightarrow periodicity of MT_s seconds
- ▶ real world frequency:

$$f = \frac{1}{MT_s}$$

Digital vs physical frequency

- ▶ set T_s , time in seconds between samples
- ▶ periodicity of M samples \rightarrow periodicity of MT_s seconds
- ▶ real world frequency:

$$f = \frac{1}{MT_s}$$

the Fourier basis

The Fourier Basis for \mathbb{C}^N

Claim: the set of N signals in \mathbb{C}^N

$$w_k[n] = e^{j\frac{2\pi}{N}nk}, \quad n, k = 0, 1, \dots, N-1$$

is an orthogonal basis in \mathbb{C}^N .

The Fourier Basis for \mathbb{C}^N

In vector notation:

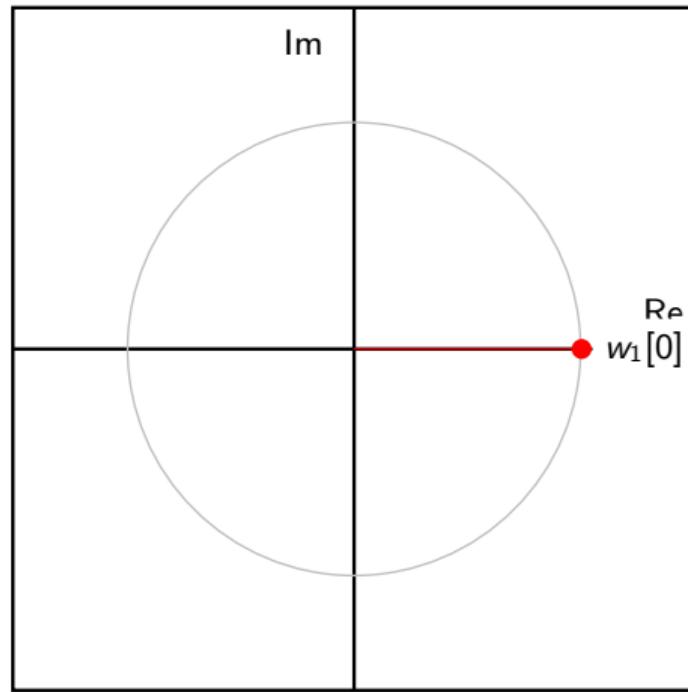
$$\{\mathbf{w}^{(k)}\}_{k=0,1,\dots,N-1}$$

with

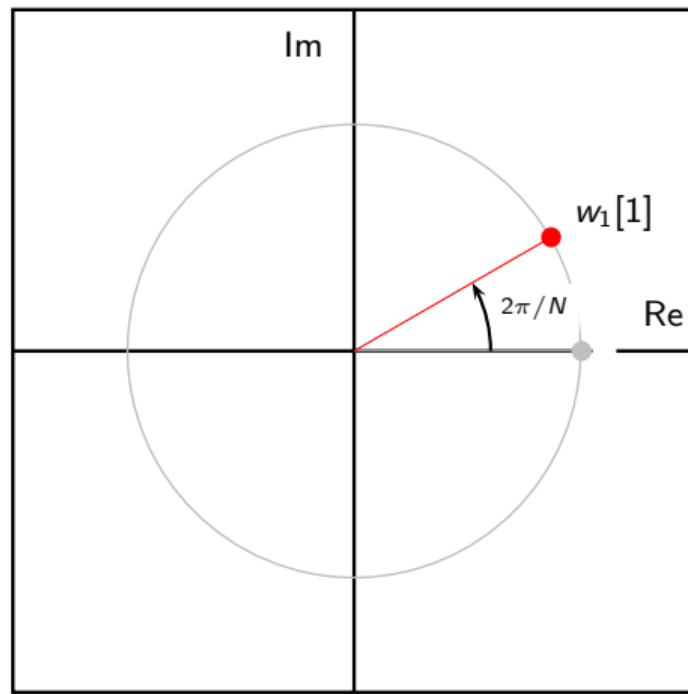
$$w_n^{(k)} = e^{j \frac{2\pi}{N} nk}$$

is an orthogonal basis in \mathbb{C}^N

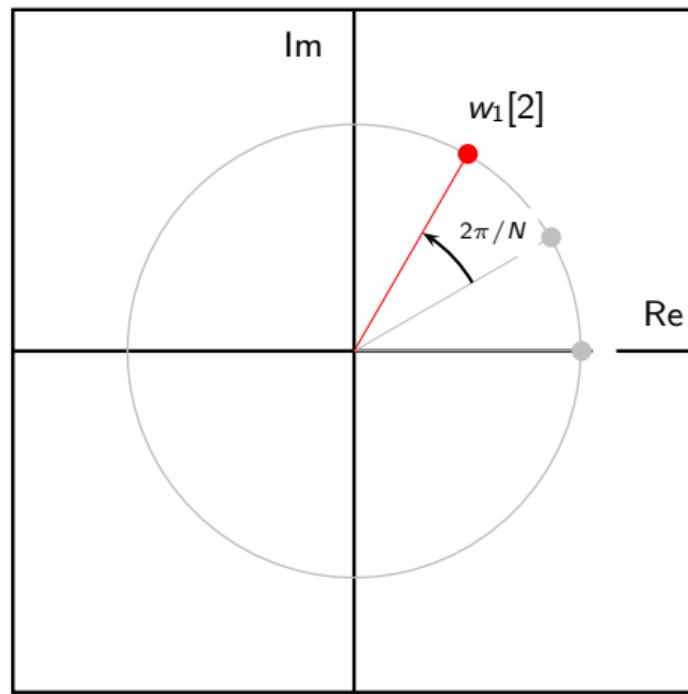
Recall the complex exponential generating machine...



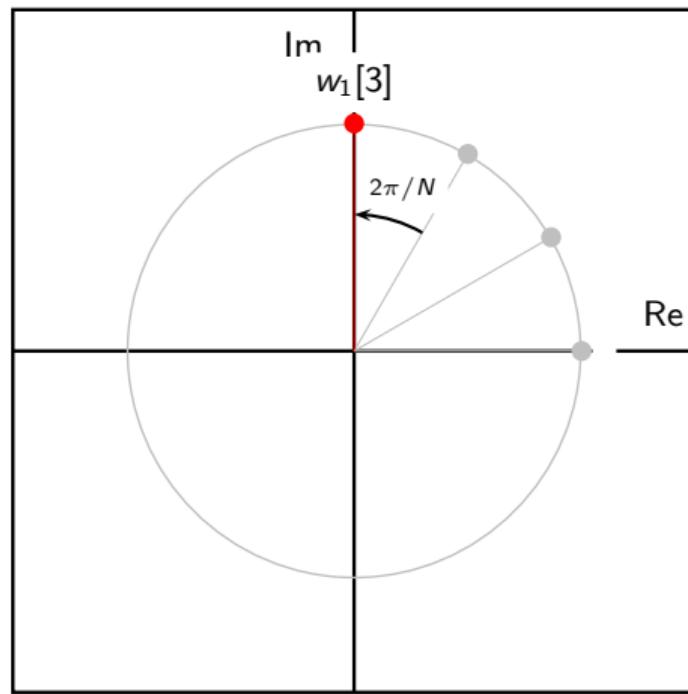
Recall the complex exponential generating machine...



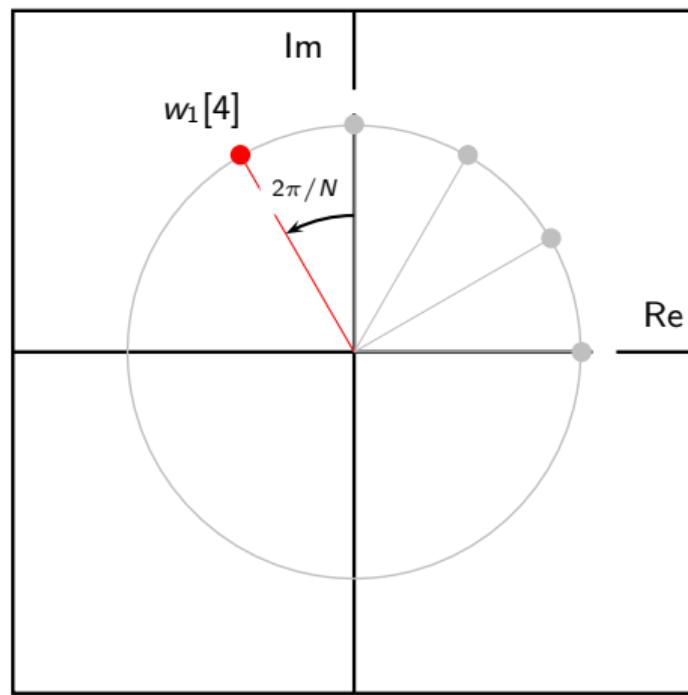
Recall the complex exponential generating machine...



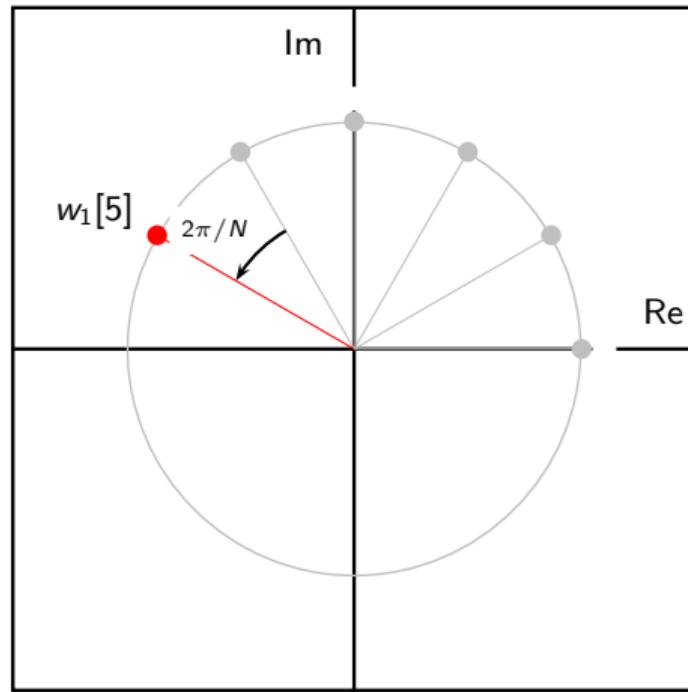
Recall the complex exponential generating machine...



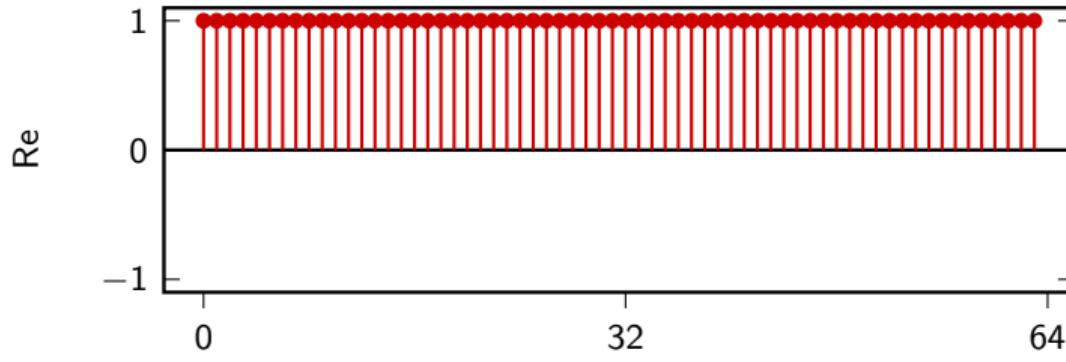
Recall the complex exponential generating machine...



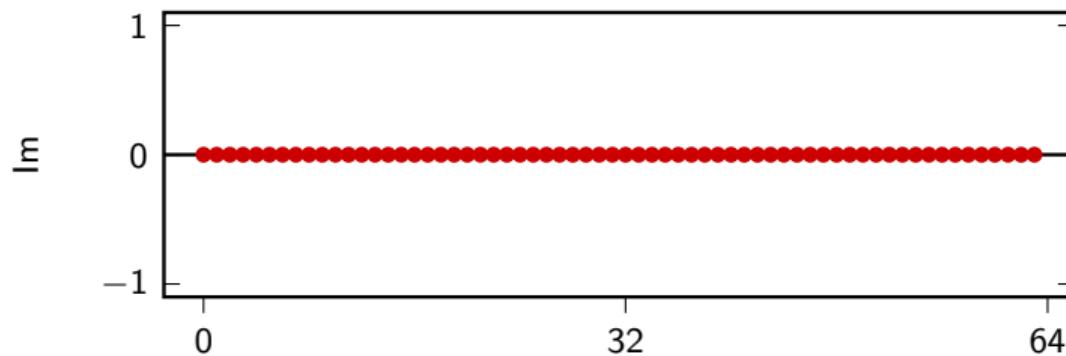
Recall the complex exponential generating machine...



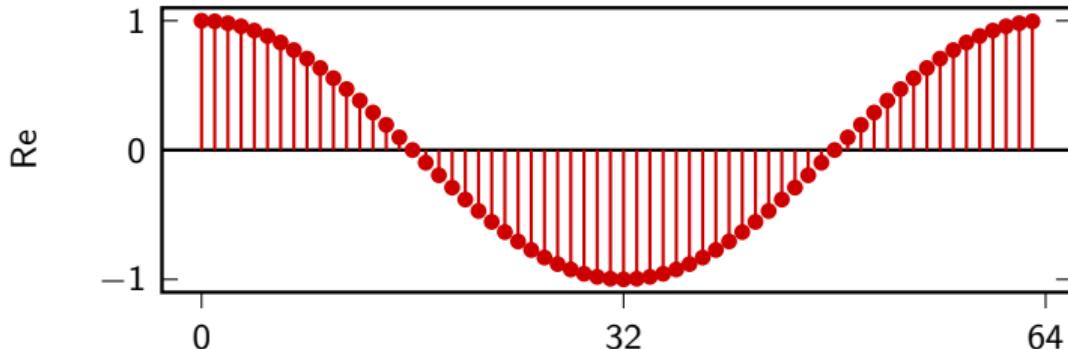
Basis vector $\mathbf{w}^{(0)} \in \mathbb{C}^{64}$



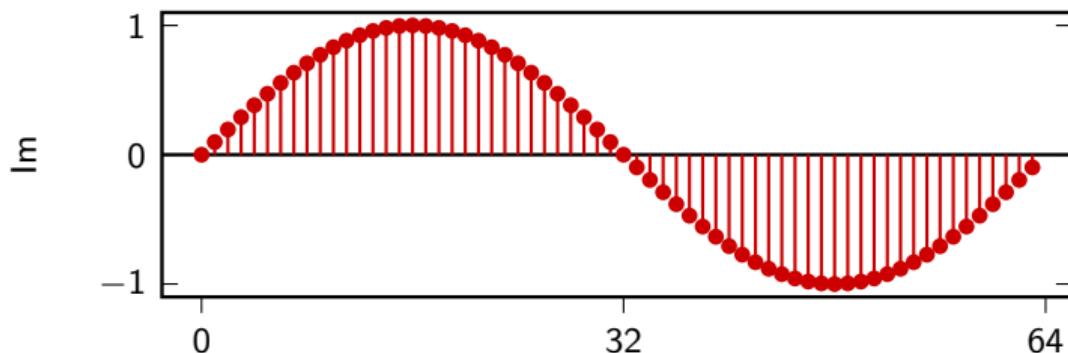
$$\omega_0 = 0$$



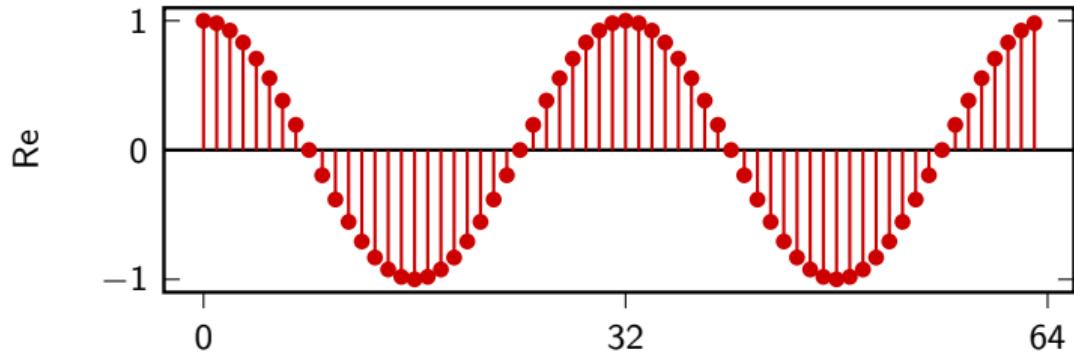
Basis vector $\mathbf{w}^{(1)} \in \mathbb{C}^{64}$



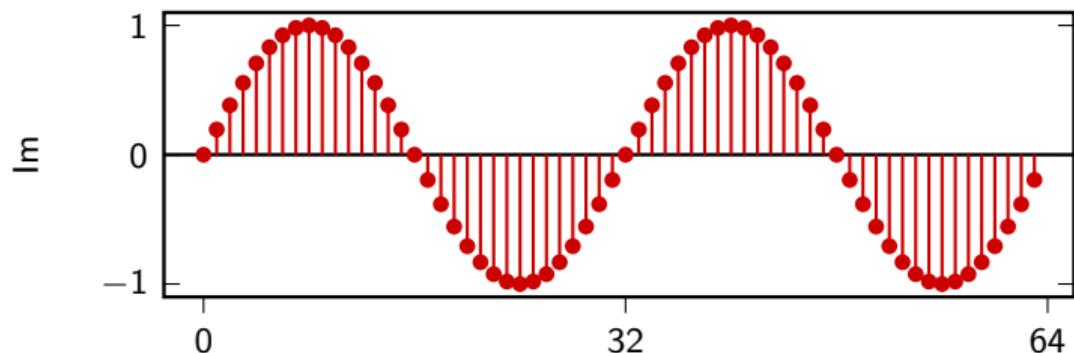
$$\omega_1 = \frac{2\pi}{64}$$



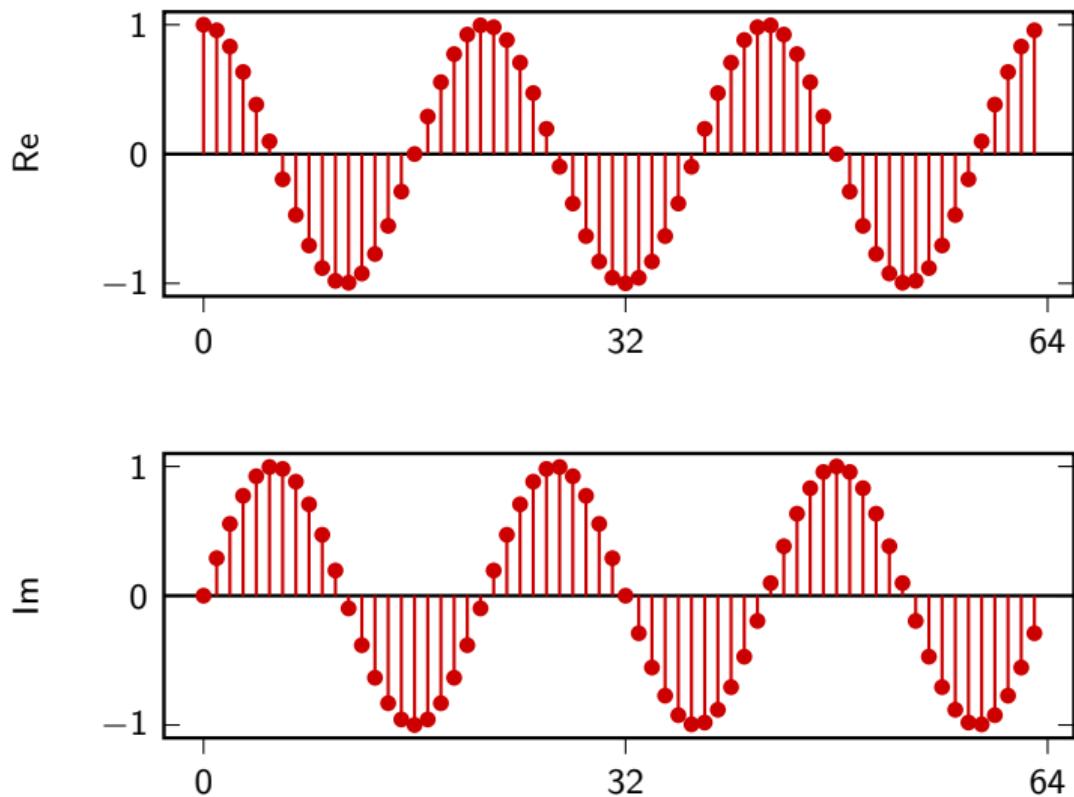
Basis vector $\mathbf{w}^{(2)} \in \mathbb{C}^{64}$



$$\omega_2 = 2\pi \frac{2}{64}$$

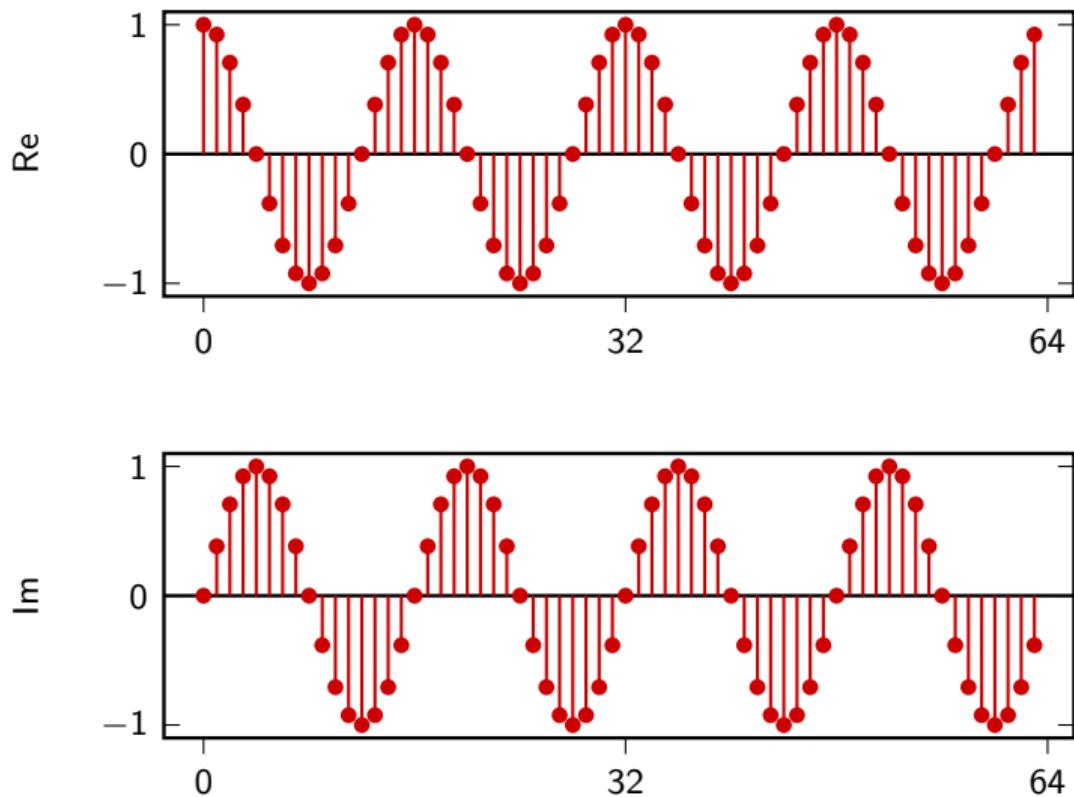


Basis vector $\mathbf{w}^{(3)} \in \mathbb{C}^{64}$



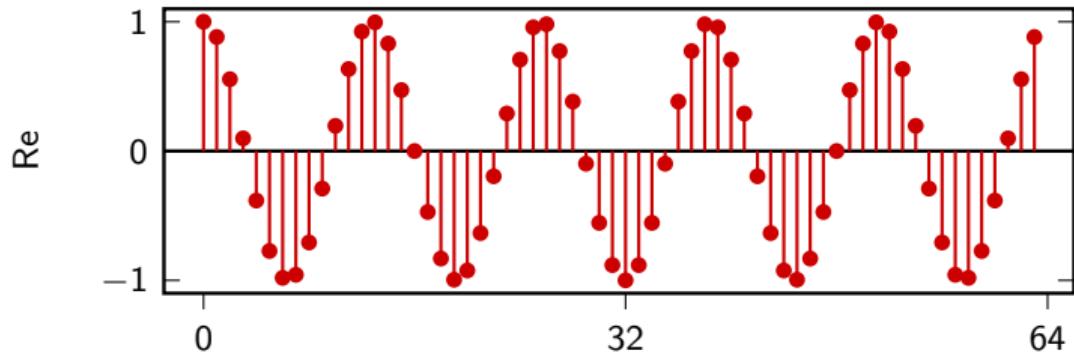
$$\omega_3 = 2\pi \frac{3}{64}$$

Basis vector $\mathbf{w}^{(4)} \in \mathbb{C}^{64}$

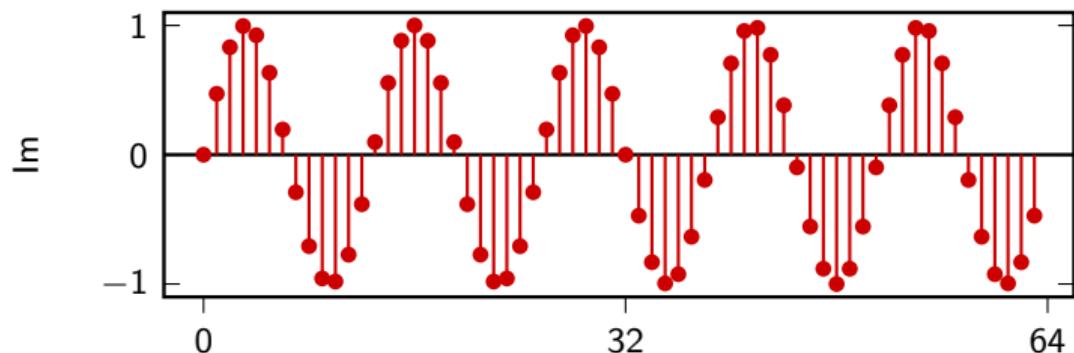


$$\omega_4 = 2\pi \frac{4}{64}$$

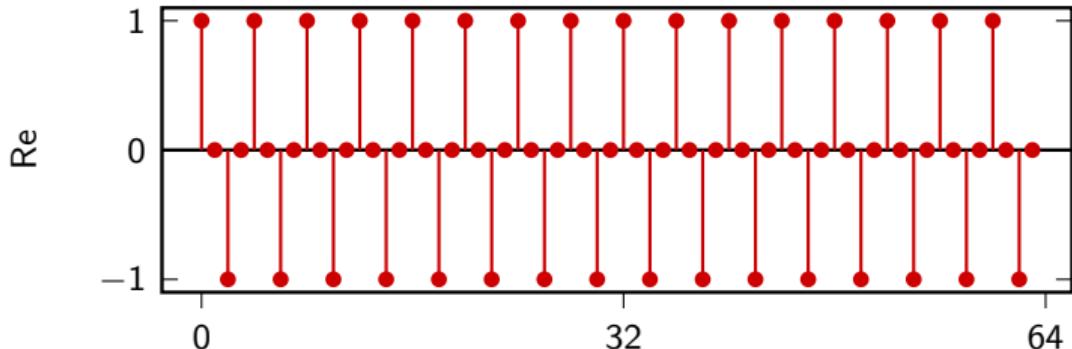
Basis vector $\mathbf{w}^{(5)} \in \mathbb{C}^{64}$



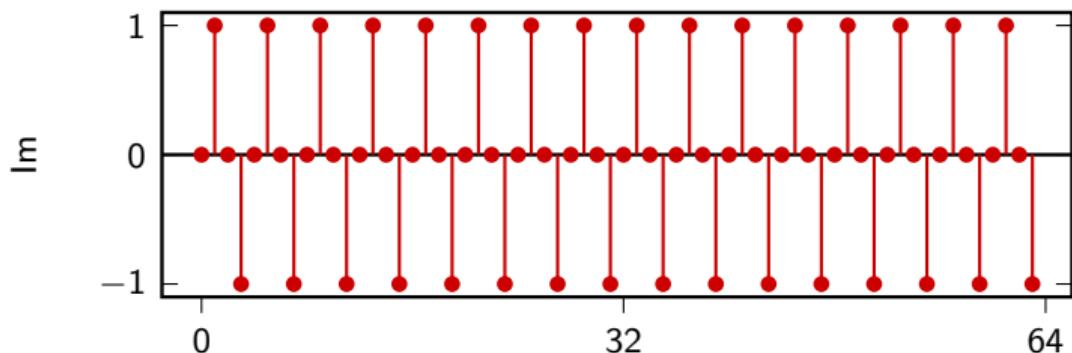
$$\omega_5 = 2\pi \frac{5}{64}$$



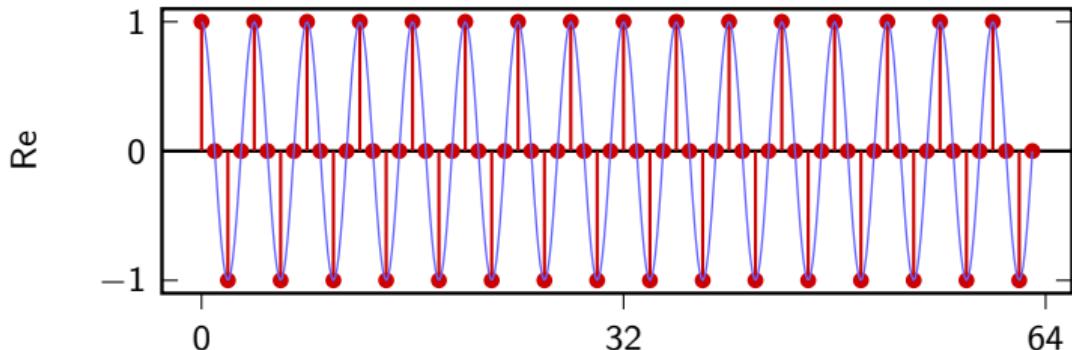
Basis vector $\mathbf{w}^{(16)} \in \mathbb{C}^{64}$



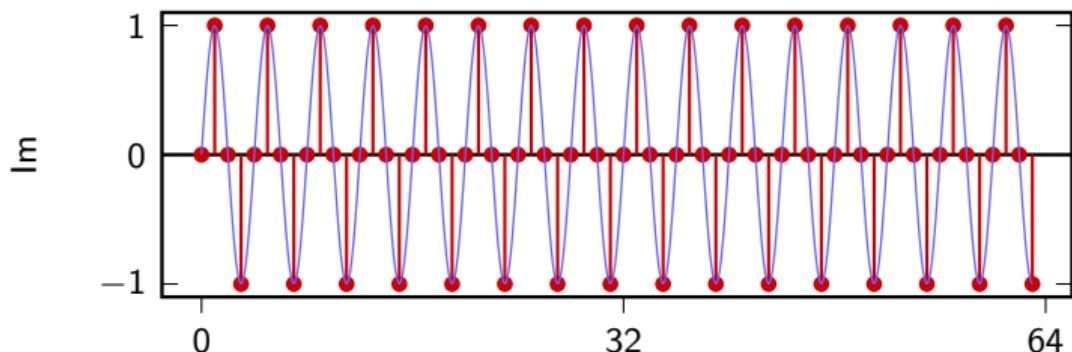
$$\omega_{16} = 2\pi \frac{16}{64}$$



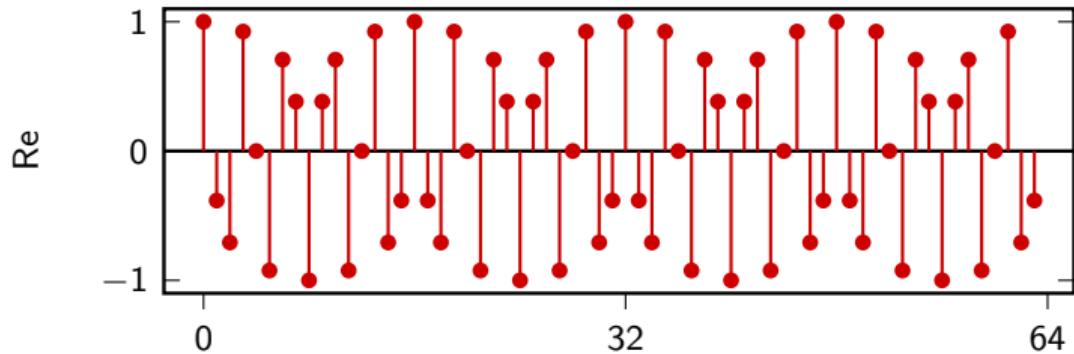
Basis vector $\mathbf{w}^{(16)} \in \mathbb{C}^{64}$



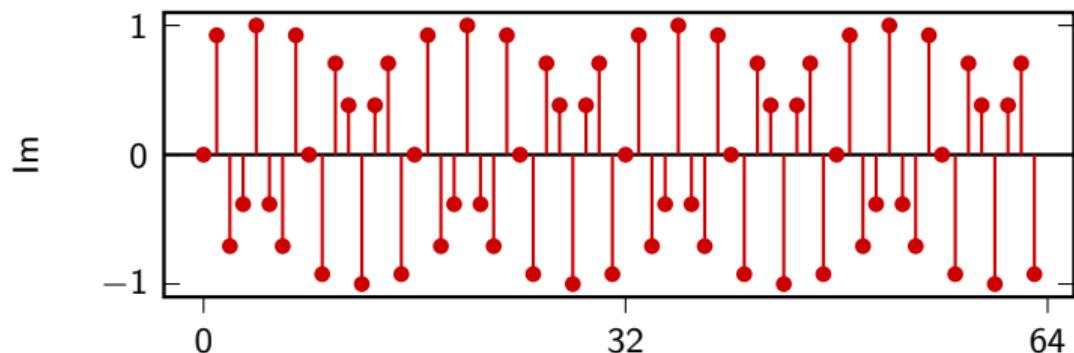
$$\omega_{16} = 2\pi \frac{16}{64}$$



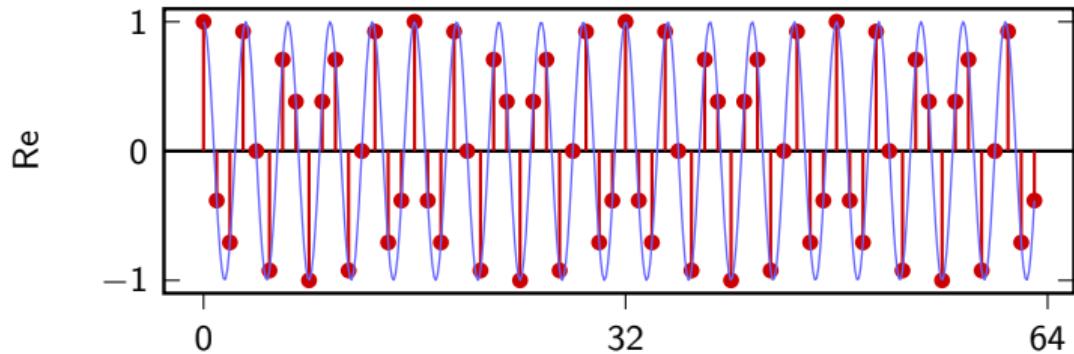
Basis vector $\mathbf{w}^{(20)} \in \mathbb{C}^{64}$



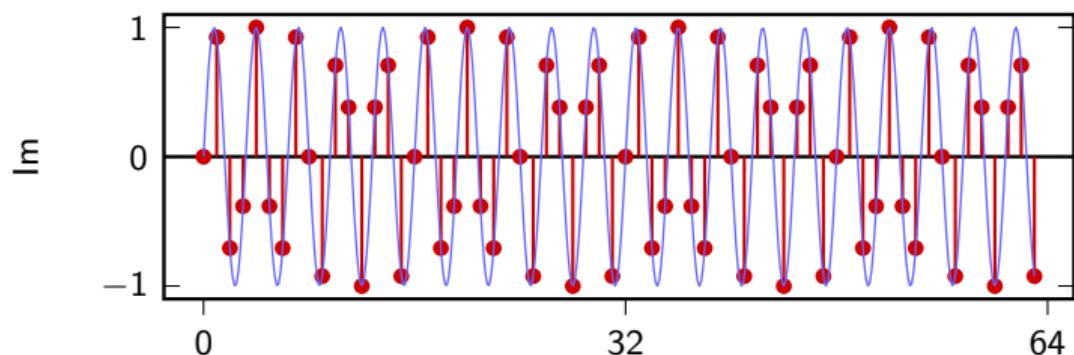
$$\omega_{20} = 2\pi \frac{20}{64}$$



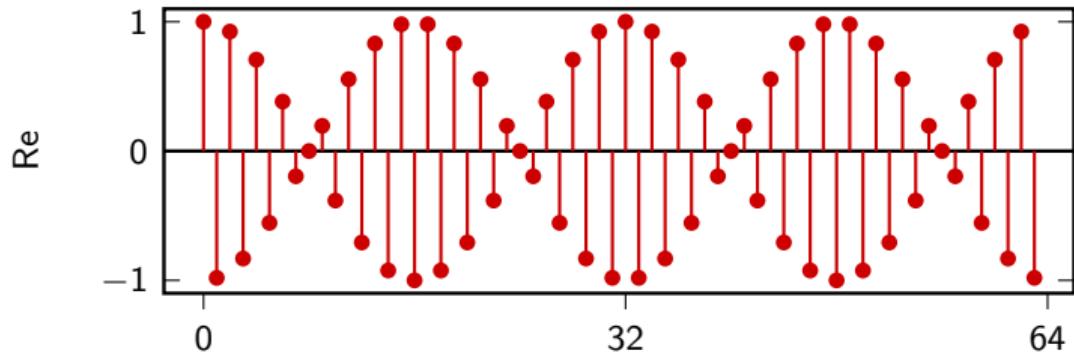
Basis vector $\mathbf{w}^{(20)} \in \mathbb{C}^{64}$



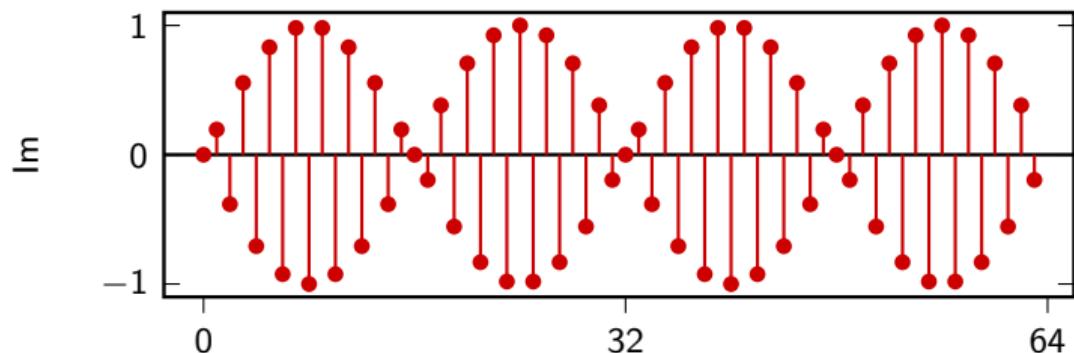
$$\omega_{20} = 2\pi \frac{20}{64}$$



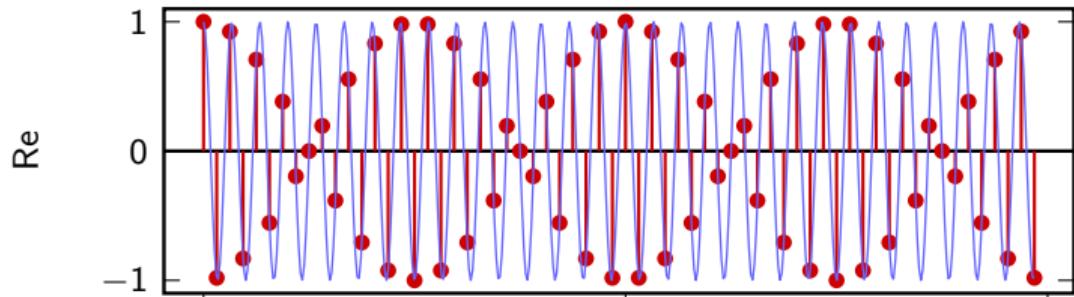
Basis vector $\mathbf{w}^{(30)} \in \mathbb{C}^{64}$



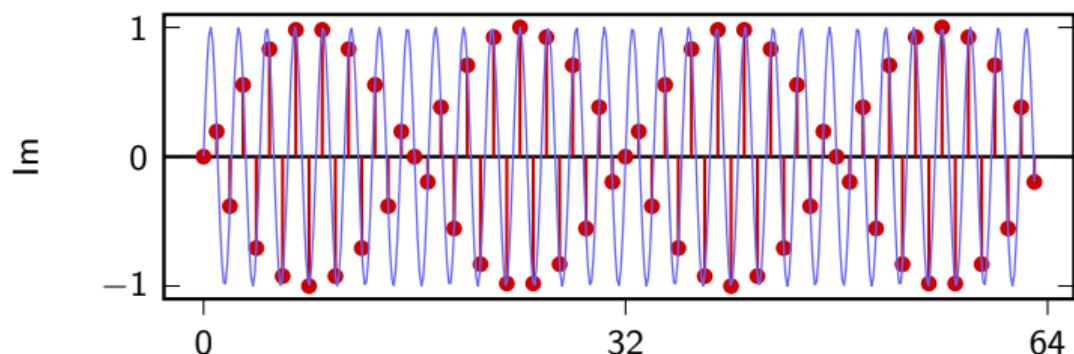
$$\omega_{30} = 2\pi \frac{30}{64}$$



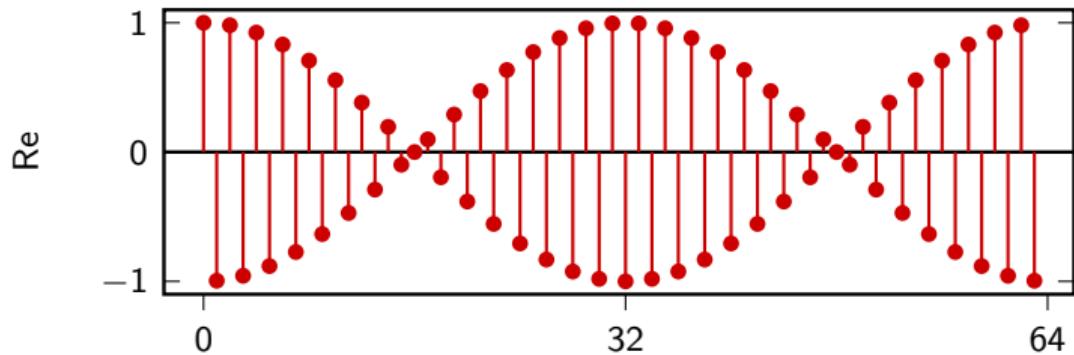
Basis vector $\mathbf{w}^{(30)} \in \mathbb{C}^{64}$



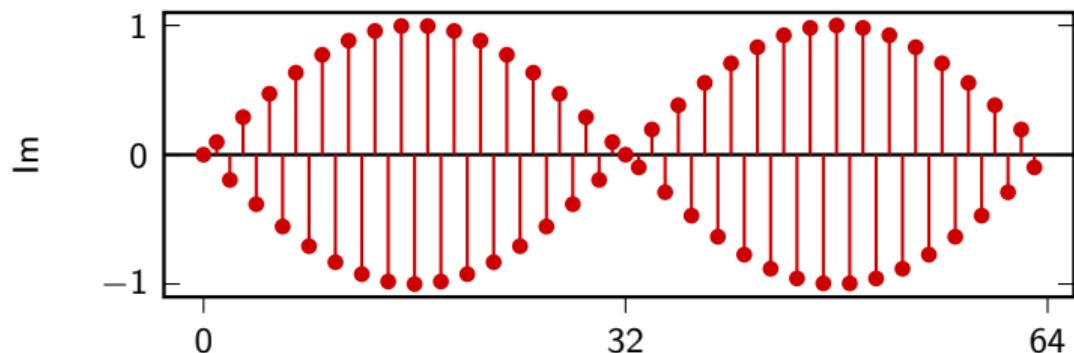
$$\omega_{30} = 2\pi \frac{30}{64}$$



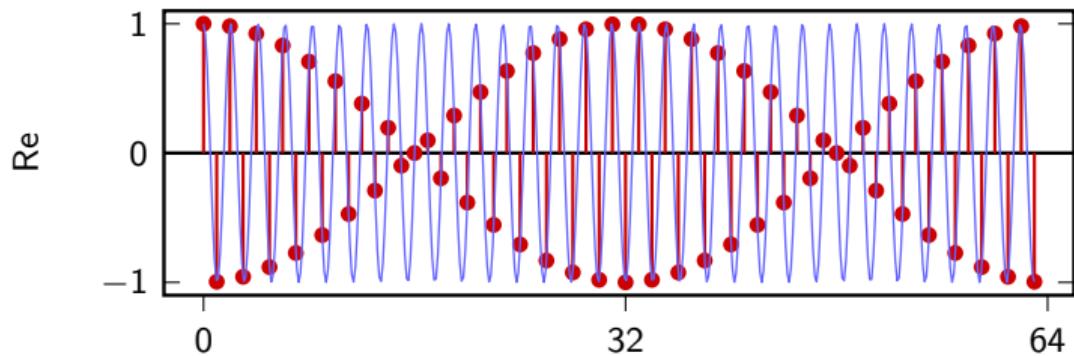
Basis vector $\mathbf{w}^{(31)} \in \mathbb{C}^{64}$



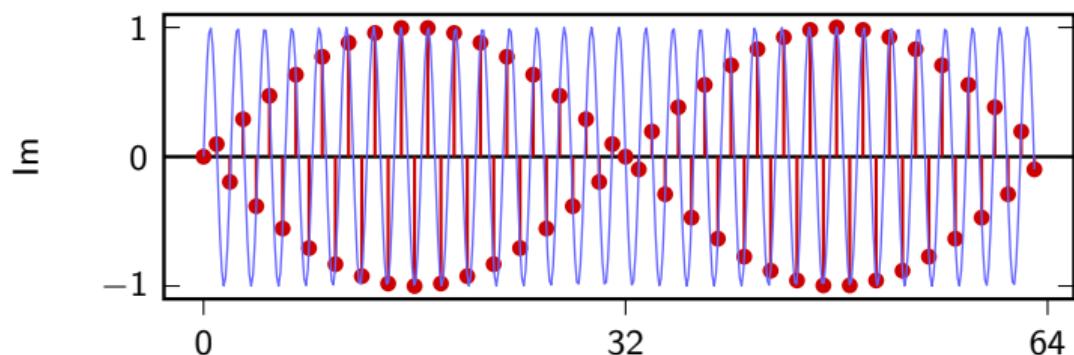
$$\omega_{31} = 2\pi \frac{31}{64}$$



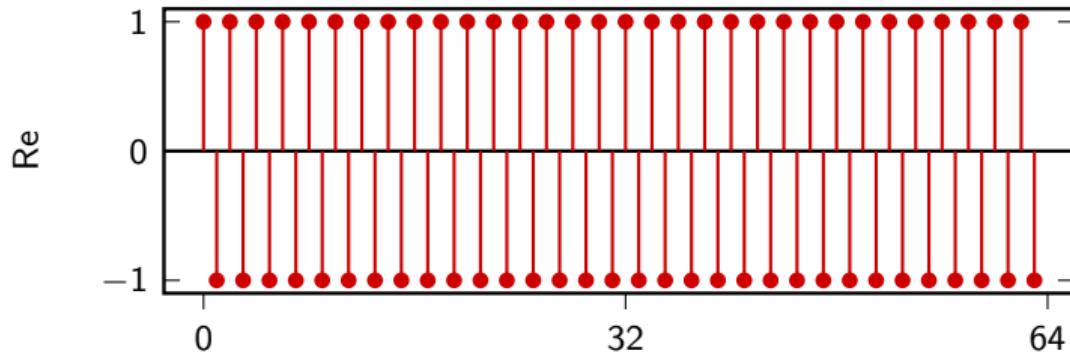
Basis vector $\mathbf{w}^{(31)} \in \mathbb{C}^{64}$



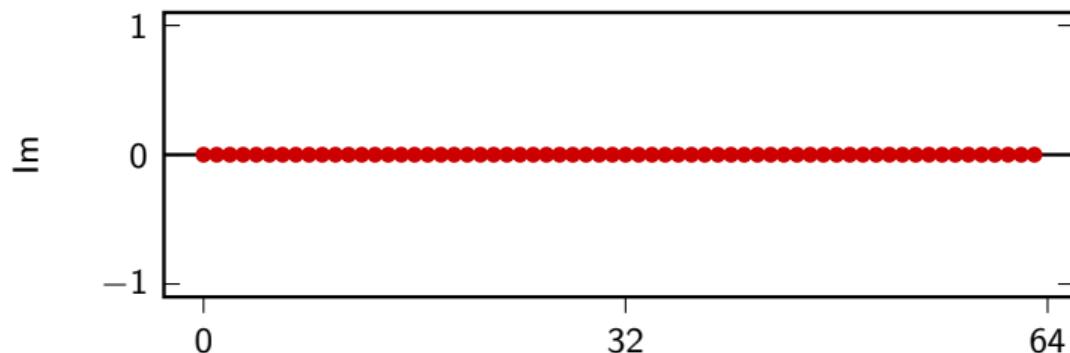
$$\omega_{31} = 2\pi \frac{31}{64}$$



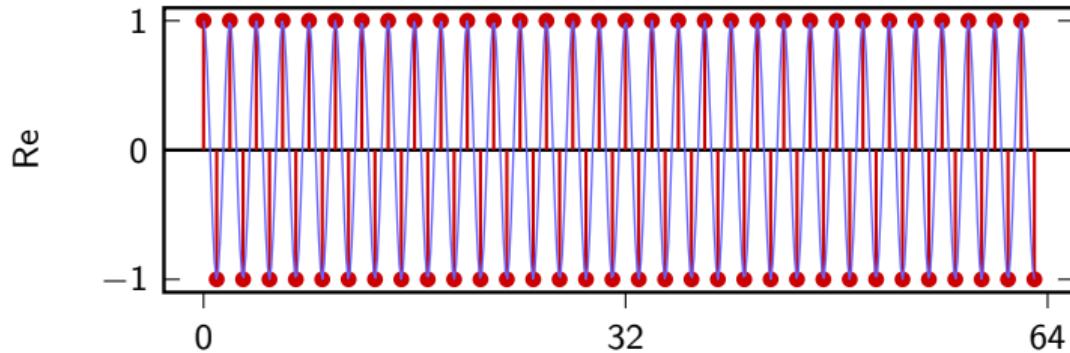
Basis vector $\mathbf{w}^{(32)} \in \mathbb{C}^{64}$



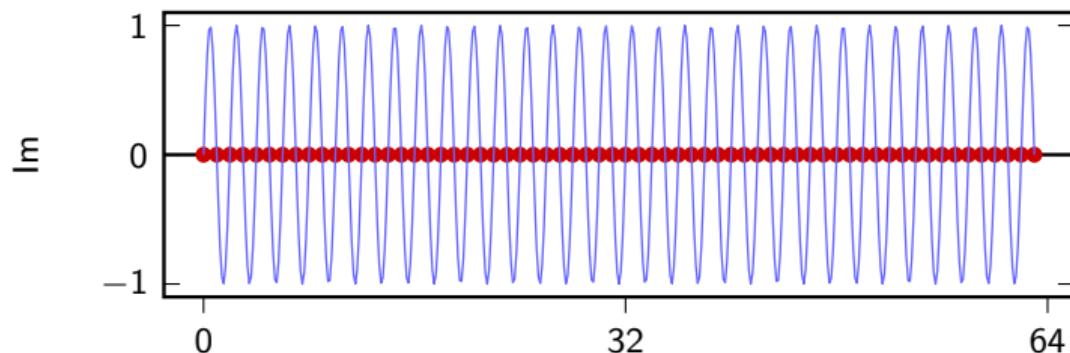
$$\omega_{32} = 2\pi \frac{32}{64}$$



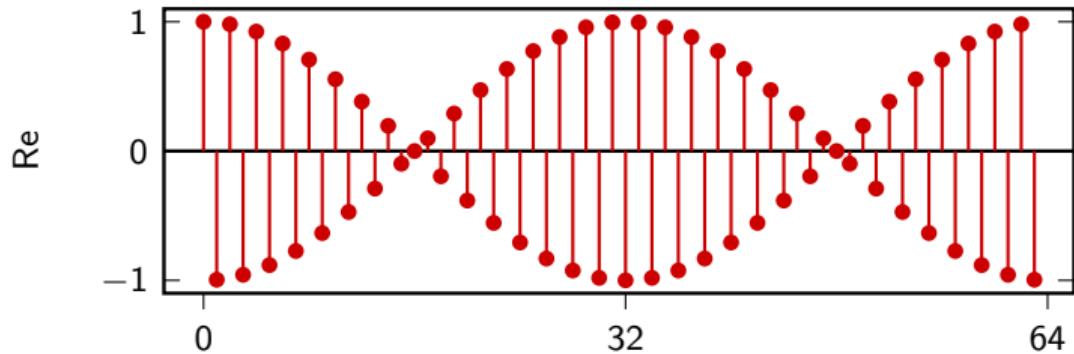
Basis vector $\mathbf{w}^{(32)} \in \mathbb{C}^{64}$



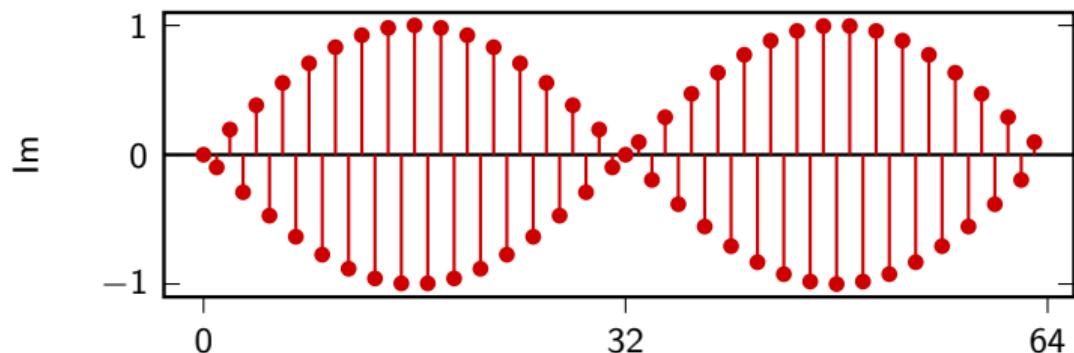
$$\omega_{32} = 2\pi \frac{32}{64}$$



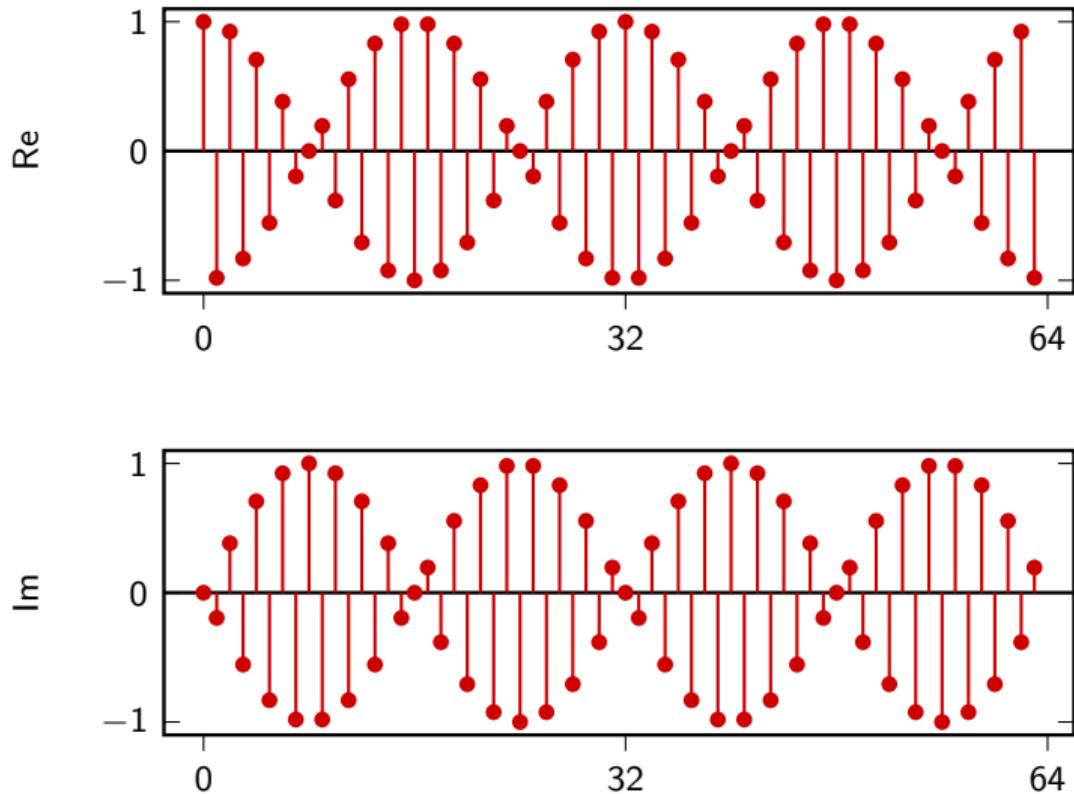
Basis vector $\mathbf{w}^{(33)} \in \mathbb{C}^{64}$



$$\omega_{33} = 2\pi \frac{33}{64}$$

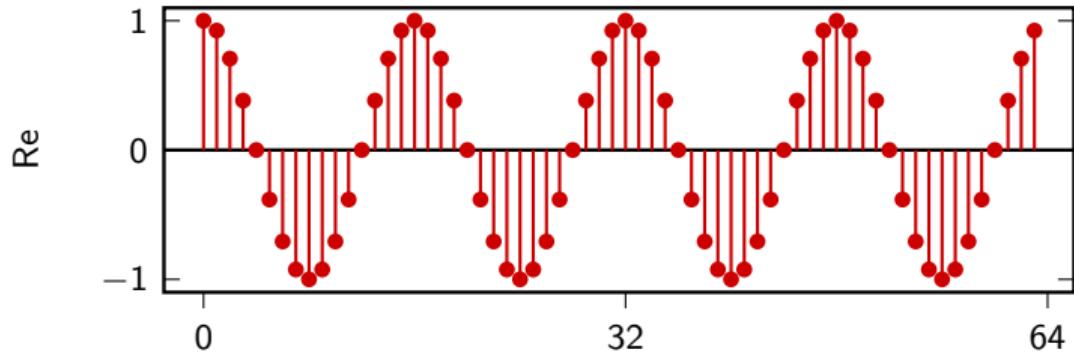


Basis vector $\mathbf{w}^{(34)} \in \mathbb{C}^{64}$

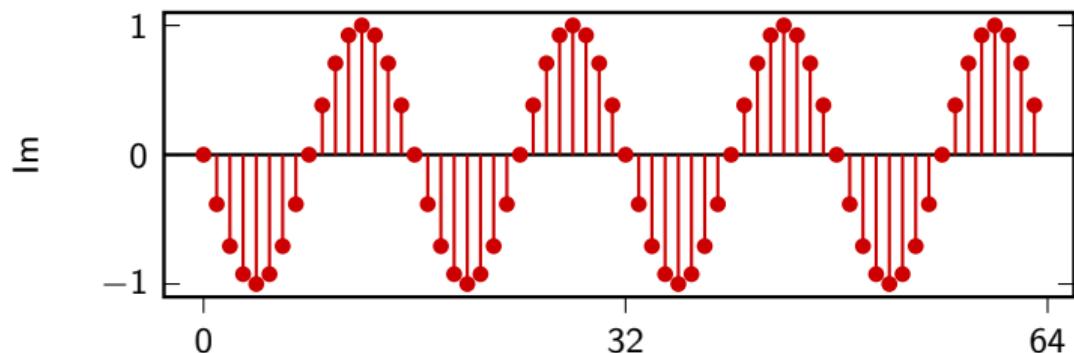


$$\omega_{34} = 2\pi \frac{34}{64}$$

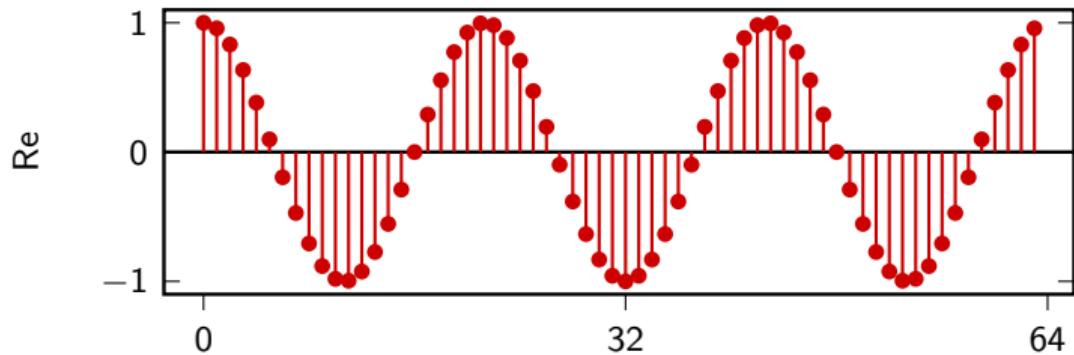
Basis vector $\mathbf{w}^{(60)} \in \mathbb{C}^{64}$



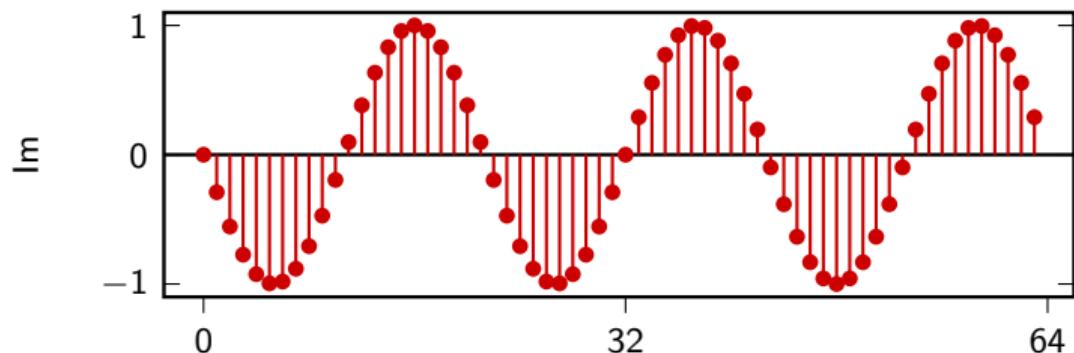
$$\omega_{60} = 2\pi \frac{60}{64}$$



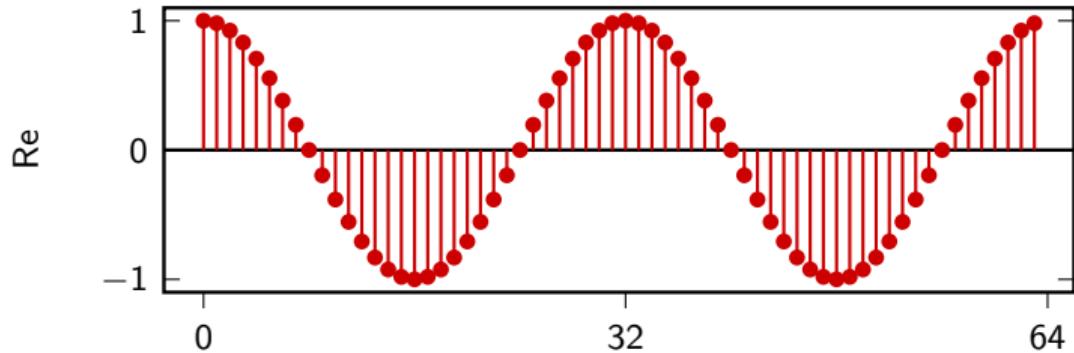
Basis vector $\mathbf{w}^{(61)} \in \mathbb{C}^{64}$



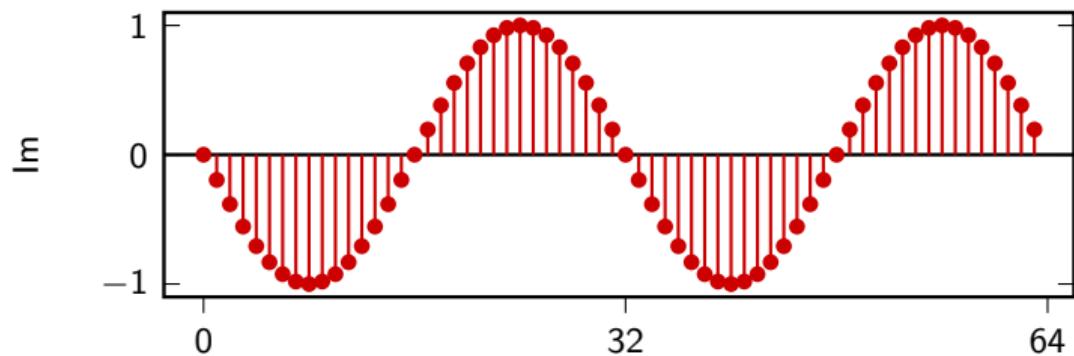
$$\omega_{61} = 2\pi \frac{61}{64}$$



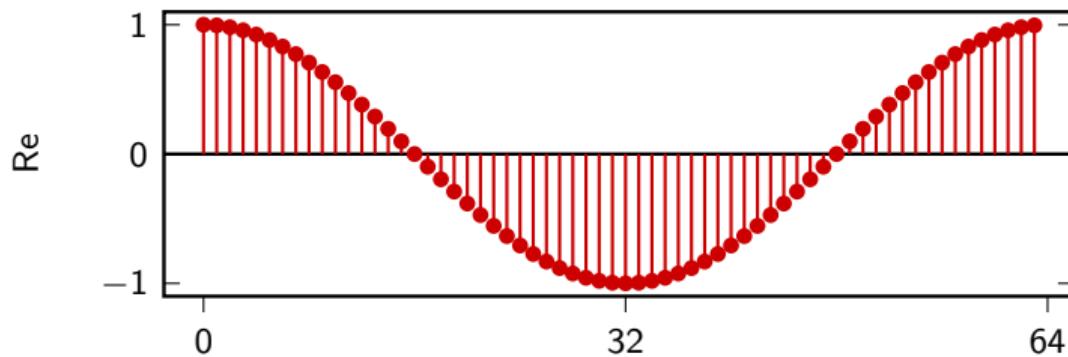
Basis vector $\mathbf{w}^{(62)} \in \mathbb{C}^{64}$



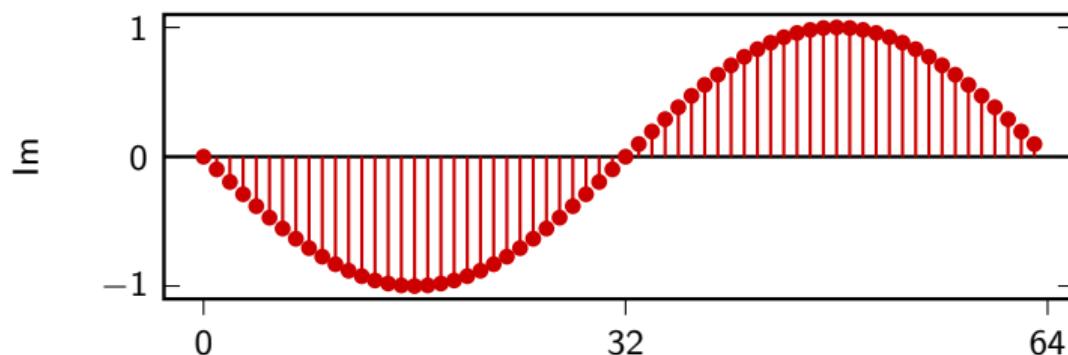
$$\omega_{62} = 2\pi \frac{62}{64}$$



Basis vector $\mathbf{w}^{(63)} \in \mathbb{C}^{64}$



$$\omega_{63} = 2\pi \frac{63}{64}$$



Proof of orthogonality

$$\begin{aligned}\langle \mathbf{w}^{(k)}, \mathbf{w}^{(h)} \rangle &= \sum_{n=0}^{N-1} (e^{j\frac{2\pi}{N}nk})^* e^{j\frac{2\pi}{N}nh} \\&= \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}(h-k)n} \\&= \begin{cases} N & \text{for } h = k \\ \frac{1 - e^{j2\pi(h-k)}}{1 - e^{j\frac{2\pi}{N}(h-k)}} = 0 & \text{otherwise} \end{cases}\end{aligned}$$

Proof of orthogonality

$$\begin{aligned}\langle \mathbf{w}^{(k)}, \mathbf{w}^{(h)} \rangle &= \sum_{n=0}^{N-1} (e^{j\frac{2\pi}{N}nk})^* e^{j\frac{2\pi}{N}nh} \\&= \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}(h-k)n} \\&= \begin{cases} N & \text{for } h = k \\ \frac{1 - e^{j2\pi(h-k)}}{1 - e^{j\frac{2\pi}{N}(h-k)}} = 0 & \text{otherwise} \end{cases}\end{aligned}$$

Proof of orthogonality

$$\begin{aligned}\langle \mathbf{w}^{(k)}, \mathbf{w}^{(h)} \rangle &= \sum_{n=0}^{N-1} (e^{j\frac{2\pi}{N}nk})^* e^{j\frac{2\pi}{N}nh} \\&= \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}(h-k)n} \\&= \begin{cases} N & \text{for } h = k \\ \frac{1 - e^{j2\pi(h-k)}}{1 - e^{j\frac{2\pi}{N}(h-k)}} = 0 & \text{otherwise} \end{cases}\end{aligned}$$

Proof of orthogonality

$$\begin{aligned}\langle \mathbf{w}^{(k)}, \mathbf{w}^{(h)} \rangle &= \sum_{n=0}^{N-1} (e^{j\frac{2\pi}{N}nk})^* e^{j\frac{2\pi}{N}nh} \\&= \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}(h-k)n} \\&= \begin{cases} N & \text{for } h = k \\ \frac{1 - e^{j2\pi(h-k)}}{1 - e^{j\frac{2\pi}{N}(h-k)}} = 0 & \text{otherwise} \end{cases}\end{aligned}$$

Remarks

- ▶ N orthogonal vectors → basis for \mathbb{C}^N
- ▶ vectors are not orthonormal. Normalization factor would be $1/\sqrt{N}$

Remarks

- ▶ N orthogonal vectors \longrightarrow basis for \mathbb{C}^N
- ▶ vectors are not orthonormal. Normalization factor would be $1/\sqrt{N}$

the Discrete Fourier Transform

The Fourier Basis for \mathbb{C}^N

- ▶ in “signal” notation: $w_k[n] = e^{j\frac{2\pi}{N}nk}$, $n, k = 0, 1, \dots, N - 1$
- ▶ in vector notation: $\{\mathbf{w}^{(k)}\}_{k=0,1,\dots,N-1}$ with $w_n^{(k)} = e^{j\frac{2\pi}{N}nk}$

The Fourier Basis for \mathbb{C}^N

- ▶ in “signal” notation: $w_k[n] = e^{j\frac{2\pi}{N}nk}$, $n, k = 0, 1, \dots, N - 1$
- ▶ in vector notation: $\{\mathbf{w}^{(k)}\}_{k=0,1,\dots,N-1}$ with $w_n^{(k)} = e^{j\frac{2\pi}{N}nk}$

The Fourier Basis for \mathbb{C}^N

- ▶ N orthogonal vectors → basis for \mathbb{C}^N
- ▶ vectors are not orthonormal. Normalization factor would be $1/\sqrt{N}$
- ▶ will keep normalization factor explicit in DFT formulas

Basis expansion

Analysis formula:

$$X_k = \langle \mathbf{w}^{(k)}, \mathbf{x} \rangle$$

Synthesis formula:

$$\mathbf{x} = \frac{1}{N} \sum_{k=0}^{N-1} X_k \mathbf{w}^{(k)}$$

Basis expansion (signal notation)

Analysis formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}, \quad k = 0, 1, \dots, N-1$$

N-point signal in the *frequency domain*

Synthesis formula:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}nk}, \quad n = 0, 1, \dots, N-1$$

N-point signal in the “*time*” domain

Basis expansion (signal notation)

Analysis formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}, \quad k = 0, 1, \dots, N-1$$

N-point signal in the *frequency domain*

Synthesis formula:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}nk}, \quad n = 0, 1, \dots, N-1$$

N-point signal in the “*time*” domain

Change of basis in matrix form

Define $W_N = e^{-j\frac{2\pi}{N}}$
(or simply W when N is evident from the context)

Change of basis matrix \mathbf{W} with $\mathbf{W}[n, m] = W_N^{nm}$:

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W^1 & W^2 & W^3 & \dots & W^{N-1} \\ 1 & W^2 & W^4 & W^6 & \dots & W^{2(N-1)} \\ & & & \ddots & & \\ 1 & W^{N-1} & W^{2(N-1)} & W^{3(N-1)} & \dots & W^{(N-1)^2} \end{bmatrix}$$

Change of basis in matrix form

Define $W_N = e^{-j\frac{2\pi}{N}}$
(or simply W when N is evident from the context)

Change of basis matrix \mathbf{W} with $\mathbf{W}[n, m] = W_N^{nm}$:

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W^1 & W^2 & W^3 & \dots & W^{N-1} \\ 1 & W^2 & W^4 & W^6 & \dots & W^{2(N-1)} \\ & & & \ddots & & \\ 1 & W^{N-1} & W^{2(N-1)} & W^{3(N-1)} & \dots & W^{(N-1)^2} \end{bmatrix}$$

Change of basis in matrix form

Analysis formula:

$$\mathbf{X} = \mathbf{W}\mathbf{x}$$

Synthesis formula:

$$\mathbf{x} = \frac{1}{N} \mathbf{W}^H \mathbf{X}$$

DFT Matrix

$$W_N^m = W_N^{(m \mod N)}$$

e.g. $W_8^{11} = W_8^3$

DFT Matrix

$$W_N^m = W_N^{(m \mod N)}$$

e.g. $W_8^{11} = W_8^3$

Small DFT matrices: $N = 2, 3$

$$W_2 = e^{-j\frac{2\pi}{2}} = -1$$

$$\mathbf{W}_2 = \begin{bmatrix} 1 & 1 \\ 1 & W \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$W_3 = e^{-j\frac{2\pi}{3}} = -(1 + j\sqrt{3})/2$$

$$\mathbf{W}_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & W & W^2 \\ 1 & W^2 & W^4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & W & W^2 \\ 1 & W^2 & W \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -(1 + j\sqrt{3})/2 & -(1 - j\sqrt{3})/2 \\ 1 & -(1 - j\sqrt{3})/2 & (1 - j\sqrt{3})/2 \end{bmatrix}$$

Small DFT matrices: $N = 4$

$$W_4 = e^{-j\frac{2\pi}{4}} = e^{-j\frac{\pi}{2}} = -j$$

$$\mathbf{W}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 \\ 1 & W^2 & W^4 & W^6 \\ 1 & W^3 & W^6 & W^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 \\ 1 & W^2 & 1 & W^2 \\ 1 & W^3 & W^2 & W \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

Small DFT matrices: $N = 5$

$$\mathbf{W}_5 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 & W^4 \\ 1 & W^2 & W^4 & W^6 & W^8 \\ 1 & W^3 & W^6 & W^9 & W^{12} \\ 1 & W^4 & W^8 & W^{12} & W^{16} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 & W^4 \\ 1 & W^2 & W^4 & W & W^3 \\ 1 & W^3 & W & W^4 & W^2 \\ 1 & W^4 & W^3 & W^2 & W \end{bmatrix}$$

Small DFT matrices: $N = 6$

$$\mathbf{W}_6 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 & W^4 & W^5 \\ 1 & W^2 & W^4 & W^6 & W^8 & W^{10} \\ 1 & W^3 & W^6 & W^9 & W^{12} & W^{15} \\ 1 & W^4 & W^8 & W^{12} & W^{16} & W^{20} \\ 1 & W^5 & W^{10} & W^{15} & W^{20} & W^{25} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 & W^4 & W^5 \\ 1 & W^2 & W^4 & 1 & W^2 & W^4 \\ 1 & W^3 & 1 & W^3 & 1 & W^3 \\ 1 & W^4 & W^2 & 1 & W^4 & W^2 \\ 1 & W^5 & W^4 & W^3 & W^2 & W \end{bmatrix}$$

COM303: Digital Signal Processing

Lecture 5: The Discrete Fourier Transform

Overview

- ▶ the Fourier basis for \mathbb{C}^N (recap)
- ▶ the DFT: definition and examples
- ▶ interpreting a DFT plot

Overview

- ▶ the Fourier basis for \mathbb{C}^N (recap)
- ▶ the DFT: definition and examples
- ▶ interpreting a DFT plot

Overview

- ▶ the Fourier basis for \mathbb{C}^N (recap)
- ▶ the DFT: definition and examples
- ▶ interpreting a DFT plot

The canonical (time) basis for \mathbb{C}^N

- ▶ in “signal” notation: $\delta_k[n] = \delta[n - k], \quad n, k = 0, 1, \dots, N - 1$
- ▶ in vector notation: $\{\delta^{(k)}\}_{k=0,1,\dots,N-1}$ with $\delta_n^{(k)} = \delta[n - k]$

The canonical (time) basis for \mathbb{C}^N

- ▶ in “signal” notation: $\delta_k[n] = \delta[n - k]$, $n, k = 0, 1, \dots, N - 1$
- ▶ in vector notation: $\{\delta^{(k)}\}_{k=0,1,\dots,N-1}$ with $\delta_n^{(k)} = \delta[n - k]$

The Fourier (frequency) basis for \mathbb{C}^N

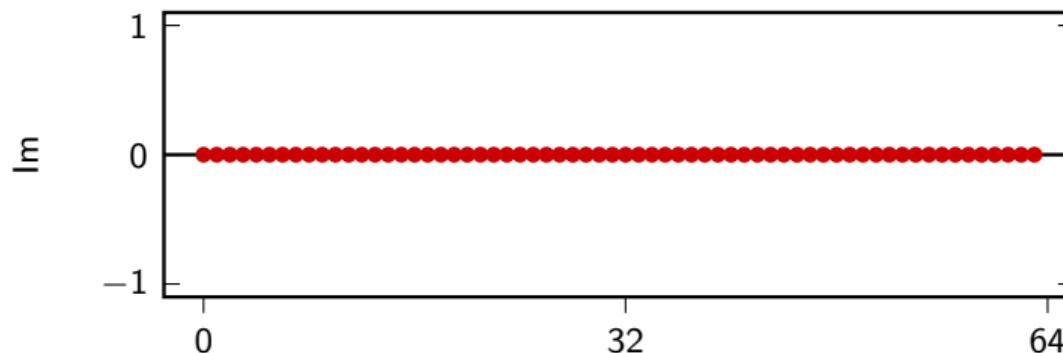
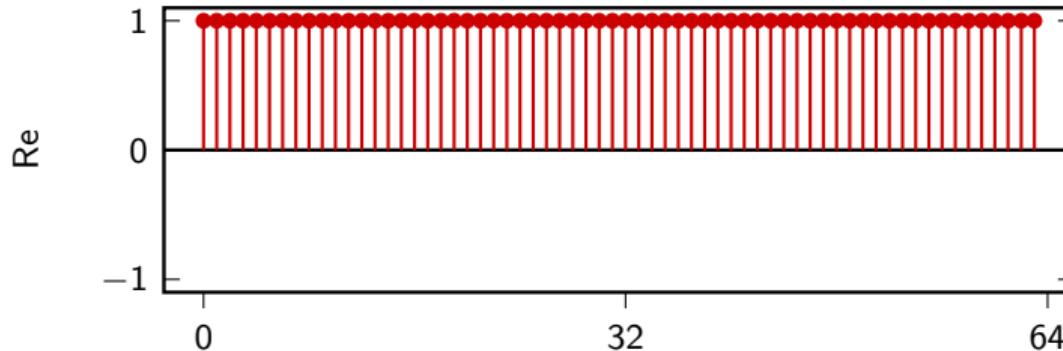
- ▶ in “signal” notation: $w_k[n] = e^{j\frac{2\pi}{N}nk}$, $n, k = 0, 1, \dots, N - 1$
- ▶ in vector notation: $\{\mathbf{w}^{(k)}\}_{k=0,1,\dots,N-1}$ with $w_n^{(k)} = e^{j\frac{2\pi}{N}nk}$

The Fourier (frequency) basis for \mathbb{C}^N

- ▶ in “signal” notation: $w_k[n] = e^{j\frac{2\pi}{N}nk}$, $n, k = 0, 1, \dots, N - 1$
- ▶ in vector notation: $\{\mathbf{w}^{(k)}\}_{k=0,1,\dots,N-1}$ with $w_n^{(k)} = e^{j\frac{2\pi}{N}nk}$

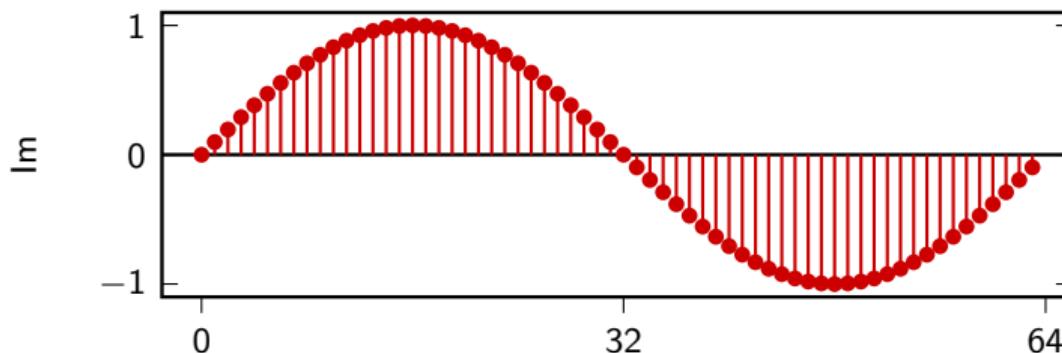
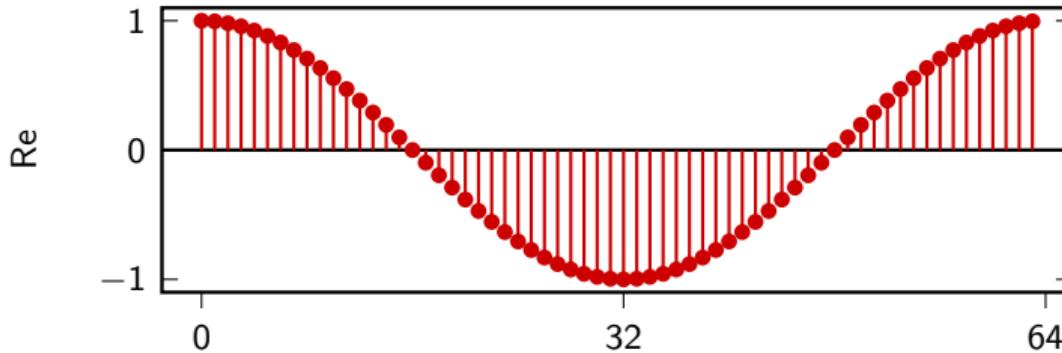
Basis vector $\mathbf{w}^{(0)} \in \mathbb{C}^{64}$

$$\omega = (2\pi/64) \cdot 0$$



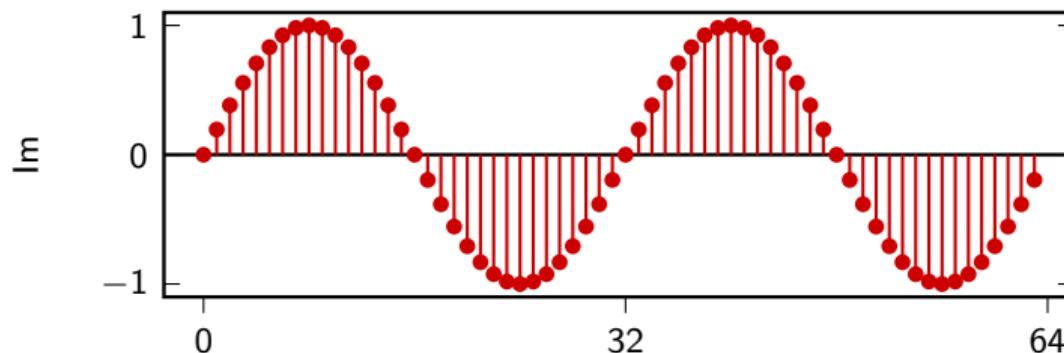
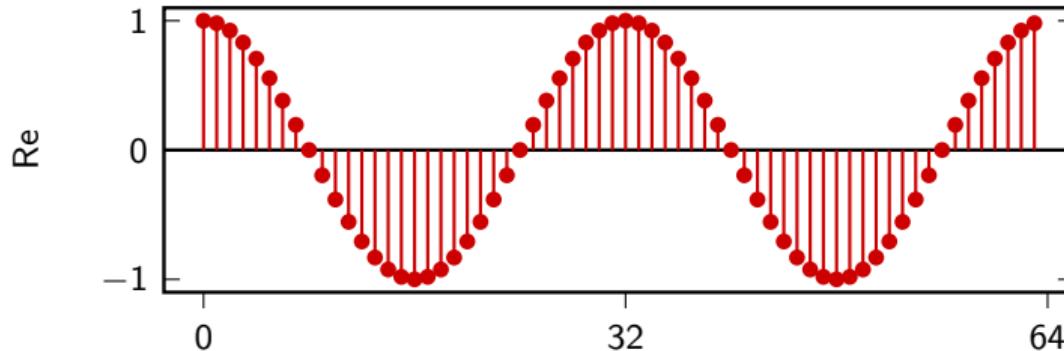
Basis vector $\mathbf{w}^{(1)} \in \mathbb{C}^{64}$

$$\omega = (2\pi/64) \cdot 1$$



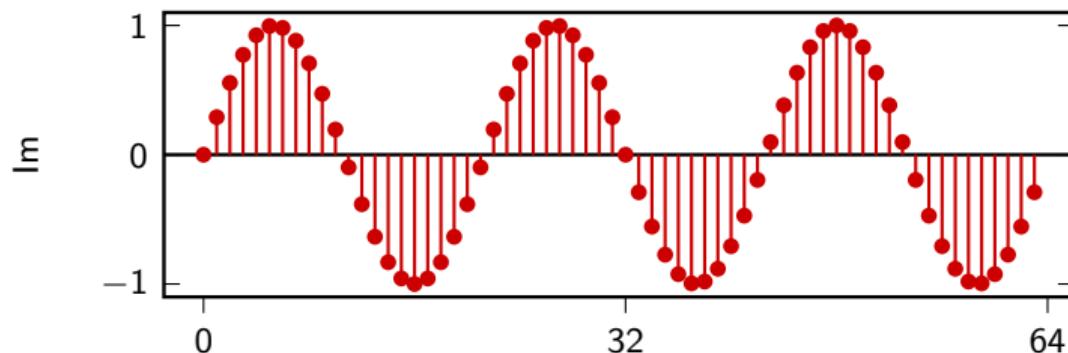
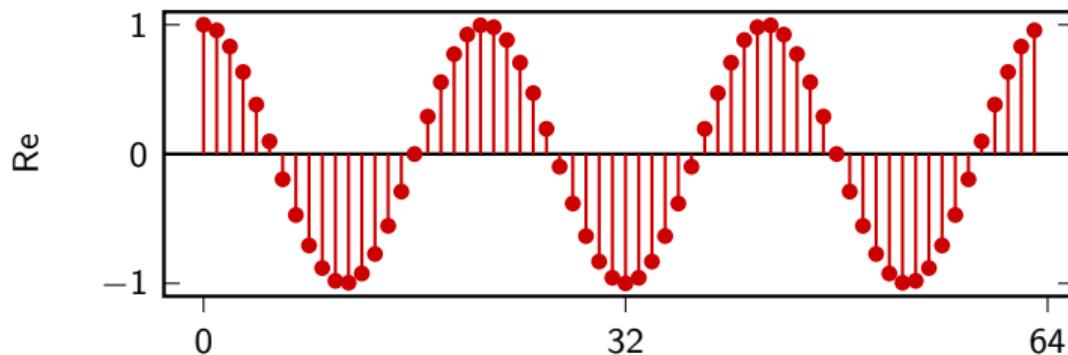
Basis vector $\mathbf{w}^{(2)} \in \mathbb{C}^{64}$

$$\omega = (2\pi/64) \cdot 2$$



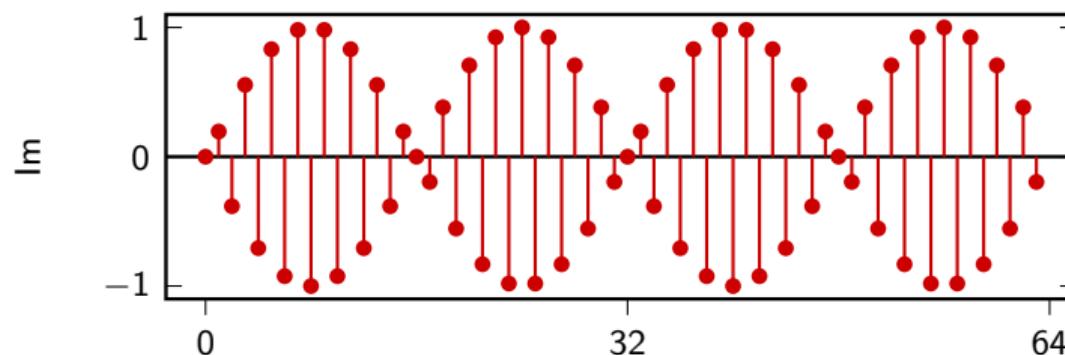
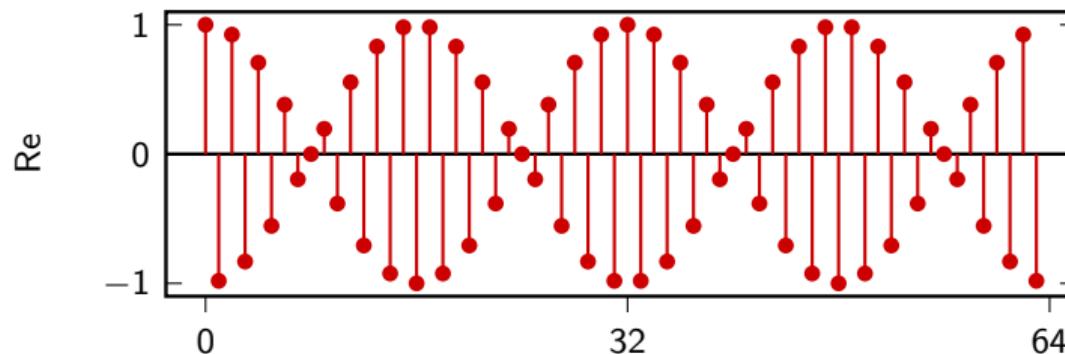
Basis vector $\mathbf{w}^{(3)} \in \mathbb{C}^{64}$

$$\omega = (2\pi/64) \cdot 3$$



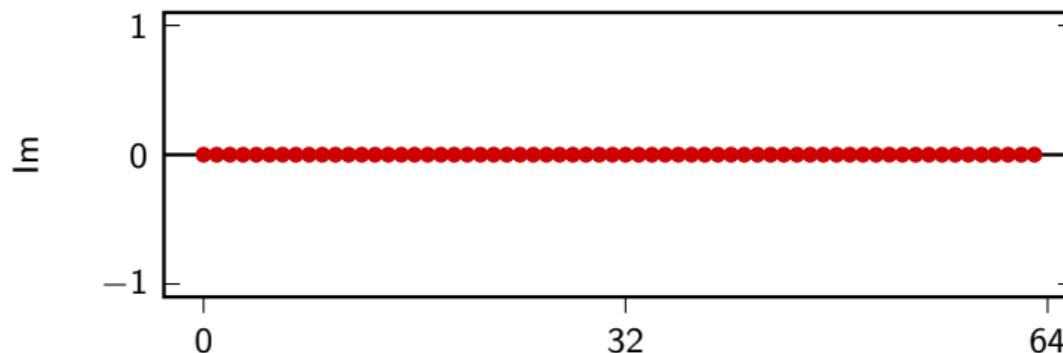
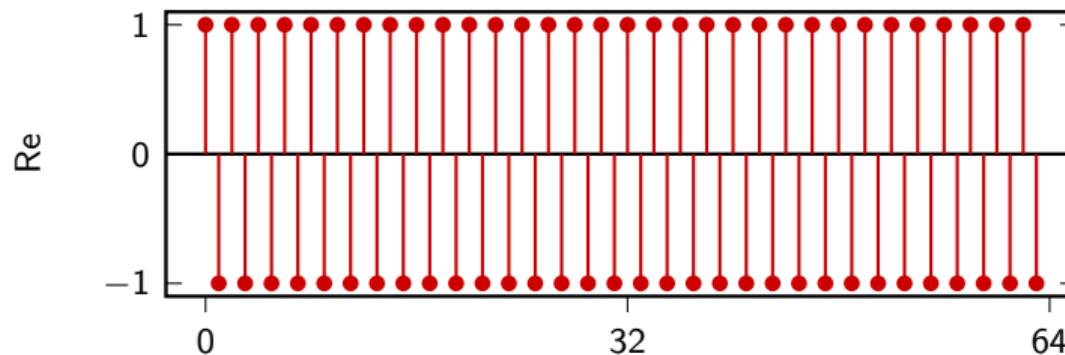
Basis vector $\mathbf{w}^{(30)} \in \mathbb{C}^{64}$

$$\omega = (2\pi/64) \cdot 30$$



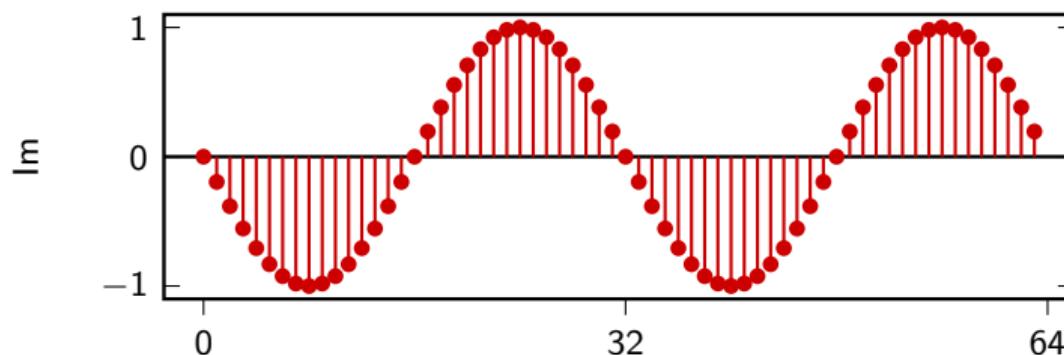
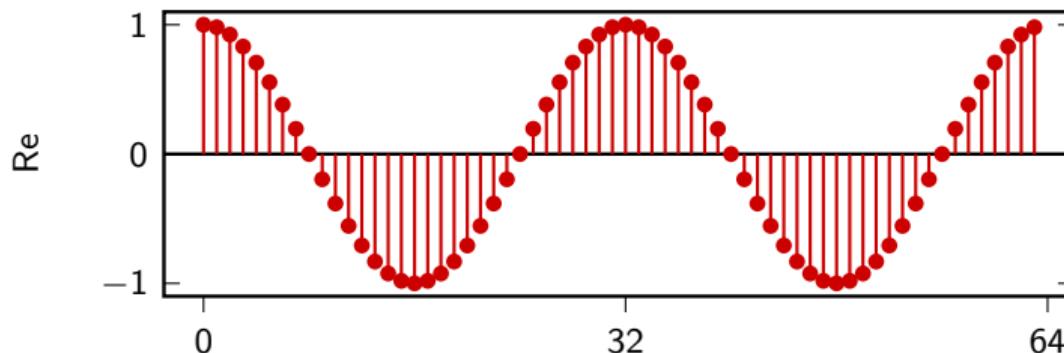
Basis vector $\mathbf{w}^{(32)} \in \mathbb{C}^{64}$

$$\omega = (2\pi/64) \cdot 32$$



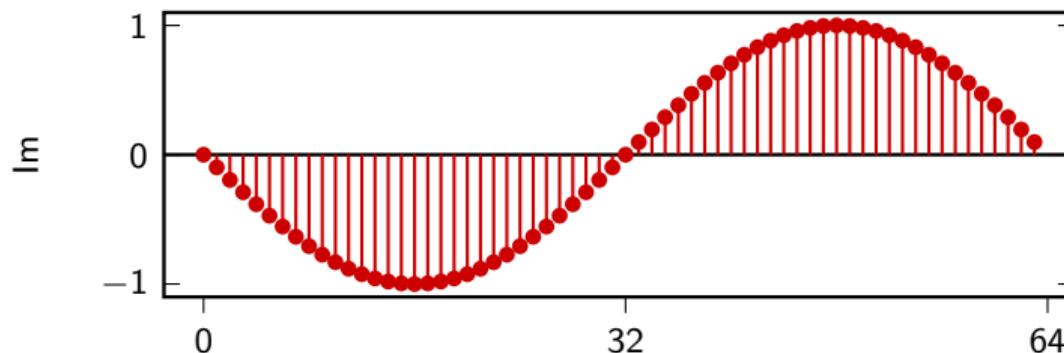
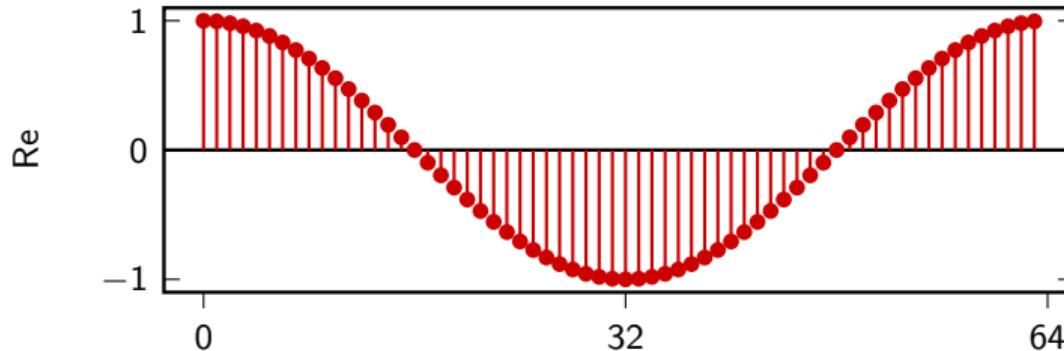
Basis vector $\mathbf{w}^{(62)} \in \mathbb{C}^{64}$

$$\omega = (2\pi/64) \cdot 62$$



Basis vector $\mathbf{w}^{(63)} \in \mathbb{C}^{64}$

$$\omega = (2\pi/64) \cdot 63$$



Proof of orthogonality

$$\begin{aligned}\langle \mathbf{w}^{(k)}, \mathbf{w}^{(h)} \rangle &= \sum_{n=0}^{N-1} (e^{j\frac{2\pi}{N}nk})^* e^{j\frac{2\pi}{N}nh} \\&= \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}(h-k)n} \\&= \begin{cases} N & \text{for } h = k \\ \frac{1 - e^{j2\pi(h-k)}}{1 - e^{j\frac{2\pi}{N}(h-k)}} = 0 & \text{otherwise} \end{cases}\end{aligned}$$

The Fourier Basis for \mathbb{C}^N

- ▶ N orthogonal vectors → basis for \mathbb{C}^N
- ▶ vectors are not orthonormal. Normalization factor would be $1/\sqrt{N}$
- ▶ will keep normalization factor explicit in DFT formulas

Basis expansion

Analysis formula:

$$X_k = \langle \mathbf{w}^{(k)}, \mathbf{x} \rangle$$

Synthesis formula:

$$\mathbf{x} = \frac{1}{N} \sum_{k=0}^{N-1} X_k \mathbf{w}^{(k)}$$

Basis expansion (signal notation)

Analysis formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}, \quad k = 0, 1, \dots, N-1$$

N-point signal in the *frequency domain*

Synthesis formula:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}nk}, \quad n = 0, 1, \dots, N-1$$

N-point signal in the “*time*” domain

Basis expansion (signal notation)

Analysis formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi}{N} nk}, \quad k = 0, 1, \dots, N-1$$

N-point signal in the *frequency domain*

Synthesis formula:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j \frac{2\pi}{N} nk}, \quad n = 0, 1, \dots, N-1$$

N-point signal in the “*time*” domain

Change of basis in matrix form

Define $W_N = e^{-j\frac{2\pi}{N}}$
(or simply W when N is evident from the context)

Change of basis matrix \mathbf{W} with $\mathbf{W}[n, m] = W_N^{nm}$:

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W^1 & W^2 & W^3 & \dots & W^{N-1} \\ 1 & W^2 & W^4 & W^6 & \dots & W^{2(N-1)} \\ & & & \ddots & & \\ 1 & W^{N-1} & W^{2(N-1)} & W^{3(N-1)} & \dots & W^{(N-1)^2} \end{bmatrix}$$

Change of basis in matrix form

Define $W_N = e^{-j\frac{2\pi}{N}}$
(or simply W when N is evident from the context)

Change of basis matrix \mathbf{W} with $\mathbf{W}[n, m] = W_N^{nm}$:

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W^1 & W^2 & W^3 & \dots & W^{N-1} \\ 1 & W^2 & W^4 & W^6 & \dots & W^{2(N-1)} \\ & & & \ddots & & \\ 1 & W^{N-1} & W^{2(N-1)} & W^{3(N-1)} & \dots & W^{(N-1)^2} \end{bmatrix}$$

Change of basis in matrix form

Analysis formula:

$$\mathbf{X} = \mathbf{W}\mathbf{x}$$

Synthesis formula:

$$\mathbf{x} = \frac{1}{N} \mathbf{W}^H \mathbf{X}$$

DFT Matrix

$$W_N^m = W_N^{(m \mod N)}$$

e.g. $W_8^{11} = W_8^3$

DFT Matrix

$$W_N^m = W_N^{(m \mod N)}$$

e.g. $W_8^{11} = W_8^3$

Small DFT matrices: $N = 2, 3$

$$W_2 = e^{-j\frac{2\pi}{2}} = -1$$

$$\mathbf{W}_2 = \begin{bmatrix} 1 & 1 \\ 1 & W \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$W_3 = e^{-j\frac{2\pi}{3}} = -(1 + j\sqrt{3})/2$$

$$\mathbf{W}_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & W & W^2 \\ 1 & W^2 & W^4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & W & W^2 \\ 1 & W^2 & W \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -(1 + j\sqrt{3})/2 & -(1 - j\sqrt{3})/2 \\ 1 & -(1 - j\sqrt{3})/2 & (1 - j\sqrt{3})/2 \end{bmatrix}$$

Small DFT matrices: $N = 4$

$$W_4 = e^{-j\frac{2\pi}{4}} = e^{-j\frac{\pi}{2}} = -j$$

$$\mathbf{W}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 \\ 1 & W^2 & W^4 & W^6 \\ 1 & W^3 & W^6 & W^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 \\ 1 & W^2 & 1 & W^2 \\ 1 & W^3 & W^2 & W \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

Small DFT matrices: $N = 5$

$$\mathbf{W}_5 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 & W^4 \\ 1 & W^2 & W^4 & W^6 & W^8 \\ 1 & W^3 & W^6 & W^9 & W^{12} \\ 1 & W^4 & W^8 & W^{12} & W^{16} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 & W^4 \\ 1 & W^2 & W^4 & W & W^3 \\ 1 & W^3 & W & W^4 & W^2 \\ 1 & W^4 & W^3 & W^2 & W \end{bmatrix}$$

Small DFT matrices: $N = 6$

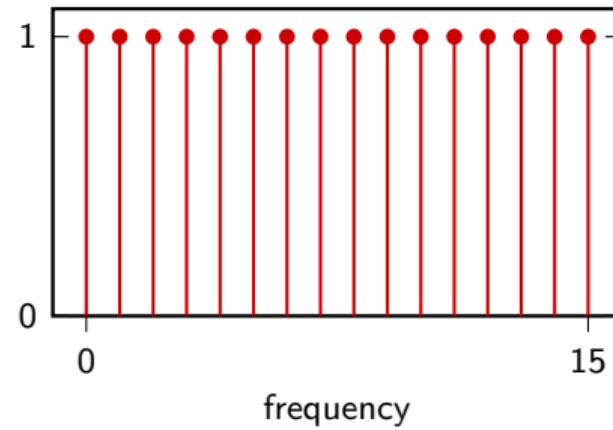
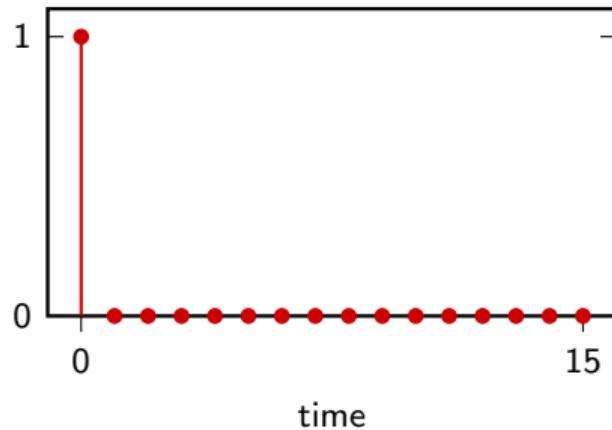
$$\mathbf{W}_6 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 & W^4 & W^5 \\ 1 & W^2 & W^4 & W^6 & W^8 & W^{10} \\ 1 & W^3 & W^6 & W^9 & W^{12} & W^{15} \\ 1 & W^4 & W^8 & W^{12} & W^{16} & W^{20} \\ 1 & W^5 & W^{10} & W^{15} & W^{20} & W^{25} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 & W^4 & W^5 \\ 1 & W^2 & W^4 & 1 & W^2 & W^4 \\ 1 & W^3 & 1 & W^3 & 1 & W^3 \\ 1 & W^4 & W^2 & 1 & W^4 & W^2 \\ 1 & W^5 & W^4 & W^3 & W^2 & W \end{bmatrix}$$

DFT is obviously linear

$$\text{DFT}\{\alpha x[n] + \beta y[n]\} = \alpha \text{DFT}\{x[n]\} + \beta \text{DFT}\{y[n]\}$$

DFT of $\delta[n] \in \mathbb{C}^N$

$$\sum_{n=0}^{N-1} \delta[n] e^{-j\frac{2\pi}{N}nk} = 1 \quad \forall k$$



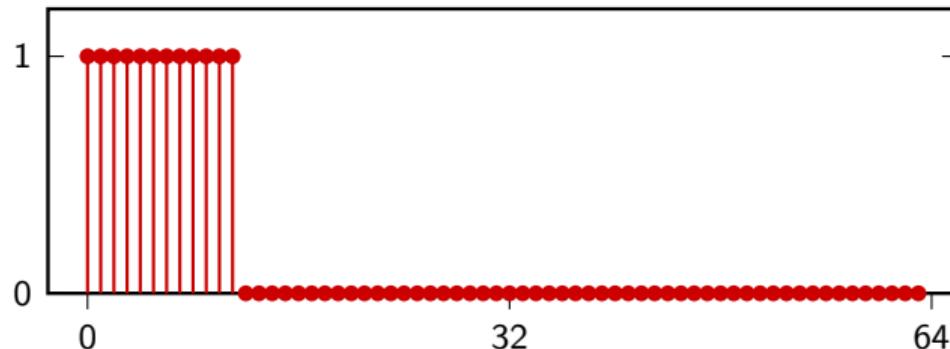
DFT of $\delta^{(m)} = \delta[n - m] \in \mathbb{C}^N$

$$\sum_{n=0}^{N-1} \delta[n - m] e^{-j \frac{2\pi}{N} nk} = e^{-j \frac{2\pi}{N} mk}$$

$$\text{DFT} \left\{ \delta^{(m)} \right\} = (\mathbf{w}^{(m)})^*$$

DFT of length- M step in \mathbb{C}^N

$$x[n] = \sum_{m=0}^{M-1} \delta[n - m], \quad n = 0, 1, \dots, N - 1$$



DFT of length- M step in \mathbb{C}^N

$$\begin{aligned} X[k] &= \sum_{m=0}^{M-1} \text{DFT} \{ \delta[n-m] \} = \sum_{m=0}^{M-1} e^{-j\frac{2\pi}{N}mk} \\ &= \frac{1 - e^{-j\frac{2\pi}{N}kM}}{1 - e^{-j\frac{2\pi}{N}k}} \\ &= \frac{e^{-j\frac{\pi}{N}kM} \left[e^{j\frac{\pi}{N}kM} - e^{-j\frac{\pi}{N}kM} \right]}{e^{-j\frac{\pi}{N}k} \left[e^{j\frac{\pi}{N}k} - e^{-j\frac{\pi}{N}k} \right]} \\ &= \frac{\sin\left(\frac{\pi}{N}Mk\right)}{\sin\left(\frac{\pi}{N}k\right)} e^{-j\frac{\pi}{N}(M-1)k} \end{aligned}$$

DFT of length- M step in \mathbb{C}^N

$$\begin{aligned} X[k] &= \sum_{m=0}^{M-1} \text{DFT} \{ \delta[n-m] \} = \sum_{m=0}^{M-1} e^{-j\frac{2\pi}{N}mk} \\ &= \frac{1 - e^{-j\frac{2\pi}{N}kM}}{1 - e^{-j\frac{2\pi}{N}k}} \\ &= \frac{e^{-j\frac{\pi}{N}kM} \left[e^{j\frac{\pi}{N}kM} - e^{-j\frac{\pi}{N}kM} \right]}{e^{-j\frac{\pi}{N}k} \left[e^{j\frac{\pi}{N}k} - e^{-j\frac{\pi}{N}k} \right]} \\ &= \frac{\sin\left(\frac{\pi}{N}Mk\right)}{\sin\left(\frac{\pi}{N}k\right)} e^{-j\frac{\pi}{N}(M-1)k} \end{aligned}$$

DFT of length- M step in \mathbb{C}^N

$$\begin{aligned} X[k] &= \sum_{m=0}^{M-1} \text{DFT} \{ \delta[n-m] \} = \sum_{m=0}^{M-1} e^{-j\frac{2\pi}{N}mk} \\ &= \frac{1 - e^{-j\frac{2\pi}{N}kM}}{1 - e^{-j\frac{2\pi}{N}k}} \\ &= \frac{e^{-j\frac{\pi}{N}kM} \left[e^{j\frac{\pi}{N}kM} - e^{-j\frac{\pi}{N}kM} \right]}{e^{-j\frac{\pi}{N}k} \left[e^{j\frac{\pi}{N}k} - e^{-j\frac{\pi}{N}k} \right]} \\ &= \frac{\sin\left(\frac{\pi}{N}Mk\right)}{\sin\left(\frac{\pi}{N}k\right)} e^{-j\frac{\pi}{N}(M-1)k} \end{aligned}$$

DFT of length- M step in \mathbb{C}^N

$$\begin{aligned} X[k] &= \sum_{m=0}^{M-1} \text{DFT} \{ \delta[n-m] \} = \sum_{m=0}^{M-1} e^{-j\frac{2\pi}{N}mk} \\ &= \frac{1 - e^{-j\frac{2\pi}{N}kM}}{1 - e^{-j\frac{2\pi}{N}k}} \\ &= \frac{e^{-j\frac{\pi}{N}kM} \left[e^{j\frac{\pi}{N}kM} - e^{-j\frac{\pi}{N}kM} \right]}{e^{-j\frac{\pi}{N}k} \left[e^{j\frac{\pi}{N}k} - e^{-j\frac{\pi}{N}k} \right]} \\ &= \frac{\sin\left(\frac{\pi}{N}Mk\right)}{\sin\left(\frac{\pi}{N}k\right)} e^{-j\frac{\pi}{N}(M-1)k} \end{aligned}$$

DFT of length- M step in \mathbb{C}^N

$$X[k] = \frac{\sin\left(\frac{\pi}{N}Mk\right)}{\sin\left(\frac{\pi}{N}k\right)} e^{-j\frac{\pi}{N}(M-1)k}$$

- ▶ $X[0] = M$, from the definition of the sum
- ▶ $X[k] = 0$ if Mk/N integer ($0 \leq k < N$)
- ▶ $\angle X[k]$ linear in k (except at sign changes for the real part)

DFT of length- M step in \mathbb{C}^N

$$X[k] = \frac{\sin\left(\frac{\pi}{N}Mk\right)}{\sin\left(\frac{\pi}{N}k\right)} e^{-j\frac{\pi}{N}(M-1)k}$$

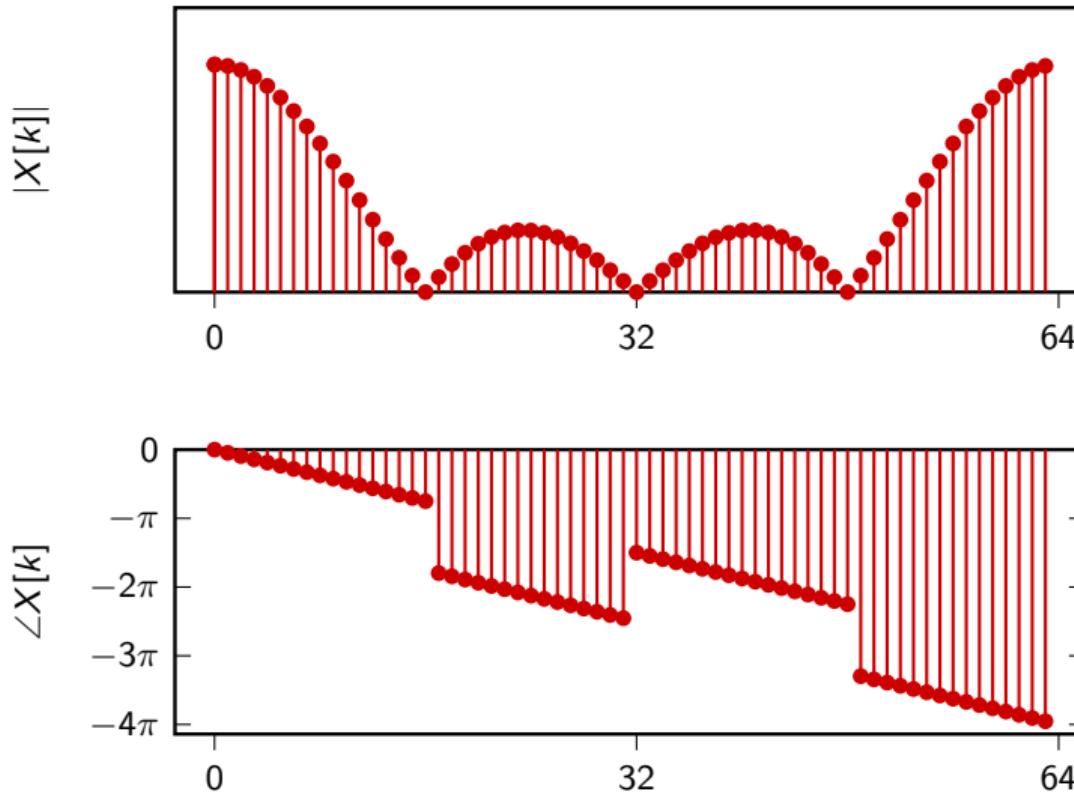
- ▶ $X[0] = M$, from the definition of the sum
- ▶ $X[k] = 0$ if Mk/N integer ($0 \leq k < N$)
- ▶ $\angle X[k]$ linear in k (except at sign changes for the real part)

DFT of length- M step in \mathbb{C}^N

$$X[k] = \frac{\sin\left(\frac{\pi}{N}Mk\right)}{\sin\left(\frac{\pi}{N}k\right)} e^{-j\frac{\pi}{N}(M-1)k}$$

- ▶ $X[0] = M$, from the definition of the sum
- ▶ $X[k] = 0$ if Mk/N integer ($0 \leq k < N$)
- ▶ $\angle X[k]$ linear in k (except at sign changes for the real part)

DFT of length-4 step in \mathbb{C}^{64}

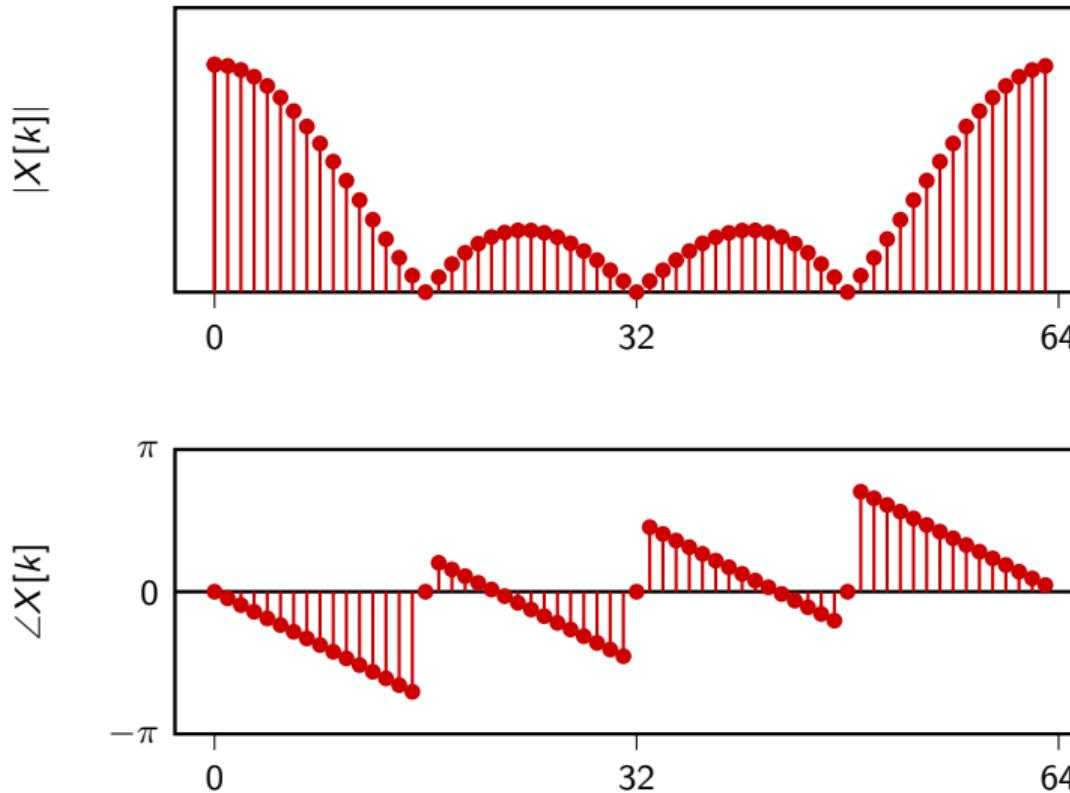


Wrapping the phase

Often the phase is displayed "wrapped" over the $[-\pi, \pi]$ interval.

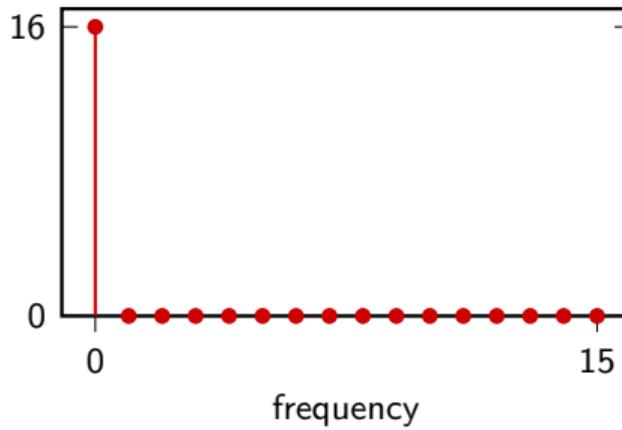
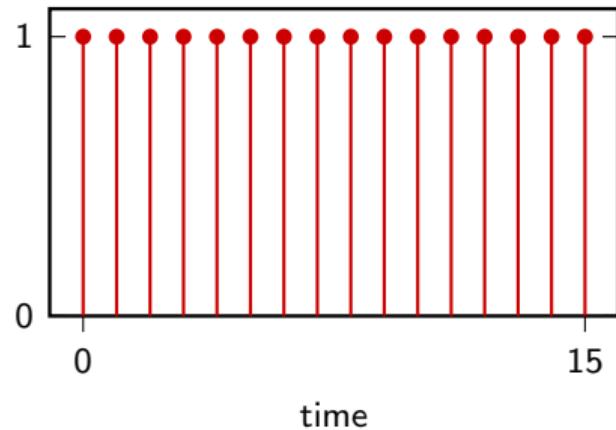
- ▶ most numerical packages return wrapped phase
- ▶ phase can be unwrapped by adding multiples of 2π

DFT of length-4 step in \mathbb{C}^{64} (phase wrapped)



DFT of $x[n] = 1$, $x[n] \in \mathbb{C}^N$

$$X[k] = \sum_{n=0}^{N-1} e^{-j\frac{2\pi}{N}nk} = N\delta[k]$$



DFT of harmonic oscillations

$$\text{DFT} \left\{ e^{j \frac{2\pi}{N} mn} \right\} = \langle \mathbf{w}^{(k)}, \mathbf{w}^{(m)} \rangle = N\delta[m - k]$$

DFT of $x[n] = 3 \cos(2\pi/16 n)$, $x[n] \in \mathbb{C}^{64}$

$$\begin{aligned}x[n] &= 3 \cos\left(\frac{2\pi}{16} n\right) \\&= 3 \cos\left(\frac{2\pi}{64} 4 n\right) \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64} 4n} + e^{-j\frac{2\pi}{64} 4n} \right] \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64} 4n} + e^{j\frac{2\pi}{64} 60n} \right] \\&= \frac{3}{2}(w_4[n] + w_{60}[n])\end{aligned}$$

DFT of $x[n] = 3 \cos(2\pi/16 n)$, $x[n] \in \mathbb{C}^{64}$

$$\begin{aligned}x[n] &= 3 \cos\left(\frac{2\pi}{16} n\right) \\&= 3 \cos\left(\frac{2\pi}{64} 4n\right) \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64} 4n} + e^{-j\frac{2\pi}{64} 4n} \right] \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64} 4n} + e^{j\frac{2\pi}{64} 60n} \right] \\&= \frac{3}{2}(w_4[n] + w_{60}[n])\end{aligned}$$

DFT of $x[n] = 3 \cos(2\pi/16 n)$, $x[n] \in \mathbb{C}^{64}$

$$\begin{aligned}x[n] &= 3 \cos\left(\frac{2\pi}{16} n\right) \\&= 3 \cos\left(\frac{2\pi}{64} 4 n\right) \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64}4n} + e^{-j\frac{2\pi}{64}4n} \right] \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64}4n} + e^{j\frac{2\pi}{64}60n} \right] \\&= \frac{3}{2}(w_4[n] + w_{60}[n])\end{aligned}$$

DFT of $x[n] = 3 \cos(2\pi/16 n)$, $x[n] \in \mathbb{C}^{64}$

$$\begin{aligned}x[n] &= 3 \cos\left(\frac{2\pi}{16} n\right) \\&= 3 \cos\left(\frac{2\pi}{64} 4 n\right) \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64}4n} + e^{-j\frac{2\pi}{64}4n} \right] \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64}4n} + e^{j\frac{2\pi}{64}60n} \right] \\&= \frac{3}{2}(w_4[n] + w_{60}[n])\end{aligned}$$

DFT of $x[n] = 3 \cos(2\pi/16 n)$, $x[n] \in \mathbb{C}^{64}$

$$\begin{aligned}x[n] &= 3 \cos\left(\frac{2\pi}{16} n\right) \\&= 3 \cos\left(\frac{2\pi}{64} 4 n\right) \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64}4n} + e^{-j\frac{2\pi}{64}4n} \right] \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64}4n} + e^{j\frac{2\pi}{64}60n} \right] \\&= \frac{3}{2}(w_4[n] + w_{60}[n])\end{aligned}$$

DFT of $x[n] = 3 \cos(2\pi/16 n)$, $x[n] \in \mathbb{C}^{64}$

$$\begin{aligned} X[k] &= \langle w_k[n], x[n] \rangle \\ &= \langle w_k[n], \frac{3}{2}(w_4[n] + w_{60}[n]) \rangle \\ &= \frac{3}{2} \langle w_k[n], w_4[n] \rangle + \frac{3}{2} \langle w_k[n], w_{60}[n] \rangle \\ &= \begin{cases} 96 & \text{for } k = 4, 60 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

DFT of $x[n] = 3 \cos(2\pi/16 n)$, $x[n] \in \mathbb{C}^{64}$

$$\begin{aligned} X[k] &= \langle w_k[n], x[n] \rangle \\ &= \langle w_k[n], \frac{3}{2}(w_4[n] + w_{60}[n]) \rangle \\ &= \frac{3}{2} \langle w_k[n], w_4[n] \rangle + \frac{3}{2} \langle w_k[n], w_{60}[n] \rangle \\ &= \begin{cases} 96 & \text{for } k = 4, 60 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

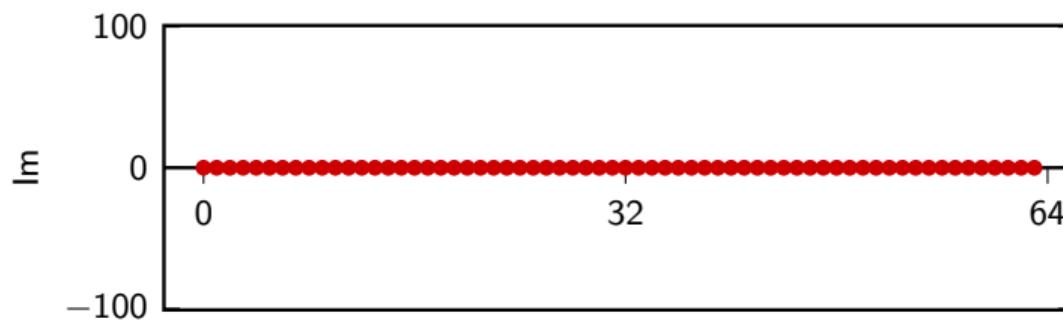
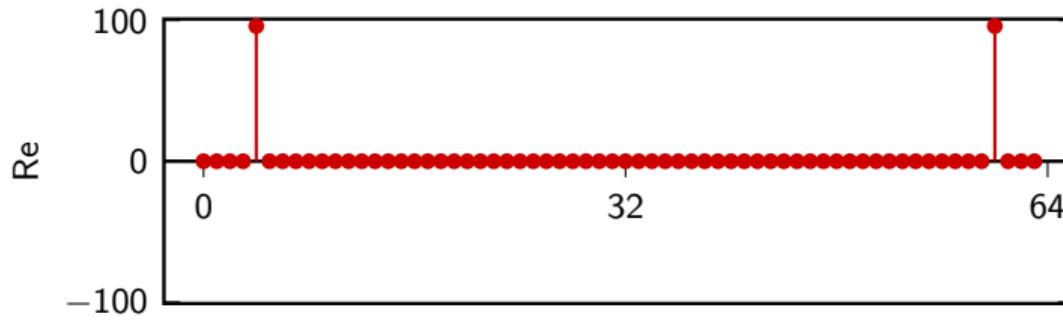
DFT of $x[n] = 3 \cos(2\pi/16 n)$, $x[n] \in \mathbb{C}^{64}$

$$\begin{aligned} X[k] &= \langle w_k[n], x[n] \rangle \\ &= \langle w_k[n], \frac{3}{2}(w_4[n] + w_{60}[n]) \rangle \\ &= \frac{3}{2} \langle w_k[n], w_4[n] \rangle + \frac{3}{2} \langle w_k[n], w_{60}[n] \rangle \\ &= \begin{cases} 96 & \text{for } k = 4, 60 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

DFT of $x[n] = 3 \cos(2\pi/16 n)$, $x[n] \in \mathbb{C}^{64}$

$$\begin{aligned} X[k] &= \langle w_k[n], x[n] \rangle \\ &= \langle w_k[n], \frac{3}{2}(w_4[n] + w_{60}[n]) \rangle \\ &= \frac{3}{2} \langle w_k[n], w_4[n] \rangle + \frac{3}{2} \langle w_k[n], w_{60}[n] \rangle \\ &= \begin{cases} 96 & \text{for } k = 4, 60 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

DFT of $x[n] = 3 \cos(2\pi/16 n)$, $x[n] \in \mathbb{C}^{64}$



DFT of $x[n] = 3 \cos(2\pi/16 n + \pi/3)$, $x[n] \in \mathbb{C}^{64}$

$$\begin{aligned}x[n] &= 3 \cos\left(\frac{2\pi}{16} n + \frac{\pi}{3}\right) \\&= 3 \cos\left(\frac{2\pi}{64} 4n + \frac{\pi}{3}\right) \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64}4n} e^{j\frac{\pi}{3}} + e^{-j\frac{2\pi}{64}4n} e^{-j\frac{\pi}{3}} \right] \\&= \frac{3}{2} (e^{j\frac{\pi}{3}} w_4[n] + e^{-j\frac{\pi}{3}} w_{60}[n])\end{aligned}$$

DFT of $x[n] = 3 \cos(2\pi/16 n + \pi/3)$, $x[n] \in \mathbb{C}^{64}$

$$\begin{aligned}x[n] &= 3 \cos\left(\frac{2\pi}{16} n + \frac{\pi}{3}\right) \\&= 3 \cos\left(\frac{2\pi}{64} 4n + \frac{\pi}{3}\right) \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64}4n} e^{j\frac{\pi}{3}} + e^{-j\frac{2\pi}{64}4n} e^{-j\frac{\pi}{3}} \right] \\&= \frac{3}{2} (e^{j\frac{\pi}{3}} w_4[n] + e^{-j\frac{\pi}{3}} w_{60}[n])\end{aligned}$$

DFT of $x[n] = 3 \cos(2\pi/16 n + \pi/3)$, $x[n] \in \mathbb{C}^{64}$

$$\begin{aligned}x[n] &= 3 \cos\left(\frac{2\pi}{16} n + \frac{\pi}{3}\right) \\&= 3 \cos\left(\frac{2\pi}{64} 4n + \frac{\pi}{3}\right) \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64}4n} e^{j\frac{\pi}{3}} + e^{-j\frac{2\pi}{64}4n} e^{-j\frac{\pi}{3}} \right] \\&= \frac{3}{2} (e^{j\frac{\pi}{3}} w_4[n] + e^{-j\frac{\pi}{3}} w_{60}[n])\end{aligned}$$

DFT of $x[n] = 3 \cos(2\pi/16 n + \pi/3)$, $x[n] \in \mathbb{C}^{64}$

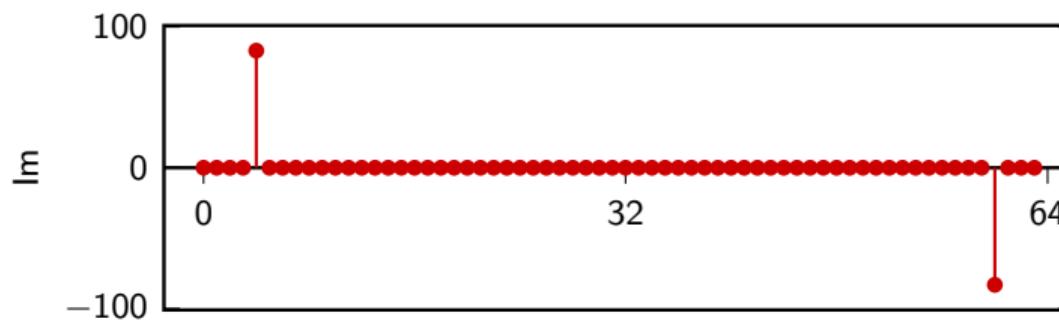
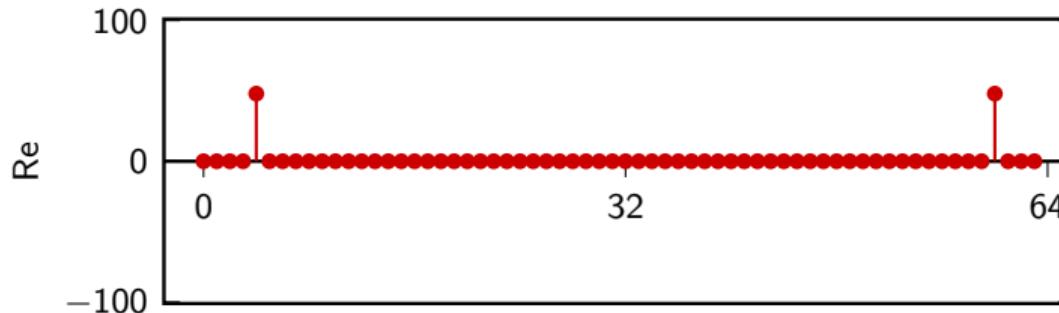
$$\begin{aligned}x[n] &= 3 \cos\left(\frac{2\pi}{16} n + \frac{\pi}{3}\right) \\&= 3 \cos\left(\frac{2\pi}{64} 4n + \frac{\pi}{3}\right) \\&= \frac{3}{2} \left[e^{j\frac{2\pi}{64}4n} e^{j\frac{\pi}{3}} + e^{-j\frac{2\pi}{64}4n} e^{-j\frac{\pi}{3}} \right] \\&= \frac{3}{2} (e^{j\frac{\pi}{3}} w_4[n] + e^{-j\frac{\pi}{3}} w_{60}[n])\end{aligned}$$

DFT of $x[n] = 3 \cos(2\pi/16 n + \pi/3)$, $x[n] \in \mathbb{C}^{64}$

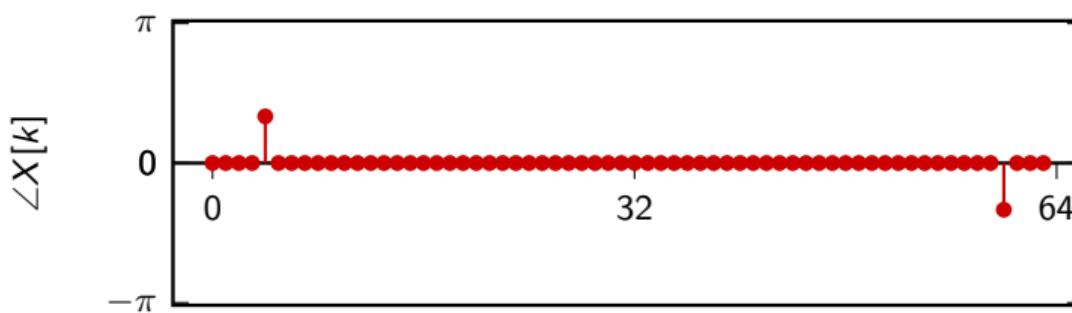
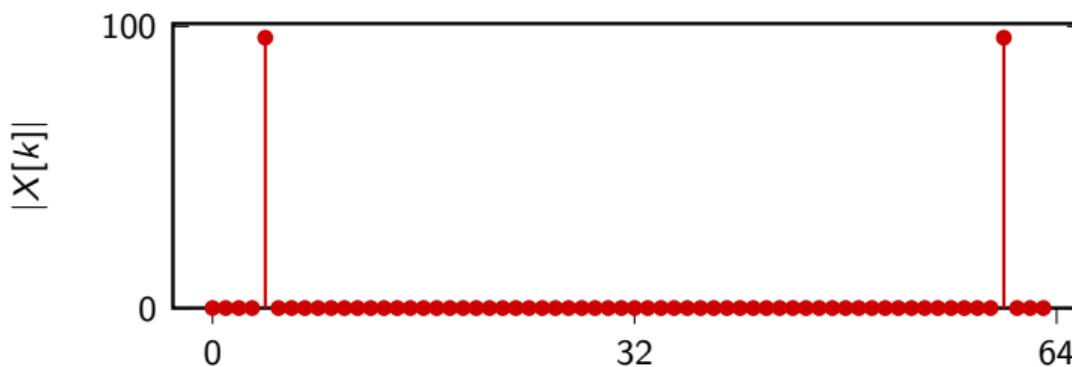
$$X[k] = \langle w_k[n], x[n] \rangle$$

$$= \begin{cases} 96e^{j\frac{\pi}{3}} & \text{for } k = 4 \\ 96e^{-j\frac{\pi}{3}} & \text{for } k = 60 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{DFT of } x[n] = 3 \cos(2\pi/16 n + \pi/3), \quad x[n] \in \mathbb{C}^{64}$$



DFT of $x[n] = 3 \cos(2\pi/16 n + \pi/3)$, $x[n] \in \mathbb{C}^{64}$



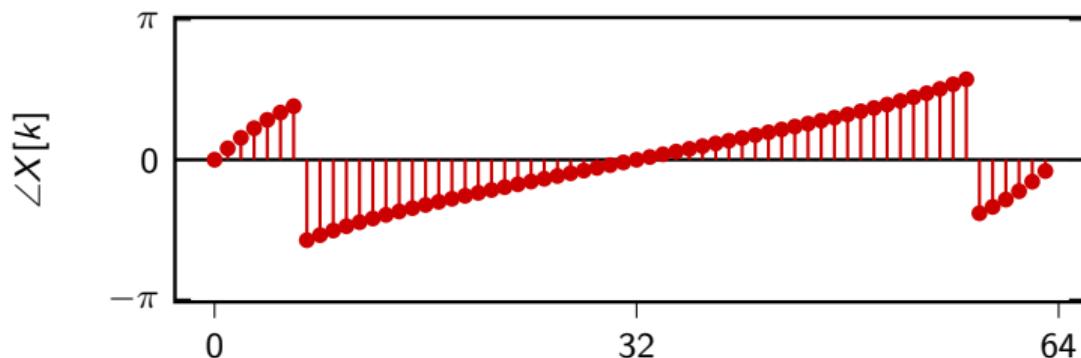
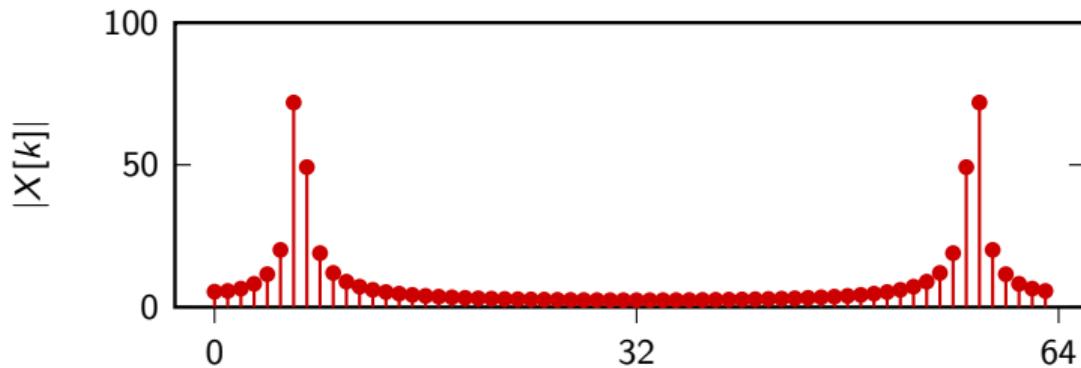
DFT of $x[n] = 3 \cos(2\pi/10 n)$, $x[n] \in \mathbb{C}^{64}$

$$\frac{2\pi}{64} 6 < \frac{2\pi}{10} < \frac{2\pi}{64} 7$$

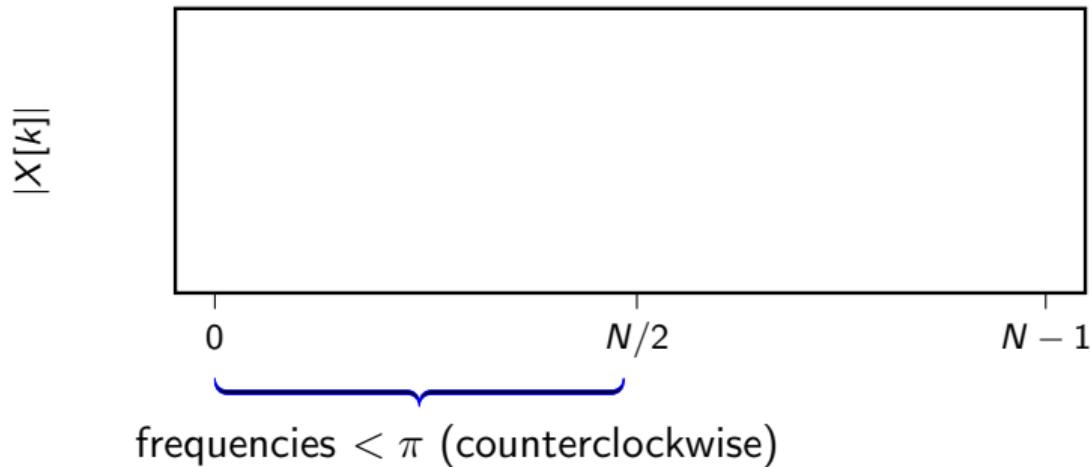
The DFT is an algorithm!

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi}{N} nk}, \quad k = 0, 1, \dots, N-1$$

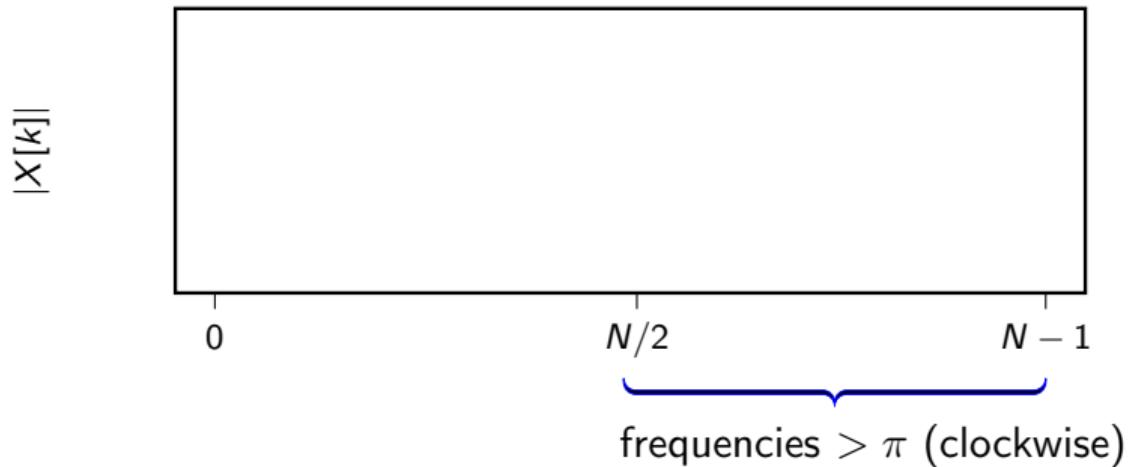
DFT of $x[n] = 3 \cos(2\pi/10 n)$, $x[n] \in \mathbb{C}^{64}$



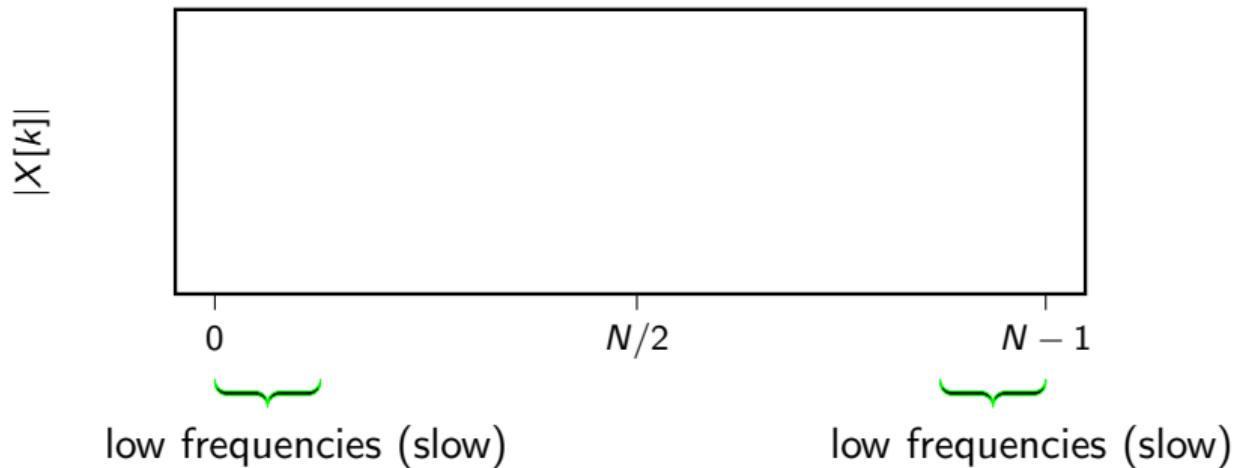
Interpreting a DFT plot



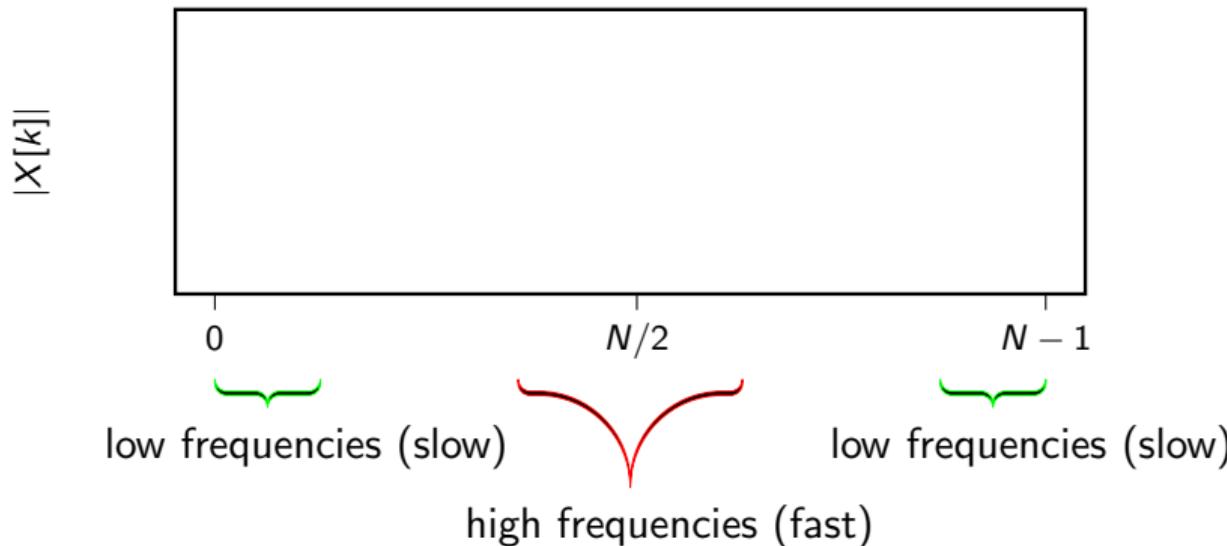
Interpreting a DFT plot



Interpreting a DFT plot

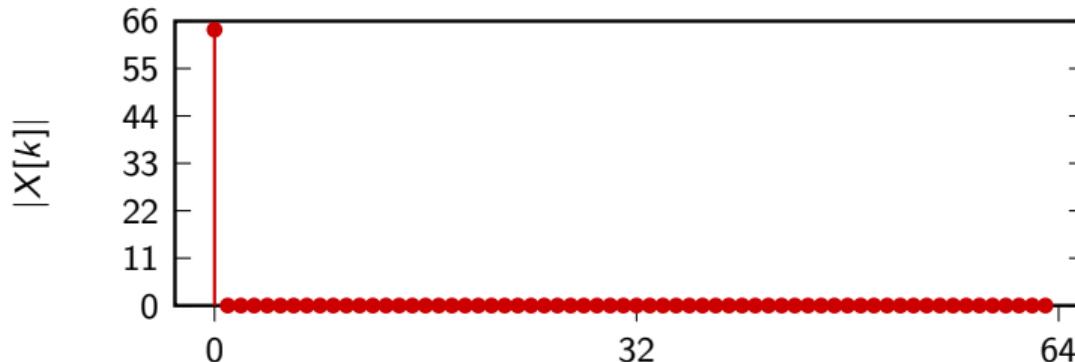


Interpreting a DFT plot



Interpreting a DFT plot

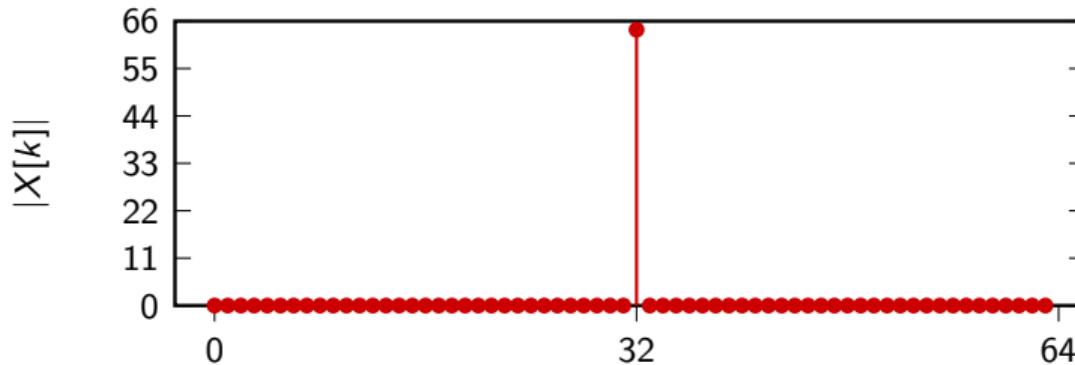
$x[n] = 1$ (slowest signal)



only lowest frequency

Interpreting a DFT plot

$$x[n] = \cos \pi n = (-1)^n \text{ (fastest signal)}$$



only highest frequency

Energy distribution

Recall Parseval's Theorem: $\|\mathbf{x}\|^2 = \sum |\alpha_k|^2$

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2$$

square magnitude of k -th DFT coefficient
proportional to signal's energy at frequency $\omega = (2\pi/N)k$

Energy distribution

Recall Parseval's Theorem: $\|\mathbf{x}\|^2 = \sum |\alpha_k|^2$

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2$$

square magnitude of k -th DFT coefficient
proportional to signal's energy at frequency $\omega = (2\pi/N)k$

Energy distribution

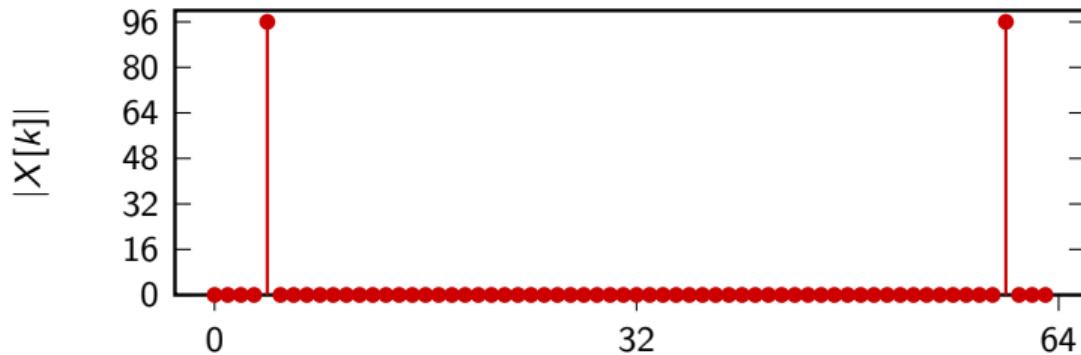
Recall Parseval's Theorem: $\|\mathbf{x}\|^2 = \sum |\alpha_k|^2$

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2$$

square magnitude of k -th DFT coefficient
proportional to signal's energy at frequency $\omega = (2\pi/N)k$

Interpreting a DFT plot

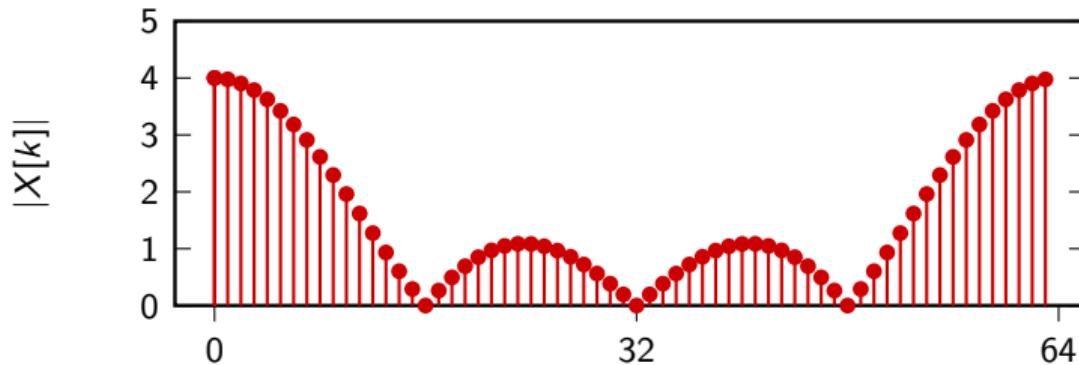
$$x[n] = 3 \cos(2\pi/16 n) \text{ (sinusoid)}$$



energy concentrated on single frequency
(counterclockwise and clockwise combine to give real signal)

Interpreting a DFT plot

$$x[n] = u[n] - u[n - 4] \text{ (step)}$$

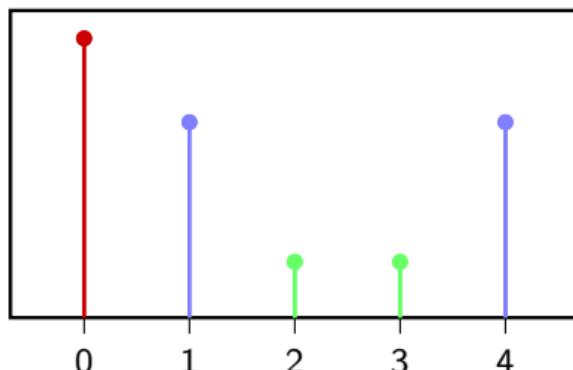


energy mostly in low frequencies

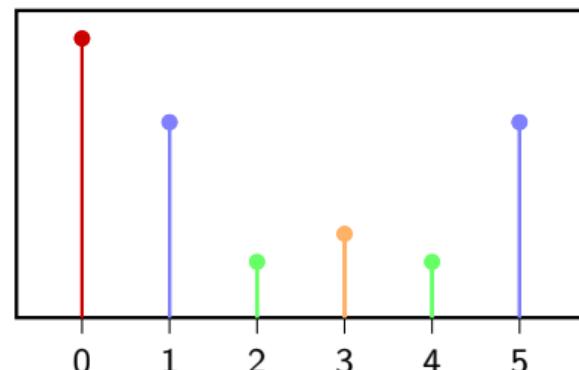
DFT of real signals

For real signals the DFT is “symmetric” in magnitude:

$$|X[k]| = |X[N - k]| \text{ for } k = 1, 2, \dots, [N/2]$$



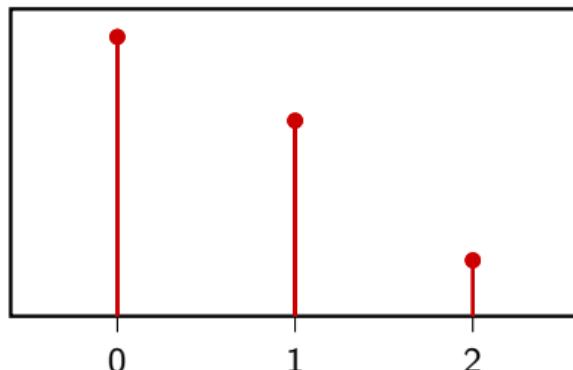
$N = 5$, odd length



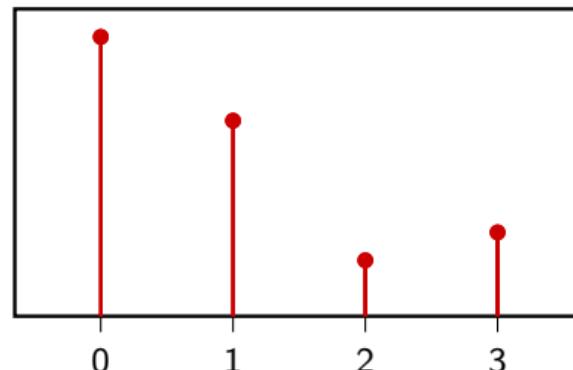
$N = 6$, even length

DFT of real signals

For real signals, magnitude plots need only $\lfloor N/2 \rfloor + 1$ points



$N = 5$, odd length



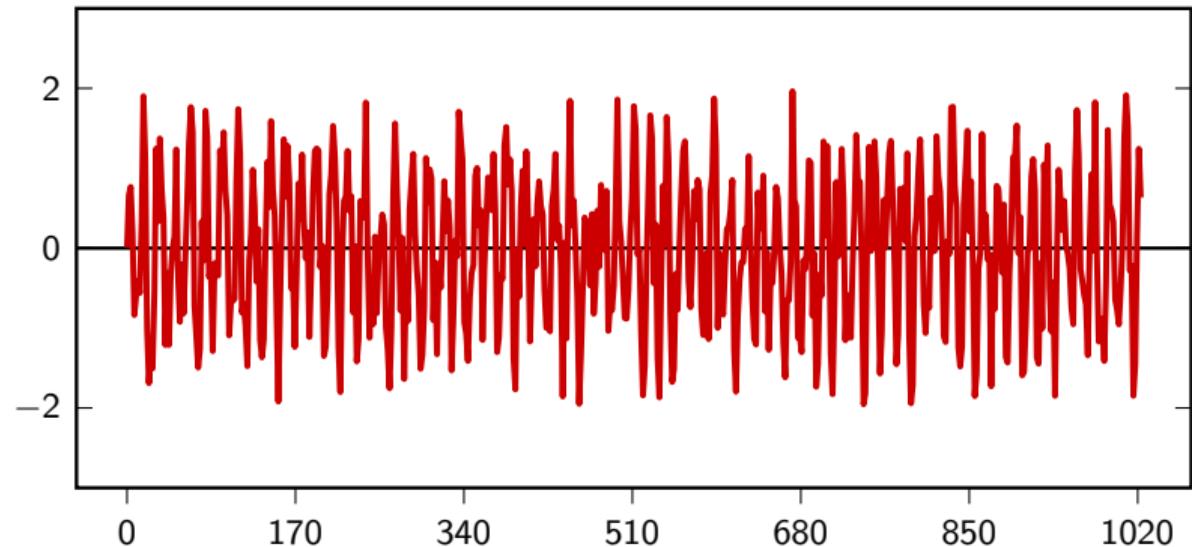
$N = 6$, even length

the DFT as an analysis tool

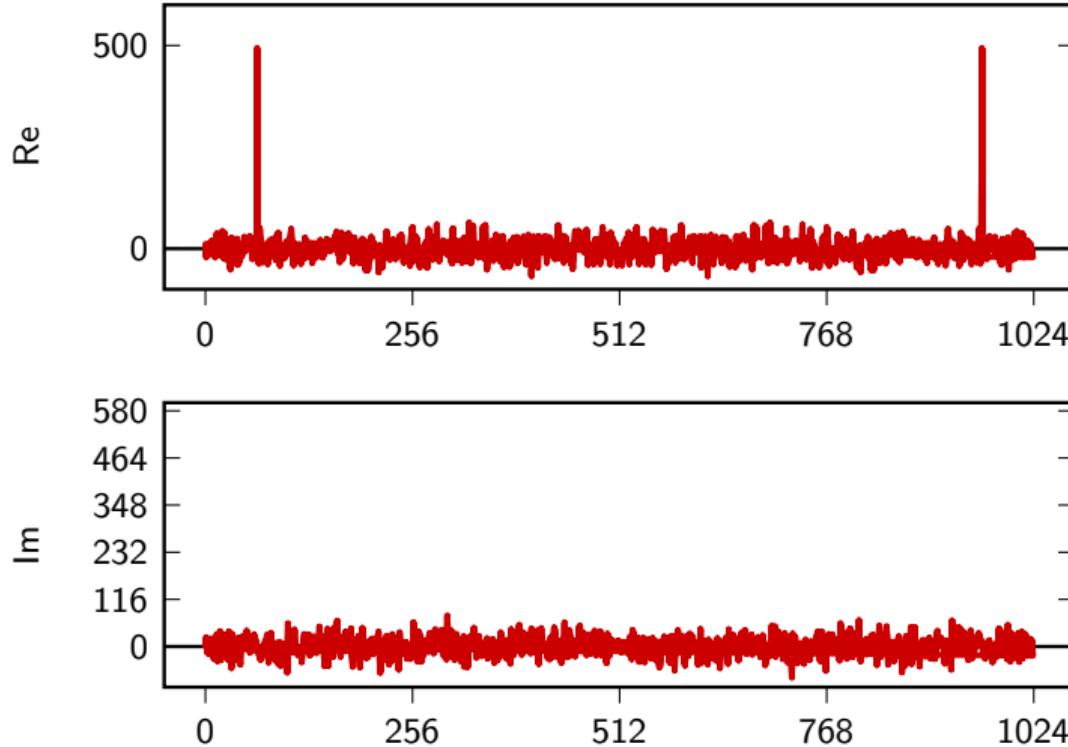
Overview

- ▶ DFT analysis examples
- ▶ Labeling the DFT axes

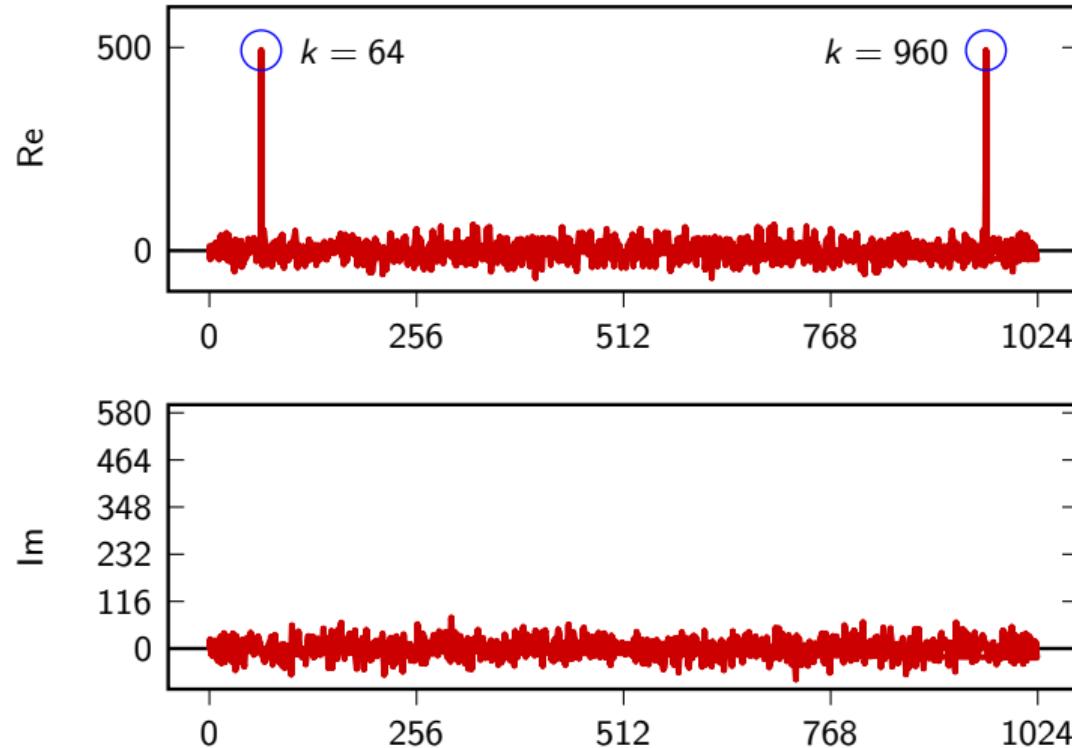
Mystery signal revisited



Mystery signal revisited



Mystery signal revisited



Mystery signal revisited

$$x[n] = \cos(\omega n + \phi) + \eta[n]$$

with

$$\begin{aligned}\phi &= 0 \\ \omega &= \frac{2\pi}{1024} 64\end{aligned}$$

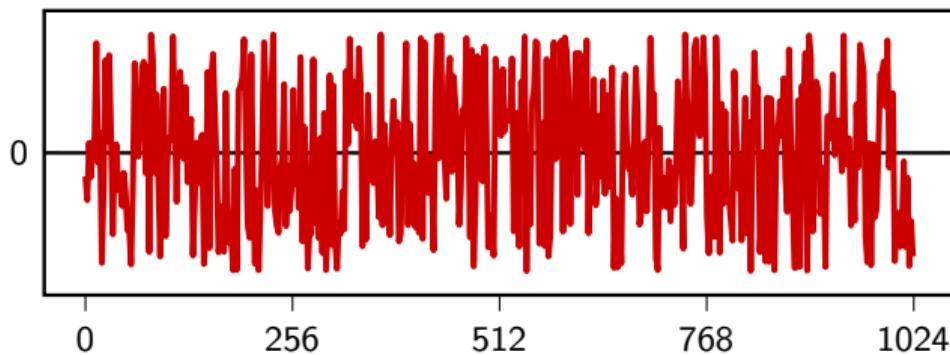
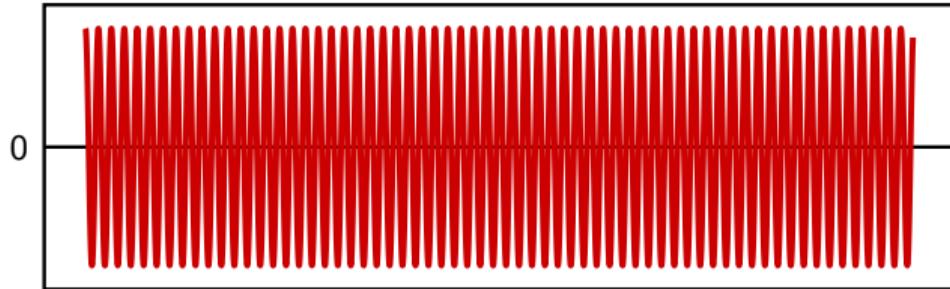
Mystery signal revisited

$$x[n] = \cos(\omega n + \phi) + \eta[n]$$

with

$$\begin{aligned}\phi &= 0 \\ \omega &= \frac{2\pi}{1024} 64\end{aligned}$$

Mystery signal unveiled



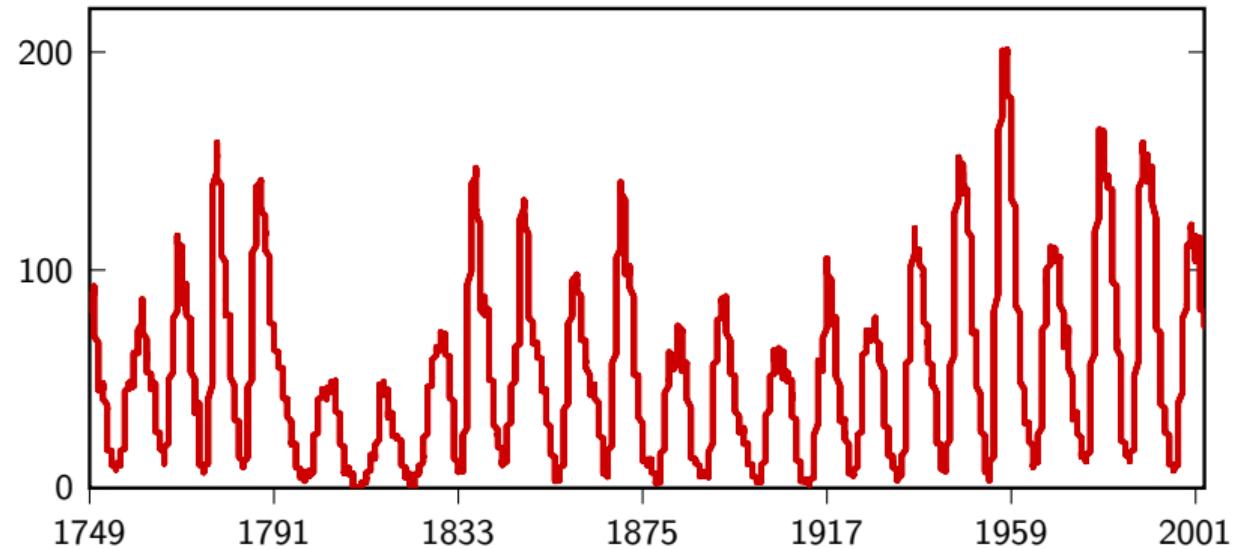
Solar spots

- ▶ sunspot number: $s = 10 \times \# \text{ of clusters} + \# \text{ of spots}$
- ▶ data set from 1749 to 2003, 2904 months

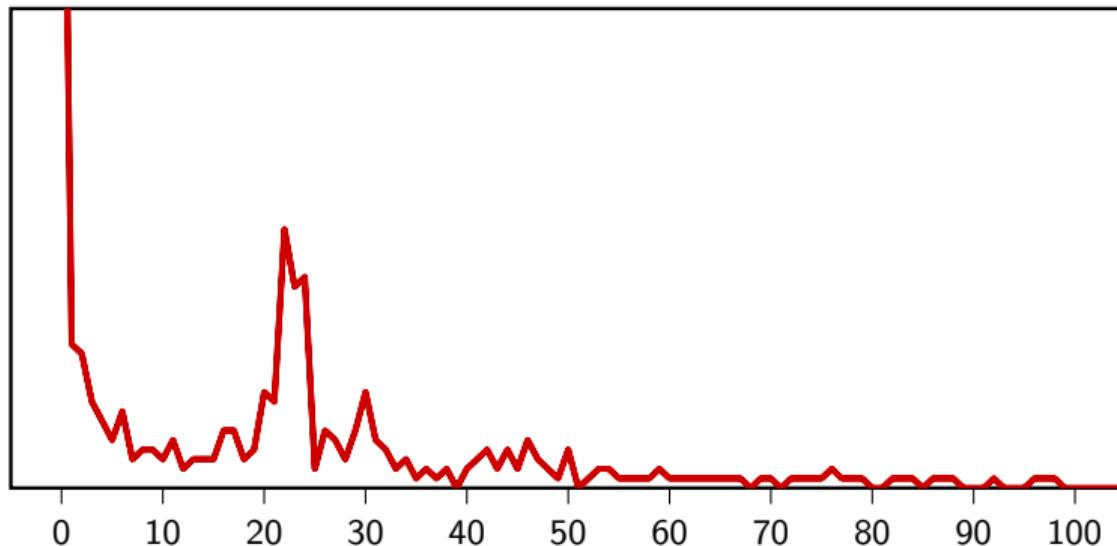
Solar spots

- ▶ sunspot number: $s = 10 \times \# \text{ of clusters} + \# \text{ of spots}$
- ▶ data set from 1749 to 2003, 2904 months

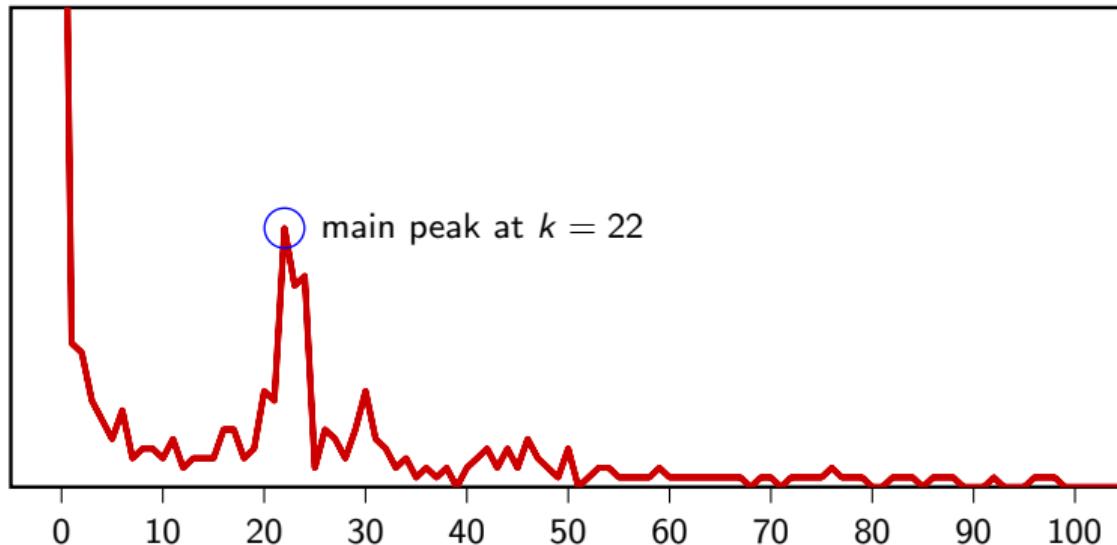
Solar spots



Solar spots



Solar spots



Solar spots

- ▶ DFT main peak for $k = 22$
- ▶ 22 cycles over 2904 months
- ▶ period: $\frac{2904}{22} \approx 11$ years

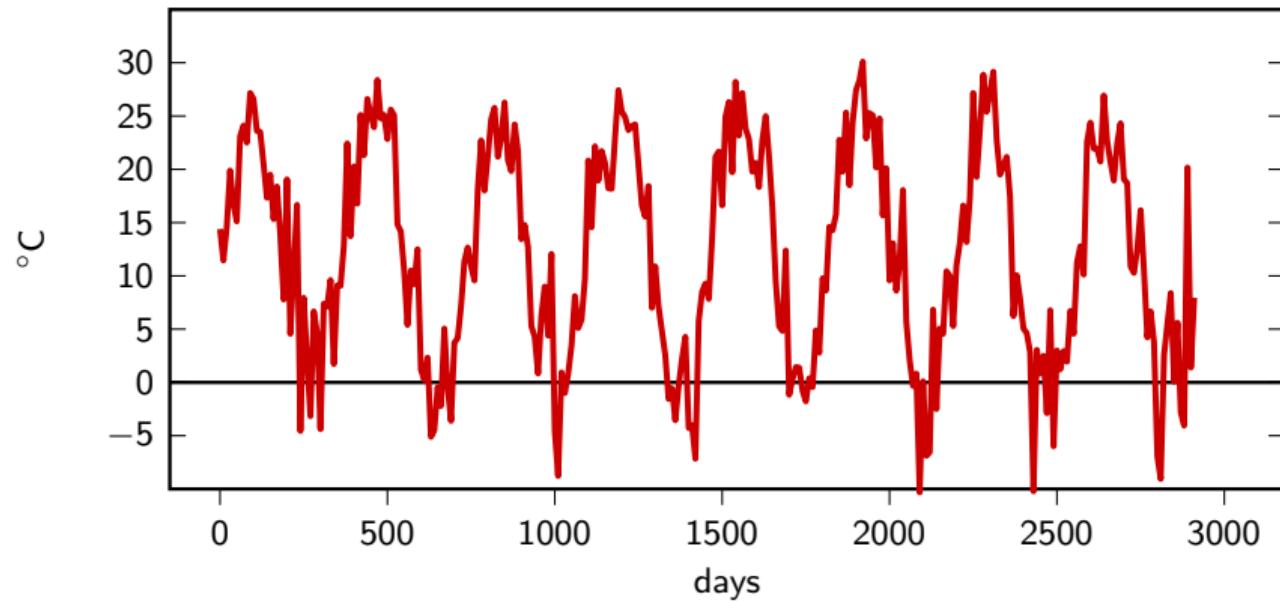
Solar spots

- ▶ DFT main peak for $k = 22$
- ▶ 22 cycles over 2904 months
- ▶ period: $\frac{2904}{22} \approx 11$ years

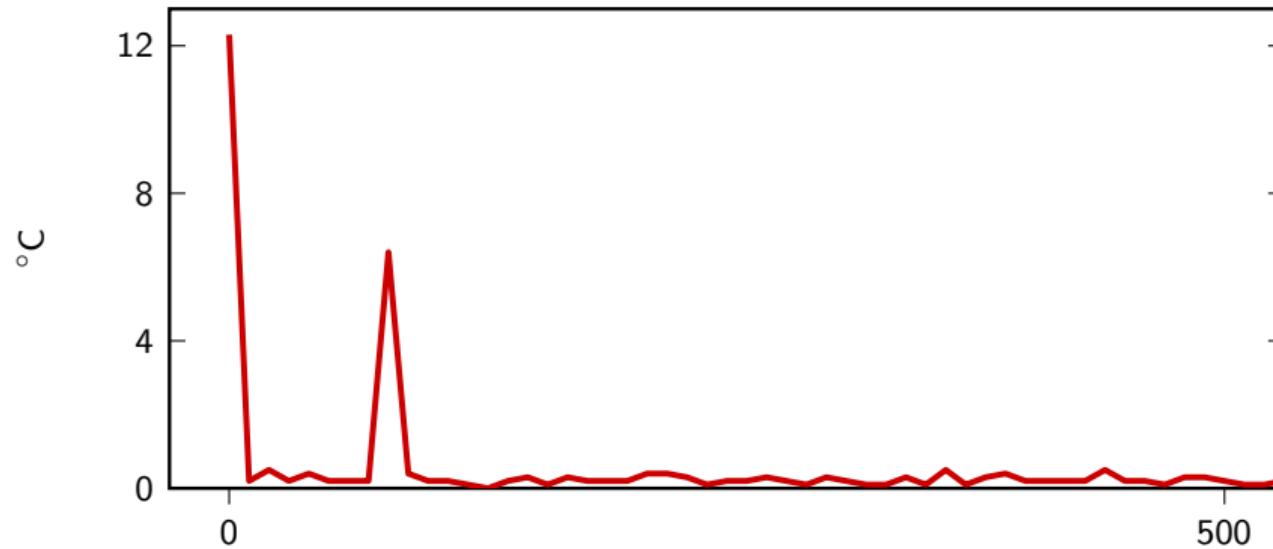
Solar spots

- ▶ DFT main peak for $k = 22$
- ▶ 22 cycles over 2904 months
- ▶ period: $\frac{2904}{22} \approx 11$ years

Daily temperature (2920 days)

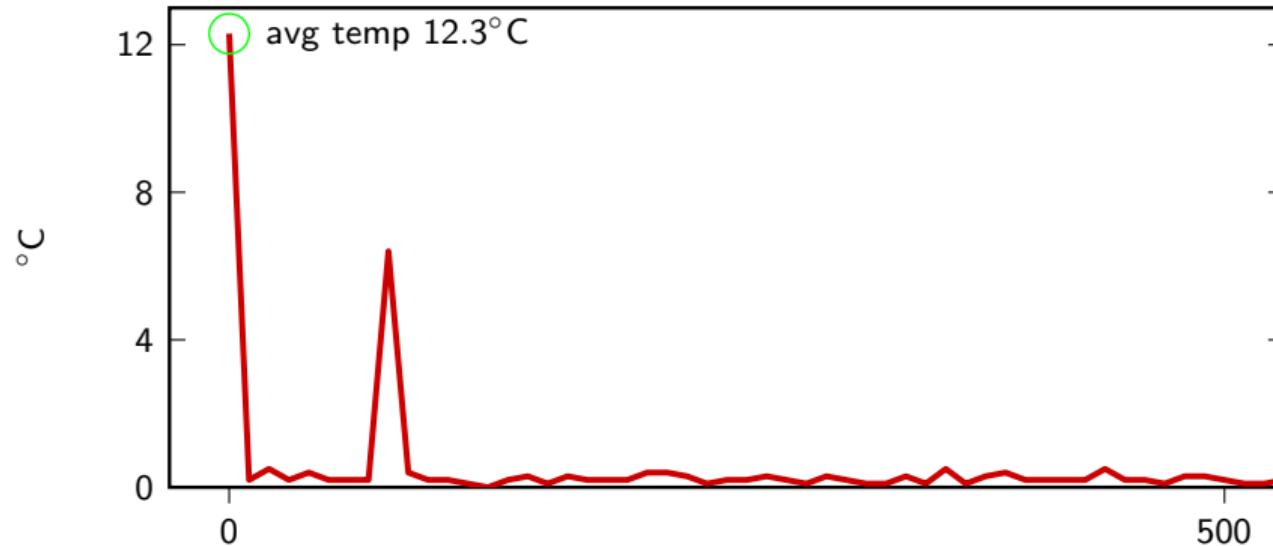


Daily temperature: DFT



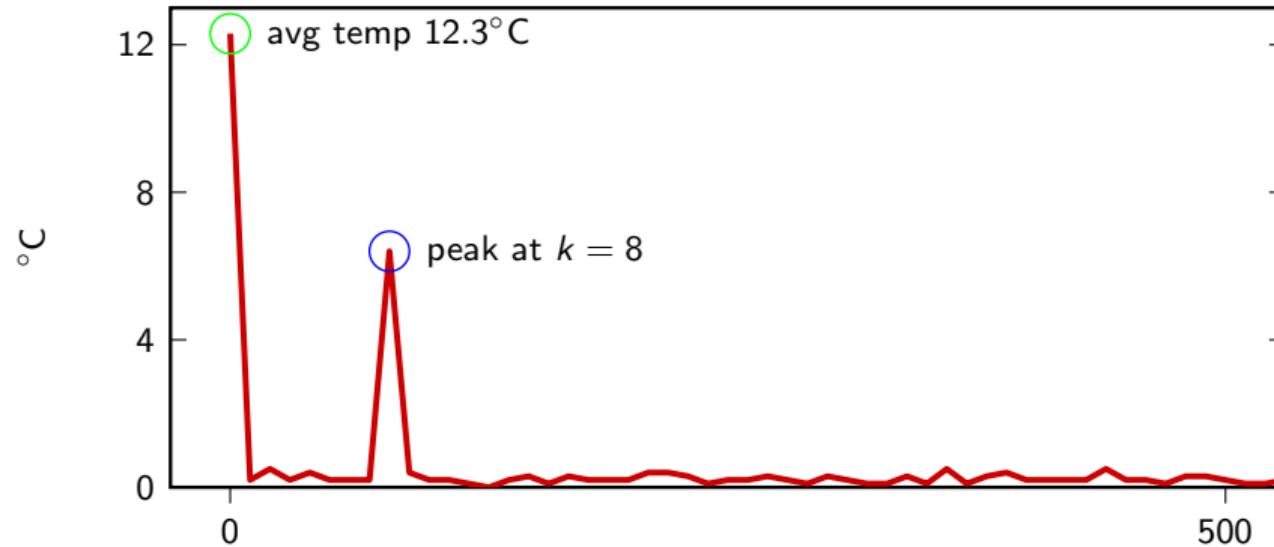
first few hundred DFT coefficients
(in magnitude and normalized by the length of the temperature vector)

Daily temperature: DFT



first few hundred DFT coefficients
(in magnitude and normalized by the length of the temperature vector)

Daily temperature: DFT



first few hundred DFT coefficients
(in magnitude and normalized by the length of the temperature vector)

Daily temperature

- ▶ average value (0-th DFT coefficient): 12.3°C
- ▶ DFT main peak for $k = 8$, value 6.4°C
- ▶ 8 cycles over 2920 days
- ▶ period: $\frac{2920}{8} = 365$ days
- ▶ temperature excursion: $12.3^{\circ}\text{C} \pm 12.8^{\circ}\text{C}$

Daily temperature

In case you're wondering why $\pm 12.8^\circ$:

$$\text{DFT} \left\{ A \cos \left(\frac{2\pi}{N} Mn \right) \right\} [k] = \begin{cases} \frac{A}{2} N & \text{for } k = M, N - M \\ 0 & \text{otherwise} \end{cases}$$

Labeling the frequency axis

If we know the “clock” of the system T_s (or its frequency F_s)

- ▶ fastest (positive) frequency is $\omega = \pi$
- ▶ sinusoid at $\omega = \pi$ needs two samples to do a full revolution
- ▶ time between samples: $T_s = 1/F_s$ seconds
- ▶ real-world period for fastest sinusoid: $2T_s$ seconds
- ▶ real-world frequency for fastest sinusoid: $F_s/2$ Hz

Labeling the frequency axis

If we know the “clock” of the system T_s (or its frequency F_s)

- ▶ fastest (positive) frequency is $\omega = \pi$
- ▶ sinusoid at $\omega = \pi$ needs two samples to do a full revolution
- ▶ time between samples: $T_s = 1/F_s$ seconds
- ▶ real-world period for fastest sinusoid: $2T_s$ seconds
- ▶ real-world frequency for fastest sinusoid: $F_s/2$ Hz

Labeling the frequency axis

If we know the “clock” of the system T_s (or its frequency F_s)

- ▶ fastest (positive) frequency is $\omega = \pi$
- ▶ sinusoid at $\omega = \pi$ needs two samples to do a full revolution
- ▶ time between samples: $T_s = 1/F_s$ seconds
- ▶ real-world period for fastest sinusoid: $2T_s$ seconds
- ▶ real-world frequency for fastest sinusoid: $F_s/2$ Hz

Labeling the frequency axis

If we know the “clock” of the system T_s (or its frequency F_s)

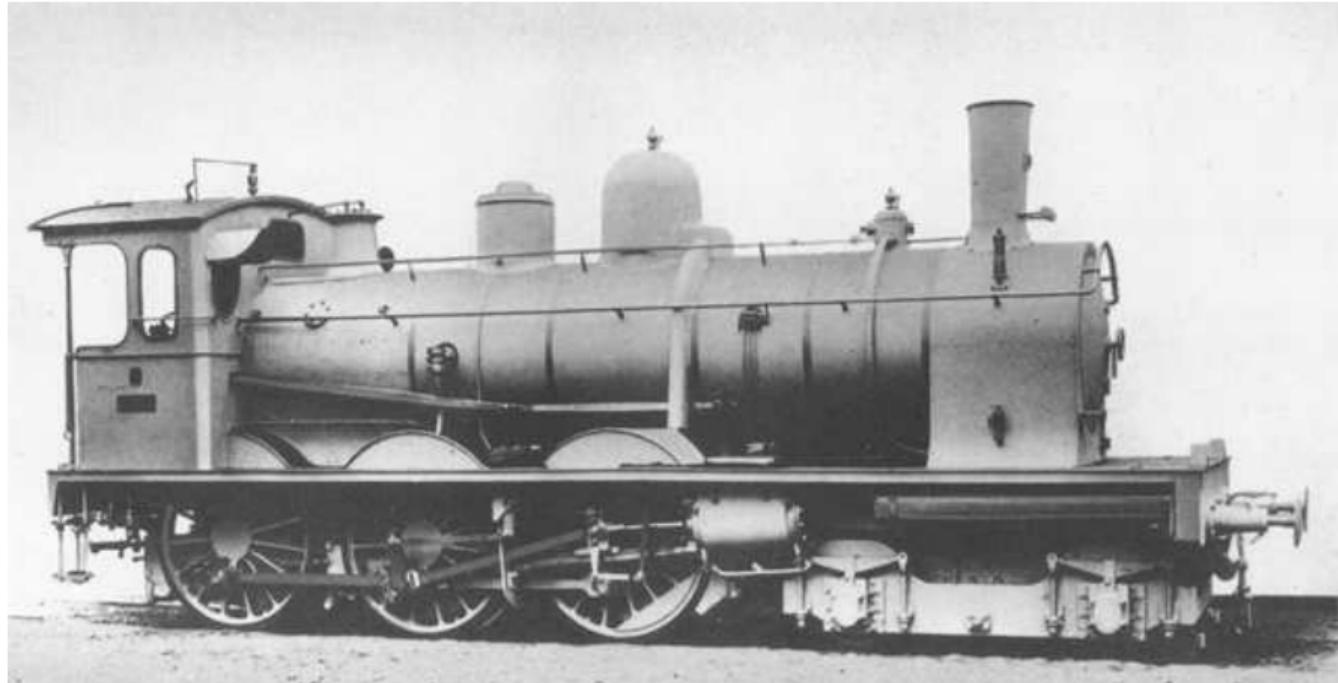
- ▶ fastest (positive) frequency is $\omega = \pi$
- ▶ sinusoid at $\omega = \pi$ needs two samples to do a full revolution
- ▶ time between samples: $T_s = 1/F_s$ seconds
- ▶ real-world period for fastest sinusoid: $2T_s$ seconds
- ▶ real-world frequency for fastest sinusoid: $F_s/2$ Hz

Labeling the frequency axis

If we know the “clock” of the system T_s (or its frequency F_s)

- ▶ fastest (positive) frequency is $\omega = \pi$
- ▶ sinusoid at $\omega = \pi$ needs two samples to do a full revolution
- ▶ time between samples: $T_s = 1/F_s$ seconds
- ▶ real-world period for fastest sinusoid: $2T_s$ seconds
- ▶ real-world frequency for fastest sinusoid: $F_s/2$ Hz

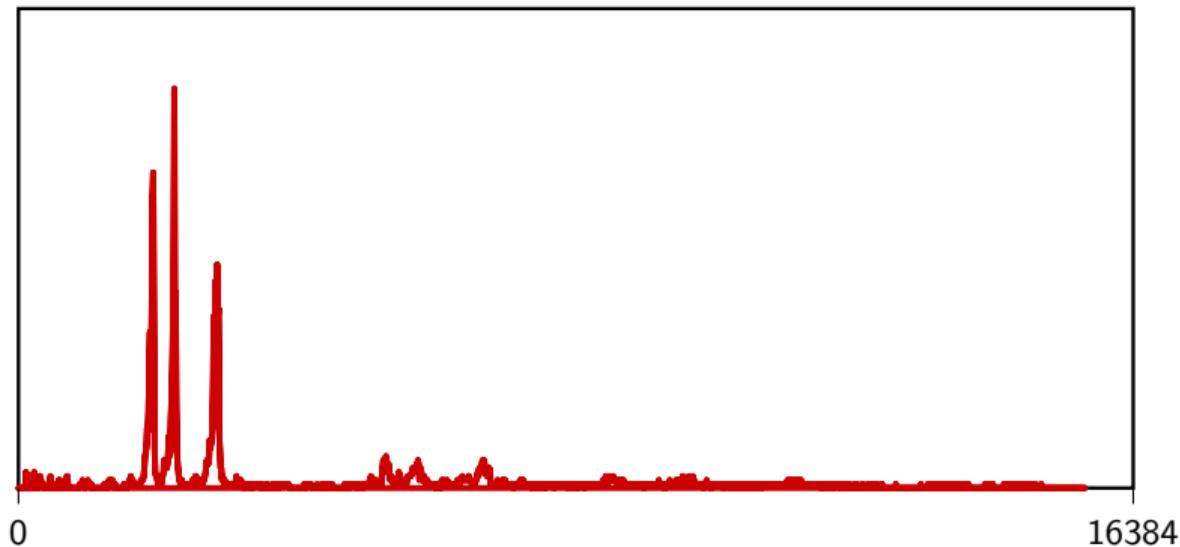
Example: train whistle



Play

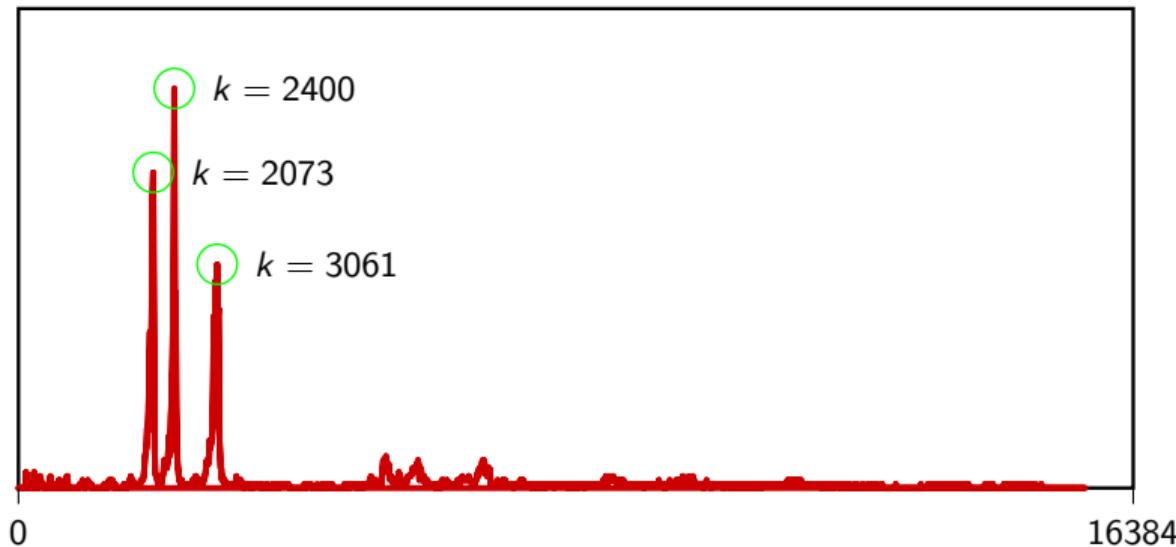
Example: train whistle

32768 samples (the “clock” of the system $F_s = 8000\text{Hz}$)



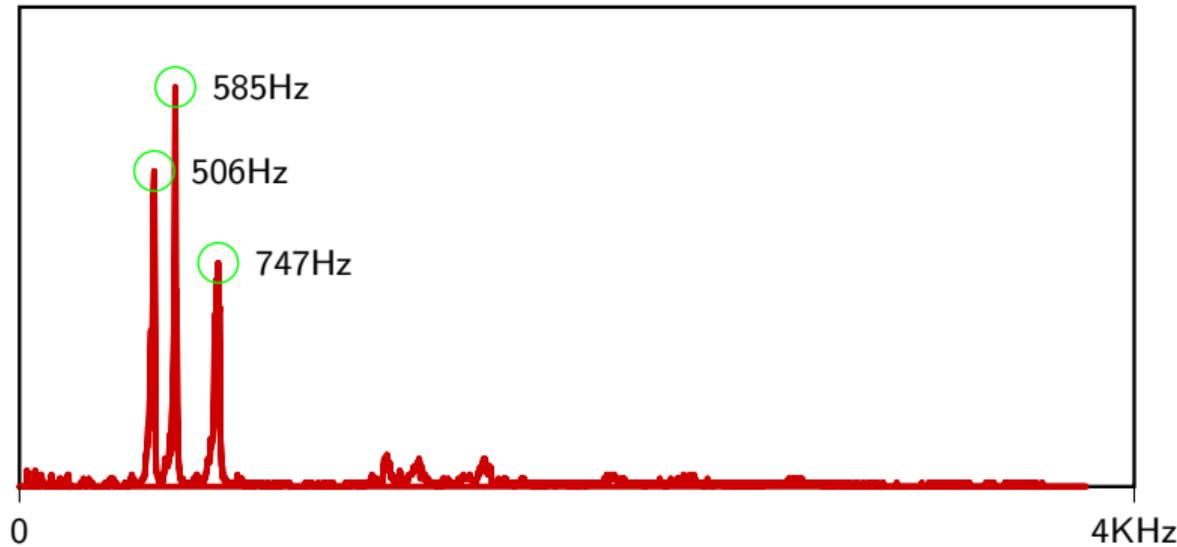
Example: train whistle

32768 samples (the “clock” of the system $F_s = 8000\text{Hz}$)



Example: train whistle

the “clock” of the system $F_s = 8000\text{Hz}$



Example: train whistle

if we look up the frequencies:



B minor chord

the DFT as a synthesis tool

DFT formulas

Analysis formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}, \quad k = 0, 1, \dots, N-1$$

N-point signal in the *frequency domain*

Synthesis formula:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}nk}, \quad n = 0, 1, \dots, N-1$$

N-point signal in the “*time*” domain

DFT formulas

Analysis formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi}{N} nk}, \quad k = 0, 1, \dots, N-1$$

N-point signal in the *frequency domain*

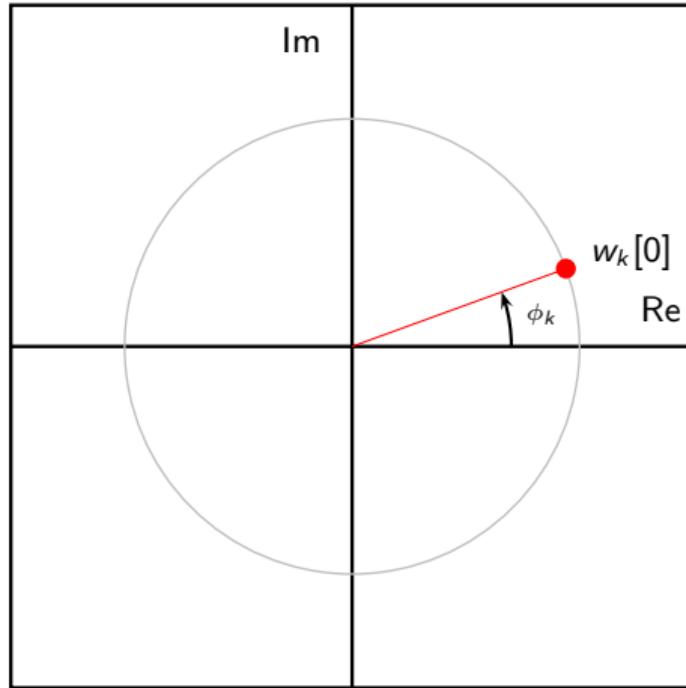
Synthesis formula:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j \frac{2\pi}{N} nk}, \quad n = 0, 1, \dots, N-1$$

N-point signal in the “*time*” domain

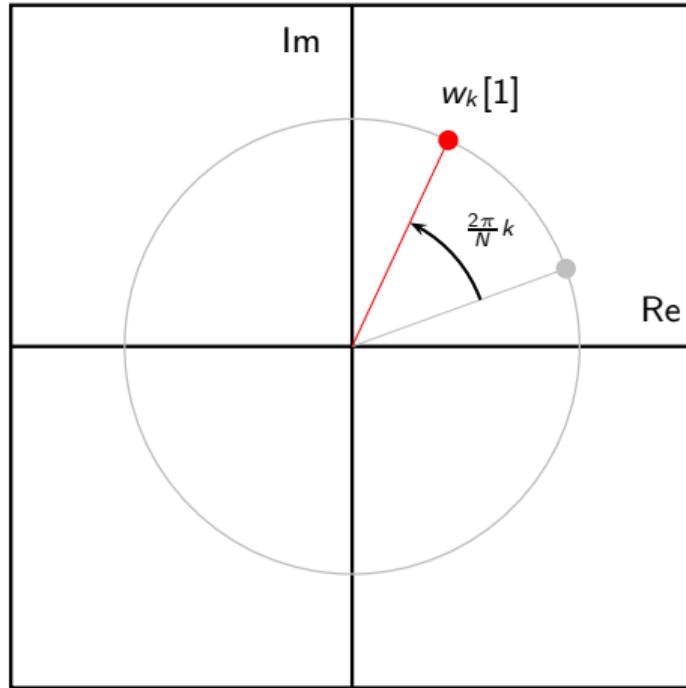
Synthesis: the sinusoidal generator

$$w_k[n] = e^{j(\frac{2\pi}{N}kn + \phi_k)}$$



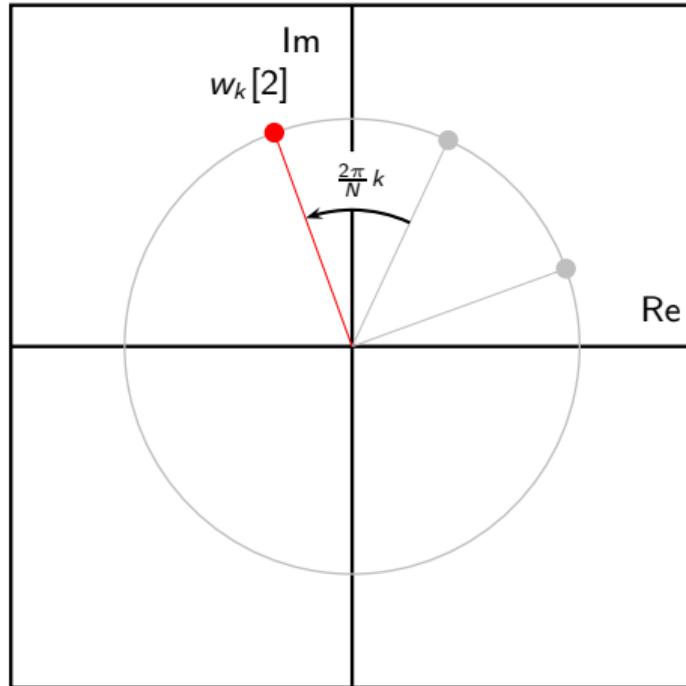
Synthesis: the sinusoidal generator

$$w_k[n] = e^{j(\frac{2\pi}{N}kn + \phi_k)}$$



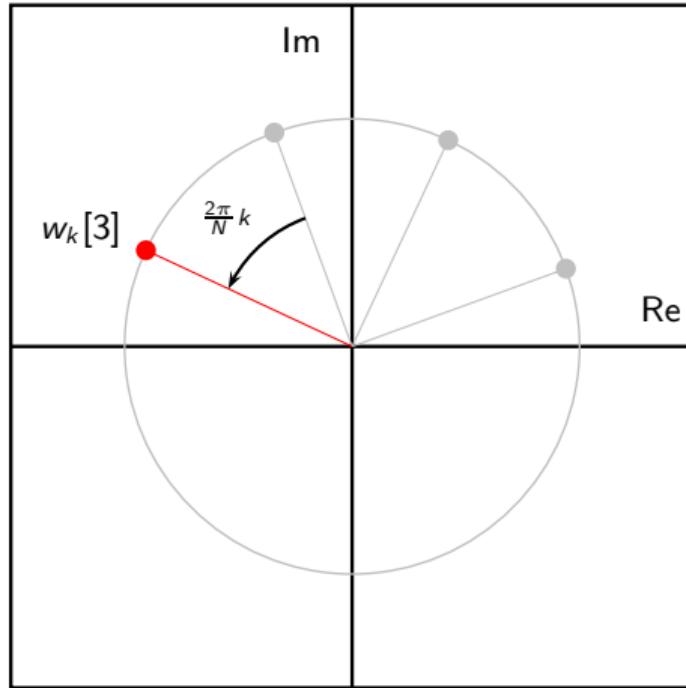
Synthesis: the sinusoidal generator

$$w_k[n] = e^{j(\frac{2\pi}{N}kn + \phi_k)}$$



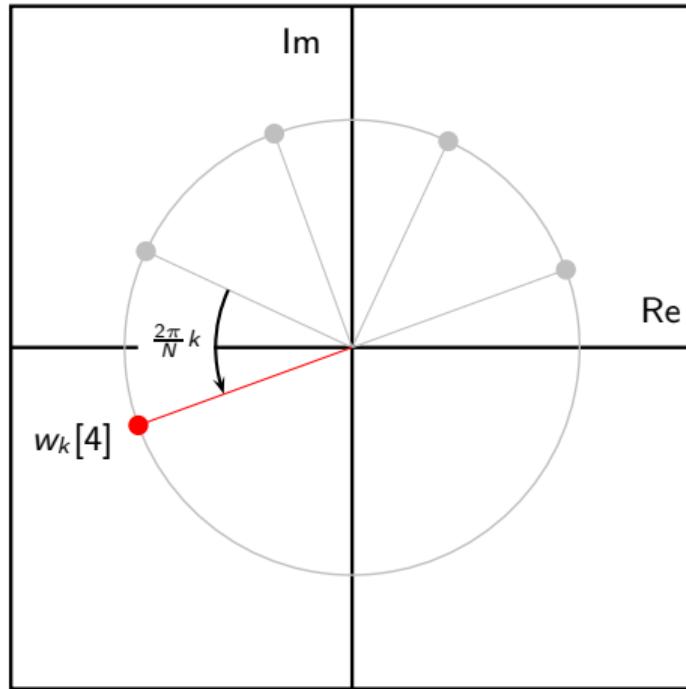
Synthesis: the sinusoidal generator

$$w_k[n] = e^{j(\frac{2\pi}{N}kn + \phi_k)}$$

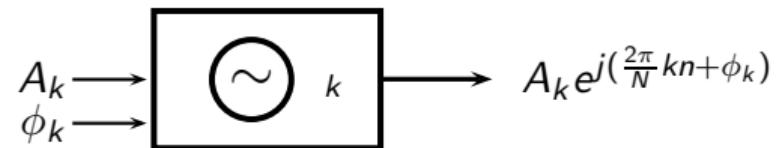


Synthesis: the sinusoidal generator

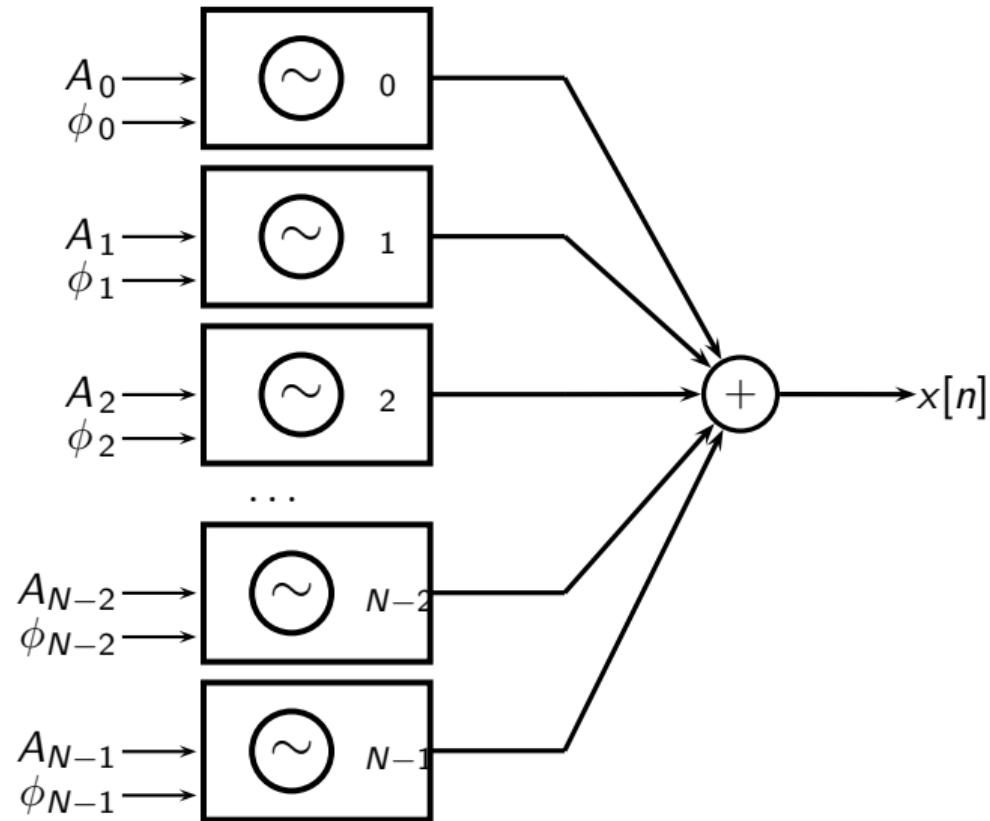
$$w_k[n] = e^{j(\frac{2\pi}{N}kn + \phi_k)}$$



Synthesis: the sinusoidal generator



DFT synthesis formula



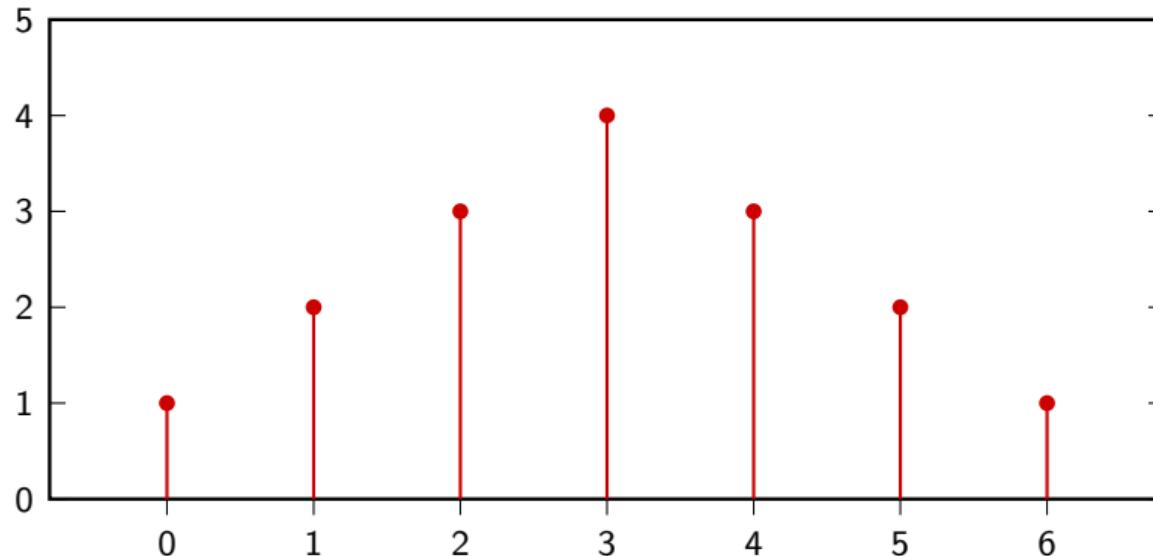
Initializing the machine

$$A_k = |X[k]|/N$$

$$\phi_k = \angle X[k]$$

Example

$$\mathbf{x} = [1 \ 2 \ 3 \ 4 \ 3 \ 2 \ 1]^T$$



Example

k	A_k	ϕ_k
0	2.2857	0.0000
1	0.7213	-2.6928
2	0.0440	0.8976
3	0.0919	-1.7952
4	0.0919	1.7952
5	0.0440	-0.8976
6	0.7213	2.6928

Example

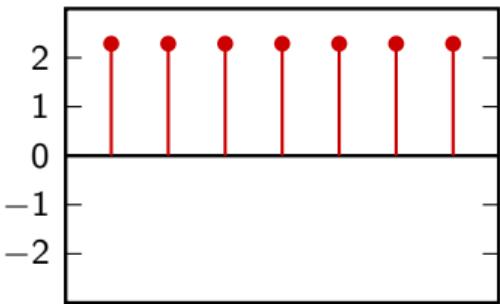
$$k = 0$$

$$A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

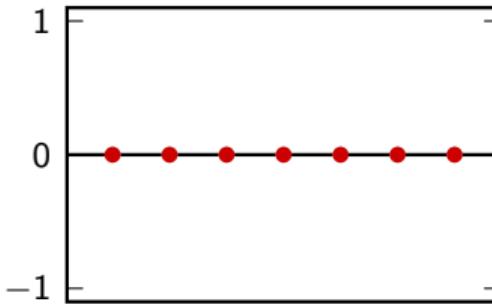
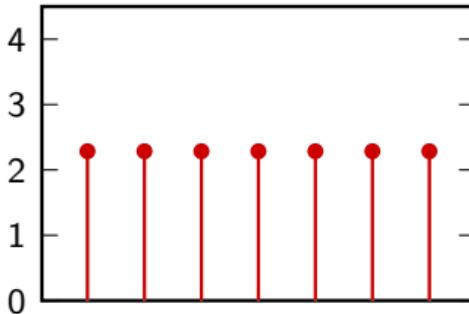
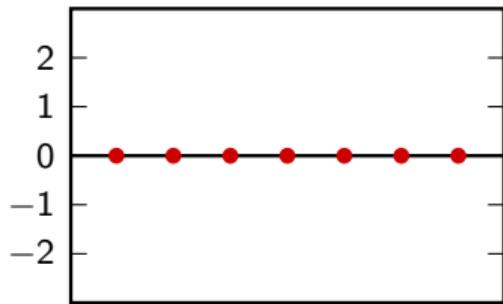
$$A_0 = 2.28, \phi_0 = 0$$

$$\sum_{k=0}^0 A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

Re



Im



Example

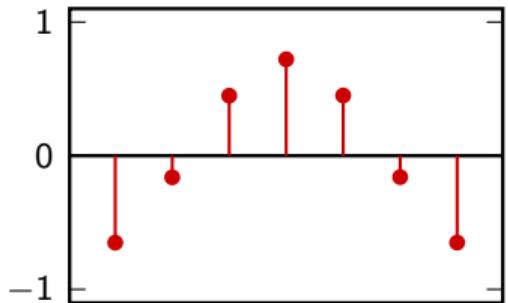
$$k = 1$$

$$A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

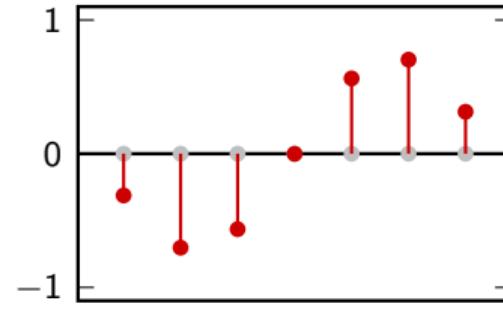
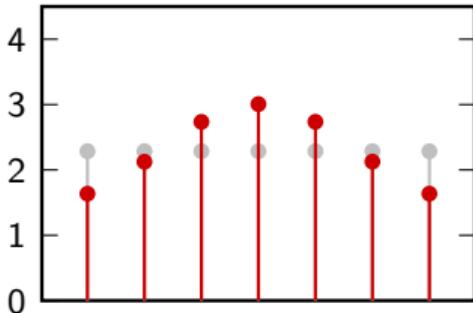
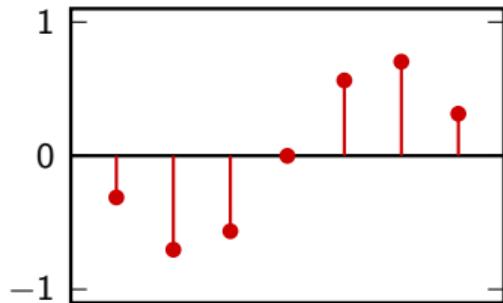
$$A_1 = 0.72, \phi_1 = -2.69$$

$$\sum_{k=0}^1 A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

Re



Im



Example

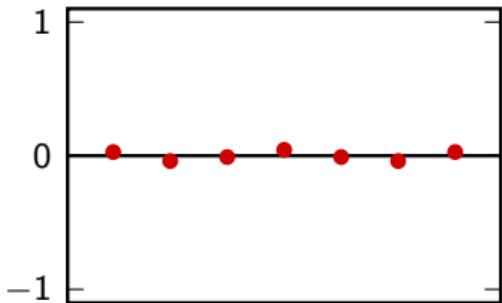
$$k = 2$$

$$A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

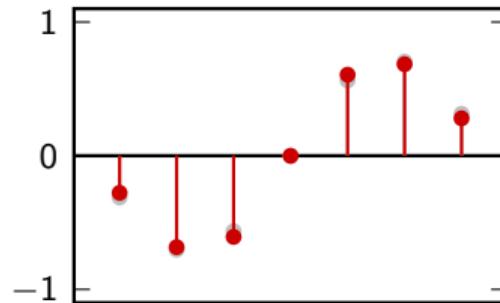
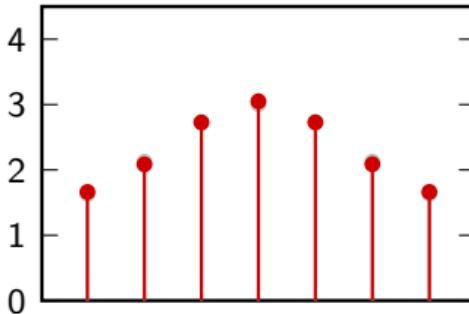
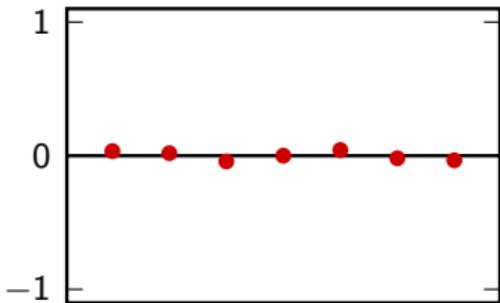
$$A_2 = 0.04, \phi_2 = 0.89$$

$$\sum_{k=0}^2 A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

Re



Im



Example

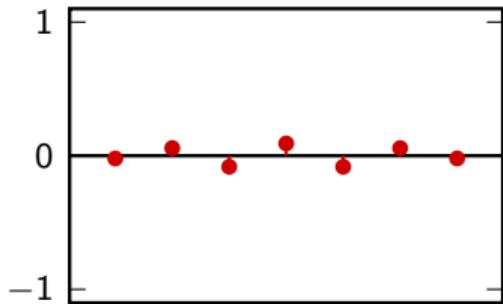
$$k = 3$$

$$A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

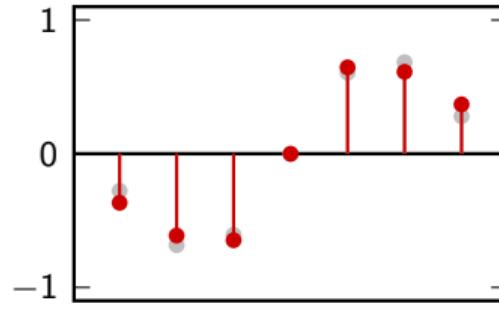
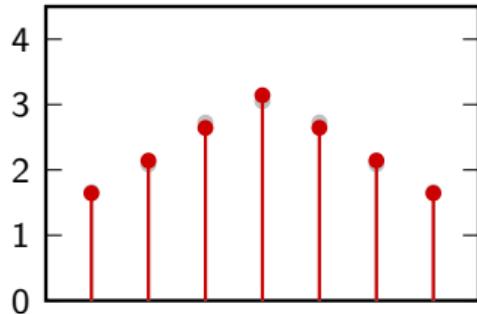
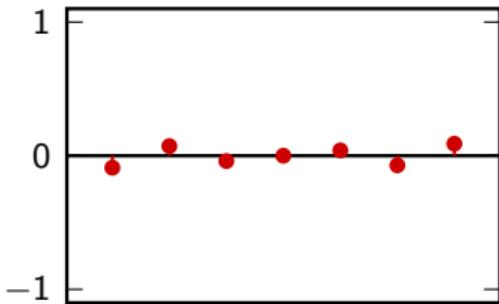
$$A_3 = 0.09, \phi_3 = -1.79$$

$$\sum_{k=0}^3 A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

Re



Im



Example

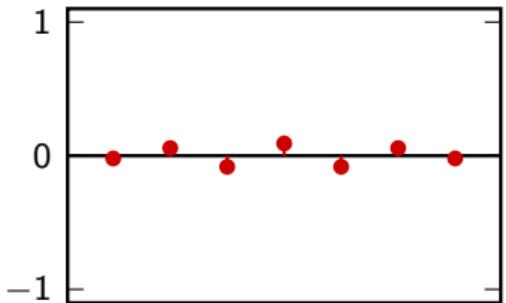
$$k = 4$$

$$A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

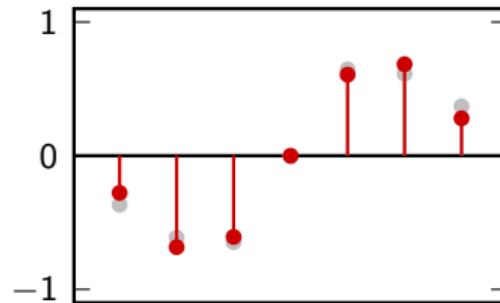
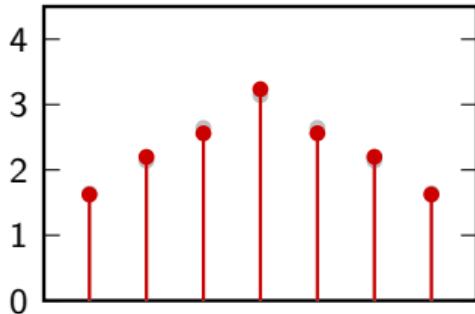
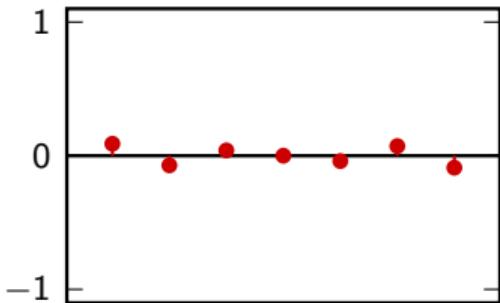
$$A_4 = 0.09, \phi_4 = 1.79$$

$$\sum_{k=0}^4 A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

Re



Im



Example

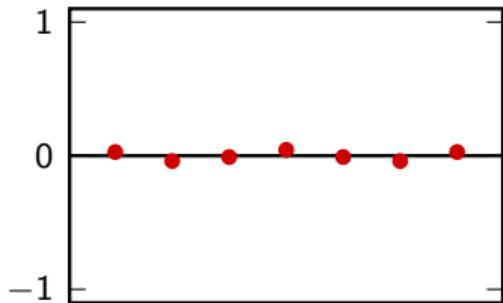
$$k = 5$$

$$A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

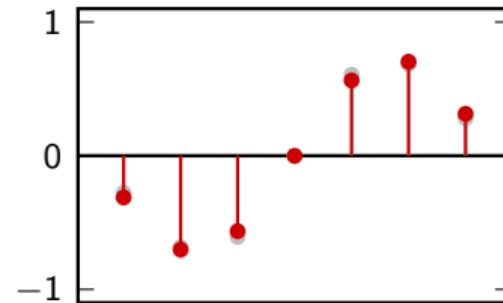
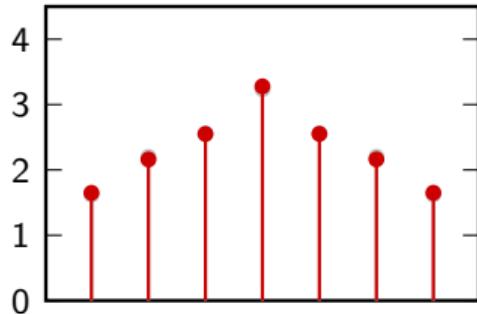
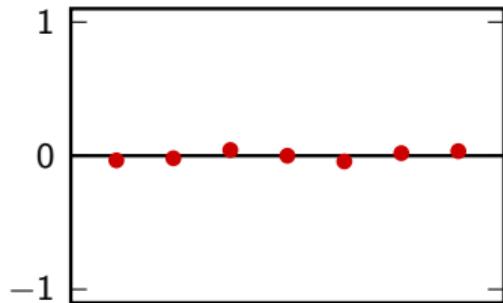
$$A_5 = 0.04, \phi_5 = -0.89$$

$$\sum_{k=0}^5 A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

Re



Im



Example

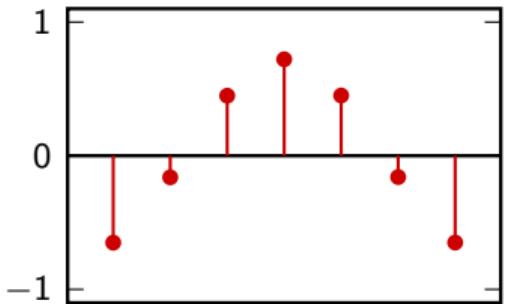
$$k = 6$$

$$A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

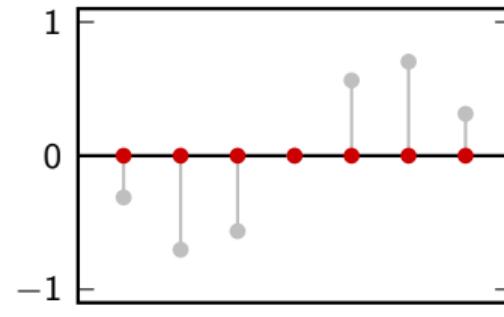
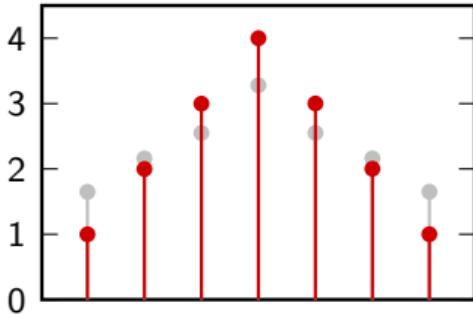
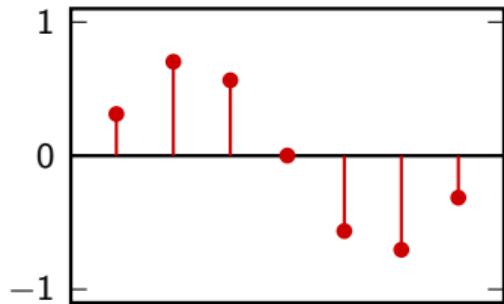
$$A_6 = 0.72, \phi_6 = 2.69$$

$$\sum_{k=0}^6 A_k e^{j(\frac{2\pi}{N} kn + \phi_k)}$$

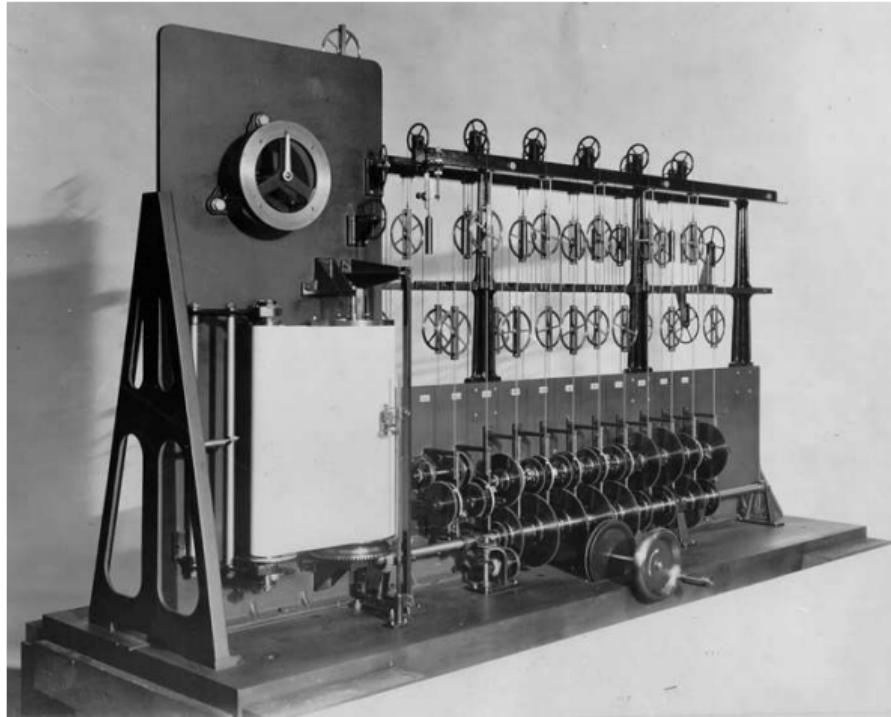
Re



Im



The machine before DSP



tide-predicting machine (originally invented by Lord Kelvin)

Wonderful website

<http://jackschaedler.github.io/circles-sines-signals>

Running the machine too long...

$$x[n + N] = x[n]$$

output signal is N -periodic!

Inherent periodicities in the DFT

the synthesis formula:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j \frac{2\pi}{N} nk}, \quad n = 0, 1, \dots, N-1$$

produces an N -point signal in the time domain

the analysis formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi}{N} nk}, \quad k = 0, 1, \dots, N-1$$

produces an N -point signal in the frequency domain

Inherent periodicities in the DFT

the synthesis formula:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}nk}, \quad n \in \mathbb{Z}$$

produces an N -periodic signal in the time domain

the analysis formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}, \quad k = 0, 1, \dots, N-1$$

produces an N -point signal in the frequency domain

Inherent periodicities in the DFT

the synthesis formula:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}nk}, \quad n \in \mathbb{Z}$$

produces an **N-periodic** signal in the time domain

the analysis formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}, \quad k \in \mathbb{Z}$$

produces an **N-periodic** signal in the frequency domain

Discrete Fourier Series (DFS)

$\text{DFS} = \text{DFT}$ with periodicity explicit

- ▶ the DFS maps an N -periodic signal onto an N -periodic sequence of Fourier coefficients
- ▶ the inverse DFS maps an N -periodic sequence of Fourier coefficients a set onto an N -periodic signal
- ▶ the DFS of an N -periodic signal is mathematically equivalent to the DFT of one period

Finite-length time shifts revisited

The DFS helps us understand how to define time shifts for finite-length signals.

For an N - periodic sequence $\tilde{x}[n]$:

- ▶ $\tilde{x}[n - M]$ is well-defined for all $M \in \mathbb{N}$
- ▶ $\text{DFS}\{\tilde{x}[n - M]\} = e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k]$ (easy derivation)
- ▶ $\text{IDFS}\left\{ \tilde{X}[k] \right\} = \tilde{x}[n - M]$

Finite-length time shifts revisited

The DFS helps us understand how to define time shifts for finite-length signals.

For an N - periodic sequence $\tilde{x}[n]$:

- ▶ $\tilde{x}[n - M]$ is well-defined for all $M \in \mathbb{N}$
- ▶ $\text{DFS}\{\tilde{x}[n - M]\} = e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k]$ (easy derivation)
- ▶ $\text{IDFS}\left\{ \tilde{X}[k] \right\} = \tilde{x}[n - M]$

Finite-length time shifts revisited

The DFS helps us understand how to define time shifts for finite-length signals.

For an N - periodic sequence $\tilde{x}[n]$:

- ▶ $\tilde{x}[n - M]$ is well-defined for all $M \in \mathbb{N}$
- ▶ $\text{DFS}\{\tilde{x}[n - M]\} = e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k]$ (easy derivation)
- ▶ $\text{IDFS}\left\{ \tilde{X}[k] \right\} = \tilde{x}[n - M]$

Finite-length time shifts revisited

The DFS helps us understand how to define time shifts for finite-length signals.

For an N - periodic sequence $\tilde{x}[n]$:

- ▶ $\tilde{x}[n - M]$ is well-defined for all $M \in \mathbb{N}$
- ▶ $\text{DFS}\{\tilde{x}[n - M]\} = e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k]$ (easy derivation)
- ▶ $\text{IDFS}\left\{e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k]\right\} = \tilde{x}[n - M]$

Finite-length time shifts revisited

The DFS helps us understand how to define time shifts for finite-length signals.

For an N - periodic sequence $\tilde{x}[n]$:

- ▶ $\tilde{x}[n - M]$ is well-defined for all $M \in \mathbb{N}$
- ▶ DFS $\{\tilde{x}[n - M]\} = e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k]$ (easy derivation)
- ▶ IDFS $\left\{ e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k] \right\} = \tilde{x}[n - M]$



a delay in time becomes a *linear phase factor* in frequency

Finite-length time shifts revisited

For an N -point signal $x[n]$:

- ▶ $x[n - M]$ is *not* well-defined
- ▶ what is IDFT $\left\{ e^{-j \frac{2\pi}{N} Mk} X[k] \right\}$?

Finite-length time shifts revisited

For an N -point signal $x[n]$:

- ▶ $x[n - M]$ is *not* well-defined
- ▶ what is IDFT $\left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\}$?

Finite-length time shifts revisited

$$\begin{aligned}\text{IDFT} \left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\} [n] &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{-j\frac{2\pi}{N}Mk} e^{j\frac{2\pi}{N}nk} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \left(\sum_{m=0}^{N-1} x[m] e^{-j\frac{2\pi}{N}mk} \right) e^{-j\frac{2\pi}{N}Mk} e^{j\frac{2\pi}{N}nk} \\ &= \frac{1}{N} \sum_{m=0}^{N-1} x[m] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k}\end{aligned}$$

Finite-length time shifts revisited

$$\begin{aligned}\text{IDFT} \left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\} [n] &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{-j\frac{2\pi}{N}Mk} e^{j\frac{2\pi}{N}nk} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \left(\sum_{m=0}^{N-1} x[m] e^{-j\frac{2\pi}{N}mk} \right) e^{-j\frac{2\pi}{N}Mk} e^{j\frac{2\pi}{N}nk} \\ &= \frac{1}{N} \sum_{m=0}^{N-1} x[m] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k}\end{aligned}$$

Finite-length time shifts revisited

$$\begin{aligned}\text{IDFT} \left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\} [n] &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{-j\frac{2\pi}{N}Mk} e^{j\frac{2\pi}{N}nk} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \left(\sum_{m=0}^{N-1} x[m] e^{-j\frac{2\pi}{N}mk} \right) e^{-j\frac{2\pi}{N}Mk} e^{j\frac{2\pi}{N}nk} \\ &= \frac{1}{N} \sum_{m=0}^{N-1} x[m] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k}\end{aligned}$$

We've seen something like this before...

$$\sum_{n=0}^{N-1} e^{-j\frac{2\pi}{N}nk} = \begin{cases} N & \text{if } k \text{ multiple of } N \\ 0 & \text{otherwise} \end{cases}$$

(remember the orthogonality proof for the DFT basis)

Finite-length time shifts revisited

$$\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}((n-M)-m)k} = \begin{cases} N & \text{for } ((n - M) - m) \text{ multiple of } N \\ 0 & \text{otherwise} \end{cases}$$

is there an m such that $0 \leq m < N$ and $((n - M) - m)$ multiple of N ?

Finite-length time shifts revisited

$$\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}((n-M)-m)k} = \begin{cases} N & \text{for } ((n - M) - m) \text{ multiple of } N \\ 0 & \text{otherwise} \end{cases}$$

is there an m such that $0 \leq m < N$ and $((n - M) - m)$ multiple of N ?

Modulo operator

given $C \in \mathbb{N}$, find m such that $0 \leq m < N$ and $C - m$ is a multiple of N

any integer C can be written as $C = pN + (C \bmod N)$, $p \in \mathbb{N}$:

- ▶ $0 \leq (C \bmod N) < N$
- ▶ $C - (C \bmod N)$ is a multiple of N

Modulo operator

given $C \in \mathbb{N}$, find m such that $0 \leq m < N$ and $C - m$ is a multiple of N

any integer C can be written as $C = pN + (C \bmod N)$, $p \in \mathbb{N}$:

- ▶ $0 \leq (C \bmod N) < N$
- ▶ $C - (C \bmod N)$ is a multiple of N

Finite-length time shifts revisited

$$\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k} = \begin{cases} N & \text{for } m = (n - M) \mod N \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{IDFT} \left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\} [n] &= \frac{1}{N} \sum_{m=0}^{N-1} x[m] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k} \\ &= x[(n - M) \mod N] \end{aligned}$$

shifts for finite-length signals are “naturally” circular

Finite-length time shifts revisited

$$\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k} = \begin{cases} N & \text{for } m = (n - M) \bmod N \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{IDFT} \left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\} [n] &= \frac{1}{N} \sum_{m=0}^{N-1} x[m] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k} \\ &= x[(n - M) \bmod N] \end{aligned}$$

shifts for finite-length signals are “naturally” circular

Finite-length time shifts revisited

$$\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k} = \begin{cases} N & \text{for } m = (n - M) \bmod N \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{IDFT} \left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\} [n] &= \frac{1}{N} \sum_{m=0}^{N-1} x[m] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k} \\ &= x[(n - M) \bmod N] \end{aligned}$$

shifts for finite-length signals are “naturally” circular

COM303: Digital Signal Processing

Lecture 6: DFS and DTFT

Overview

- ▶ periodicity in the DFT
- ▶ the DFS
- ▶ the DTFT

Overview

- ▶ periodicity in the DFT
- ▶ the DFS
- ▶ the DTFT

Overview

- ▶ periodicity in the DFT
- ▶ the DFS
- ▶ the DTFT

DFT formulas

Analysis formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}, \quad k = 0, 1, \dots, N-1$$

N-point signal in the *frequency domain*

Synthesis formula:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}nk}, \quad n = 0, 1, \dots, N-1$$

N-point signal in the “*time*” domain

DFT formulas

Analysis formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}, \quad k = 0, 1, \dots, N-1$$

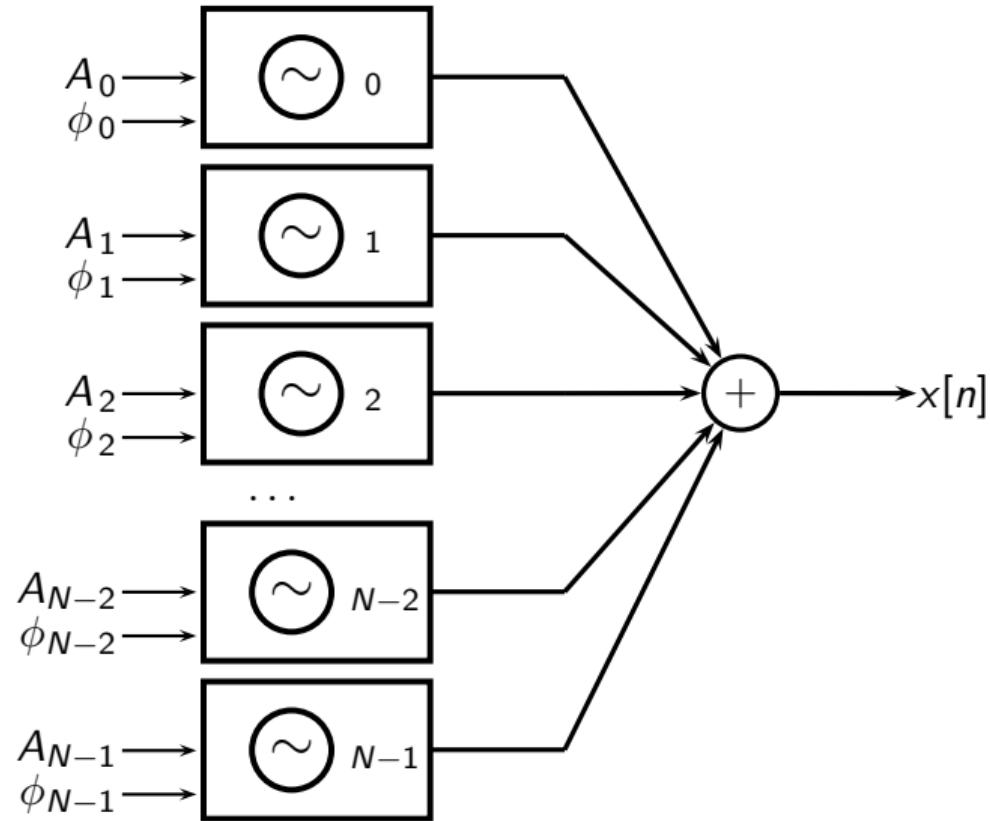
N-point signal in the *frequency domain*

Synthesis formula:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}nk}, \quad n = 0, 1, \dots, N-1$$

N-point signal in the “*time*” domain

DFT synthesis formula



Running the machine too long...

$$x[n + N] = x[n]$$

output signal is N -periodic!

Inherent periodicities in the DFT

the synthesis formula:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j \frac{2\pi}{N} nk}, \quad n = 0, 1, \dots, N-1$$

produces an N -point signal in the time domain

the analysis formula:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi}{N} nk}, \quad k = 0, 1, \dots, N-1$$

produces an N -point signal in the frequency domain

Inherent periodicities in the DFT

the synthesis formula:

$$\tilde{x}[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}nk}, \quad n \in \mathbb{Z}$$

produces an N -periodic signal in the time domain

the analysis formula:

$$\tilde{X}[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}, \quad k = 0, 1, \dots, N-1$$

produces an N -point signal in the frequency domain

Inherent periodicities in the DFT

the synthesis formula:

$$\tilde{x}[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}nk}, \quad n \in \mathbb{Z}$$

produces an N -periodic signal in the time domain

the analysis formula:

$$\tilde{X}[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}, \quad k \in \mathbb{Z}$$

produces an N -periodic signal in the frequency domain

Discrete Fourier Series (DFS)

$\text{DFS} = \text{DFT}$ with periodicity explicit

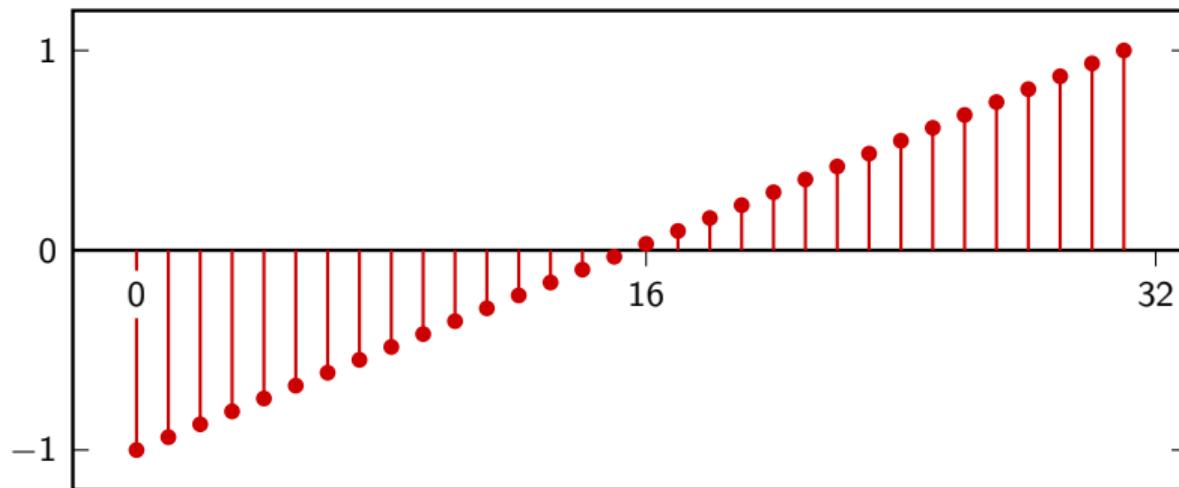
- ▶ the DFS maps an N -periodic signal onto an N -periodic sequence of Fourier coefficients
- ▶ the inverse DFS maps an N -periodic sequence of Fourier coefficients a set onto an N -periodic signal
- ▶ the DFS of an N -periodic signal is mathematically equivalent to the DFT of one period

the Fourier transform for periodic signals

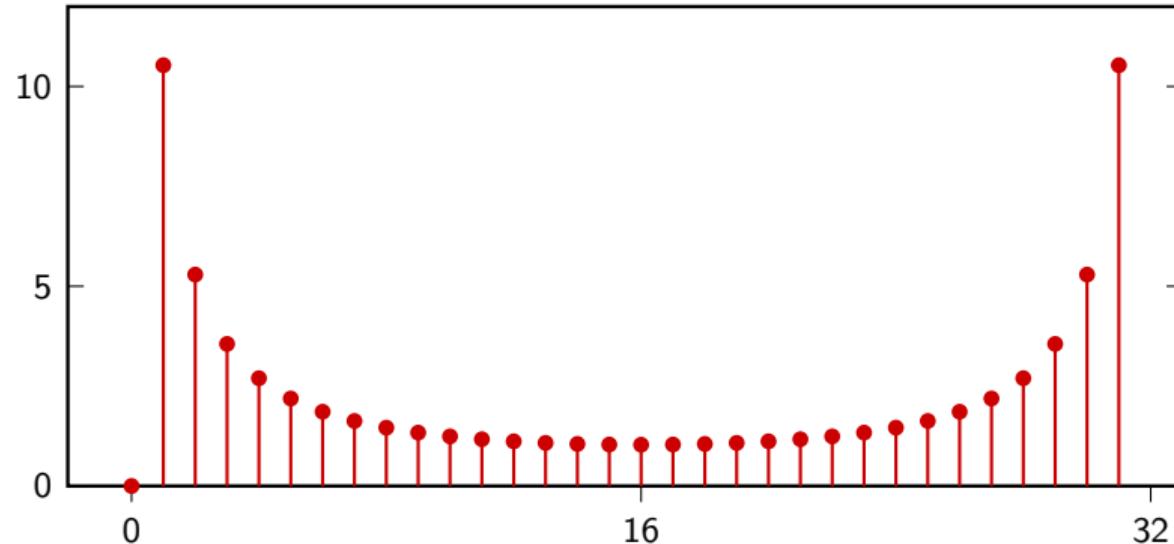
Periodic sequences: a bridge to infinite-length signals

- ▶ N -periodic sequence: N degrees of freedom
- ▶ DFS: only N Fourier coefficients capture all the information

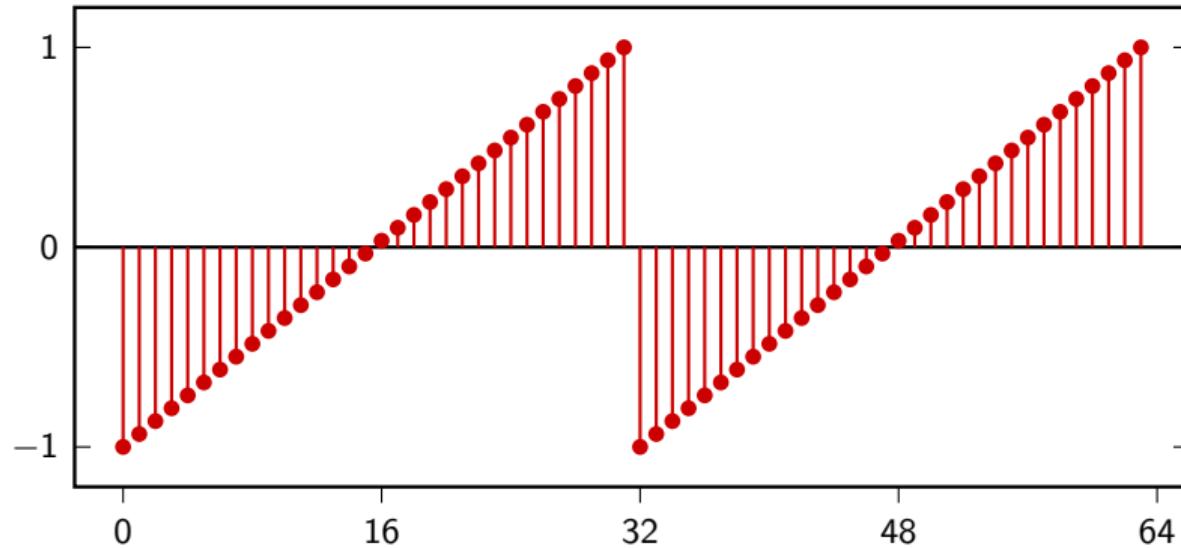
Example: 32-tap sawtooth wave



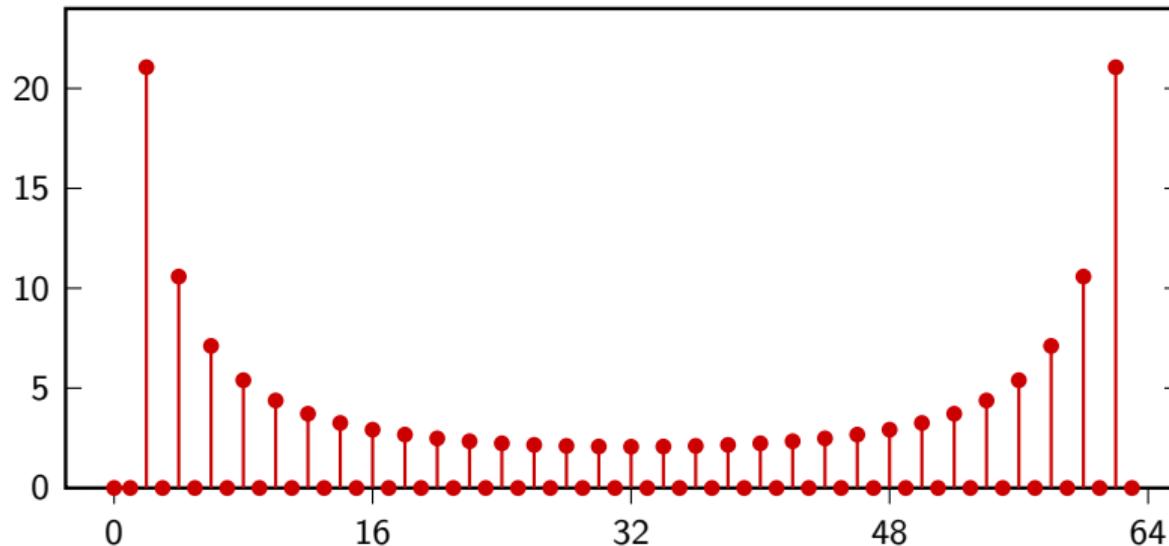
Example: DFT of 32-tap sawtooth wave



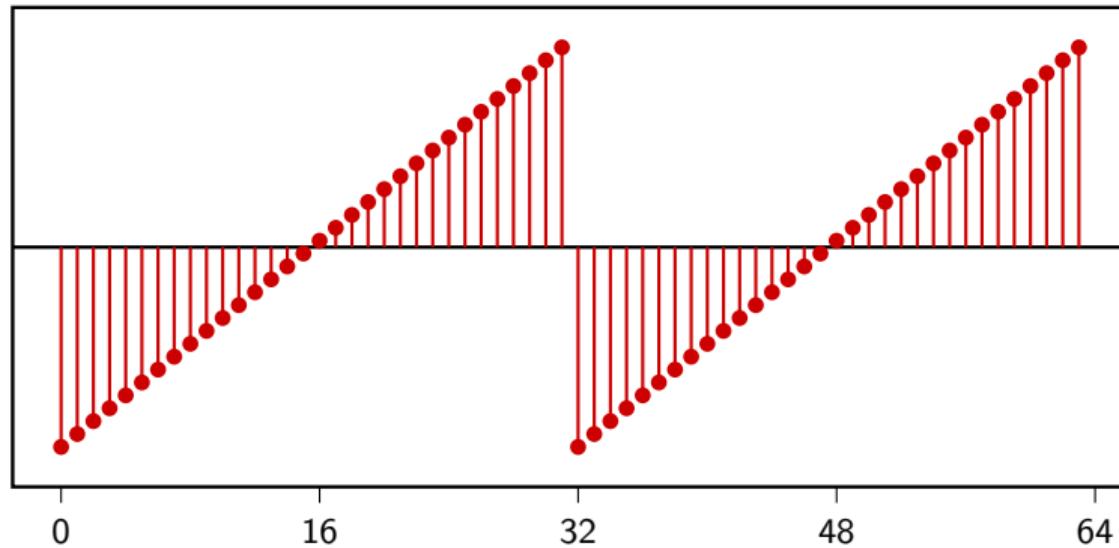
What if we take the DFT of two periods?



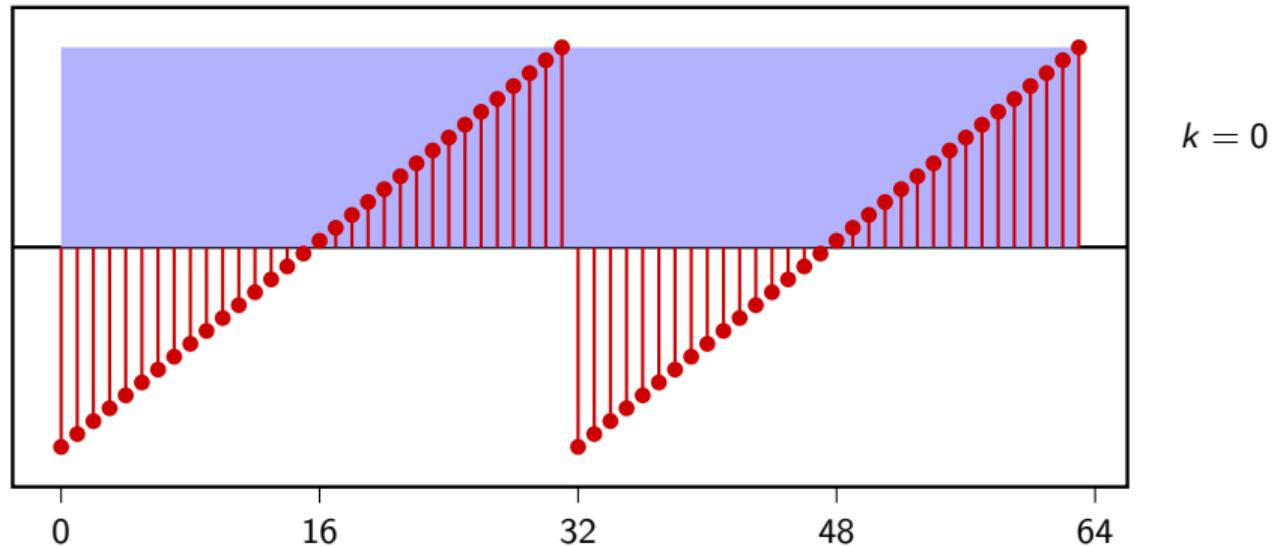
Example: 64-point DFT of two periods



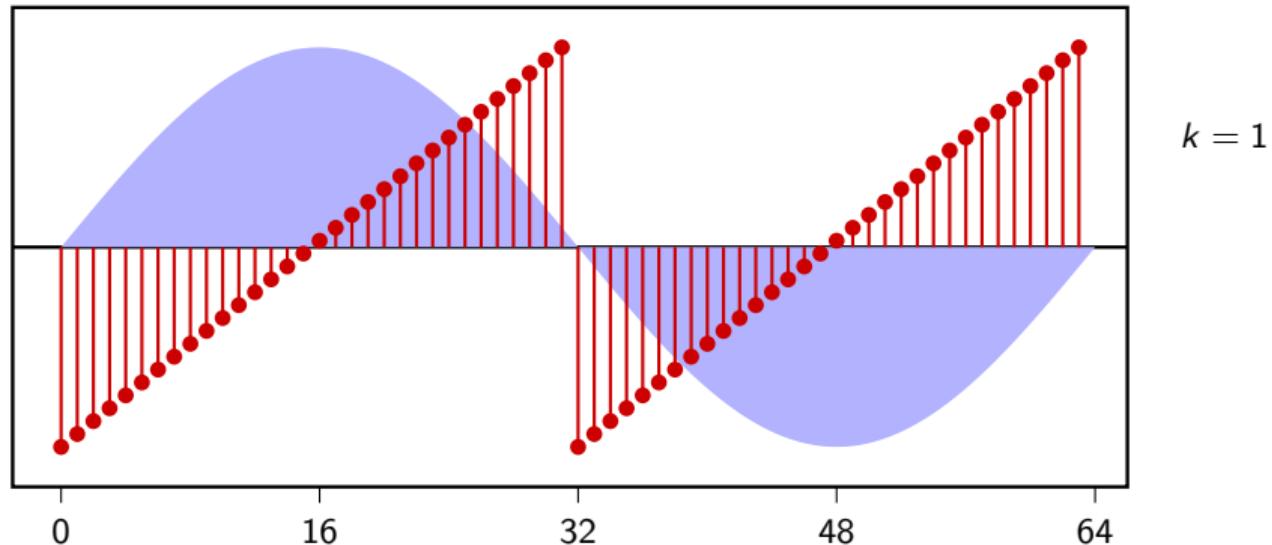
DFT of two periods: intuition



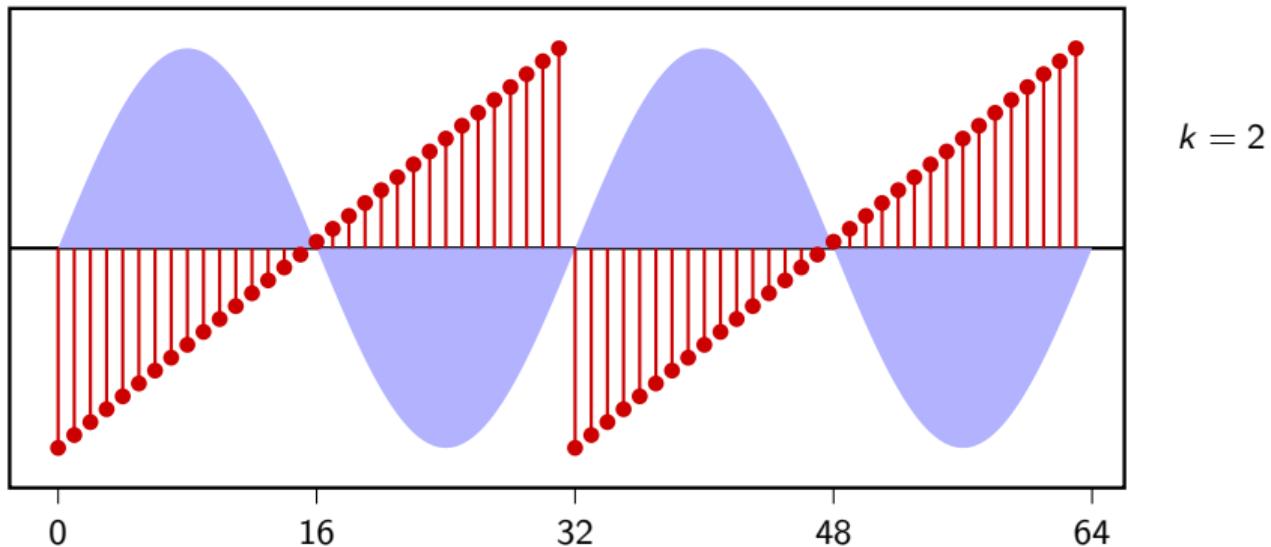
DFT of two periods: intuition



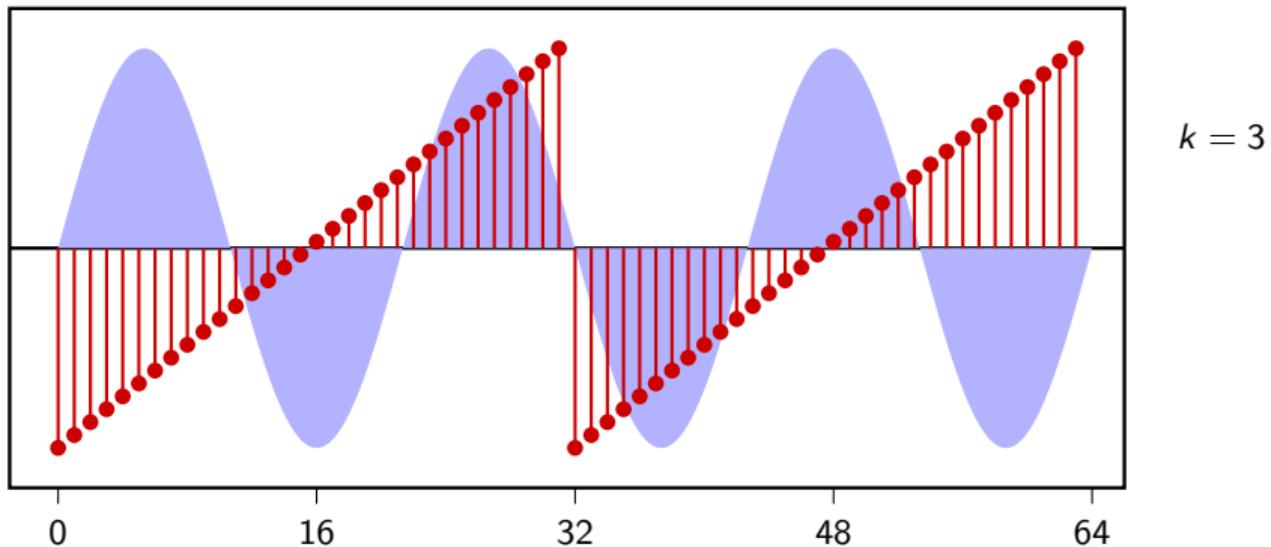
DFT of two periods: intuition



DFT of two periods: intuition



DFT of two periods: intuition



DFT of L periods

ingredients:

- ▶ finite-length signal $x[n]$, $n = 0, 1, \dots, N - 1$
- ▶ N -periodic signal: $\tilde{x}[n] = x[n \bmod N]$
- ▶ obviously $\tilde{x}[n] = \tilde{x}[n + pN]$ for all $p \in \mathbb{Z}$

DFT of L periods

$$\begin{aligned} X_L[k] &= \sum_{n=0}^{LN-1} \tilde{x}[n] e^{-j\frac{2\pi}{LN}nk} \quad k = 0, 1, 2, \dots, LN - 1 \\ &= \sum_{p=0}^{L-1} \sum_{n=0}^{N-1} \tilde{x}[n + pN] e^{-j\frac{2\pi}{LN}(n+pN)k} \\ &= \sum_{p=0}^{L-1} \sum_{n=0}^{N-1} \tilde{x}[n] e^{-j\frac{2\pi}{LN}nk} e^{-j\frac{2\pi}{L}pk} \\ &= \left(\sum_{p=0}^{L-1} e^{-j\frac{2\pi}{L}pk} \right) \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{LN}nk} \end{aligned}$$

DFT of L periods

$$\begin{aligned} X_L[k] &= \sum_{n=0}^{LN-1} \tilde{x}[n] e^{-j\frac{2\pi}{LN}nk} \quad k = 0, 1, 2, \dots, LN - 1 \\ &= \sum_{p=0}^{L-1} \sum_{n=0}^{N-1} \tilde{x}[n + pN] e^{-j\frac{2\pi}{LN}(n+pN)k} \\ &= \sum_{p=0}^{L-1} \sum_{n=0}^{N-1} \tilde{x}[n] e^{-j\frac{2\pi}{LN}nk} e^{-j\frac{2\pi}{L}pk} \\ &= \left(\sum_{p=0}^{L-1} e^{-j\frac{2\pi}{L}pk} \right) \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{LN}nk} \end{aligned}$$

DFT of L periods

$$\begin{aligned} X_L[k] &= \sum_{n=0}^{LN-1} \tilde{x}[n] e^{-j\frac{2\pi}{LN}nk} \quad k = 0, 1, 2, \dots, LN - 1 \\ &= \sum_{p=0}^{L-1} \sum_{n=0}^{N-1} \tilde{x}[n + pN] e^{-j\frac{2\pi}{LN}(n+pN)k} \\ &= \sum_{p=0}^{L-1} \sum_{n=0}^{N-1} \tilde{x}[n] e^{-j\frac{2\pi}{LN}nk} e^{-j\frac{2\pi}{L}pk} \\ &= \left(\sum_{p=0}^{L-1} e^{-j\frac{2\pi}{L}pk} \right) \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{LN}nk} \end{aligned}$$

DFT of L periods

$$\begin{aligned} X_L[k] &= \sum_{n=0}^{LN-1} \tilde{x}[n] e^{-j\frac{2\pi}{LN}nk} \quad k = 0, 1, 2, \dots, LN - 1 \\ &= \sum_{p=0}^{L-1} \sum_{n=0}^{N-1} \tilde{x}[n + pN] e^{-j\frac{2\pi}{LN}(n+pN)k} \\ &= \sum_{p=0}^{L-1} \sum_{n=0}^{N-1} \tilde{x}[n] e^{-j\frac{2\pi}{LN}nk} e^{-j\frac{2\pi}{L}pk} \\ &= \left(\sum_{p=0}^{L-1} e^{-j\frac{2\pi}{L}pk} \right) \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{LN}nk} \end{aligned}$$

We've seen this before

$$\sum_{p=0}^{L-1} e^{-j\frac{2\pi}{L}pk} = \begin{cases} L & \text{if } k \text{ multiple of } L \\ 0 & \text{otherwise} \end{cases}$$

(remember the orthogonality proof for the DFT basis)

DFT of L periods

if k is a multiple of L then k/L is an integer, so:

$$\sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi}{N} n \frac{k}{L}} = X[k/L]$$

DFT of L periods

$$X_L[k] = \begin{cases} L X[k/L] & \text{if } k = 0, L, 2L, 3L, \dots \\ 0 & \text{otherwise} \end{cases}$$

DFT and DFS

- ▶ again, all the spectral information for a periodic signal is contained in the DFT coefficients of a single period
- ▶ to stress the periodicity of the underlying signal, we use the term DFS

DFT and DFS

- ▶ again, all the spectral information for a periodic signal is contained in the DFT coefficients of a single period
- ▶ to stress the periodicity of the underlying signal, we use the term DFS

Finite-length time shifts revisited

The DFS helps us understand how to define time shifts for finite-length signals.

For an N - periodic sequence $\tilde{x}[n]$:

- ▶ $\tilde{x}[n - M]$ is well-defined for all $M \in \mathbb{N}$
- ▶ $\text{DFS}\{\tilde{x}[n - M]\} = e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k]$ (easy derivation)
- ▶ $\text{IDFS}\left\{ \tilde{X}[k] \right\} = \tilde{x}[n - M]$

Finite-length time shifts revisited

The DFS helps us understand how to define time shifts for finite-length signals.

For an N - periodic sequence $\tilde{x}[n]$:

- ▶ $\tilde{x}[n - M]$ is well-defined for all $M \in \mathbb{N}$
- ▶ $\text{DFS}\{\tilde{x}[n - M]\} = e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k]$ (easy derivation)
- ▶ $\text{IDFS}\left\{ \tilde{X}[k] \right\} = \tilde{x}[n - M]$

Finite-length time shifts revisited

The DFS helps us understand how to define time shifts for finite-length signals.

For an N - periodic sequence $\tilde{x}[n]$:

- ▶ $\tilde{x}[n - M]$ is well-defined for all $M \in \mathbb{N}$
- ▶ $\text{DFS}\{\tilde{x}[n - M]\} = e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k]$ (easy derivation)
- ▶ $\text{IDFS}\left\{ \tilde{X}[k] \right\} = \tilde{x}[n - M]$

Finite-length time shifts revisited

The DFS helps us understand how to define time shifts for finite-length signals.

For an N - periodic sequence $\tilde{x}[n]$:

- ▶ $\tilde{x}[n - M]$ is well-defined for all $M \in \mathbb{N}$
- ▶ $\text{DFS}\{\tilde{x}[n - M]\} = e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k]$ (easy derivation)
- ▶ $\text{IDFS}\left\{e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k]\right\} = \tilde{x}[n - M]$

Finite-length time shifts revisited

The DFS helps us understand how to define time shifts for finite-length signals.

For an N - periodic sequence $\tilde{x}[n]$:

- ▶ $\tilde{x}[n - M]$ is well-defined for all $M \in \mathbb{N}$
- ▶ DFS $\{\tilde{x}[n - M]\} = e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k]$ (easy derivation)
- ▶ IDFS $\left\{ e^{-j\frac{2\pi}{N}Mk} \tilde{X}[k] \right\} = \tilde{x}[n - M]$



a delay in time becomes a *linear phase factor* in frequency

Finite-length time shifts revisited

For an N -point signal $x[n]$:

- ▶ $x[n - M]$ is *not* well-defined
- ▶ what is IDFT $\left\{ e^{-j \frac{2\pi}{N} Mk} X[k] \right\}$?

Finite-length time shifts revisited

For an N -point signal $x[n]$:

- ▶ $x[n - M]$ is *not* well-defined
- ▶ what is IDFT $\left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\}$?

Finite-length time shifts revisited

$$\begin{aligned}\text{IDFT} \left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\} [n] &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{-j\frac{2\pi}{N}Mk} e^{j\frac{2\pi}{N}nk} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \left(\sum_{m=0}^{N-1} x[m] e^{-j\frac{2\pi}{N}mk} \right) e^{-j\frac{2\pi}{N}Mk} e^{j\frac{2\pi}{N}nk} \\ &= \frac{1}{N} \sum_{m=0}^{N-1} x[m] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k}\end{aligned}$$

Finite-length time shifts revisited

$$\begin{aligned}\text{IDFT} \left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\} [n] &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{-j\frac{2\pi}{N}Mk} e^{j\frac{2\pi}{N}nk} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \left(\sum_{m=0}^{N-1} x[m] e^{-j\frac{2\pi}{N}mk} \right) e^{-j\frac{2\pi}{N}Mk} e^{j\frac{2\pi}{N}nk} \\ &= \frac{1}{N} \sum_{m=0}^{N-1} x[m] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k}\end{aligned}$$

Finite-length time shifts revisited

$$\begin{aligned}\text{IDFT} \left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\} [n] &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{-j\frac{2\pi}{N}Mk} e^{j\frac{2\pi}{N}nk} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \left(\sum_{m=0}^{N-1} x[m] e^{-j\frac{2\pi}{N}mk} \right) e^{-j\frac{2\pi}{N}Mk} e^{j\frac{2\pi}{N}nk} \\ &= \frac{1}{N} \sum_{m=0}^{N-1} x[m] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k}\end{aligned}$$

We've seen something like this before...

$$\sum_{k=0}^{N-1} e^{-j\frac{2\pi}{N}nk} = \begin{cases} N & \text{if } n \text{ multiple of } N \\ 0 & \text{otherwise} \end{cases}$$

(remember the orthogonality proof for the DFT basis)

Finite-length time shifts revisited

$$\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}((n-M)-m)k} = \begin{cases} N & \text{for } ((n - M) - m) \text{ multiple of } N \\ 0 & \text{otherwise} \end{cases}$$

- ▶ n, M are fixed
- ▶ m goes from 0 to $N - 1$
- ▶ is there always a value for m that makes $((n - M) - m)$ a multiple of N ?

Finite-length time shifts revisited

$$\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}((n-M)-m)k} = \begin{cases} N & \text{for } ((n - M) - m) \text{ multiple of } N \\ 0 & \text{otherwise} \end{cases}$$

- ▶ n, M are fixed
- ▶ m goes from 0 to $N - 1$
- ▶ is there always a value for m that makes $((n - M) - m)$ a multiple of N ?

Modulo operator

given $C \in \mathbb{N}$, find m such that $0 \leq m < N$ and $C - m$ is a multiple of N

any integer C can be written as $C = pN + (C \bmod N)$, $p \in \mathbb{N}$:

- ▶ $0 \leq (C \bmod N) < N$
- ▶ $C - (C \bmod N)$ is a multiple of N

Modulo operator

given $C \in \mathbb{N}$, find m such that $0 \leq m < N$ and $C - m$ is a multiple of N

any integer C can be written as $C = pN + (C \bmod N)$, $p \in \mathbb{N}$:

- ▶ $0 \leq (C \bmod N) < N$
- ▶ $C - (C \bmod N)$ is a multiple of N

Finite-length time shifts revisited

$$\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k} = \begin{cases} N & \text{for } m = (n - M) \mod N \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{IDFT} \left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\} [n] &= \frac{1}{N} \sum_{m=0}^{N-1} x[m] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k} \\ &= x[(n - M) \mod N] \end{aligned}$$

shifts for finite-length signals are “naturally” circular

Finite-length time shifts revisited

$$\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k} = \begin{cases} N & \text{for } m = (n - M) \bmod N \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{IDFT} \left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\} [n] &= \frac{1}{N} \sum_{m=0}^{N-1} x[m] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k} \\ &= x[(n - M) \bmod N] \end{aligned}$$

shifts for finite-length signals are “naturally” circular

Finite-length time shifts revisited

$$\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k} = \begin{cases} N & \text{for } m = (n - M) \bmod N \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{IDFT} \left\{ e^{-j\frac{2\pi}{N}Mk} X[k] \right\} [n] &= \frac{1}{N} \sum_{m=0}^{N-1} x[m] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}(n-M-m)k} \\ &= x[(n - M) \bmod N] \end{aligned}$$

shifts for finite-length signals are “naturally” circular

The situation so far

Fourier representation for signal classes:

- ▶ N -point finite-length: DFT
- ▶ N -point periodic: DFS
- ▶ infinite length: ?

The situation so far

Fourier representation for signal classes:

- ▶ N -point finite-length: DFT
- ▶ N -point periodic: DFS
- ▶ infinite length: ?

The situation so far

Fourier representation for signal classes:

- ▶ N -point finite-length: DFT
- ▶ N -point periodic: DFS
- ▶ infinite length: ?

the Fourier transform for infinite-length signals

DFT of increasingly long signals

- ▶ Start with the DFT. What happens when $N \rightarrow \infty$?

- ▶ $\frac{2\pi}{N}k$ becomes denser in $[0, 2\pi]$...

- ▶ In the limit $\frac{2\pi}{N}k \rightarrow \omega$:

$$\sum_n x[n] e^{-j\omega n} \quad \omega \in \mathbb{R}$$

DFT of increasingly long signals

- ▶ Start with the DFT. What happens when $N \rightarrow \infty$?
- ▶ $\frac{2\pi}{N}k$ becomes denser in $[0, 2\pi]$...
- ▶ In the limit $\frac{2\pi}{N}k \rightarrow \omega$:

$$\sum_n x[n] e^{-j\omega n} \quad \omega \in \mathbb{R}$$

DFT of increasingly long signals

- ▶ Start with the DFT. What happens when $N \rightarrow \infty$?
- ▶ $\frac{2\pi}{N}k$ becomes denser in $[0, 2\pi] \dots$
- ▶ In the limit $\frac{2\pi}{N}k \rightarrow \omega$:

$$\sum_n x[n] e^{-j\omega n} \quad \omega \in \mathbb{R}$$

Discrete-Time Fourier Transform (DTFT)

Formal definition:

- ▶ $x[n] \in \ell_2(\mathbb{Z})$
- ▶ define the *function* of $\omega \in \mathbb{R}$

$$F(\omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$$

- ▶ inversion (when $F(\omega)$ exists):

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(\omega) e^{j\omega n} d\omega, \quad n \in \mathbb{Z}$$

Discrete-Time Fourier Transform (DTFT)

Formal definition:

- ▶ $x[n] \in \ell_2(\mathbb{Z})$
- ▶ define the *function* of $\omega \in \mathbb{R}$

$$F(\omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$$

- ▶ inversion (when $F(\omega)$ exists):

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(\omega) e^{j\omega n} d\omega, \quad n \in \mathbb{Z}$$

Discrete-Time Fourier Transform (DTFT)

Formal definition:

- ▶ $x[n] \in \ell_2(\mathbb{Z})$
- ▶ define the *function* of $\omega \in \mathbb{R}$

$$F(\omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$$

- ▶ inversion (when $F(\omega)$ exists):

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(\omega) e^{j\omega n} d\omega, \quad n \in \mathbb{Z}$$

DTFT periodicity and notation

- ▶ $e^{j\omega n} = e^{j(\omega+2k\pi)n} \quad \forall k \in \mathbb{N}$
- ▶ $F(\omega)$ is 2π -periodic
- ▶ to stress periodicity (and for other reasons) we will write

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

- ▶ by convention, $X(e^{j\omega})$ is represented over $[-\pi, \pi]$

DTFT periodicity and notation

- ▶ $e^{j\omega n} = e^{j(\omega+2k\pi)n} \quad \forall k \in \mathbb{N}$
- ▶ $F(\omega)$ is 2π -periodic
- ▶ to stress periodicity (and for other reasons) we will write

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

- ▶ by convention, $X(e^{j\omega})$ is represented over $[-\pi, \pi]$

DTFT periodicity and notation

- ▶ $e^{j\omega n} = e^{j(\omega+2k\pi)n} \quad \forall k \in \mathbb{N}$
- ▶ $F(\omega)$ is 2π -periodic
- ▶ to stress periodicity (and for other reasons) we will write

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

- ▶ by convention, $X(e^{j\omega})$ is represented over $[-\pi, \pi]$

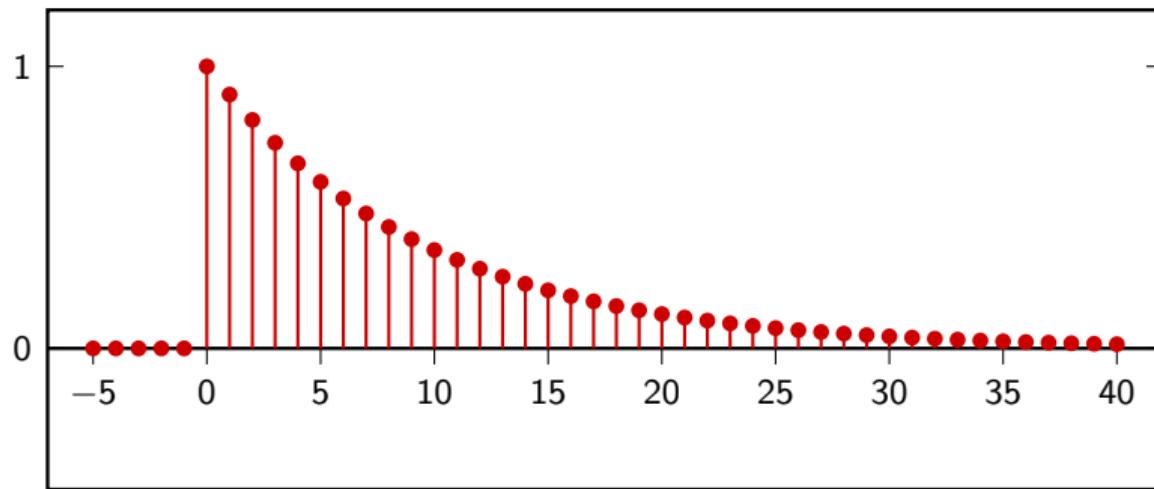
DTFT periodicity and notation

- ▶ $e^{j\omega n} = e^{j(\omega+2k\pi)n} \quad \forall k \in \mathbb{N}$
- ▶ $F(\omega)$ is 2π -periodic
- ▶ to stress periodicity (and for other reasons) we will write

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

- ▶ by convention, $X(e^{j\omega})$ is represented over $[-\pi, \pi]$

$$x[n] = \alpha^n u[n], \quad |\alpha| < 1$$



DTFT of $x[n] = \alpha^n u[n]$, $|\alpha| < 1$

$$\begin{aligned} X(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \\ &= \sum_{n=0}^{\infty} \alpha^n e^{-j\omega n} \\ &= \sum_{n=0}^{\infty} (\alpha e^{-j\omega})^n \\ &= \frac{1}{1 - \alpha e^{-j\omega}} \end{aligned}$$

DTFT of $x[n] = \alpha^n u[n]$, $|\alpha| < 1$

$$\begin{aligned} X(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \\ &= \sum_{n=0}^{\infty} \alpha^n e^{-j\omega n} \\ &= \sum_{n=0}^{\infty} (\alpha e^{-j\omega})^n \\ &= \frac{1}{1 - \alpha e^{-j\omega}} \end{aligned}$$

DTFT of $x[n] = \alpha^n u[n]$, $|\alpha| < 1$

$$\begin{aligned} X(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \\ &= \sum_{n=0}^{\infty} \alpha^n e^{-j\omega n} \\ &= \sum_{n=0}^{\infty} (\alpha e^{-j\omega})^n \\ &= \frac{1}{1 - \alpha e^{-j\omega}} \end{aligned}$$

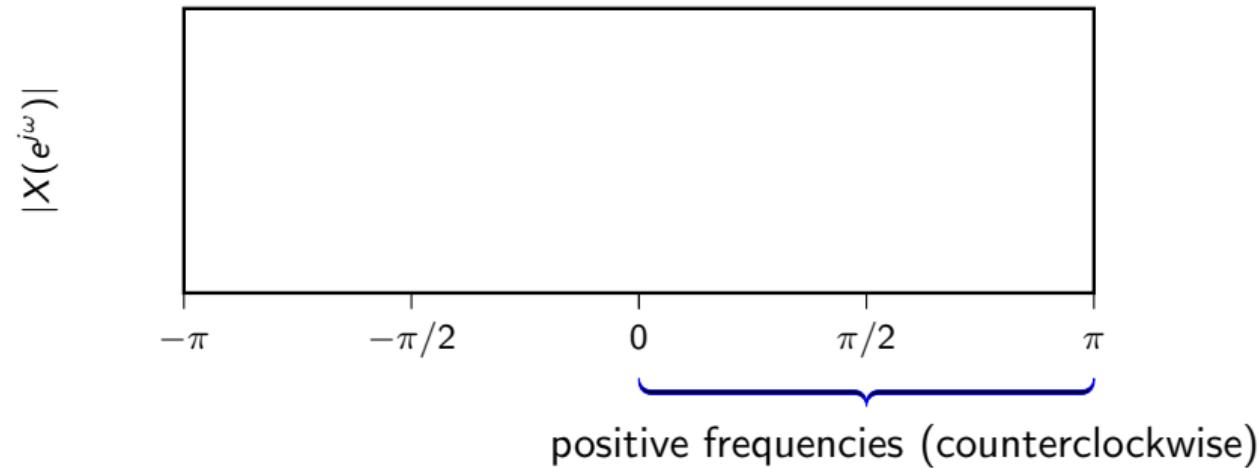
DTFT of $x[n] = \alpha^n u[n]$, $|\alpha| < 1$

$$\begin{aligned} X(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \\ &= \sum_{n=0}^{\infty} \alpha^n e^{-j\omega n} \\ &= \sum_{n=0}^{\infty} (\alpha e^{-j\omega})^n \\ &= \frac{1}{1 - \alpha e^{-j\omega}} \end{aligned}$$

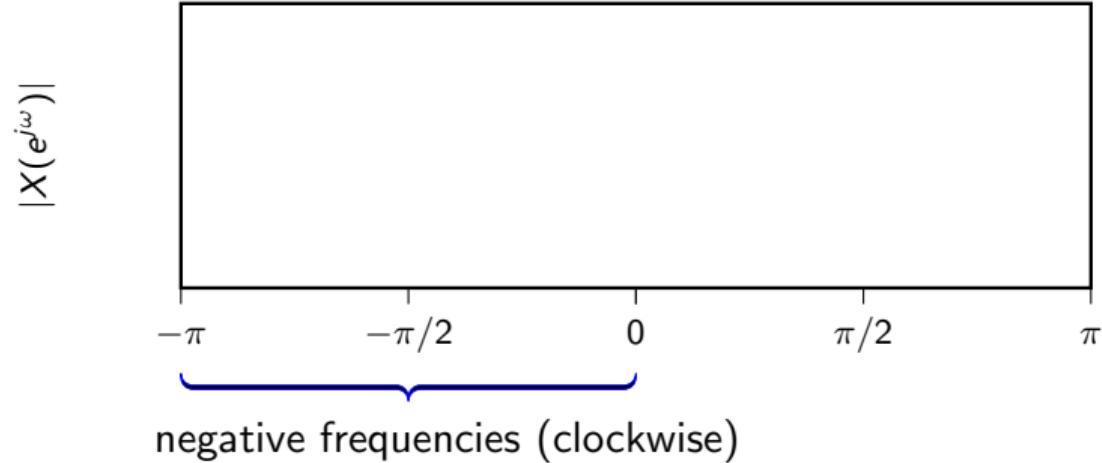
DTFT of $x[n] = \alpha^n u[n]$, $|\alpha| < 1$

$$|X(e^{j\omega})|^2 = \frac{1}{1 + \alpha^2 - 2\alpha \cos \omega}$$

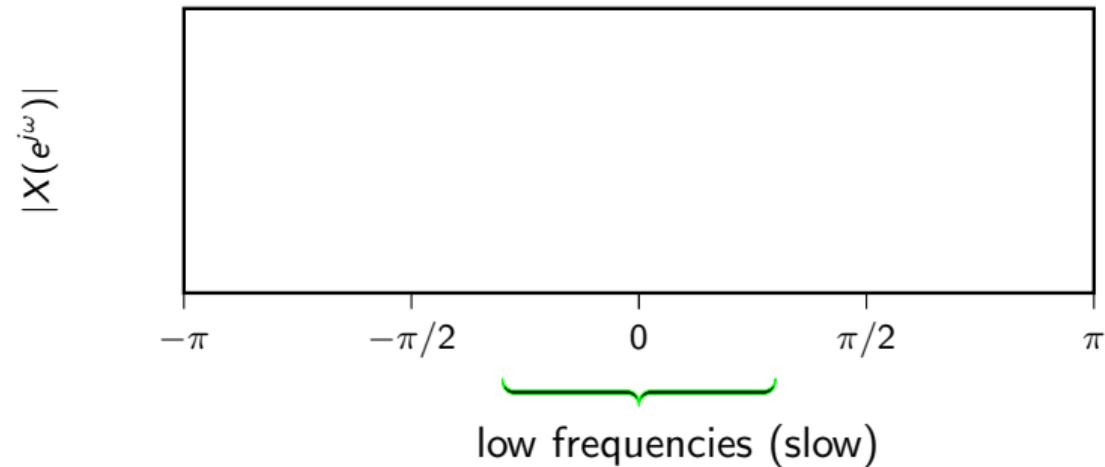
Plotting the DTFT



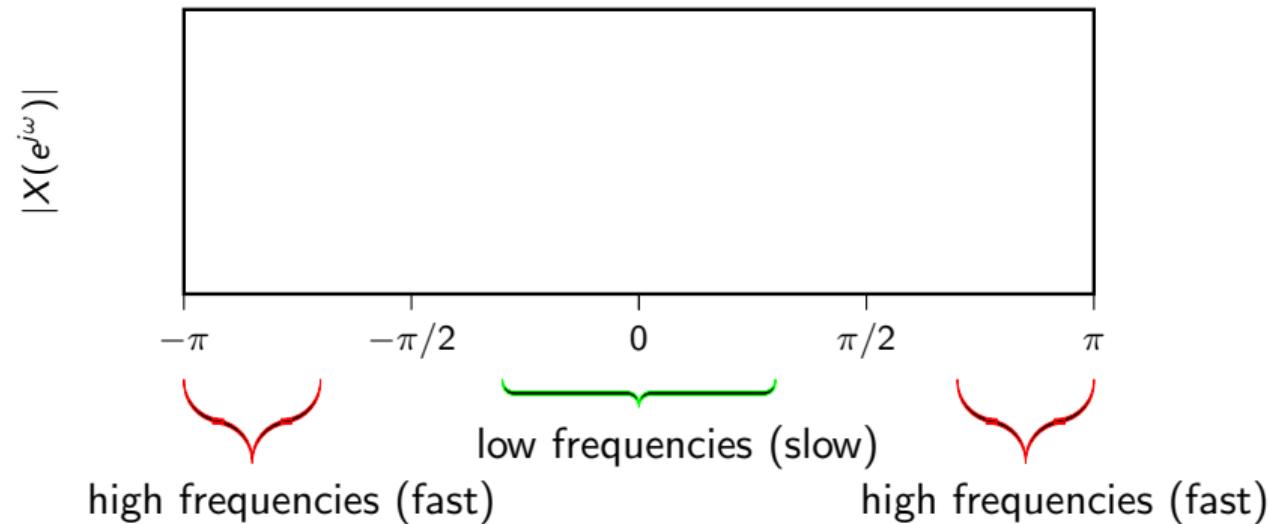
Plotting the DTFT



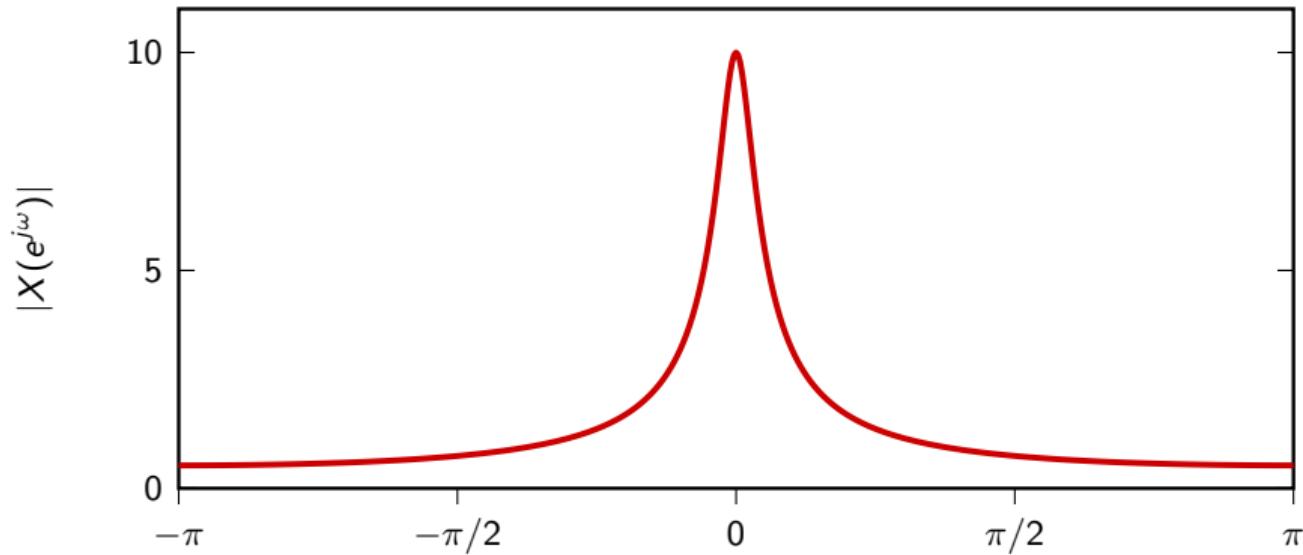
Plotting the DTFT



Plotting the DTFT

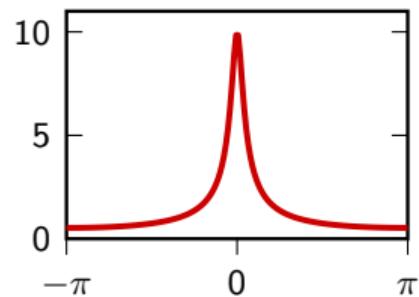


DTFT of $x[n] = \alpha^n u[n]$, $|\alpha| < 1$

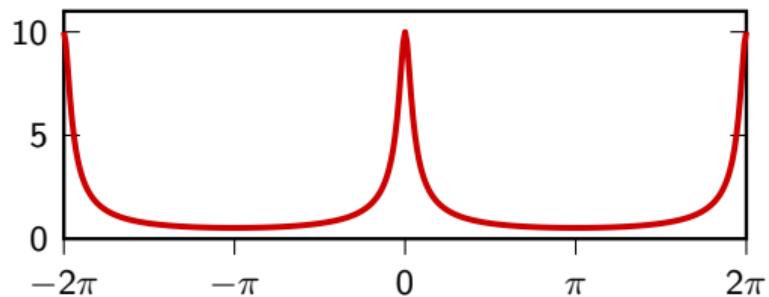


$$\alpha = 0.9$$

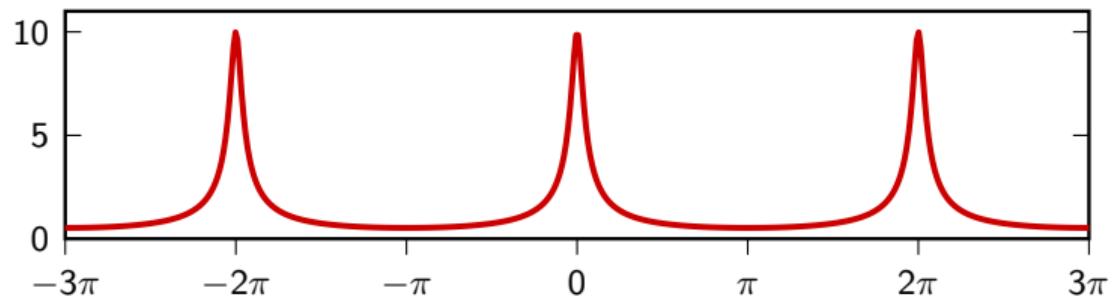
Remember the periodicity!



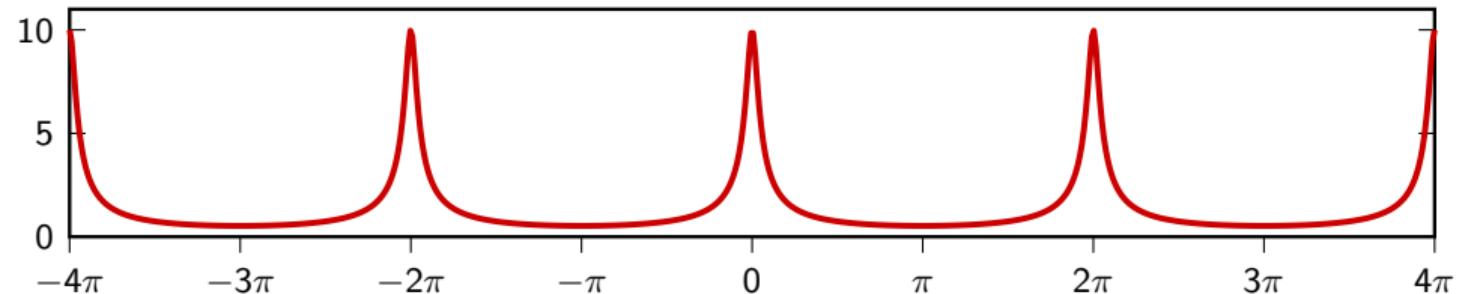
Remember the periodicity!



Remember the periodicity!



Remember the periodicity!



DTFT intuition and properties

Overview:

- ▶ DTFT Existence
- ▶ Properties
- ▶ DTFT as basis expansion

Discrete-Time Fourier Transform

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

- ▶ when does it exist?
- ▶ is it a change of basis?

Discrete-Time Fourier Transform

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

- ▶ when does it exist?
- ▶ is it a change of basis?

Existence easy for absolutely summable sequences

$$\begin{aligned}|X(e^{j\omega})| &= \left| \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \right| \\&\leq \sum_{n=-\infty}^{\infty} |x[n] e^{-j\omega n}| \\&= \sum_{n=-\infty}^{\infty} |x[n]| \\&< \infty\end{aligned}$$

Existence easy for absolutely summable sequences

$$\begin{aligned}|X(e^{j\omega})| &= \left| \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \right| \\&\leq \sum_{n=-\infty}^{\infty} |x[n] e^{-j\omega n}| \\&= \sum_{n=-\infty}^{\infty} |x[n]| \\&< \infty\end{aligned}$$

Existence easy for absolutely summable sequences

$$\begin{aligned}|X(e^{j\omega})| &= \left| \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \right| \\&\leq \sum_{n=-\infty}^{\infty} |x[n] e^{-j\omega n}| \\&= \sum_{n=-\infty}^{\infty} |x[n]| \\&< \infty\end{aligned}$$

Existence easy for absolutely summable sequences

$$\begin{aligned}|X(e^{j\omega})| &= \left| \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \right| \\&\leq \sum_{n=-\infty}^{\infty} |x[n] e^{-j\omega n}| \\&= \sum_{n=-\infty}^{\infty} |x[n]| \\&< \infty\end{aligned}$$

Inversion easy for absolutely summable sequences

$$\begin{aligned}\frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \left(\sum_{k=-\infty}^{\infty} x[k] e^{-j\omega k} \right) e^{j\omega n} d\omega \\ &= \sum_{k=-\infty}^{\infty} x[k] \int_{-\pi}^{\pi} \frac{e^{j\omega(n-k)}}{2\pi} d\omega \\ &= x[n]\end{aligned}$$

Inversion easy for absolutely summable sequences

$$\begin{aligned}\frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \left(\sum_{k=-\infty}^{\infty} x[k] e^{-j\omega k} \right) e^{j\omega n} d\omega \\ &= \sum_{k=-\infty}^{\infty} x[k] \int_{-\pi}^{\pi} \frac{e^{j\omega(n-k)}}{2\pi} d\omega \\ &= x[n]\end{aligned}$$

Inversion easy for absolutely summable sequences

$$\begin{aligned}\frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \left(\sum_{k=-\infty}^{\infty} x[k] e^{-j\omega k} \right) e^{j\omega n} d\omega \\ &= \sum_{k=-\infty}^{\infty} x[k] \int_{-\pi}^{\pi} \frac{e^{j\omega(n-k)}}{2\pi} d\omega \\ &= x[n]\end{aligned}$$

Inversion easy for absolutely summable sequences

$$\int_{-\pi}^{\pi} \frac{e^{j\omega m}}{2\pi} d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} d\omega = 1 \quad \text{for } m = 0$$

$$= \frac{1}{2\pi} \frac{1}{jm} e^{j\omega m} \Big|_{-\pi}^{\pi}$$

$$= \frac{1}{2\pi} \frac{1}{jm} (e^{j\pi m} - e^{-j\pi m}) = 0 \quad \text{otherwise}$$

the DTFT as the limit of the DFS

Synopsis

- ▶ $x[n]$ absolutely summable $\Rightarrow X(e^{j\omega})$ exists formally
- ▶ $x[n]$ absolutely summable \Rightarrow we can *periodize* it into $\tilde{x}_N[n]$
- ▶ natural Fourier representation for $\tilde{x}_N[n]$ is DFS
- ▶ DFS of $\tilde{x}_N[n]$ turns out to be $X(e^{j\omega})$ at $\omega = (2\pi/N)k$
- ▶ as N grows to infinity $\tilde{x}_N[n]$ becomes $x[n]$
- ▶ as N grows to infinity natural Fourier representation becomes $X(e^{j\omega})$

Synopsis

- ▶ $x[n]$ absolutely summable $\Rightarrow X(e^{j\omega})$ exists formally
- ▶ $x[n]$ absolutely summable \Rightarrow we can *periodize* it into $\tilde{x}_N[n]$
- ▶ natural Fourier representation for $\tilde{x}_N[n]$ is DFS
- ▶ DFS of $\tilde{x}_N[n]$ turns out to be $X(e^{j\omega})$ at $\omega = (2\pi/N)k$
- ▶ as N grows to infinity $\tilde{x}_N[n]$ becomes $x[n]$
- ▶ as N grows to infinity natural Fourier representation becomes $X(e^{j\omega})$

Synopsis

- ▶ $x[n]$ absolutely summable $\Rightarrow X(e^{j\omega})$ exists formally
- ▶ $x[n]$ absolutely summable \Rightarrow we can *periodize* it into $\tilde{x}_N[n]$
- ▶ natural Fourier representation for $\tilde{x}_N[n]$ is DFS
- ▶ DFS of $\tilde{x}_N[n]$ turns out to be $X(e^{j\omega})$ at $\omega = (2\pi/N)k$
- ▶ as N grows to infinity $\tilde{x}_N[n]$ becomes $x[n]$
- ▶ as N grows to infinity natural Fourier representation becomes $X(e^{j\omega})$

Synopsis

- ▶ $x[n]$ absolutely summable $\Rightarrow X(e^{j\omega})$ exists formally
- ▶ $x[n]$ absolutely summable \Rightarrow we can *periodize* it into $\tilde{x}_N[n]$
- ▶ natural Fourier representation for $\tilde{x}_N[n]$ is DFS
- ▶ DFS of $\tilde{x}_N[n]$ turns out to be $X(e^{j\omega})$ at $\omega = (2\pi/N)k$
- ▶ as N grows to infinity $\tilde{x}_N[n]$ becomes $x[n]$
- ▶ as N grows to infinity natural Fourier representation becomes $X(e^{j\omega})$

Synopsis

- ▶ $x[n]$ absolutely summable $\Rightarrow X(e^{j\omega})$ exists formally
- ▶ $x[n]$ absolutely summable \Rightarrow we can *periodize* it into $\tilde{x}_N[n]$
- ▶ natural Fourier representation for $\tilde{x}_N[n]$ is DFS
- ▶ DFS of $\tilde{x}_N[n]$ turns out to be $X(e^{j\omega})$ at $\omega = (2\pi/N)k$
- ▶ as N grows to infinity $\tilde{x}_N[n]$ becomes $x[n]$
- ▶ as N grows to infinity natural Fourier representation becomes $X(e^{j\omega})$

Synopsis

- ▶ $x[n]$ absolutely summable $\Rightarrow X(e^{j\omega})$ exists formally
- ▶ $x[n]$ absolutely summable \Rightarrow we can *periodize* it into $\tilde{x}_N[n]$
- ▶ natural Fourier representation for $\tilde{x}_N[n]$ is DFS
- ▶ DFS of $\tilde{x}_N[n]$ turns out to be $X(e^{j\omega})$ at $\omega = (2\pi/N)k$
- ▶ as N grows to infinity $\tilde{x}_N[n]$ becomes $x[n]$
- ▶ as N grows to infinity natural Fourier representation becomes $X(e^{j\omega})$

Some intuition

With $x[n]$ absolutely summable we can build arbitrarily “periodized” sequences:

$$\tilde{x}_N[n] = \sum_{p=-\infty}^{\infty} x[n + pN]$$

clearly $\tilde{x}_N[n] = \tilde{x}_N[n + N]$

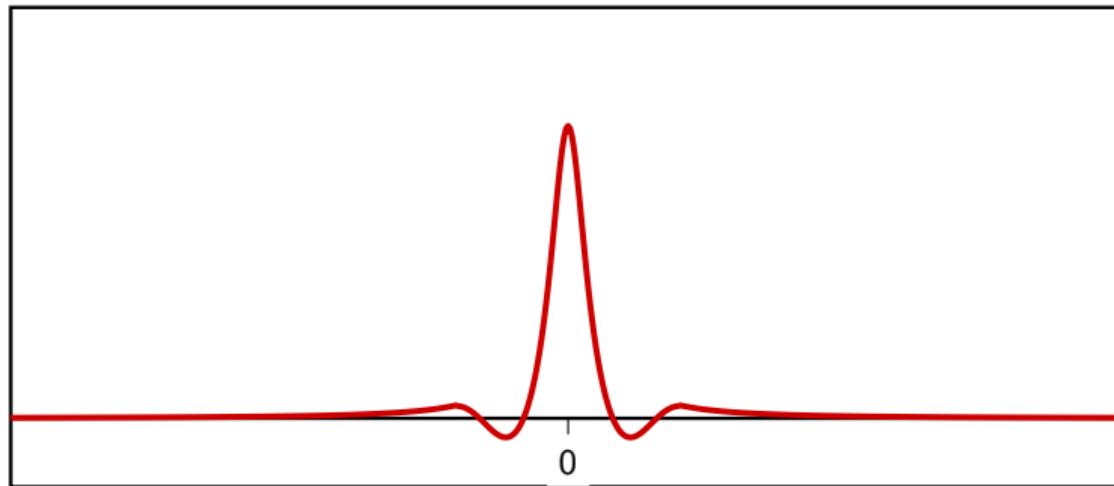
Some intuition

With $x[n]$ absolutely summable we can build arbitrarily “periodized” sequences:

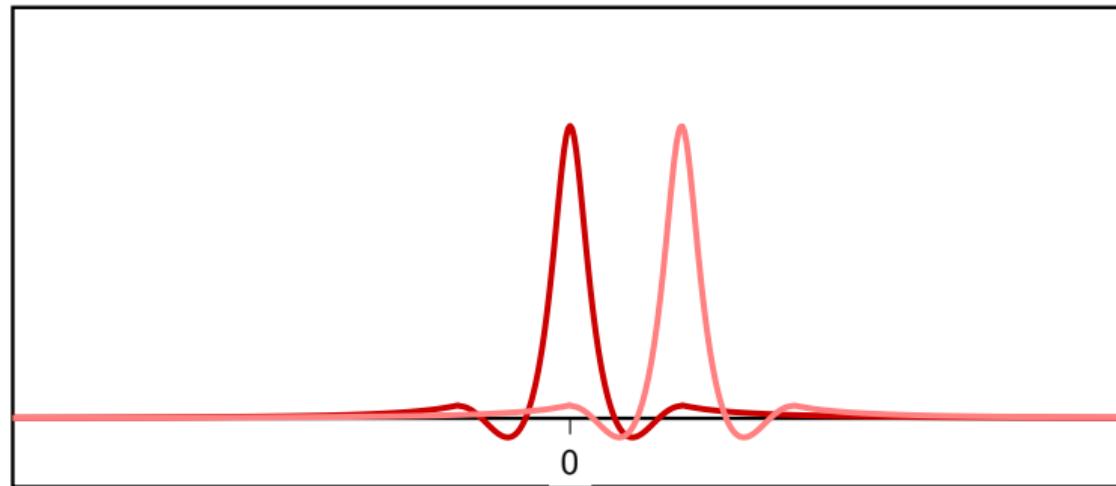
$$\tilde{x}_N[n] = \sum_{p=-\infty}^{\infty} x[n + pN]$$

clearly $\tilde{x}_N[n] = \tilde{x}_N[n + N]$

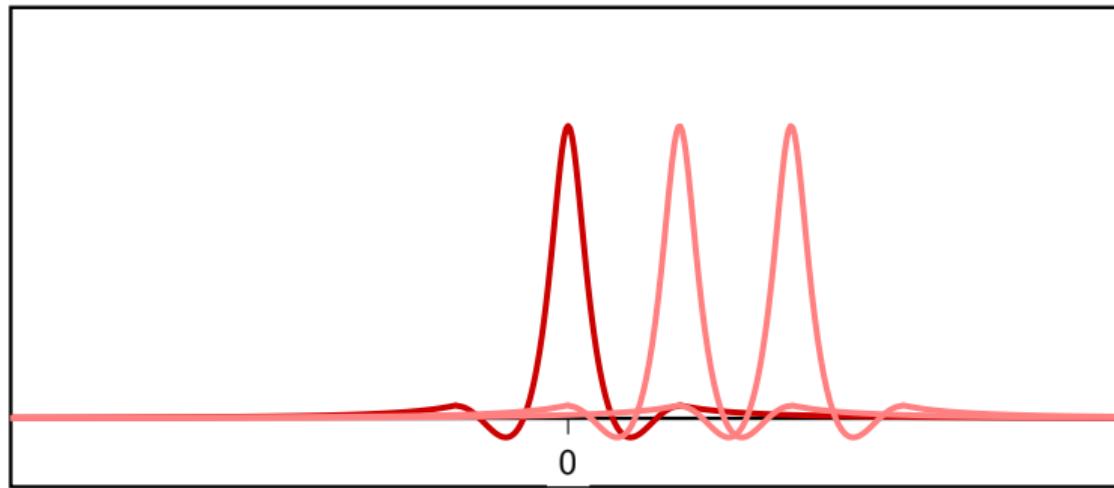
Periodization



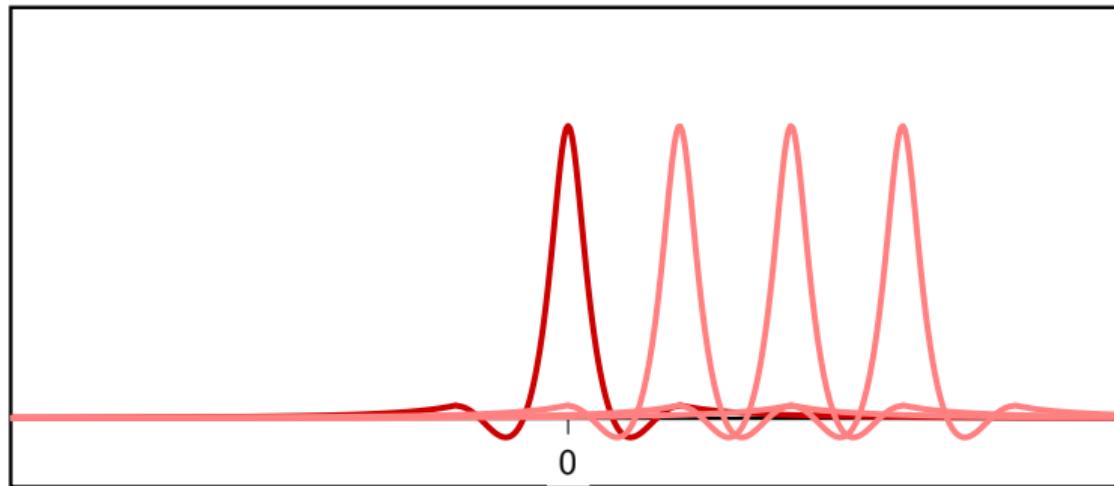
Periodization



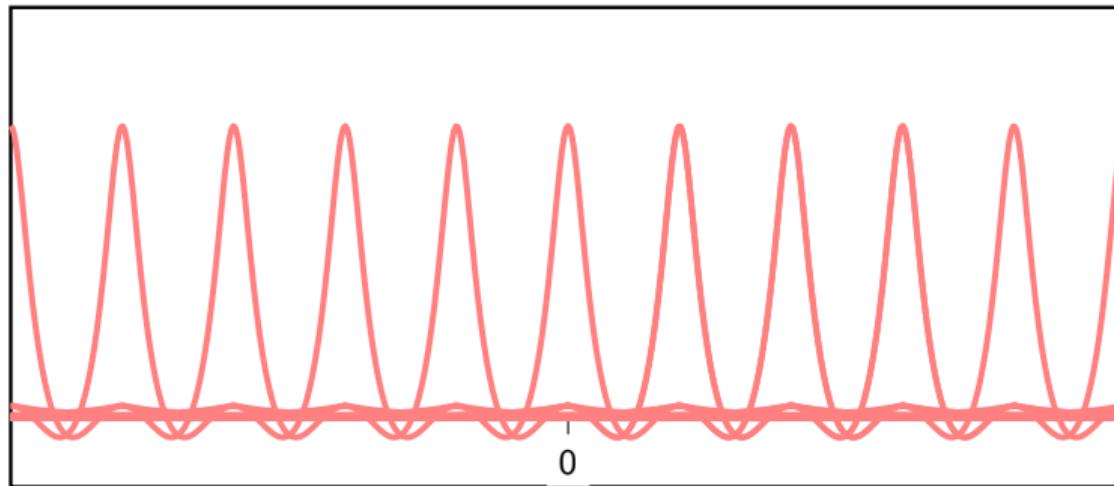
Periodization



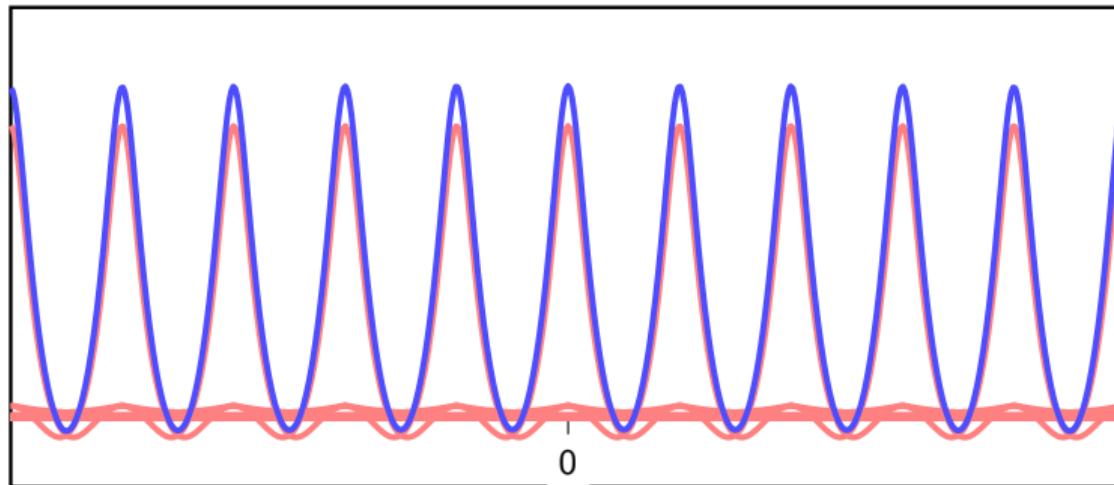
Periodization



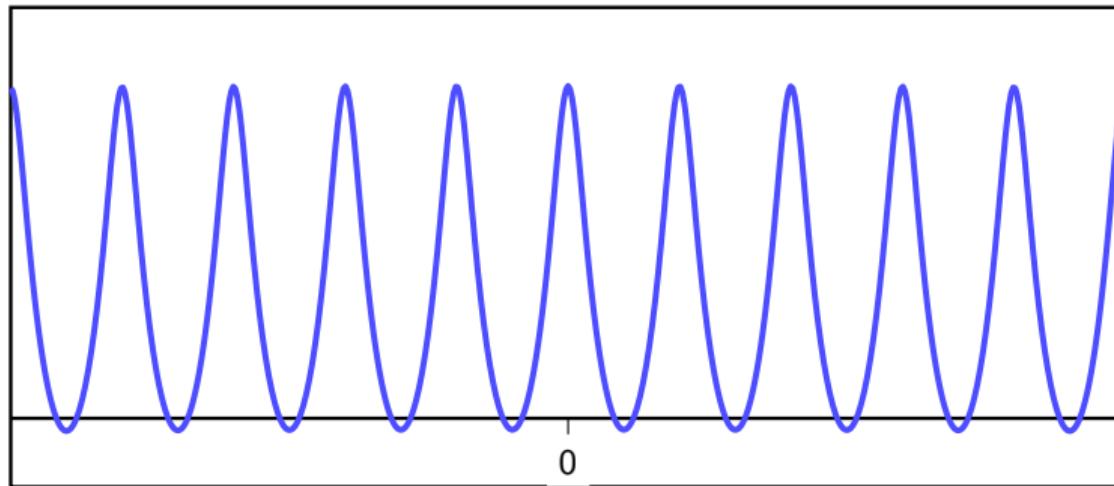
Periodization



Periodization

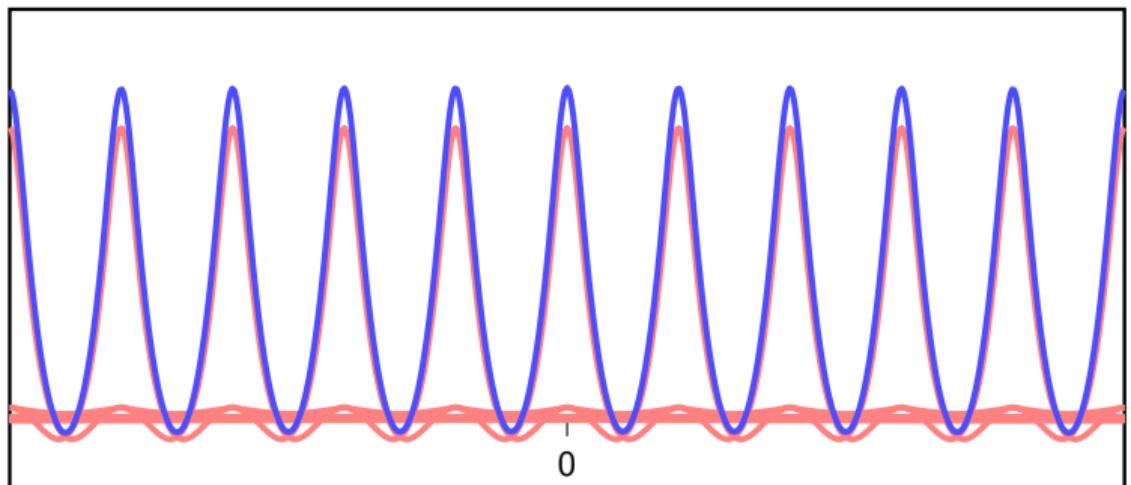


Periodization



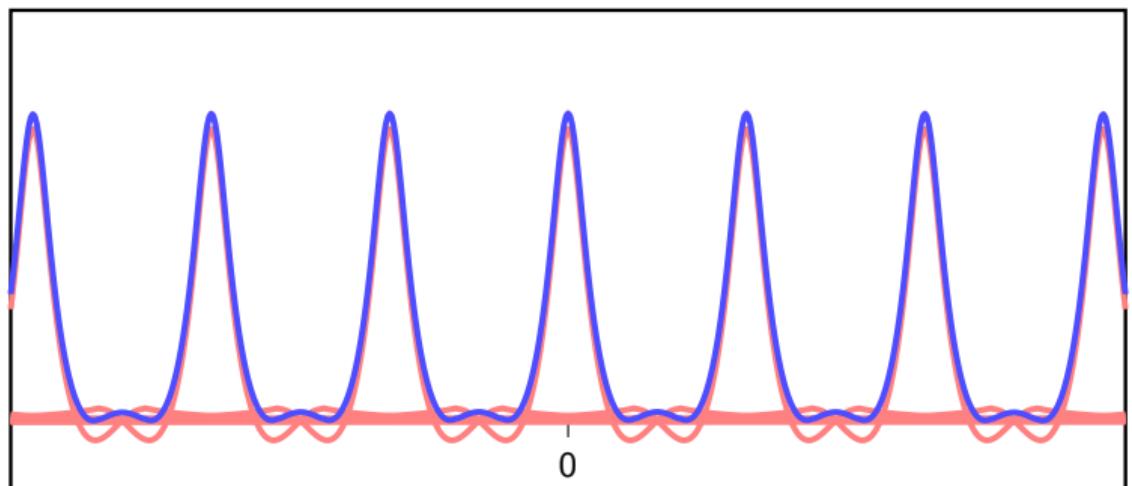
Periodization

Let N grow large...



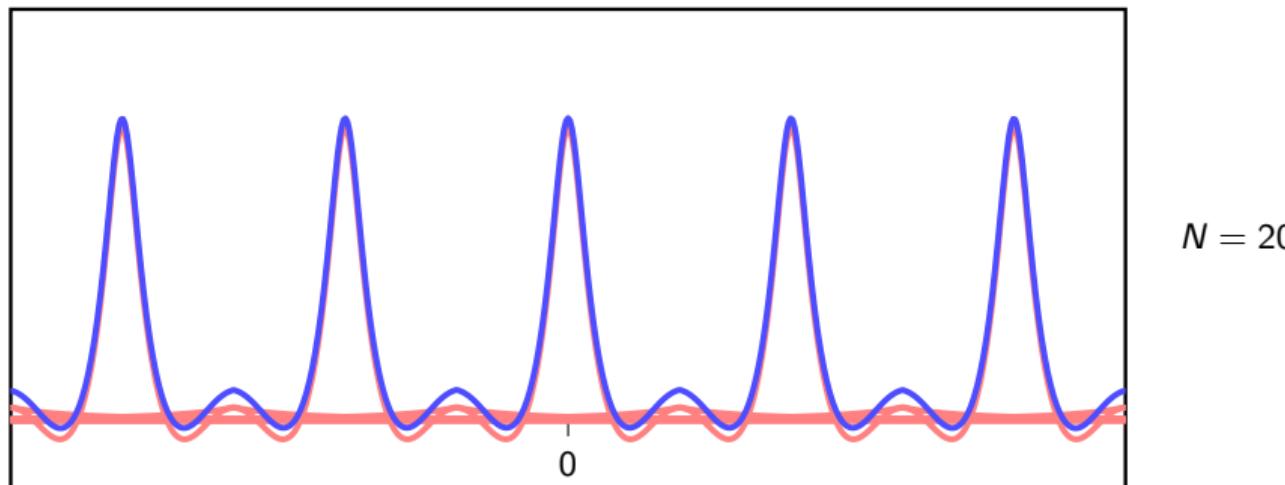
Periodization

Let N grow large...



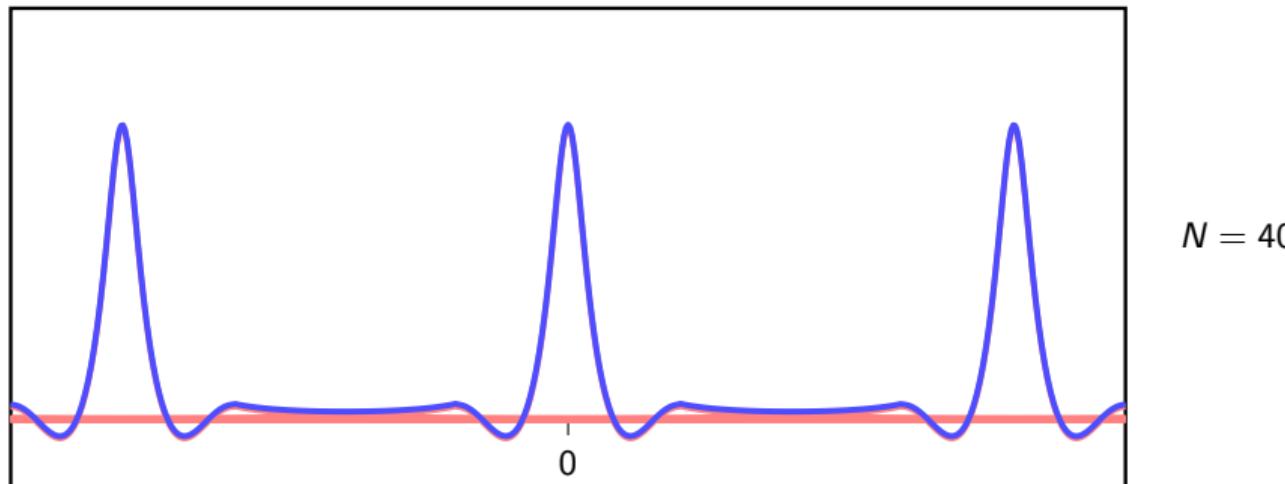
Periodization

Let N grow large...



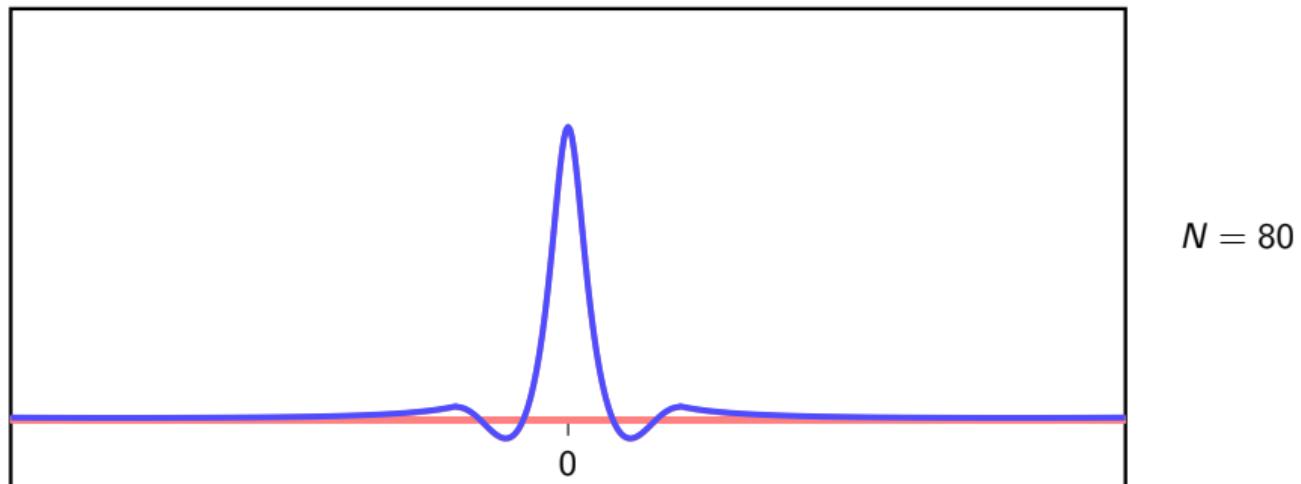
Periodization

Let N grow large...



Periodization

... as N grows, $\tilde{x}_N[n] \rightarrow x[n]$



From DFS to DTFT

Natural spectral representation for $\tilde{x}_N[n]$ is the DFS:

$$\begin{aligned}\tilde{X}[k] &= \sum_{n=0}^{N-1} \tilde{x}_N[n] e^{-j\frac{2\pi}{N}nk} \\ &= \sum_{n=0}^{N-1} \sum_{p=-\infty}^{\infty} x[n+pN] e^{-j\frac{2\pi}{N}nk} \\ &= \sum_{p=-\infty}^{\infty} \sum_{n=0}^{N-1} x[n+pN] e^{-j\frac{2\pi}{N}(n+pN)k} \\ &\quad (\text{remember } e^{j\alpha} = e^{j(\alpha+2K\pi)} \quad \forall K)\end{aligned}$$

From DFS to DTFT

Natural spectral representation for $\tilde{x}_N[n]$ is the DFS:

$$\begin{aligned}\tilde{X}[k] &= \sum_{n=0}^{N-1} \tilde{x}_N[n] e^{-j\frac{2\pi}{N}nk} \\ &= \sum_{n=0}^{N-1} \sum_{p=-\infty}^{\infty} x[n+pN] e^{-j\frac{2\pi}{N}nk} \\ &= \sum_{p=-\infty}^{\infty} \sum_{n=0}^{N-1} x[n+pN] e^{-j\frac{2\pi}{N}(n+pN)k} \\ &\quad (\text{remember } e^{j\alpha} = e^{j(\alpha+2K\pi)} \quad \forall K)\end{aligned}$$

From DFS to DTFT

Natural spectral representation for $\tilde{x}_N[n]$ is the DFS:

$$\begin{aligned}\tilde{X}[k] &= \sum_{n=0}^{N-1} \tilde{x}_N[n] e^{-j\frac{2\pi}{N}nk} \\ &= \sum_{n=0}^{N-1} \sum_{p=-\infty}^{\infty} x[n+pN] e^{-j\frac{2\pi}{N}nk} \\ &= \sum_{p=-\infty}^{\infty} \sum_{n=0}^{N-1} x[n+pN] e^{-j\frac{2\pi}{N}(n+pN)k} \\ &\quad (\text{remember } e^{j\alpha} = e^{j(\alpha+2K\pi)} \quad \forall K)\end{aligned}$$

From double sum to single sum

we can always write for all $N \in \mathbb{N}^+$

$$\sum_{m=-\infty}^{\infty} y[m] = \sum_{p=-\infty}^{\infty} \sum_{n=0}^{N-1} y[n + pN]$$

Example (N=4)

	n	0	1	2	3
p					
	...				
	-1				
	0				
	1				
	2				
	...				

$$m = n + 4p$$

Example (N=4)

n	0	1	2	3
\dots				
-1				
0	0			
1				
2				
\dots				

$$m = n + 4p$$

Example (N=4)

n	0	1	2	3
\dots				
-1				
0	0	1		
1				
2				
\dots				

$$m = n + 4p$$

Example (N=4)

	n			
	0	1	2	3
p	0	0	1	2
...				
-1				
...				
1				
2				
...				

$$m = n + 4p$$

Example (N=4)

n	0	1	2	3
\dots				
-1				
0	0	1	2	3
1				
2				
\dots				

$$m = n + 4p$$

Example (N=4)

n	0	1	2	3
\dots				
-1				
0	0	1	2	3
1	4			
2				
\dots				

$$m = n + 4p$$

Example (N=4)

	n			
	0	1	2	3
p	0	0	1	2
-1				
0	0	1	2	3
1	4	5		
2				
...				

$$m = n + 4p$$

Example (N=4)

n	0	1	2	3	
\dots					
-1					
p	0	0	1	2	3
	1	4	5	6	
	2				
\dots					

$$m = n + 4p$$

Example (N=4)

	n			
	0	1	2	3
\dots				
-1				
0	0	1	2	3
1	4	5	6	7
2				
\dots				

$$m = n + 4p$$

Example (N=4)

		m				
		0	1	2	3	
		-1	-4	-3	-2	-1
p		0	0	1	2	3
1		1	4	5	6	7
2		2	8	9	10	11
...						

$$n = 4p + m$$

From DFS to DTFT

$$\begin{aligned}\tilde{X}[k] &= \sum_{p=-\infty}^{\infty} \sum_{n=0}^{N-1} x[n + pN] e^{-j\frac{2\pi}{N}(n+pN)k} \\ &= \sum_{m=-\infty}^{\infty} x[m] e^{-j\frac{2\pi}{N}km} \\ &= X(e^{j\omega})|_{\omega=\frac{2\pi}{N}k}\end{aligned}$$

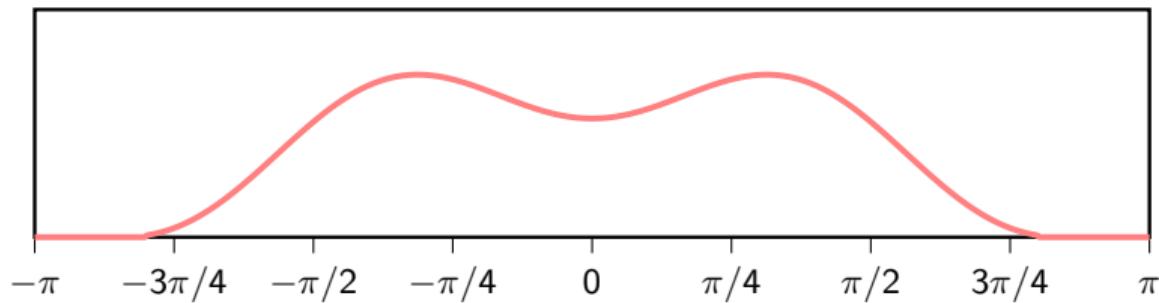
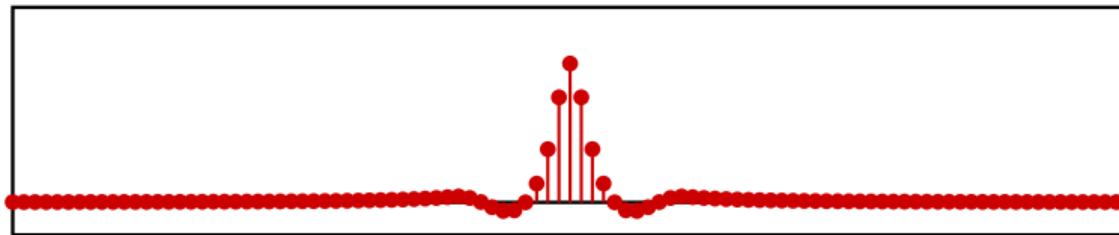
From DFS to DTFT

$$\begin{aligned}\tilde{X}[k] &= \sum_{p=-\infty}^{\infty} \sum_{n=0}^{N-1} x[n + pN] e^{-j\frac{2\pi}{N}(n+pN)k} \\ &= \sum_{m=-\infty}^{\infty} x[m] e^{-j\frac{2\pi}{N}km} \\ &= X(e^{j\omega})|_{\omega=\frac{2\pi}{N}k}\end{aligned}$$

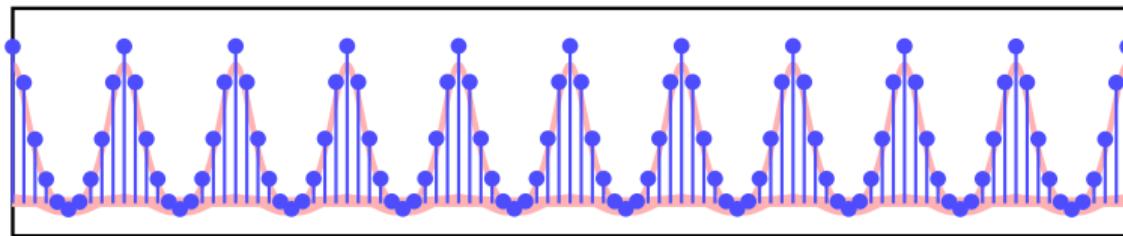
From DFS to DTFT

$$\begin{aligned}\tilde{X}[k] &= \sum_{p=-\infty}^{\infty} \sum_{n=0}^{N-1} x[n + pN] e^{-j\frac{2\pi}{N}(n+pN)k} \\ &= \sum_{m=-\infty}^{\infty} x[m] e^{-j\frac{2\pi}{N}km} \\ &= X(e^{j\omega})|_{\omega=\frac{2\pi}{N}k}\end{aligned}$$

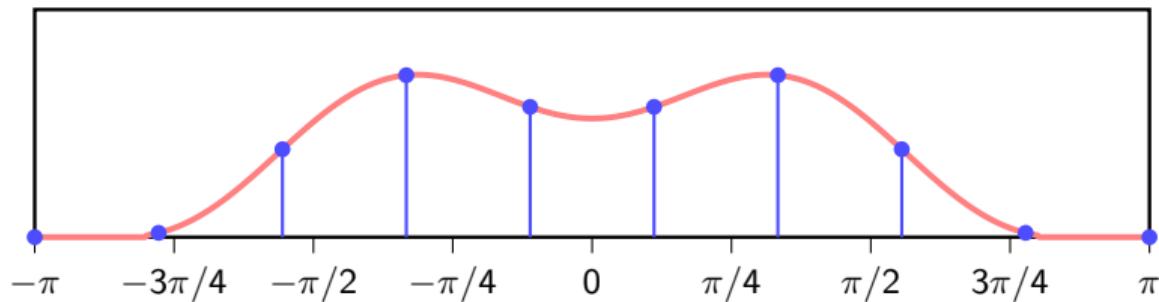
From DFS to DTFT



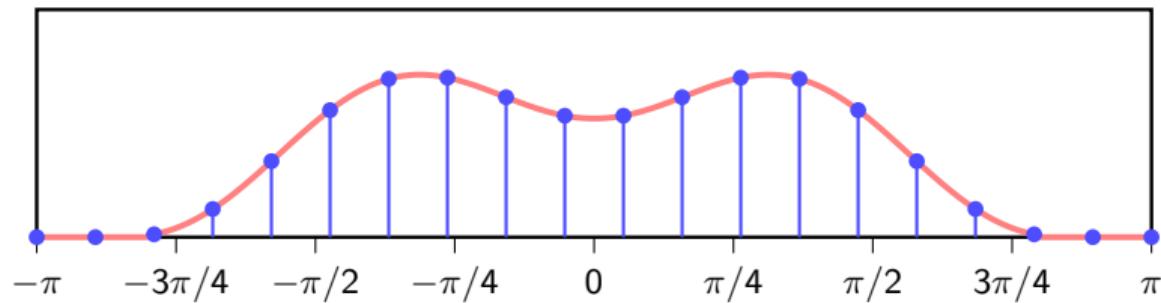
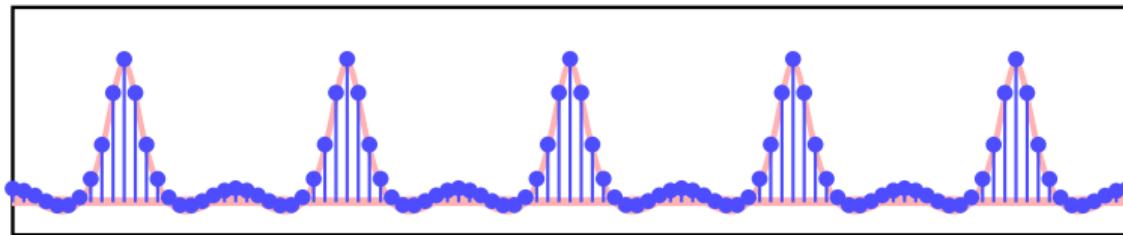
From DFS to DTFT



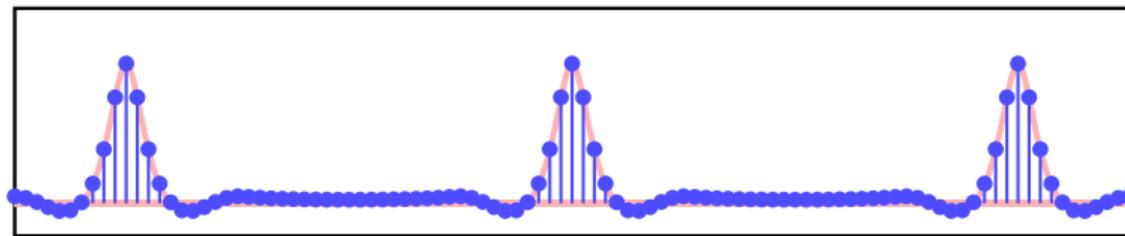
$N = 10$



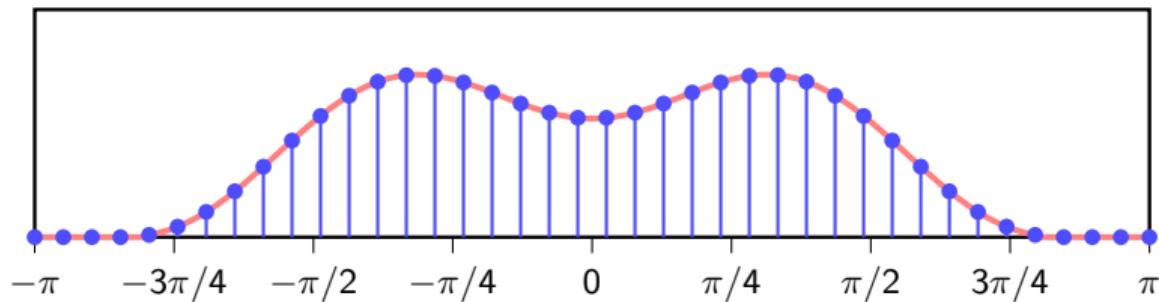
From DFS to DTFT



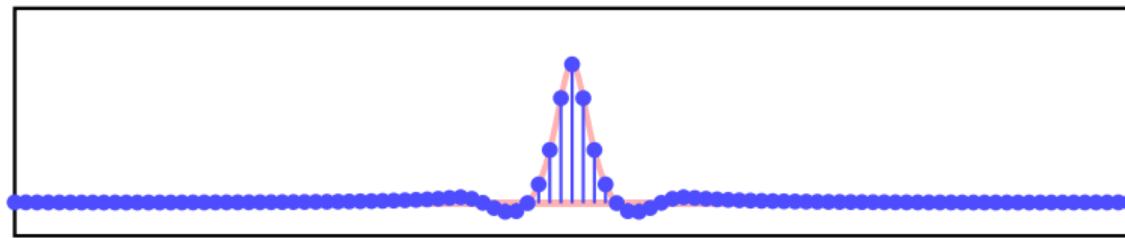
From DFS to DTFT



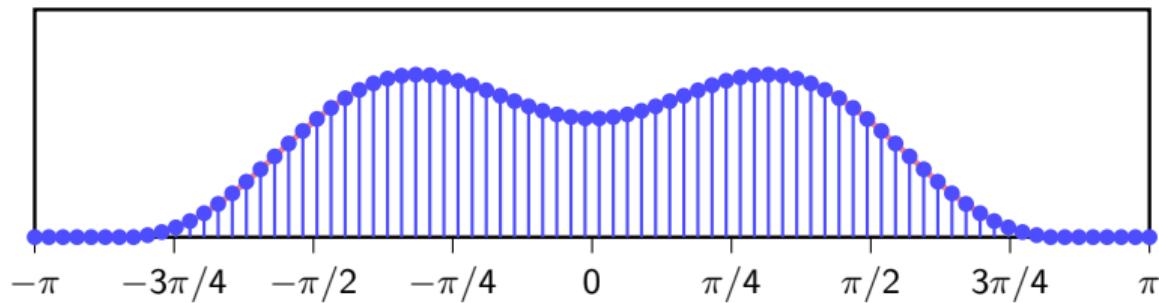
$N = 40$



From DFS to DTFT



$N = 80$



From DFS to DTFT

- ▶ we're comfortable with DFS: change of basis, energy conservation, etc.
- ▶ as N grows, $\tilde{x}_N[n] \rightarrow x[n]$ and the spectral representation "becomes" the DTFT
- ▶ we can retain the "change of basis" paradigm for the DTFT

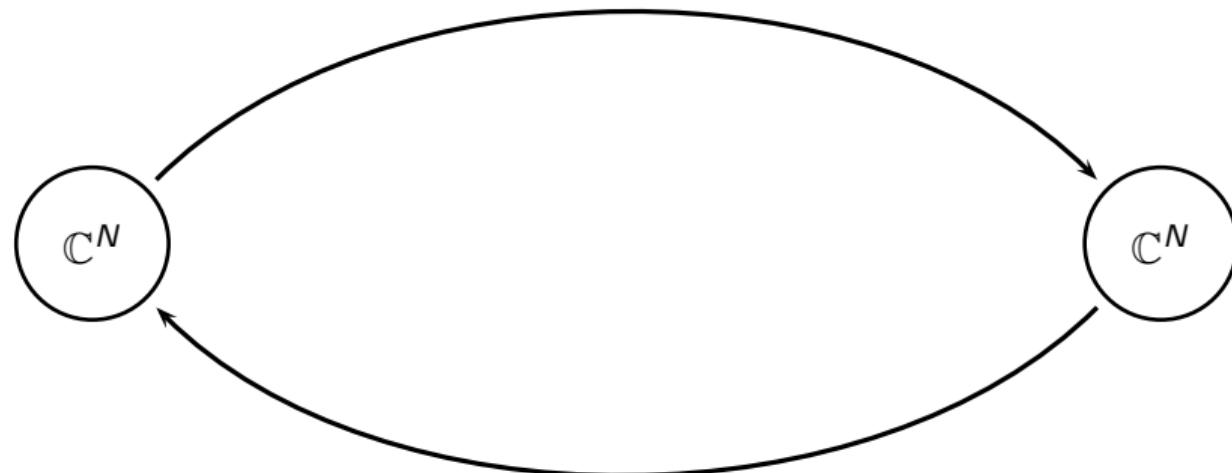
From DFS to DTFT

- ▶ we're comfortable with DFS: change of basis, energy conservation, etc.
- ▶ as N grows, $\tilde{x}_N[n] \rightarrow x[n]$ and the spectral representation "becomes" the DTFT
- ▶ we can retain the "change of basis" paradigm for the DTFT

From DFS to DTFT

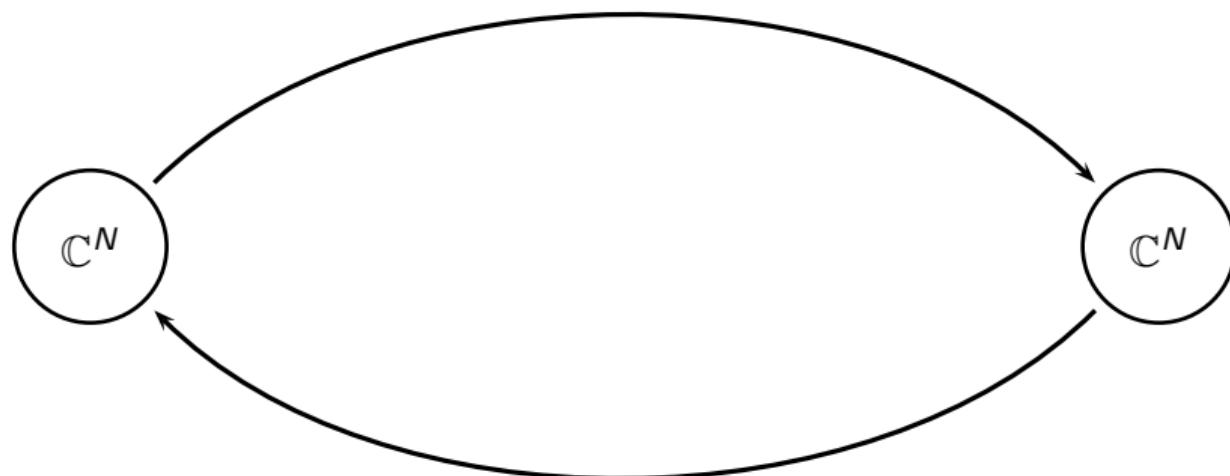
- ▶ we're comfortable with DFS: change of basis, energy conservation, etc.
- ▶ as N grows, $\tilde{x}_N[n] \rightarrow x[n]$ and the spectral representation "becomes" the DTFT
- ▶ we can retain the "change of basis" paradigm for the DTFT

Review: DFT



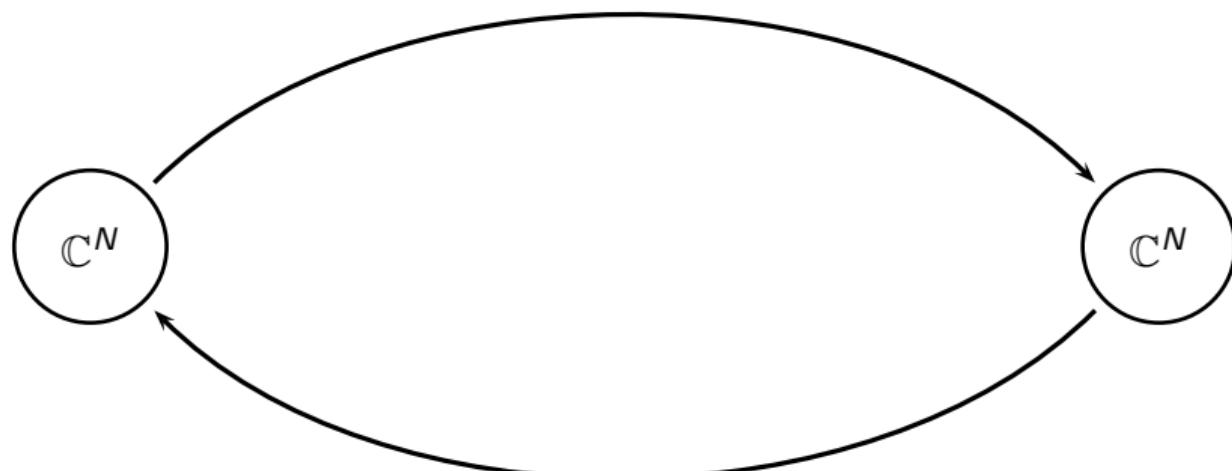
Review: DFT

$$X[k] = \langle e^{j\frac{2\pi}{N}nk}, x[n] \rangle$$



Review: DFT

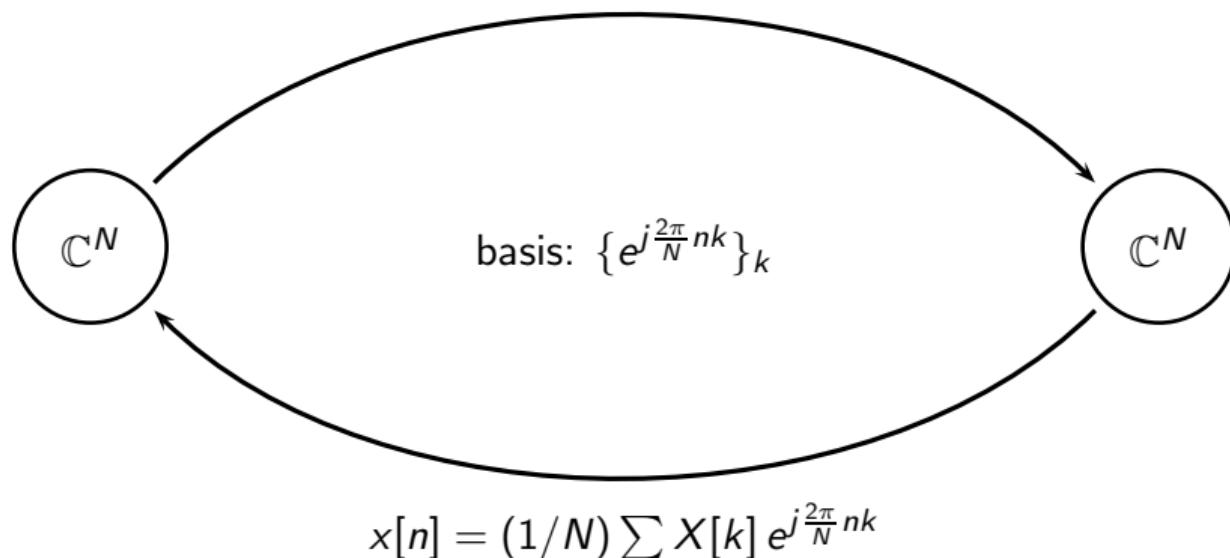
$$X[k] = \langle e^{j\frac{2\pi}{N}nk}, x[n] \rangle$$



$$x[n] = (1/N) \sum X[k] e^{j\frac{2\pi}{N}nk}$$

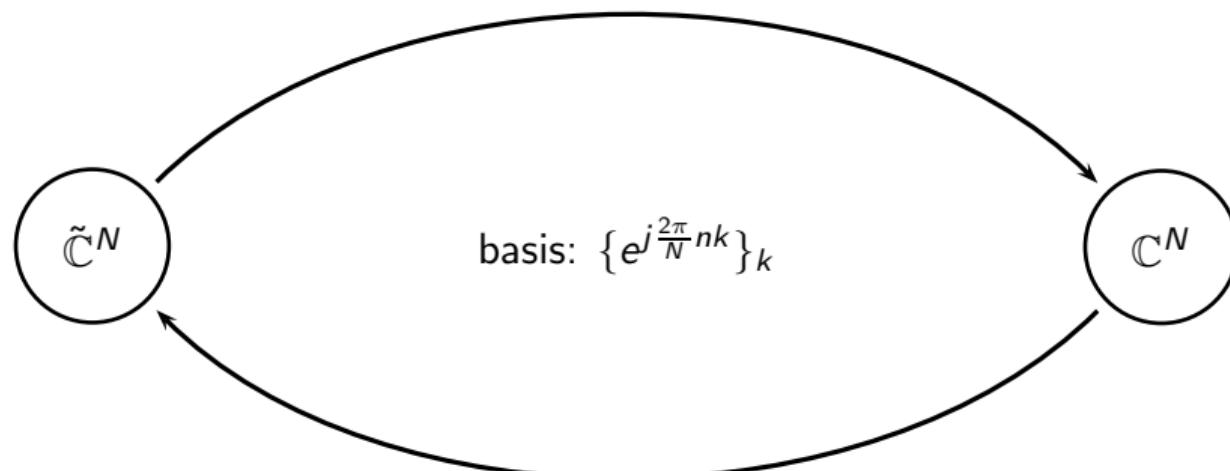
Review: DFT

$$X[k] = \langle e^{j\frac{2\pi}{N}nk}, x[n] \rangle$$



Review: DFS

$$\tilde{X}[k] = \langle e^{j\frac{2\pi}{N}nk}, \tilde{x}[n] \rangle$$



$$\tilde{x}[n] = (1/N) \sum \tilde{X}[k] e^{j\frac{2\pi}{N}nk}$$

What about the DTFT?

- ▶ formally DTFT is an inner product in \mathbb{C}^∞ :

$$\sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} = \langle e^{j\omega n}, x[n] \rangle$$

- ▶ “basis” is an infinite, uncountable basis: $\{e^{j\omega n}\}_{\omega \in \mathbb{R}}$
- ▶ something “breaks down”: we start with sequences but the transform is a function
- ▶ we used absolutely summable sequences but DTFT exists for all square-summable sequences (proof is rather technical)

What about the DTFT?

- ▶ formally DTFT is an inner product in \mathbb{C}^∞ :

$$\sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} = \langle e^{j\omega n}, x[n] \rangle$$

- ▶ “basis” is an infinite, uncountable basis: $\{e^{j\omega n}\}_{\omega \in \mathbb{R}}$
- ▶ something “breaks down”: we start with sequences but the transform is a function
- ▶ we used absolutely summable sequences but DTFT exists for all square-summable sequences (proof is rather technical)

What about the DTFT?

- ▶ formally DTFT is an inner product in \mathbb{C}^∞ :

$$\sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} = \langle e^{j\omega n}, x[n] \rangle$$

- ▶ “basis” is an infinite, uncountable basis: $\{e^{j\omega n}\}_{\omega \in \mathbb{R}}$
- ▶ something “breaks down”: we start with sequences but the transform is a function
- ▶ we used absolutely summable sequences but DTFT exists for all square-summable sequences (proof is rather technical)

What about the DTFT?

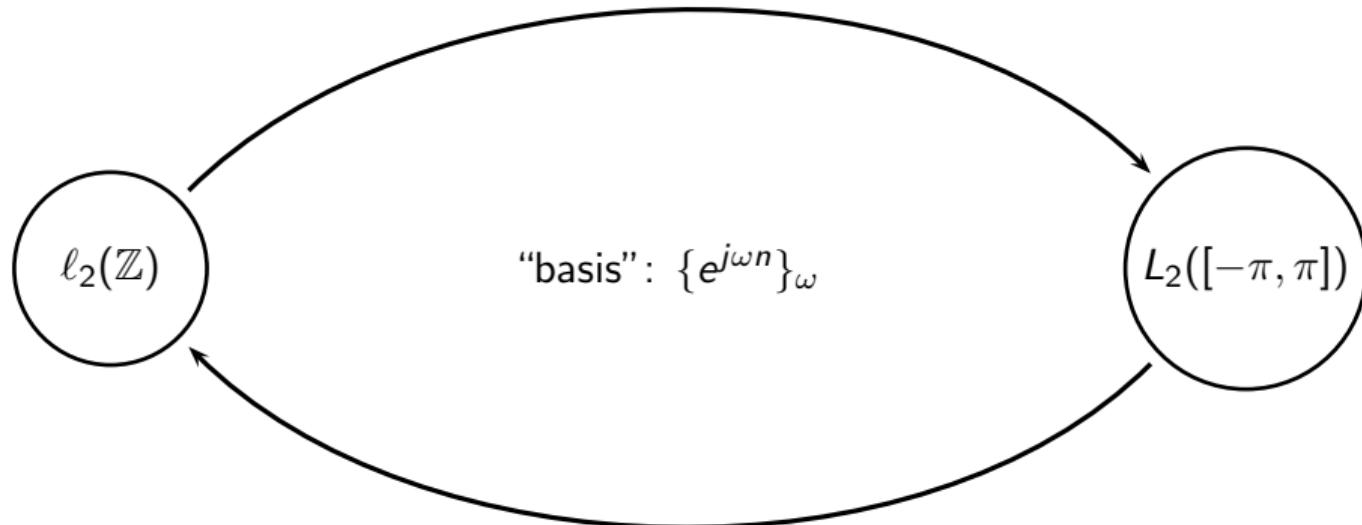
- ▶ formally DTFT is an inner product in \mathbb{C}^∞ :

$$\sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} = \langle e^{j\omega n}, x[n] \rangle$$

- ▶ “basis” is an infinite, uncountable basis: $\{e^{j\omega n}\}_{\omega \in \mathbb{R}}$
- ▶ something “breaks down”: we start with sequences but the transform is a function
- ▶ we used absolutely summable sequences but DTFT exists for all square-summable sequences (proof is rather technical)

DTFT

$$X(e^{j\omega}) = \langle e^{j\omega n}, x[n] \rangle$$



$$x[n] = (1/2\pi) \int X(e^{j\omega}) e^{j\omega n} d\omega$$

DTFT properties

- ▶ linearity

$$\text{DTFT}\{\alpha x[n] + \beta y[n]\} = \alpha X(e^{j\omega}) + \beta Y(e^{j\omega})$$

- ▶ time shift

$$\text{DTFT}\{x[n - M]\} = e^{-j\omega M} X(e^{j\omega})$$

- ▶ modulation (dual)

$$\text{DTFT}\{e^{j\omega_0 n} x[n]\} = X(e^{j(\omega - \omega_0)})$$

DTFT properties

- ▶ linearity

$$\text{DTFT}\{\alpha x[n] + \beta y[n]\} = \alpha X(e^{j\omega}) + \beta Y(e^{j\omega})$$

- ▶ time shift

$$\text{DTFT}\{x[n - M]\} = e^{-j\omega M} X(e^{j\omega})$$

- ▶ modulation (dual)

$$\text{DTFT}\{e^{j\omega_0 n} x[n]\} = X(e^{j(\omega - \omega_0)})$$

DTFT properties

- ▶ linearity

$$\text{DTFT}\{\alpha x[n] + \beta y[n]\} = \alpha X(e^{j\omega}) + \beta Y(e^{j\omega})$$

- ▶ time shift

$$\text{DTFT}\{x[n - M]\} = e^{-j\omega M} X(e^{j\omega})$$

- ▶ modulation (dual)

$$\text{DTFT}\{e^{j\omega_0 n} x[n]\} = X(e^{j(\omega - \omega_0)})$$

DTFT properties

- ▶ time reversal

$$\text{DTFT}\{x[-n]\} = X(e^{-j\omega})$$

- ▶ conjugation

$$\text{DTFT}\{x^*[n]\} = X^*(e^{-j\omega})$$

DTFT properties

- ▶ time reversal

$$\text{DTFT}\{x[-n]\} = X(e^{-j\omega})$$

- ▶ conjugation

$$\text{DTFT}\{x^*[n]\} = X^*(e^{-j\omega})$$

Some particular cases:

- ▶ if $x[n]$ is symmetric, the DTFT is symmetric:

$$x[n] = x[-n] \iff X(e^{j\omega}) = X(e^{-j\omega})$$

- ▶ if $x[n]$ is real, the DTFT is Hermitian-symmetric:

$$x[n] = x^*[n] \iff X(e^{j\omega}) = X^*(e^{-j\omega})$$

- ▶ in other words: if $x[n]$ is real, the magnitude of the DTFT is symmetric:

$$x[n] \in \mathbb{R} \iff |X(e^{j\omega})| = |X(e^{-j\omega})|$$

- ▶ finally, if $x[n]$ is real and symmetric, the DTFT is also real and symmetric!

Some particular cases:

- ▶ if $x[n]$ is symmetric, the DTFT is symmetric:

$$x[n] = x[-n] \iff X(e^{j\omega}) = X(e^{-j\omega})$$

- ▶ if $x[n]$ is real, the DTFT is Hermitian-symmetric:

$$x[n] = x^*[n] \iff X(e^{j\omega}) = X^*(e^{-j\omega})$$

- ▶ in other words: if $x[n]$ is real, the magnitude of the DTFT is symmetric:

$$x[n] \in \mathbb{R} \iff |X(e^{j\omega})| = |X(e^{-j\omega})|$$

- ▶ finally, if $x[n]$ is real and symmetric, the DTFT is also real and symmetric!

Some particular cases:

- ▶ if $x[n]$ is symmetric, the DTFT is symmetric:

$$x[n] = x[-n] \iff X(e^{j\omega}) = X(e^{-j\omega})$$

- ▶ if $x[n]$ is real, the DTFT is Hermitian-symmetric:

$$x[n] = x^*[n] \iff X(e^{j\omega}) = X^*(e^{-j\omega})$$

- ▶ in other words: if $x[n]$ is real, the magnitude of the DTFT is symmetric:

$$x[n] \in \mathbb{R} \iff |X(e^{j\omega})| = |X(e^{-j\omega})|$$

- ▶ finally, if $x[n]$ is real and symmetric, the DTFT is also real and symmetric!

Some particular cases:

- ▶ if $x[n]$ is symmetric, the DTFT is symmetric:

$$x[n] = x[-n] \iff X(e^{j\omega}) = X(e^{-j\omega})$$

- ▶ if $x[n]$ is real, the DTFT is Hermitian-symmetric:

$$x[n] = x^*[n] \iff X(e^{j\omega}) = X^*(e^{-j\omega})$$

- ▶ in other words: if $x[n]$ is real, the magnitude of the DTFT is symmetric:

$$x[n] \in \mathbb{R} \iff |X(e^{j\omega})| = |X(e^{-j\omega})|$$

- ▶ finally, if $x[n]$ is real and symmetric, the DTFT is also real and symmetric!

DTFT as basis expansion

Some things are OK:

- ▶ DFT $\{\delta[n]\} = 1$
- ▶ DTFT $\{\delta[n]\} = \langle e^{j\omega n}, \delta[n] \rangle = 1$

DTFT as basis expansion

Some things are OK:

- ▶ DFT $\{\delta[n]\} = 1$
- ▶ DTFT $\{\delta[n]\} = \langle e^{j\omega n}, \delta[n] \rangle = 1$

DTFT as basis expansion

Some things aren't:

- ▶ DFT $\{1\} = \delta[n]$
- ▶ DTFT $\{1\} = \sum_{n=-\infty}^{\infty} e^{-j\omega n} = ?$

- ▶ problem: too many interesting sequences are *not* square summable!

DTFT as basis expansion

Some things aren't:

- ▶ DFT $\{1\} = \delta[n]$
- ▶ DTFT $\{1\} = \sum_{n=-\infty}^{\infty} e^{-j\omega n} = ?$

- ▶ problem: too many interesting sequences are *not* square summable!

DTFT as basis expansion

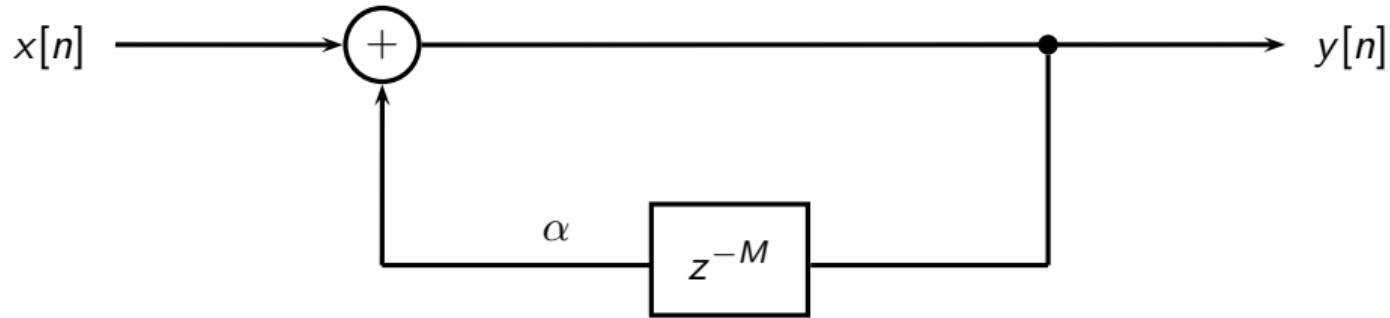
Some things aren't:

- ▶ DFT $\{1\} = \delta[n]$
- ▶ DTFT $\{1\} = \sum_{n=-\infty}^{\infty} e^{-j\omega n} = ?$

- ▶ problem: too many interesting sequences are *not* square summable!

The DTFT of the Karplus-Strong output

Karplus-Strong revisited



$$y[n] = \alpha y[n - M] + x[n]$$

Karplus-Strong revisited

- ▶ choose a signal $\bar{x}[n]$ that is nonzero only for $0 \leq n < M$
- ▶ generated signal is infinite-length but not periodic:

$$y[n] = \bar{x}[0], \bar{x}[1], \dots, \bar{x}[M-1], \alpha\bar{x}[0], \alpha\bar{x}[1], \dots, \alpha\bar{x}[M-1], \alpha^2\bar{x}[0], \alpha^2\bar{x}[1], \dots$$

- ▶ what is the DTFT of this signal?

Karplus-Strong revisited

- ▶ choose a signal $\bar{x}[n]$ that is nonzero only for $0 \leq n < M$
- ▶ generated signal is infinite-length but not periodic:

$$y[n] = \underbrace{\bar{x}[0], \bar{x}[1], \dots, \bar{x}[M-1]}_{1^{\text{st}} \text{ period}}, \underbrace{\alpha \bar{x}[0], \alpha \bar{x}[1], \dots, \alpha \bar{x}[M-1]}_{2^{\text{nd}} \text{ period}}, \underbrace{\alpha^2 \bar{x}[0], \alpha^2 \bar{x}[1], \dots}_{\dots}$$

- ▶ what is the DTFT of this signal?

Karplus-Strong revisited

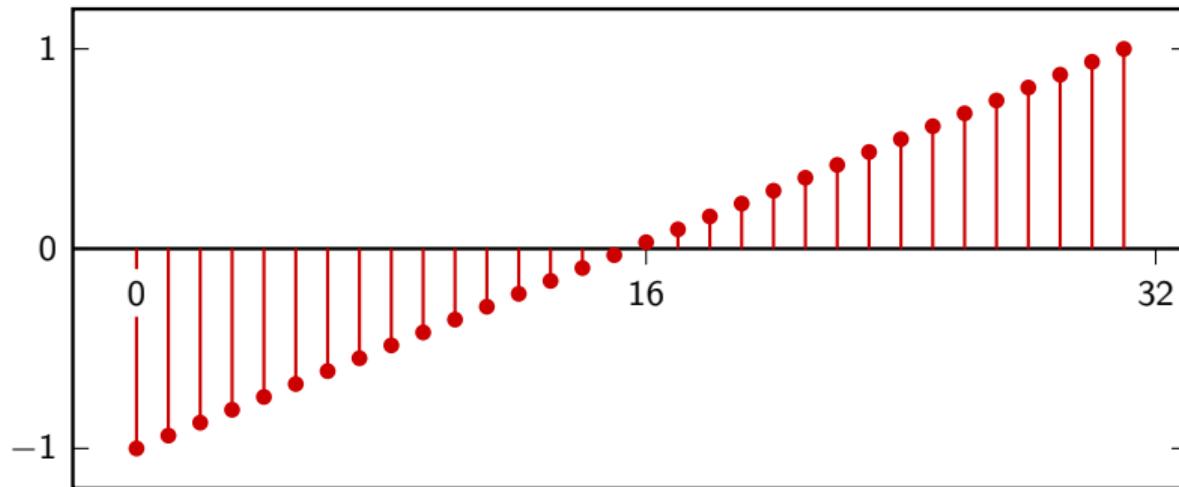
- ▶ choose a signal $\bar{x}[n]$ that is nonzero only for $0 \leq n < M$
- ▶ generated signal is infinite-length but not periodic:

$$y[n] = \underbrace{\bar{x}[0], \bar{x}[1], \dots, \bar{x}[M-1]}_{1^{\text{st}} \text{ period}}, \underbrace{\alpha \bar{x}[0], \alpha \bar{x}[1], \dots, \alpha \bar{x}[M-1]}_{2^{\text{nd}} \text{ period}}, \underbrace{\alpha^2 \bar{x}[0], \alpha^2 \bar{x}[1], \dots}_{\dots}$$

- ▶ what is the DTFT of this signal?

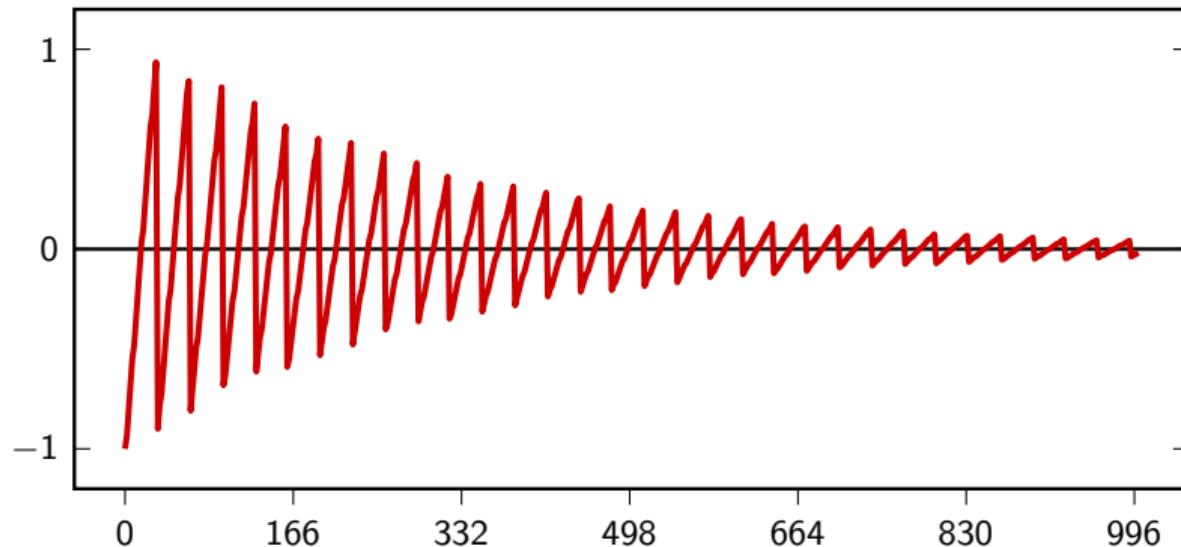
KS revisited: 32-tap sawtooth wave

$$x[n] = 2n/(M - 1) - 1, \quad n = 0, 1, \dots, M - 1$$



KS revisited: decay $\alpha = 0.9$

$$y[n] = \alpha^{\lfloor n/M \rfloor} \bar{x}[n \bmod M] u[n]$$



DTFT of KS signal

$$Y(e^{j\omega}) = \sum_{n=-\infty}^{\infty} y[n] e^{-j\omega n}$$

Same trick we used before:

$$\sum_{m=-\infty}^{\infty} y[m] = \sum_{p=-\infty}^{\infty} \sum_{n=0}^{N-1} y[n + pN]$$

DTFT of KS signal

$$\begin{aligned} Y(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} y[n] e^{-j\omega n} \\ &= \sum_{p=0}^{\infty} \sum_{m=0}^{M-1} \alpha^p \bar{x}[m] e^{-j\omega(pM+m)} \\ &= \sum_{p=0}^{\infty} \alpha^p e^{-j\omega Mp} \sum_{m=0}^{M-1} \bar{x}[m] e^{-j\omega m} \\ &= \sum_{p=0}^{\infty} \alpha^p e^{-j\omega Mp} \sum_{m=-\infty}^{\infty} \bar{x}[m] e^{-j\omega m} \\ &= A(e^{j\omega M}) \bar{X}(e^{j\omega}) \end{aligned}$$

DTFT of KS signal

$$\begin{aligned} Y(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} y[n] e^{-j\omega n} \\ &= \sum_{p=0}^{\infty} \sum_{m=0}^{M-1} \alpha^p \bar{x}[m] e^{-j\omega(pM+m)} \\ &= \sum_{p=0}^{\infty} \alpha^p e^{-j\omega Mp} \sum_{m=0}^{M-1} \bar{x}[m] e^{-j\omega m} \\ &= \sum_{p=0}^{\infty} \alpha^p e^{-j\omega Mp} \sum_{m=-\infty}^{\infty} \bar{x}[m] e^{-j\omega m} \\ &= A(e^{j\omega M}) \bar{X}(e^{j\omega}) \end{aligned}$$

DTFT of KS signal

$$\begin{aligned} Y(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} y[n] e^{-j\omega n} \\ &= \sum_{p=0}^{\infty} \sum_{m=0}^{M-1} \alpha^p \bar{x}[m] e^{-j\omega(pM+m)} \\ &= \sum_{p=0}^{\infty} \alpha^p e^{-j\omega Mp} \sum_{m=0}^{M-1} \bar{x}[m] e^{-j\omega m} \\ &= \sum_{p=0}^{\infty} \alpha^p e^{-j\omega Mp} \sum_{m=-\infty}^{\infty} \bar{x}[m] e^{-j\omega m} \\ &= A(e^{j\omega M}) \bar{X}(e^{j\omega}) \end{aligned}$$

DTFT of KS signal

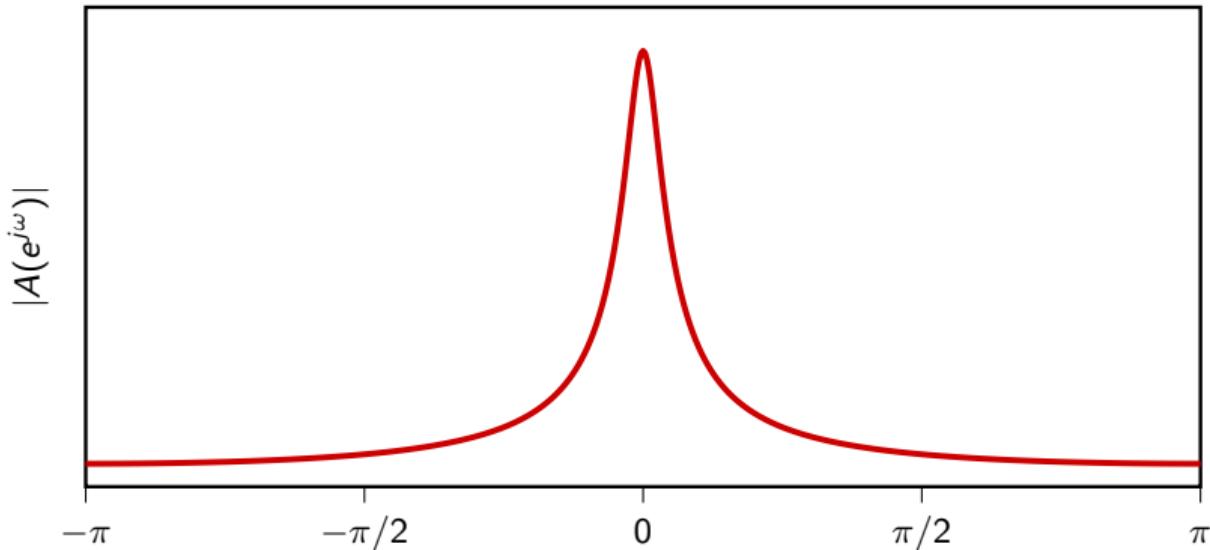
$$\begin{aligned} Y(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} y[n] e^{-j\omega n} \\ &= \sum_{p=0}^{\infty} \sum_{m=0}^{M-1} \alpha^p \bar{x}[m] e^{-j\omega(pM+m)} \\ &= \sum_{p=0}^{\infty} \alpha^p e^{-j\omega Mp} \sum_{m=0}^{M-1} \bar{x}[m] e^{-j\omega m} \\ &= \sum_{p=0}^{\infty} \alpha^p e^{-j\omega Mp} \sum_{m=-\infty}^{\infty} \bar{x}[m] e^{-j\omega m} \\ &= A(e^{j\omega M}) \bar{X}(e^{j\omega}) \end{aligned}$$

DTFT of KS signal

$$\begin{aligned} Y(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} y[n] e^{-j\omega n} \\ &= \sum_{p=0}^{\infty} \sum_{m=0}^{M-1} \alpha^p \bar{x}[m] e^{-j\omega(pM+m)} \\ &= \sum_{p=0}^{\infty} \alpha^p e^{-j\omega Mp} \sum_{m=0}^{M-1} \bar{x}[m] e^{-j\omega m} \\ &= \sum_{p=0}^{\infty} \alpha^p e^{-j\omega Mp} \sum_{m=-\infty}^{\infty} \bar{x}[m] e^{-j\omega m} \\ &= A(e^{j\omega M}) \bar{X}(e^{j\omega}) \end{aligned}$$

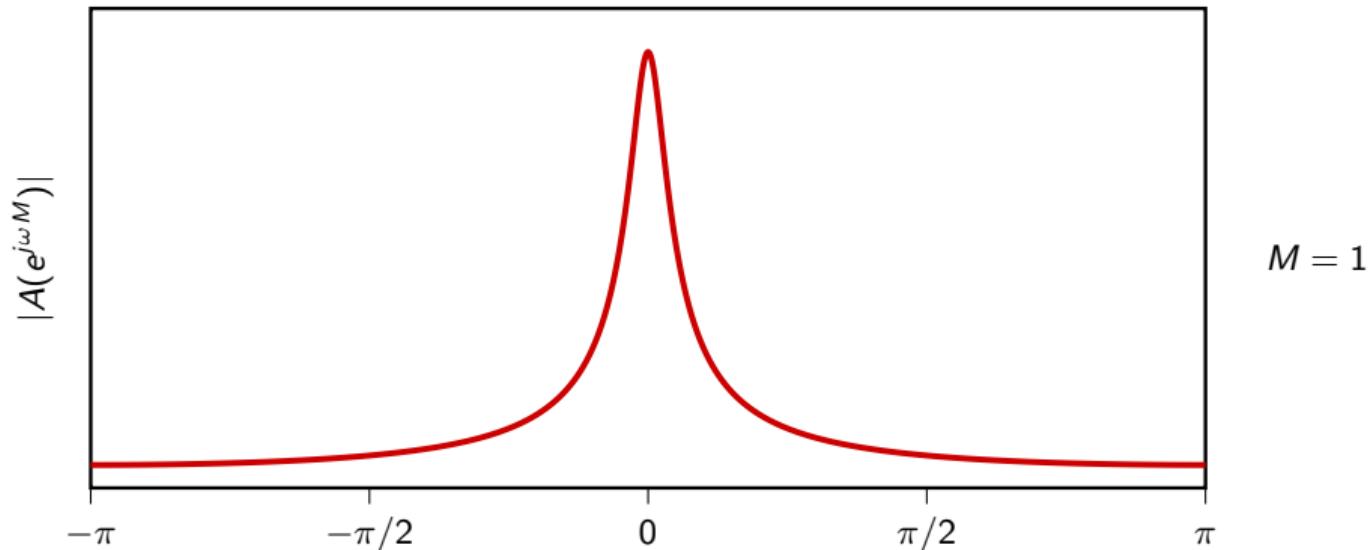
We know the first term

$$A(e^{j\omega}) = \text{DTFT} \{ \alpha^n u[n] \} = \frac{1}{1 - \alpha e^{-j\omega}}$$



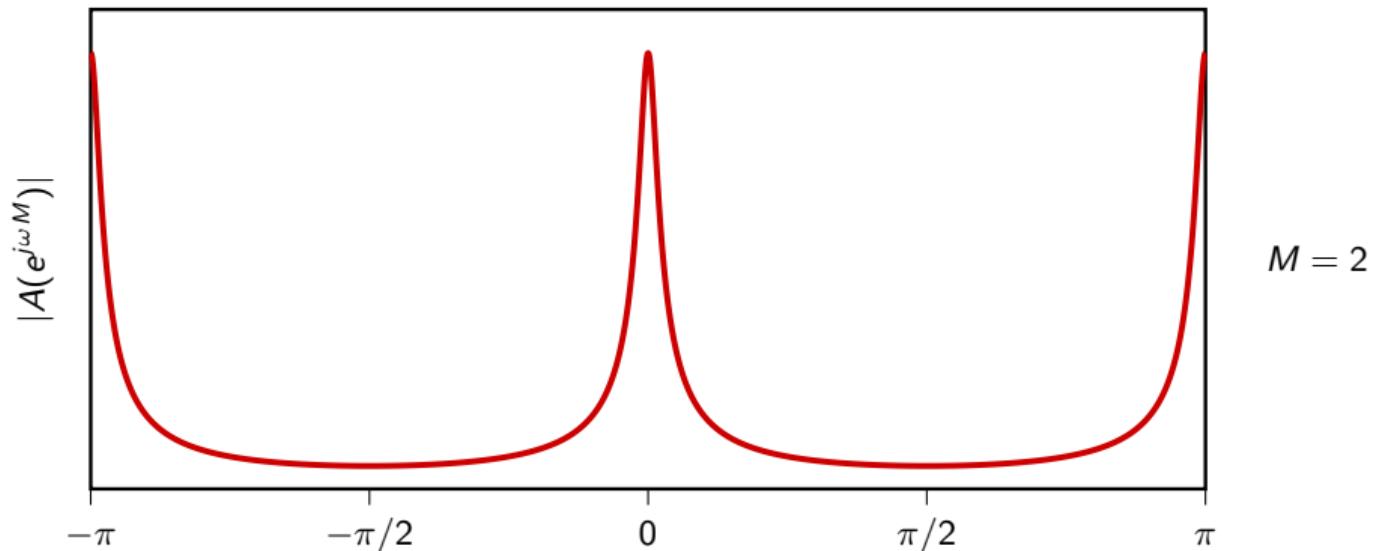
We know the first term

$A(e^{j\omega M})$ rescales the frequency axis: **periodicity!**



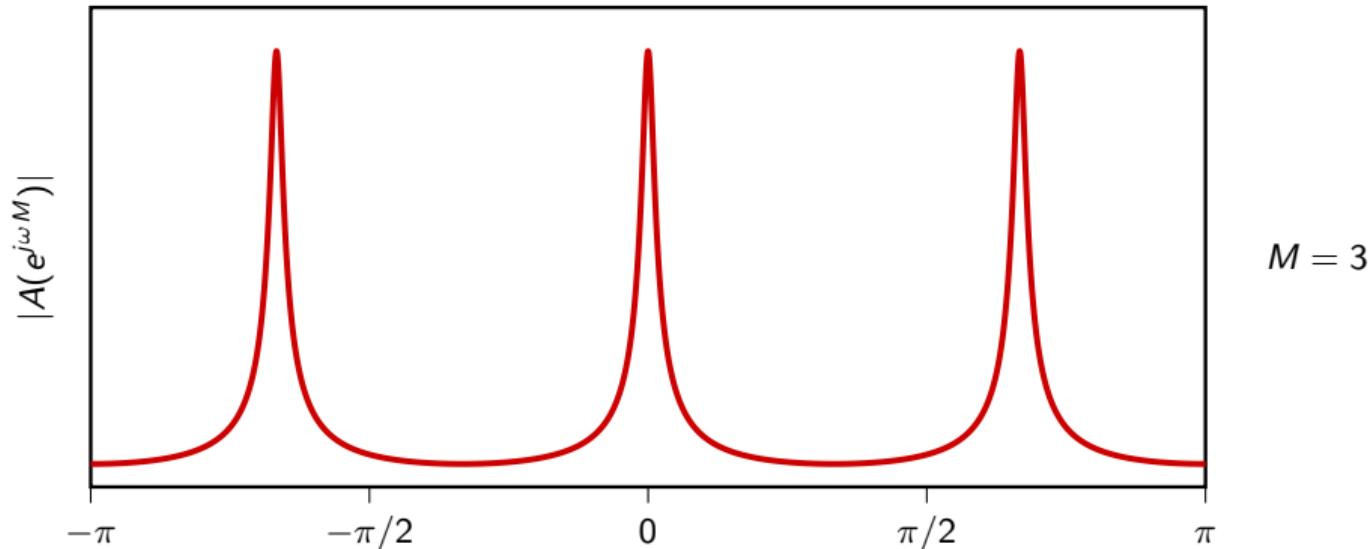
We know the first term

$A(e^{j\omega M})$ rescales the frequency axis: **periodicity!**



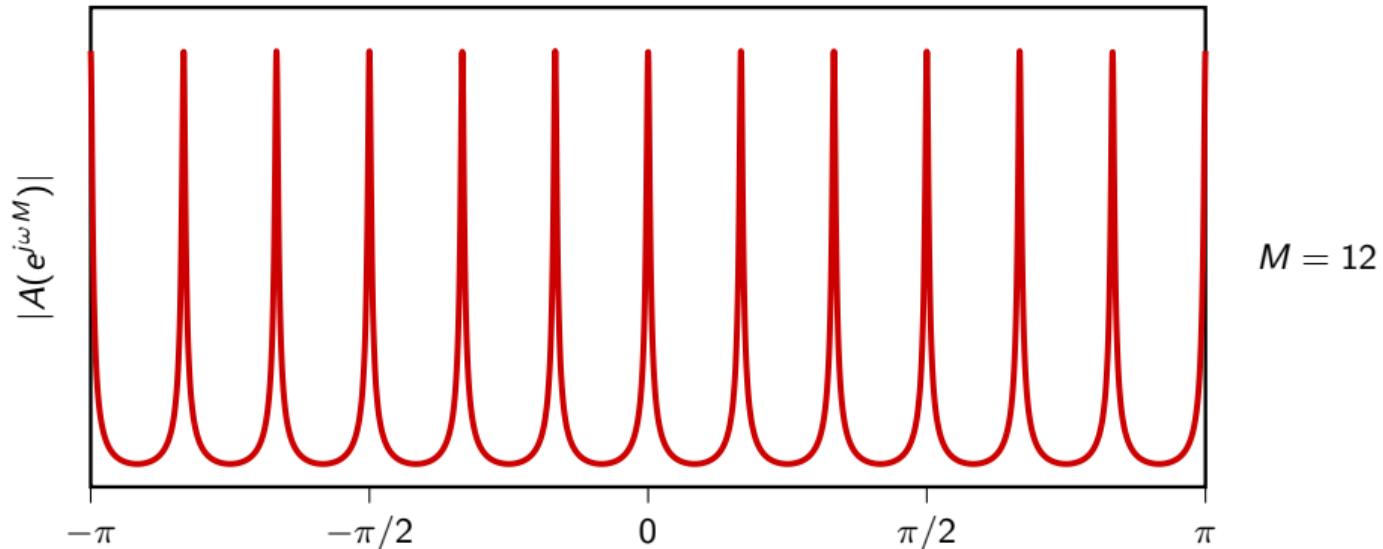
We know the first term

$A(e^{j\omega M})$ rescales the frequency axis: **periodicity!**



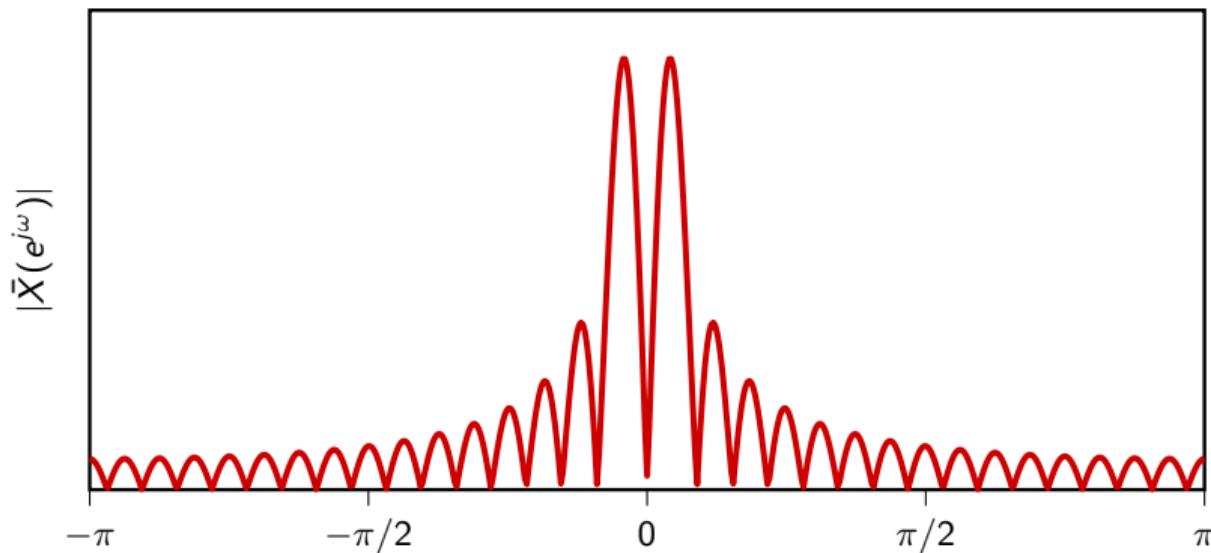
We know the first term

$A(e^{j\omega M})$ rescales the frequency axis: **periodicity!**



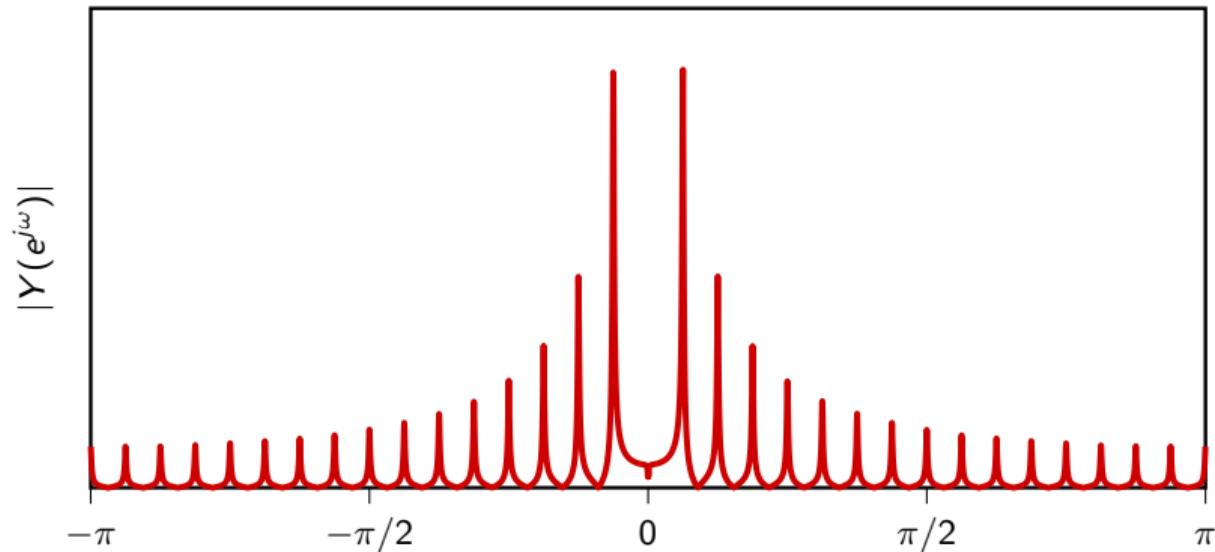
Second term is left as an exercise

$$\bar{X}(e^{j\omega}) = e^{-j\omega} \left(\frac{M+1}{M-1} \right) \frac{1 - e^{-j(M-1)\omega}}{(1 - e^{-j\omega})^2} - \frac{1 - e^{-j(M+1)\omega}}{(1 - e^{-j\omega})^2}$$



DTFT of KS with decay

$$Y(e^{j\omega}) = A(e^{j\omega M}) \bar{X}(e^{j\omega})$$



COM303: Digital Signal Processing

Lecture 7: The DTFT Formalism

Overview:

- ▶ the DTFT of non square-summable sequences
- ▶ relationships between transforms
- ▶ modulation

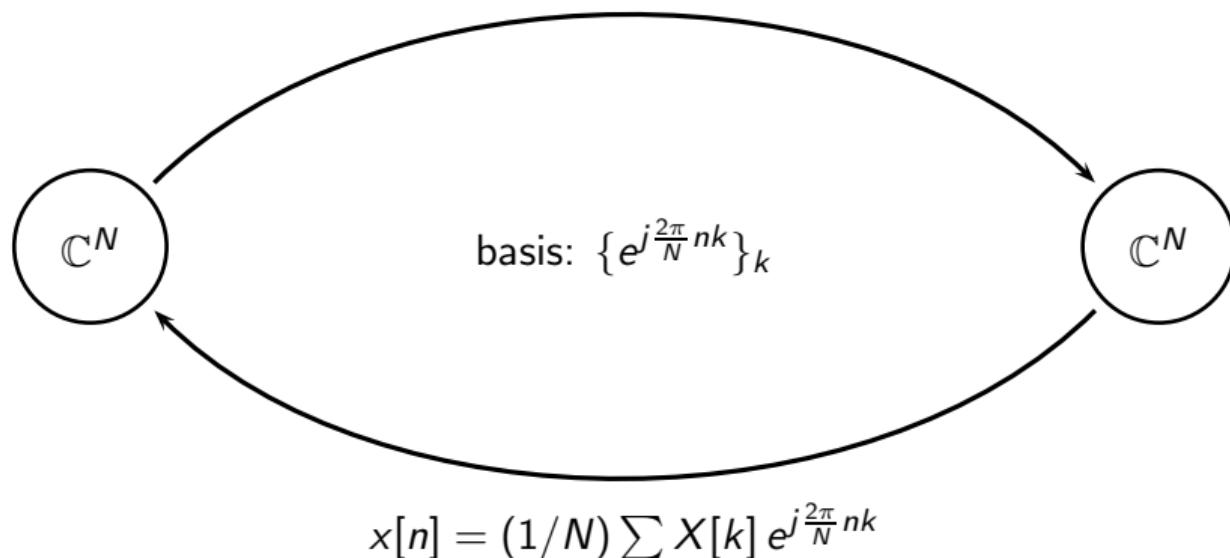
the DTFT formalism for non ℓ_2 sequences

The path to the DTFT

- ▶ DFT: simple change of basis in \mathbb{C}^N
- ▶ DFS: same, with N -periodicity explicit

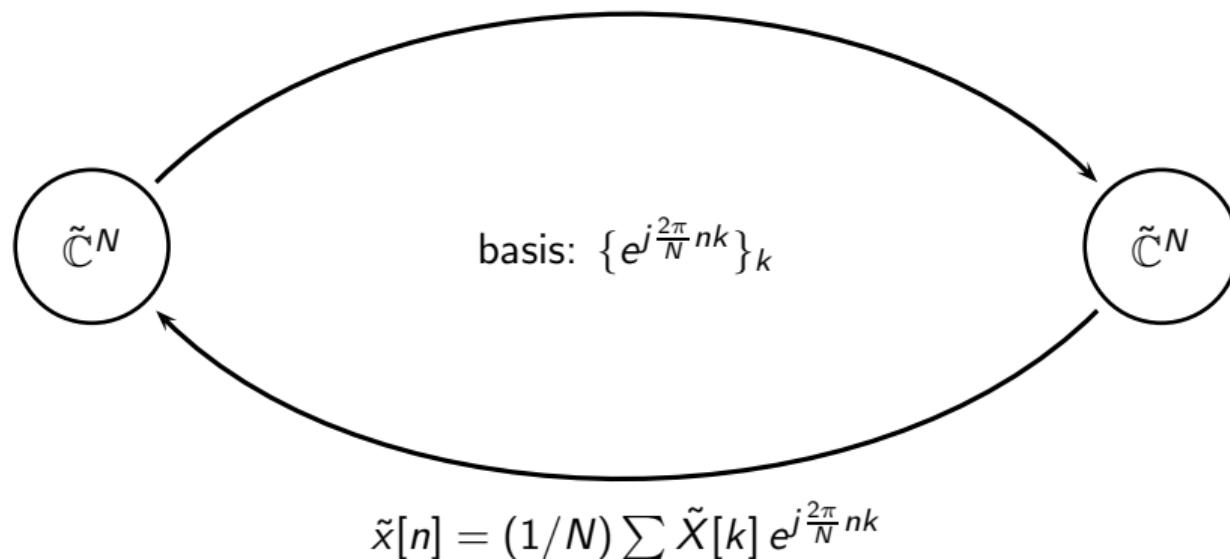
Review: DFT

$$X[k] = \langle e^{j\frac{2\pi}{N}nk}, x[n] \rangle$$



Review: DFS

$$\tilde{X}[k] = \langle e^{j\frac{2\pi}{N}nk}, \tilde{x}[n] \rangle$$



The path to the DTFT

- ▶ DFT: simple change of basis in \mathbb{C}^N
- ▶ DFS: same, with N -periodicity explicit
- ▶ when $N \rightarrow \infty$:

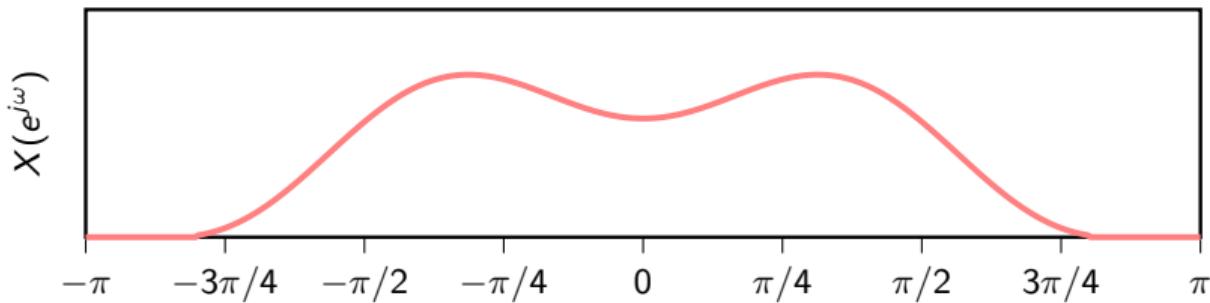
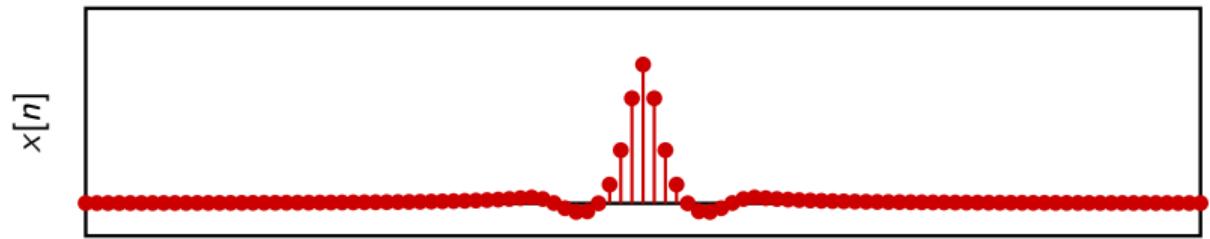
- DFT (informally): $\sum_n x[n]e^{j\frac{2\pi}{N}kn} \rightarrow \sum_n x[n]e^{j\omega n}$

- DFS (formally): $\sum_{n=0}^{N-1} \tilde{x}_N[n]e^{j\frac{2\pi}{N}kn} = \sum_{n=-\infty}^{\infty} x[n]e^{j\omega n} \Big|_{\omega=\frac{2\pi}{N}k}$

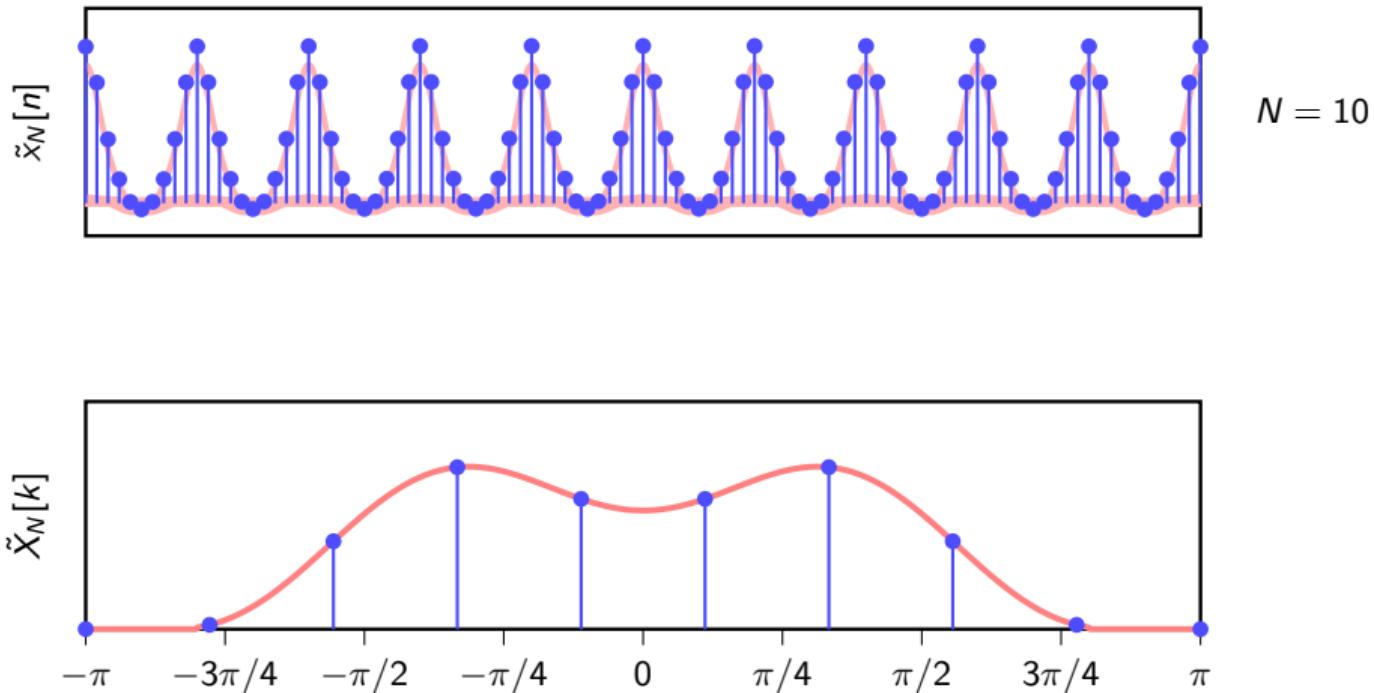
Discrete-Time Fourier Transform

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

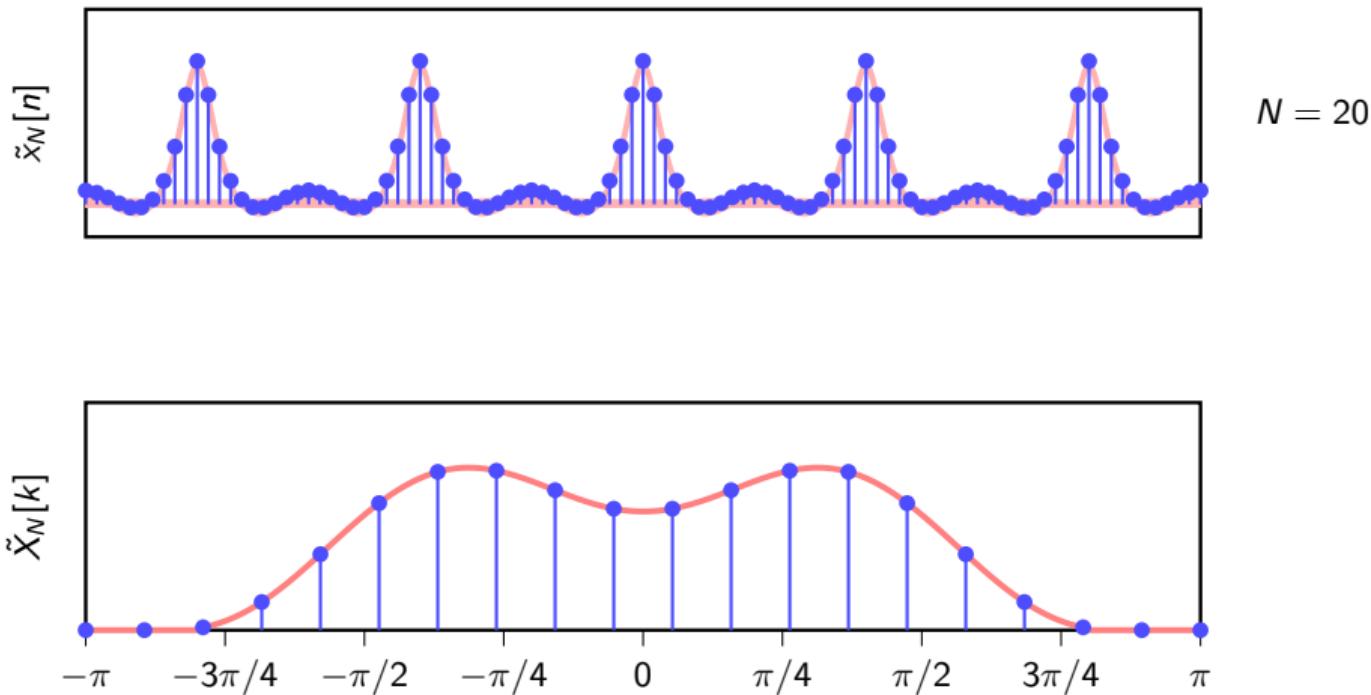
From DFS to DTFT



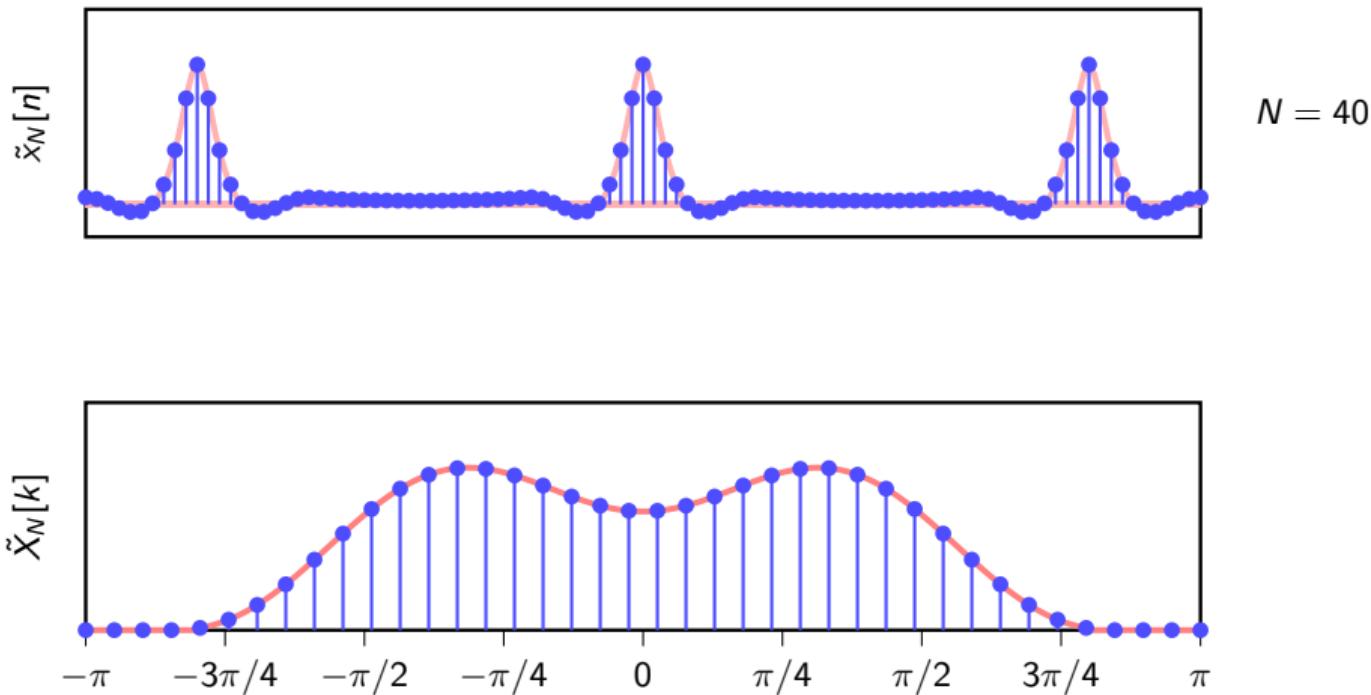
From DFS to DTFT



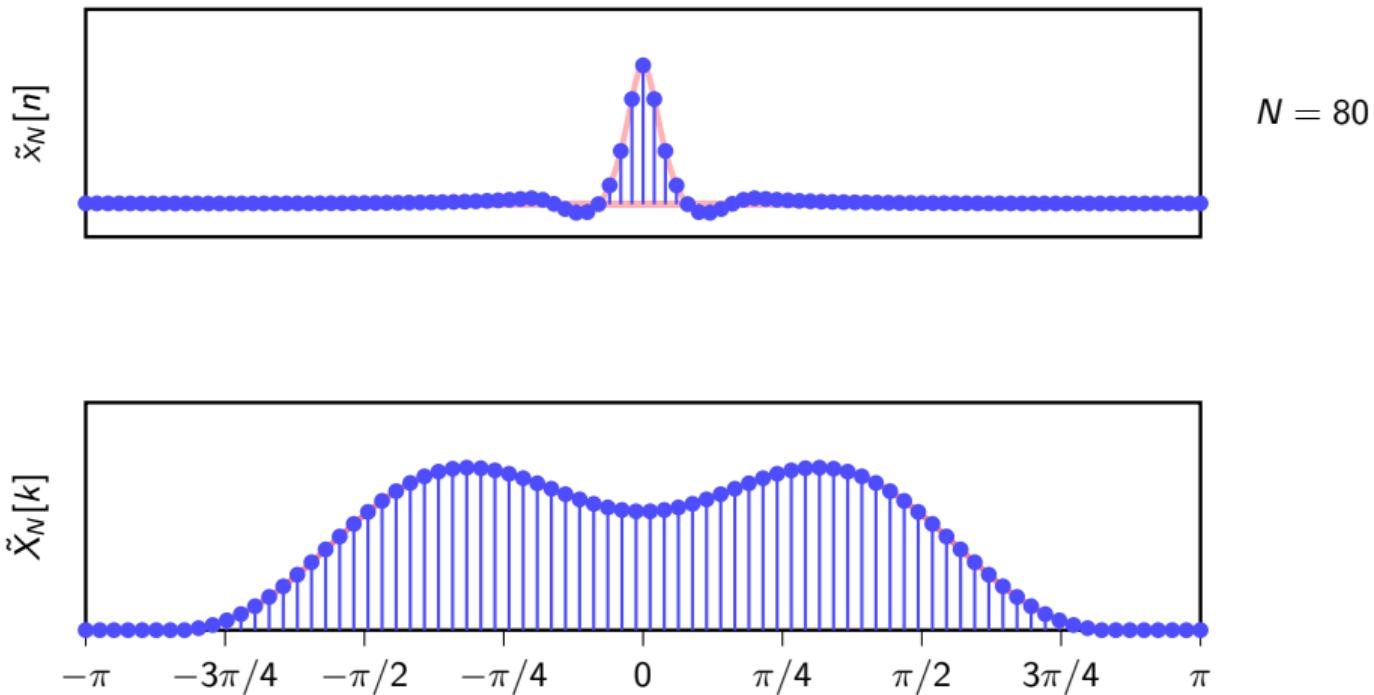
From DFS to DTFT



From DFS to DTFT



From DFS to DTFT



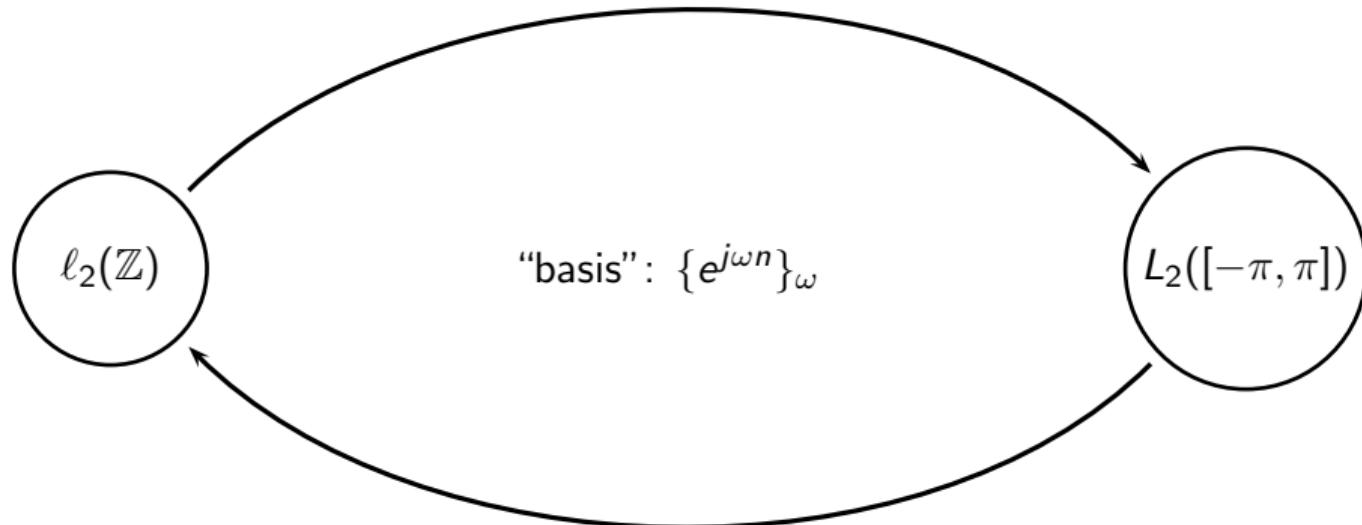
Discrete-Time Fourier Transform

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}, \quad \omega \in [-\pi, \pi]$$

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega, \quad n \in \mathbb{Z}$$

DTFT

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



$$x[n] = (1/2\pi) \int X(e^{j\omega}) e^{j\omega n} d\omega$$

Inverse DTFT as a basis expansion

- ▶ DTFT is an (invertible) mapping from $\ell_2(\mathbb{Z})$ to $L_2([-\pi, \pi])$
- ▶ $L_2([-\pi, \pi])$ is a Hilbert space
- ▶ the set of 2π -periodic functions $\{e^{-j\omega n}\}_n$ is an orthogonal basis for $L_2([-\pi, \pi])$:

$$\langle e^{-j\omega n}, e^{-j\omega m} \rangle = 2\pi\delta[n - m]$$

- ▶ the inverse DTFT is a basis expansion; the analysis coefficients are the time-domain values:

$$x[n] \propto \langle e^{-j\omega n}, X(e^{j\omega n}) \rangle$$

Inverse DTFT as a basis expansion

- ▶ DTFT is an (invertible) mapping from $\ell_2(\mathbb{Z})$ to $L_2([-\pi, \pi])$
- ▶ $L_2([-\pi, \pi])$ is a Hilbert space
- ▶ the set of 2π -periodic functions $\{e^{-j\omega n}\}_n$ is an orthogonal basis for $L_2([-\pi, \pi])$:

$$\langle e^{-j\omega n}, e^{-j\omega m} \rangle = 2\pi\delta[n - m]$$

- ▶ the inverse DTFT is a basis expansion; the analysis coefficients are the time-domain values:

$$x[n] \propto \langle e^{-j\omega n}, X(e^{j\omega n}) \rangle$$

Inverse DTFT as a basis expansion

- ▶ DTFT is an (invertible) mapping from $\ell_2(\mathbb{Z})$ to $L_2([-\pi, \pi])$
- ▶ $L_2([-\pi, \pi])$ is a Hilbert space
- ▶ the set of 2π -periodic functions $\{e^{-j\omega n}\}_n$ is an orthogonal basis for $L_2([-\pi, \pi])$:

$$\langle e^{-j\omega n}, e^{-j\omega m} \rangle = 2\pi\delta[n - m]$$

- ▶ the inverse DTFT is a basis expansion; the analysis coefficients are the time-domain values:

$$x[n] \propto \langle e^{-j\omega n}, X(e^{j\omega n}) \rangle$$

Inverse DTFT as a basis expansion

- ▶ DTFT is an (invertible) mapping from $\ell_2(\mathbb{Z})$ to $L_2([-\pi, \pi])$
- ▶ $L_2([-\pi, \pi])$ is a Hilbert space
- ▶ the set of 2π -periodic functions $\{e^{-j\omega n}\}_n$ is an orthogonal basis for $L_2([-\pi, \pi])$:

$$\langle e^{-j\omega n}, e^{-j\omega m} \rangle = 2\pi\delta[n - m]$$

- ▶ the inverse DTFT is a basis expansion; the analysis coefficients are the time-domain values:

$$x[n] \propto \langle e^{-j\omega n}, X(e^{j\omega n}) \rangle$$

DTFT as a formal basis expansion

- ▶ DTFT can be seen as an inner product

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} = \langle e^{j\omega n}, x[n] \rangle$$

- ▶ the set $\{e^{j\omega n}\}_{\omega}$ is not countable
- ▶ the “basis vectors” don’t even belong to $\ell_2(\mathbb{Z})$

DTFT as a formal basis expansion

- ▶ DTFT can be seen as an inner product

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} = \langle e^{j\omega n}, x[n] \rangle$$

- ▶ the set $\{e^{j\omega n}\}_{\omega}$ is not countable
- ▶ the “basis vectors” don’t even belong to $\ell_2(\mathbb{Z})$

DTFT as a formal basis expansion

- ▶ DTFT can be seen as an inner product

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} = \langle e^{j\omega n}, x[n] \rangle$$

- ▶ the set $\{e^{j\omega n}\}_{\omega}$ is not countable
- ▶ the “basis vectors” don’t even belong to $\ell_2(\mathbb{Z})$

DTFT as basis expansion

Some things are OK:

- ▶ DFT $\{\delta[n]\} = 1$
- ▶ DTFT $\{\delta[n]\} = \langle e^{j\omega n}, \delta[n] \rangle = 1$

DTFT as basis expansion

Some things are OK:

- ▶ DFT $\{\delta[n]\} = 1$
- ▶ DTFT $\{\delta[n]\} = \langle e^{j\omega n}, \delta[n] \rangle = 1$

DTFT as basis expansion

Some things aren't:

- ▶ DFT $\{1\} = \delta[n]$
- ▶ DTFT $\{1\} = \sum_{n=-\infty}^{\infty} e^{-j\omega n} = ?$

- ▶ problem: too many interesting sequences are *not* square summable!

The Dirac delta

The Dirac delta functional

the functional $\delta(t)$ is defined by its “sifting” property:

$$\int_{-\infty}^{\infty} \delta(t-s)f(t)dt = f(s)$$

for all $f(t)$, $t \in \mathbb{R}$.

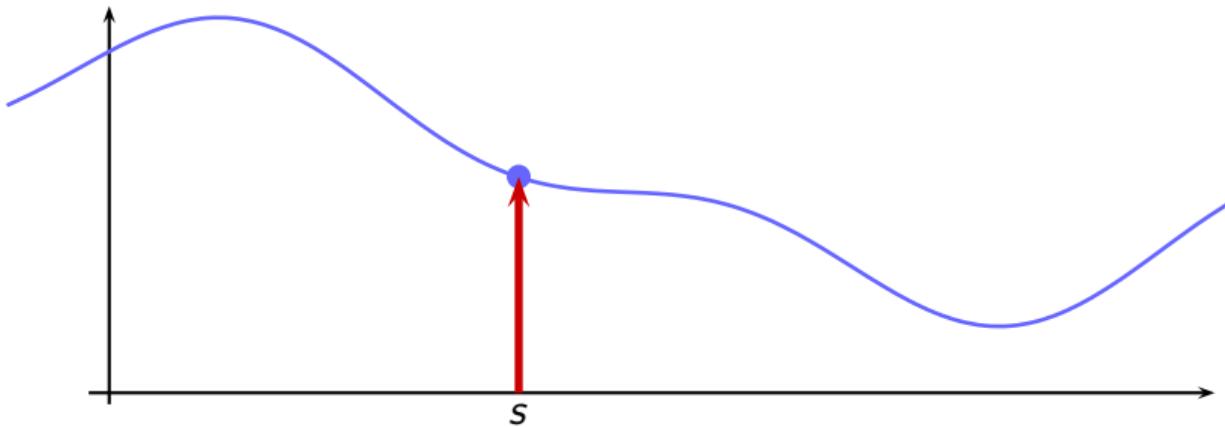
The Dirac delta functional

the functional $\delta(t)$ is defined by its “sifting” property:

$$\int_{s-\delta}^{s+\delta} \delta(t-s)f(t)dt = f(s) \quad \forall \delta \in \mathbb{R}^+$$

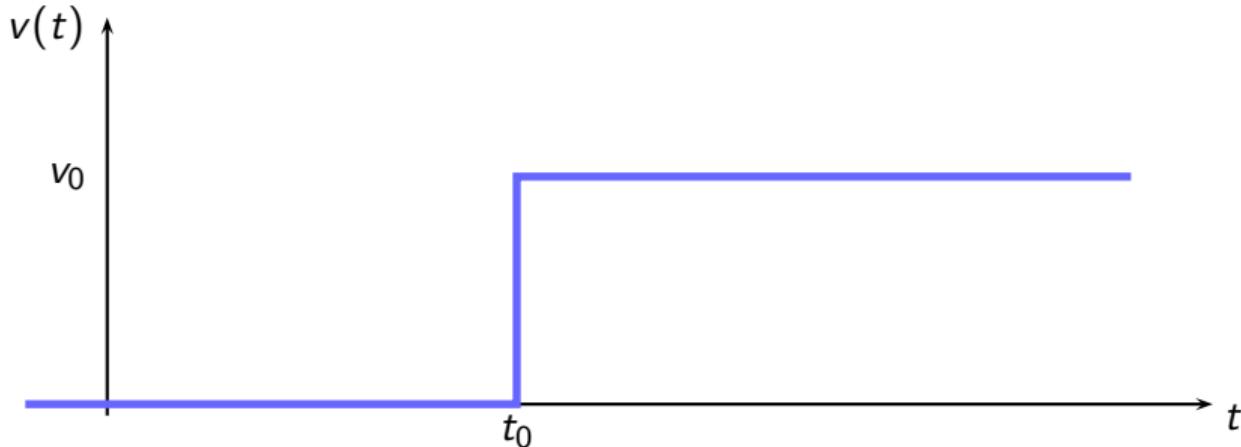
for all $f(t)$, $t \in \mathbb{R}$.

The Dirac delta functional



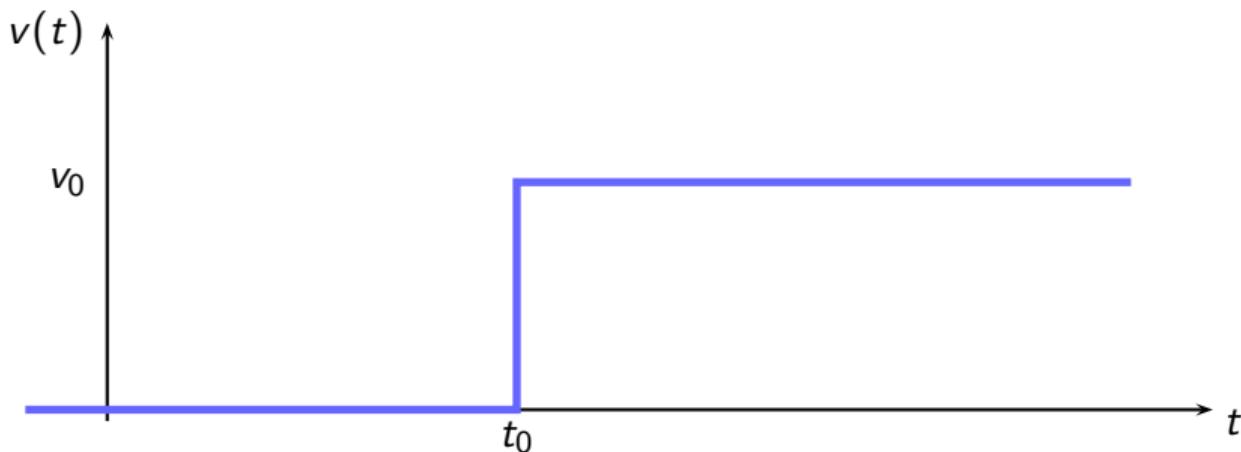
$$\int_{-\infty}^{\infty} \delta(t - s)f(t)dt = f(s)$$

The Dirac delta functional in physics



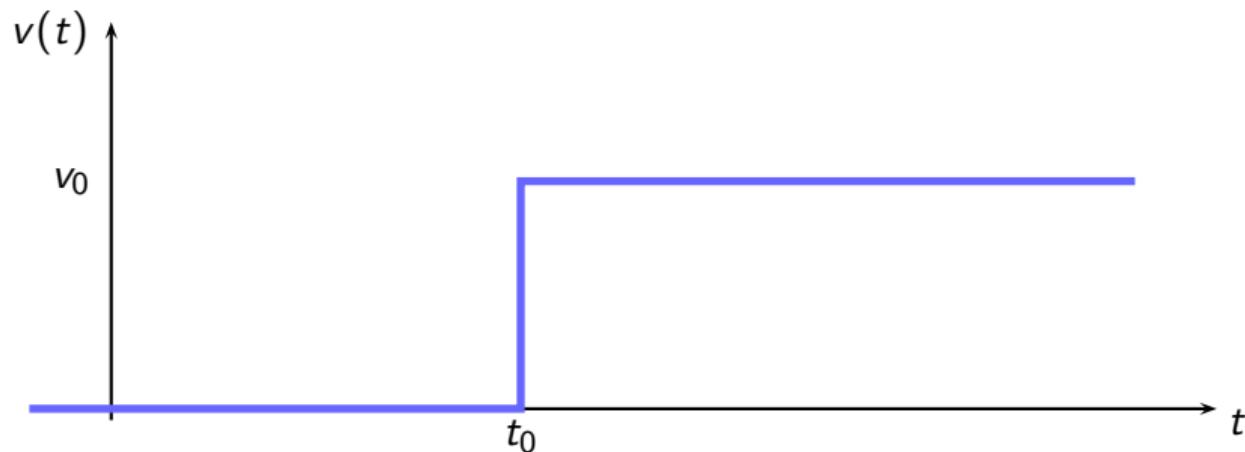
$$F(t) = ma(t) = m \frac{\partial v(t)}{\partial t}$$

The Dirac delta functional in physics



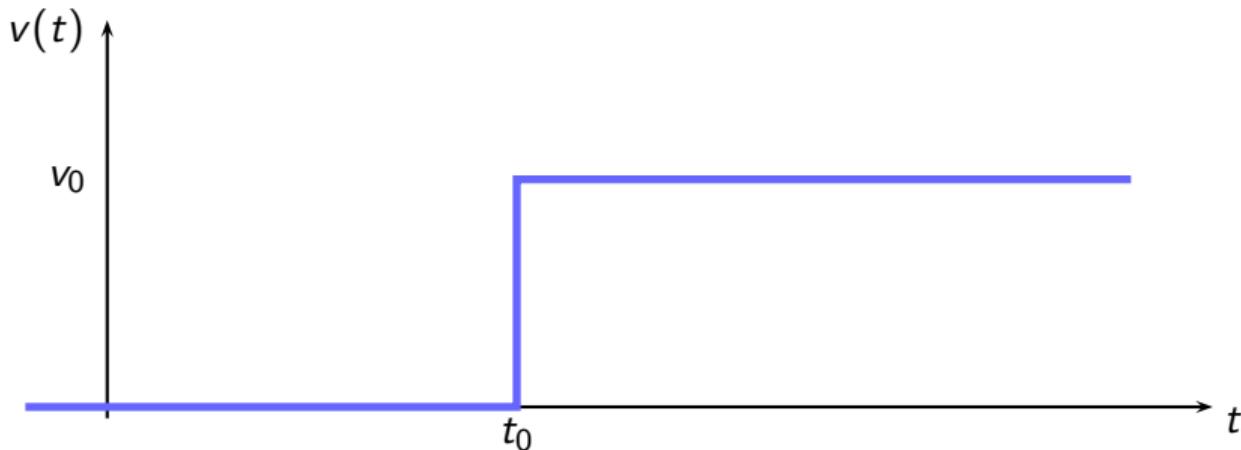
$$a(t_0) = \infty?$$

The Dirac delta functional in physics



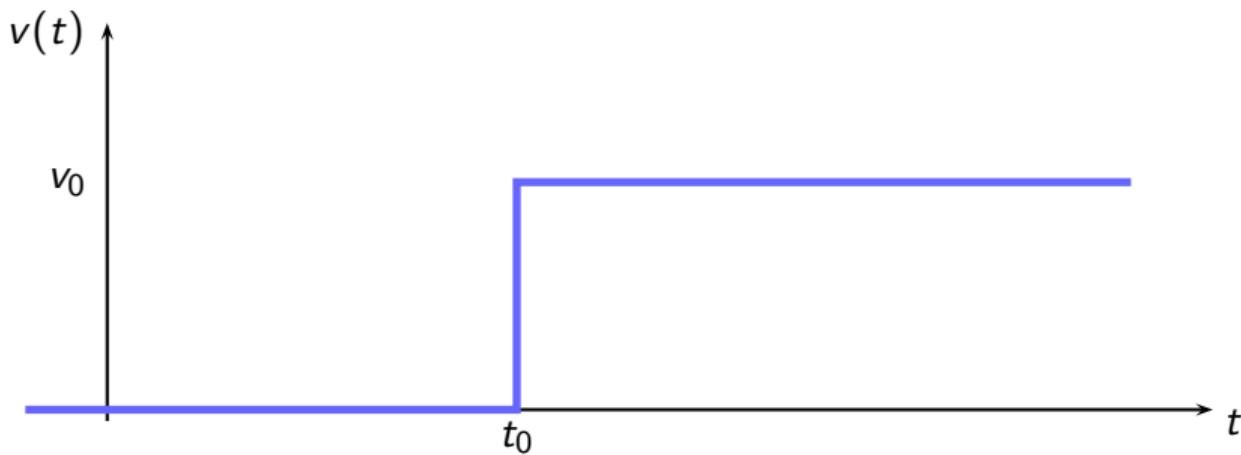
from the other side: $v(t) = \int_{-\infty}^t a(\tau)d\tau = \begin{cases} 0 & \text{for } t < t_0 \\ v_0 & \text{for } t > t_0 \end{cases}$

The Dirac delta functional in physics



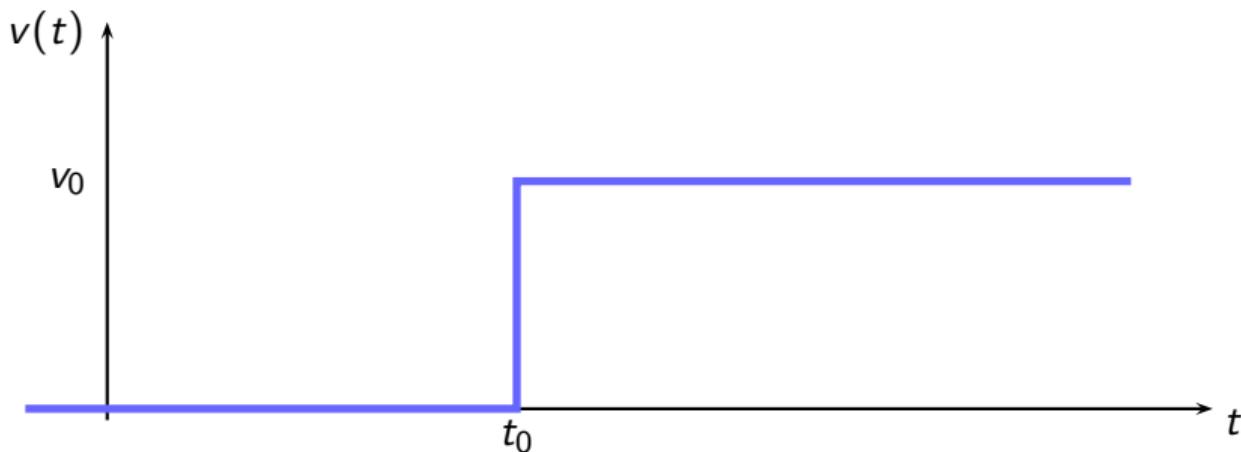
from the other side: $v(t) = \int_{-\infty}^t v_0 \delta(\tau - t_0) d\tau$

The Dirac delta functional in physics



$$a(t) = v_0 \delta(t - t_0)$$

The Dirac delta functional in physics



$$F(t) \propto \delta(t - t_0) \approx \begin{cases} \infty & \text{for } t = t_0 \\ 0 & \text{otherwise} \end{cases}$$

Intuition

consider a family of *localizing* functions $r_k(t)$ with $k \in \mathbb{N}$ and $t \in \mathbb{R}$ where:

- ▶ support inversely proportional to k
- ▶ constant area

Intuition

consider a family of *localizing* functions $r_k(t)$ with $k \in \mathbb{N}$ and $t \in \mathbb{R}$ where:

- ▶ support inversely proportional to k
- ▶ constant area

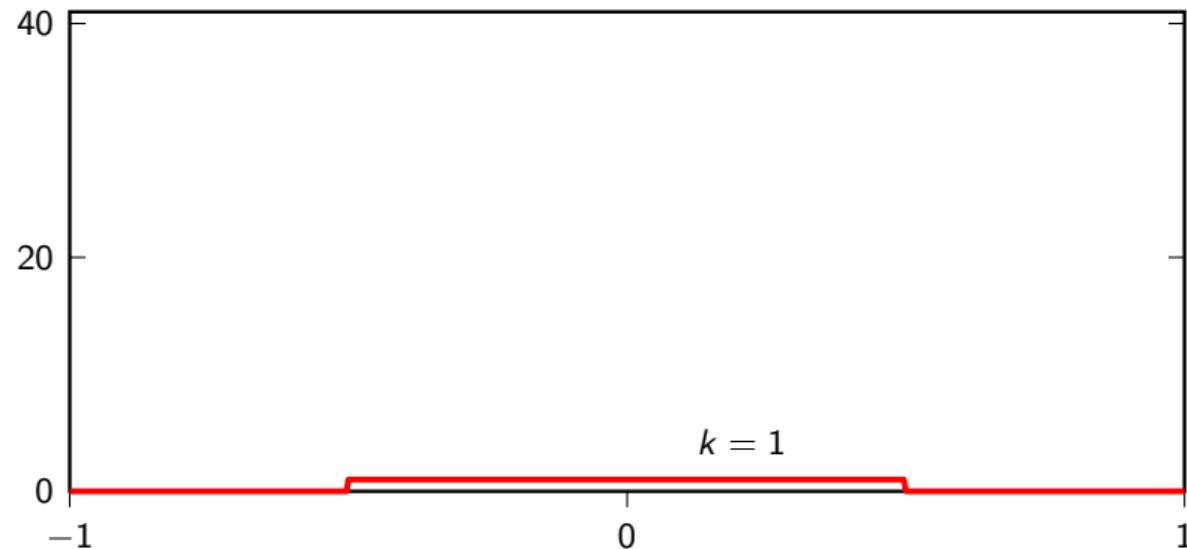
Intuition

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

we can build a localizing family as $r_k(t) = k \text{rect}(kt)$:

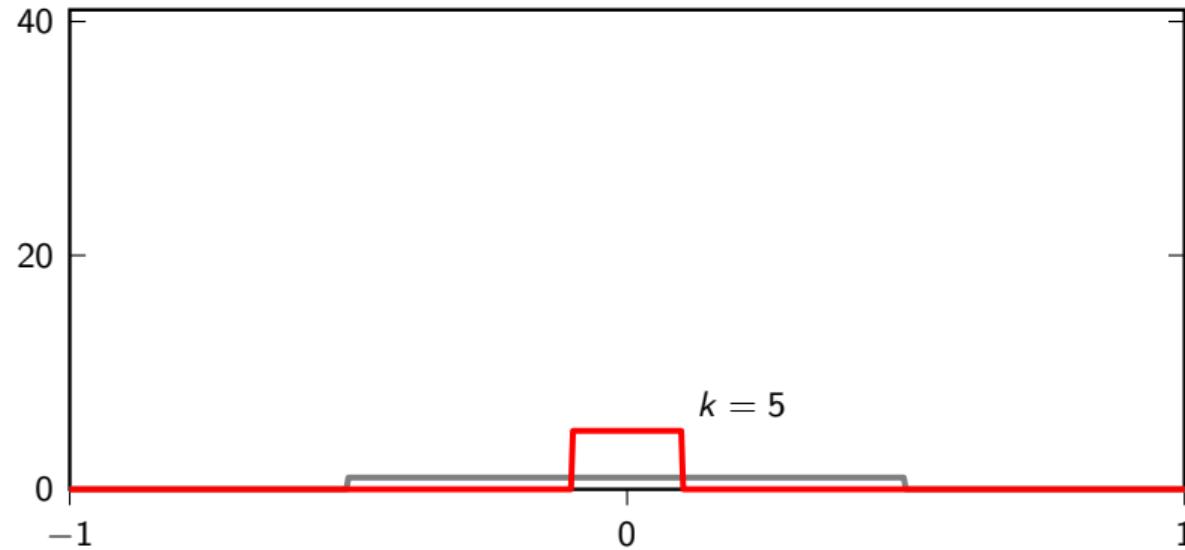
- ▶ nonzero over $[-1/2k, 1/2k]$, i.e. support is $1/k$
- ▶ area is 1

The family $r_k(t) = k \operatorname{rect}(kt)$

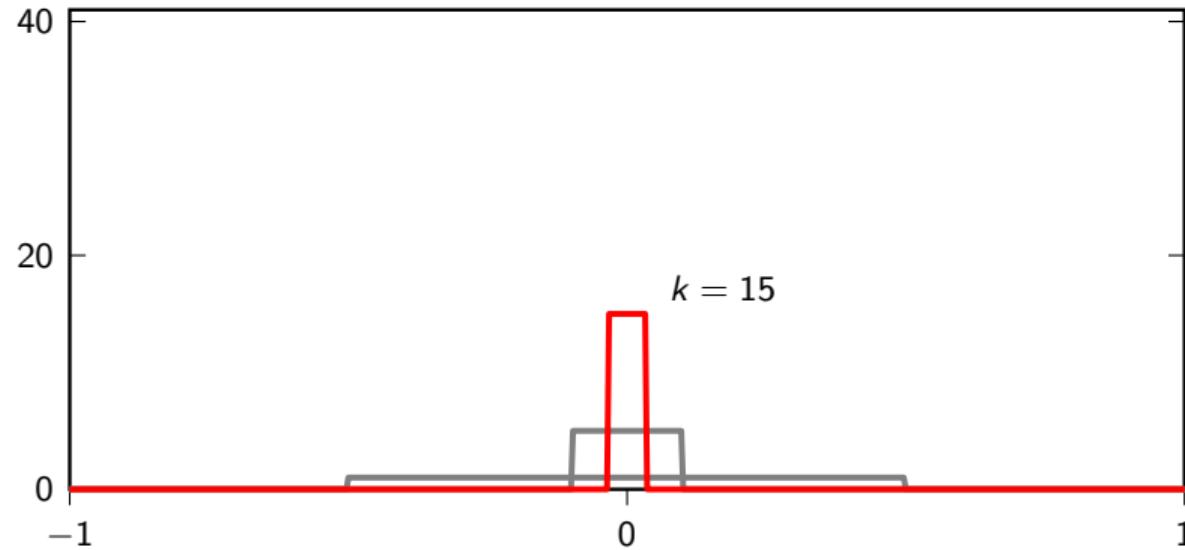


$$k = 1$$

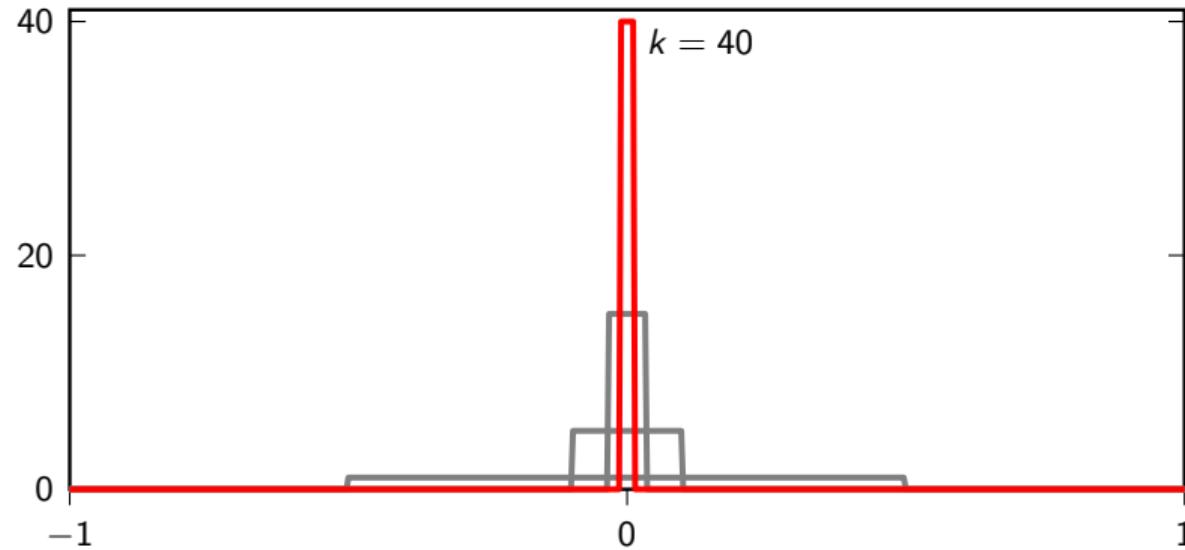
The family $r_k(t) = k \operatorname{rect}(kt)$



The family $r_k(t) = k \operatorname{rect}(kt)$



The family $r_k(t) = k \operatorname{rect}(kt)$

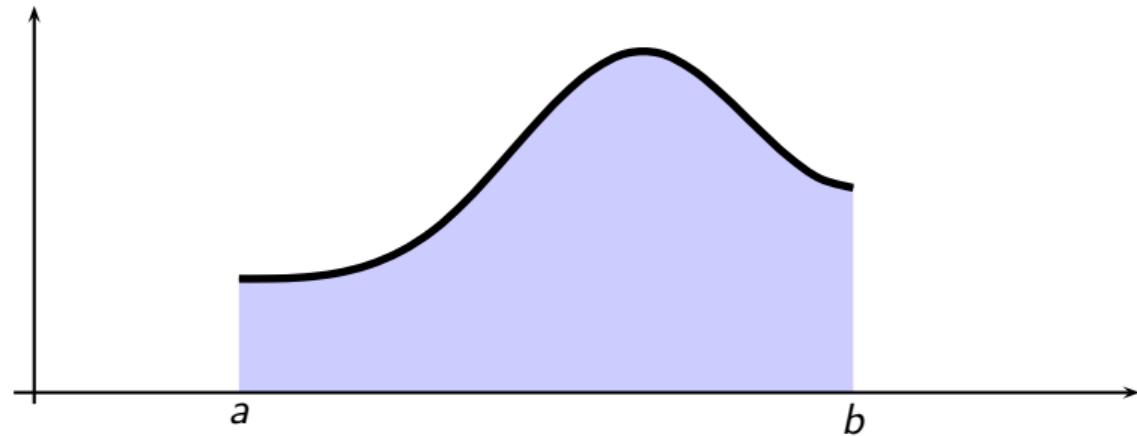


Remember the Mean Value Theorem?

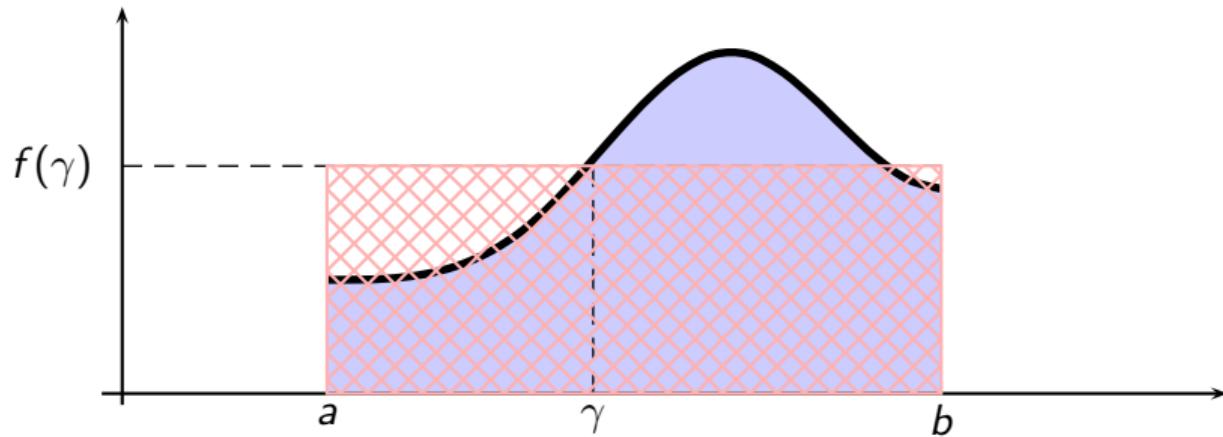
for any continuous function over the interval $[a, b]$ there exists $\gamma \in [a, b]$ s.t.

$$\int_a^b f(t)dt = (b - a) f(\gamma)$$

The Mean Value Theorem



The Mean Value Theorem



Extracting a point value

for our family of localizing functions:

$$\begin{aligned}\int_{-\infty}^{\infty} r_k(t)f(t)dt &= k \int_{-1/2k}^{1/2k} f(t)dt \\ &= f(\gamma)|_{\gamma \in [-1/2k, 1/2k]}\end{aligned}$$

and so:

$$\lim_{k \rightarrow \infty} \int_{-\infty}^{\infty} r_k(t)f(t)dt = f(0)$$

The Dirac delta functional

The delta functional is a shorthand. Instead of writing

$$\lim_{k \rightarrow \infty} \int_{-\infty}^{\infty} r_k(t-s) f(t) dt$$

we write

$$\int_{-\infty}^{\infty} \delta(t-s) f(t) dt.$$

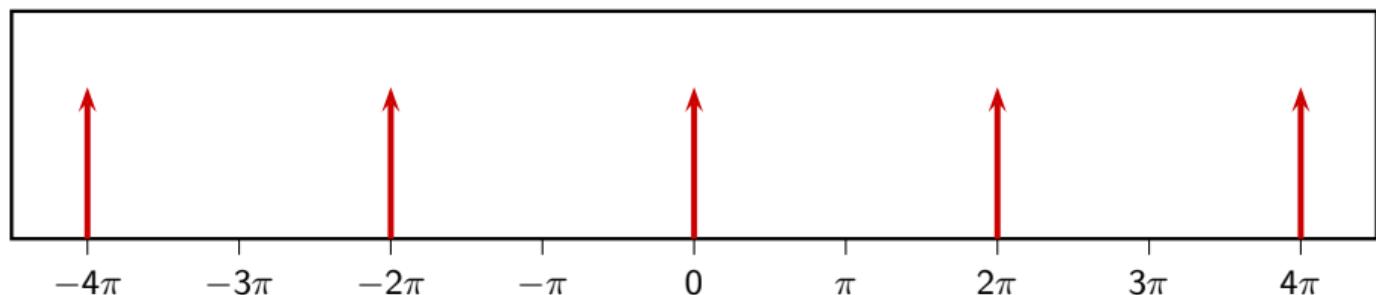
as if $\lim_{k \rightarrow \infty} r_k(t) = \delta(t)$,

The “pulse train”

$$\tilde{\delta}(\omega) = 2\pi \sum_{k=-\infty}^{\infty} \delta(\omega - 2\pi k)$$

just a technicality to use the Dirac delta in the space of 2π -periodic functions

Graphical representation



Now let the show begin!

$$\begin{aligned}\text{IDTFT} \left\{ \tilde{\delta}(\omega) \right\} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \tilde{\delta}(\omega) e^{j\omega n} d\omega \\ &= \int_{-\pi}^{\pi} \delta(\omega) e^{j\omega n} d\omega \\ &= e^{j\omega n} \Big|_{\omega=0} \\ &= 1\end{aligned}$$

Now let the show begin!

$$\begin{aligned}\text{IDTFT} \left\{ \tilde{\delta}(\omega) \right\} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \tilde{\delta}(\omega) e^{j\omega n} d\omega \\ &= \int_{-\pi}^{\pi} \delta(\omega) e^{j\omega n} d\omega \\ &= e^{j\omega n} \Big|_{\omega=0} \\ &= 1\end{aligned}$$

Now let the show begin!

$$\begin{aligned}\text{IDTFT} \left\{ \tilde{\delta}(\omega) \right\} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \tilde{\delta}(\omega) e^{j\omega n} d\omega \\ &= \int_{-\pi}^{\pi} \delta(\omega) e^{j\omega n} d\omega \\ &= e^{j\omega n} \Big|_{\omega=0} \\ &= 1\end{aligned}$$

Now let the show begin!

$$\begin{aligned}\text{IDTFT} \left\{ \tilde{\delta}(\omega) \right\} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \tilde{\delta}(\omega) e^{j\omega n} d\omega \\ &= \int_{-\pi}^{\pi} \delta(\omega) e^{j\omega n} d\omega \\ &= e^{j\omega n} \Big|_{\omega=0} \\ &= 1\end{aligned}$$

In other words

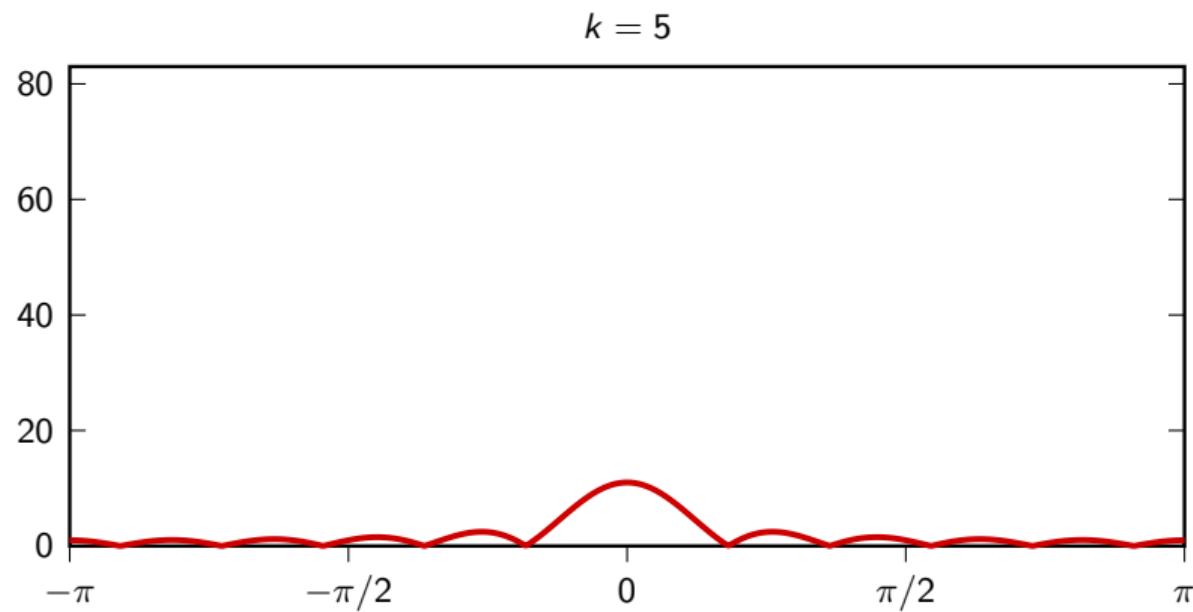
$$\text{DTFT } \{1\} = \tilde{\delta}(\omega)$$

Does it make sense?

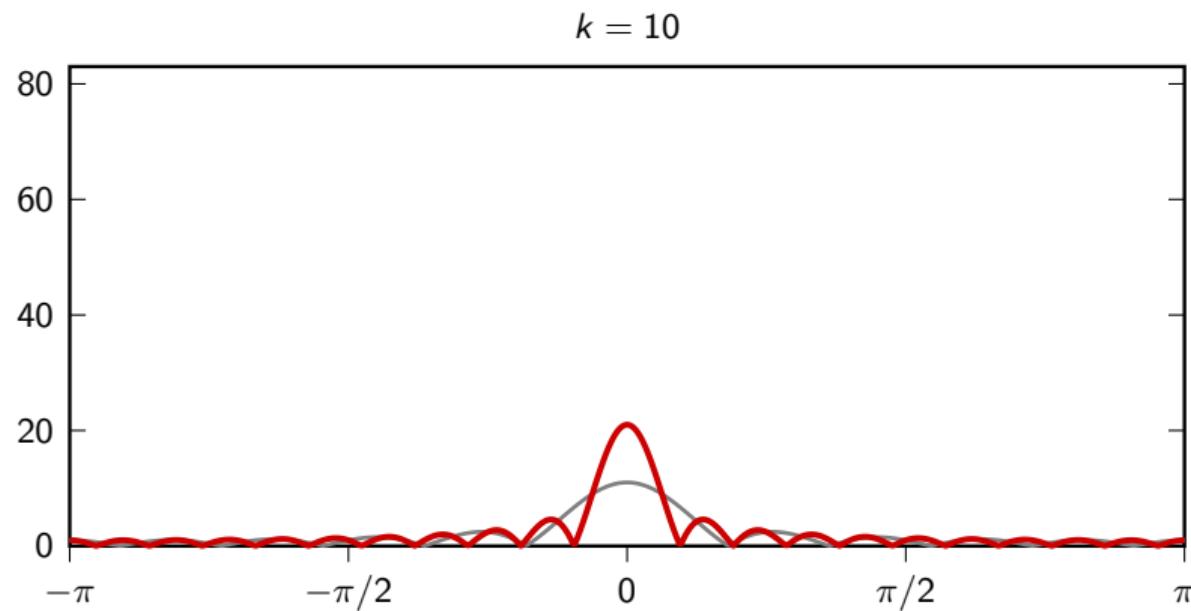
Partial DTFT sum:

$$S_k(\omega) = \sum_{n=-k}^k e^{-j\omega n}$$

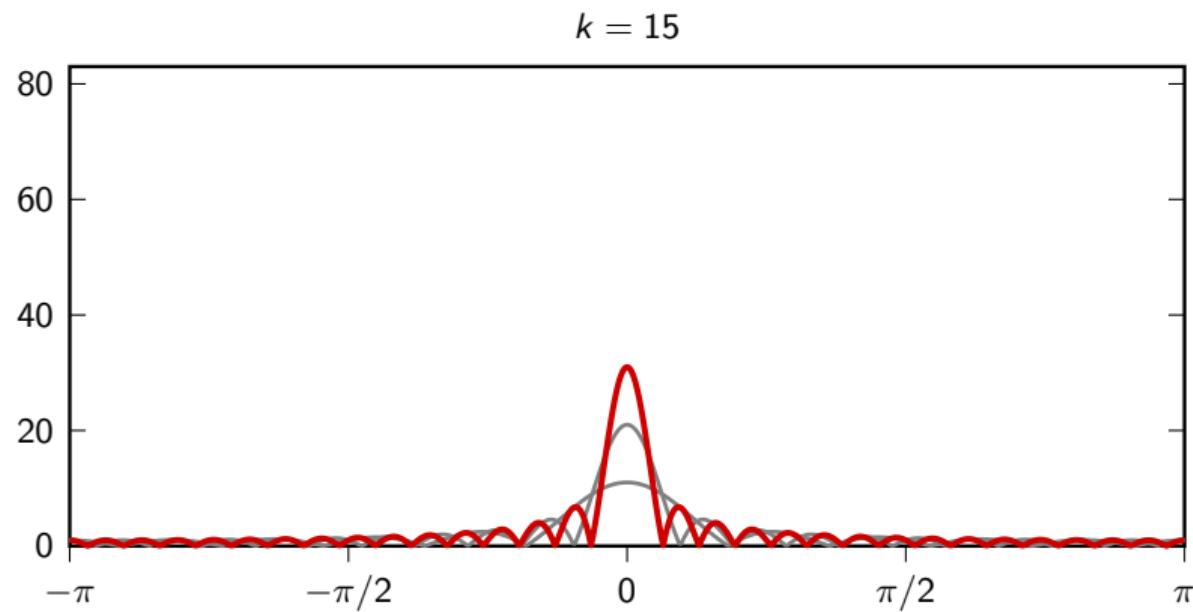
Plotting $|S_k(\omega)|$



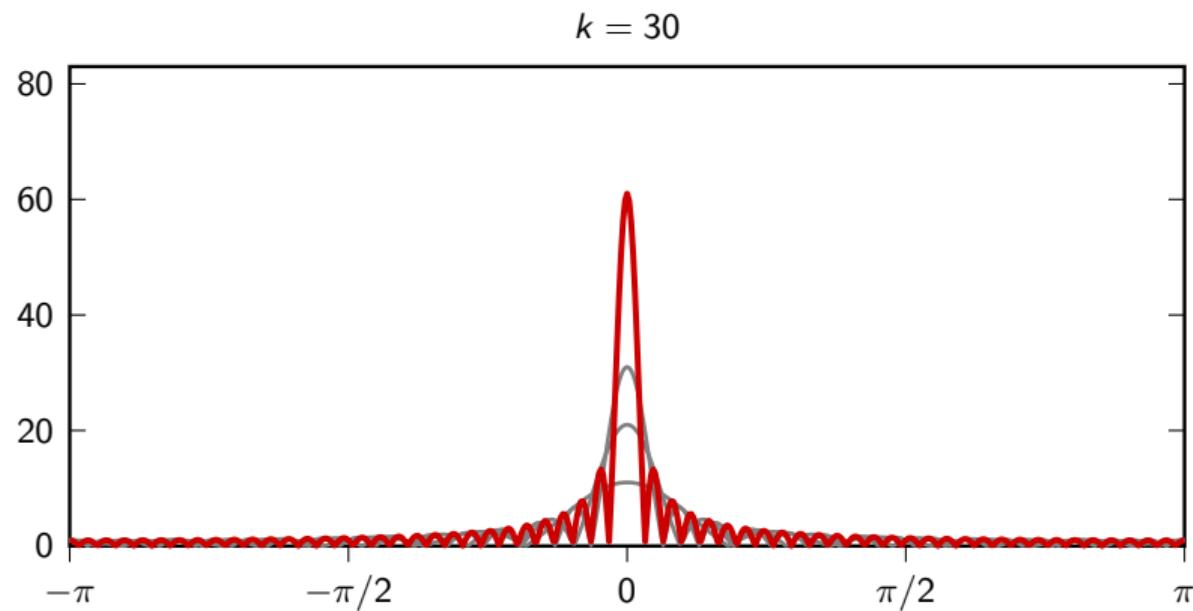
Plotting $|S_k(\omega)|$



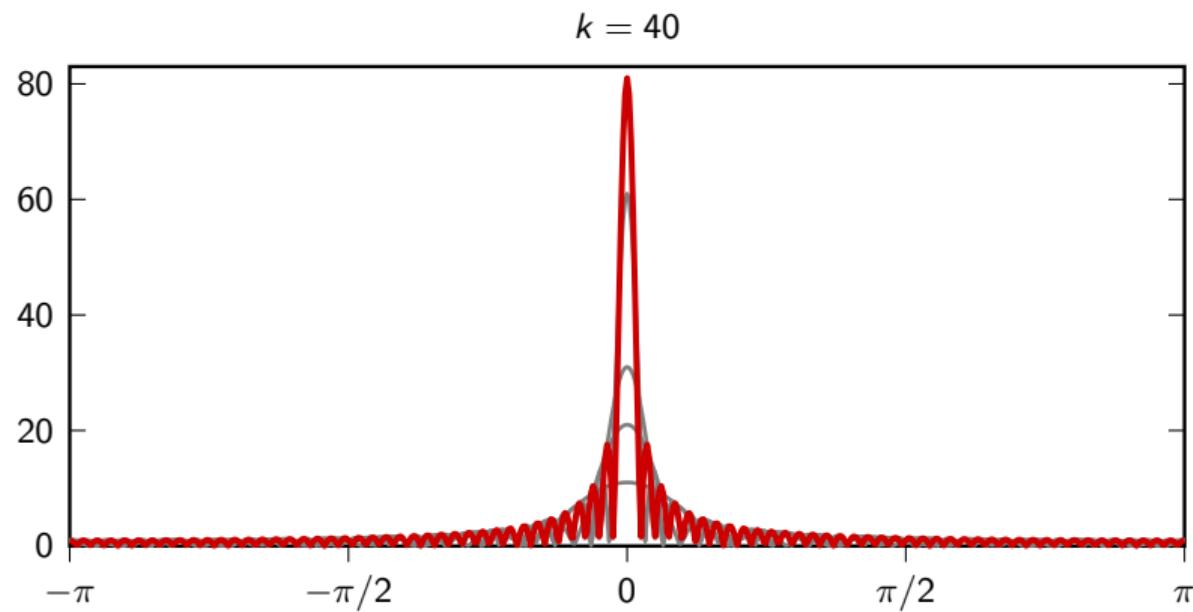
Plotting $|S_k(\omega)|$



Plotting $|S_k(\omega)|$



Plotting $|S_k(\omega)|$



Does it make sense?

Partial DTFT sums look like a family of localizing functions:

$$S_k(\omega) \rightarrow \tilde{\delta}(\omega)$$

Using the same technique

$$\text{IDTFT} \left\{ \tilde{\delta}(\omega - \omega_0) \right\} = e^{j\omega_0 n}$$

So:

- ▶ DTFT $\{1\} = \tilde{\delta}(\omega)$
- ▶ DTFT $\{e^{j\omega_0 n}\} = \tilde{\delta}(\omega - \omega_0)$
- ▶ DTFT $\{\cos \omega_0 n\} = [\tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0)]/2$
- ▶ DTFT $\{\sin \omega_0 n\} = -j[\tilde{\delta}(\omega - \omega_0) - \tilde{\delta}(\omega + \omega_0)]/2$

Using the same technique

$$\text{IDTFT} \left\{ \tilde{\delta}(\omega - \omega_0) \right\} = e^{j\omega_0 n}$$

So:

- ▶ DTFT $\{1\} = \tilde{\delta}(\omega)$
- ▶ DTFT $\{e^{j\omega_0 n}\} = \tilde{\delta}(\omega - \omega_0)$
- ▶ DTFT $\{\cos \omega_0 n\} = [\tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0)]/2$
- ▶ DTFT $\{\sin \omega_0 n\} = -j[\tilde{\delta}(\omega - \omega_0) - \tilde{\delta}(\omega + \omega_0)]/2$

Using the same technique

$$\text{IDTFT} \left\{ \tilde{\delta}(\omega - \omega_0) \right\} = e^{j\omega_0 n}$$

So:

- ▶ DTFT $\{1\} = \tilde{\delta}(\omega)$
- ▶ DTFT $\{e^{j\omega_0 n}\} = \tilde{\delta}(\omega - \omega_0)$
- ▶ DTFT $\{\cos \omega_0 n\} = [\tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0)]/2$
- ▶ DTFT $\{\sin \omega_0 n\} = -j[\tilde{\delta}(\omega - \omega_0) - \tilde{\delta}(\omega + \omega_0)]/2$

Using the same technique

$$\text{IDTFT} \left\{ \tilde{\delta}(\omega - \omega_0) \right\} = e^{j\omega_0 n}$$

So:

- ▶ DTFT $\{1\} = \tilde{\delta}(\omega)$
- ▶ DTFT $\{e^{j\omega_0 n}\} = \tilde{\delta}(\omega - \omega_0)$
- ▶ DTFT $\{\cos \omega_0 n\} = [\tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0)]/2$
- ▶ DTFT $\{\sin \omega_0 n\} = -j[\tilde{\delta}(\omega - \omega_0) - \tilde{\delta}(\omega + \omega_0)]/2$

Warning: use with caution!

- ▶ Dirac delta in the DTFT \Rightarrow signal is NOT finite-energy (eg. periodic, constant etc)
- ▶ signal must still be a power signal (finite energy over finite sections)
- ▶ Dirac deltas make sense only if integrals are involved; where are the integrals?
 - $\bar{w}[n] = 1$ only for $M \leq n \leq N \Rightarrow$ DTFT $\{x[n]\bar{w}[n]\} = \int_{-\pi}^{\pi} X(e^{j(\sigma-\omega)})W(e^{j\sigma})d\sigma$: OK

Warning: use with caution!

- ▶ Dirac delta in the DTFT \Rightarrow signal is NOT finite-energy (eg. periodic, constant etc)
- ▶ signal must still be a power signal (finite energy over finite sections)
- ▶ Dirac deltas make sense only if integrals are involved; where are the integrals?
 - $\bar{w}[n] = 1$ only for $M \leq n \leq N \Rightarrow$ DTFT $\{x[n]\bar{w}[n]\} = \int_{-\pi}^{\pi} X(e^{j(\sigma-\omega)})W(e^{j\sigma})d\sigma$: OK

Warning: use with caution!

- ▶ Dirac delta in the DTFT \Rightarrow signal is NOT finite-energy (eg. periodic, constant etc)
- ▶ signal must still be a power signal (finite energy over finite sections)
- ▶ Dirac deltas make sense only if integrals are involved; where are the integrals?
 - $\bar{w}[n] = 1$ only for $M \leq n \leq N \Rightarrow$ DTFT $\{x[n]\bar{w}[n]\} = \int_{-\pi}^{\pi} X(e^{j(\sigma-\omega)})W(e^{j\sigma})d\sigma$: OK

Warning: use with caution!

- ▶ Dirac delta in the DTFT \Rightarrow signal is NOT finite-energy (eg. periodic, constant etc)
- ▶ signal must still be a power signal (finite energy over finite sections)
- ▶ Dirac deltas make sense only if integrals are involved; where are the integrals?
 - $\bar{w}[n] = 1$ only for $M \leq n \leq N \Rightarrow$ DTFT $\{x[n]\bar{w}[n]\} = \int_{-\pi}^{\pi} X(e^{j(\sigma-\omega)})W(e^{j\sigma})d\sigma$: OK

relationships between transforms

Overview:

- ▶ DFT, DFS, DTFT
- ▶ DTFT of periodic sequences
- ▶ DTFT of finite-support sequences
- ▶ Zero padding

Transforms

- ▶ DFT, DFS: change of basis in \mathbb{C}^N
- ▶ DTFT: “formal” change of basis in $\ell_2(\mathbb{Z})$
- ▶ basis vectors are “building blocks” for any signal
- ▶ DFT: numerical algorithm (computable)
- ▶ DTFT: mathematical tool (proofs)

Transforms

- ▶ DFT, DFS: change of basis in \mathbb{C}^N
- ▶ DTFT: “formal” change of basis in $\ell_2(\mathbb{Z})$
- ▶ basis vectors are “building blocks” for any signal
- ▶ DFT: numerical algorithm (computable)
- ▶ DTFT: mathematical tool (proofs)

Transforms

- ▶ DFT, DFS: change of basis in \mathbb{C}^N
 - ▶ DTFT: “formal” change of basis in $\ell_2(\mathbb{Z})$
 - ▶ basis vectors are “building blocks” for any signal
-
- ▶ DFT: numerical algorithm (computable)
 - ▶ DTFT: mathematical tool (proofs)

Transforms

- ▶ DFT, DFS: change of basis in \mathbb{C}^N
- ▶ DTFT: “formal” change of basis in $\ell_2(\mathbb{Z})$
- ▶ basis vectors are “building blocks” for any signal
- ▶ DFT: numerical algorithm (computable)
- ▶ DTFT: mathematical tool (proofs)

Transforms

- ▶ DFT, DFS: change of basis in \mathbb{C}^N
- ▶ DTFT: “formal” change of basis in $\ell_2(\mathbb{Z})$
- ▶ basis vectors are “building blocks” for any signal

- ▶ DFT: numerical algorithm (computable)
- ▶ DTFT: mathematical tool (proofs)

Embedding finite-length signals

- ▶ N -tap signal $x[n]$
- ▶ natural spectral representation: DFT $X[k]$
- ▶ two ways to embed $x[n]$ into an infinite sequence:
 - periodic extension: $\tilde{x}[n] = x[n \bmod N]$
 - finite-support extension: $\tilde{x}[n] = \begin{cases} x[n] & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$
- ▶ how does $X[k]$ relate to the DTFT of the embedded signals?

Embedding finite-length signals

- ▶ N -tap signal $x[n]$
- ▶ natural spectral representation: DFT $X[k]$
- ▶ two ways to embed $x[n]$ into an infinite sequence:
 - periodic extension: $\tilde{x}[n] = x[n \bmod N]$
 - finite-support extension: $\tilde{x}[n] = \begin{cases} x[n] & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$
- ▶ how does $X[k]$ relate to the DTFT of the embedded signals?

Embedding finite-length signals

- ▶ N -tap signal $x[n]$
- ▶ natural spectral representation: DFT $X[k]$
- ▶ two ways to embed $x[n]$ into an infinite sequence:
 - periodic extension: $\tilde{x}[n] = x[n \bmod N]$
 - finite-support extension: $\bar{x}[n] = \begin{cases} x[n] & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$
- ▶ how does $X[k]$ relate to the DTFT of the embedded signals?

Embedding finite-length signals

- ▶ N -tap signal $x[n]$
- ▶ natural spectral representation: DFT $X[k]$
- ▶ two ways to embed $x[n]$ into an infinite sequence:
 - periodic extension: $\tilde{x}[n] = x[n \bmod N]$
 - finite-support extension: $\bar{x}[n] = \begin{cases} x[n] & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$
- ▶ how does $X[k]$ relate to the DTFT of the embedded signals?

Embedding finite-length signals

- ▶ N -tap signal $x[n]$
- ▶ natural spectral representation: DFT $X[k]$
- ▶ two ways to embed $x[n]$ into an infinite sequence:
 - periodic extension: $\tilde{x}[n] = x[n \bmod N]$
 - finite-support extension: $\bar{x}[n] = \begin{cases} x[n] & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$
- ▶ how does $X[k]$ relate to the DTFT of the embedded signals?

Embedding finite-length signals

- ▶ N -tap signal $x[n]$
- ▶ natural spectral representation: DFT $X[k]$
- ▶ two ways to embed $x[n]$ into an infinite sequence:
 - periodic extension: $\tilde{x}[n] = x[n \bmod N]$
 - finite-support extension: $\bar{x}[n] = \begin{cases} x[n] & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$
- ▶ how does $X[k]$ relate to the DTFT of the embedded signals?

DTFT of periodic signals

$$\tilde{x}[n] = x[n \bmod N]$$

$$\begin{aligned}\tilde{X}(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} \tilde{x}[n] e^{-j\omega n} \\ &= \sum_{n=-\infty}^{\infty} \left(\frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}[k] e^{j\frac{2\pi}{N} nk} \right) e^{-j\omega n} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}[k] \left(\sum_{n=-\infty}^{\infty} e^{j\frac{2\pi}{N} nk} e^{-j\omega n} \right)\end{aligned}$$

DTFT of periodic signals

$$\tilde{x}[n] = x[n \bmod N]$$

$$\begin{aligned}\tilde{X}(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} \tilde{x}[n] e^{-j\omega n} \\ &= \sum_{n=-\infty}^{\infty} \left(\frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}[k] e^{j\frac{2\pi}{N} nk} \right) e^{-j\omega n} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}[k] \left(\sum_{n=-\infty}^{\infty} e^{j\frac{2\pi}{N} nk} e^{-j\omega n} \right)\end{aligned}$$

DTFT of periodic signals

$$\tilde{x}[n] = x[n \bmod N]$$

$$\begin{aligned}\tilde{X}(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} \tilde{x}[n] e^{-j\omega n} \\ &= \sum_{n=-\infty}^{\infty} \left(\frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}[k] e^{j\frac{2\pi}{N} nk} \right) e^{-j\omega n} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}[k] \left(\sum_{n=-\infty}^{\infty} e^{j\frac{2\pi}{N} nk} e^{-j\omega n} \right)\end{aligned}$$

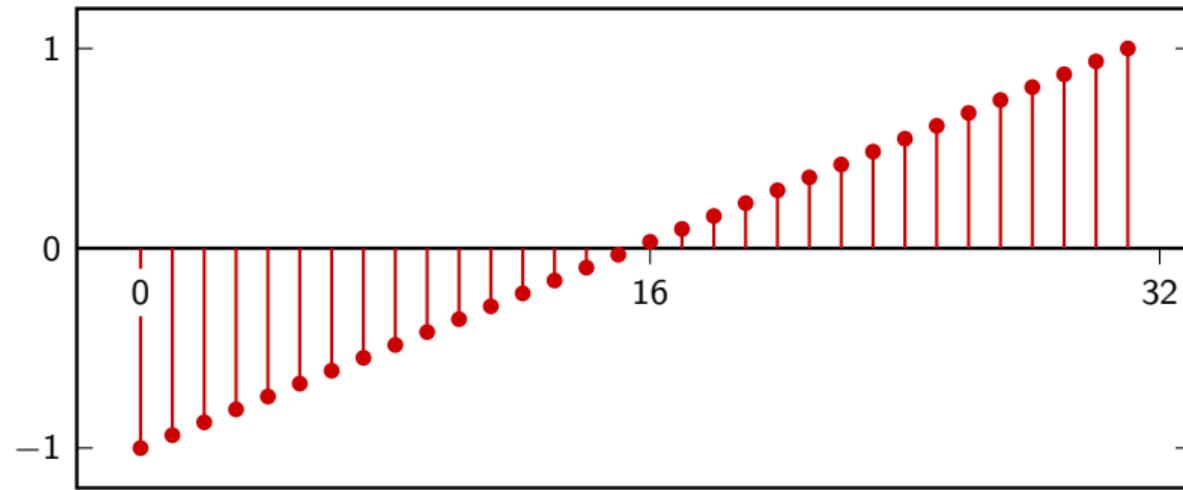
We've seen this before

$$\sum_{n=-\infty}^{\infty} e^{j\frac{2\pi}{N}nk} e^{-j\omega n} = \text{DTFT} \left\{ e^{j\frac{2\pi}{N}nk} \right\}$$
$$= \tilde{\delta}\left(\omega - \frac{2\pi}{N}k\right)$$

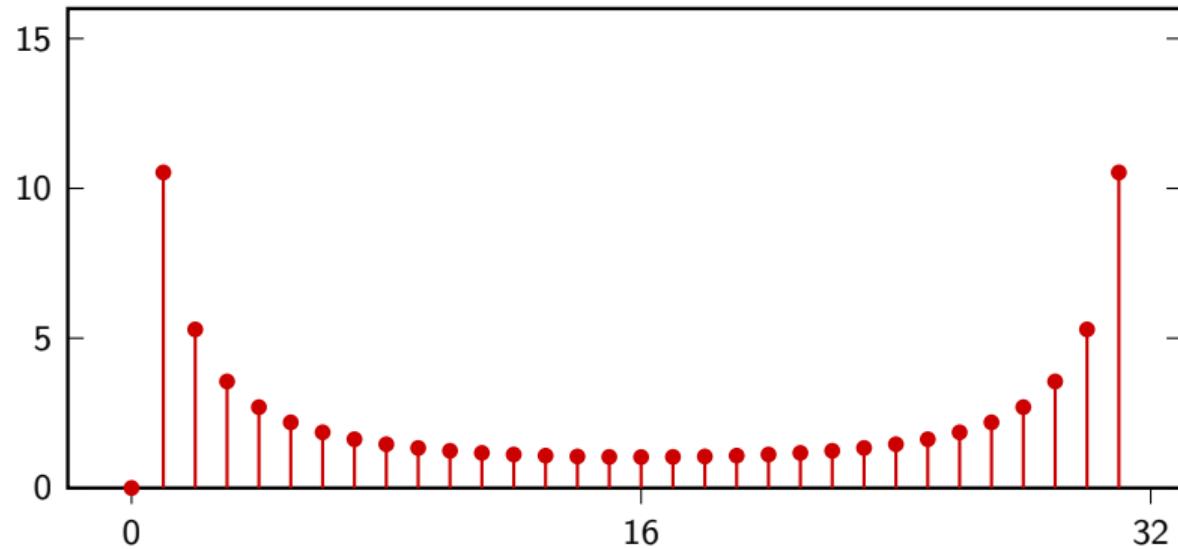
DTFT of periodic signals

$$\tilde{X}(e^{j\omega}) = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \tilde{\delta}\left(\omega - \frac{2\pi}{N}k\right)$$

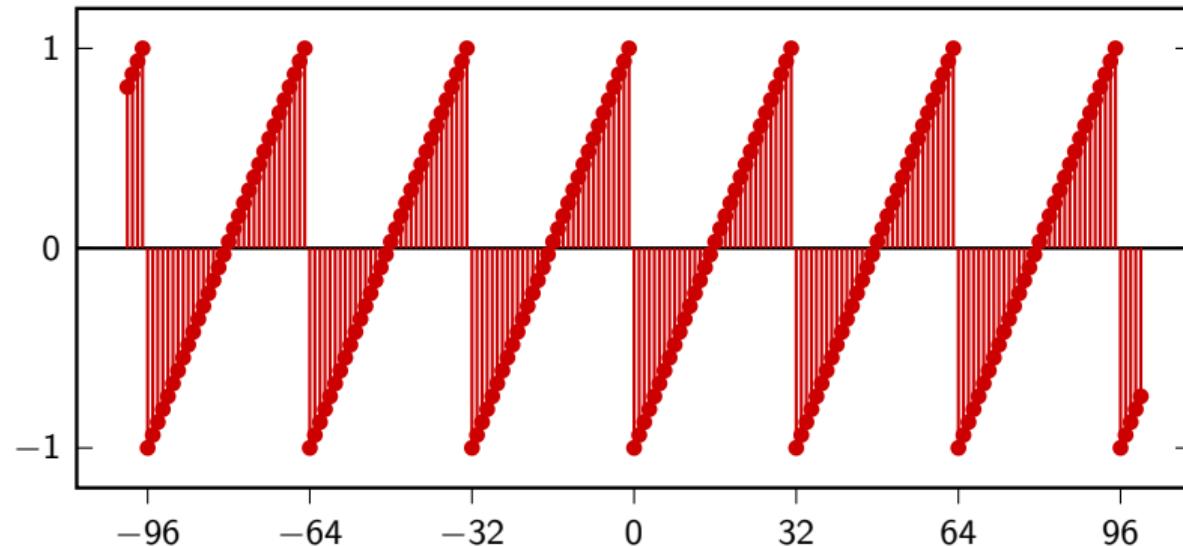
32-tap sawtooth



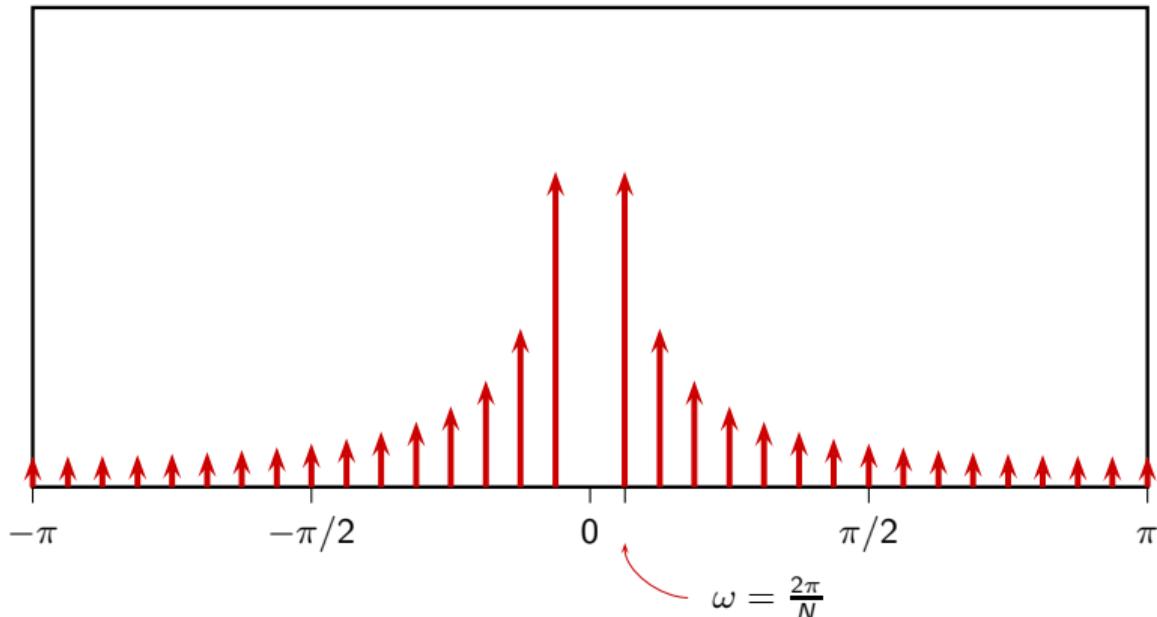
DFT of 32-tap sawtooth



32-periodic sawtooth



DTFT of periodic extension



DTFT of finite-support signals

$$\bar{x}[n] = \begin{cases} x[n] & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned}\bar{X}(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} \bar{x}[n] e^{-j\omega n} = \sum_{n=0}^{N-1} x[n] e^{-j\omega n} \\ &= \sum_{n=0}^{N-1} \left(\frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N} nk} \right) e^{-j\omega n} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] \left(\sum_{n=0}^{N-1} e^{-j(\omega - \frac{2\pi}{N} k)n} \right)\end{aligned}$$

DTFT of finite-support signals

$$\bar{x}[n] = \begin{cases} x[n] & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned}\bar{X}(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} \bar{x}[n] e^{-j\omega n} = \sum_{n=0}^{N-1} x[n] e^{-j\omega n} \\ &= \sum_{n=0}^{N-1} \left(\frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N} nk} \right) e^{-j\omega n} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] \left(\sum_{n=0}^{N-1} e^{-j(\omega - \frac{2\pi}{N} k)n} \right)\end{aligned}$$

DTFT of finite-support signals

$$\bar{x}[n] = \begin{cases} x[n] & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned}\bar{X}(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} \bar{x}[n] e^{-j\omega n} = \sum_{n=0}^{N-1} x[n] e^{-j\omega n} \\ &= \sum_{n=0}^{N-1} \left(\frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N} nk} \right) e^{-j\omega n} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] \left(\sum_{n=0}^{N-1} e^{-j(\omega - \frac{2\pi}{N} k)n} \right)\end{aligned}$$

DTFT of finite-support signals

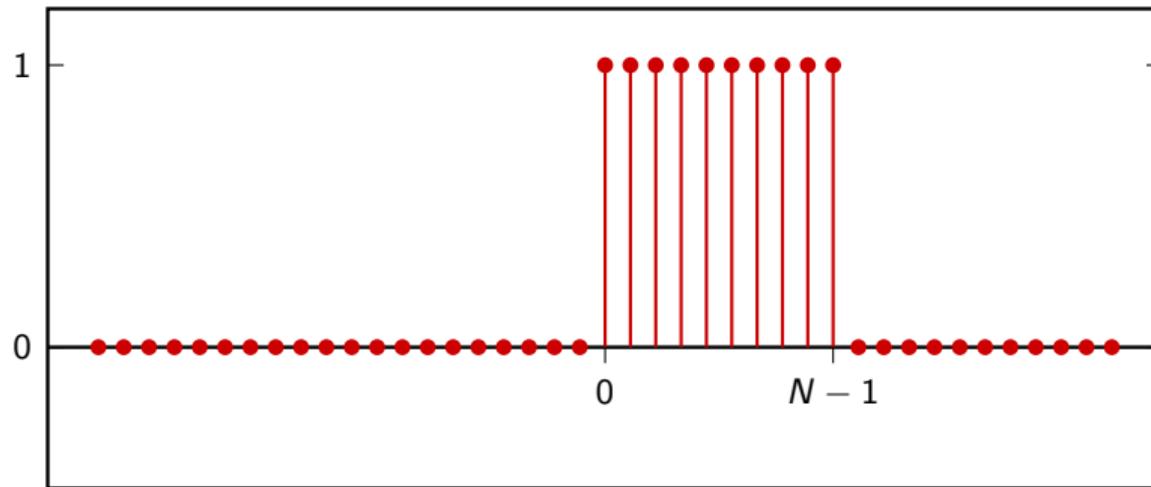
$$\sum_{n=0}^{N-1} e^{-j(\omega - \frac{2\pi}{N}k)n} = \bar{R}(e^{j(\omega - \frac{2\pi}{N}k)})$$

where $\bar{R}(e^{j\omega})$ is the DTFT of $\bar{r}[n]$, the rectangular signal:

$$\bar{r}[n] = \begin{cases} 1 & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$$

Rectangular step signal

$$\bar{r}[n] = \begin{cases} 1 & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$$



DTFT of rectangular step signal

$$\begin{aligned}\bar{R}(e^{j\omega}) &= \sum_{n=0}^{N-1} e^{-j\omega n} \\&= \frac{1 - e^{-j\omega N}}{1 - e^{-j\omega}} \\&= \frac{e^{-j\frac{\omega N}{2}} \left[e^{j\frac{\omega N}{2}} - e^{-j\frac{\omega N}{2}} \right]}{e^{-j\frac{\omega}{2}} \left[e^{j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}} \right]} \\&= \frac{\sin\left(\frac{\omega}{2}N\right)}{\sin\left(\frac{\omega}{2}\right)} e^{-j\frac{\omega}{2}(N-1)}\end{aligned}$$

DTFT of rectangular step signal

$$\begin{aligned}\bar{R}(e^{j\omega}) &= \sum_{n=0}^{N-1} e^{-j\omega n} \\&= \frac{1 - e^{-j\omega N}}{1 - e^{-j\omega}} \\&= \frac{e^{-j\frac{\omega N}{2}} \left[e^{j\frac{\omega N}{2}} - e^{-j\frac{\omega N}{2}} \right]}{e^{-j\frac{\omega}{2}} \left[e^{j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}} \right]} \\&= \frac{\sin\left(\frac{\omega}{2}N\right)}{\sin\left(\frac{\omega}{2}\right)} e^{-j\frac{\omega}{2}(N-1)}\end{aligned}$$

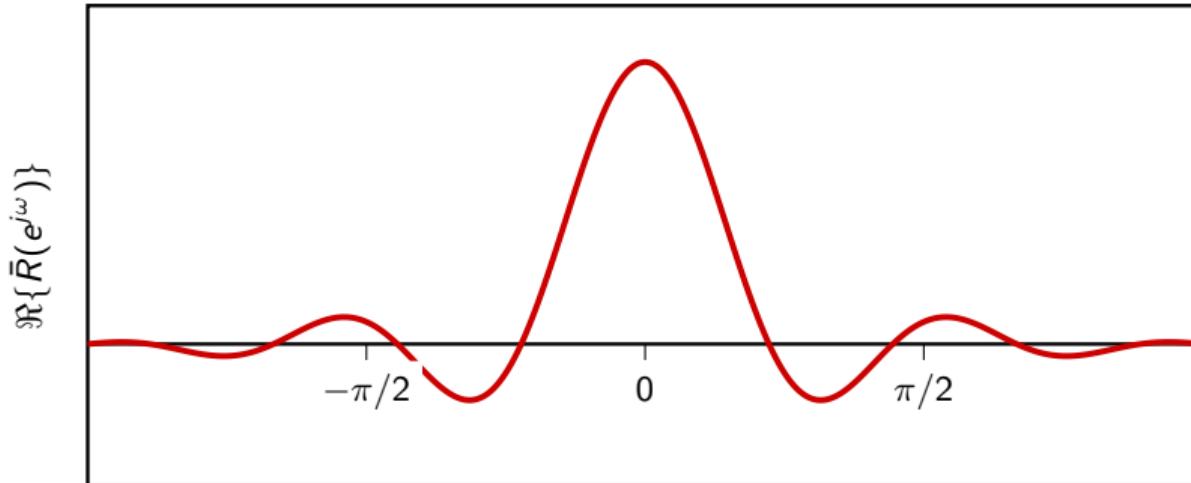
DTFT of rectangular step signal

$$\begin{aligned}\bar{R}(e^{j\omega}) &= \sum_{n=0}^{N-1} e^{-j\omega n} \\&= \frac{1 - e^{-j\omega N}}{1 - e^{-j\omega}} \\&= \frac{e^{-j\frac{\omega N}{2}} \left[e^{j\frac{\omega N}{2}} - e^{-j\frac{\omega N}{2}} \right]}{e^{-j\frac{\omega}{2}} \left[e^{j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}} \right]} \\&= \frac{\sin\left(\frac{\omega}{2}N\right)}{\sin\left(\frac{\omega}{2}\right)} e^{-j\frac{\omega}{2}(N-1)}\end{aligned}$$

DTFT of rectangular step signal

$$\begin{aligned}\bar{R}(e^{j\omega}) &= \sum_{n=0}^{N-1} e^{-j\omega n} \\&= \frac{1 - e^{-j\omega N}}{1 - e^{-j\omega}} \\&= \frac{e^{-j\frac{\omega N}{2}} \left[e^{j\frac{\omega N}{2}} - e^{-j\frac{\omega N}{2}} \right]}{e^{-j\frac{\omega}{2}} \left[e^{j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}} \right]} \\&= \frac{\sin\left(\frac{\omega}{2}N\right)}{\sin\left(\frac{\omega}{2}\right)} e^{-j\frac{\omega}{2}(N-1)}\end{aligned}$$

DTFT of interval signal ($N = 9$)



note that $\bar{R}(e^{j\omega}) = 0$ for $\omega = 2\pi k/N$, $k \in \mathbb{Z}/\{0\}$

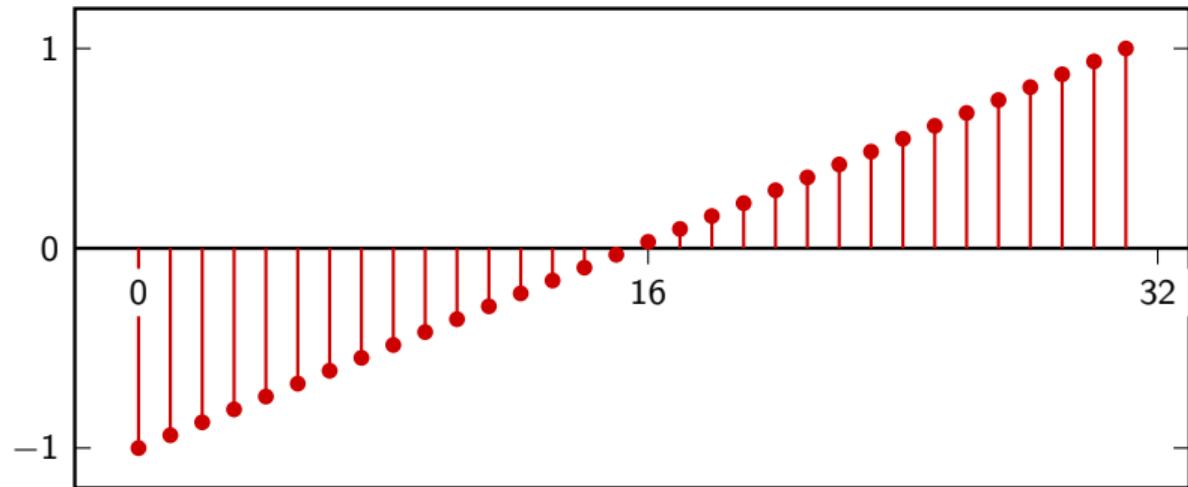
DTFT of finite-support signals

define $\Lambda(\omega) = \frac{1}{N} \bar{R}(e^{j\omega})$

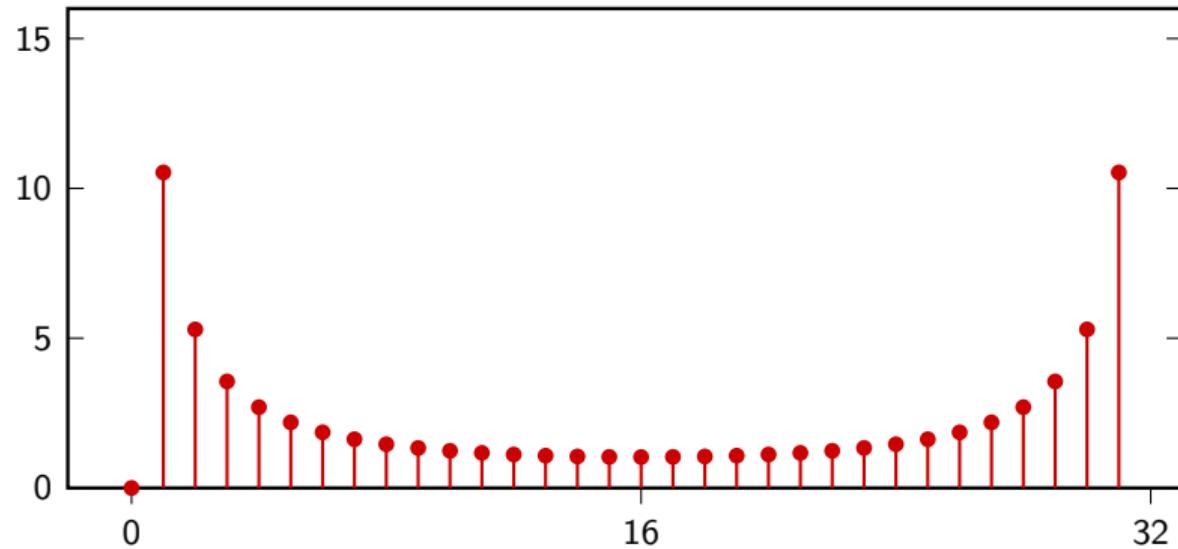
$$\bar{X}(e^{j\omega}) = \sum_{k=0}^{N-1} X[k] \Lambda(\omega - \frac{2\pi}{N} k)$$

smooth interpolation of DFT values

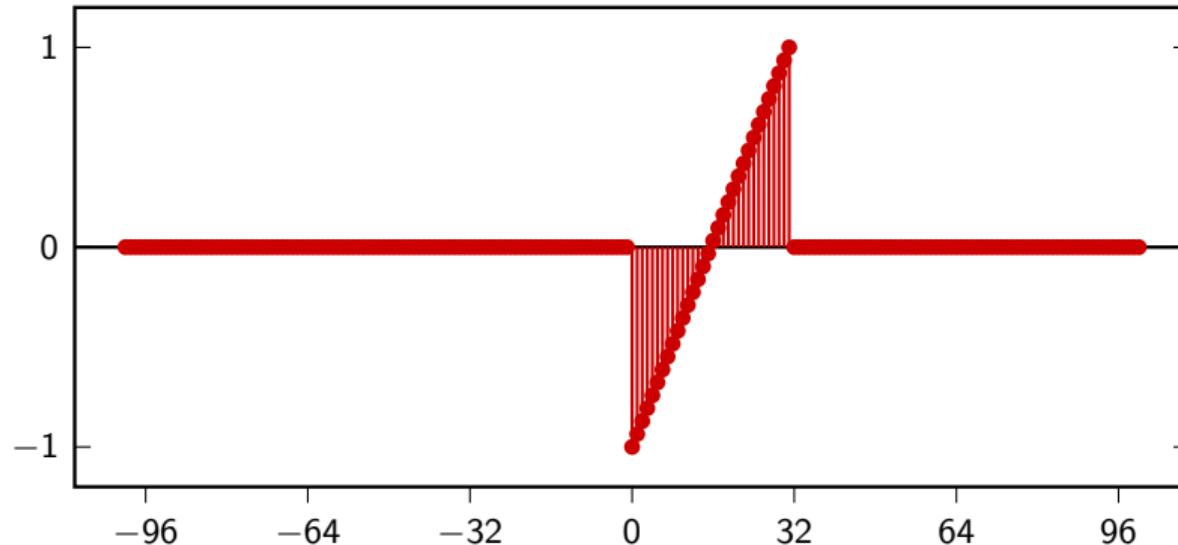
32-tap sawtooth



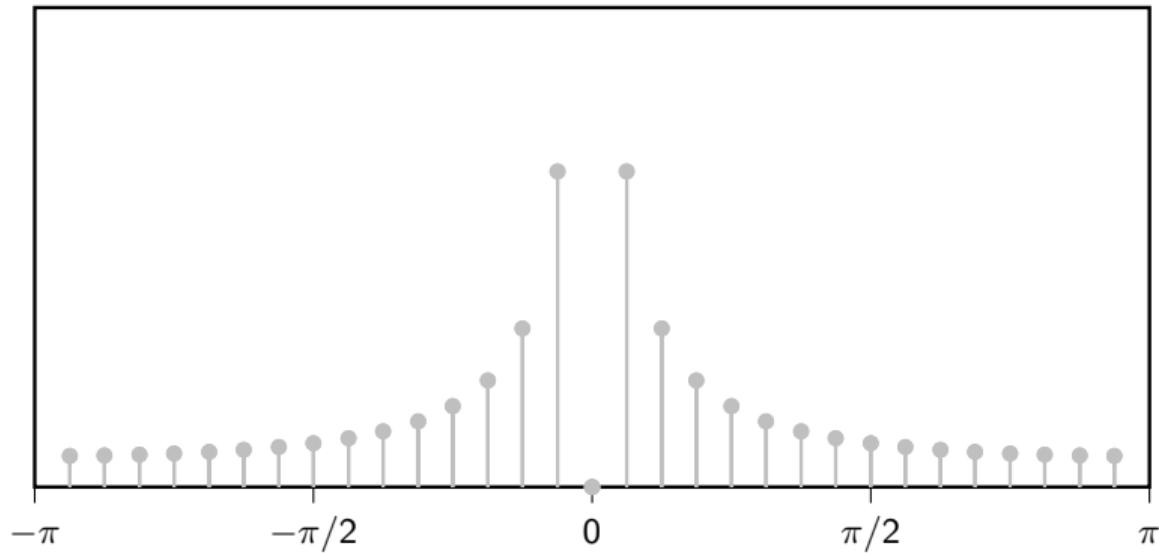
DFT of 32-tap sawtooth



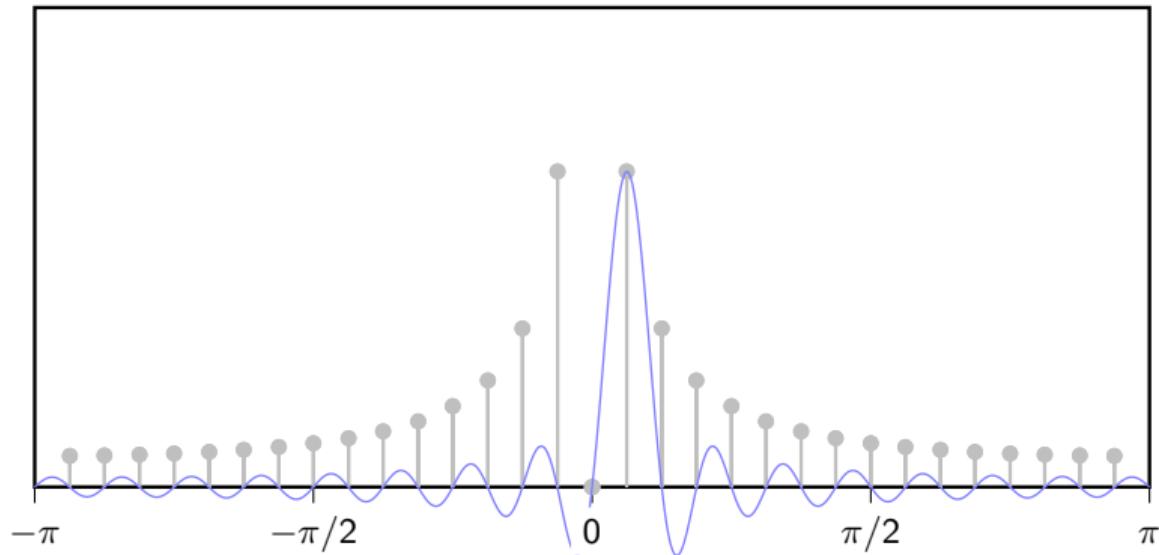
Sawtooth: finite support extension



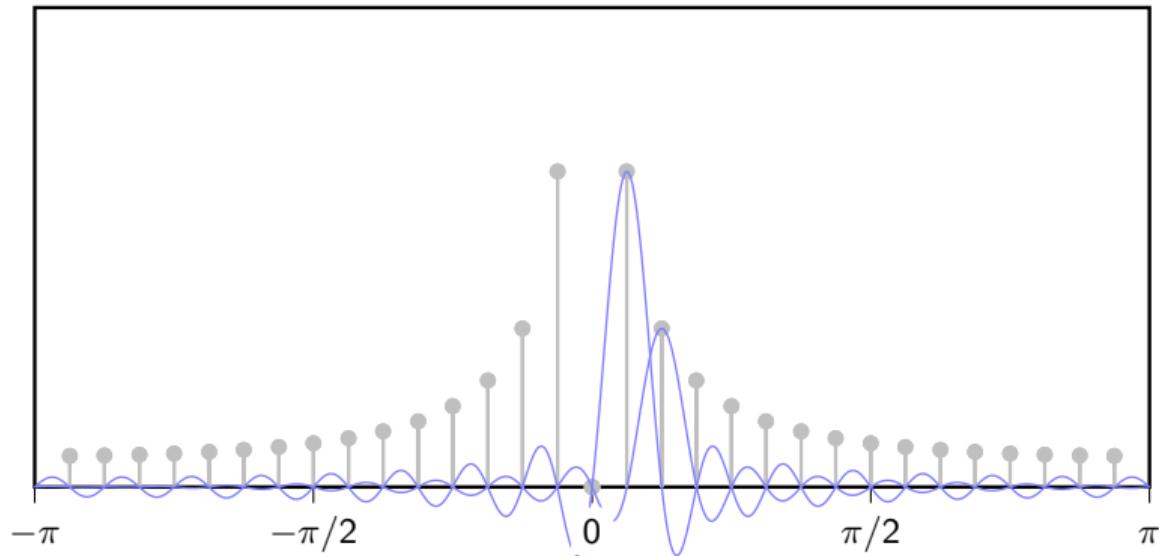
DTFT of finite support extension (sketch)



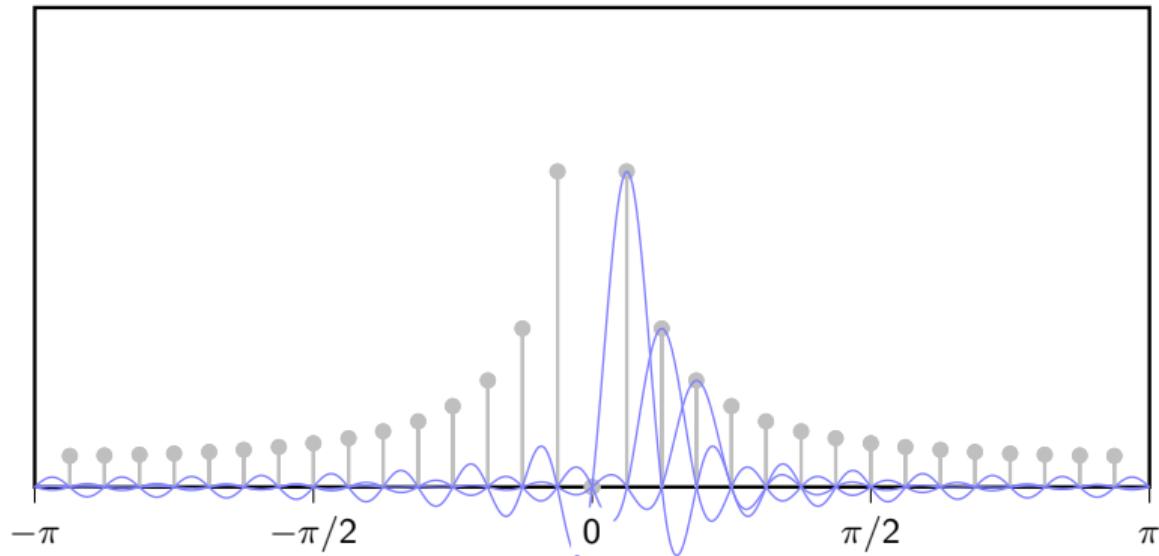
DTFT of finite support extension (sketch)



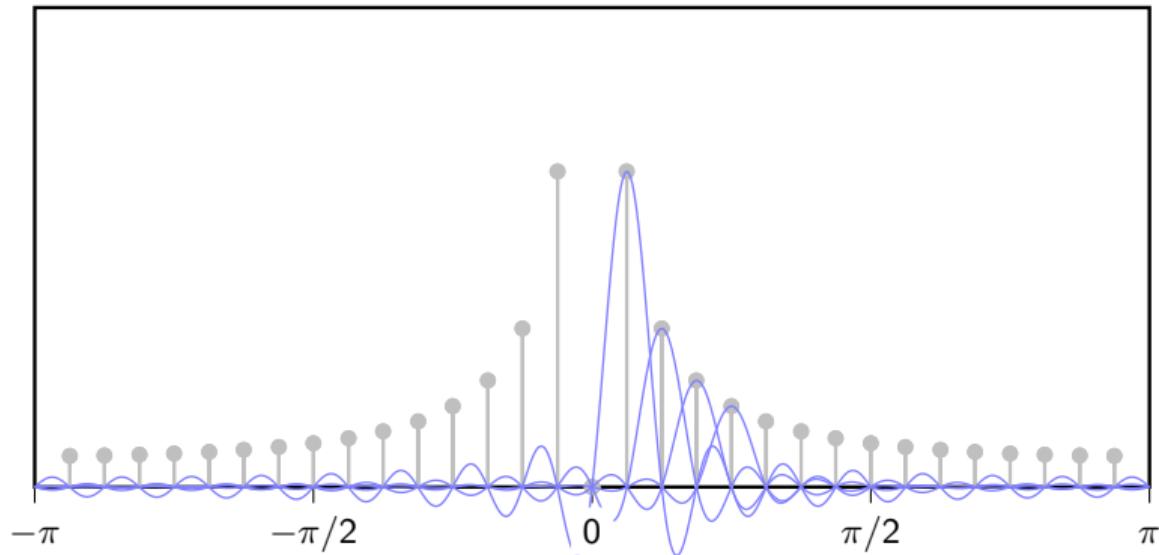
DTFT of finite support extension (sketch)



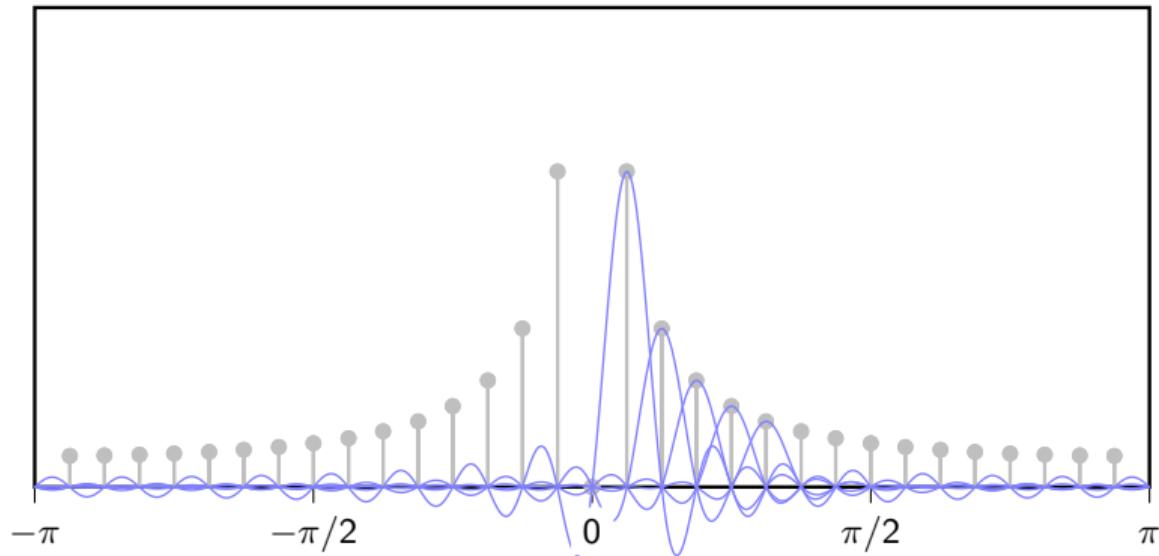
DTFT of finite support extension (sketch)



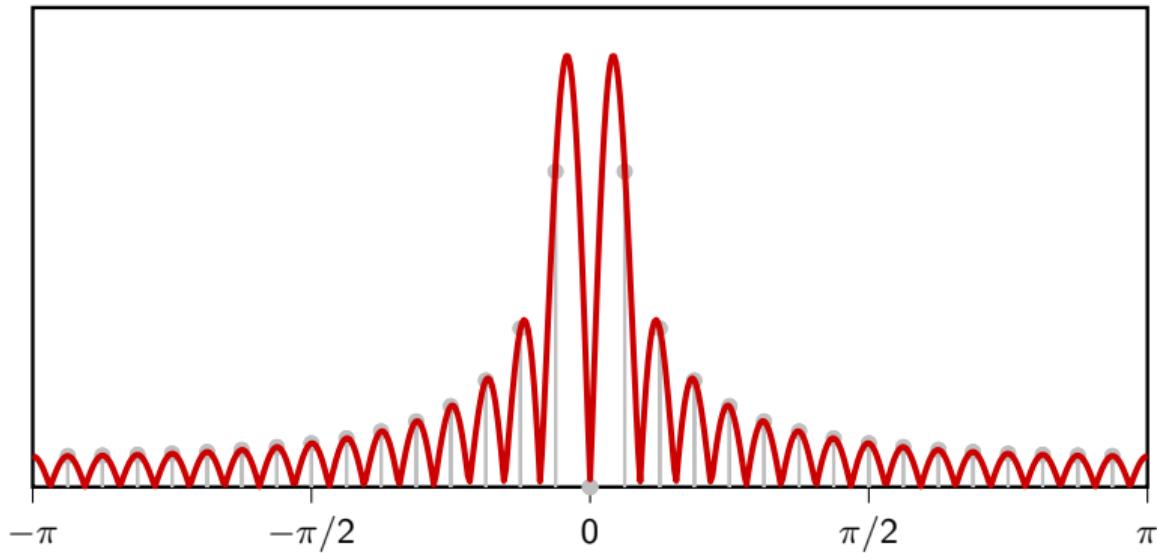
DTFT of finite support extension (sketch)



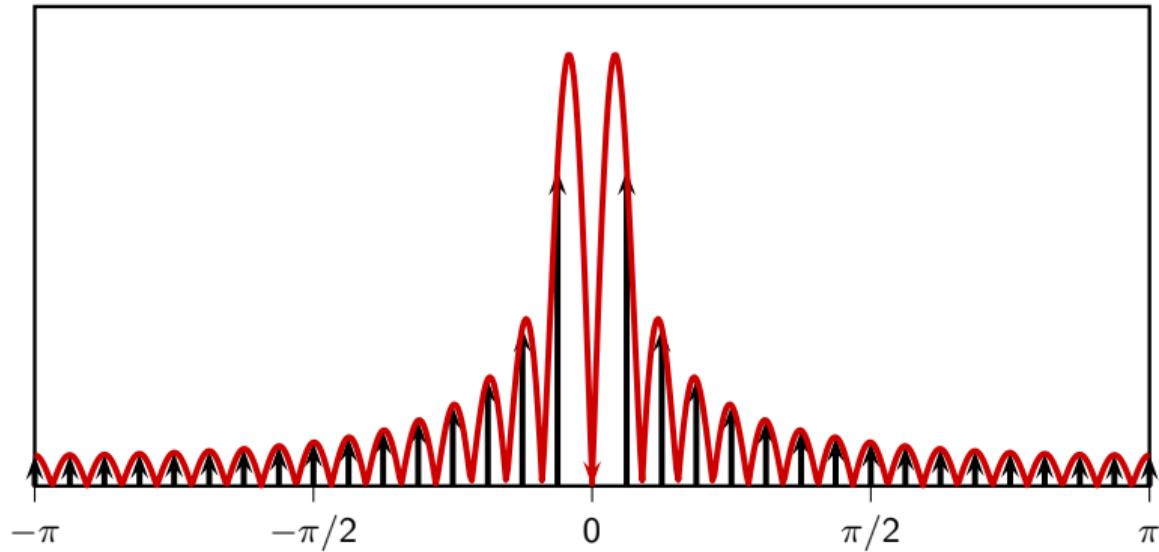
DTFT of finite support extension (sketch)



DTFT of finite support extension



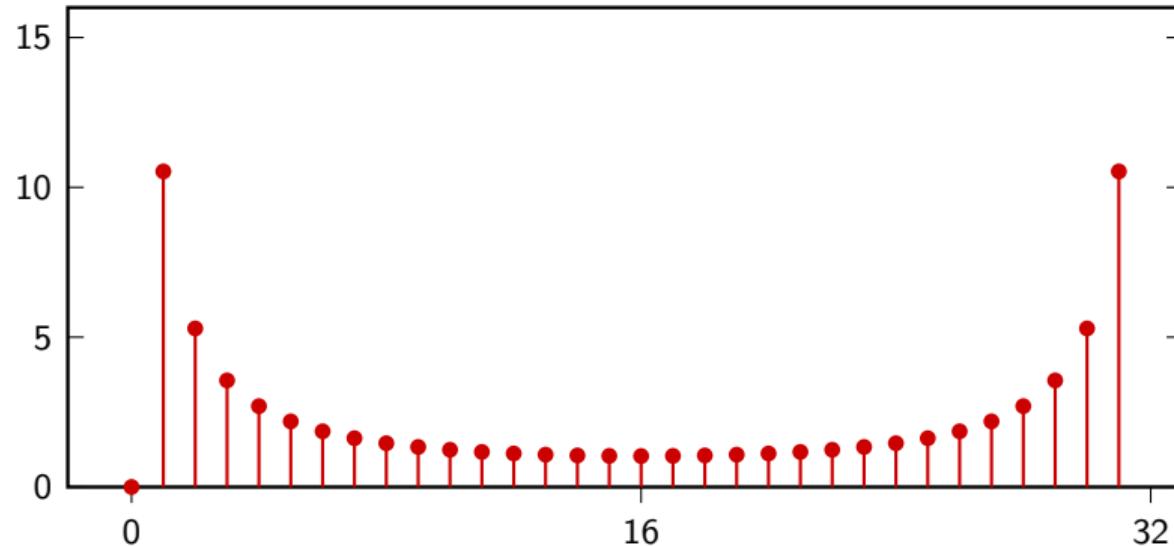
As a comparison...



About zero-padding

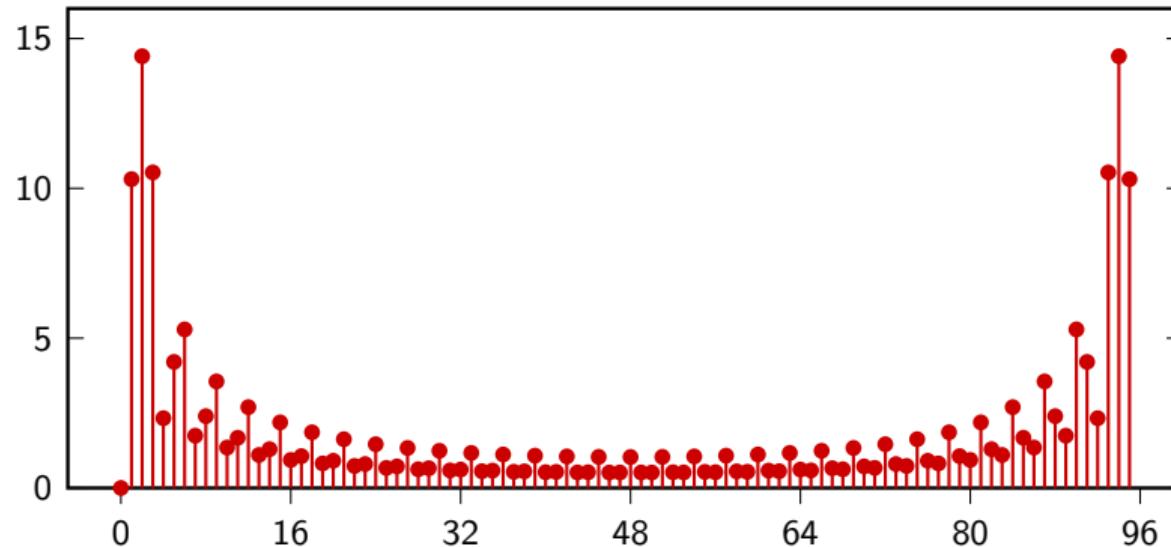
When computing the DFT numerically
one may “pad” the data vector with zeros to obtain “nicer” plots

DFT of 32-tap sawtooth



$$\mathbf{x} = [x_0 \ x_1 \ \dots \ x_{31}]$$

DFT of 32-tap sawtooth, zero-padded to 96 points



$$\mathbf{x} = [x_0 \ x_1 \ \dots \ x_{31} \ 0 \ \dots \ 0]$$

About zero-padding

$$x_M[n] = \begin{cases} x[n] & \text{for } 0 \leq n < N \\ 0 & \text{for } N \leq n < M \end{cases}$$

About zero-padding

$$\begin{aligned} X_M[h] &= \sum_{n=0}^{M-1} x'[n] e^{-j\frac{2\pi}{M} nh} = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{M} nh} \\ &= \sum_{n=0}^{N-1} \left(\frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N} nk} \right) e^{-j\frac{2\pi}{M} nh} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] \left(\sum_{n=0}^{N-1} e^{-j(\frac{2\pi}{M} h - \frac{2\pi}{N} k)n} \right) \\ &= \bar{X}(e^{j\omega})|_{\omega=\frac{2\pi}{M} h} \end{aligned}$$

About zero-padding

$$\begin{aligned} X_M[h] &= \sum_{n=0}^{M-1} x'[n] e^{-j\frac{2\pi}{M} nh} = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{M} nh} \\ &= \sum_{n=0}^{N-1} \left(\frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N} nk} \right) e^{-j\frac{2\pi}{M} nh} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] \left(\sum_{n=0}^{N-1} e^{-j(\frac{2\pi}{M} h - \frac{2\pi}{N} k)n} \right) \\ &= \bar{X}(e^{j\omega})|_{\omega=\frac{2\pi}{M} h} \end{aligned}$$

About zero-padding

$$\begin{aligned} X_M[h] &= \sum_{n=0}^{M-1} x'[n] e^{-j\frac{2\pi}{M} nh} = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{M} nh} \\ &= \sum_{n=0}^{N-1} \left(\frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N} nk} \right) e^{-j\frac{2\pi}{M} nh} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] \left(\sum_{n=0}^{N-1} e^{-j(\frac{2\pi}{M} h - \frac{2\pi}{N} k)n} \right) \\ &= \bar{X}(e^{j\omega})|_{\omega=\frac{2\pi}{M} h} \end{aligned}$$

About zero-padding

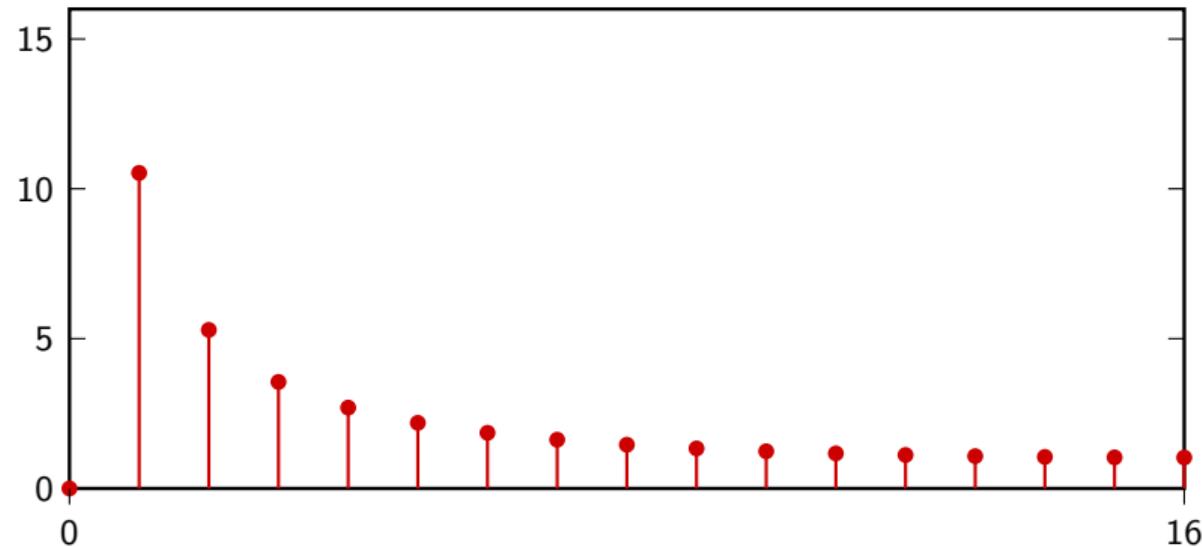
$$\begin{aligned} X_M[h] &= \sum_{n=0}^{M-1} x'[n] e^{-j\frac{2\pi}{M} nh} = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{M} nh} \\ &= \sum_{n=0}^{N-1} \left(\frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N} nk} \right) e^{-j\frac{2\pi}{M} nh} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] \left(\sum_{n=0}^{N-1} e^{-j(\frac{2\pi}{M} h - \frac{2\pi}{N} k)n} \right) \\ &= \bar{X}(e^{j\omega})|_{\omega=\frac{2\pi}{M} h} \end{aligned}$$

About zero-padding

- ▶ zero padding does not add information
- ▶ a zero-padded DFT is simply a sampled DTFT of the finite-support extension

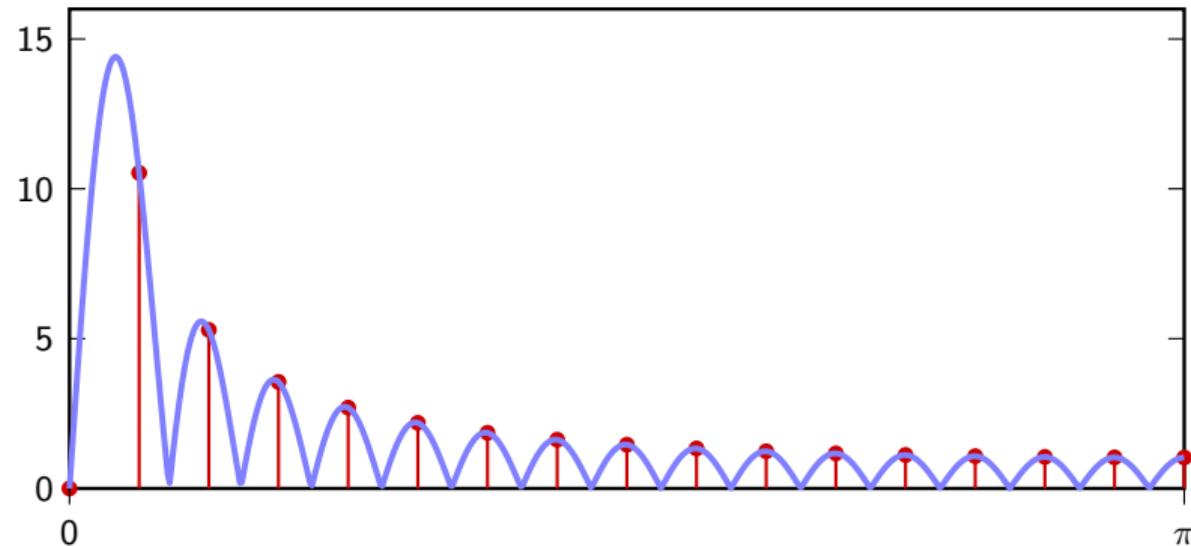
DFT of 32-tap sawtooth, zero-padded

32-point DFT



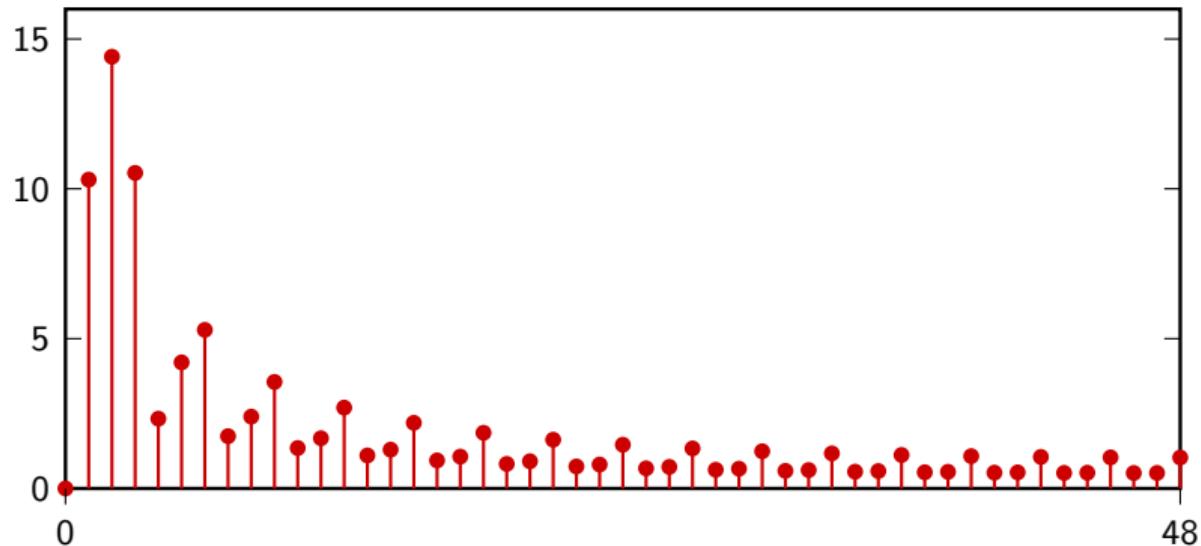
DFT of 32-tap sawtooth, zero-padded

32-point DFT



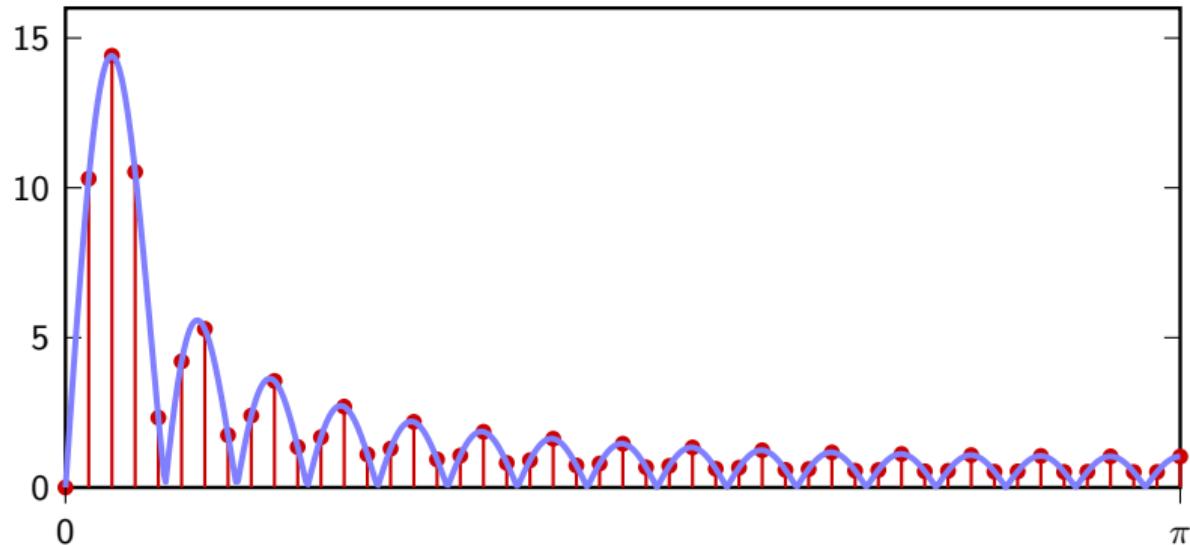
DFT of 32-tap sawtooth, zero-padded

96-point DFT



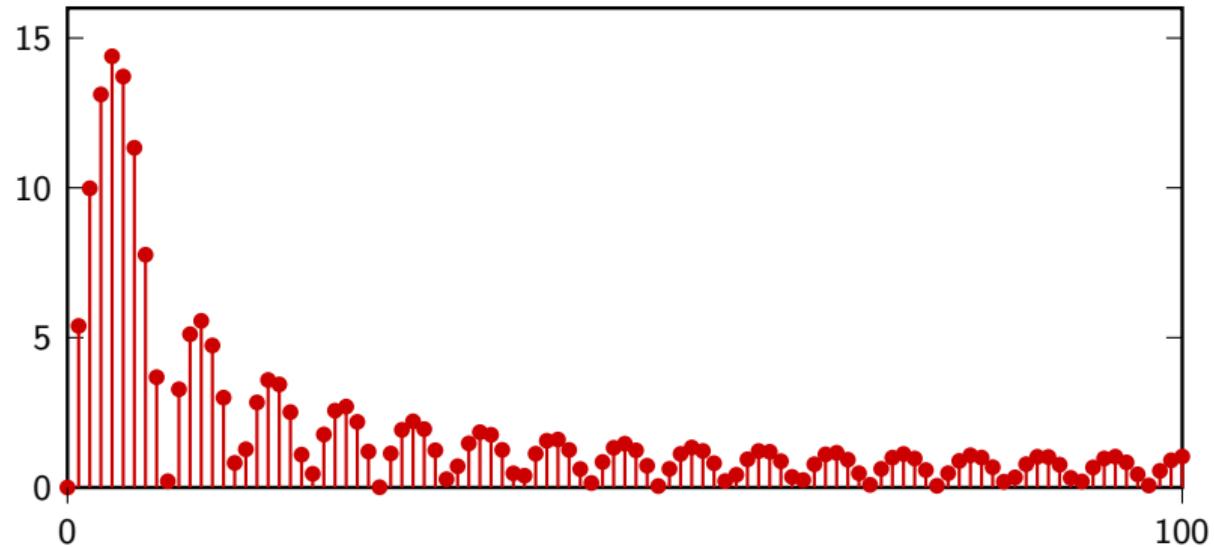
DFT of 32-tap sawtooth, zero-padded

96-point DFT



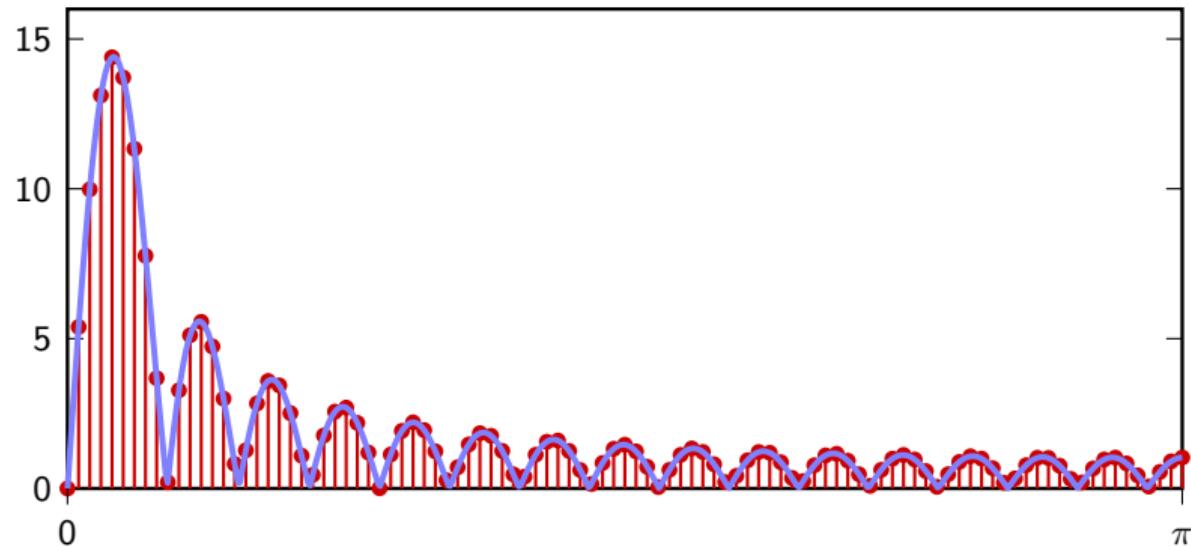
DFT of 32-tap sawtooth, zero-padded

200-point DFT



DFT of 32-tap sawtooth, zero-padded

200-point DFT



modulation

Overview:

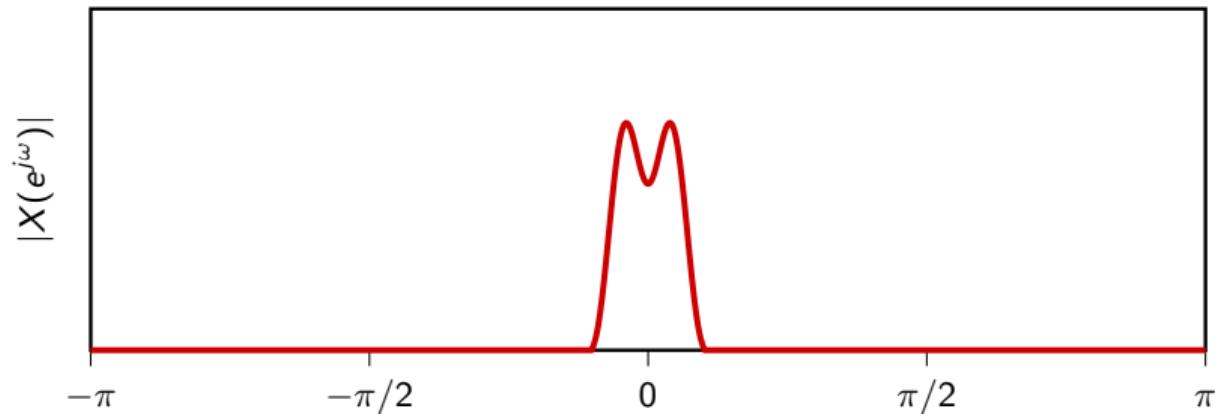
- ▶ Lowpass, highpass and bandpass signals
- ▶ Sinusoidal modulation
- ▶ Tuning a guitar

Classifying signals in frequency

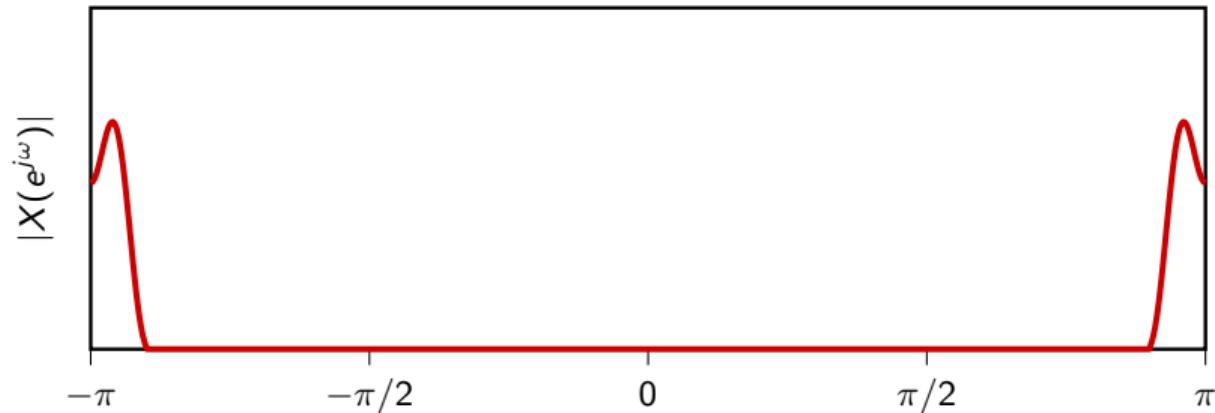
Three broad categories according to where most of the spectral energy resides:

- ▶ lowpass signals (also known as “baseband” signals)
- ▶ highpass signals
- ▶ bandpass signals

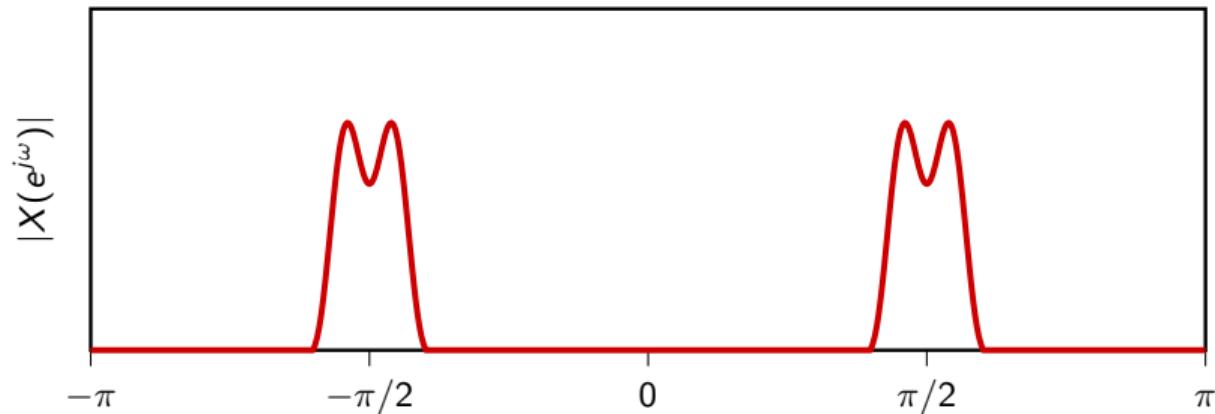
Lowpass example



Highpass example



Bandpass example



Sinusoidal modulation

$$\begin{aligned}\text{DTFT} \{x[n] \cos(\omega_c n)\} &= \text{DTFT} \left\{ \frac{1}{2} e^{j\omega_c n} x[n] + \frac{1}{2} e^{-j\omega_c n} x[n] \right\} \\ &= \frac{1}{2} [X(e^{j(\omega-\omega_c)}) + X(e^{j(\omega+\omega_c)})]\end{aligned}$$

- ▶ usually $x[n]$ baseband
- ▶ ω_c is the *carrier* frequency

Sinusoidal modulation

$$\begin{aligned}\text{DTFT } \{x[n] \cos(\omega_c n)\} &= \text{DTFT} \left\{ \frac{1}{2} e^{j\omega_c n} x[n] + \frac{1}{2} e^{-j\omega_c n} x[n] \right\} \\ &= \frac{1}{2} [X(e^{j(\omega-\omega_c)}) + X(e^{j(\omega+\omega_c)})]\end{aligned}$$

- ▶ usually $x[n]$ baseband
- ▶ ω_c is the *carrier* frequency

Sinusoidal modulation

$$\begin{aligned}\text{DTFT } \{x[n] \cos(\omega_c n)\} &= \text{DTFT} \left\{ \frac{1}{2} e^{j\omega_c n} x[n] + \frac{1}{2} e^{-j\omega_c n} x[n] \right\} \\ &= \frac{1}{2} [X(e^{j(\omega-\omega_c)}) + X(e^{j(\omega+\omega_c)})]\end{aligned}$$

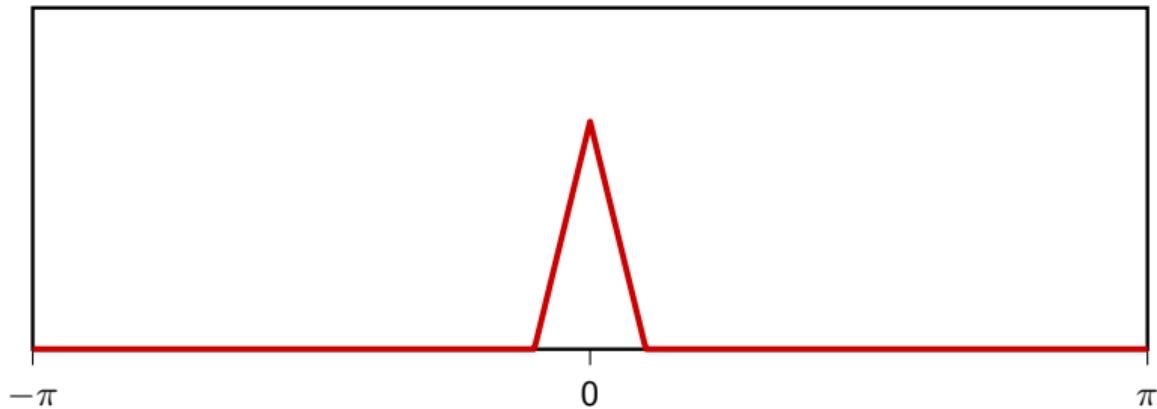
- ▶ usually $x[n]$ baseband
- ▶ ω_c is the *carrier* frequency

Sinusoidal modulation

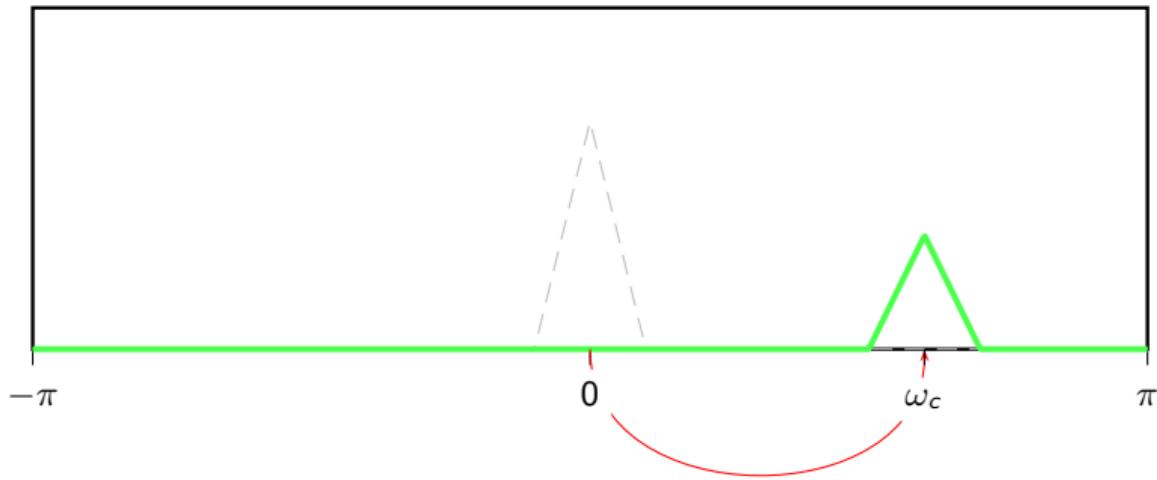
$$\begin{aligned}\text{DTFT } \{x[n] \cos(\omega_c n)\} &= \text{DTFT} \left\{ \frac{1}{2} e^{j\omega_c n} x[n] + \frac{1}{2} e^{-j\omega_c n} x[n] \right\} \\ &= \frac{1}{2} [X(e^{j(\omega-\omega_c)}) + X(e^{j(\omega+\omega_c)})]\end{aligned}$$

- ▶ usually $x[n]$ baseband
- ▶ ω_c is the *carrier* frequency

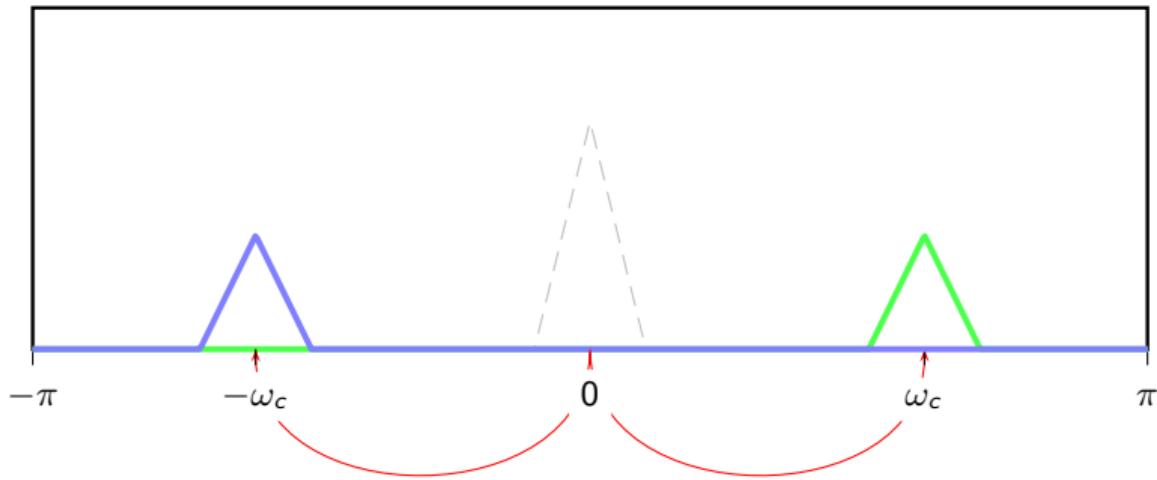
Example



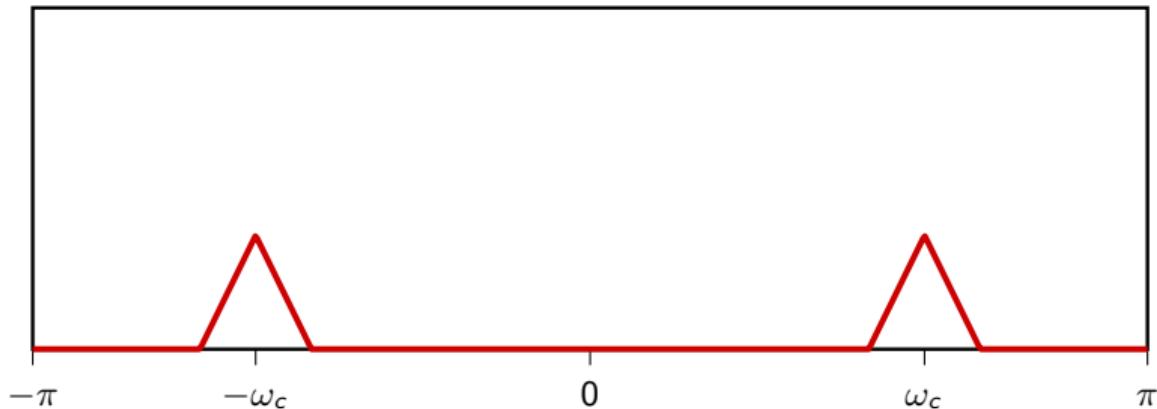
Example



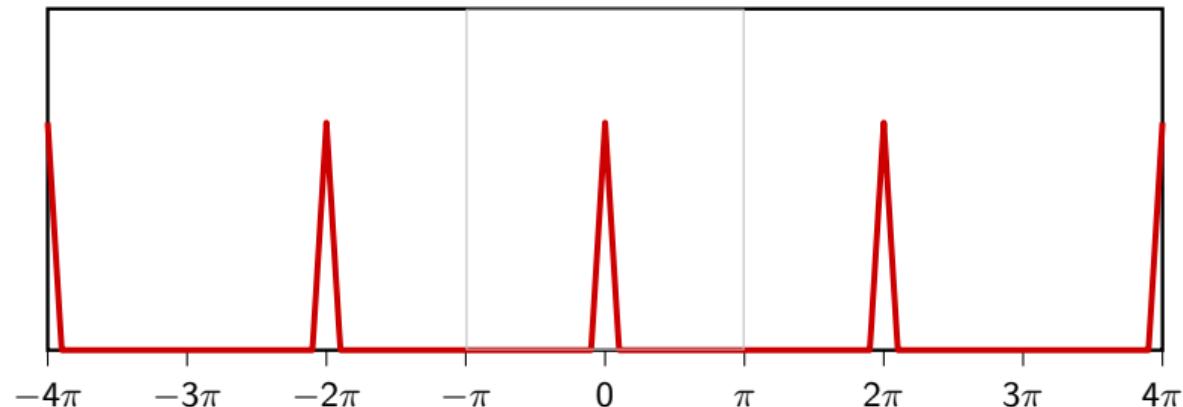
Example



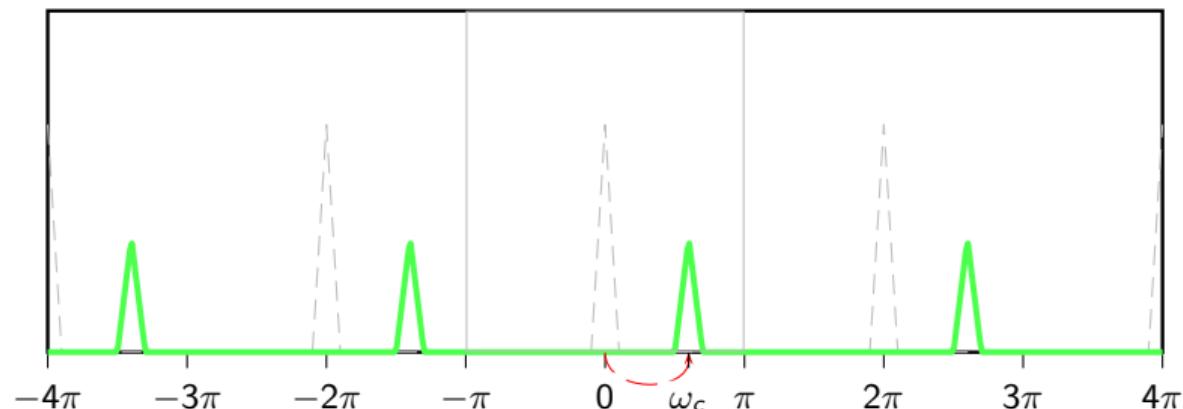
Example



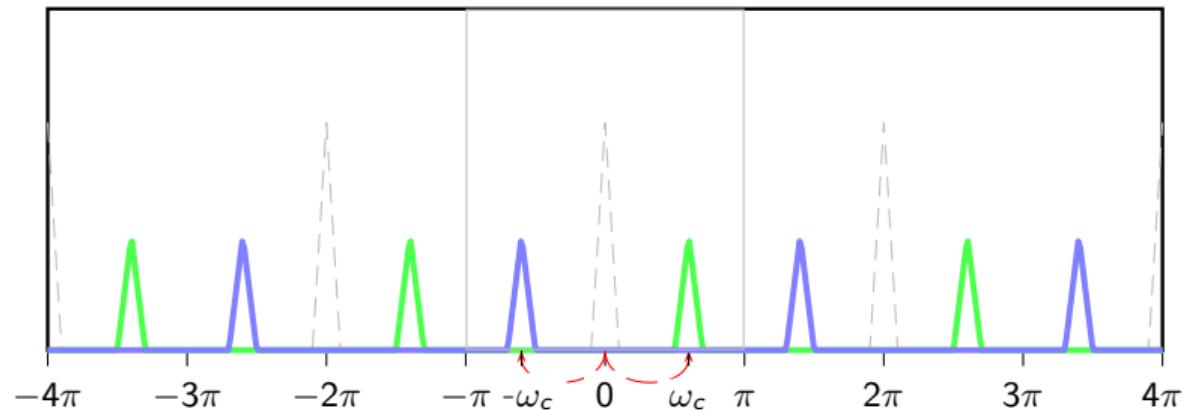
Again, explicitly showing the periodicity of the spectrum



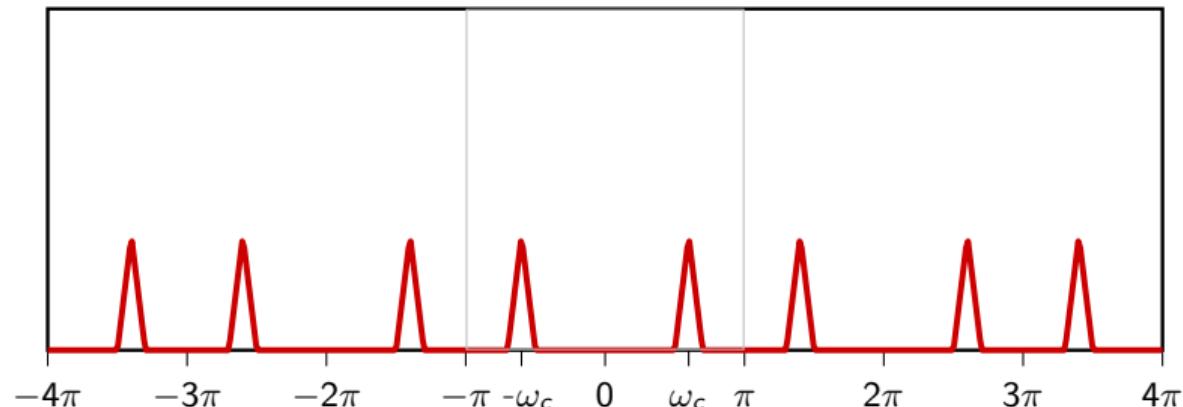
Again, explicitly showing the periodicity of the spectrum



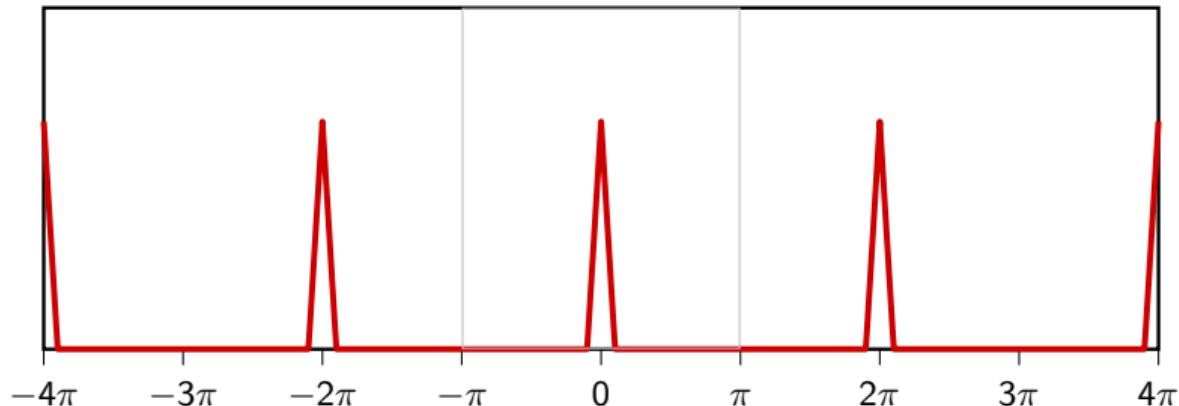
Again, explicitly showing the periodicity of the spectrum



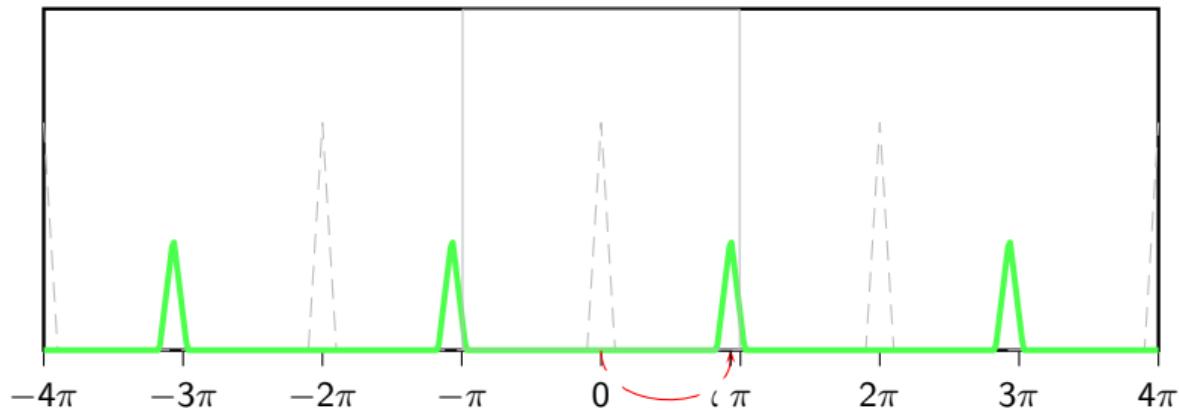
Again, explicitly showing the periodicity of the spectrum



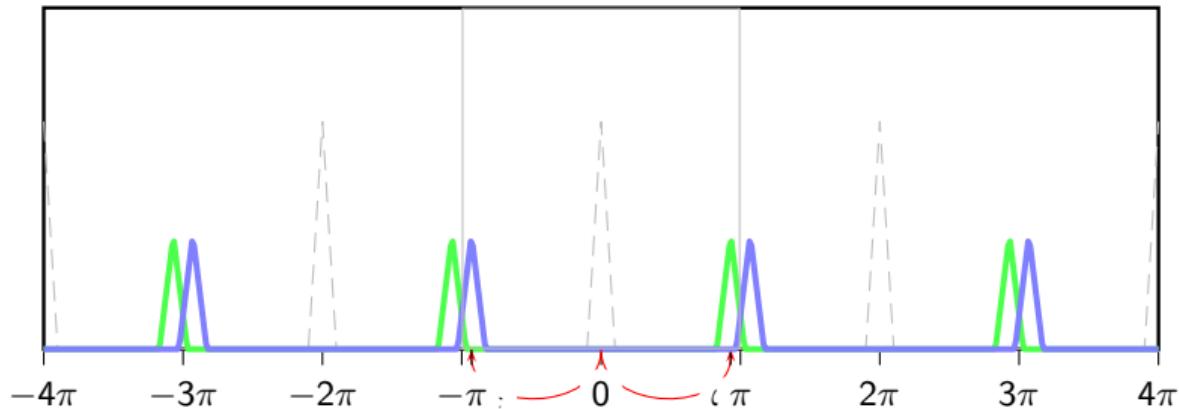
Careful when the modulation frequency is too large!



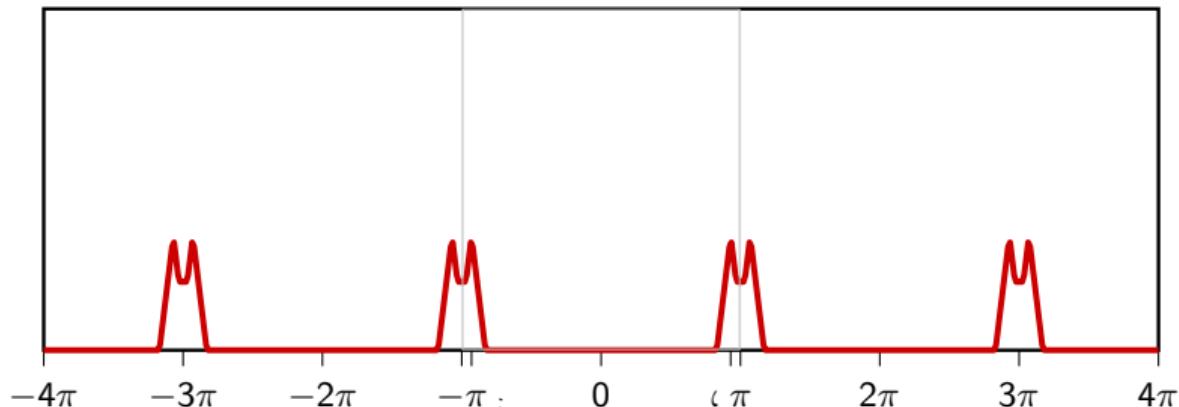
Careful when the modulation frequency is too large!



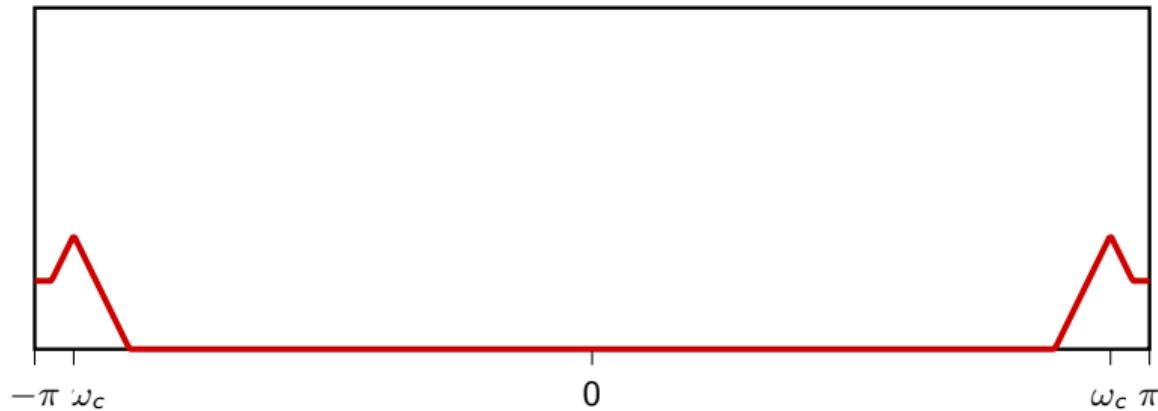
Careful when the modulation frequency is too large!



Careful when the modulation frequency is too large!



Careful when the modulation frequency is too large!



Sinusoidal modulation: applications

- ▶ voice and music are lowpass signals
- ▶ radio channels are bandpass, in much higher frequencies
- ▶ modulation brings the baseband signal in the transmission band
- ▶ demodulation at the receiver brings it back

Sinusoidal demodulation

just multiply the received signal by the carrier again

$$y[n] = x[n] \cos(\omega_c n) \quad Y(e^{j\omega}) = \frac{1}{2} [X(e^{j(\omega-\omega_c)}) + X(e^{j(\omega+\omega_c)})]$$

$$\begin{aligned}\text{DTFT } \{y[n] \cdot 2 \cos(\omega_c n)\} &= Y(e^{j(\omega-\omega_c)}) + Y(e^{j(\omega+\omega_c)}) \\ &= \frac{1}{2} [X(e^{j(\omega-2\omega_c)}) + X(e^{j(\omega)}) + X(e^{j(\omega)}) + X(e^{j(\omega+2\omega_c)})] \\ &= X(e^{j(\omega)}) + \frac{1}{2} [X(e^{j(\omega-2\omega_c)}) + X(e^{j(\omega+2\omega_c)})]\end{aligned}$$

Sinusoidal demodulation

just multiply the received signal by the carrier again

$$y[n] = x[n] \cos(\omega_c n) \quad Y(e^{j\omega}) = \frac{1}{2} [X(e^{j(\omega-\omega_c)}) + X(e^{j(\omega+\omega_c)})]$$

$$\begin{aligned}\text{DTFT } \{y[n] \cdot 2 \cos(\omega_c n)\} &= Y(e^{j(\omega-\omega_c)}) + Y(e^{j(\omega+\omega_c)}) \\ &= \frac{1}{2} [X(e^{j(\omega-2\omega_c)}) + X(e^{j(\omega)}) + X(e^{j(\omega)}) + X(e^{j(\omega+2\omega_c)})] \\ &= X(e^{j(\omega)}) + \frac{1}{2} [X(e^{j(\omega-2\omega_c)}) + X(e^{j(\omega+2\omega_c)})]\end{aligned}$$

Sinusoidal demodulation

just multiply the received signal by the carrier again

$$y[n] = x[n] \cos(\omega_c n) \quad Y(e^{j\omega}) = \frac{1}{2} [X(e^{j(\omega-\omega_c)}) + X(e^{j(\omega+\omega_c)})]$$

$$\begin{aligned}\text{DTFT } \{y[n] \cdot 2 \cos(\omega_c n)\} &= Y(e^{j(\omega-\omega_c)}) + Y(e^{j(\omega+\omega_c)}) \\ &= \frac{1}{2} [X(e^{j(\omega-2\omega_c)}) + X(e^{j(\omega)}) + X(e^{j(\omega)}) + X(e^{j(\omega+2\omega_c)})] \\ &= X(e^{j(\omega)}) + \frac{1}{2} [X(e^{j(\omega-2\omega_c)}) + X(e^{j(\omega+2\omega_c)})]\end{aligned}$$

Sinusoidal demodulation

just multiply the received signal by the carrier again

$$y[n] = x[n] \cos(\omega_c n) \quad Y(e^{j\omega}) = \frac{1}{2} [X(e^{j(\omega-\omega_c)}) + X(e^{j(\omega+\omega_c)})]$$

$$\begin{aligned}\text{DTFT } \{y[n] \cdot 2 \cos(\omega_c n)\} &= Y(e^{j(\omega-\omega_c)}) + Y(e^{j(\omega+\omega_c)}) \\ &= \frac{1}{2} [X(e^{j(\omega-2\omega_c)}) + X(e^{j(\omega)}) + X(e^{j(\omega)}) + X(e^{j(\omega+2\omega_c)})] \\ &= X(e^{j(\omega)}) + \frac{1}{2} [X(e^{j(\omega-2\omega_c)}) + X(e^{j(\omega+2\omega_c)})]\end{aligned}$$

Sinusoidal demodulation

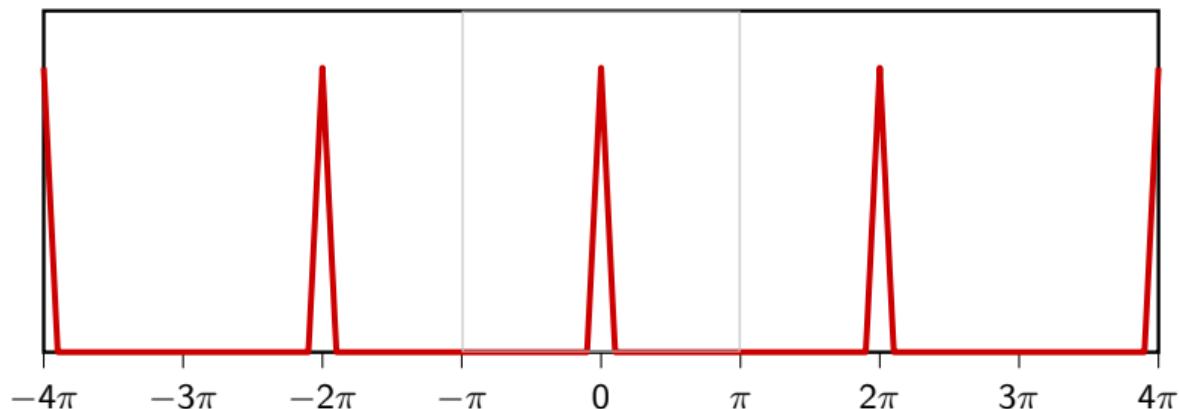
just multiply the received signal by the carrier again

$$y[n] = x[n] \cos(\omega_c n) \quad Y(e^{j\omega}) = \frac{1}{2} [X(e^{j(\omega-\omega_c)}) + X(e^{j(\omega+\omega_c)})]$$

$$\begin{aligned}\text{DTFT } \{y[n] \cdot 2 \cos(\omega_c n)\} &= Y(e^{j(\omega-\omega_c)}) + Y(e^{j(\omega+\omega_c)}) \\ &= \frac{1}{2} [X(e^{j(\omega-2\omega_c)}) + X(e^{j(\omega)}) + X(e^{j(\omega)}) + X(e^{j(\omega+2\omega_c)})] \\ &= X(e^{j(\omega)}) + \frac{1}{2} [X(e^{j(\omega-2\omega_c)}) + X(e^{j(\omega+2\omega_c)})]\end{aligned}$$

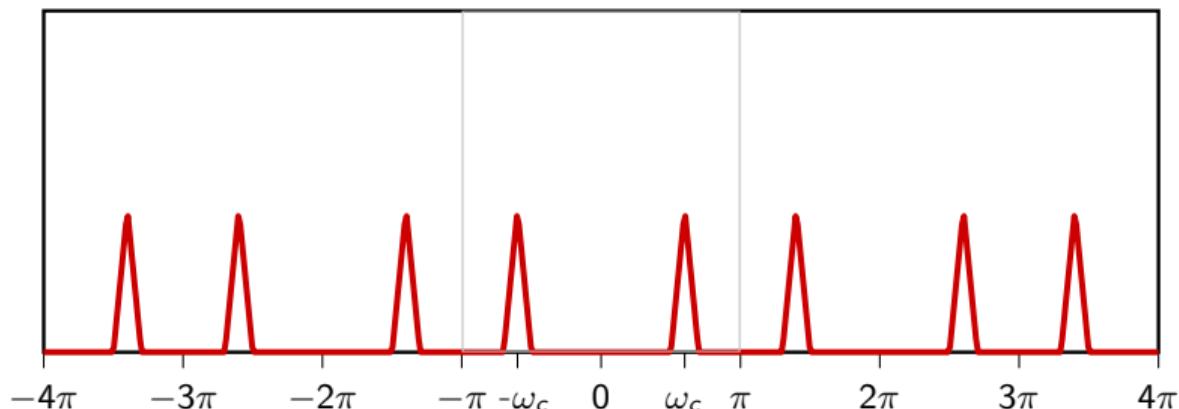
Demodulation in the frequency domain

DTFT $\{x[n]\}$



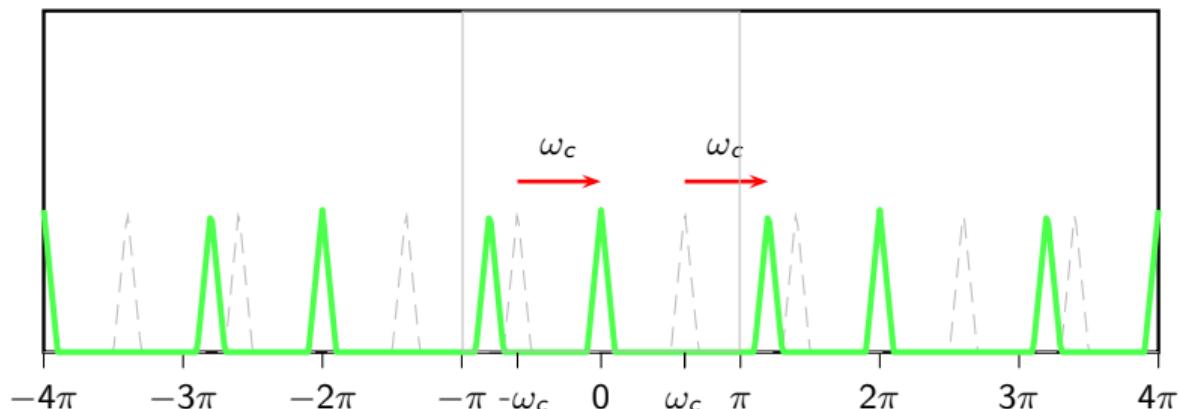
Demodulation in the frequency domain

$$\text{DTFT } \{y[n]\} = \text{DTFT } \{x[n] \cos \omega_c n\}$$



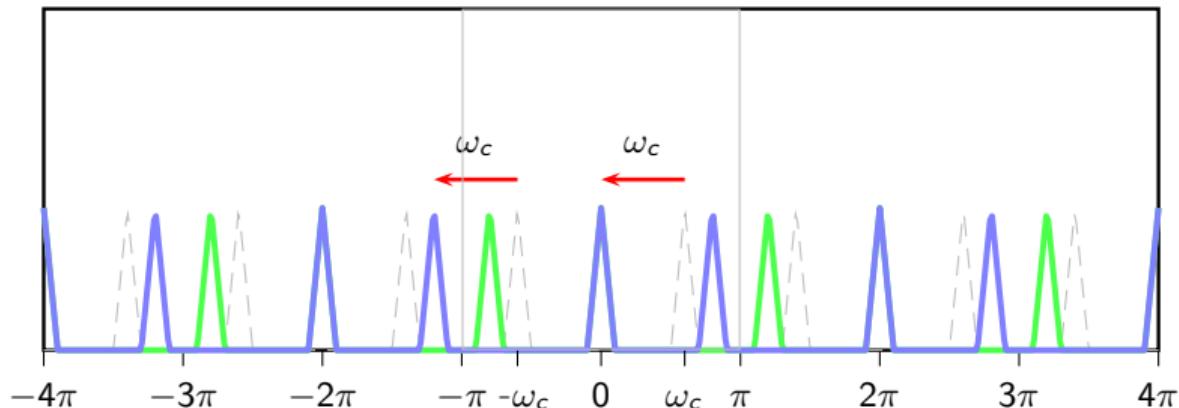
Demodulation in the frequency domain

$$\text{DTFT } \{y[n] 2 \cos \omega_c n\}$$



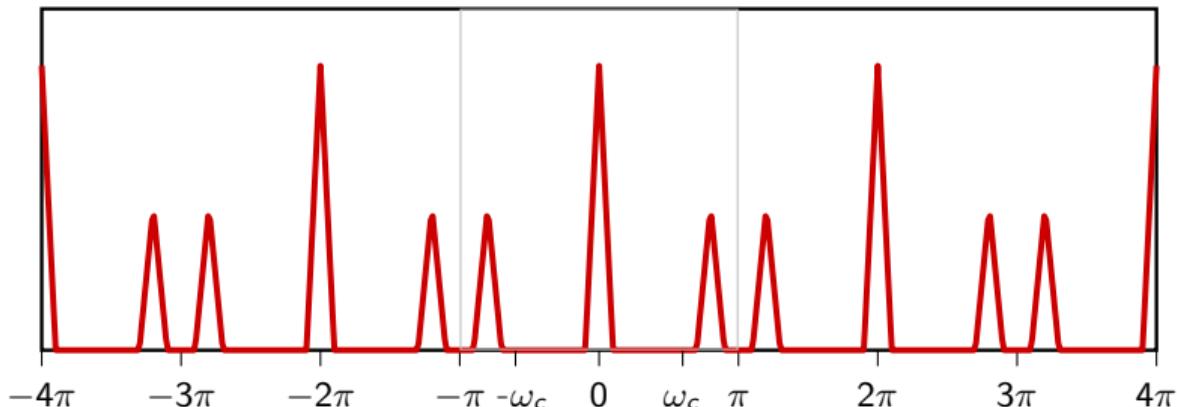
Demodulation in the frequency domain

$$\text{DTFT } \{y[n] 2 \cos \omega_c n\}$$



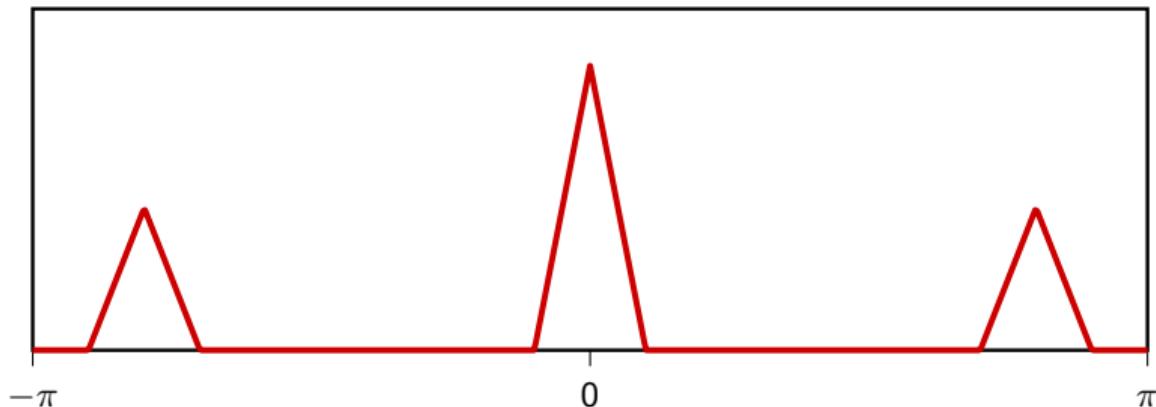
Demodulation in the frequency domain

$$\text{DTFT } \{y[n] 2 \cos \omega_c n\}$$



Demodulation in the frequency domain

$$\text{DTFT } \{y[n] \cos \omega_c n\}$$



Demodulation in the frequency domain

- ▶ we recovered the baseband signal exactly...
- ▶ but we have some spurious high-frequency components
- ▶ in the next lectures we will learn how to get rid of them!

Demodulation in the frequency domain

- ▶ we recovered the baseband signal exactly...
- ▶ but we have some spurious high-frequency components
- ▶ in the next lectures we will learn how to get rid of them!

Demodulation in the frequency domain

- ▶ we recovered the baseband signal exactly...
- ▶ but we have some spurious high-frequency components
- ▶ in the next lectures we will learn how to get rid of them!

Another application: tuning a guitar

Problem (abstraction):

- ▶ reference sinusoid at frequency ω_0
- ▶ tunable sinusoid of frequency ω
- ▶ make $\omega = \omega_0$ "by ear"

Another application: tuning a guitar

Problem (abstraction):

- ▶ reference sinusoid at frequency ω_0
- ▶ tunable sinusoid of frequency ω
- ▶ make $\omega = \omega_0$ "by ear"

Another application: tuning a guitar

Problem (abstraction):

- ▶ reference sinusoid at frequency ω_0
- ▶ tunable sinusoid of frequency ω
- ▶ make $\omega = \omega_0$ “by ear”

The procedure

1. bring ω close to ω_0 (easy)
2. when $\omega \approx \omega_0$ play both sinusoids together
3. trigonometry comes to the rescue:

$$\begin{aligned}x[n] &= \cos(\omega_0 n) + \cos(\omega n) \\&= 2 \cos\left(\frac{\omega_0 + \omega}{2} n\right) \cos\left(\frac{\omega_0 - \omega}{2} n\right) \\&\approx 2 \cos(\Delta_\omega n) \cos(\omega_0 n)\end{aligned}$$

The procedure

1. bring ω close to ω_0 (easy)
2. when $\omega \approx \omega_0$ play both sinusoids together
3. trigonometry comes to the rescue:

$$\begin{aligned}x[n] &= \cos(\omega_0 n) + \cos(\omega n) \\&= 2 \cos\left(\frac{\omega_0 + \omega}{2} n\right) \cos\left(\frac{\omega_0 - \omega}{2} n\right) \\&\approx 2 \cos(\Delta_\omega n) \cos(\omega_0 n)\end{aligned}$$

The procedure

1. bring ω close to ω_0 (easy)
2. when $\omega \approx \omega_0$ play both sinusoids together
3. trigonometry comes to the rescue:

$$\begin{aligned}x[n] &= \cos(\omega_0 n) + \cos(\omega n) \\&= 2 \cos\left(\frac{\omega_0 + \omega}{2}n\right) \cos\left(\frac{\omega_0 - \omega}{2}n\right) \\&\approx 2 \cos(\Delta_\omega n) \cos(\omega_0 n)\end{aligned}$$

Let's see what's happening

$$x[n] \approx 2 \cos(\Delta_\omega n) \cos(\omega_0 n)$$

- ▶ “error” signal
- ▶ modulation at ω_0
- ▶ when $\omega \approx \omega_0$, the error signal is too low to be heard; modulation brings it up to hearing range and we perceive it as amplitude oscillations of the carrier frequency

Let's see what's happening

$$x[n] \approx [2 \cos(\Delta\omega n)] \cdot [\cos(\omega_0 n)]$$

- ▶ “error” signal
- ▶ modulation at ω_0
- ▶ when $\omega \approx \omega_0$, the error signal is too low to be heard; modulation brings it up to hearing range and we perceive it as amplitude oscillations of the carrier frequency

Let's see what's happening

$$x[n] \approx 2 \cos(\Delta\omega n) \cdot \cos(\omega_0 n)$$

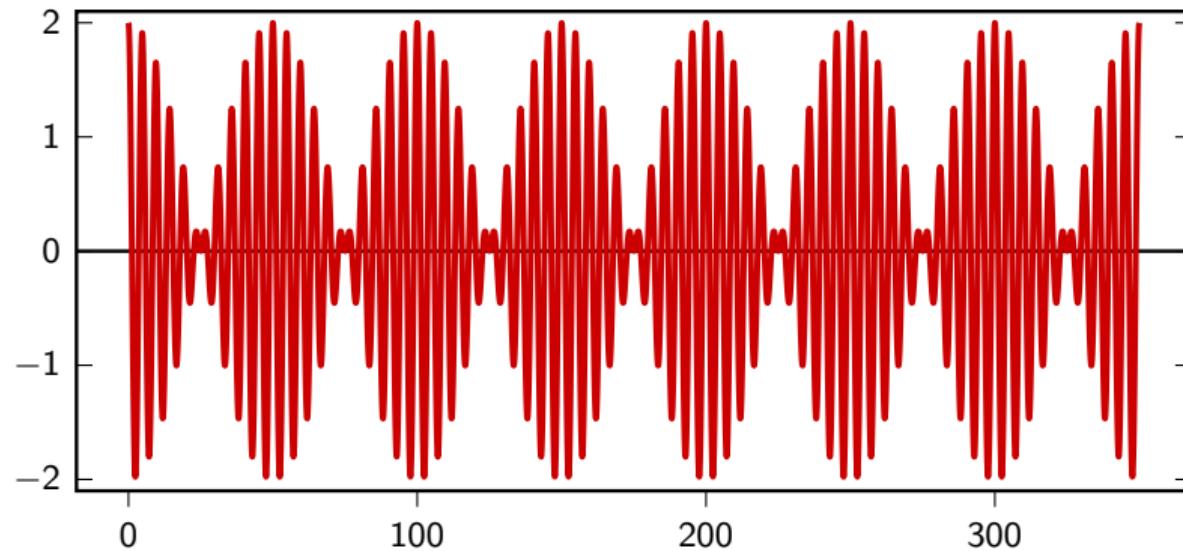
- ▶ “error” signal
- ▶ modulation at ω_0
- ▶ when $\omega \approx \omega_0$, the error signal is too low to be heard; modulation brings it up to hearing range and we perceive it as amplitude oscillations of the carrier frequency

Let's see what's happening

$$x[n] \approx 2 \cos(\Delta\omega n) \cdot \cos(\omega_0 n)$$

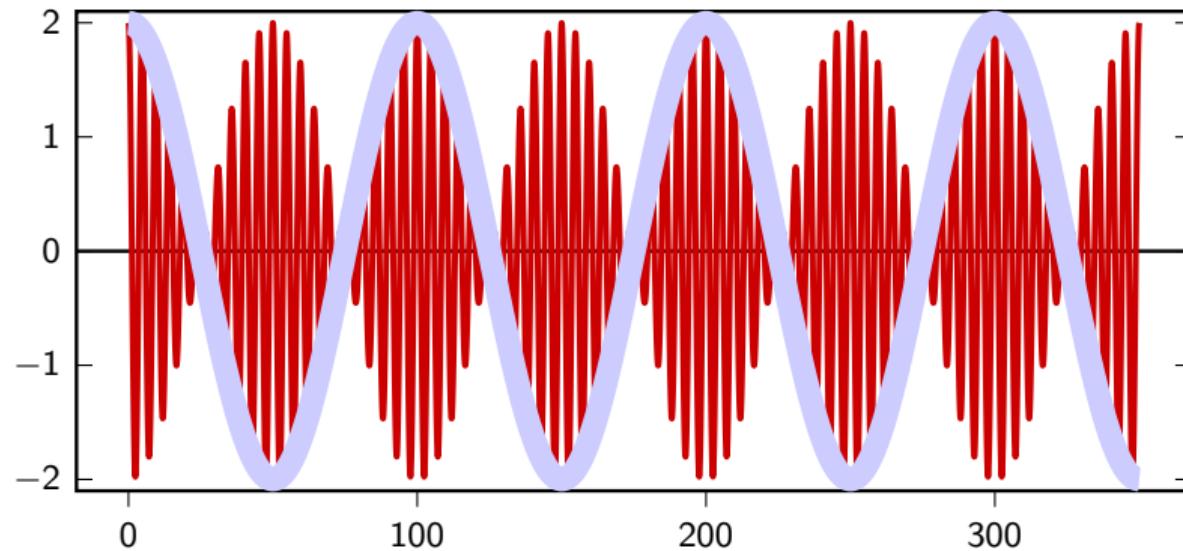
- ▶ “error” signal
- ▶ modulation at ω_0
- ▶ when $\omega \approx \omega_0$, the error signal is too low to be heard; modulation brings it up to hearing range and we perceive it as amplitude oscillations of the carrier frequency

In the time domain...



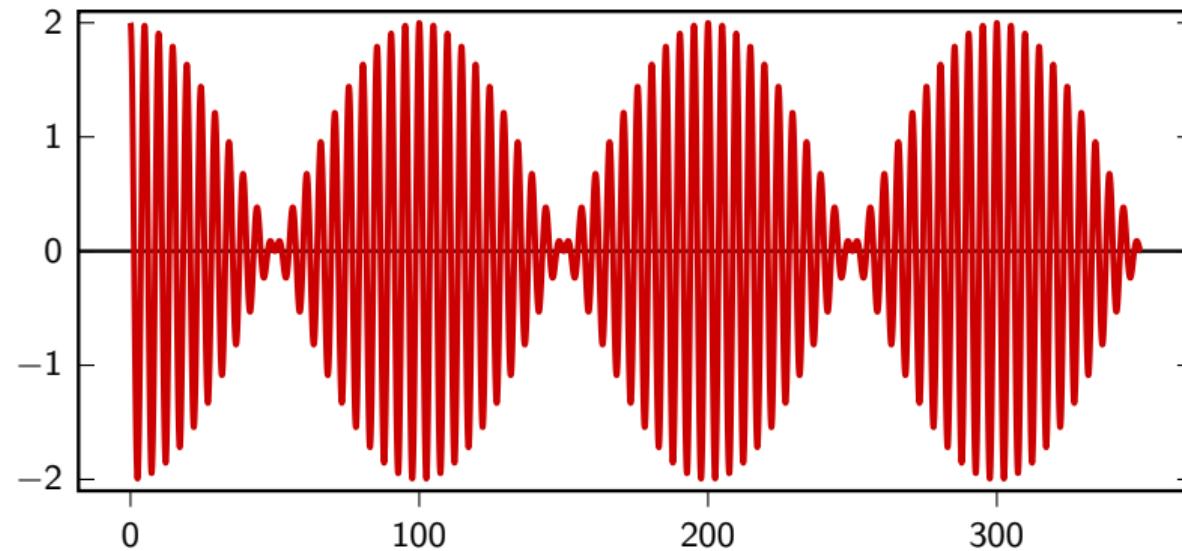
$$\omega_0 = 2\pi \cdot 0.2, \quad \omega = 2\pi \cdot 0.22, \quad \Delta\omega = 2\pi \cdot 0.0100$$

In the time domain...



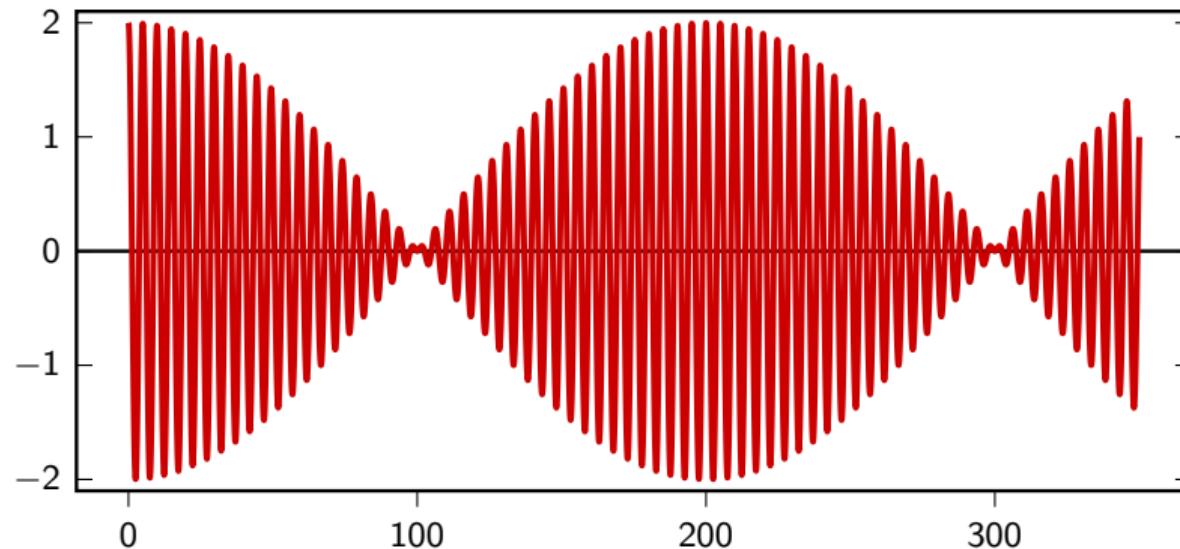
$$\omega_0 = 2\pi \cdot 0.2, \quad \omega = 2\pi \cdot 0.22, \quad \Delta\omega = 2\pi \cdot 0.0100$$

In the time domain...



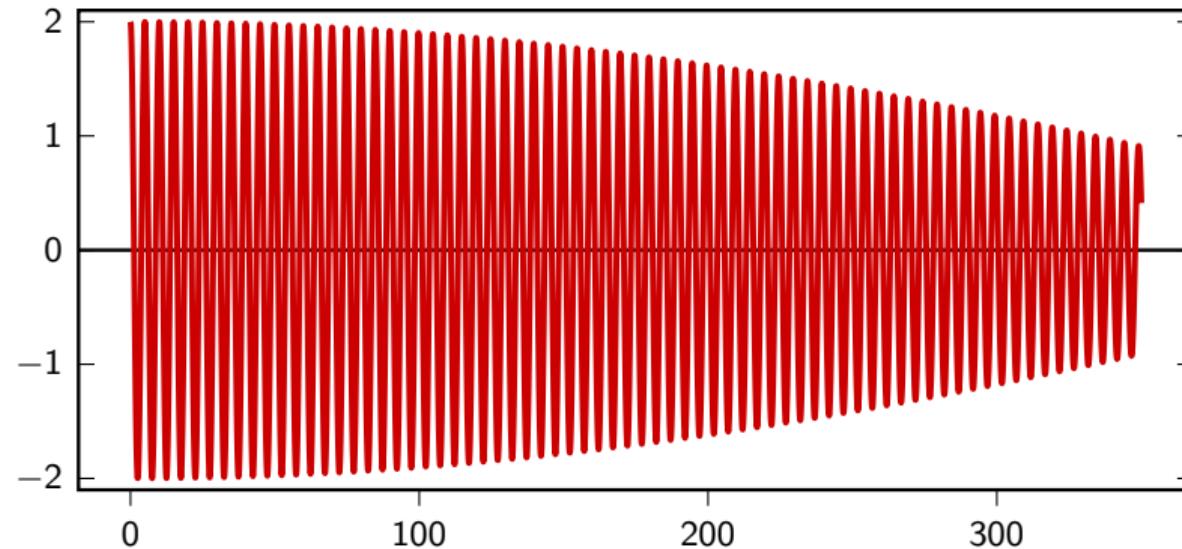
$$\omega_0 = 2\pi \cdot 0.2, \quad \omega = 2\pi \cdot 0.21, \quad \Delta\omega = 2\pi \cdot 0.0050$$

In the time domain...



$$\omega_0 = 2\pi \cdot 0.2, \quad \omega = 2\pi \cdot 0.205, \quad \Delta\omega = 2\pi \cdot 0.0025$$

In the time domain...



$$\omega_0 = 2\pi \cdot 0.2, \quad \omega = 2\pi \cdot 0.201, \quad \Delta\omega = 2\pi \cdot 0.0005$$

demonstration

COM303: Digital Signal Processing

Lecture 8: Numerical Fourier Analysis

Overview:

- ▶ the short-time Fourier transform (aka the "spectrogram")
- ▶ the Fast Fourier Transform algorithm

the short-time Fourier transform (STFT)

Overview:

- ▶ Time vs frequency representations
- ▶ The STFT and the spectrogram
- ▶ Time-Frequency tilings

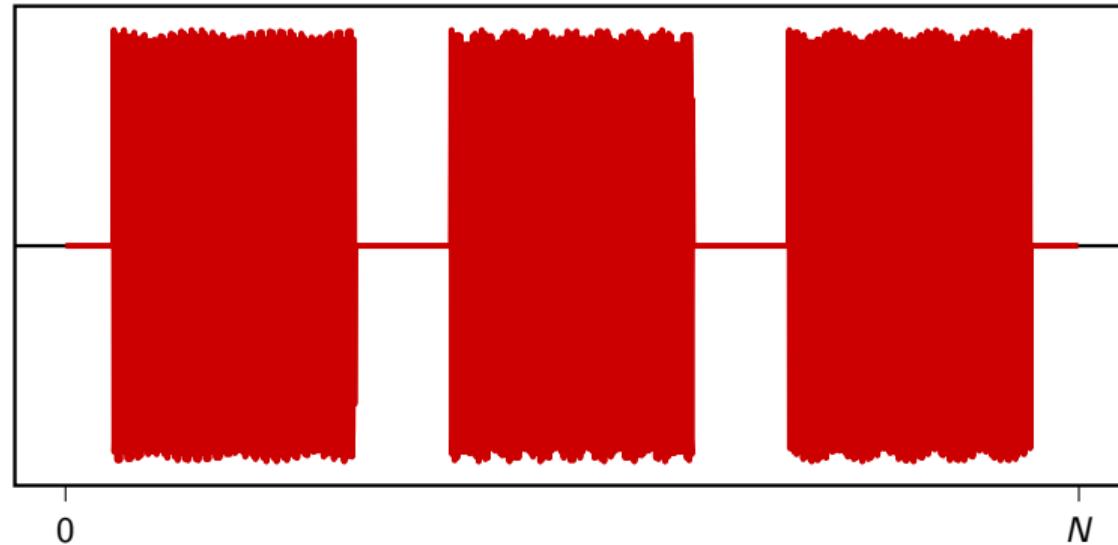
Dual-Tone Multi Frequency dialing



DTMF signaling

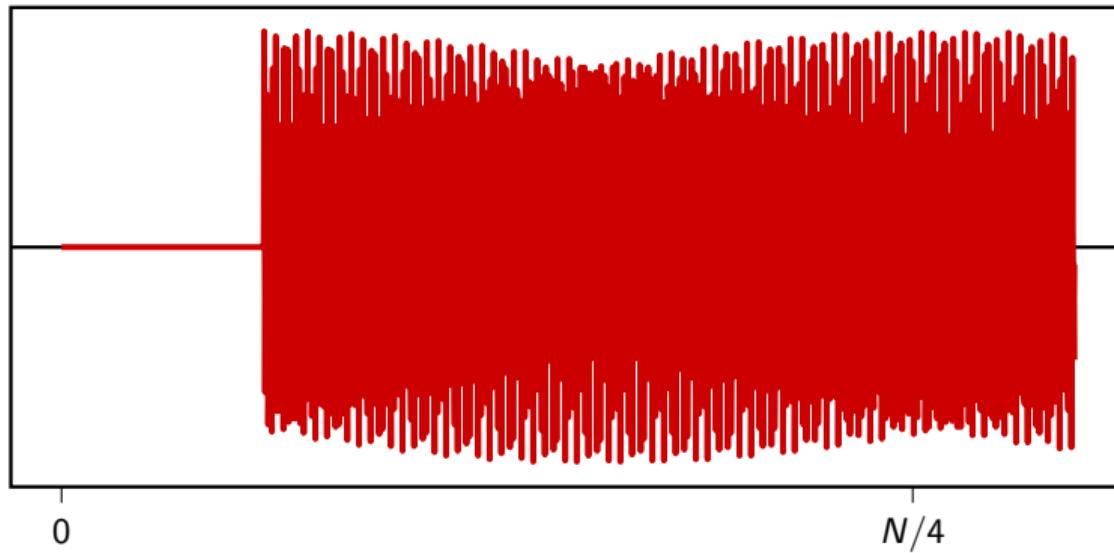
	1209Hz	1336Hz	1477Hz
697Hz	1	2	3
770Hz	4	5	6
852Hz	7	8	9
941Hz	*	0	#

1-5-9 in time

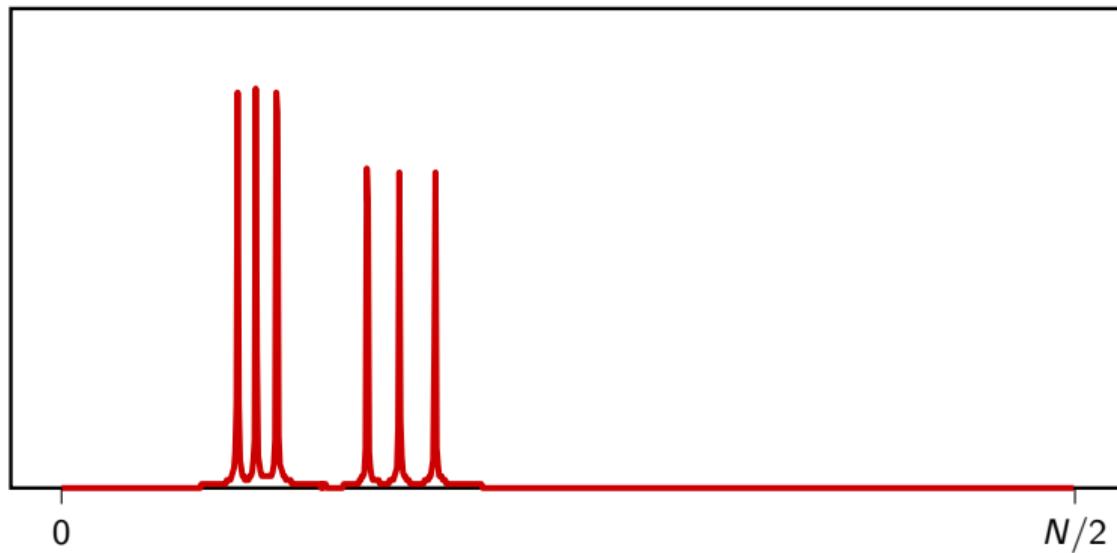


Play

1-5-9 in time (detail)



1-5-9 in frequency (magnitude)



The fundamental tradeoff

- ▶ time representation obfuscates frequency
- ▶ frequency representation obfuscates time

Short-Time Fourier Transform

Idea:

- ▶ take small signal pieces of length L
- ▶ look at the DFT of each piece:

$$X[m; k] = \sum_{n=0}^{L-1} x[m + n] e^{-j \frac{2\pi}{L} nk}$$

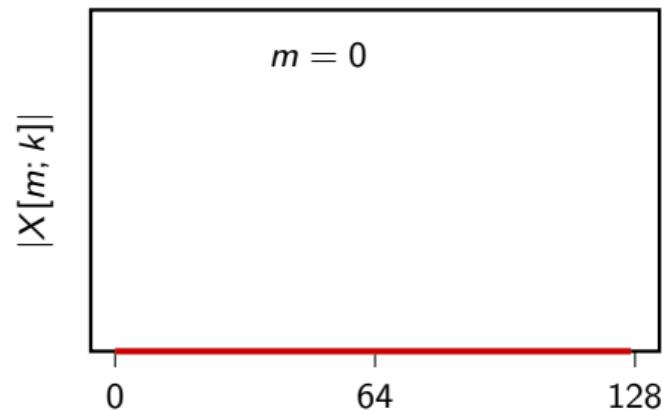
Short-Time Fourier Transform

Idea:

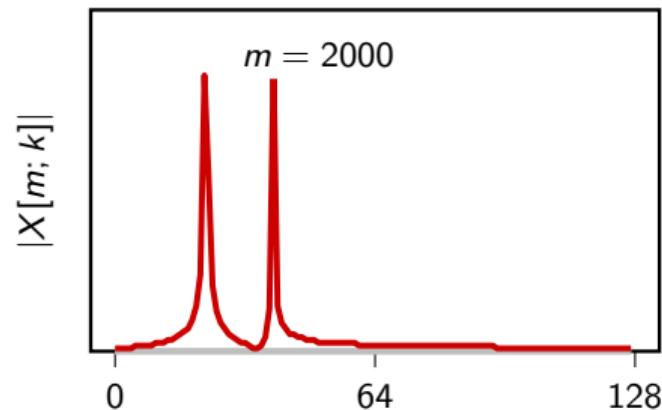
- ▶ take small signal pieces of length L
- ▶ look at the DFT of each piece:

$$X[m; k] = \sum_{n=0}^{L-1} x[m + n] e^{-j \frac{2\pi}{L} nk}$$

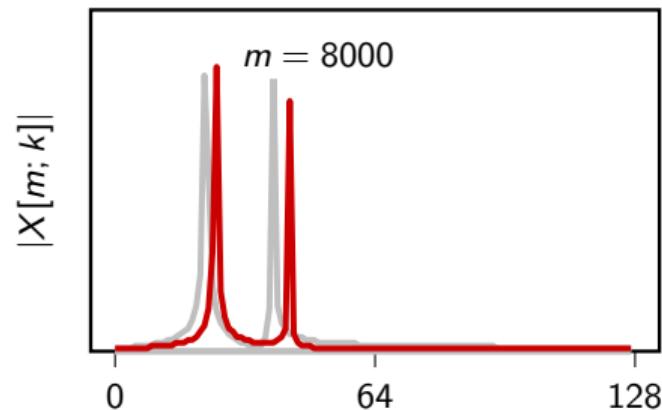
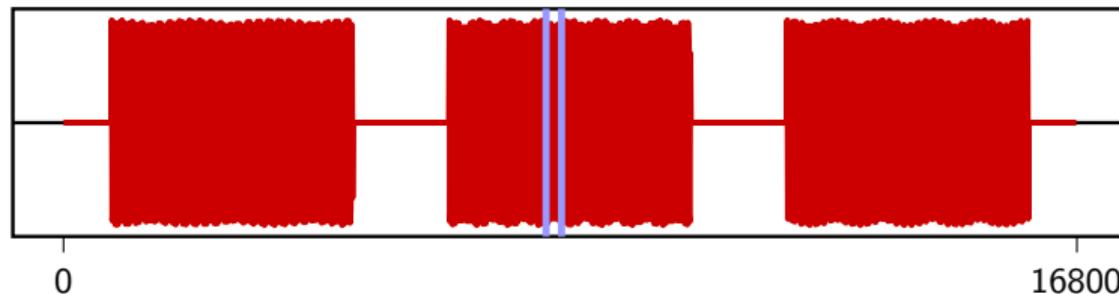
Short-Time Fourier Transform ($L = 256$)



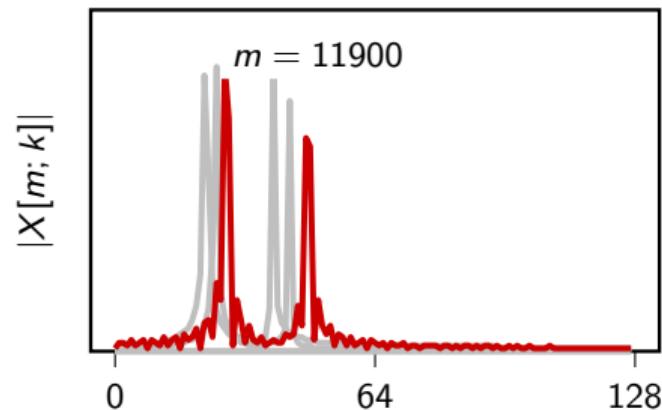
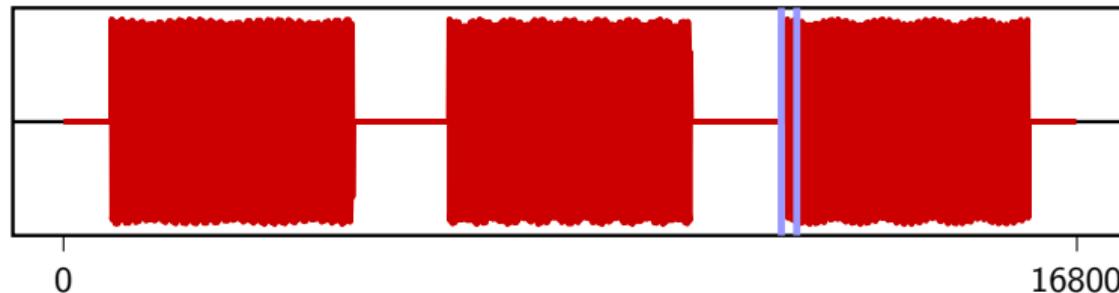
Short-Time Fourier Transform ($L = 256$)



Short-Time Fourier Transform ($L = 256$)



Short-Time Fourier Transform ($L = 256$)



The Spectrogram

Idea:

- ▶ color-code the magnitude: dark is small, white is large
- ▶ use $10 \log_{10}(|X[m; k]|)$ to see better (power in dBs)
- ▶ plot spectral slices one after another

The Spectrogram

Idea:

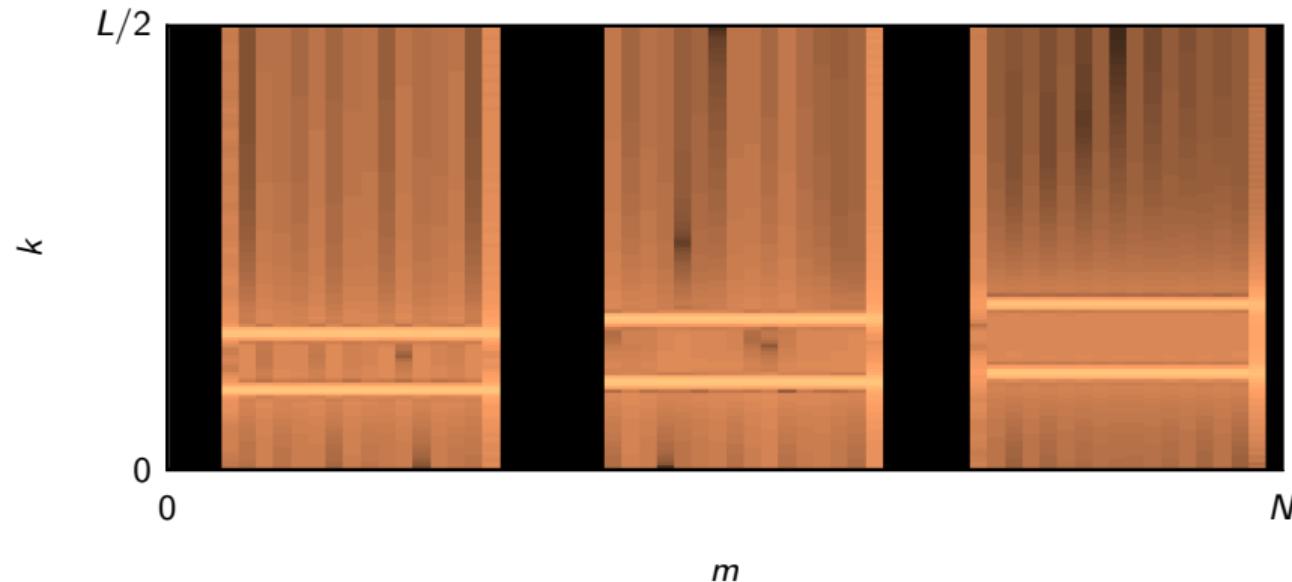
- ▶ color-code the magnitude: dark is small, white is large
- ▶ use $10 \log_{10}(|X[m; k]|)$ to see better (power in dBs)
- ▶ plot spectral slices one after another

The Spectrogram

Idea:

- ▶ color-code the magnitude: dark is small, white is large
- ▶ use $10 \log_{10}(|X[m; k]|)$ to see better (power in dBs)
- ▶ plot spectral slices one after another

DTMF spectrogram



Labeling the Spectrogram

If we know the “system clock” $F_s = 1/T_s$ we can label the axis

- ▶ highest positive frequency: $F_s/2$ Hz
- ▶ frequency resolution: F_s/L Hz
- ▶ width of time slices: $L T_s$ seconds

Labeling the Spectrogram

If we know the “system clock” $F_s = 1/T_s$ we can label the axis

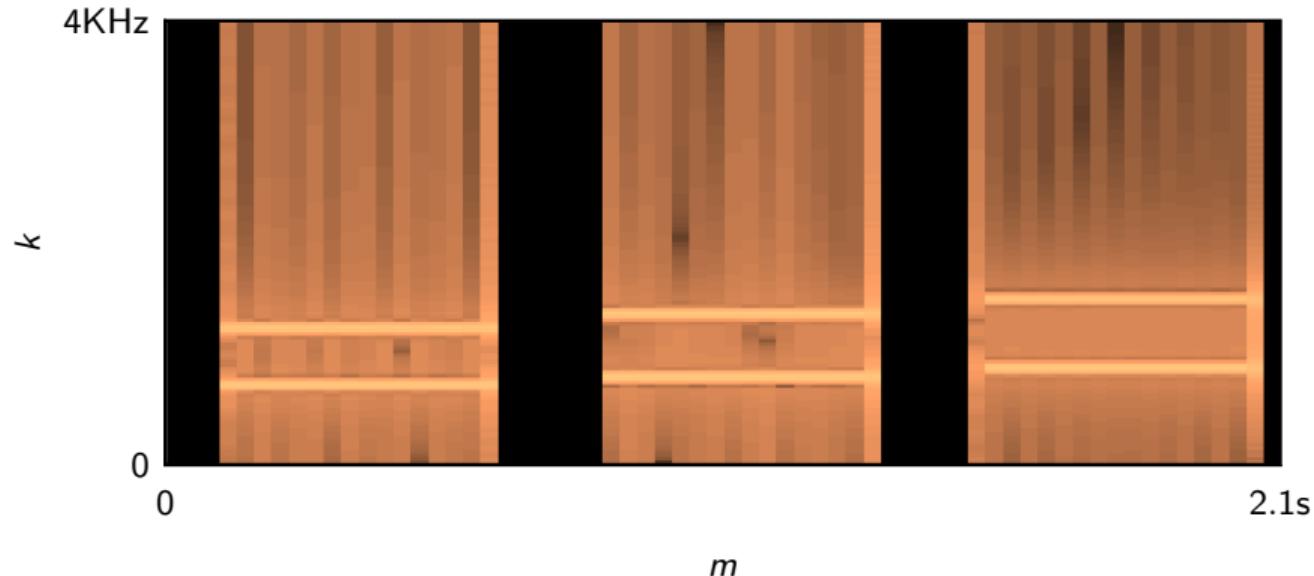
- ▶ highest positive frequency: $F_s/2$ Hz
- ▶ frequency resolution: F_s/L Hz
- ▶ width of time slices: $L T_s$ seconds

Labeling the Spectrogram

If we know the “system clock” $F_s = 1/T_s$ we can label the axis

- ▶ highest positive frequency: $F_s/2$ Hz
- ▶ frequency resolution: F_s/L Hz
- ▶ width of time slices: LT_s seconds

DTMF spectrogram ($F_s = 8000$)



The Spectrogram

Questions:

- ▶ width of the analysis window?
- ▶ position of the windows (overlapping?)
- ▶ shape of the window (weighing the samples)

The Spectrogram

Questions:

- ▶ width of the analysis window?
- ▶ position of the windows (overlapping?)
- ▶ shape of the window (weighing the samples)

The Spectrogram

Questions:

- ▶ width of the analysis window?
- ▶ position of the windows (overlapping?)
- ▶ shape of the window (weighing the samples)

Wideband vs Narrowband

Long window: narrowband spectrogram

- ▶ long window \Rightarrow more DFT points \Rightarrow more frequency resolution
- ▶ long window \Rightarrow more “things can happen” \Rightarrow less precision in time

Short window: wideband spectrogram

- ▶ short window \Rightarrow many time slices \Rightarrow precise location of transitions
- ▶ short window \Rightarrow fewer DFT points \Rightarrow poor frequency resolution

Wideband vs Narrowband

Long window: narrowband spectrogram

- ▶ long window \Rightarrow more DFT points \Rightarrow more frequency resolution
- ▶ long window \Rightarrow more “things can happen” \Rightarrow less precision in time

Short window: wideband spectrogram

- ▶ short window \Rightarrow many time slices \Rightarrow precise location of transitions
- ▶ short window \Rightarrow fewer DFT points \Rightarrow poor frequency resolution

Wideband vs Narrowband

Long window: narrowband spectrogram

- ▶ long window \Rightarrow more DFT points \Rightarrow more frequency resolution
- ▶ long window \Rightarrow more “things can happen” \Rightarrow less precision in time

Short window: wideband spectrogram

- ▶ short window \Rightarrow many time slices \Rightarrow precise location of transitions
- ▶ short window \Rightarrow fewer DFT points \Rightarrow poor frequency resolution

Wideband vs Narrowband

Long window: narrowband spectrogram

- ▶ long window \Rightarrow more DFT points \Rightarrow more frequency resolution
- ▶ long window \Rightarrow more “things can happen” \Rightarrow less precision in time

Short window: wideband spectrogram

- ▶ short window \Rightarrow many time slices \Rightarrow precise location of transitions
- ▶ short window \Rightarrow fewer DFT points \Rightarrow poor frequency resolution

Wideband vs Narrowband

Long window: narrowband spectrogram

- ▶ long window \Rightarrow more DFT points \Rightarrow more frequency resolution
- ▶ long window \Rightarrow more “things can happen” \Rightarrow less precision in time

Short window: wideband spectrogram

- ▶ short window \Rightarrow many time slices \Rightarrow precise location of transitions
- ▶ short window \Rightarrow fewer DFT points \Rightarrow poor frequency resolution

Wideband vs Narrowband

Long window: narrowband spectrogram

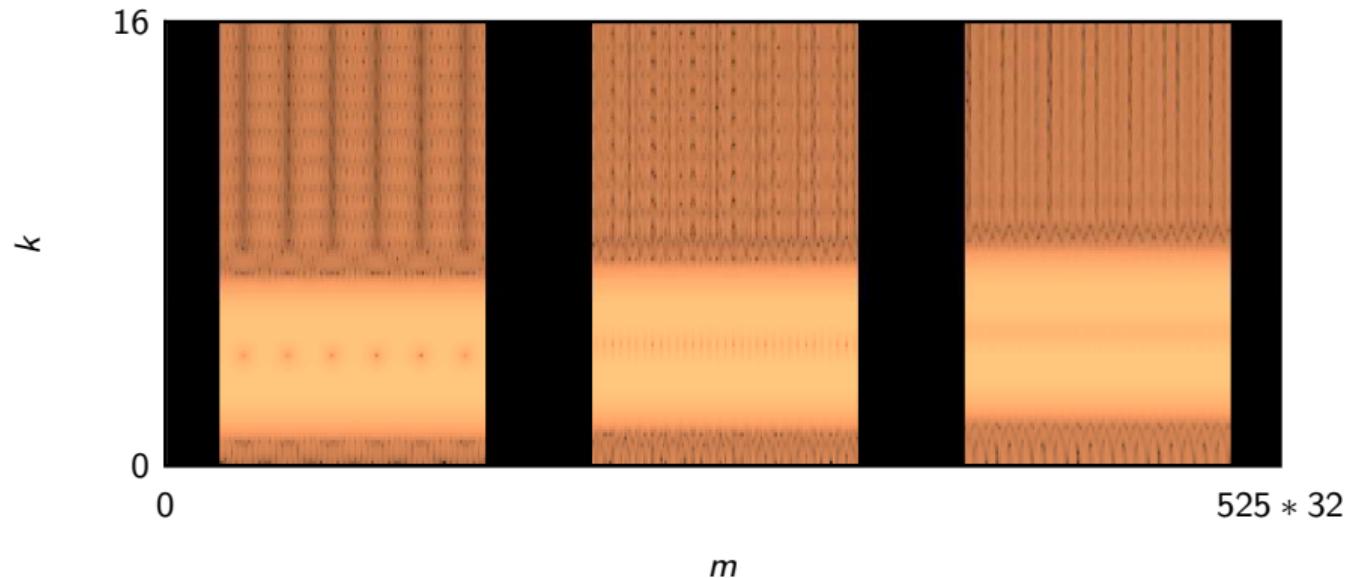
- ▶ long window \Rightarrow more DFT points \Rightarrow more frequency resolution
- ▶ long window \Rightarrow more “things can happen” \Rightarrow less precision in time

Short window: wideband spectrogram

- ▶ short window \Rightarrow many time slices \Rightarrow precise location of transitions
- ▶ short window \Rightarrow fewer DFT points \Rightarrow poor frequency resolution

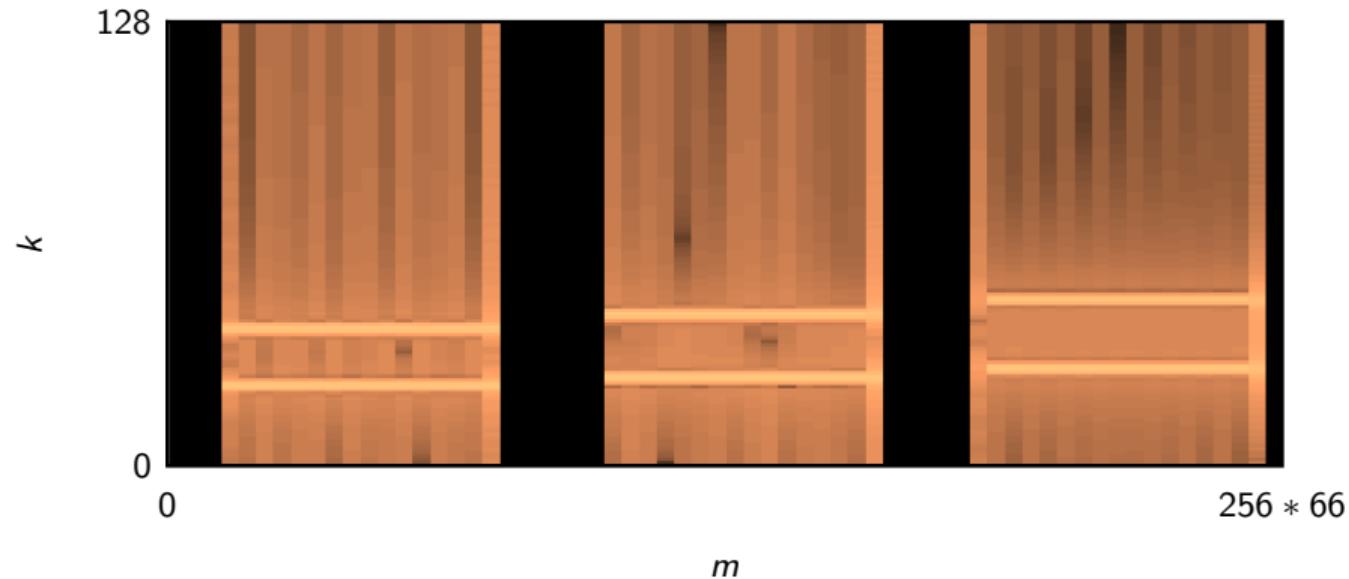
DTMF spectrogram (wideband)

$$N = 16800, L = 32$$



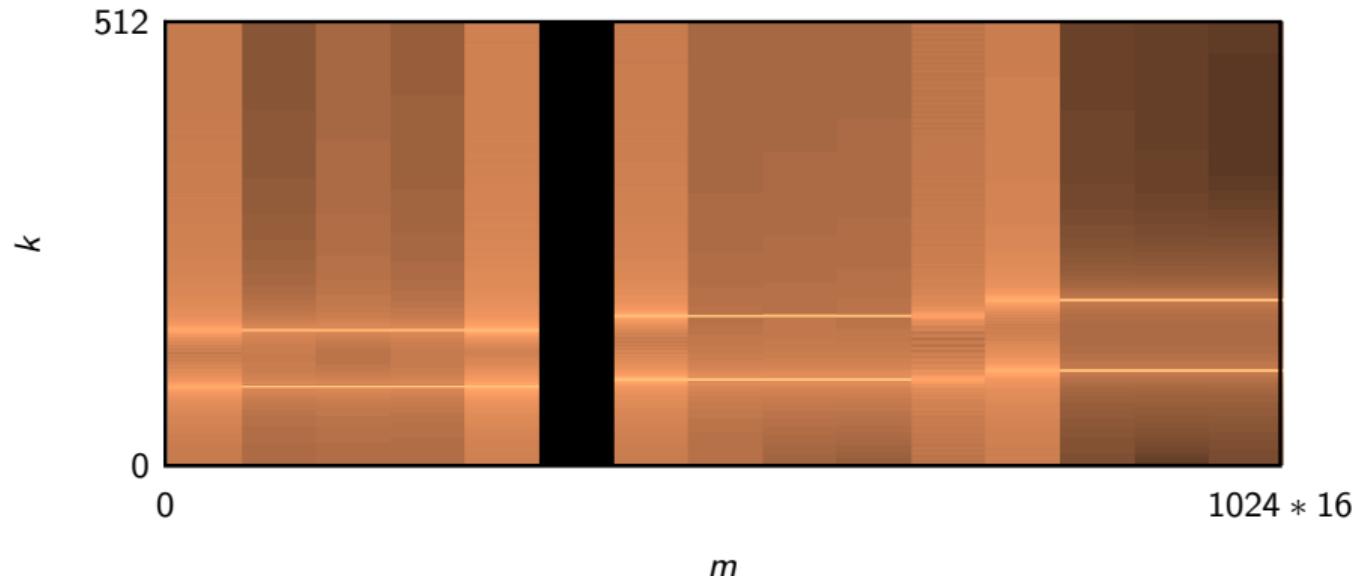
DTMF spectrogram

$N = 16800, L = 256$

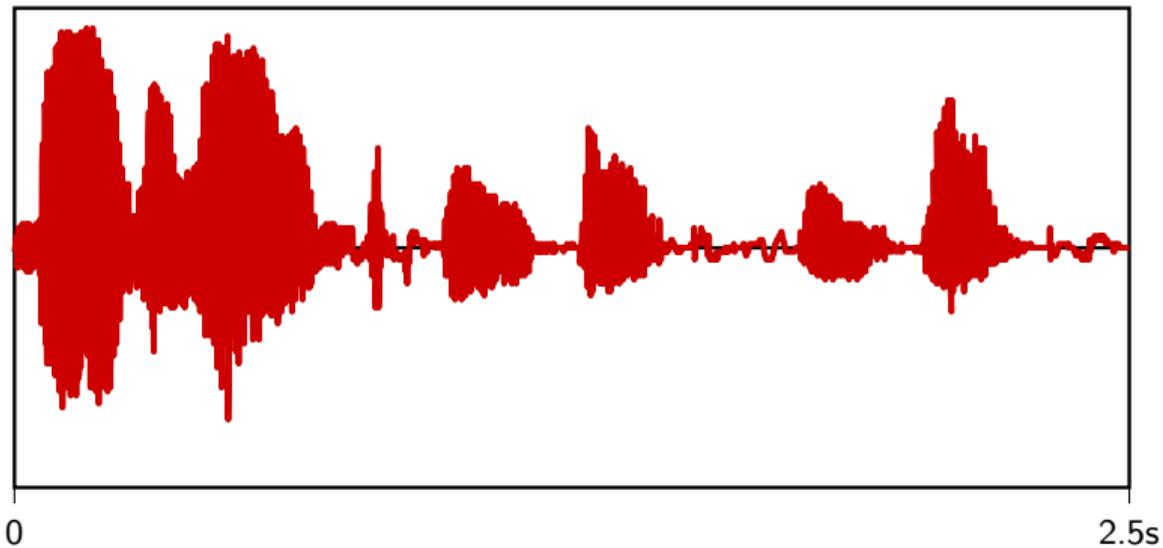


DTMF spectrogram (narrowband)

$N = 16800, L = 1024$



Speech analysis



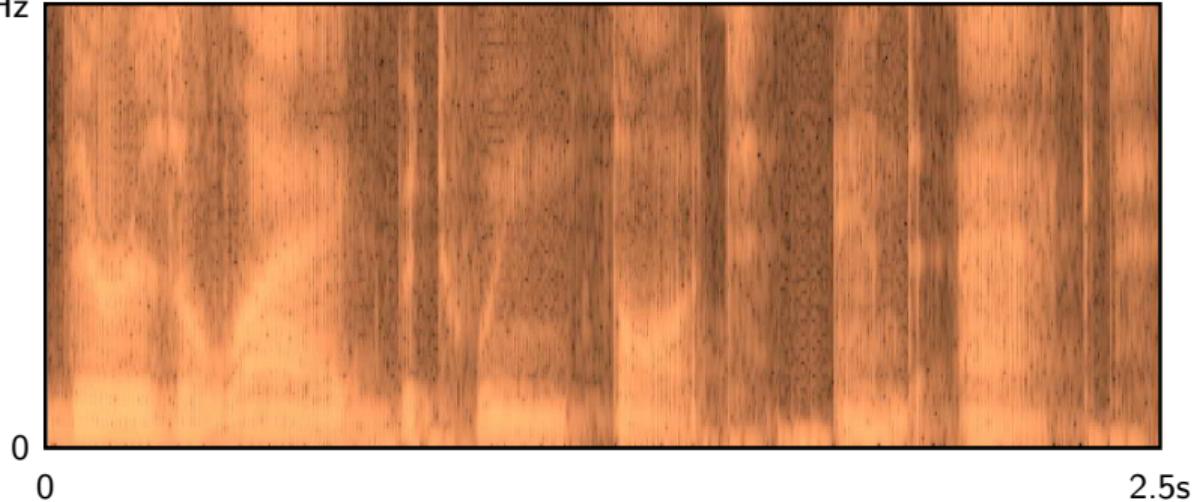
Play

Speech analysis

8ms analysis window (125Hz frequency bins) , 4ms shifts

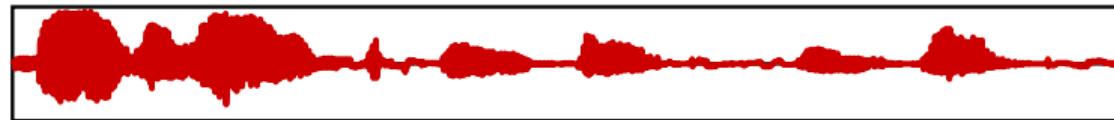


4KHz

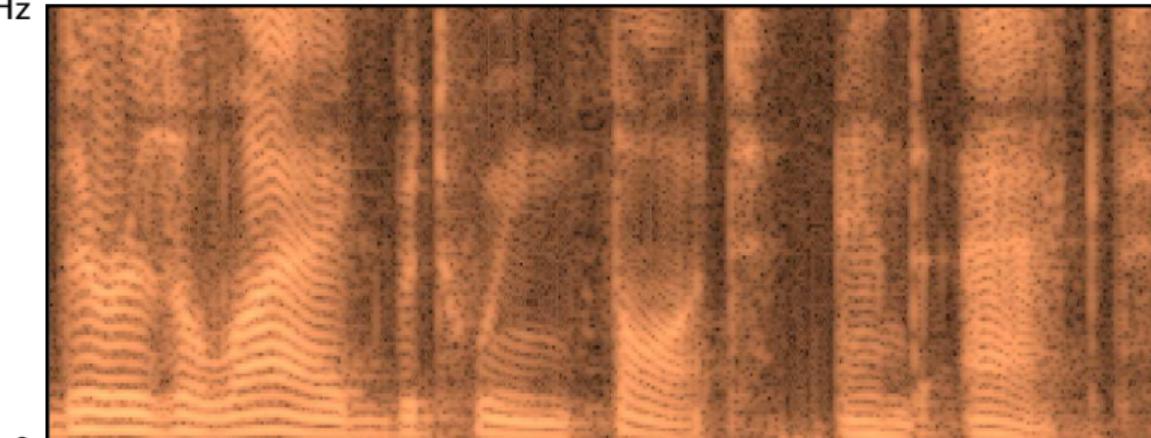


Speech analysis

32ms analysis window (31Hz frequency bins), 4ms shifts



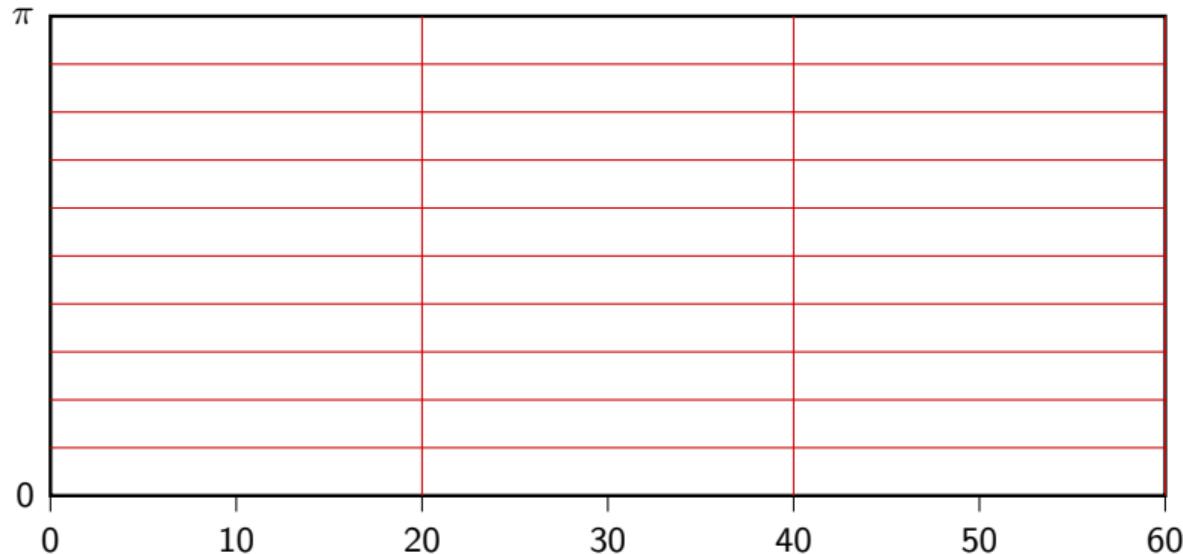
4KHz



2.5s

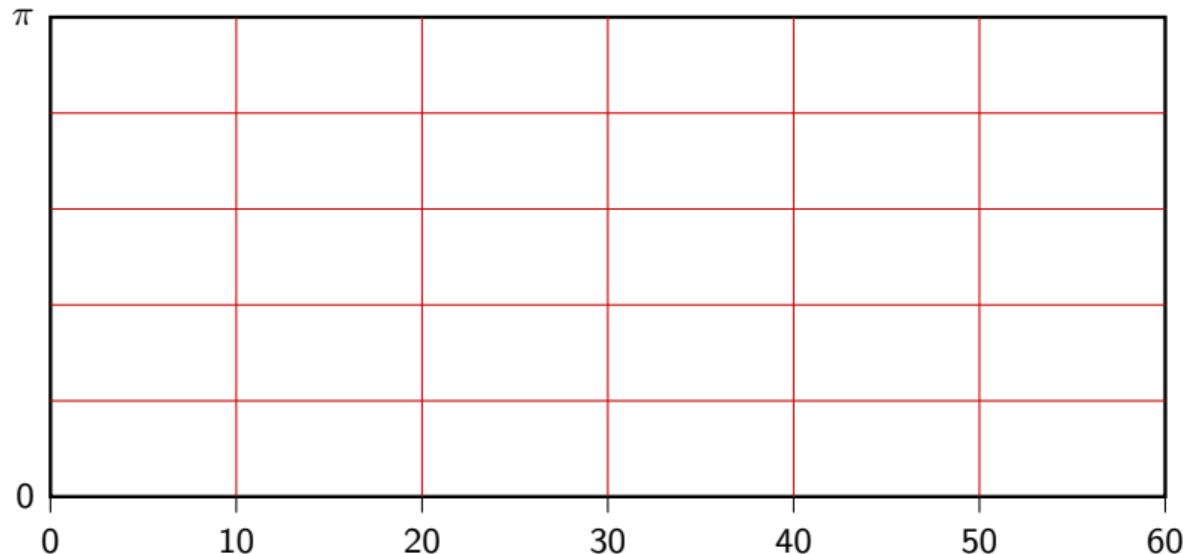
Time-Frequency tiling

$$L = 20$$



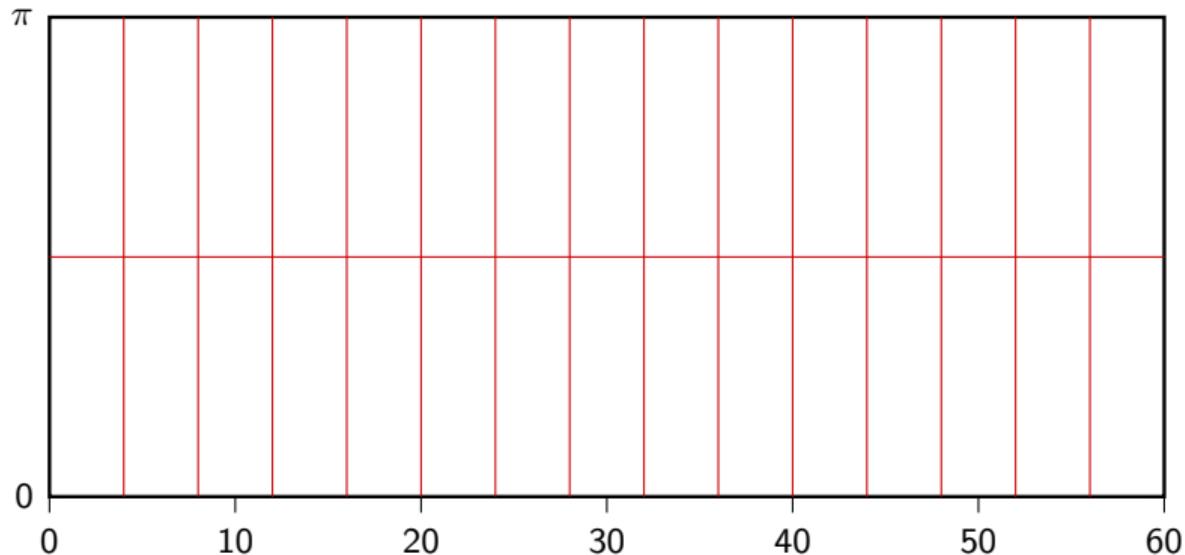
Time-Frequency tiling

$$L = 10$$



Time-Frequency tiling

$$L = 4$$



Food for thought

- ▶ time “resolution” $\Delta t = L$
- ▶ frequency “resolution” $\Delta f = 2\pi/L$
- ▶ $\Delta t \Delta f = 2\pi$

uncertainty principle!

Food for thought

- ▶ time “resolution” $\Delta t = L$
- ▶ frequency “resolution” $\Delta f = 2\pi/L$
- ▶ $\Delta t \Delta f = 2\pi$

uncertainty principle!

Food for thought

- ▶ time “resolution” $\Delta t = L$
- ▶ frequency “resolution” $\Delta f = 2\pi/L$
- ▶ $\Delta t \Delta f = 2\pi$

uncertainty principle!

Food for thought

- ▶ time “resolution” $\Delta t = L$
- ▶ frequency “resolution” $\Delta f = 2\pi/L$
- ▶ $\Delta t \Delta f = 2\pi$

uncertainty principle!

Even more food for thought

more sophisticated tilings of the time-frequency planes
can be obtained with the *wavelet* transform

the Fast Fourier Transform (FFT)

Overview

- ▶ A bit of history: From Gauss to the fastest FFT in the west
- ▶ Small DFT matrices
- ▶ The Cooley-Tukey FFT
- ▶ Decimation-in-Time FFT for length 2^N FFTs
- ▶ Conclusions: There are FFTs for any length!

Fourier had the Fourier transform



But Gauss had the FFT all along ;)



History

- ▶ Gauss computes trigonometric series efficiently in 1805
- ▶ Fourier invents Fourier series in 1807
- ▶ People start computing Fourier series, and develop tricks
- ▶ Good comes up with an algorithm in 1958
- ▶ Cooley and Tukey (re)-discover the fast Fourier transform algorithm in 1965 for N a power of a prime
- ▶ Winograd combines all methods to give the most efficient FFTs in 1978
- ▶ Frigo and Johnson develop the FFTW in 1999

History

- ▶ Gauss computes trigonometric series efficiently in 1805
- ▶ Fourier invents Fourier series in 1807
- ▶ People start computing Fourier series, and develop tricks
- ▶ Good comes up with an algorithm in 1958
- ▶ Cooley and Tukey (re)-discover the fast Fourier transform algorithm in 1965 for N a power of a prime
- ▶ Winograd combines all methods to give the most efficient FFTs in 1978
- ▶ Frigo and Johnson develop the FFTW in 1999

History

- ▶ Gauss computes trigonometric series efficiently in 1805
- ▶ Fourier invents Fourier series in 1807
- ▶ People start computing Fourier series, and develop tricks
- ▶ Good comes up with an algorithm in 1958
- ▶ Cooley and Tukey (re)-discover the fast Fourier transform algorithm in 1965 for N a power of a prime
- ▶ Winograd combines all methods to give the most efficient FFTs in 1978
- ▶ Frigo and Johnson develop the FFTW in 1999

History

- ▶ Gauss computes trigonometric series efficiently in 1805
- ▶ Fourier invents Fourier series in 1807
- ▶ People start computing Fourier series, and develop tricks
- ▶ Good comes up with an algorithm in 1958
- ▶ Cooley and Tukey (re)-discover the fast Fourier transform algorithm in 1965 for N a power of a prime
- ▶ Winograd combines all methods to give the most efficient FFTs in 1978
- ▶ Frigo and Johnson develop the FFTW in 1999

History

- ▶ Gauss computes trigonometric series efficiently in 1805
- ▶ Fourier invents Fourier series in 1807
- ▶ People start computing Fourier series, and develop tricks
- ▶ Good comes up with an algorithm in 1958
- ▶ Cooley and Tukey (re)-discover the fast Fourier transform algorithm in 1965 for N a power of a prime
- ▶ Winograd combines all methods to give the most efficient FFTs in 1978
- ▶ Frigo and Johnson develop the FFTW in 1999

History

- ▶ Gauss computes trigonometric series efficiently in 1805
- ▶ Fourier invents Fourier series in 1807
- ▶ People start computing Fourier series, and develop tricks
- ▶ Good comes up with an algorithm in 1958
- ▶ Cooley and Tukey (re)-discover the fast Fourier transform algorithm in 1965 for N a power of a prime
- ▶ Winograd combines all methods to give the most efficient FFTs in 1978
- ▶ Frigo and Johnson develop the FFTW in 1999

History

- ▶ Gauss computes trigonometric series efficiently in 1805
- ▶ Fourier invents Fourier series in 1807
- ▶ People start computing Fourier series, and develop tricks
- ▶ Good comes up with an algorithm in 1958
- ▶ Cooley and Tukey (re)-discover the fast Fourier transform algorithm in 1965 for N a power of a prime
- ▶ Winograd combines all methods to give the most efficient FFTs in 1978
- ▶ Frigo and Johnson develop the FFTW in 1999

The DFT matrix

- ▶ $W_N = e^{-j\frac{2\pi}{N}}$ (or simply W when N is clear from the context)
- ▶ powers of N can be taken modulo N , since $W^N = 1$.
- ▶ DFT Matrix of size N by N :

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W^1 & W^2 & W^3 & \dots & W^{N-1} \\ 1 & W^2 & W^4 & W^6 & \dots & W^{2(N-1)} \\ & & & \ddots & & \\ 1 & W^{N-1} & W^{2(N-1)} & W^{3(N-1)} & \dots & W^{(N-1)^2} \end{bmatrix}$$

The DFT matrix

- ▶ $W_N = e^{-j\frac{2\pi}{N}}$ (or simply W when N is clear from the context)
- ▶ powers of N can be taken modulo N , since $W^N = 1$.
- ▶ DFT Matrix of size N by N :

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W^1 & W^2 & W^3 & \dots & W^{N-1} \\ 1 & W^2 & W^4 & W^6 & \dots & W^{2(N-1)} \\ & & & \ddots & & \\ 1 & W^{N-1} & W^{2(N-1)} & W^{3(N-1)} & \dots & W^{(N-1)^2} \end{bmatrix}$$

The DFT matrix

- ▶ $W_N = e^{-j\frac{2\pi}{N}}$ (or simply W when N is clear from the context)
- ▶ powers of N can be taken modulo N , since $W^N = 1$.
- ▶ DFT Matrix of size N by N :

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W^1 & W^2 & W^3 & \dots & W^{N-1} \\ 1 & W^2 & W^4 & W^6 & \dots & W^{2(N-1)} \\ & & & \ddots & & \\ 1 & W^{N-1} & W^{2(N-1)} & W^{3(N-1)} & \dots & W^{(N-1)^2} \end{bmatrix}$$

Small DFT matrices: $N = 2$

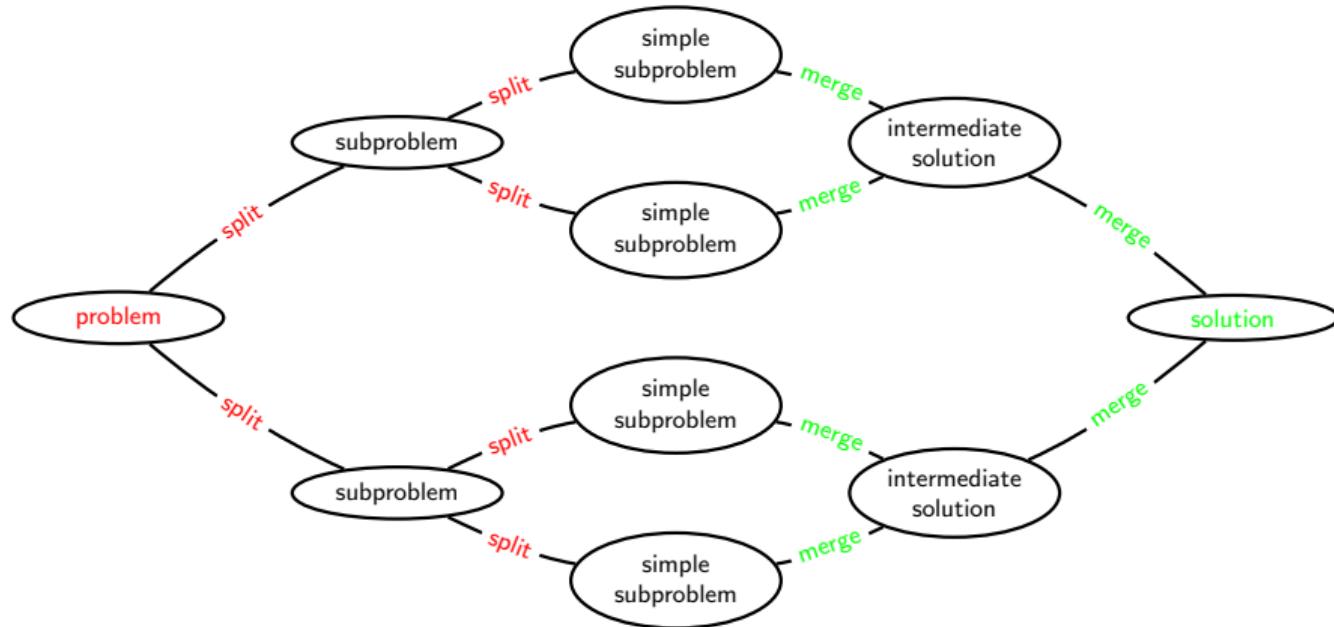
$$\mathbf{w}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Small DFT matrices: $N = 4$

$$\mathbf{W}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 \\ 1 & W^2 & W^4 & W^6 \\ 1 & W^3 & W^6 & W^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 \\ 1 & W^2 & 1 & W^2 \\ 1 & W^3 & W^2 & W \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

Divide et impera - Divide and Conquer (Julius Caesar)

Divide and conquer is a standard attack for developing fast algorithms.



Divide and Conquer for DFT - One step

Recall: computing $\mathbf{X} = \mathbf{W}_N \mathbf{x}$ has complexity $O(N^2)$.

Idea:

- ▶ Assume N even
- ▶ Split the problem into two subproblems of size $N/2$; cost is $N^2/4$ each
- ▶ If the cost to recover the full solution is linear N ...
 - ▶ ... the divide-and-conquer solution costs $N^2/2 + N$ for one step
 - ▶ For $N \geq 4$ this is better than N^2

Divide and Conquer for DFT - One step

Recall: computing $\mathbf{X} = \mathbf{W}_N \mathbf{x}$ has complexity $O(N^2)$.

Idea:

- ▶ Assume N even
- ▶ Split the problem into two subproblems of size $N/2$; cost is $N^2/4$ each
- ▶ *If* the cost to recover the full solution is linear N ...
- ▶ ... the divide-and-conquer solution costs $N^2/2 + N$ for one step
- ▶ For $N \geq 4$ this is better than N^2

Divide and Conquer for DFT - One step

Recall: computing $\mathbf{X} = \mathbf{W}_N \mathbf{x}$ has complexity $O(N^2)$.

Idea:

- ▶ Assume N even
- ▶ Split the problem into two subproblems of size $N/2$; cost is $N^2/4$ each
- ▶ *If* the cost to recover the full solution is linear N ...
 - ▶ ... the divide-and-conquer solution costs $N^2/2 + N$ for one step
 - ▶ For $N \geq 4$ this is better than N^2

Divide and Conquer for DFT - One step

Recall: computing $\mathbf{X} = \mathbf{W}_N \mathbf{x}$ has complexity $O(N^2)$.

Idea:

- ▶ Assume N even
- ▶ Split the problem into two subproblems of size $N/2$; cost is $N^2/4$ each
- ▶ *If* the cost to recover the full solution is linear N ...
 - ▶ ... the divide-and-conquer solution costs $N^2/2 + N$ for one step
 - ▶ For $N \geq 4$ this is better than N^2

Divide and Conquer for DFT - One step

Recall: computing $\mathbf{X} = \mathbf{W}_N \mathbf{x}$ has complexity $O(N^2)$.

Idea:

- ▶ Assume N even
- ▶ Split the problem into two subproblems of size $N/2$; cost is $N^2/4$ each
- ▶ *If* the cost to recover the full solution is linear N ...
 - ▶ ... the divide-and-conquer solution costs $N^2/2 + N$ for one step
- ▶ For $N \geq 4$ this is better than N^2

Divide and Conquer for DFT - One step

Graphically

- ▶ Split DFT input into 2 pieces of size $N/2$

Divide and Conquer for DFT - One step

Graphically

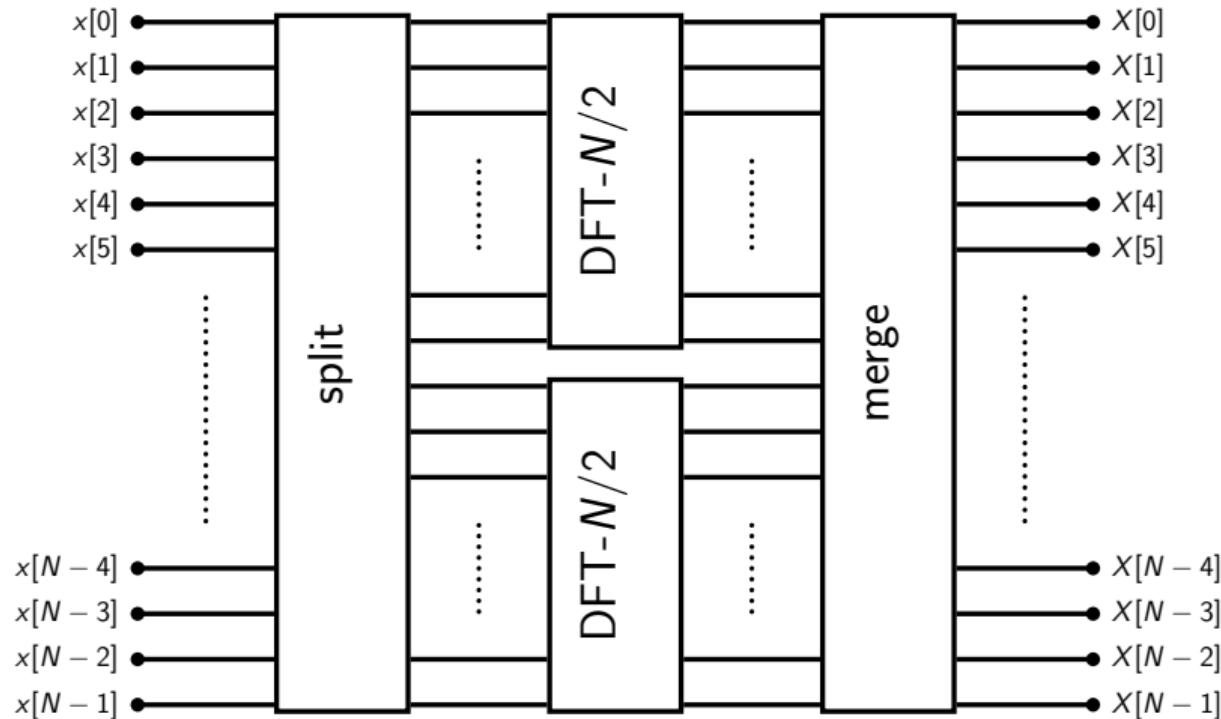
- ▶ Split DFT input into 2 pieces of size $N/2$
- ▶ Compute two DFT's of size $N/2$

Divide and Conquer for DFT - One step

Graphically

- ▶ Split DFT input into 2 pieces of size $N/2$
- ▶ Compute two DFT's of size $N/2$
- ▶ Merge the two results

Divide and Conquer for DFT - One step



Divide and Conquer for DFT - Multiple steps

Idea: if $N = 2^K$, divide and conquer can be reapplied!

- ▶ Cut the two problems of size $N/2$ into 4 problems of size $N/4$
- ▶ Assume complexity to recover the full solution still linear, e.g. N at each step
- ▶ You can do this $\log_2 N - 1 = K - 1$ times, until problem of size 2 is obtained
- ▶ The divide-and-conquer solution has therefore complexity of order $N \log_2 N$
- ▶ For $N \geq 4$ this is much better than N^2 !

Divide and Conquer for DFT - Multiple steps

Idea: if $N = 2^K$, divide and conquer can be reapplied!

- ▶ Cut the two problems of size $N/2$ into 4 problems of size $N/4$
- ▶ Assume complexity to recover the full solution still linear, e.g. N at each step
- ▶ You can do this $\log_2 N - 1 = K - 1$ times, until problem of size 2 is obtained
- ▶ The divide-and-conquer solution has therefore complexity of order $N \log_2 N$
- ▶ For $N \geq 4$ this is much better than N^2 !

Divide and Conquer for DFT - Multiple steps

Idea: if $N = 2^K$, divide and conquer can be reapplied!

- ▶ Cut the two problems of size $N/2$ into 4 problems of size $N/4$
- ▶ Assume complexity to recover the full solution still linear, e.g. N at each step
- ▶ You can do this $\log_2 N - 1 = K - 1$ times, until problem of size 2 is obtained
- ▶ The divide-and-conquer solution has therefore complexity of order $N \log_2 N$
- ▶ For $N \geq 4$ this is much better than N^2 !

Divide and Conquer for DFT - Multiple steps

Idea: if $N = 2^K$, divide and conquer can be reapplied!

- ▶ Cut the two problems of size $N/2$ into 4 problems of size $N/4$
- ▶ Assume complexity to recover the full solution still linear, e.g. N at each step
- ▶ You can do this $\log_2 N - 1 = K - 1$ times, until problem of size 2 is obtained
- ▶ The divide-and-conquer solution has therefore complexity of order $N \log_2 N$
- ▶ For $N \geq 4$ this is much better than N^2 !

Divide and Conquer for DFT - Multiple steps

Idea: if $N = 2^K$, divide and conquer can be reapplied!

- ▶ Cut the two problems of size $N/2$ into 4 problems of size $N/4$
- ▶ Assume complexity to recover the full solution still linear, e.g. N at each step
- ▶ You can do this $\log_2 N - 1 = K - 1$ times, until problem of size 2 is obtained
- ▶ The divide-and-conquer solution has therefore complexity of order $N \log_2 N$
- ▶ For $N \geq 4$ this is much better than N^2 !

Divide and Conquer for DFT - Multiple steps

N	N^2	$N \log N$
10	100	10
100	10,000	200
1000	1M	3000
10,000	100M (10^8)	40,000 ($4 \cdot 10^4$)
100,000	10B (10^{10})	500,000 ($5 \cdot 10^5$)

Divide and Conquer for DFT - Multiple steps

Graphically

- ▶ Split DFT input into 2, 4 and 8 pieces of sizes $N/2$, $N/4$ and $N/8$, respectively
- ▶ Compute 8 DFT's of size $N/8$
- ▶ Merge the results successively into DFT's of size $N/4$, $N/2$ and finally N

Divide and Conquer for DFT - Multiple steps

Graphically

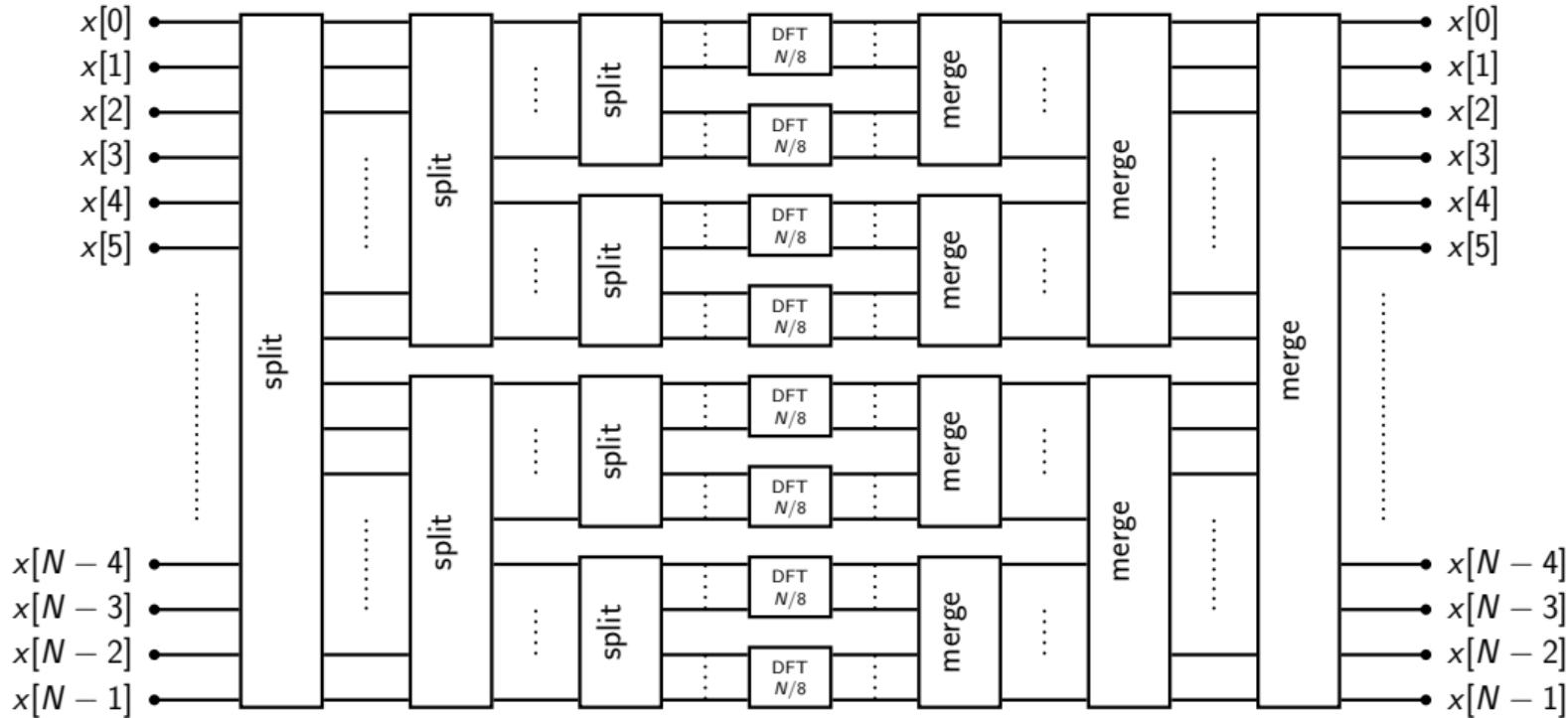
- ▶ Split DFT input into 2, 4 and 8 pieces of sizes $N/2$, $N/4$ and $N/8$, respectively
- ▶ Compute 8 DFT's of size $N/8$
- ▶ Merge the results successively into DFT's of size $N/4$, $N/2$ and finally N

Divide and Conquer for DFT - Multiple steps

Graphically

- ▶ Split DFT input into 2, 4 and 8 pieces of sizes $N/2$, $N/4$ and $N/8$, respectively
- ▶ Compute 8 DFT's of size $N/8$
- ▶ Merge the results successively into DFT's of size $N/4$, $N/2$ and finally N

Divide and Conquer for DFT - Multiple steps



Divide and Conquer for DFT- Analysis of DIT

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \quad k = 0, 1, \dots, N-1, \quad W_N = e^{-j\frac{2\pi}{N}}$$

Idea (a good guess is half of the answer!):

- ▶ break input into even and odd indexed terms (so-called "decimation in time"):

$$x[n], \quad n = 0, 1, \dots, N-1 \longrightarrow x[2n] \text{ and } x[2n+1], \quad n = 0, \dots, N/2-1$$

- ▶ break output into first and second half

$$X[k], \quad k = 0, 1, \dots, N-1 \longrightarrow X[k] \text{ and } X[k+N/2], \quad k = 0, \dots, N/2-1$$

Divide and Conquer for DFT- Analysis of DIT

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \quad k = 0, 1, \dots, N-1, \quad W_N = e^{-j\frac{2\pi}{N}}$$

Idea (a good guess is half of the answer!):

- ▶ break input into even and odd indexed terms (so-called "decimation in time"):

$$x[n], \quad n = 0, 1, \dots, N-1 \longrightarrow x[2n] \text{ and } x[2n+1], \quad n = 0, \dots, N/2-1$$

- ▶ break output into first and second half

$$X[k], \quad k = 0, 1, \dots, N-1 \longrightarrow X[k] \text{ and } X[k+N/2], \quad k = 0, \dots, N/2-1$$

Divide and Conquer for DFT- Analysis of DIT

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \quad k = 0, 1, \dots, N-1, \quad W_N = e^{-j\frac{2\pi}{N}}$$

Idea (a good guess is half of the answer!):

- ▶ break input into even and odd indexed terms (so-called "decimation in time"):

$$x[n], \quad n = 0, 1, \dots, N-1 \longrightarrow x[2n] \text{ and } x[2n+1], \quad n = 0, \dots, N/2-1$$

- ▶ break output into first and second half

$$X[k], \quad k = 0, 1, \dots, N-1 \longrightarrow X[k] \text{ and } X[k + N/2], \quad k = 0, \dots, N/2-1$$

Divide and Conquer for DFT- Analysis of DIT

Consider even and odd inputs separately:

$$\begin{aligned} X[k] &= \sum_{n=0}^{N/2-1} x[2n] W_N^{2nk} + \sum_{n=0}^{N/2-1} x[2n+1] W_N^{(2n+1)k} \\ &= \sum_{n=0}^{N/2-1} x[2n] W_N^{2nk} + \sum_{n=0}^{N/2-1} x[2n+1] W_N^{2nk+k} \\ &= \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{nk} + W^k \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{nk} \\ &= X_A[k] + W_N^k X_B[k], \quad k = 0, 1, \dots, N-1 \end{aligned}$$

Divide and Conquer for DFT- Analysis of DIT

Consider even and odd inputs separately:

$$\begin{aligned} X[k] &= \sum_{n=0}^{N/2-1} x[2n] W_N^{2nk} + \sum_{n=0}^{N/2-1} x[2n+1] W_N^{(2n+1)k} \\ &= \sum_{n=0}^{N/2-1} x[2n] W_N^{2nk} + \sum_{n=0}^{N/2-1} x[2n+1] W_N^{2nk+k} \\ &= \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{nk} + W^k \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{nk} \\ &= X_A[k] + W_N^k X_B[k], \quad k = 0, 1, \dots, N-1 \end{aligned}$$

Divide and Conquer for DFT- Analysis of DIT

Consider even and odd inputs separately:

$$\begin{aligned} X[k] &= \sum_{n=0}^{N/2-1} x[2n] W_N^{2nk} + \sum_{n=0}^{N/2-1} x[2n+1] W_N^{(2n+1)k} \\ &= \sum_{n=0}^{N/2-1} x[2n] W_N^{2nk} + \sum_{n=0}^{N/2-1} x[2n+1] W_N^{2nk+k} \\ &= \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{nk} + W^k \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{nk} \\ &= X_A[k] + W_N^k X_B[k], \quad k = 0, 1, \dots, N-1 \end{aligned}$$

Divide and Conquer for DFT- Analysis of DIT

Consider even and odd inputs separately:

$$\begin{aligned} X[k] &= \sum_{n=0}^{N/2-1} x[2n] W_N^{2nk} + \sum_{n=0}^{N/2-1} x[2n+1] W_N^{(2n+1)k} \\ &= \sum_{n=0}^{N/2-1} x[2n] W_N^{2nk} + \sum_{n=0}^{N/2-1} x[2n+1] W_N^{2nk+k} \\ &= \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{nk} + W^k \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{nk} \\ &= X_A[k] + W_N^k X_B[k], \quad k = 0, 1, \dots, N-1 \end{aligned}$$

Divide and Conquer for DFT- Analysis of DIT

hmmm, we haven't gained much so far:

- ▶ both $X_A[k]$ and $X_B[k]$ require $N/2$ multiplications
- ▶ multiplying the second DFT by W_N^k requires another multiplication
- ▶ to compute for all k we need $N(N/2 + N/2 + 1) \approx N^2$
- ▶ but here comes the trick!

Divide and Conquer for DFT- Analysis of DIT

hmmm, we haven't gained much so far:

- ▶ both $X_A[k]$ and $X_B[k]$ require $N/2$ multiplications
- ▶ multiplying the second DFT by W_N^k requires another multiplication
- ▶ to compute for all k we need $N(N/2 + N/2 + 1) \approx N^2$
- ▶ but here comes the trick!

Divide and Conquer for DFT- Analysis of DIT

hmmm, we haven't gained much so far:

- ▶ both $X_A[k]$ and $X_B[k]$ require $N/2$ multiplications
- ▶ multiplying the second DFT by W_N^k requires another multiplication
- ▶ to compute for all k we need $N(N/2 + N/2 + 1) \approx N^2$
- ▶ but here comes the trick!

Divide and Conquer for DFT- Analysis of DIT

hmmm, we haven't gained much so far:

- ▶ both $X_A[k]$ and $X_B[k]$ require $N/2$ multiplications
- ▶ multiplying the second DFT by W_N^k requires another multiplication
- ▶ to compute for all k we need $N(N/2 + N/2 + 1) \approx N^2$
- ▶ but here comes the trick!

Divide and Conquer for DFT- Analysis of DIT

Consider now the first and second half of the outputs separately:

$$\begin{aligned} X[k] &= X_A[k] + W^k X_B[k] \\ &= \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{nk} + W^k \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{nk} \\ X[k + N/2] &= \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{n(k+N/2)} + W_N^{k+N/2} \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{n(k+N/2)} \\ &= \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{nk} - W^k \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{nk} \\ &= X_A[k] - W_N^k X_B[k], \quad k = 0, 1, \dots, N/2 - 1 \end{aligned}$$

Divide and Conquer for DFT- Analysis of DIT

Consider now the first and second half of the outputs separately:

$$\begin{aligned} X[k] &= X_A[k] + W^k X_B[k] \\ &= \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{nk} + W^k \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{nk} \\ X[k+N/2] &= \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{n(k+N/2)} + W_N^{k+N/2} \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{n(k+N/2)} \\ &= \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{nk} - W^k \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{nk} \\ &= X_A[k] - W_N^k X_B[k], \quad k = 0, 1, \dots, N/2 - 1 \end{aligned}$$

Divide and Conquer for DFT- Analysis of DIT

Consider now the first and second half of the outputs separately:

$$\begin{aligned} X[k] &= X_A[k] + W^k X_B[k] \\ &= \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{nk} + W^k \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{nk} \\ X[k+N/2] &= \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{n(k+N/2)} + W_N^{k+N/2} \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{n(k+N/2)} \\ &= \sum_{n=0}^{N/2-1} x[2n] W_{N/2}^{nk} - W^k \sum_{n=0}^{N/2-1} x[2n+1] W_{N/2}^{nk} \\ &= X_A[k] - W_N^k X_B[k], \quad k = 0, 1, \dots, N/2 - 1 \end{aligned}$$

Divide and Conquer for DFT- Analysis of DIT

so the trick is that we only need to compute for half the range of k :

- ▶ both $X_A[k]$ and $X_B[k]$ require $N/2$ multiplications
- ▶ multiplying the second DFT by W_N^k requires another multiplication
- ▶ to compute for all k we need $(N/2)(N/2 + N/2 + 1) \approx N^2/2$
- ▶ the rest is just sums and differences

Divide and Conquer for DFT- Analysis of DIT

so the trick is that we only need to compute for half the range of k :

- ▶ both $X_A[k]$ and $X_B[k]$ require $N/2$ multiplications
- ▶ multiplying the second DFT by W_N^k requires another multiplication
- ▶ to compute for all k we need $(N/2)(N/2 + N/2 + 1) \approx N^2/2$
- ▶ the rest is just sums and differences

Divide and Conquer for DFT- Analysis of DIT

so the trick is that we only need to compute for half the range of k :

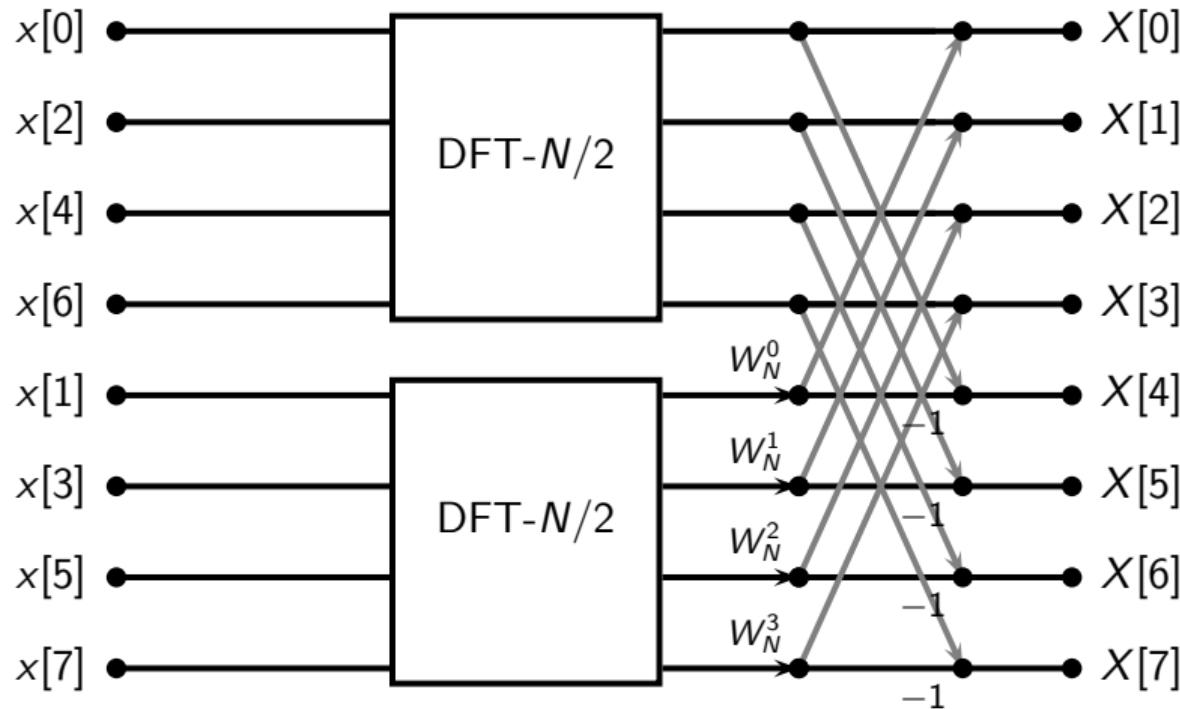
- ▶ both $X_A[k]$ and $X_B[k]$ require $N/2$ multiplications
- ▶ multiplying the second DFT by W_N^k requires another multiplication
- ▶ to compute for all k we need $(N/2)(N/2 + N/2 + 1) \approx N^2/2$
- ▶ the rest is just sums and differences

Divide and Conquer for DFT- Analysis of DIT

so the trick is that we only need to compute for half the range of k :

- ▶ both $X_A[k]$ and $X_B[k]$ require $N/2$ multiplications
- ▶ multiplying the second DFT by W_N^k requires another multiplication
- ▶ to compute for all k we need $(N/2)(N/2 + N/2 + 1) \approx N^2/2$
- ▶ the rest is just sums and differences

Divide and Conquer for DFT- Analysis of DIT



Divide and Conquer for DFT- Analysis of DIT

So, what is the complexity now?

- ▶ Split DFT input into 2 pieces of size $N/2$: free!
- ▶ Compute 2 DFT- $N/2$: twice $(N/2)^2$, or $N^2/2$
- ▶ Merge the two results: multiplication by $N/2$ complex numbers W^k
- ▶ Total: $N^2/2 + N/2$ which is indeed smaller than N^2 for any $N \geq 4$,
- ▶ In general, about half the complexity of the initial problem!

Divide and Conquer for DFT- Analysis of DIT

So, what is the complexity now?

- ▶ Split DFT input into 2 pieces of size $N/2$: free!
- ▶ Compute 2 DFT- $N/2$: twice $(N/2)^2$, or $N^2/2$
- ▶ Merge the two results: multiplication by $N/2$ complex numbers W^k
- ▶ Total: $N^2/2 + N/2$ which is indeed smaller than N^2 for any $N \geq 4$,
- ▶ In general, about half the complexity of the initial problem!

Divide and Conquer for DFT- Analysis of DIT

So, what is the complexity now?

- ▶ Split DFT input into 2 pieces of size $N/2$: free!
- ▶ Compute 2 DFT- $N/2$: twice $(N/2)^2$, or $N^2/2$
- ▶ Merge the two results: multiplication by $N/2$ complex numbers W^k
- ▶ Total: $N^2/2 + N/2$ which is indeed smaller than N^2 for any $N \geq 4$,
- ▶ In general, about half the complexity of the initial problem!

Divide and Conquer for DFT- Analysis of DIT

So, what is the complexity now?

- ▶ Split DFT input into 2 pieces of size $N/2$: free!
- ▶ Compute 2 DFT- $N/2$: twice $(N/2)^2$, or $N^2/2$
- ▶ Merge the two results: multiplication by $N/2$ complex numbers W^k
- ▶ Total: $N^2/2 + N/2$ which is indeed smaller than N^2 for any $N \geq 4$,
- ▶ In general, about half the complexity of the initial problem!

Divide and Conquer for DFT- Analysis of DIT

So, what is the complexity now?

- ▶ Split DFT input into 2 pieces of size $N/2$: free!
- ▶ Compute 2 DFT- $N/2$: twice $(N/2)^2$, or $N^2/2$
- ▶ Merge the two results: multiplication by $N/2$ complex numbers W^k
- ▶ Total: $N^2/2 + N/2$ which is indeed smaller than N^2 for any $N \geq 4$,
- ▶ In general, about half the complexity of the initial problem!

Divide and Conquer for DFT- Analysis of DIT

So, what is the complexity now?

- ▶ Split DFT input into 2 pieces of size $N/2$: free!
- ▶ Compute 2 DFT- $N/2$: twice $(N/2)^2$, or $N^2/2$
- ▶ Merge the two results: multiplication by $N/2$ complex numbers W^k
- ▶ Total: $N^2/2 + N/2$ which is indeed smaller than N^2 for any $N \geq 4$,
- ▶ In general, about half the complexity of the initial problem!

Divide and Conquer for DFT- Analysis of DIT

So, what if we repeat the process?

- ▶ Go until DFT-2, since that is trivial (sum and difference)
- ▶ Requires $\log_2 N - 1$ steps
- ▶ Each step requires a merger of order $N/2$ multiplications and N additions
- ▶ Total: $(N/2)(\log_2 N - 1)$ multiplications and $N \log_2 N$ additions

Key Result: **A DFT of size N requires order $N \log_2 N$ operations!**

Divide and Conquer for DFT- Analysis of DIT

So, what if we repeat the process?

- ▶ Go until DFT-2, since that is trivial (sum and difference)
- ▶ Requires $\log_2 N - 1$ steps
- ▶ Each step requires a merger of order $N/2$ multiplications and N additions
- ▶ Total: $(N/2)(\log_2 N - 1)$ multiplications and $N \log_2 N$ additions

Key Result: **A DFT of size N requires order $N \log_2 N$ operations!**

Divide and Conquer for DFT- Analysis of DIT

So, what if we repeat the process?

- ▶ Go until DFT-2, since that is trivial (sum and difference)
- ▶ Requires $\log_2 N - 1$ steps
- ▶ Each step requires a merger of order $N/2$ multiplications and N additions
- ▶ Total: $(N/2)(\log_2 N - 1)$ multiplications and $N \log_2 N$ additions

Key Result: A DFT of size N requires order $N \log_2 N$ operations!

Divide and Conquer for DFT- Analysis of DIT

So, what if we repeat the process?

- ▶ Go until DFT-2, since that is trivial (sum and difference)
- ▶ Requires $\log_2 N - 1$ steps
- ▶ Each step requires a merger of order $N/2$ multiplications and N additions
- ▶ Total: $(N/2)(\log_2 N - 1)$ multiplications and $N \log_2 N$ additions

Key Result: A DFT of size N requires order $N \log_2 N$ operations!

Divide and Conquer for DFT- Analysis of DIT

So, what if we repeat the process?

- ▶ Go until DFT-2, since that is trivial (sum and difference)
- ▶ Requires $\log_2 N - 1$ steps
- ▶ Each step requires a merger of order $N/2$ multiplications and N additions
- ▶ Total: $(N/2)(\log_2 N - 1)$ multiplications and $N \log_2 N$ additions

Key Result: **A DFT of size N requires order $N \log_2 N$ operations!**

Matrix factorization view of DFT, N = 4

- ▶ Separate even and odd samples
- ▶ Compute two DFT's of size 2 having output $X'[k]$ and $X''[k]$
- ▶ Compute sum and difference of $X'[k]$ and $W^k X''[k]$

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -j \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & j \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This uses 8 additions and no multiplications!

Matrix factorization view of DFT, N = 4

- ▶ Separate even and odd samples
- ▶ Compute two DFT's of size 2 having output $X'[k]$ and $X''[k]$
- ▶ Compute sum and difference of $X'[k]$ and $W^k X''[k]$

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -j \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & j \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This uses 8 additions and no multiplications!

Matrix factorization view of DFT, N = 4

- ▶ Separate even and odd samples
- ▶ Compute two DFT's of size 2 having output $X'[k]$ and $X''[k]$
- ▶ Compute sum and difference of $X'[k]$ and $W^k X''[k]$

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -j \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & j \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This uses 8 additions and no multiplications!

Matrix factorization view of DFT, N = 4

- ▶ Separate even and odd samples
- ▶ Compute two DFT's of size 2 having output $X'[k]$ and $X''[k]$
- ▶ Compute sum and difference of $X'[k]$ and $W^k X''[k]$

$$\mathbf{W}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -j \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & j \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This uses 8 additions and no multiplications!

Matrix factorization view of DFT, N = 8, 1/8

Now this is going to be big...

Too big for a single slide!

$$\mathbf{W}_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W^1 & W^2 & W^3 & \dots & W^7 \\ 1 & W^2 & W^4 & W^6 & \dots & W^{14} \\ 1 & W^7 & W^{14} & W^{21} & \dots & W^{49} \end{bmatrix} = \dots$$

Matrix factorization view of DFT, N = 8, 2/8

Step 1: separate even from odd indexed samples

Call this \mathbf{D}_8 for decimation of size 8

$$\mathbf{D}_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This requires no arithmetic operations!

Matrix factorization view of DFT, N = 8, 3/8

Step 2: Compute two DFTs of size N/2 on the even and on the odd indexed samples

Each submatrix is \mathbf{W}_4 , and the matrix is block diagonal, where $\mathbf{0}_4$ stands for a matrix of 0's

$$\begin{bmatrix} \mathbf{W}_4 & \mathbf{0}_4 \\ \mathbf{0}_4 & \mathbf{W}_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \\ & & 1 & 1 & 1 & 1 \\ & & 1 & j & -1 & -j \\ & & 1 & -1 & 1 & -1 \\ & & 1 & -j & -1 & j \end{bmatrix}$$

This requires two DFT-4, or a total of 16 additions!

Matrix factorization view of DFT, N = 8, 4/8

Step 3: Multiply output of second DFT of size 4 by W^k

This is a diagonal matrix, with \mathbf{I}_4 for the identity of size 4,

$$\begin{bmatrix} \mathbf{I}_4 & \mathbf{0}_4 \\ \mathbf{0}_4 & \Lambda_4 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \quad \text{where } \Lambda_4 = \begin{bmatrix} 1 & & & \\ & W & & \\ & & W^2 & \\ & & & W^3 \end{bmatrix}$$

This requires 2 multiplications ($W^2 = -j$ is free)

Matrix factorization view of DFT, N = 8, 5/8

Step 4: Recombine final output $X[k]$ and $X[k + N/2]$ by sum and difference, \mathbf{S}_8

$$\mathbf{S}_8 = \begin{bmatrix} \mathbf{I}_4 & \mathbf{I}_4 \\ \mathbf{I}_4 & -\mathbf{I}_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}$$

This requires 8 additions!

Matrix factorization view of DFT, N = 8, 6/8

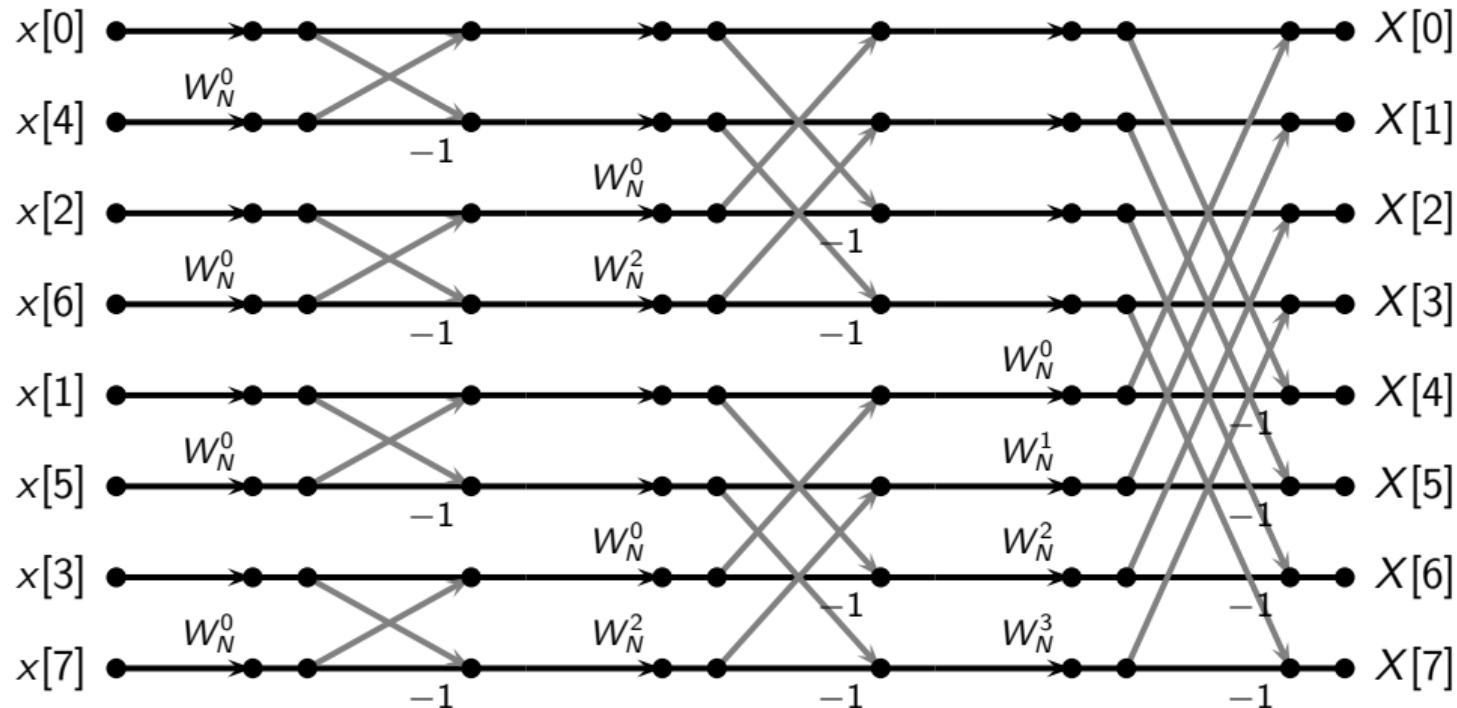
In total:

Product of 4 matrices

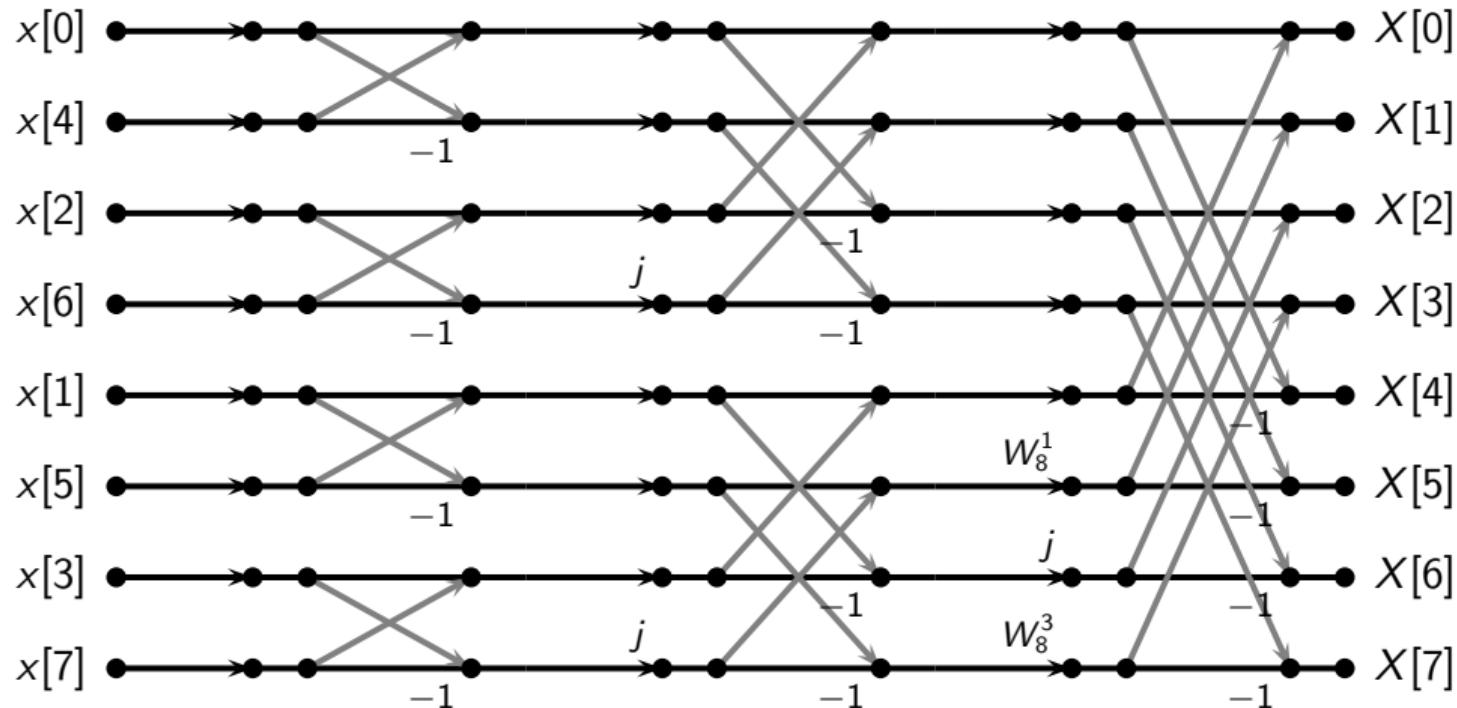
$$\mathbf{W}_8 = \begin{bmatrix} \mathbf{I}_4 & \mathbf{I}_4 \\ \mathbf{I}_4 & -\mathbf{I}_4 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I}_4 & \mathbf{0}_4 \\ \mathbf{0}_4 & \Lambda_4 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{W}_4 & \mathbf{0}_4 \\ \mathbf{0}_4 & \mathbf{W}_4 \end{bmatrix} \cdot \mathbf{D}_8$$

This requires 24 additions and 2 multiplications!

Flowgraph view of FFT, $N = 8$



Flowgraph view of FFT, $N = 8$



Matrix factorization view of DFT, N = 8, 8/8

Is this a big deal?

- ▶ In image processing (e.g. digital photography) one takes block of 8 by 8 pixels
- ▶ One computes a transform (called DCT) similar to a DFT
- ▶ It has a fast algorithm inspired by what we just saw

Matrix factorization view of DFT, N = 8, 8/8

Is this a big deal?

- ▶ In image processing (e.g. digital photography) one takes block of 8 by 8 pixels
- ▶ One computes a transform (called DCT) similar to a DFT
- ▶ It has a fast algorithm inspired by what we just saw

Matrix factorization view of DFT, N = 8, 8/8

Is this a big deal?

- ▶ In image processing (e.g. digital photography) one takes block of 8 by 8 pixels
- ▶ One computes a transform (called DCT) similar to a DFT
- ▶ It has a fast algorithm inspired by what we just saw

Some examples

image processing (JPEG compression)

- ▶ image is divided into 8×8 -pixel blocks
- ▶ DFT performed on rows and columns: 16 8-point DFTs
- ▶ direct computation: $16 \times 8^2 = 1024$ multiplications
- ▶ FFT: $16 \times 2 = 32$ multiplications

Some examples

image processing (JPEG compression)

- ▶ image is divided into 8×8 -pixel blocks
- ▶ DFT performed on rows and columns: 16 8-point DFTs
- ▶ direct computation: $16 \times 8^2 = 1024$ multiplications
- ▶ FFT: $16 \times 2 = 32$ multiplications

Some examples

image processing (JPEG compression)

- ▶ image is divided into 8×8 -pixel blocks
- ▶ DFT performed on rows and columns: 16 8-point DFTs
- ▶ direct computation: $16 \times 8^2 = 1024$ multiplications
- ▶ FFT: $16 \times 2 = 32$ multiplications

Some examples

image processing (JPEG compression)

- ▶ image is divided into 8×8 -pixel blocks
- ▶ DFT performed on rows and columns: 16 8-point DFTs
- ▶ direct computation: $16 \times 8^2 = 1024$ multiplications
- ▶ FFT: $16 \times 2 = 32$ multiplications

Some examples

audio processing (MP3 compression)

- ▶ audio is split into 1152-point frames
- ▶ direct DFT computation: $1.3 \cdot 10^6$ multiplications
- ▶ FFT: 3500 multiplications

Some examples

audio processing (MP3 compression)

- ▶ audio is split into 1152-point frames
- ▶ direct DFT computation: $1.3 \cdot 10^6$ multiplications
- ▶ FFT: 3500 multiplications

Some examples

audio processing (MP3 compression)

- ▶ audio is split into 1152-point frames
- ▶ direct DFT computation: $1.3 \cdot 10^6$ multiplications
- ▶ FFT: 3500 multiplications

Conclusions

Don't worry, be happy!

- ▶ The Cooley-Tukey is the most popular algorithm, mostly for $N = 2^N$
- ▶ Note that there is always a good FFT algorithm around the corner
- ▶ Do not zero-pad to lengthen a vector to have a size equal to a power of 2
- ▶ There are good packages out there (e.g. Fastest Fourier Transform in the West, SPIRAL)
- ▶ It does make a BIG difference!



COM303: Digital Signal Processing

Lecture 9: Linear Systems

Overview

- ▶ linear systems
- ▶ filtering by example
- ▶ stability

Overview

- ▶ linear systems
- ▶ filtering by example
- ▶ stability

Overview

- ▶ linear systems
- ▶ filtering by example
- ▶ stability

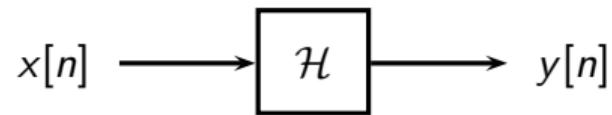
Overview:

- ▶ Linearity and time invariance
- ▶ Convolution

Overview:

- ▶ Linearity and time invariance
- ▶ Convolution

A generic signal processing device

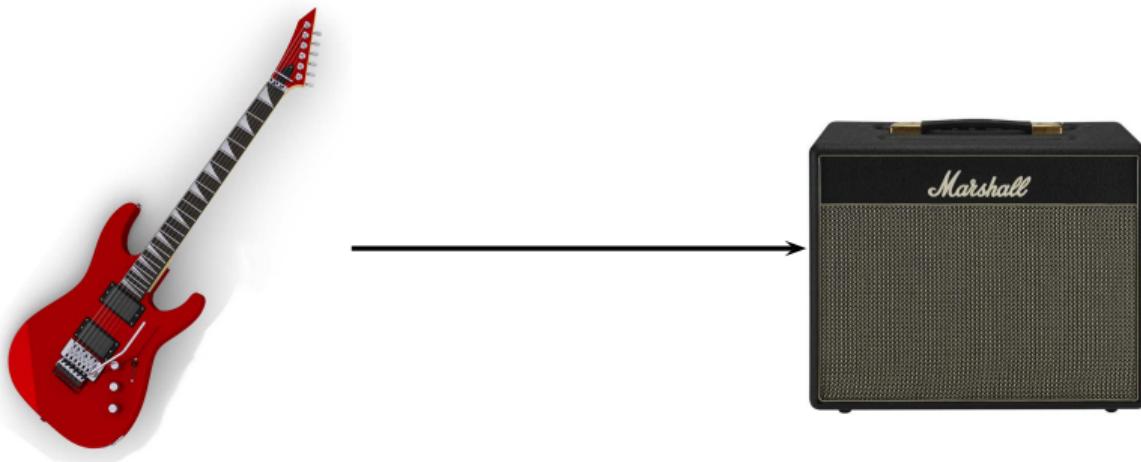


$$y[n] = \mathcal{H}\{x[n]\}$$

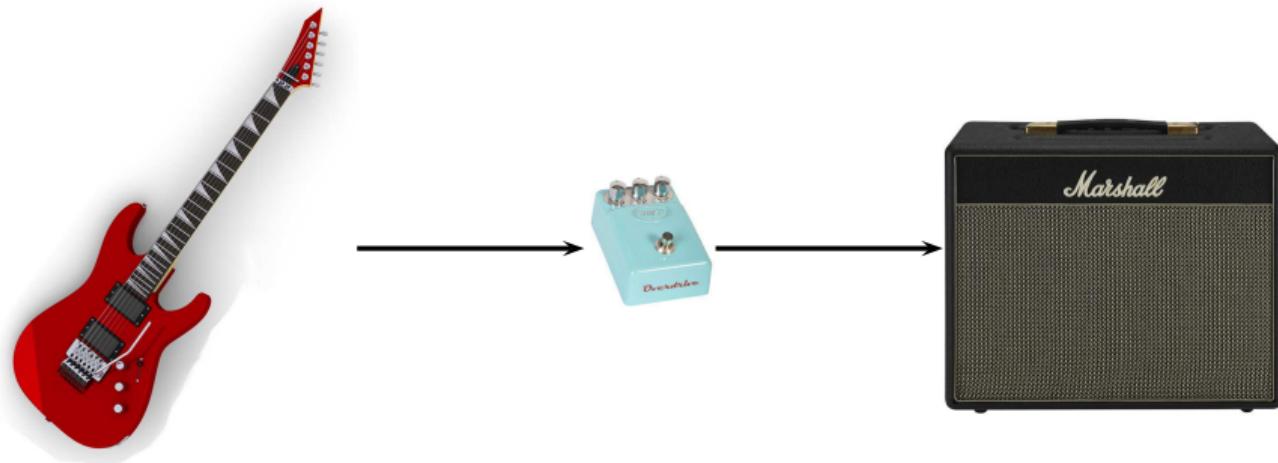
Linearity

$$\mathcal{H}\{\alpha x_1[n] + \beta x_2[n]\} = \alpha \mathcal{H}\{x_1[n]\} + \beta \mathcal{H}\{x_2[n]\}$$

Linearity



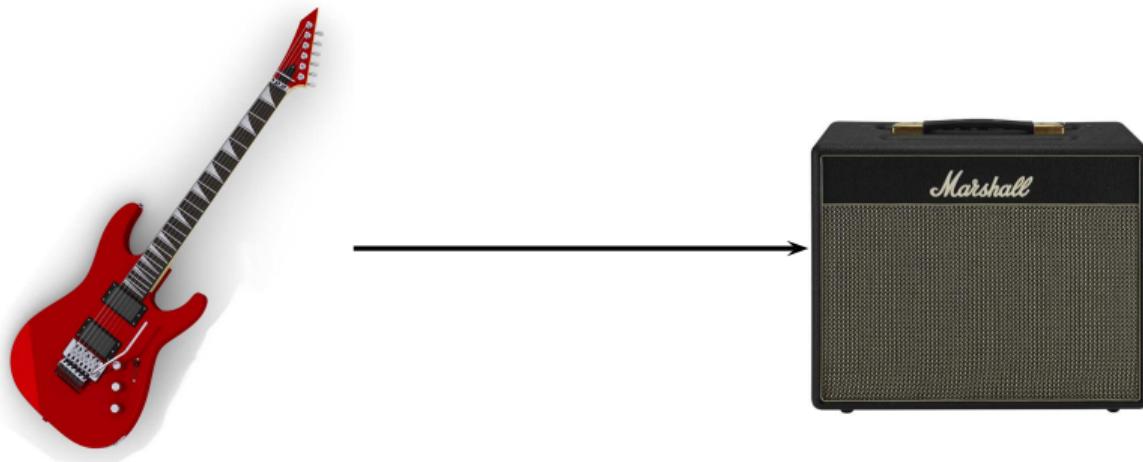
(Non) Linearity



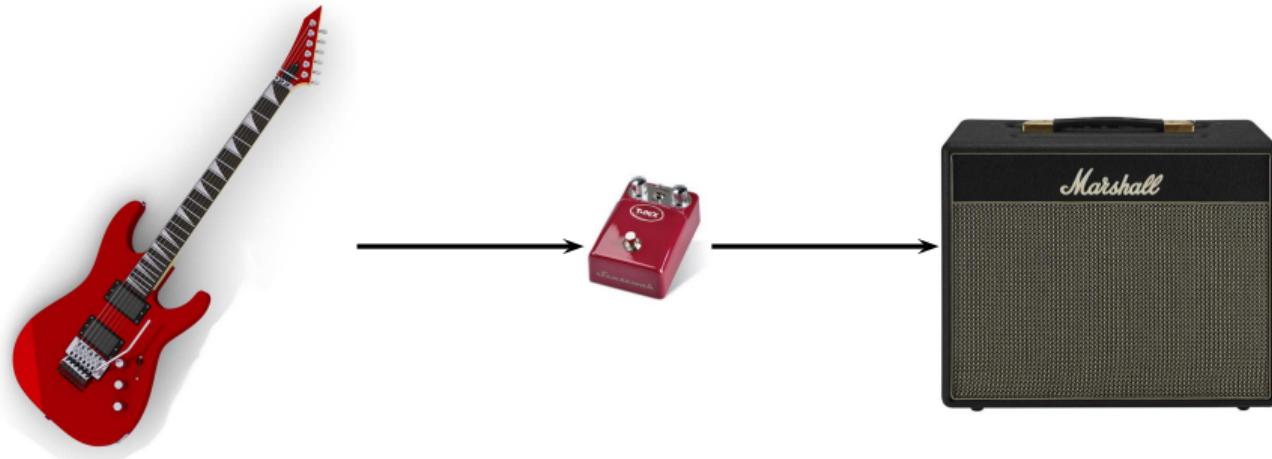
Time invariance

$$y[n] = \mathcal{H}\{x[n]\} \iff \mathcal{H}\{x[n - n_0]\} = y[n - n_0]$$

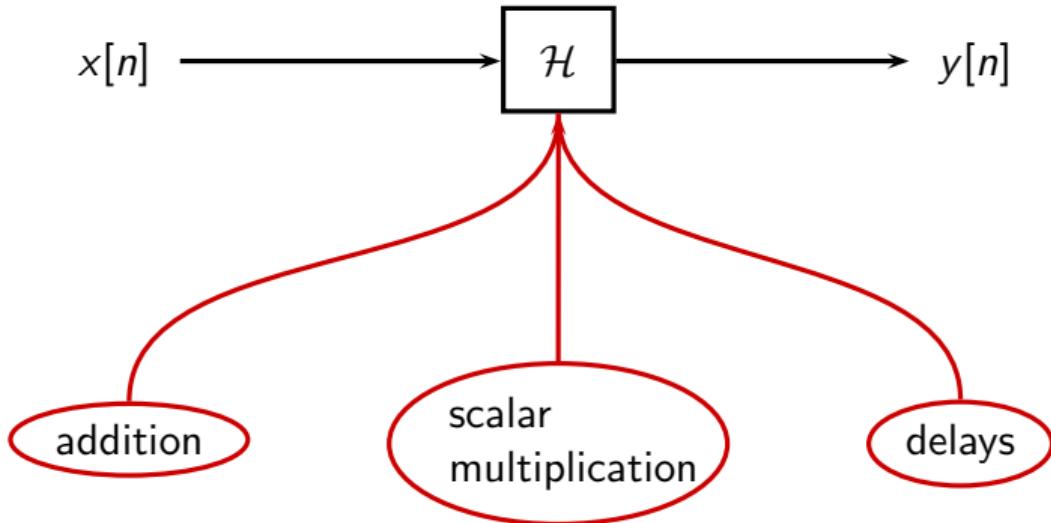
Time invariance



Time (in)variance



Linear, time-invariant systems



Linear, time-invariant systems

$$y[n] = H(x[n], x[n - 1], x[n - 2], \dots, y[n - 1], y[n - 2], \dots)$$

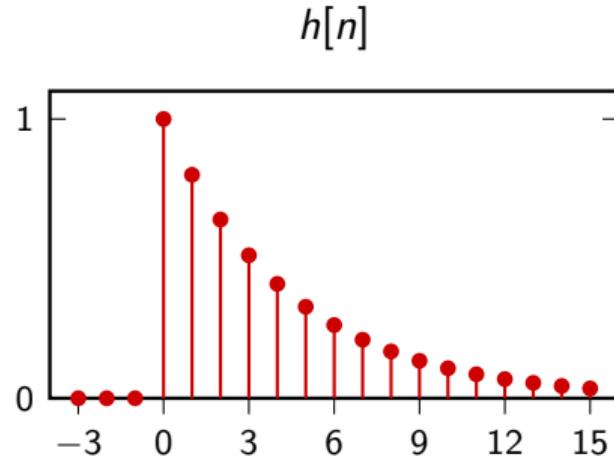
with $H(\cdot)$ a linear function of its arguments

Impulse response

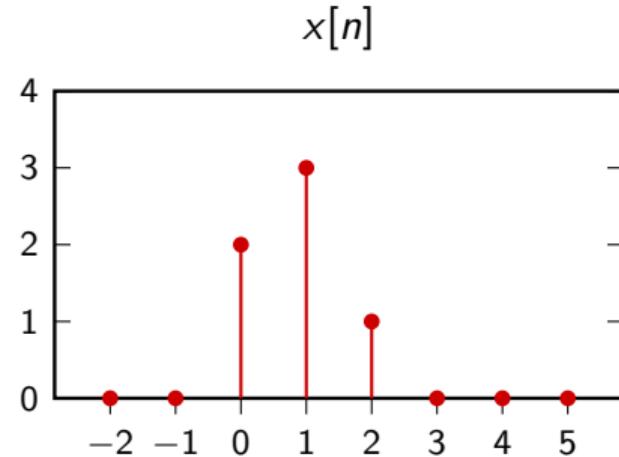
$$h[n] = \mathcal{H}\{\delta[n]\}$$

Fundamental result: impulse response fully characterizes the LTI system!

Example



$$h[n] = \alpha^n u[n]$$



$$x[n] = \begin{cases} 2 & n = 0 \\ 3 & n = 1 \\ 1 & n = 2 \\ 0 & \text{otherwise} \end{cases}$$

Example

- ▶ $x[n] = 2\delta[n] + 3\delta[n - 1] + \delta[n - 2]$
- ▶ we know the impulse response $h[n] = \mathcal{H}\{\delta[n]\}$;
- ▶ compute $y[n] = \mathcal{H}\{x[n]\}$ exploiting linearity and time-invariance

Example

- ▶ $x[n] = 2\delta[n] + 3\delta[n - 1] + \delta[n - 2]$
- ▶ we know the impulse response $h[n] = \mathcal{H}\{\delta[n]\}$;
- ▶ compute $y[n] = \mathcal{H}\{x[n]\}$ exploiting linearity and time-invariance

Example

- ▶ $x[n] = 2\delta[n] + 3\delta[n - 1] + \delta[n - 2]$
- ▶ we know the impulse response $h[n] = \mathcal{H}\{\delta[n]\}$;
- ▶ compute $y[n] = \mathcal{H}\{x[n]\}$ exploiting linearity and time-invariance

Example

$$y[n] = \mathcal{H}\{2\delta[n] + 3\delta[n - 1] + \delta[n - 2]\}$$

$$= 2\mathcal{H}\{\delta[n]\} + 3\mathcal{H}\{\delta[n - 1]\} + \mathcal{H}\{\delta[n - 2]\}$$

$$= 2h[n] + 3h[n - 1] + h[n - 2]$$

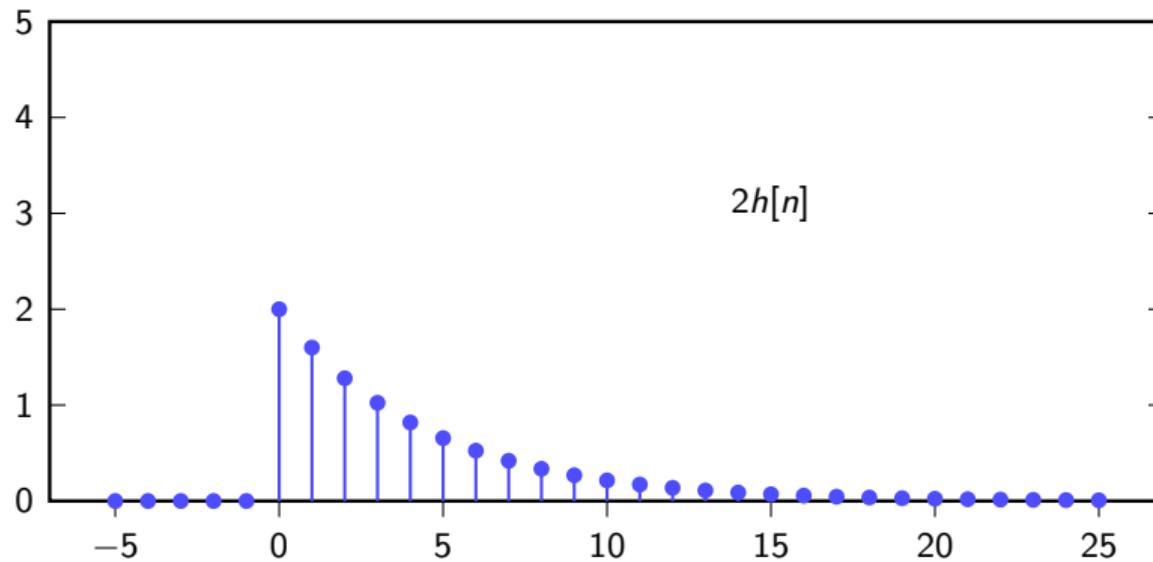
Example

$$\begin{aligned}y[n] &= \mathcal{H}\{2\delta[n] + 3\delta[n - 1] + \delta[n - 2]\} \\&= 2\mathcal{H}\{\delta[n]\} + 3\mathcal{H}\{\delta[n - 1]\} + \mathcal{H}\{\delta[n - 2]\} \\&= 2h[n] + 3h[n - 1] + h[n - 2]\end{aligned}$$

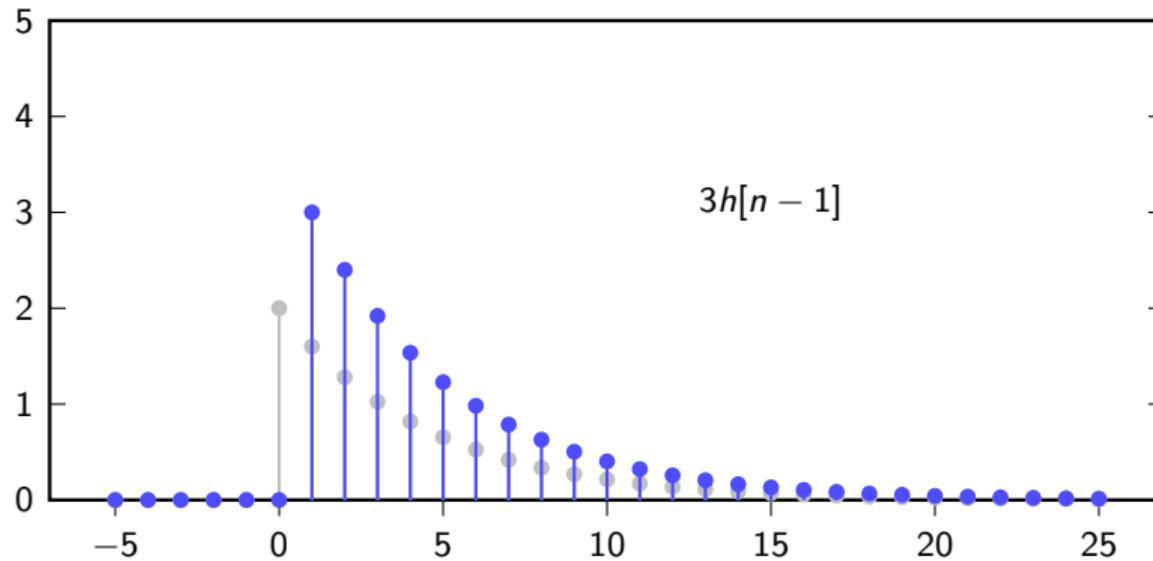
Example

$$\begin{aligned}y[n] &= \mathcal{H}\{2\delta[n] + 3\delta[n - 1] + \delta[n - 2]\} \\&= 2\mathcal{H}\{\delta[n]\} + 3\mathcal{H}\{\delta[n - 1]\} + \mathcal{H}\{\delta[n - 2]\} \\&= 2h[n] + 3h[n - 1] + h[n - 2]\end{aligned}$$

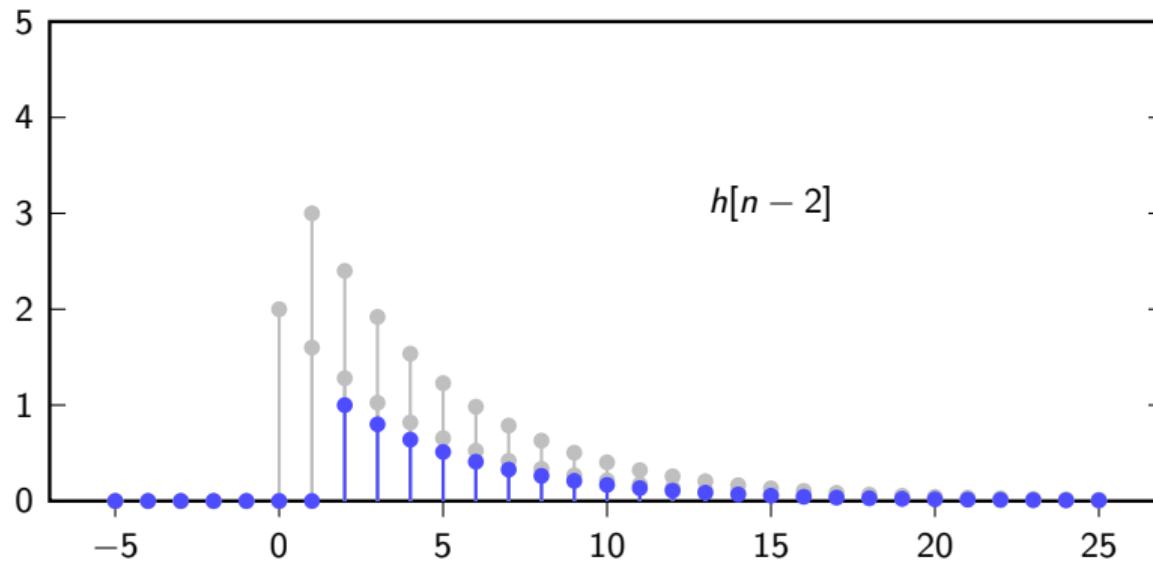
Example



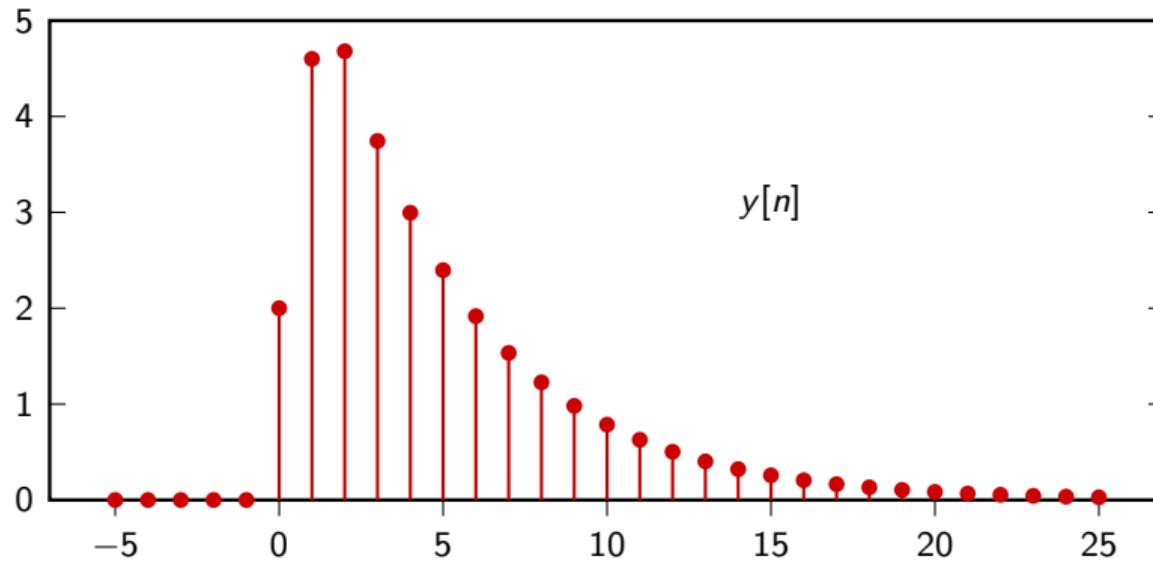
Example



Example



Example



Convolution

We can always write a canonical-base decomposition:

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n - k]$$

by linearity and time invariance:

$$\begin{aligned} y[n] &= \sum_{k=-\infty}^{\infty} x[k]h[n - k] \\ &= x[n] * h[n] \end{aligned}$$

Convolution

We can always write a canonical-base decomposition:

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n - k]$$

by linearity and time invariance:

$$\begin{aligned} y[n] &= \sum_{k=-\infty}^{\infty} x[k]h[n - k] \\ &= x[n] * h[n] \end{aligned}$$

Performing the convolution algorithmically

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

The recipe:

Ingredients:

- ▶ a sequence $x[m]$
 - ▶ a second sequence $h[m]$
- ▶ time-reverse $h[m]$
 - ▶ at each step n (from $-\infty$ to ∞):
 - center the time-reversed $h[m]$ in n
(i.e. shift by $-n$)
 - compute the inner product

Performing the convolution algorithmically

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

Ingredients:

- ▶ a sequence $x[m]$
- ▶ a second sequence $h[m]$

The recipe:

- ▶ time-reverse $h[m]$
- ▶ at each step n (from $-\infty$ to ∞):
 - center the time-reversed $h[m]$ in n
(i.e. shift by $-n$)
 - compute the inner product

Performing the convolution algorithmically

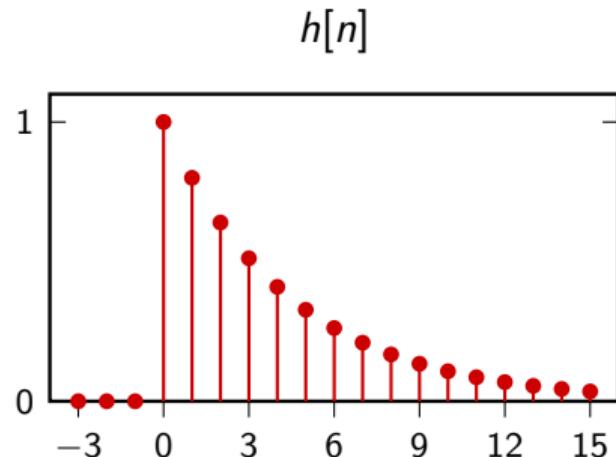
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

The recipe:

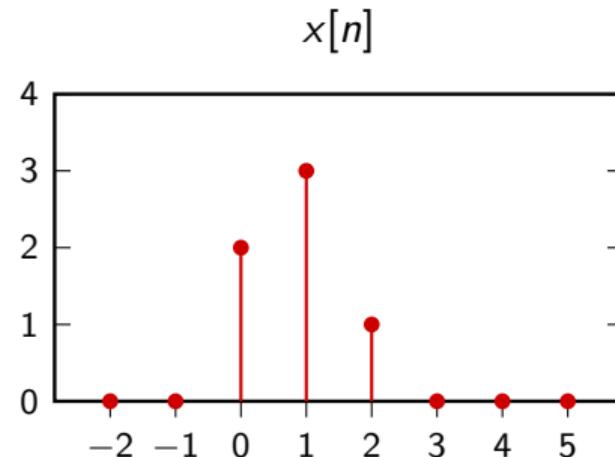
Ingredients:

- ▶ a sequence $x[m]$
 - ▶ a second sequence $h[m]$
- ▶ time-reverse $h[m]$
 - ▶ at each step n (from $-\infty$ to ∞):
 - center the time-reversed $h[m]$ in n
(i.e. shift by $-n$)
 - compute the inner product

Same example, different perspective

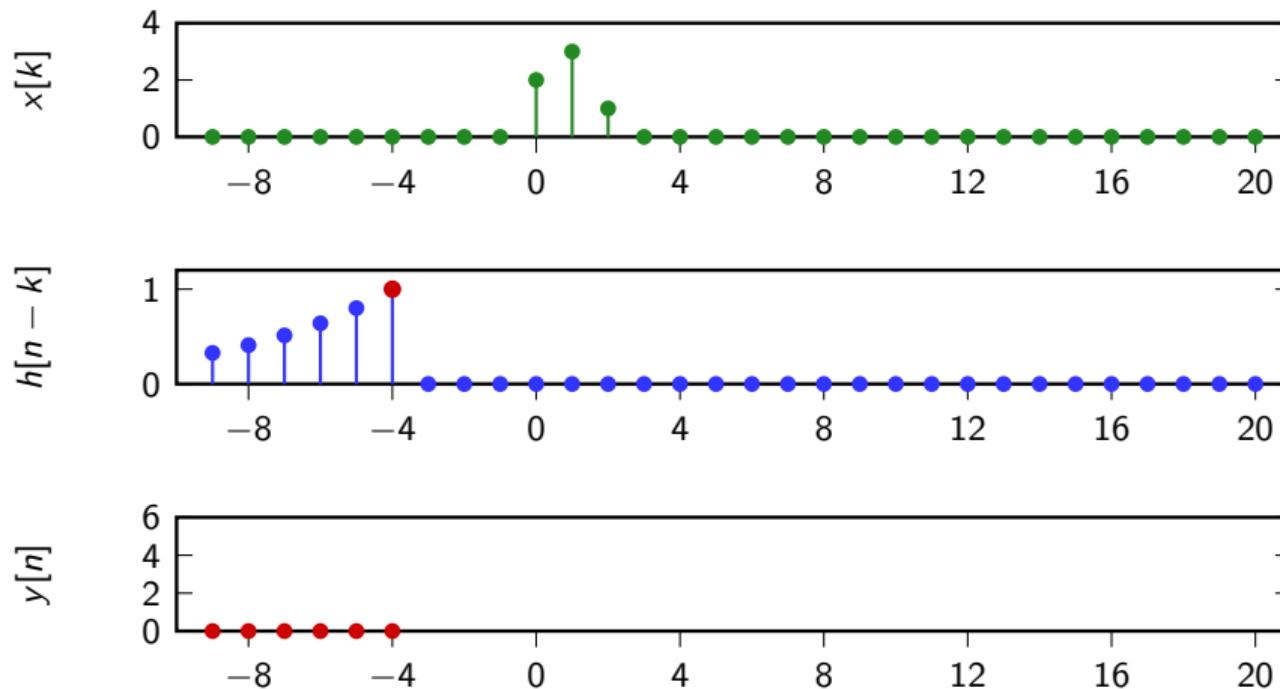


$$h[n] = \alpha^n u[n]$$

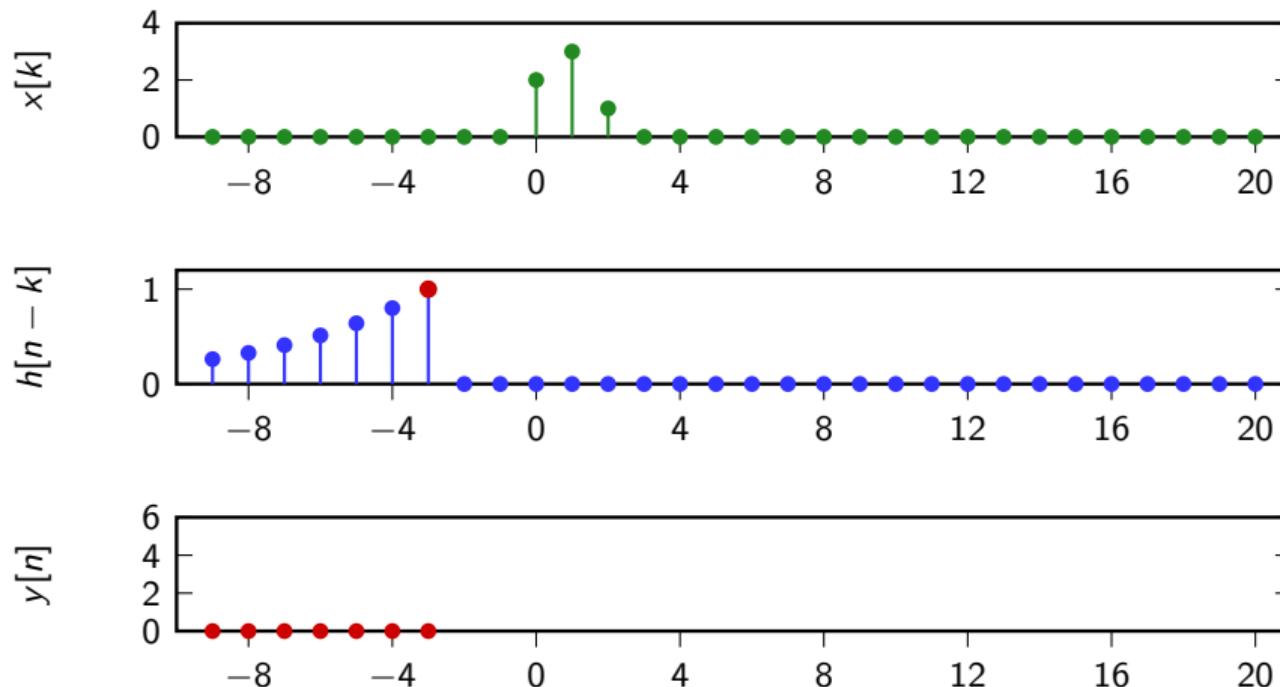


$$x[n] = \begin{cases} 2 & n = 0 \\ 3 & n = 1 \\ 1 & n = 2 \\ 0 & \text{otherwise} \end{cases}$$

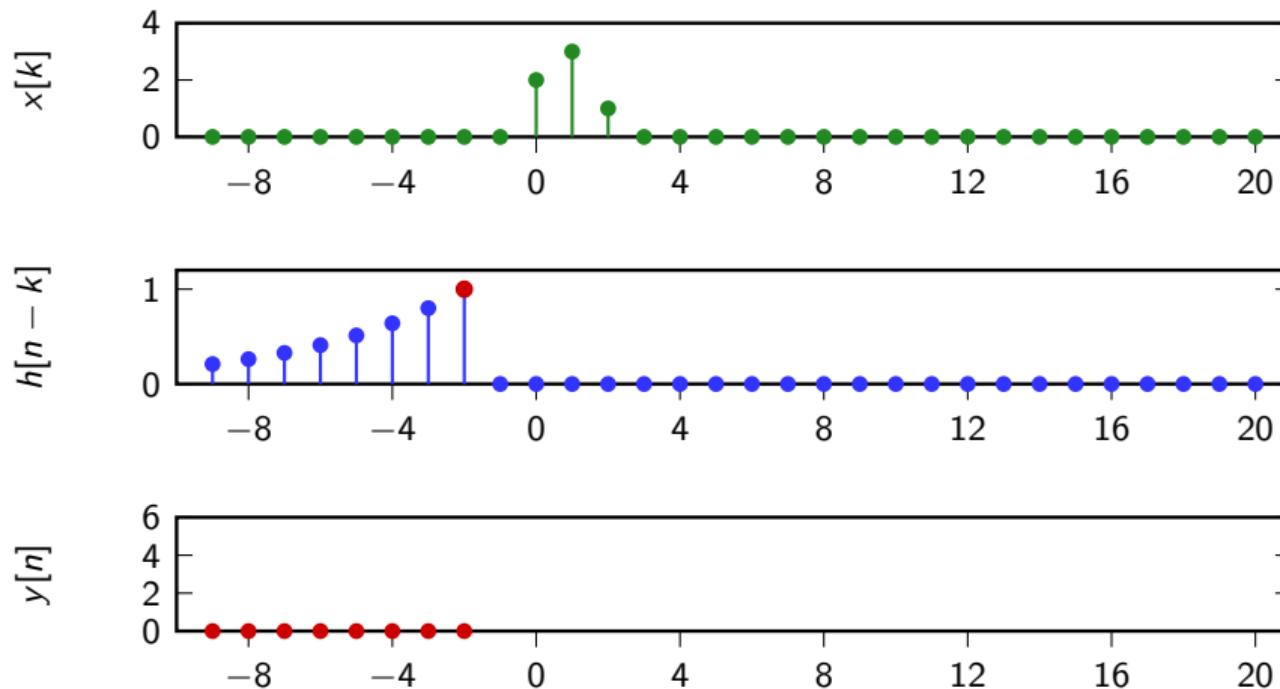
Convolution example



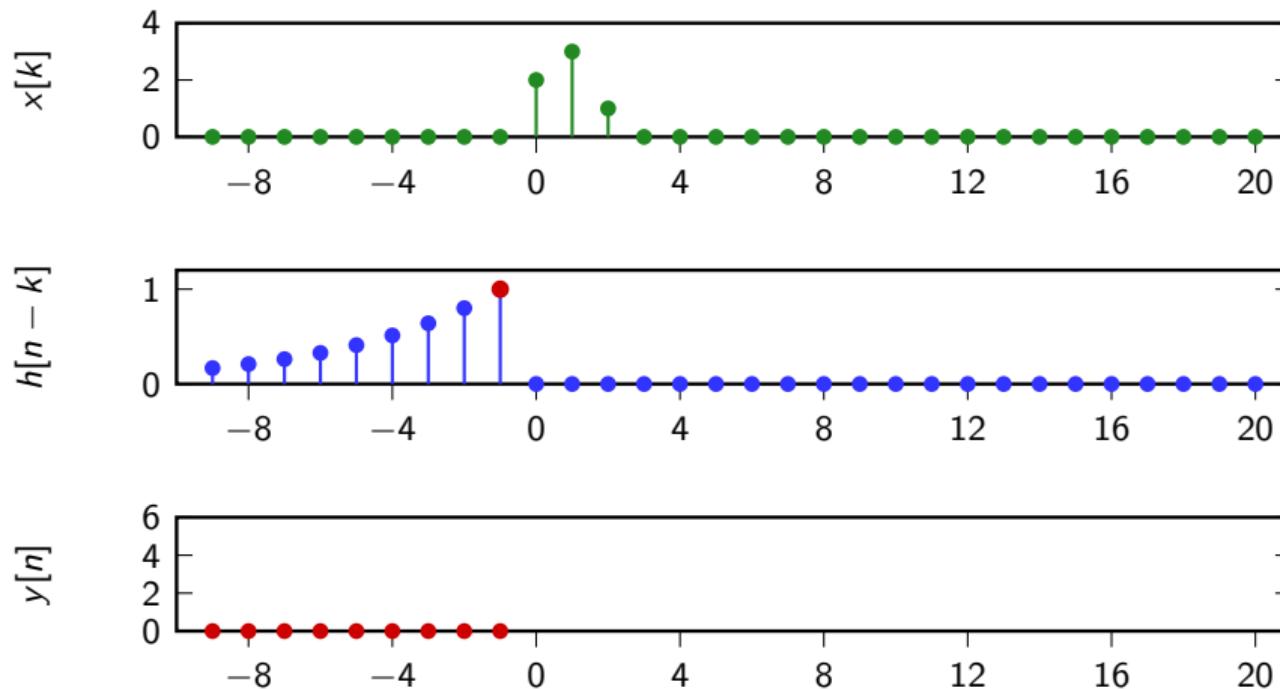
Convolution example



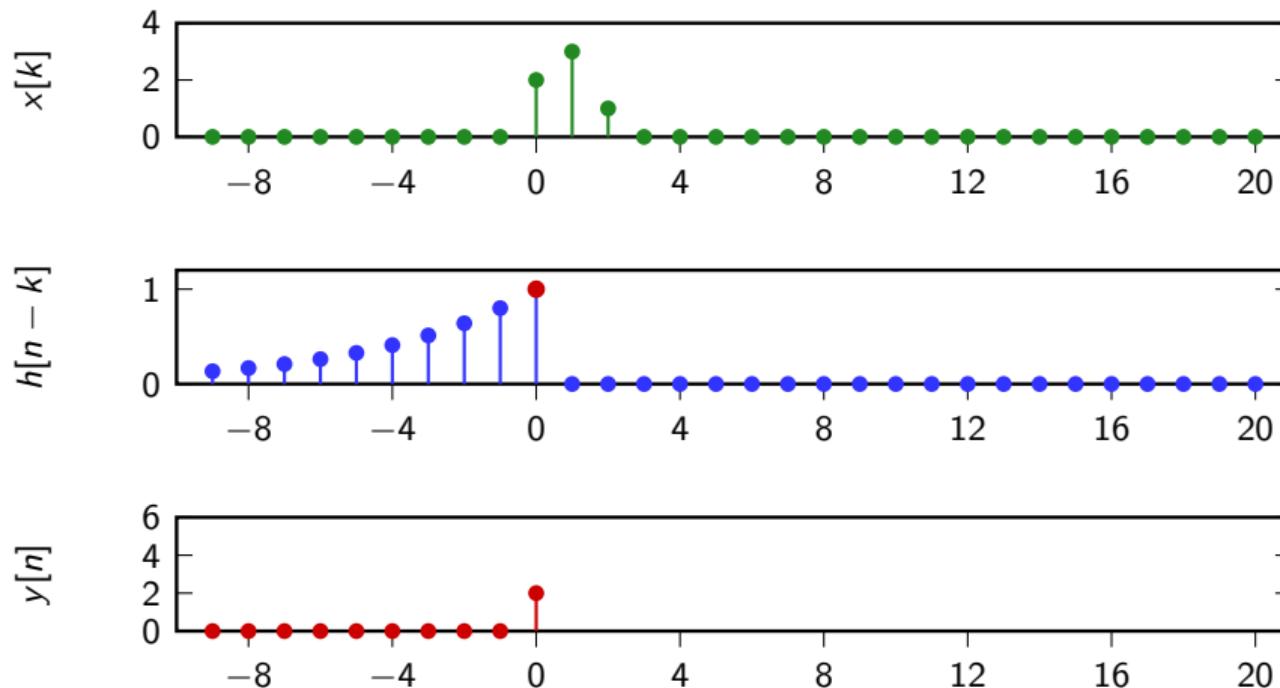
Convolution example



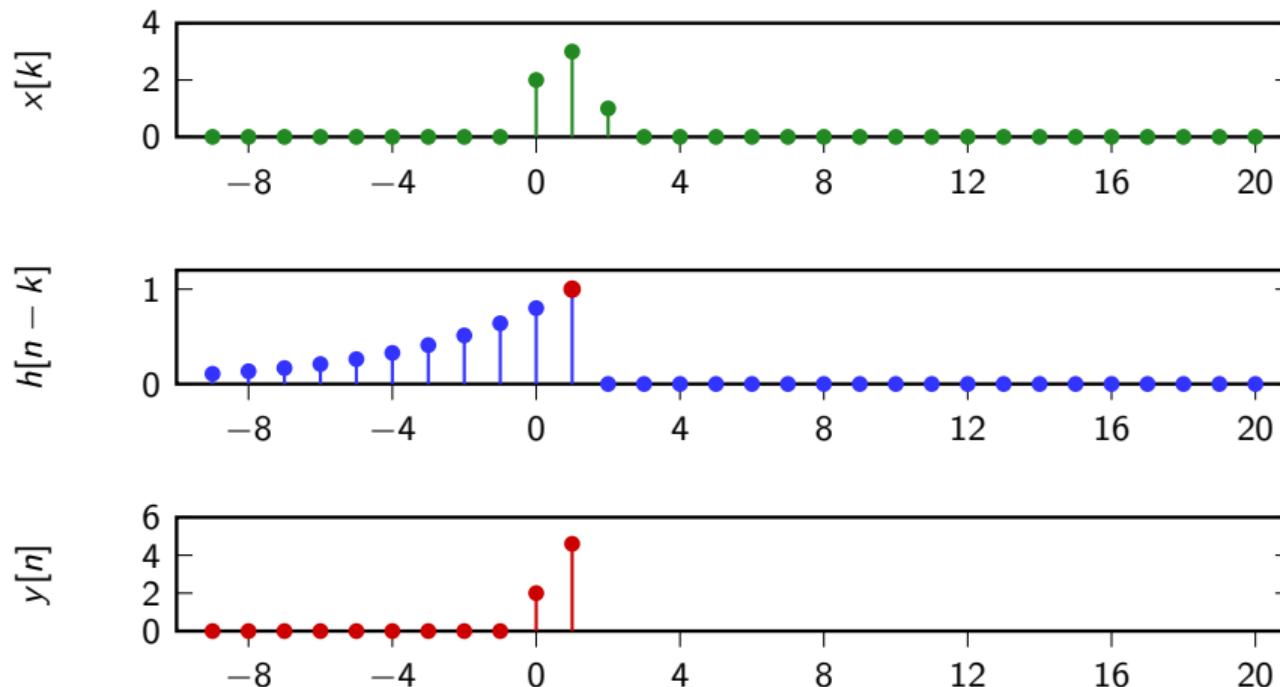
Convolution example



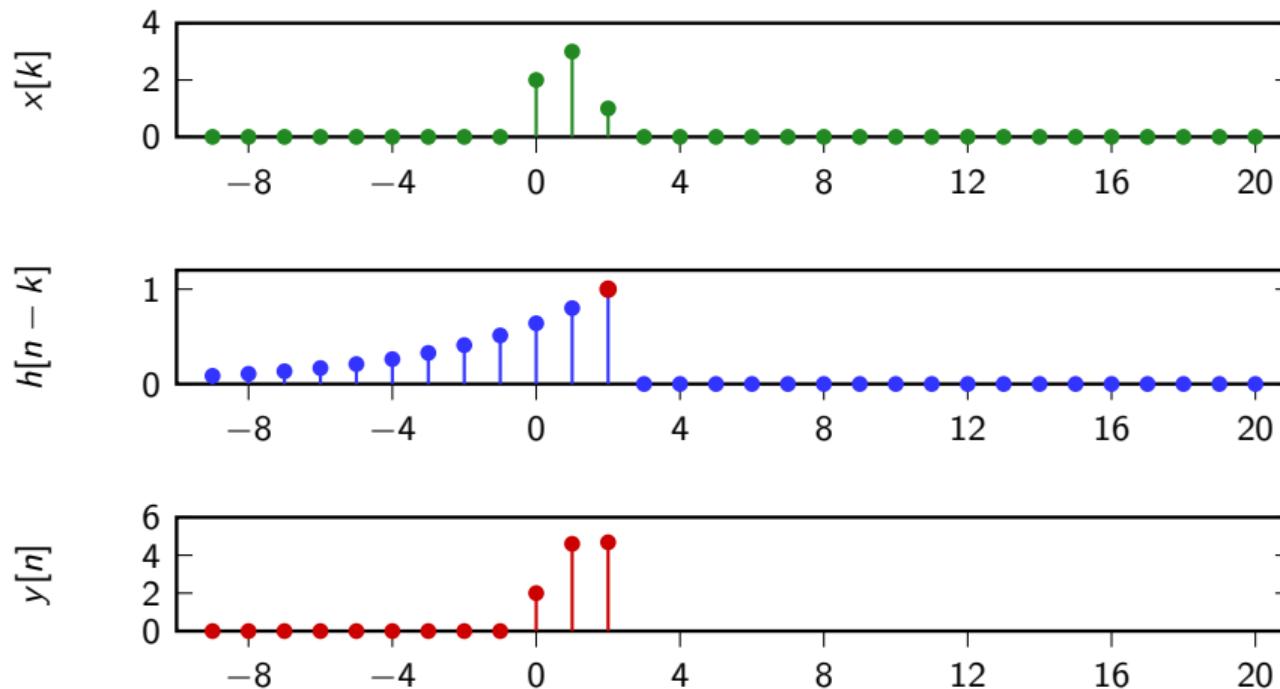
Convolution example



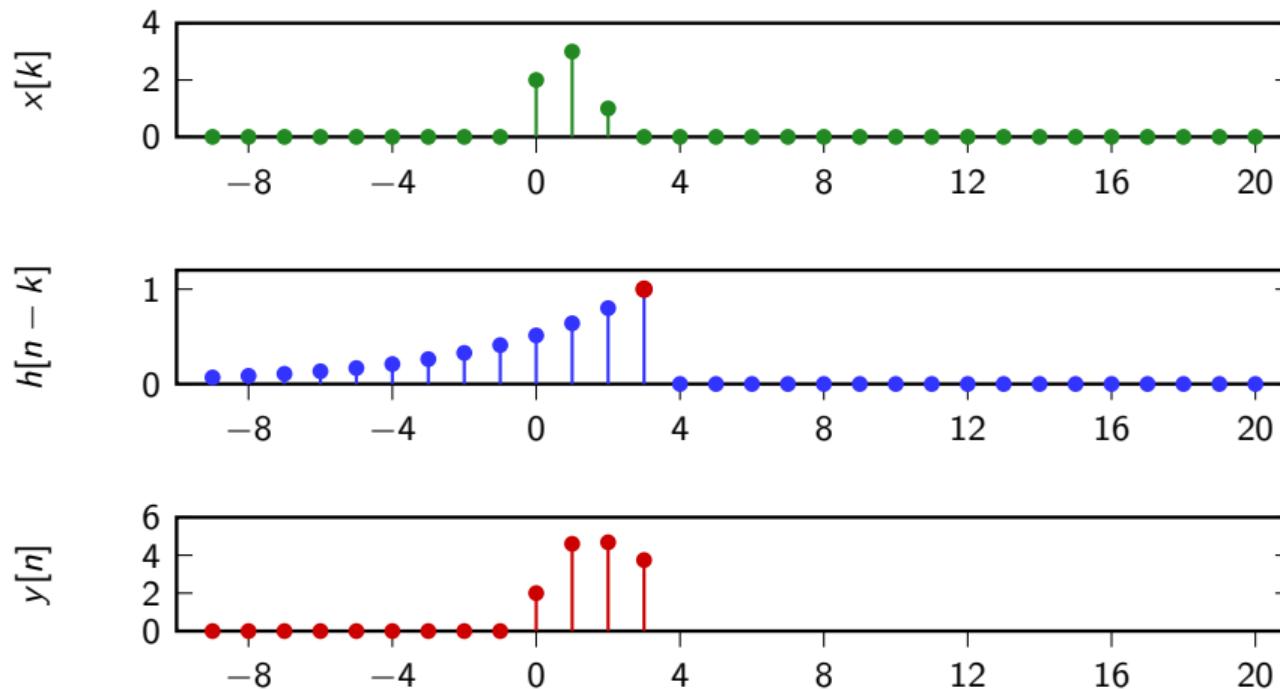
Convolution example



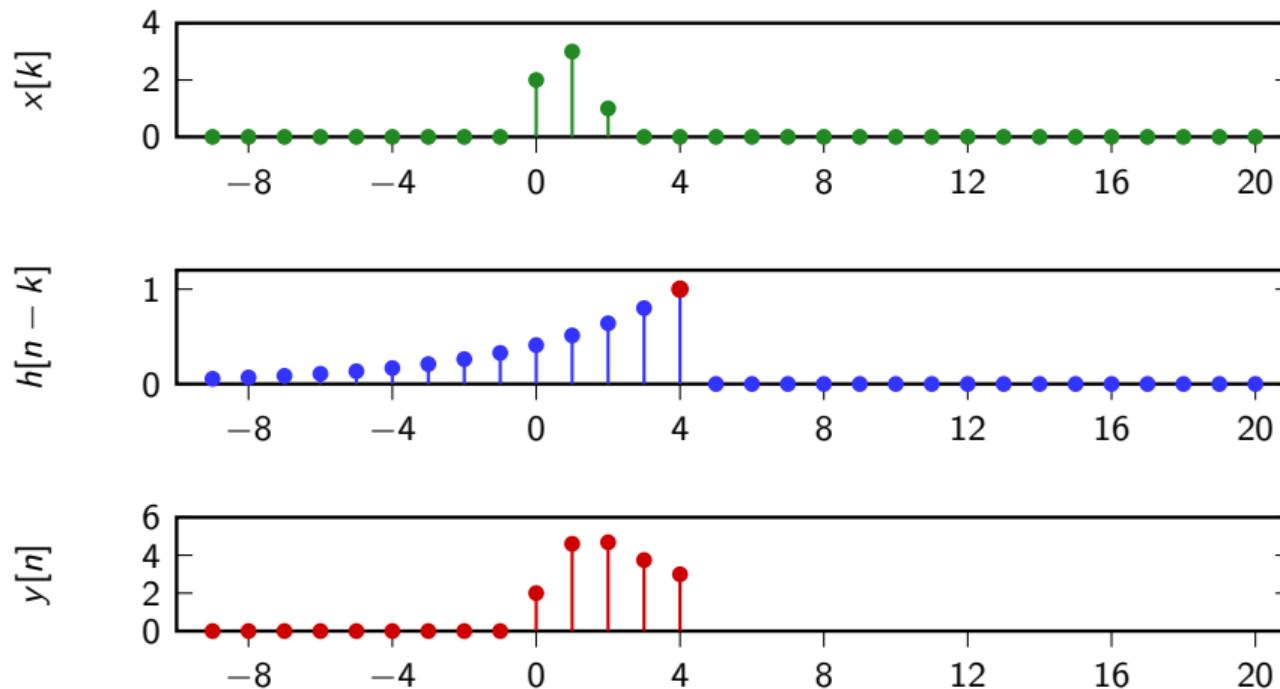
Convolution example



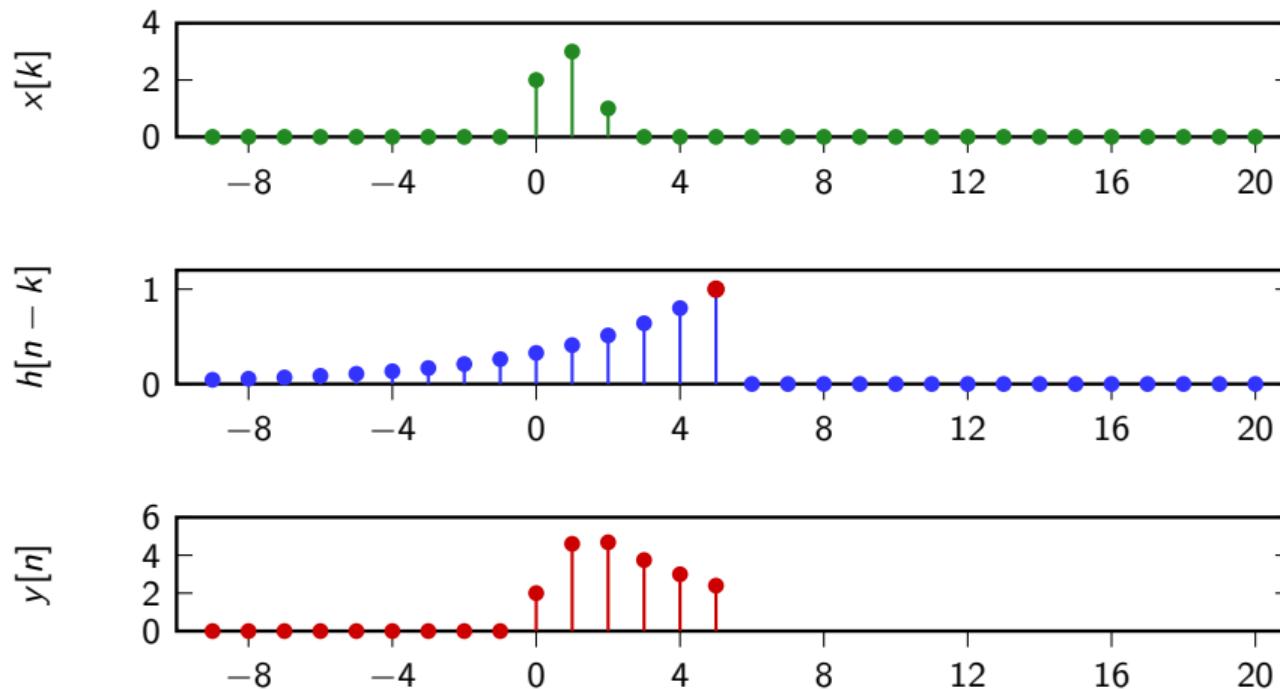
Convolution example



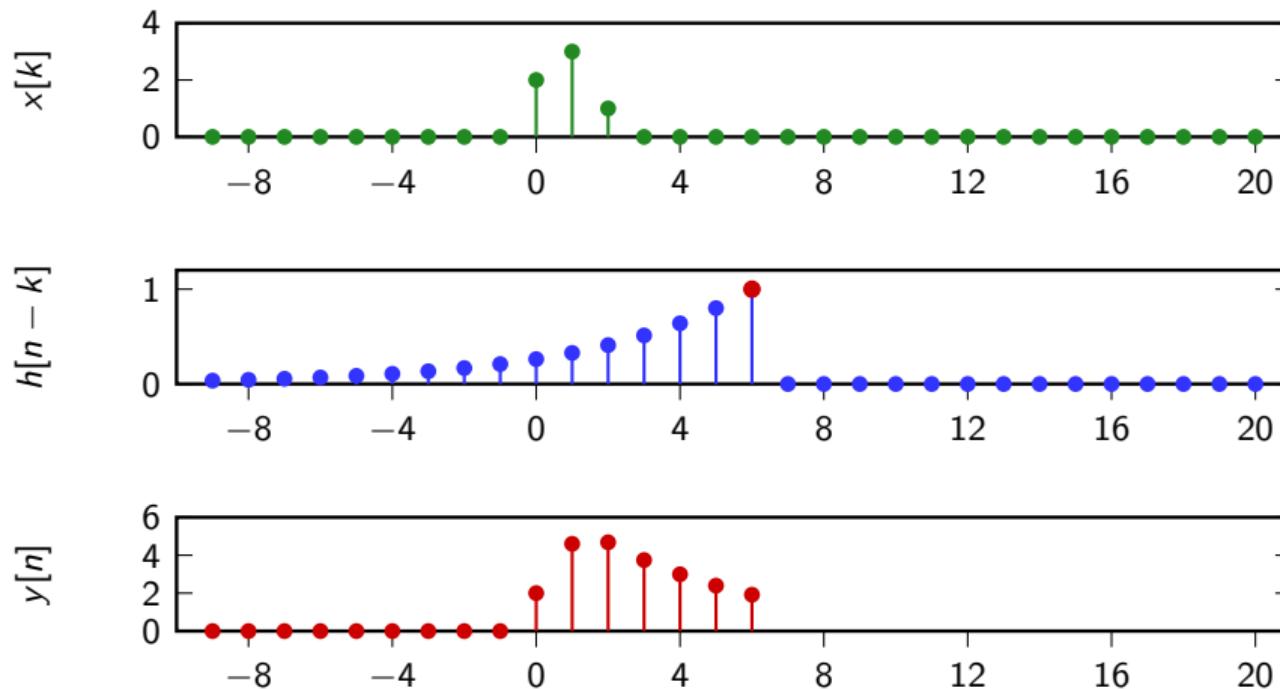
Convolution example



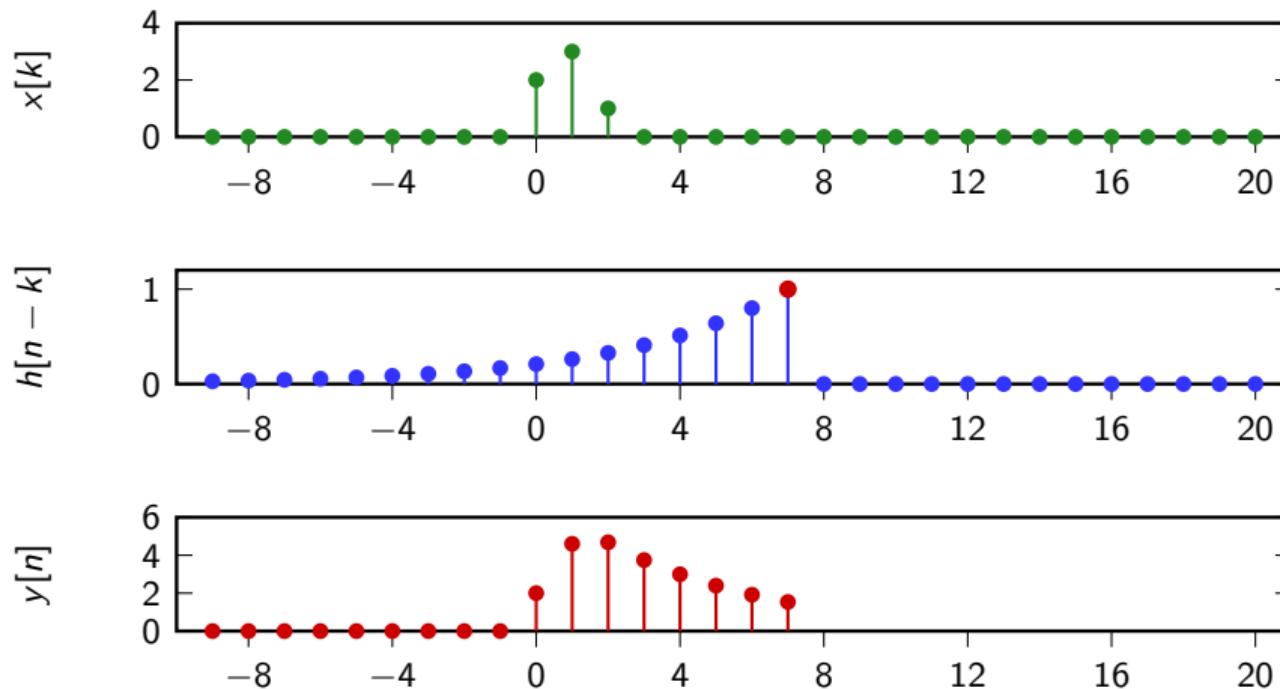
Convolution example



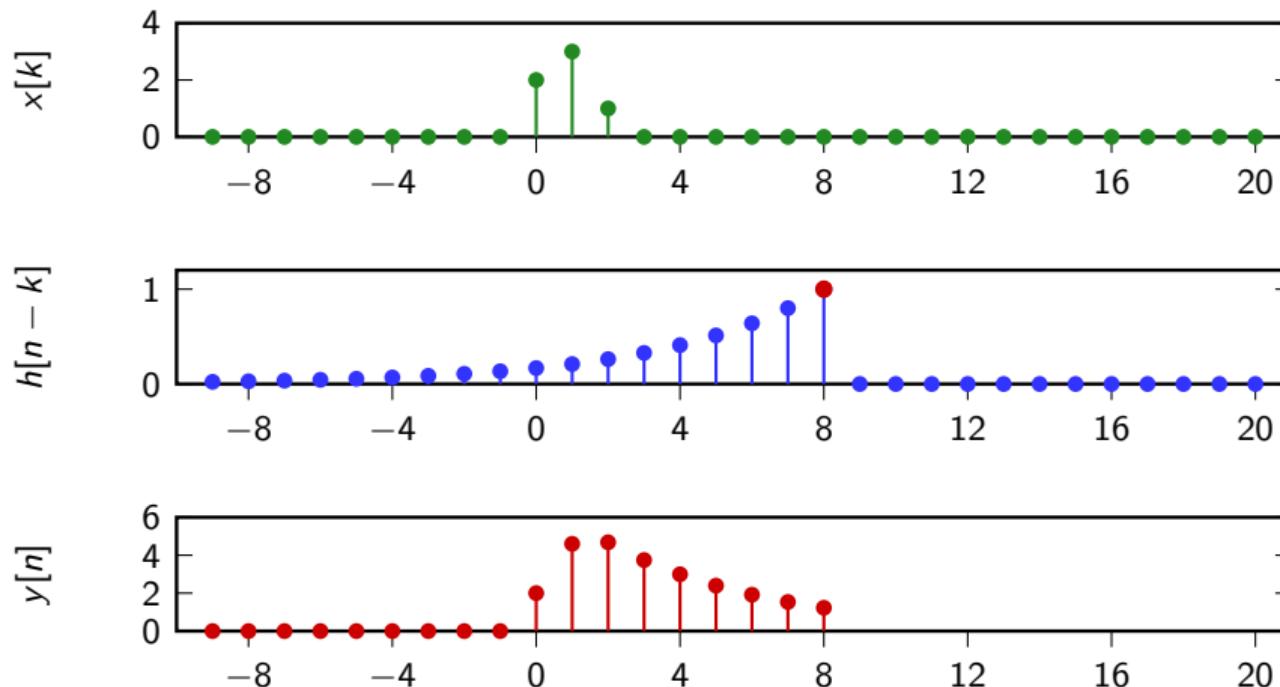
Convolution example



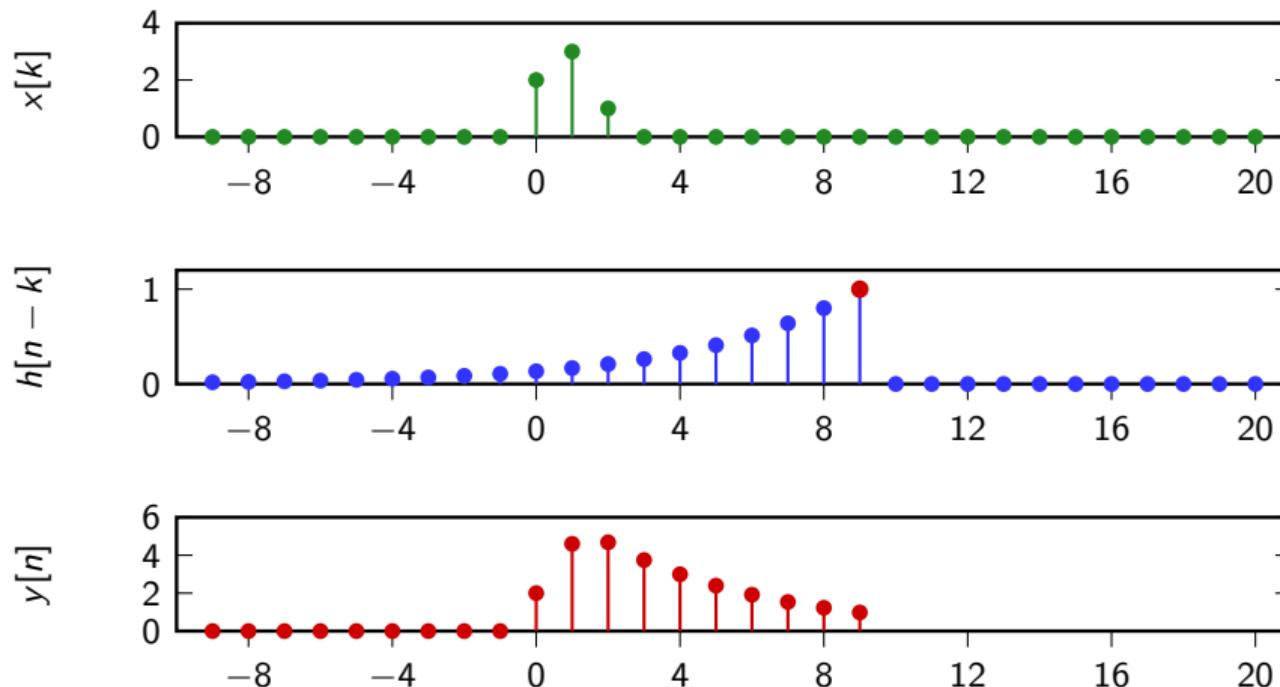
Convolution example



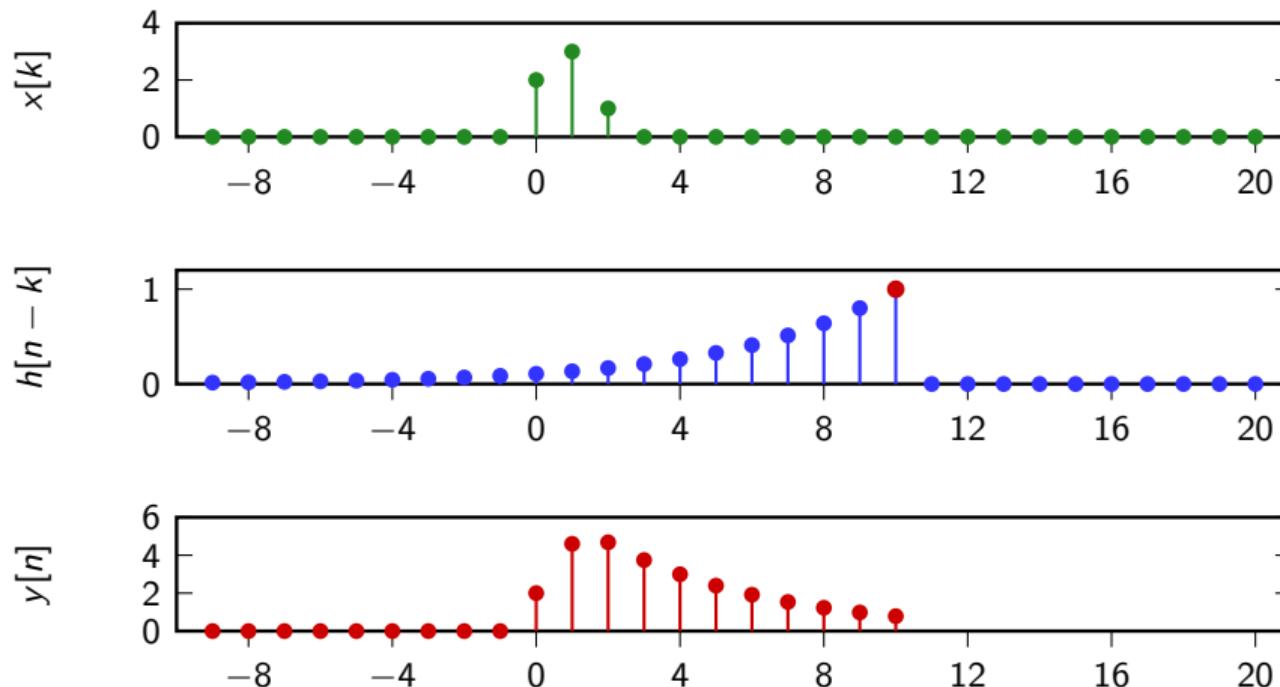
Convolution example



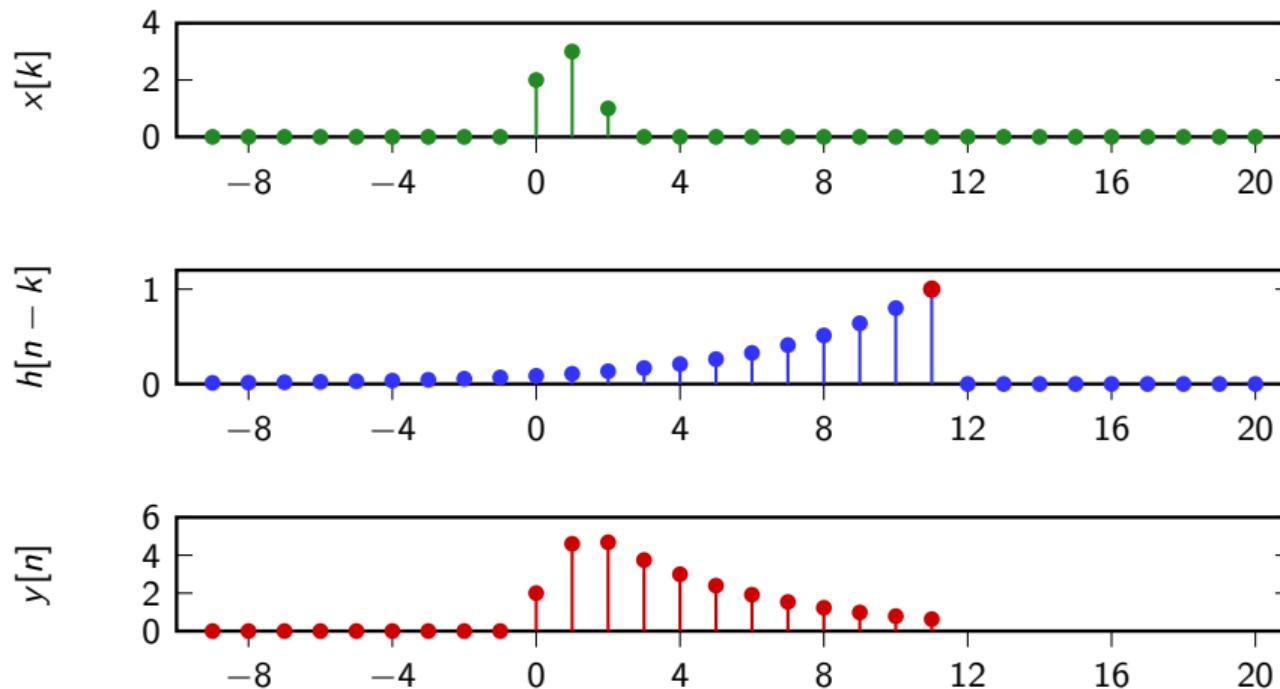
Convolution example



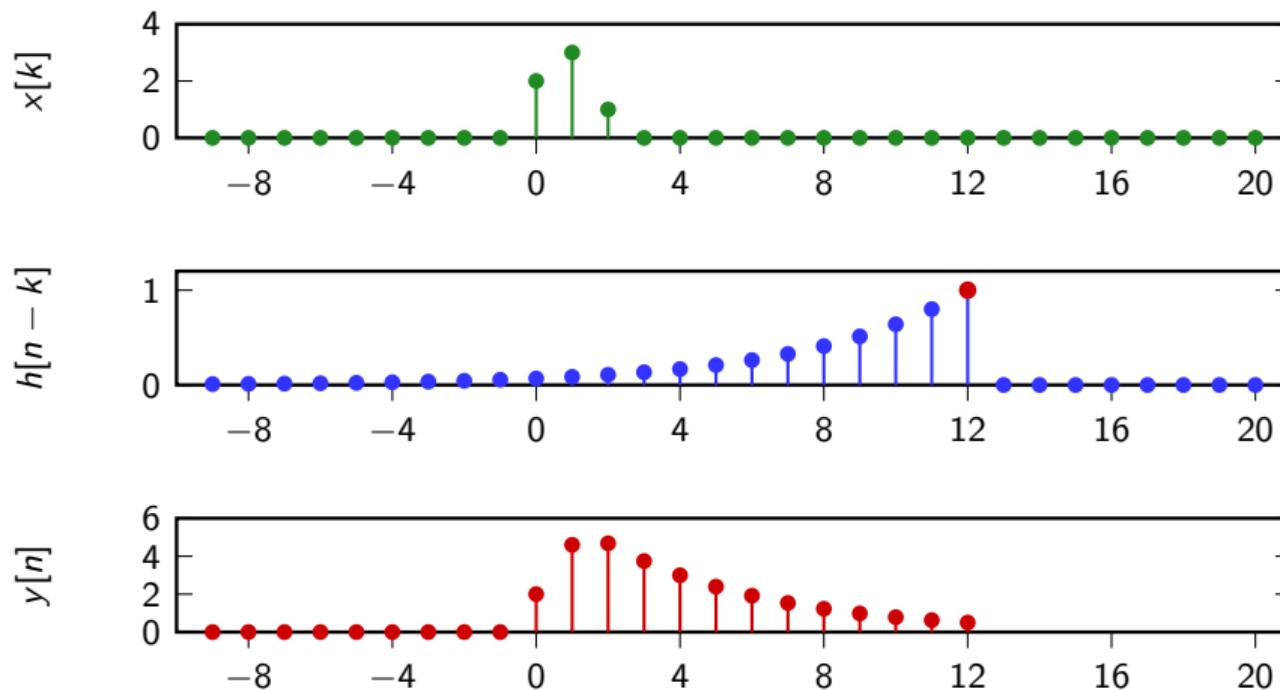
Convolution example



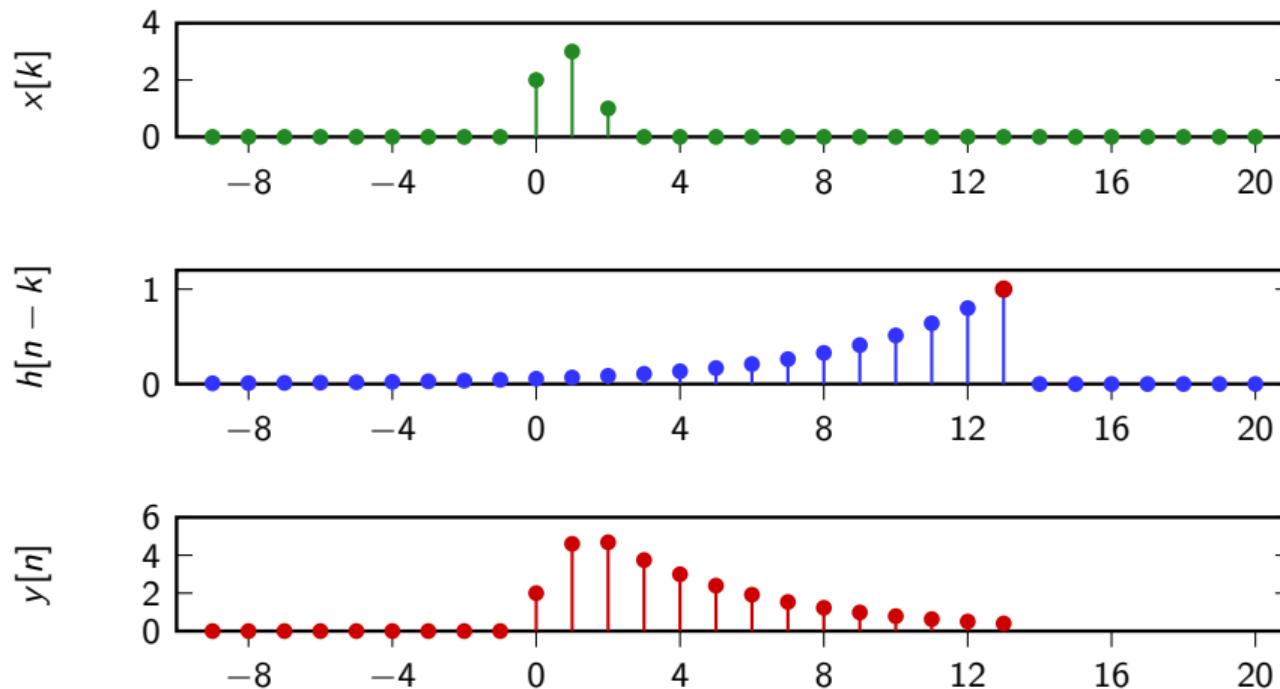
Convolution example



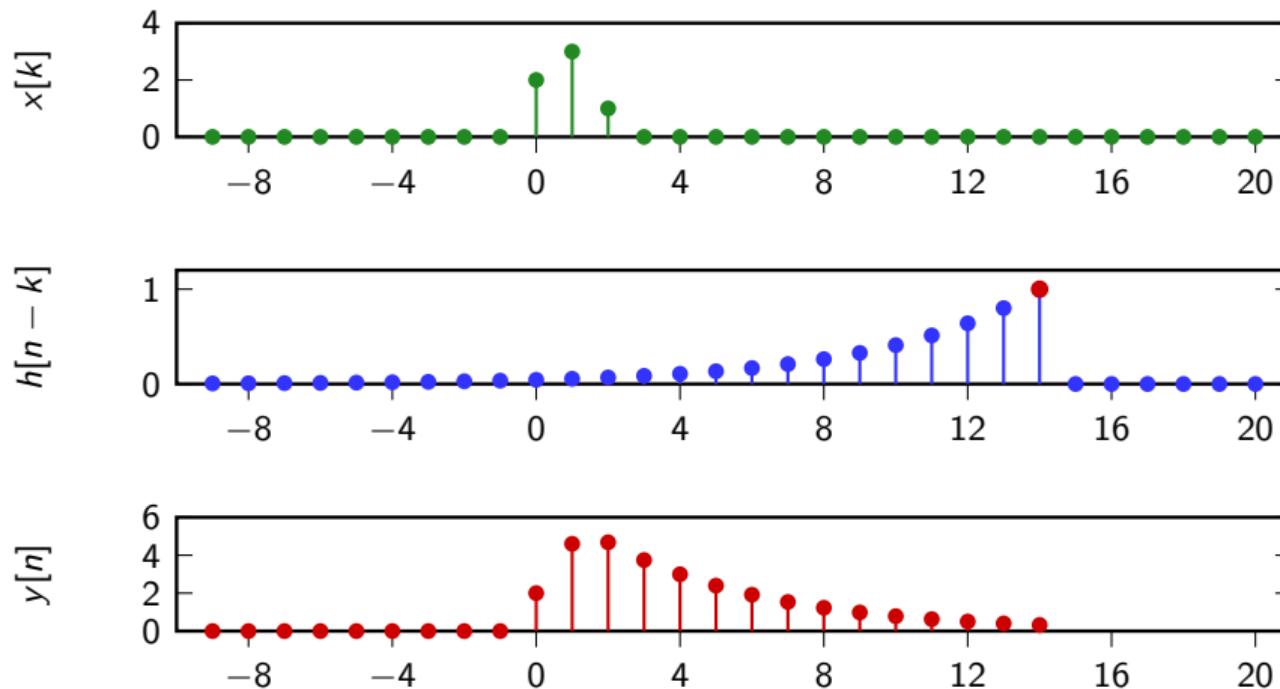
Convolution example



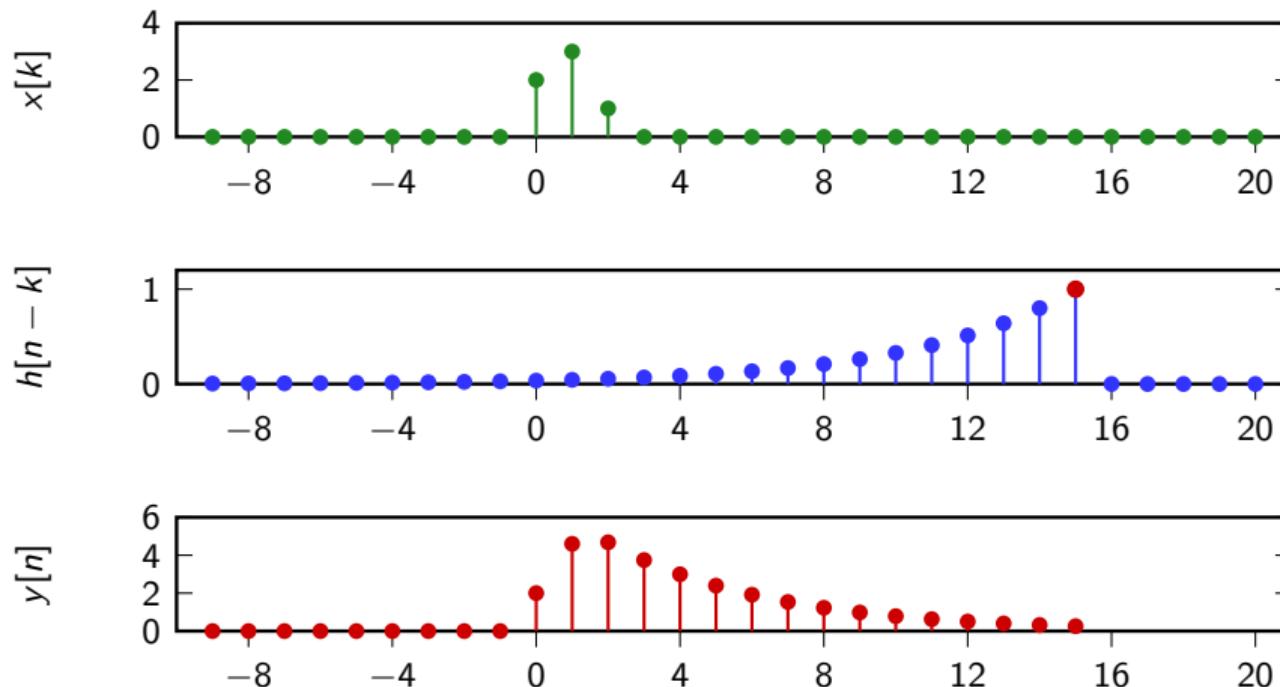
Convolution example



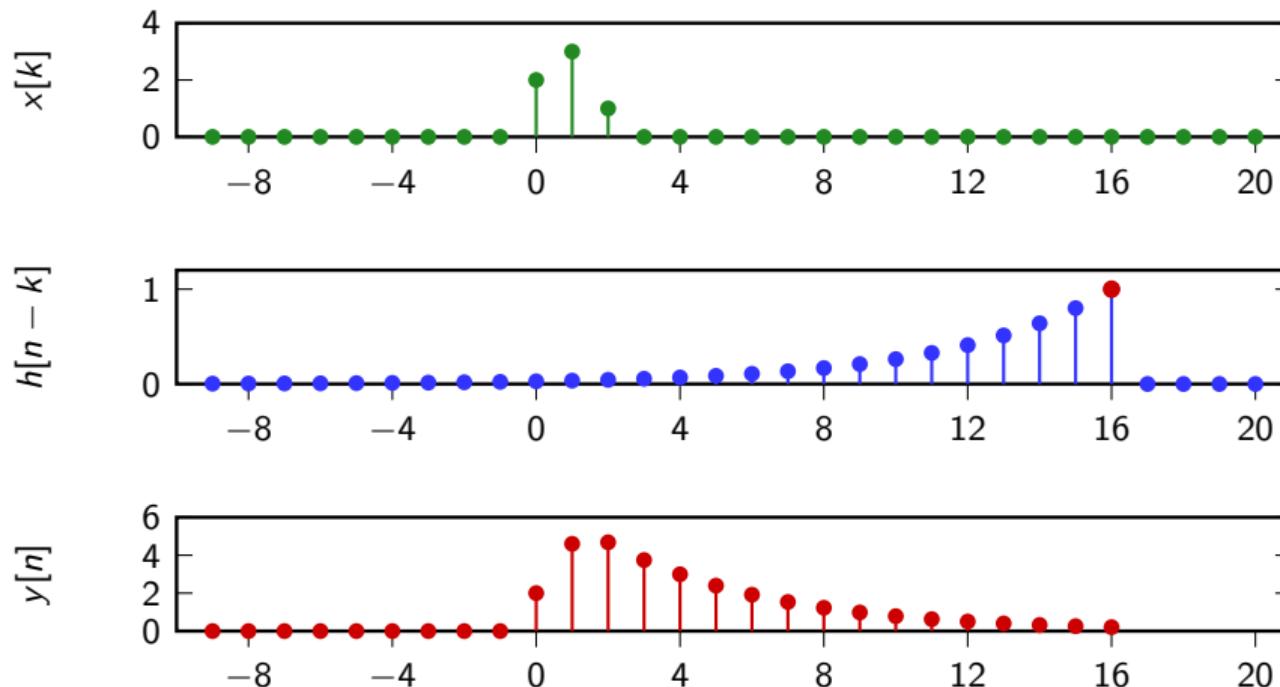
Convolution example



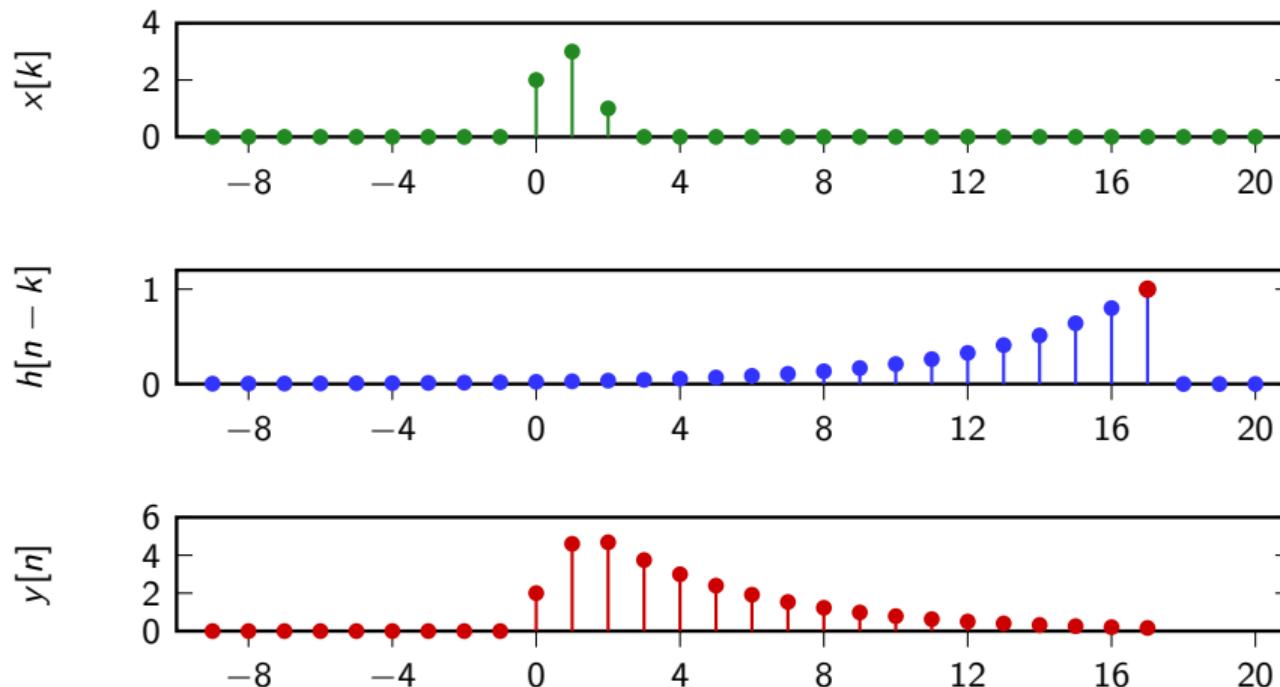
Convolution example



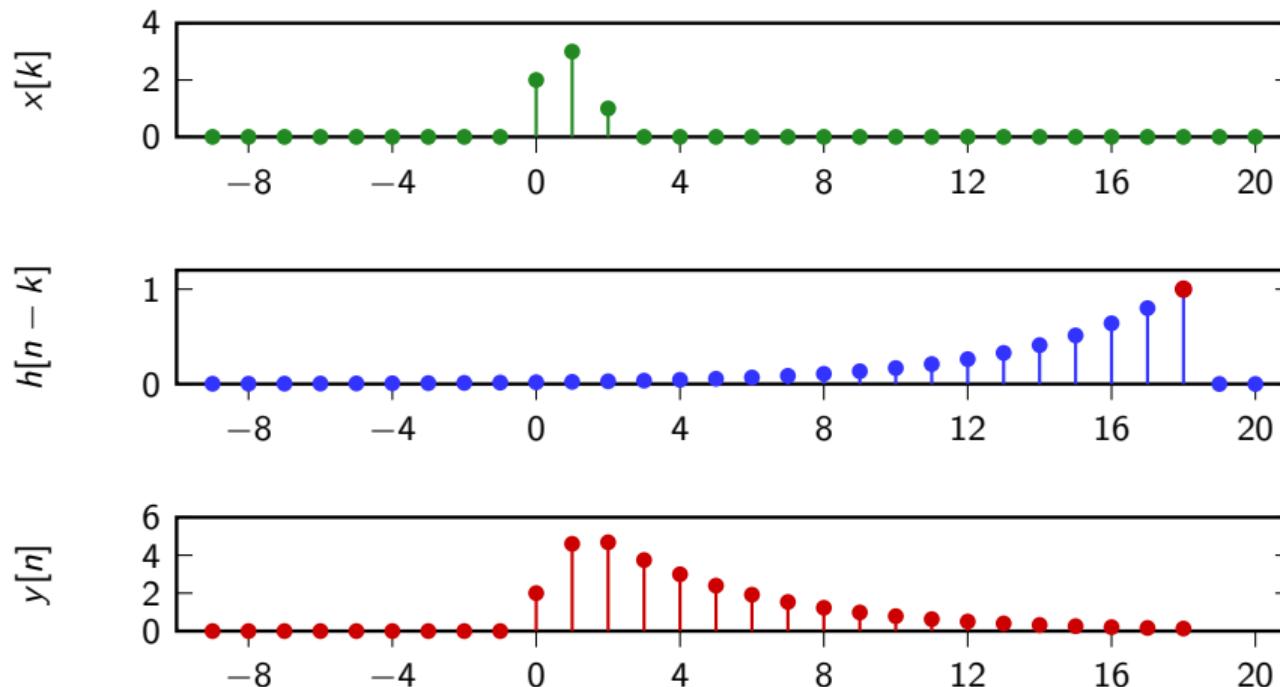
Convolution example



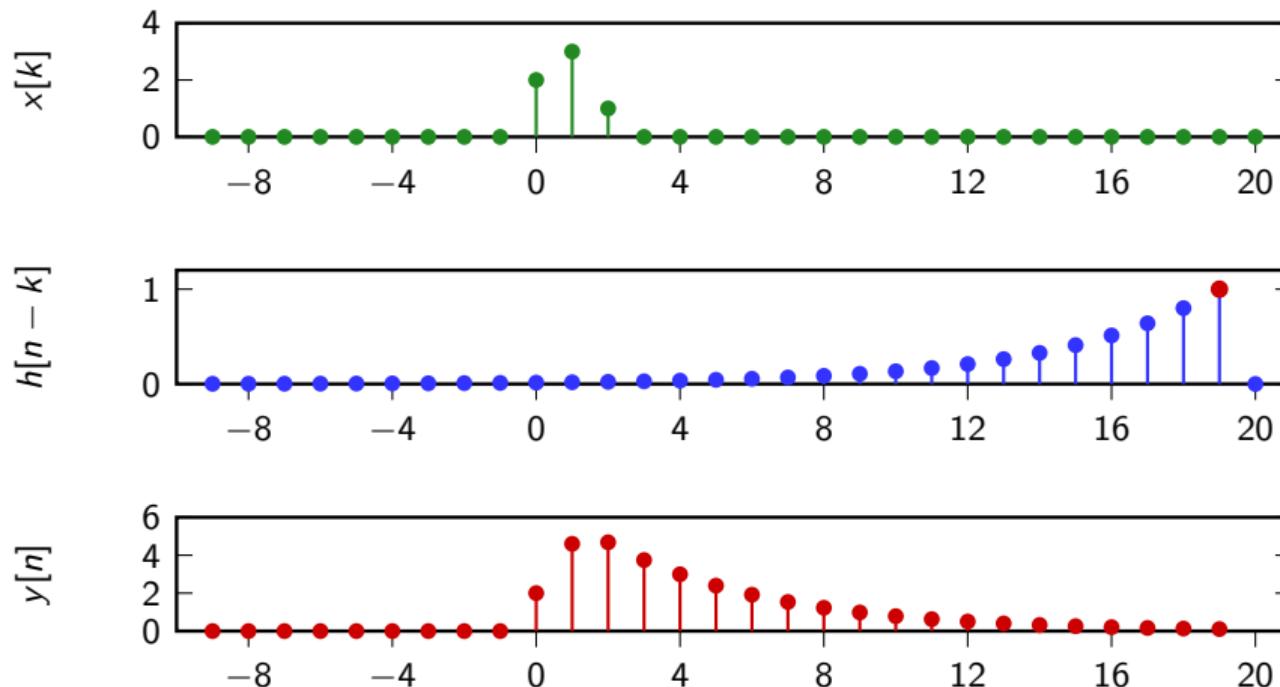
Convolution example



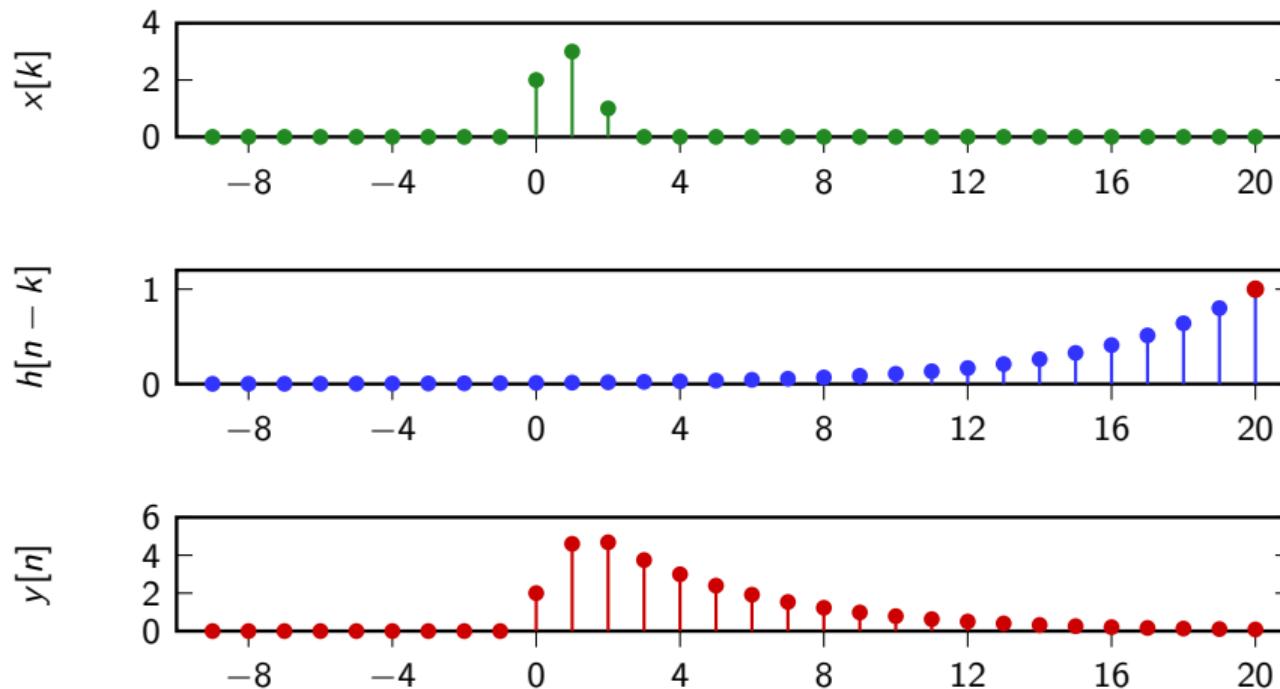
Convolution example



Convolution example



Convolution example



Convolution properties

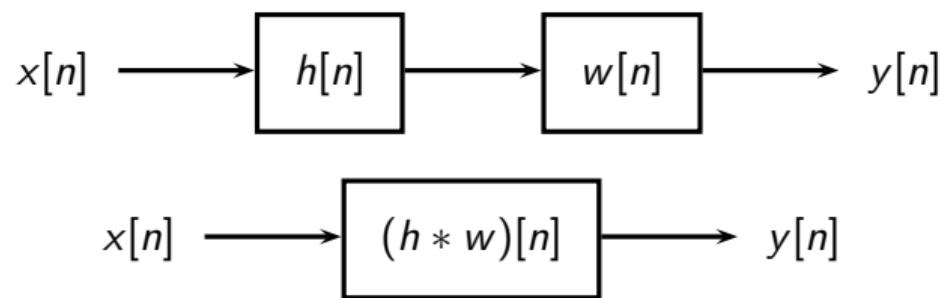
- ▶ linearity and time invariance (by definition)
- ▶ commutativity: $(x * h)[n] = (h * x)[n]$
- ▶ associativity for absolutely- and square-summable sequences:
 $((x * h) * w)[n] = (x * (h * w))[n]$

Convolution properties

- ▶ linearity and time invariance (by definition)
- ▶ commutativity: $(x * h)[n] = (h * x)[n]$
- ▶ associativity for absolutely- and square-summable sequences:
 $((x * h) * w)[n] = (x * (h * w))[n]$

Convolution properties

- ▶ linearity and time invariance (by definition)
- ▶ commutativity: $(x * h)[n] = (h * x)[n]$
- ▶ associativity for absolutely- and square-summable sequences:
 $((x * h) * w)[n] = (x * (h * w))[n]$



filtering by example

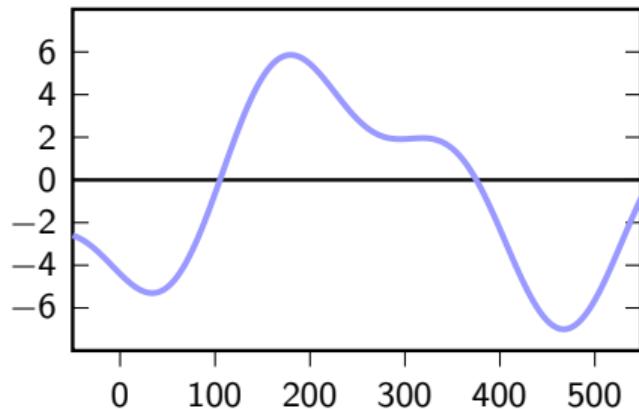
Overview:

- ▶ Moving average filter
- ▶ Leaky integrator

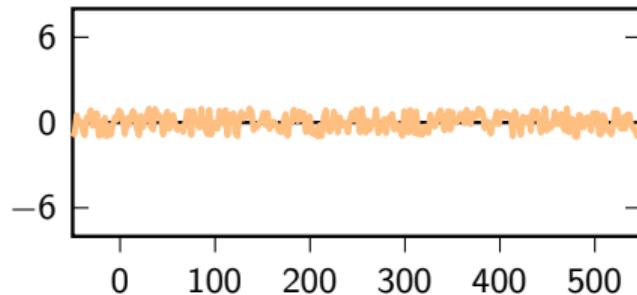
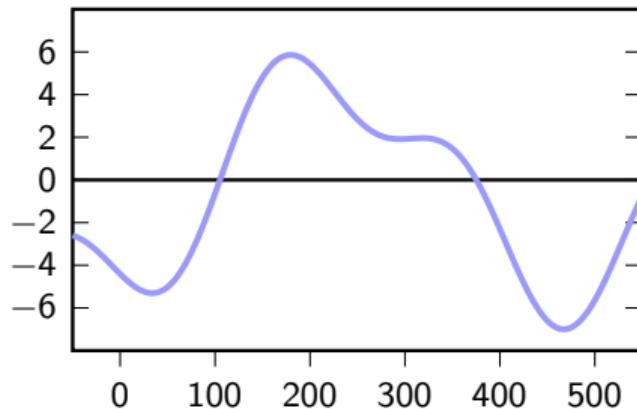
Overview:

- ▶ Moving average filter
- ▶ Leaky integrator

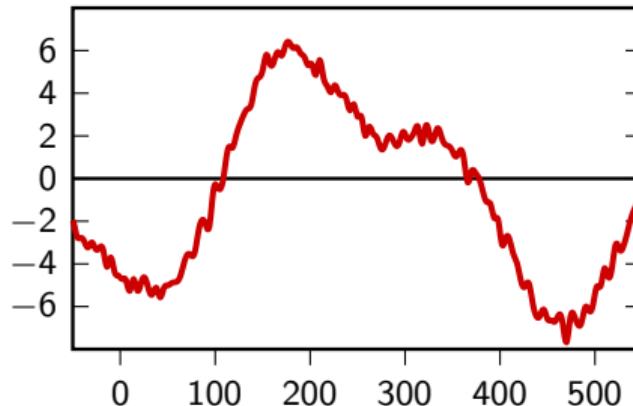
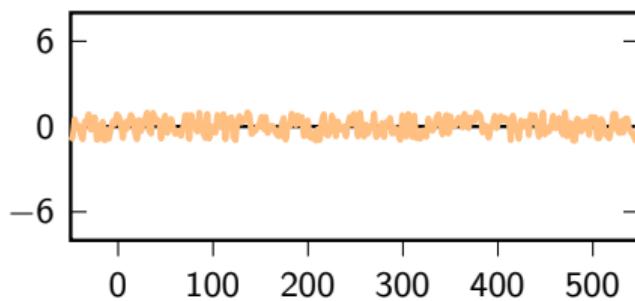
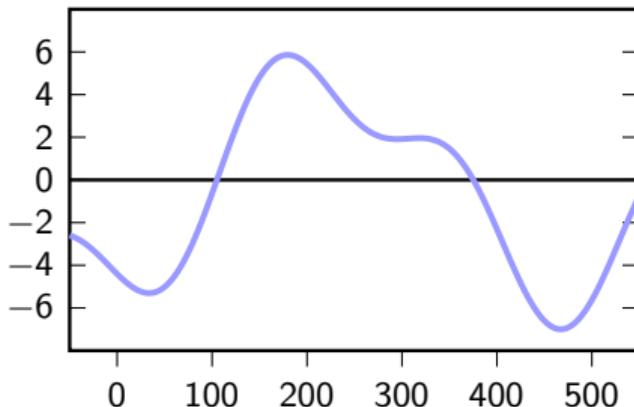
Typical filtering scenario: denoising



Typical filtering scenario: denoising



Typical filtering scenario: denoising



Denoising by Moving Average

- ▶ idea: replace each sample by the local average
- ▶ for instance: $y[n] = (x[n] + x[n - 1])/2$
- ▶ more generally:

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n - k]$$

Denoising by Moving Average

- ▶ idea: replace each sample by the local average
- ▶ for instance: $y[n] = (x[n] + x[n - 1])/2$
- ▶ more generally:

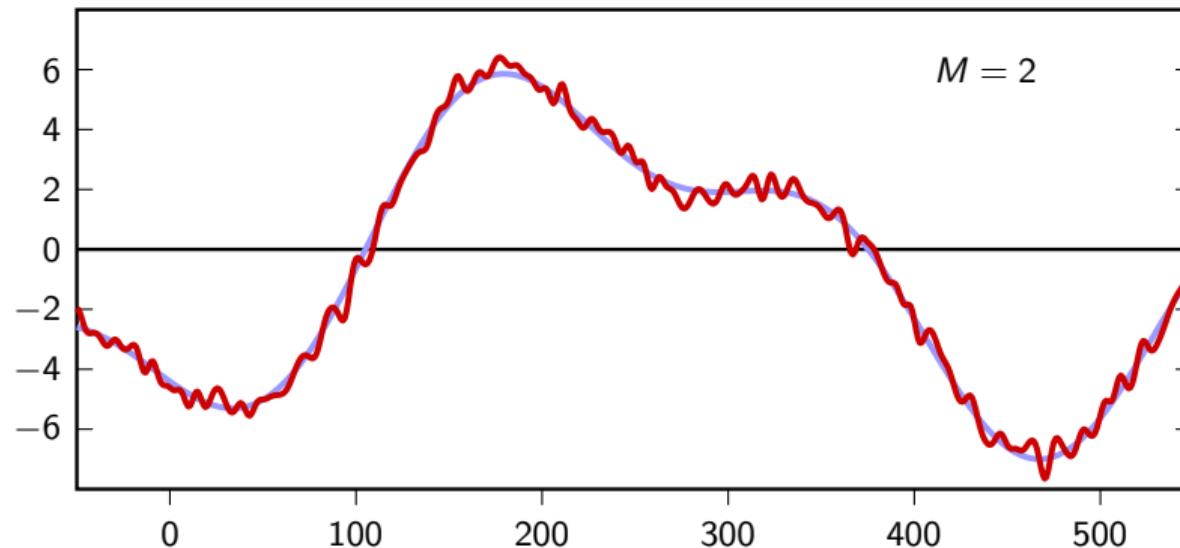
$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n - k]$$

Denoising by Moving Average

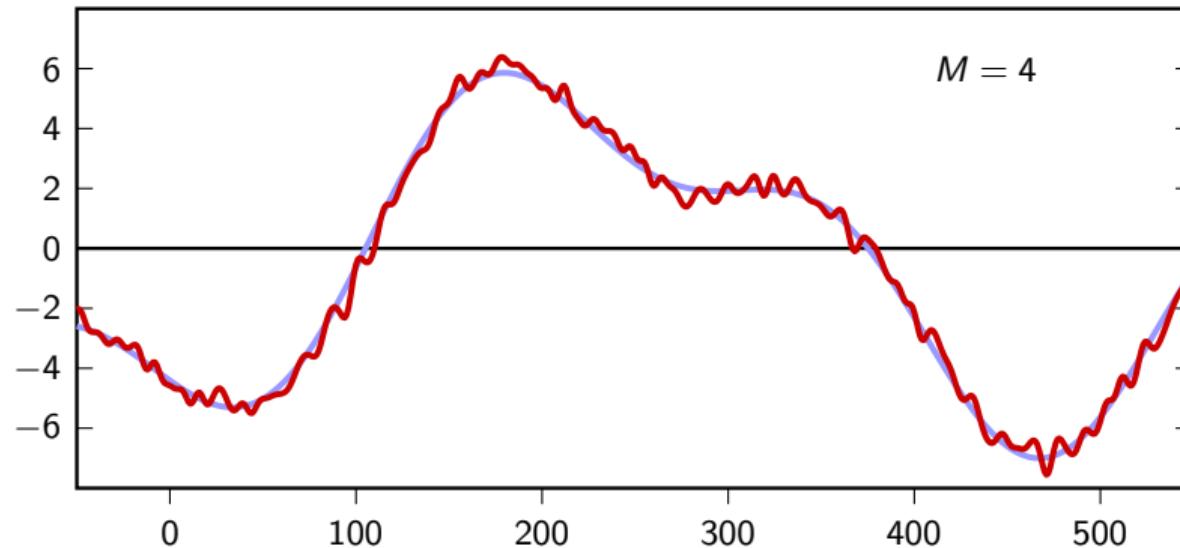
- ▶ idea: replace each sample by the local average
- ▶ for instance: $y[n] = (x[n] + x[n - 1])/2$
- ▶ more generally:

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n - k]$$

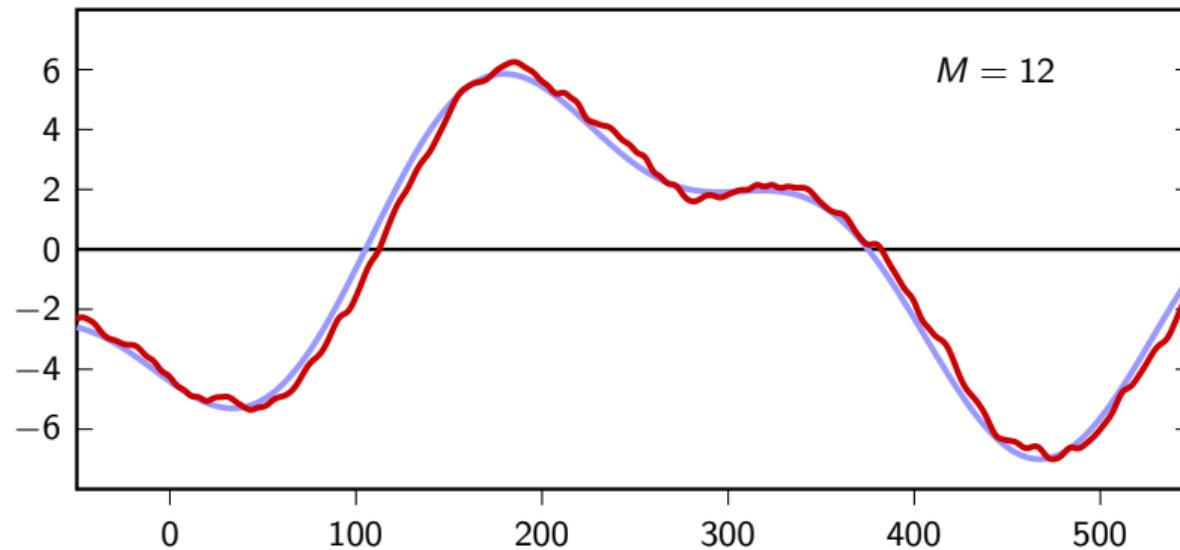
Denoising by Moving Average



Denoising by Moving Average

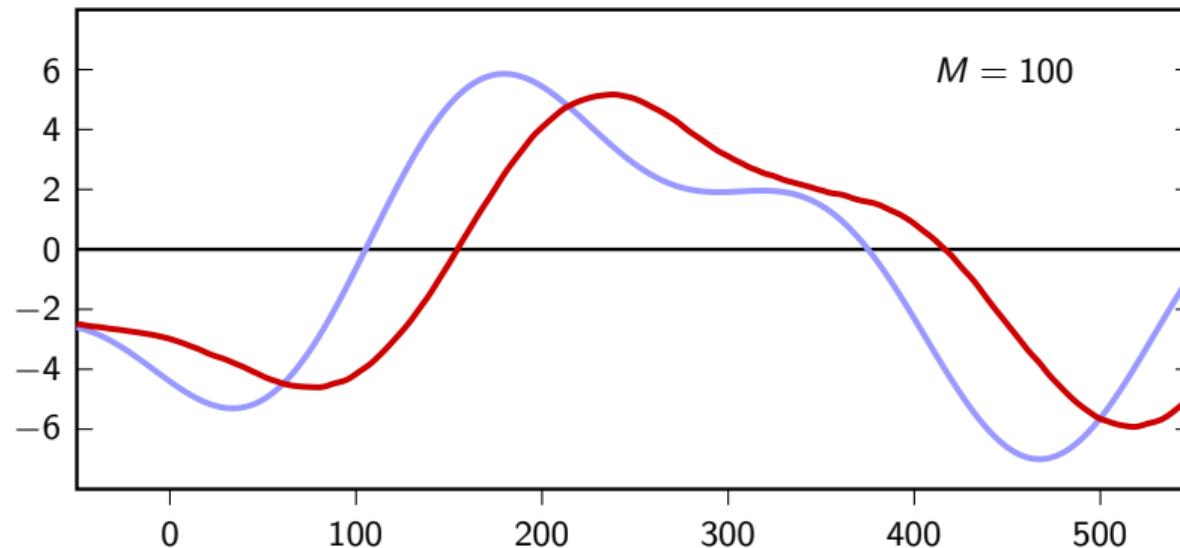


Denoising by Moving Average



$M = 12$

Denoising by Moving Average



MA: impulse response

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n - k]$$

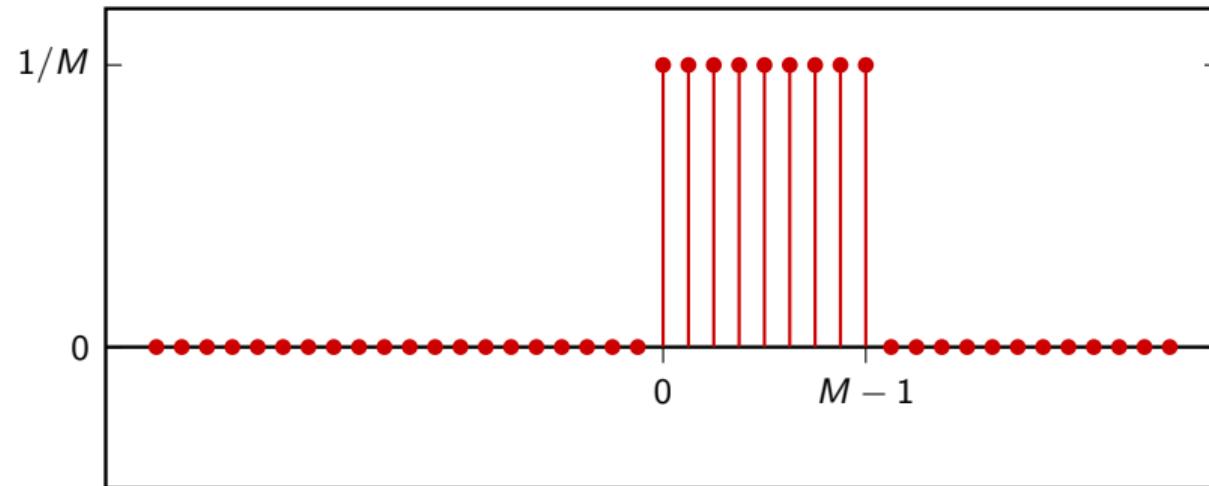
MA: impulse response

$$h[n] = \frac{1}{M} \sum_{k=0}^{M-1} \delta[n - k]$$

MA: impulse response

$$h[n] = \frac{1}{M} \sum_{k=0}^{M-1} \delta[n - k]$$
$$= \begin{cases} 1/M & \text{for } 0 \leq n < M \\ 0 & \text{otherwise} \end{cases}$$

MA: impulse response



MA: analysis

- ▶ smoothing effect proportional to M
- ▶ number of operations and storage also proportional to M

From the MA to a first-order recursion

$$y_M[n] = \frac{1}{M} (x[n] + x[n - 1] + x[n - 2] + \dots + x[n - M + 1])$$

moving average over M points



From the MA to a first-order recursion

$$y_M[n] = \frac{1}{M} (x[n] + x[n-1] + x[n-2] + \dots + x[n-M+1])$$

moving average over M points

$$y_{M-1}[n] = \frac{1}{M-1} (x[n] + x[n-1] + x[n-2] + \dots + x[n-M+2])$$

From the MA to a first-order recursion

$$y_M[n] = \frac{1}{M}x[n] + \frac{1}{M} (x[n-1] + x[n-2] + \dots + x[n-M+1])$$

From the MA to a first-order recursion

$$y_M[n] = \frac{1}{M}x[n] + \frac{1}{M}(x[n-1] + x[n-2] + \dots + x[n-M+1])$$

“almost” $y_{M-1}[n-1]$

i.e., moving average over $M - 1$ points, delayed by one

From the MA to a first-order recursion

$$y_M[n] = \frac{1}{M}x[n] + \left[\frac{1}{M} (x[n-1] + x[n-2] + \dots + x[n-M+1]) \right]$$

$$y_M[n] = \frac{1}{M}x[n] + \frac{M-1}{M}y_{M-1}[n-1]$$

From the MA to a first-order recursion

$$y_M[n] = \frac{M-1}{M}y_{M-1}[n-1] + \frac{1}{M}x[n]$$

$$y_M[n] = \lambda y_{M-1}[n-1] + (1-\lambda)x[n], \quad \lambda = \frac{M-1}{M}$$

From the MA to a first-order recursion

$$y_M[n] = \frac{M-1}{M}y_{M-1}[n-1] + \frac{1}{M}x[n]$$

$$y_M[n] = \lambda y_{M-1}[n-1] + (1-\lambda)x[n], \quad \lambda = \frac{M-1}{M}$$

The Leaky Integrator

- ▶ when M is large, $y_{M-1}[n] \approx y_M[n]$ (and $\lambda \approx 1$)

- ▶ try the filter

$$y[n] = \lambda y[n - 1] + (1 - \lambda)x[n]$$

- ▶ filter is now recursive, since it uses its previous output value

The Leaky Integrator

- ▶ when M is large, $y_{M-1}[n] \approx y_M[n]$ (and $\lambda \approx 1$)
- ▶ try the filter

$$y[n] = \lambda y[n - 1] + (1 - \lambda)x[n]$$

- ▶ filter is now recursive, since it uses its previous output value

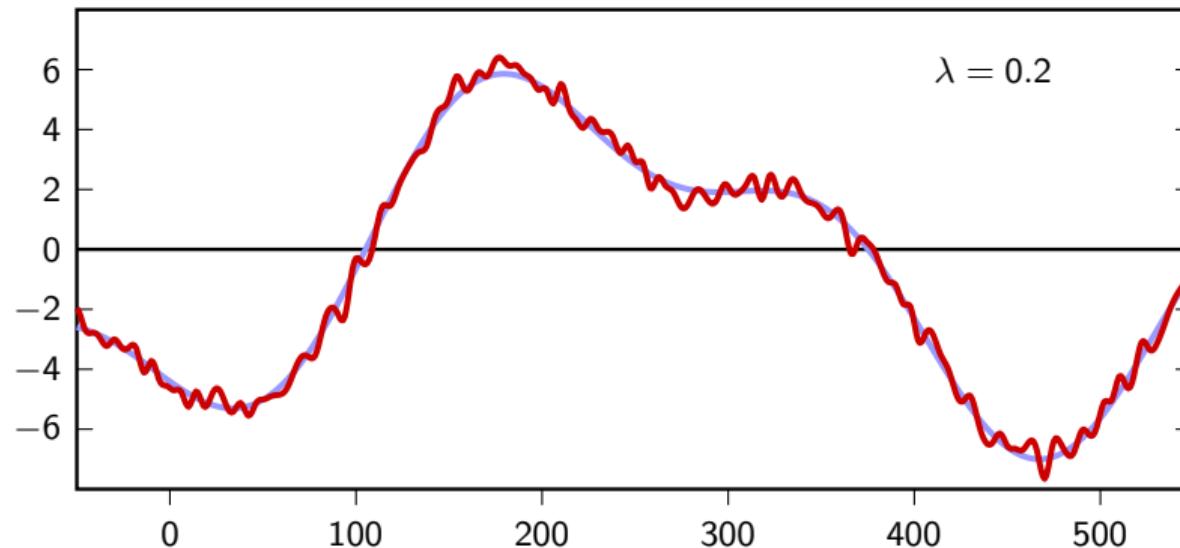
The Leaky Integrator

- ▶ when M is large, $y_{M-1}[n] \approx y_M[n]$ (and $\lambda \approx 1$)
- ▶ try the filter

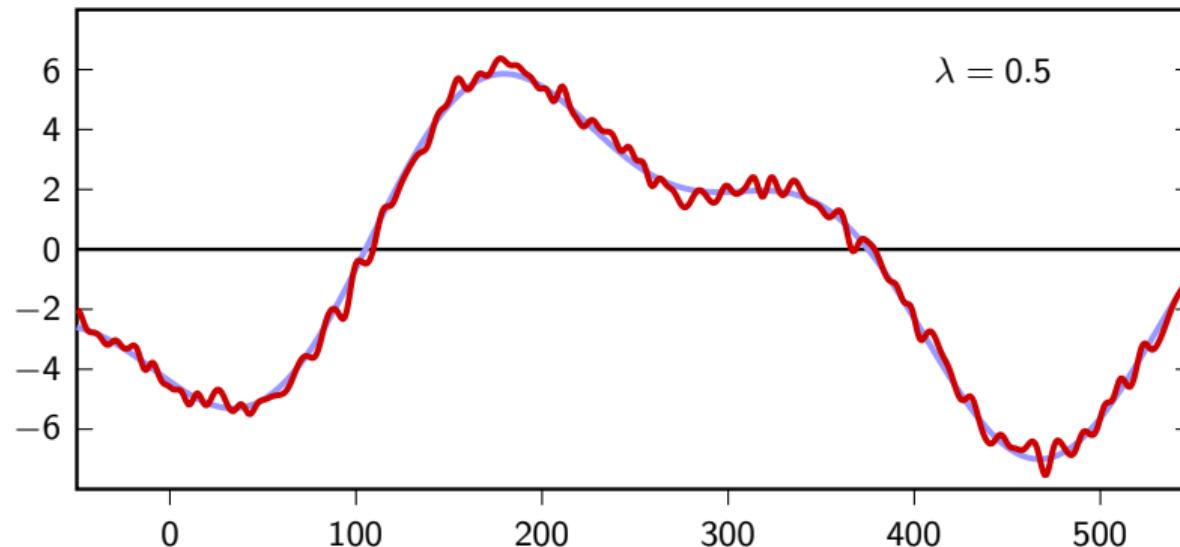
$$y[n] = \lambda y[n - 1] + (1 - \lambda)x[n]$$

- ▶ filter is now recursive, since it uses its previous output value

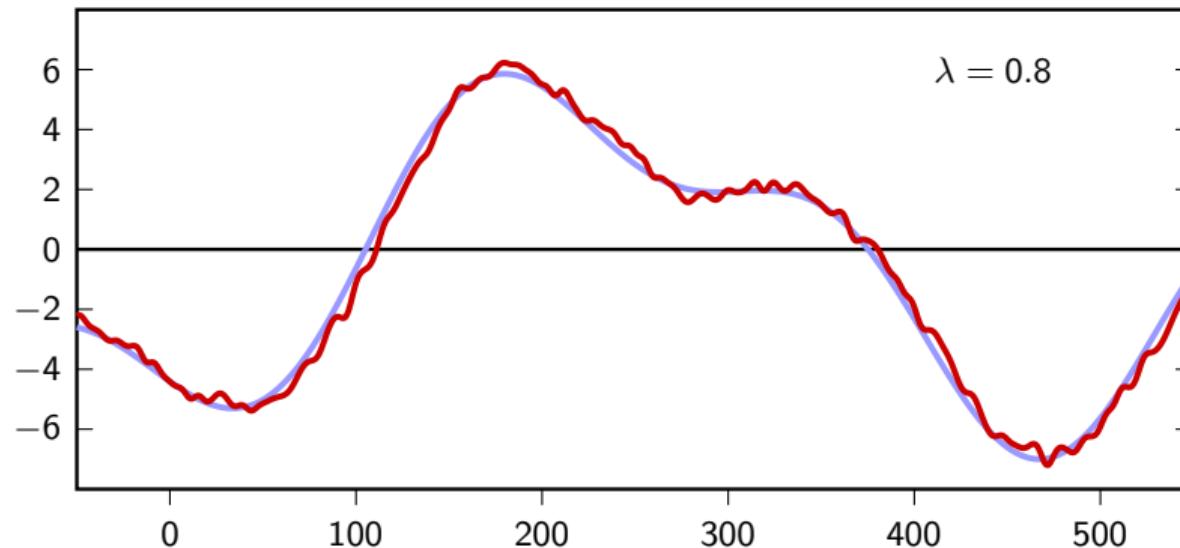
Denoising recursively with the Leaky Integrator



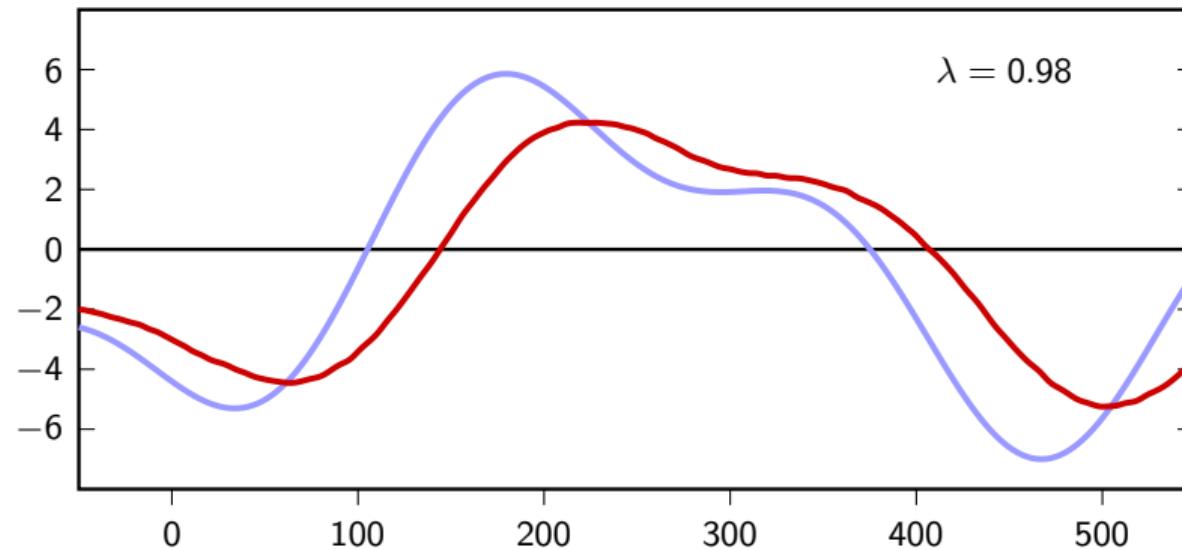
Denoising recursively with the Leaky Integrator



Denoising recursively with the Leaky Integrator



Denoising recursively with the Leaky Integrator



What about the impulse response?

$$y[n] = \lambda y[n - 1] + (1 - \lambda)\delta[n]$$

- ▶ $y[n] = 0$ for all $n < 0$
- ▶ $y[0] = \lambda y[-1] + (1 - \lambda)\delta[0] = (1 - \lambda)$
- ▶ $y[1] = \lambda y[0] + (1 - \lambda)\delta[1] = \lambda(1 - \lambda)$
- ▶ $y[2] = \lambda y[1] + (1 - \lambda)\delta[2] = \lambda^2(1 - \lambda)$
- ▶ $y[3] = \lambda y[2] + (1 - \lambda)\delta[3] = \lambda^3(1 - \lambda)$
- ▶ ...

What about the impulse response?

$$y[n] = \lambda y[n - 1] + (1 - \lambda)\delta[n]$$

- ▶ $y[n] = 0$ for all $n < 0$
- ▶ $y[0] = \lambda y[-1] + (1 - \lambda)\delta[0] = (1 - \lambda)$
- ▶ $y[1] = \lambda y[0] + (1 - \lambda)\delta[1] = \lambda(1 - \lambda)$
- ▶ $y[2] = \lambda y[1] + (1 - \lambda)\delta[2] = \lambda^2(1 - \lambda)$
- ▶ $y[3] = \lambda y[2] + (1 - \lambda)\delta[3] = \lambda^3(1 - \lambda)$
- ▶ ...

What about the impulse response?

$$y[n] = \lambda y[n - 1] + (1 - \lambda)\delta[n]$$

- ▶ $y[n] = 0$ for all $n < 0$
- ▶ $y[0] = \lambda y[-1] + (1 - \lambda)\delta[0] = (1 - \lambda)$
- ▶ $y[1] = \lambda y[0] + (1 - \lambda)\delta[1] = \lambda(1 - \lambda)$
- ▶ $y[2] = \lambda y[1] + (1 - \lambda)\delta[2] = \lambda^2(1 - \lambda)$
- ▶ $y[3] = \lambda y[2] + (1 - \lambda)\delta[3] = \lambda^3(1 - \lambda)$
- ▶ ...

What about the impulse response?

$$y[n] = \lambda y[n - 1] + (1 - \lambda)\delta[n]$$

- ▶ $y[n] = 0$ for all $n < 0$
- ▶ $y[0] = \lambda y[-1] + (1 - \lambda)\delta[0] = (1 - \lambda)$
- ▶ $y[1] = \lambda y[0] + (1 - \lambda)\delta[1] = \lambda(1 - \lambda)$
- ▶ $y[2] = \lambda y[1] + (1 - \lambda)\delta[2] = \lambda^2(1 - \lambda)$
- ▶ $y[3] = \lambda y[2] + (1 - \lambda)\delta[3] = \lambda^3(1 - \lambda)$
- ▶ ...

What about the impulse response?

$$y[n] = \lambda y[n - 1] + (1 - \lambda)\delta[n]$$

- ▶ $y[n] = 0$ for all $n < 0$
- ▶ $y[0] = \lambda y[-1] + (1 - \lambda)\delta[0] = (1 - \lambda)$
- ▶ $y[1] = \lambda y[0] + (1 - \lambda)\delta[1] = \lambda(1 - \lambda)$
- ▶ $y[2] = \lambda y[1] + (1 - \lambda)\delta[2] = \lambda^2(1 - \lambda)$
- ▶ $y[3] = \lambda y[2] + (1 - \lambda)\delta[3] = \lambda^3(1 - \lambda)$
- ▶ ...

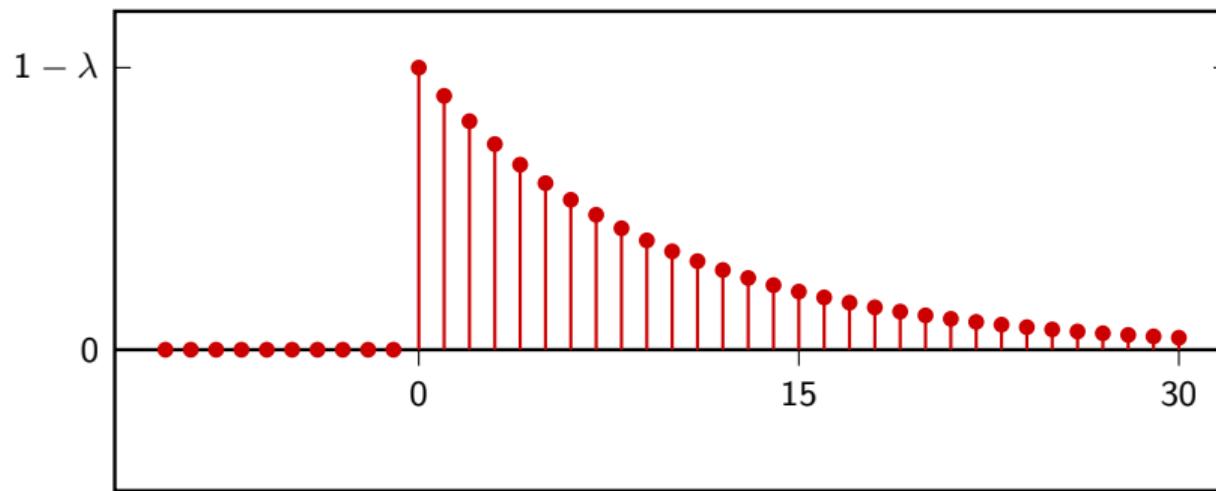
What about the impulse response?

$$y[n] = \lambda y[n - 1] + (1 - \lambda)\delta[n]$$

- ▶ $y[n] = 0$ for all $n < 0$
- ▶ $y[0] = \lambda y[-1] + (1 - \lambda)\delta[0] = (1 - \lambda)$
- ▶ $y[1] = \lambda y[0] + (1 - \lambda)\delta[1] = \lambda(1 - \lambda)$
- ▶ $y[2] = \lambda y[1] + (1 - \lambda)\delta[2] = \lambda^2(1 - \lambda)$
- ▶ $y[3] = \lambda y[2] + (1 - \lambda)\delta[3] = \lambda^3(1 - \lambda)$
- ▶ ...

Impulse response

$$h[n] = (1 - \lambda)\lambda^n u[n]$$



Leaky Integrator: why the name

Discrete-time integrator is a boundless accumulator:

$$y[n] = \sum_{k=-\infty}^n x[k]$$

We can rewrite the integrator as

$$y[n] = y[n - 1] + x[n]$$

Leaky Integrator: why the name

To prevent “explosion” pick $\lambda < 1$

$$y[n] = \lambda y[n - 1] + (1 - \lambda)x[n]$$

keep only a fraction λ of
the accumulated value
so far and forget
("leak") a fraction $1 - \lambda$

add only a fraction $1 - \lambda$
of the current value to
the accumulator

LI: analysis

- ▶ smoothing effect dependent on λ
- ▶ number of operations and storage: *independent of λ*
- ▶ recursion generates infinite-length impulse response
- ▶ infinite-length impulse responses are computable

filter stability

Filter types according to impulse response

- ▶ Finite Impulse Response (FIR)
- ▶ Infinite Impulse Response (IIR)
- ▶ causal
- ▶ noncausal

Filter types according to impulse response

- ▶ Finite Impulse Response (FIR)
- ▶ Infinite Impulse Response (IIR)
- ▶ causal
- ▶ noncausal

Filter types according to impulse response

- ▶ Finite Impulse Response (FIR)
- ▶ Infinite Impulse Response (IIR)
- ▶ causal
- ▶ noncausal

Filter types according to impulse response

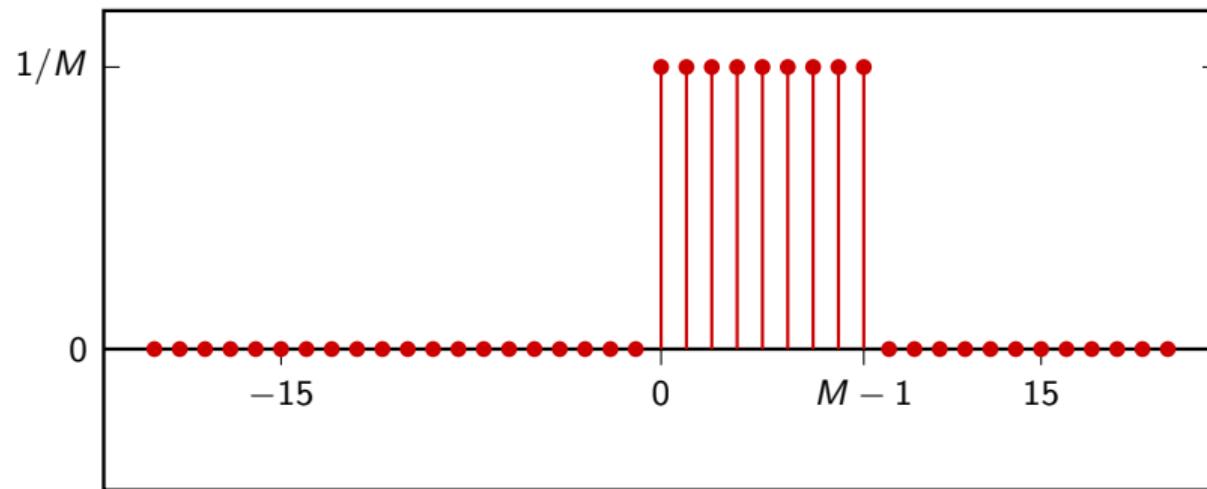
- ▶ Finite Impulse Response (FIR)
- ▶ Infinite Impulse Response (IIR)
- ▶ causal
- ▶ noncausal

FIR

- ▶ impulse response has finite support
- ▶ only a finite number of samples are involved in the computation of each output sample

FIR (example)

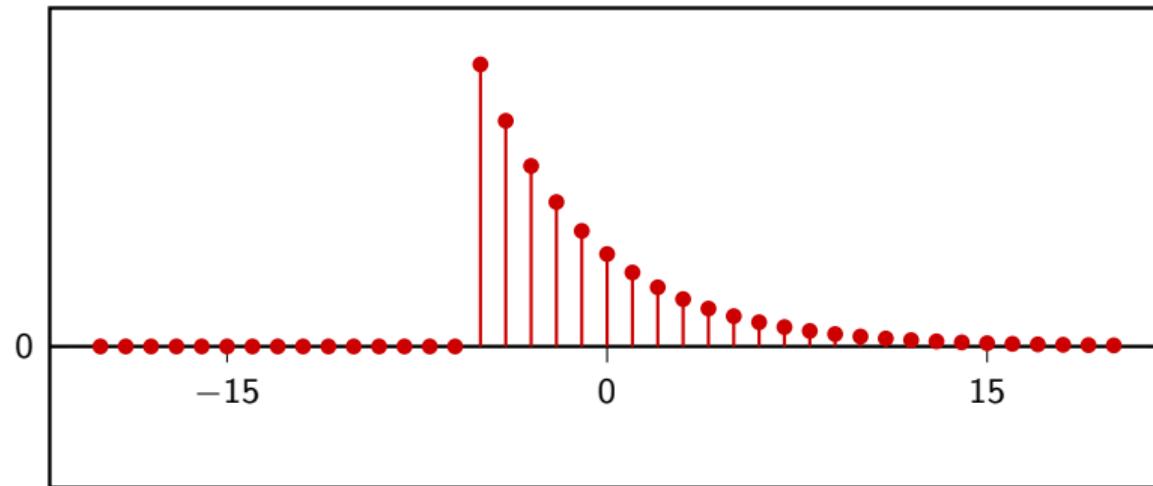
Moving Average filter



- ▶ impulse response has infinite support
- ▶ a potentially infinite number of samples are involved in the computation of each output sample
- ▶ surprisingly, in many cases the computation can still be performed in a finite amount of steps

IIR (example)

Leaky Integrator



Causal vs Noncausal

► causal:

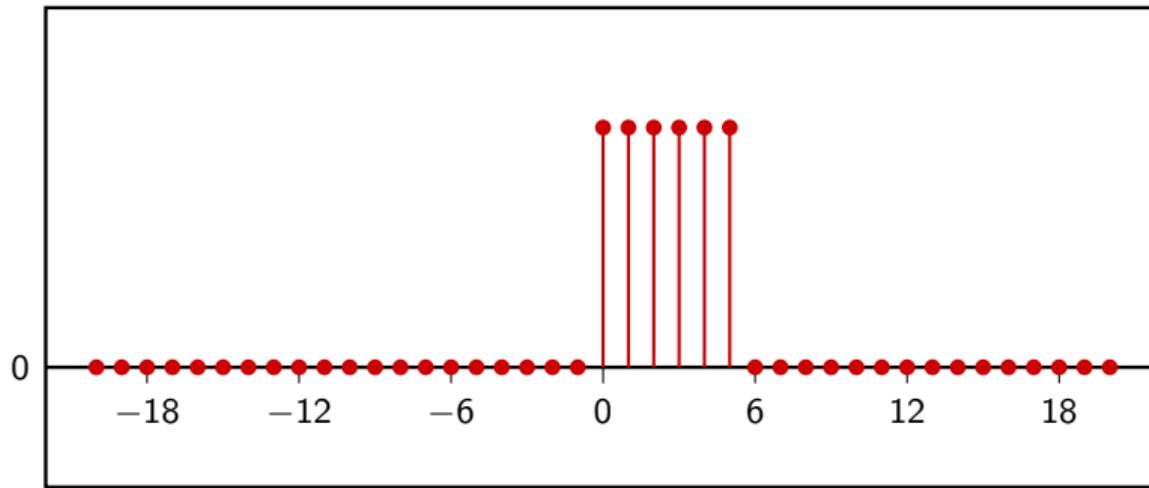
- impulse response is zero for $n < 0$
- only past samples (with respect to the present) are involved in the computation of each output sample
- causal filters can work “on line” since they only need the past

► noncausal:

- impulse response is nonzero for some (or all) $n < 0$
- can still be implemented in an offline fashion (when all input data is available on storage, e.g. in Image Processing)

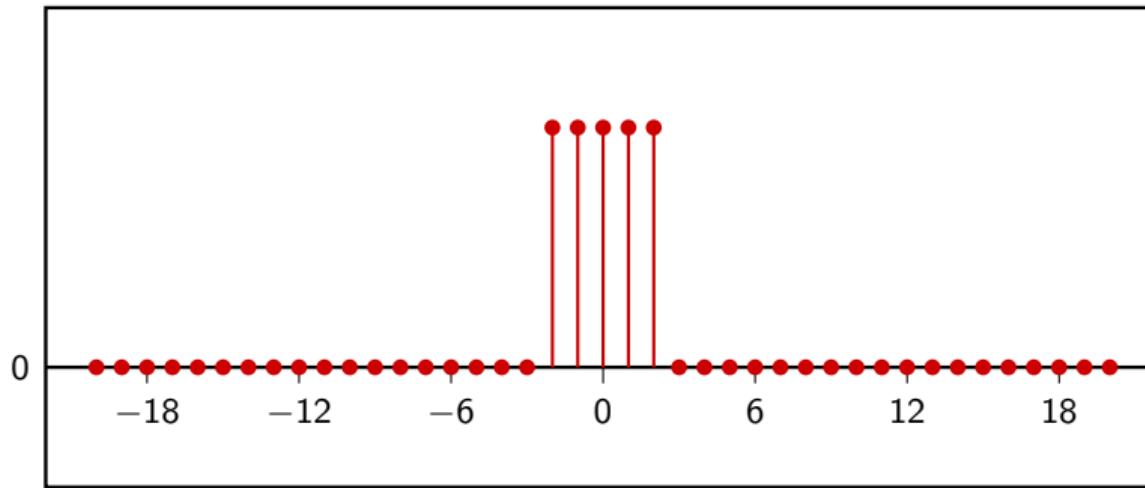
Causal example

Moving Average filter

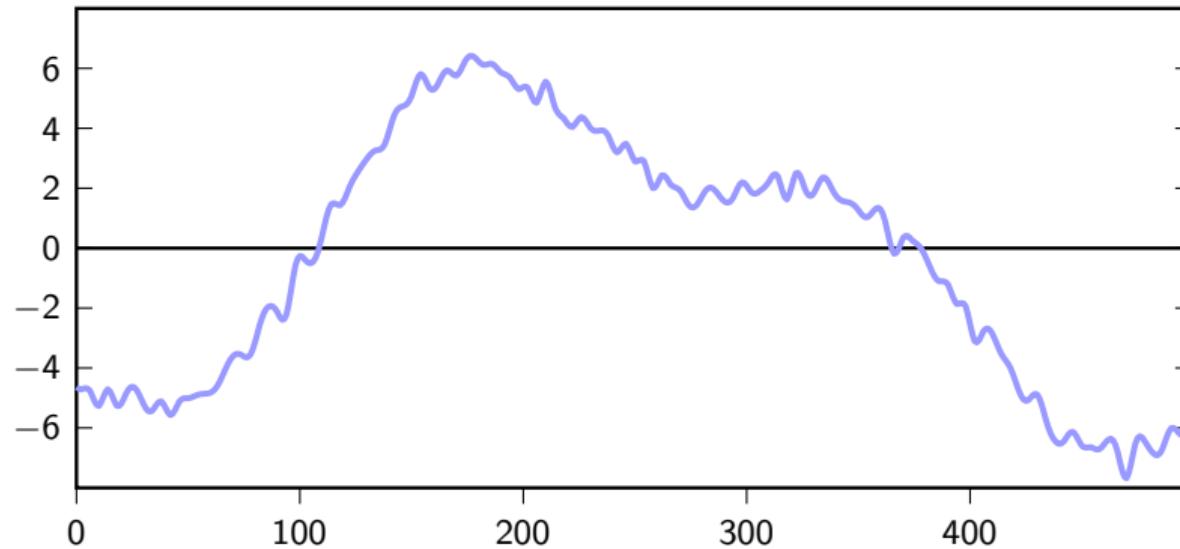


Noncausal example

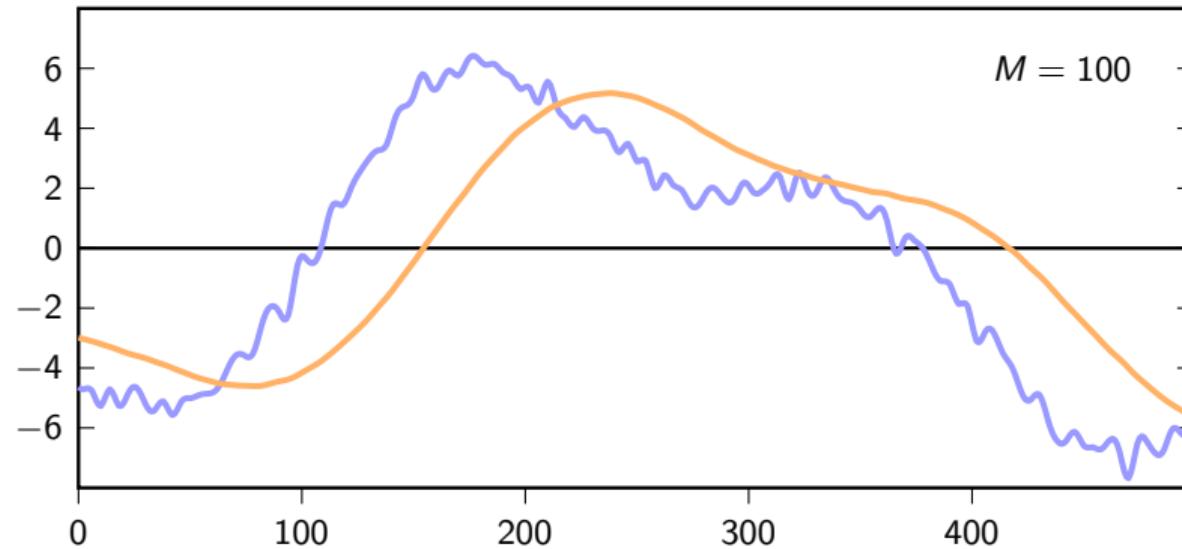
Zero-centered Moving Average filter



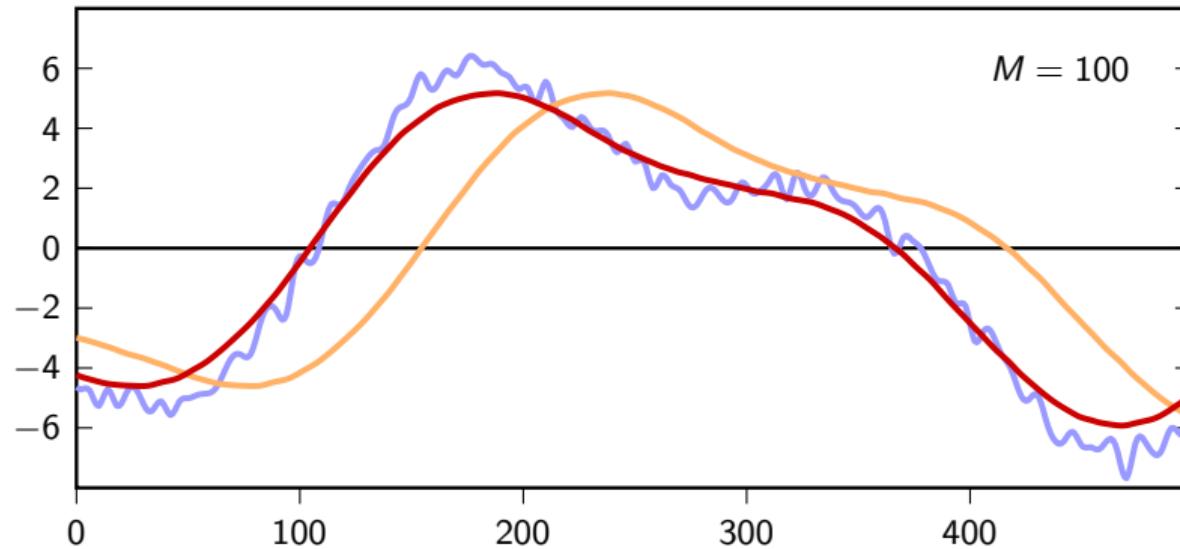
Causal and Noncausal Moving Average



Causal and Noncausal Moving Average



Causal and Noncausal Moving Average



Stability

- ▶ key concept: avoid “explosions” if the input is nice
- ▶ a nice signal is a bounded signal: $|x[n]| < M$ for all n
- ▶ Bounded-Input Bounded-Output (BIBO) stability: if the input is nice the output should be nice

Stability

- ▶ key concept: avoid “explosions” if the input is nice
- ▶ a nice signal is a bounded signal: $|x[n]| < M$ for all n
- ▶ Bounded-Input Bounded-Output (BIBO) stability: if the input is nice the output should be nice

Stability

- ▶ key concept: avoid “explosions” if the input is nice
- ▶ a nice signal is a bounded signal: $|x[n]| < M$ for all n
- ▶ Bounded-Input Bounded-Output (BIBO) stability: if the input is nice the output should be nice

Fundamental Stability Theorem

A filter is BIBO stable if and only if its impulse response is absolutely summable

Proof (\Rightarrow)

Proof:

Hypotheses:

- ▶ $|x[n]| < M$
- ▶ $\sum_n |h[n]| = L < \infty$

Thesis:

- ▶ $|y[n]| < \infty$

$$\begin{aligned}|y[n]| &= \left| \sum_{k=-\infty}^{\infty} h[k]x[n-k] \right| \\ &\leq \sum_{k=-\infty}^{\infty} |h[k]x[n-k]| \\ &\leq M \sum_{k=-\infty}^{\infty} |h[k]| \\ &\leq ML\end{aligned}$$

Proof (\Rightarrow)

Proof:

Hypotheses:

- ▶ $|x[n]| < M$
- ▶ $\sum_n |h[n]| = L < \infty$

Thesis:

- ▶ $|y[n]| < \infty$

$$|y[n]| = \left| \sum_{k=-\infty}^{\infty} h[k]x[n-k] \right|$$

$$\leq \sum_{k=-\infty}^{\infty} |h[k]x[n-k]|$$

$$\leq M \sum_{k=-\infty}^{\infty} |h[k]|$$

$$\leq ML$$

Proof (\Rightarrow)

Proof:

Hypotheses:

- ▶ $|x[n]| < M$
- ▶ $\sum_n |h[n]| = L < \infty$

Thesis:

- ▶ $|y[n]| < \infty$

$$|y[n]| = \left| \sum_{k=-\infty}^{\infty} h[k]x[n-k] \right|$$

$$\leq \sum_{k=-\infty}^{\infty} |h[k]x[n-k]|$$

$$\leq M \sum_{k=-\infty}^{\infty} |h[k]|$$

$$\leq ML$$

Proof (\Rightarrow)

Proof:

Hypotheses:

- ▶ $|x[n]| < M$
- ▶ $\sum_n |h[n]| = L < \infty$

Thesis:

- ▶ $|y[n]| < \infty$

$$\begin{aligned}|y[n]| &= \left| \sum_{k=-\infty}^{\infty} h[k]x[n-k] \right| \\ &\leq \sum_{k=-\infty}^{\infty} |h[k]x[n-k]| \\ &\leq M \sum_{k=-\infty}^{\infty} |h[k]| \\ &\leq ML\end{aligned}$$

Proof (\Rightarrow)

Proof:

Hypotheses:

- ▶ $|x[n]| < M$
- ▶ $\sum_n |h[n]| = L < \infty$

Thesis:

- ▶ $|y[n]| < \infty$

$$|y[n]| = \left| \sum_{k=-\infty}^{\infty} h[k]x[n-k] \right|$$

$$\leq \sum_{k=-\infty}^{\infty} |h[k]x[n-k]|$$

$$\leq M \sum_{k=-\infty}^{\infty} |h[k]|$$

$$\leq ML$$

Proof (\Leftarrow)

Proof (by contradiction):

- ▶ assume hypothesis true, yet $\sum_n |h[n]| = \infty$

Hypothesis:

- ▶ $\forall |x[n]| < \infty, |(x * h)[n]| < \infty$

- ▶ build $x[n] = \begin{cases} +1 & \text{if } h[-n] \geq 0 \\ -1 & \text{if } h[-n] < 0 \end{cases}$

- ▶ clearly, $|x[n]| < \infty$

Thesis:

- ▶ $\sum_n |h[n]| < \infty$

$$(x * h)[0] = \sum_{k=-\infty}^{\infty} h[k]x[-k] = \sum_{k=-\infty}^{\infty} |h[k]| = \infty$$

Proof (\Leftarrow)

Proof (by contradiction):

- ▶ assume hypothesis true, yet $\sum_n |h[n]| = \infty$

Hypothesis:

- ▶ $\forall |x[n]| < \infty,$
 $|x * h[n]| < \infty$

- ▶ build $x[n] = \begin{cases} +1 & \text{if } h[-n] \geq 0 \\ -1 & \text{if } h[-n] < 0 \end{cases}$

- ▶ clearly, $|x[n]| < \infty$

- ▶ however

Thesis:

- ▶ $\sum_n |h[n]| < \infty$

$$(x * h)[0] = \sum_{k=-\infty}^{\infty} h[k]x[-k] = \sum_{k=-\infty}^{\infty} |h[k]| = \infty$$

Proof (\Leftarrow)

Proof (by contradiction):

- ▶ assume hypothesis true, yet $\sum_n |h[n]| = \infty$

Hypothesis:

- ▶ $\forall |x[n]| < \infty, |(x * h)[n]| < \infty$

- ▶ build $x[n] = \begin{cases} +1 & \text{if } h[-n] \geq 0 \\ -1 & \text{if } h[-n] < 0 \end{cases}$

- ▶ clearly, $|x[n]| < \infty$

Thesis:

- ▶ $\sum_n |h[n]| < \infty$

$$(x * h)[0] = \sum_{k=-\infty}^{\infty} h[k]x[-k] = \sum_{k=-\infty}^{\infty} |h[k]| = \infty$$

Proof (\Leftarrow)

Proof (by contradiction):

- ▶ assume hypothesis true, yet $\sum_n |h[n]| = \infty$

Hypothesis:

- ▶ $\forall |x[n]| < \infty, |(x * h)[n]| < \infty$

- ▶ build $x[n] = \begin{cases} +1 & \text{if } h[-n] \geq 0 \\ -1 & \text{if } h[-n] < 0 \end{cases}$

- ▶ clearly, $|x[n]| < \infty$

Thesis:

- ▶ $\sum_n |h[n]| < \infty$

▶ however

$$(x * h)[0] = \sum_{k=-\infty}^{\infty} h[k]x[-k] = \sum_{k=-\infty}^{\infty} |h[k]| = \infty$$

Proof (\Leftarrow)

Proof (by contradiction):

- ▶ assume hypothesis true, yet $\sum_n |h[n]| = \infty$

Hypothesis:

- ▶ $\forall |x[n]| < \infty, |(x * h)[n]| < \infty$

- ▶ build $x[n] = \begin{cases} +1 & \text{if } h[-n] \geq 0 \\ -1 & \text{if } h[-n] < 0 \end{cases}$

- ▶ clearly, $|x[n]| < \infty$

- ▶ however

- ▶ $\sum_n |h[n]| < \infty$

$$(x * h)[0] = \sum_{k=-\infty}^{\infty} h[k]x[-k] = \sum_{k=-\infty}^{\infty} |h[k]| = \infty$$

The good news

FIR filters are always stable

Checking the stability of IIRs

Let's check the Leaky Integrator:

$$\begin{aligned}\sum_{n=-\infty}^{\infty} |h[n]| &= |1 - \lambda| \sum_{n=0}^{\infty} |\lambda|^n \\ &= \lim_{n \rightarrow \infty} |1 - \lambda| \frac{1 - |\lambda|^{n+1}}{1 - |\lambda|} \\ &< \infty \quad \text{for } |\lambda| < 1\end{aligned}$$

stability is guaranteed for $|\lambda| < 1$

Checking the stability of IIRs

Let's check the Leaky Integrator:

$$\begin{aligned} \sum_{n=-\infty}^{\infty} |h[n]| &= |1 - \lambda| \sum_{n=0}^{\infty} |\lambda|^n \\ &= \lim_{n \rightarrow \infty} |1 - \lambda| \frac{1 - |\lambda|^{n+1}}{1 - |\lambda|} \\ &< \infty \quad \text{for } |\lambda| < 1 \end{aligned}$$

stability is guaranteed for $|\lambda| < 1$

Checking the stability of IIRs

Let's check the Leaky Integrator:

$$\begin{aligned} \sum_{n=-\infty}^{\infty} |h[n]| &= |1 - \lambda| \sum_{n=0}^{\infty} |\lambda|^n \\ &= \lim_{n \rightarrow \infty} |1 - \lambda| \frac{1 - |\lambda|^{n+1}}{1 - |\lambda|} \\ &< \infty \quad \text{for } |\lambda| < 1 \end{aligned}$$

stability is guaranteed for $|\lambda| < 1$

Checking the stability of IIRs

Let's check the Leaky Integrator:

$$\begin{aligned} \sum_{n=-\infty}^{\infty} |h[n]| &= |1 - \lambda| \sum_{n=0}^{\infty} |\lambda|^n \\ &= \lim_{n \rightarrow \infty} |1 - \lambda| \frac{1 - |\lambda|^{n+1}}{1 - |\lambda|} \\ &< \infty \quad \text{for } |\lambda| < 1 \end{aligned}$$

stability is guaranteed for $|\lambda| < 1$

Checking the stability of IIRs

We will study indirect methods for filter stability later

frequency response

Overview:

- ▶ Eigensequences
- ▶ Convolution theorem
- ▶ Frequency and phase response

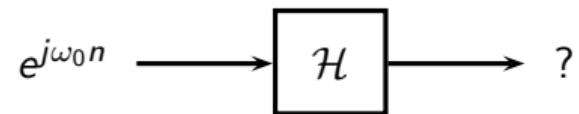
Overview:

- ▶ Eigensequences
- ▶ Convolution theorem
- ▶ Frequency and phase response

Overview:

- ▶ Eigensequences
- ▶ Convolution theorem
- ▶ Frequency and phase response

A remarkable result



A remarkable result

$$\begin{aligned}y[n] &= e^{j\omega_0 n} * h[n] \\&= h[n] * e^{j\omega_0 n} \\&= \sum_{k=-\infty}^{\infty} h[k] e^{j\omega_0(n-k)} \\&= e^{j\omega_0 n} \sum_{k=-\infty}^{\infty} h[k] e^{-j\omega_0 k} \\&= H(e^{j\omega_0}) e^{j\omega_0 n}\end{aligned}$$

A remarkable result

$$\begin{aligned}y[n] &= e^{j\omega_0 n} * h[n] \\&= h[n] * e^{j\omega_0 n} \\&= \sum_{k=-\infty}^{\infty} h[k] e^{j\omega_0(n-k)} \\&= e^{j\omega_0 n} \sum_{k=-\infty}^{\infty} h[k] e^{-j\omega_0 k} \\&= H(e^{j\omega_0}) e^{j\omega_0 n}\end{aligned}$$

A remarkable result

$$\begin{aligned}y[n] &= e^{j\omega_0 n} * h[n] \\&= h[n] * e^{j\omega_0 n} \\&= \sum_{k=-\infty}^{\infty} h[k] e^{j\omega_0(n-k)} \\&= e^{j\omega_0 n} \sum_{k=-\infty}^{\infty} h[k] e^{-j\omega_0 k} \\&= H(e^{j\omega_0}) e^{j\omega_0 n}\end{aligned}$$

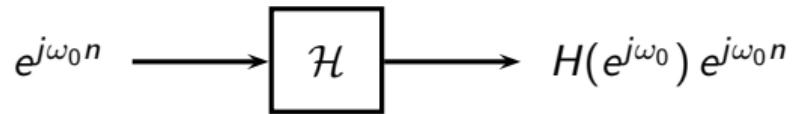
A remarkable result

$$\begin{aligned}y[n] &= e^{j\omega_0 n} * h[n] \\&= h[n] * e^{j\omega_0 n} \\&= \sum_{k=-\infty}^{\infty} h[k] e^{j\omega_0(n-k)} \\&= e^{j\omega_0 n} \sum_{k=-\infty}^{\infty} h[k] e^{-j\omega_0 k} \\&= H(e^{j\omega_0}) e^{j\omega_0 n}\end{aligned}$$

A remarkable result

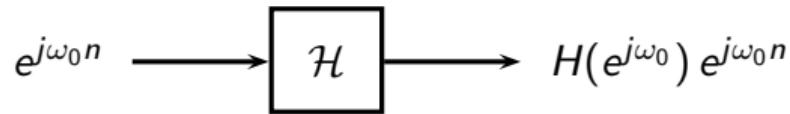
$$\begin{aligned}y[n] &= e^{j\omega_0 n} * h[n] \\&= h[n] * e^{j\omega_0 n} \\&= \sum_{k=-\infty}^{\infty} h[k] e^{j\omega_0(n-k)} \\&= e^{j\omega_0 n} \sum_{k=-\infty}^{\infty} h[k] e^{-j\omega_0 k} \\&= H(e^{j\omega_0}) e^{j\omega_0 n}\end{aligned}$$

A remarkable result



- ▶ complex exponentials are *eigensequences* of LTI systems, i.e., linear filters cannot change the frequency of sinusoids
- ▶ DTFT of impulse response determines the frequency characteristic of a filter

A remarkable result



- ▶ complex exponentials are *eigensequences* of LTI systems, i.e., linear filters cannot change the frequency of sinusoids
- ▶ DTFT of impulse response determines the frequency characteristic of a filter

Magnitude and phase

If $H(e^{j\omega_0}) = Ae^{j\theta}$, then

$$\mathcal{H}\{e^{j\omega_0 n}\} = Ae^{j(\omega_0 n + \theta)}$$

amplitude:

amplification ($A > 1$)

or attenuation ($0 \leq A < 1$)



phase shift:

delay ($\theta < 0$)

or advancement ($\theta > 0$)

The convolution theorem

In general:

$$\text{DTFT } \{x[n] * h[n]\} = ?$$

Intuition: the DTFT reconstruction formula tells us how to build $x[n]$ from a set of complex exponential “basis” functions. By linearity...

The convolution theorem

In general:

$$\text{DTFT } \{x[n] * h[n]\} = ?$$

Intuition: the DTFT reconstruction formula tells us how to build $x[n]$ from a set of complex exponential “basis” functions. By linearity...

The convolution theorem

$$\begin{aligned}\text{DTFT} \{x[n] * h[n]\} &= \sum_{n=-\infty}^{\infty} (x * h)[n] e^{-j\omega n} \\&= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k] h[n-k] e^{-j\omega n} \\&= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k] h[n-k] e^{-j\omega(n-k)} e^{-j\omega k} \\&= \sum_{k=-\infty}^{\infty} x[k] e^{-j\omega k} \sum_{n=-\infty}^{\infty} h[n-k] e^{-j\omega(n-k)} \\&= H(e^{j\omega}) X(e^{j\omega})\end{aligned}$$

The convolution theorem

$$\begin{aligned}\text{DTFT} \{x[n] * h[n]\} &= \sum_{n=-\infty}^{\infty} (x * h)[n] e^{-j\omega n} \\&= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k] h[n-k] e^{-j\omega n} \\&= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k] h[n-k] e^{-j\omega(n-k)} e^{-j\omega k} \\&= \sum_{k=-\infty}^{\infty} x[k] e^{-j\omega k} \sum_{n=-\infty}^{\infty} h[n-k] e^{-j\omega(n-k)} \\&= H(e^{j\omega}) X(e^{j\omega})\end{aligned}$$

The convolution theorem

$$\begin{aligned}\text{DTFT} \{x[n] * h[n]\} &= \sum_{n=-\infty}^{\infty} (x * h)[n] e^{-j\omega n} \\&= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k] h[n-k] e^{-j\omega n} \\&= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k] h[n-k] e^{-j\omega(n-k)} e^{-j\omega k} \\&= \sum_{k=-\infty}^{\infty} x[k] e^{-j\omega k} \sum_{n=-\infty}^{\infty} h[n-k] e^{-j\omega(n-k)} \\&= H(e^{j\omega}) X(e^{j\omega})\end{aligned}$$

The convolution theorem

$$\begin{aligned}\text{DTFT} \{x[n] * h[n]\} &= \sum_{n=-\infty}^{\infty} (x * h)[n] e^{-j\omega n} \\&= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k] h[n-k] e^{-j\omega n} \\&= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k] h[n-k] e^{-j\omega(n-k)} e^{-j\omega k} \\&= \sum_{k=-\infty}^{\infty} x[k] e^{-j\omega k} \sum_{n=-\infty}^{\infty} h[n-k] e^{-j\omega(n-k)} \\&= H(e^{j\omega}) X(e^{j\omega})\end{aligned}$$

The convolution theorem

$$\begin{aligned}\text{DTFT} \{x[n] * h[n]\} &= \sum_{n=-\infty}^{\infty} (x * h)[n] e^{-j\omega n} \\&= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k] h[n-k] e^{-j\omega n} \\&= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k] h[n-k] e^{-j\omega(n-k)} e^{-j\omega k} \\&= \sum_{k=-\infty}^{\infty} x[k] e^{-j\omega k} \sum_{n=-\infty}^{\infty} h[n-k] e^{-j\omega(n-k)} \\&= H(e^{j\omega}) X(e^{j\omega})\end{aligned}$$

Frequency response

$$H(e^{j\omega}) = \text{DTFT} \{ h[n] \}$$

Two effects:

- ▶ **magnitude:** amplification ($|H(e^{j\omega})| > 1$) or attenuation ($|H(e^{j\omega})| < 1$) of input frequencies
- ▶ **phase:** overall delay and shape changes

Frequency response

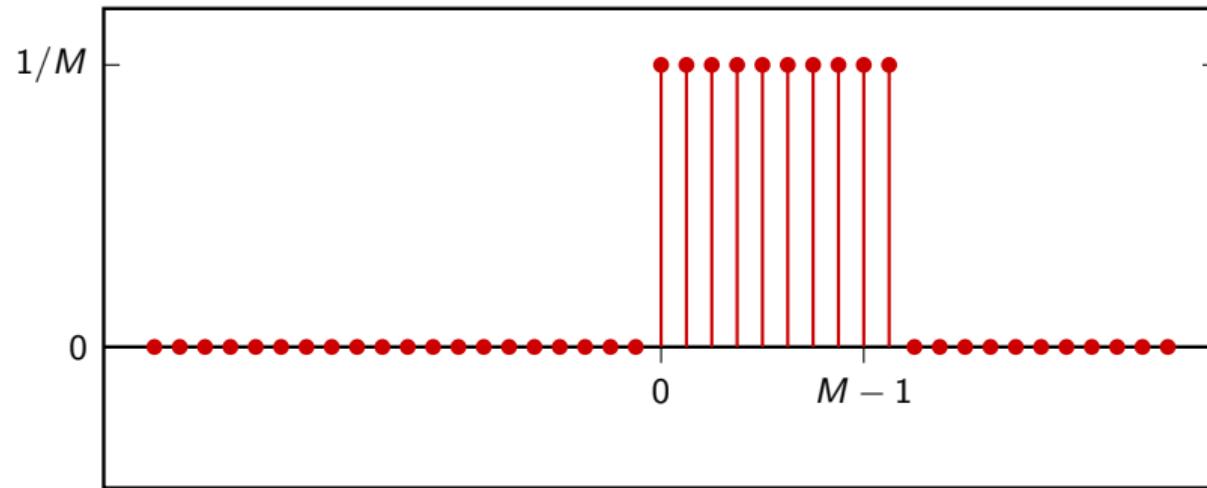
$$H(e^{j\omega}) = \text{DTFT} \{ h[n] \}$$

Two effects:

- ▶ **magnitude:** amplification ($|H(e^{j\omega})| > 1$) or attenuation ($|H(e^{j\omega})| < 1$) of input frequencies
- ▶ **phase:** overall delay and shape changes

Moving Average revisited

$$h[n] = (u[n] - u[n - M])/M$$

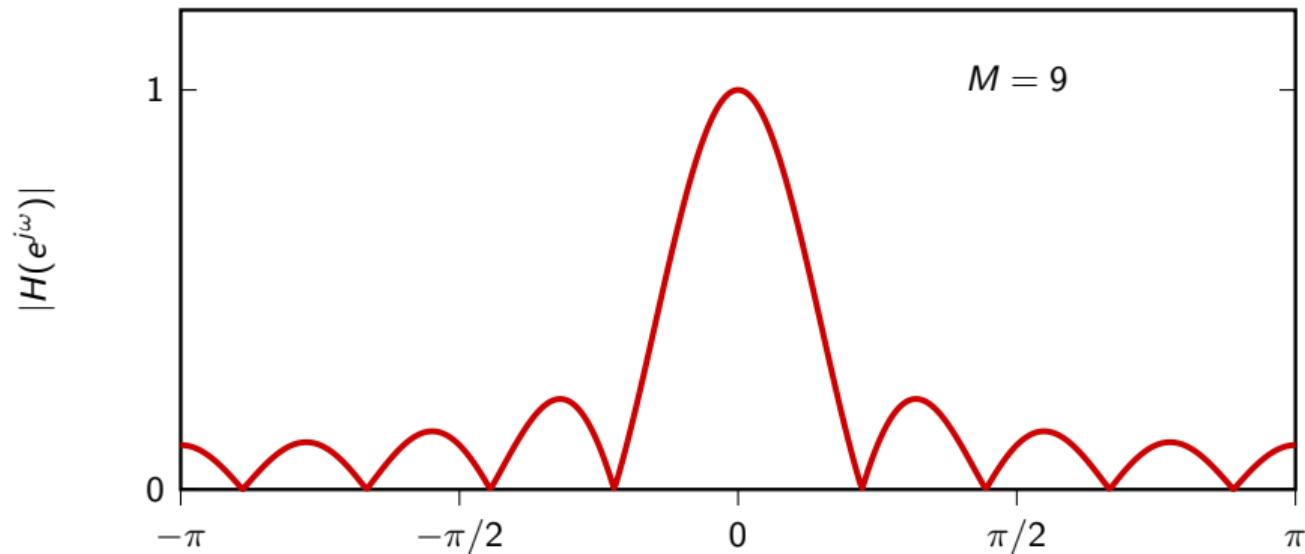


Moving Average revisited

$$\begin{aligned} H(e^{j\omega}) &= \sum_{n=0}^{M-1} \frac{1}{M} e^{-j\omega n} \\ &= \frac{1}{M} \frac{1 - e^{-j\omega M}}{1 - e^{-j\omega}} \\ &= \frac{1}{M} \frac{e^{-j\frac{\omega M}{2}} \left[e^{j\frac{\omega M}{2}} - e^{-j\frac{\omega M}{2}} \right]}{e^{-j\frac{\omega}{2}} \left[e^{j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}} \right]} \\ &= \frac{1}{M} \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} e^{-j\frac{\omega}{2}(M-1)} \end{aligned}$$

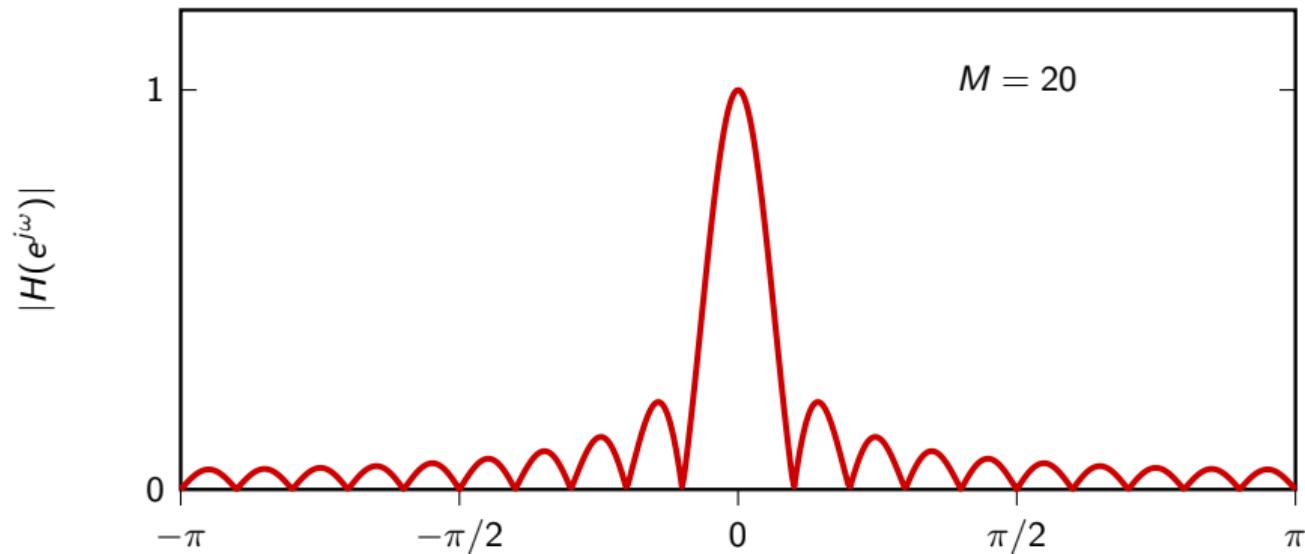
Moving Average, magnitude response

$$|H(e^{j\omega})| = \frac{1}{M} \left| \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} \right|$$



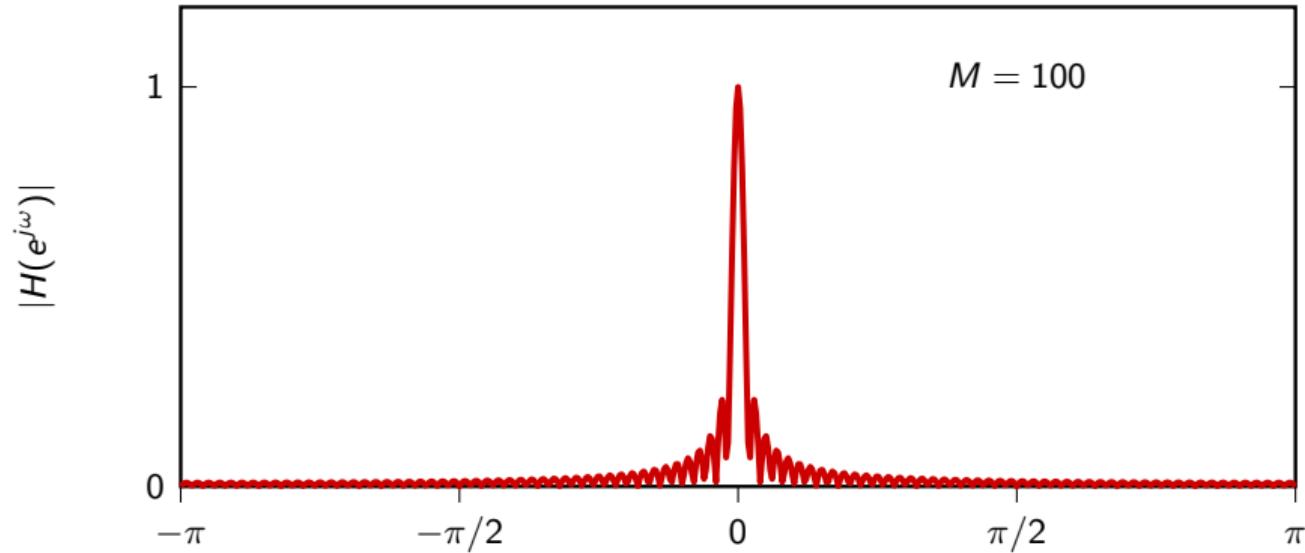
Moving Average, magnitude response

$$|H(e^{j\omega})| = \frac{1}{M} \left| \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} \right|$$

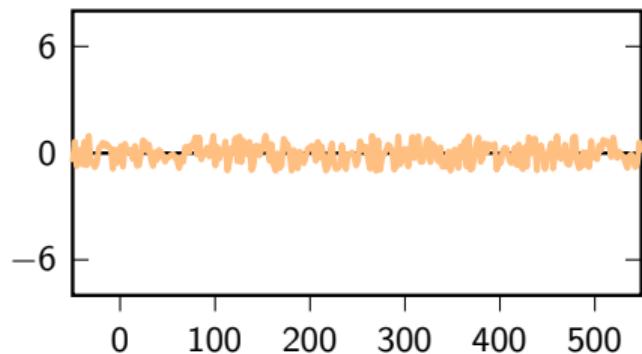
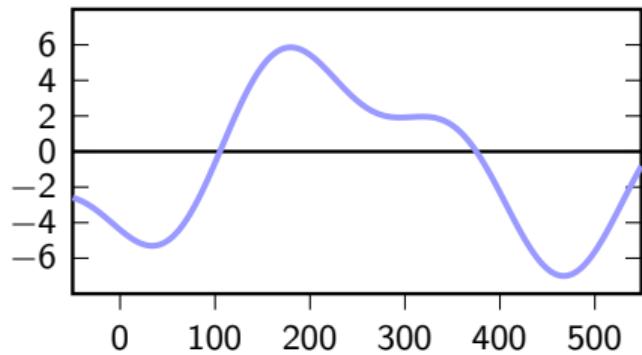


Moving Average, magnitude response

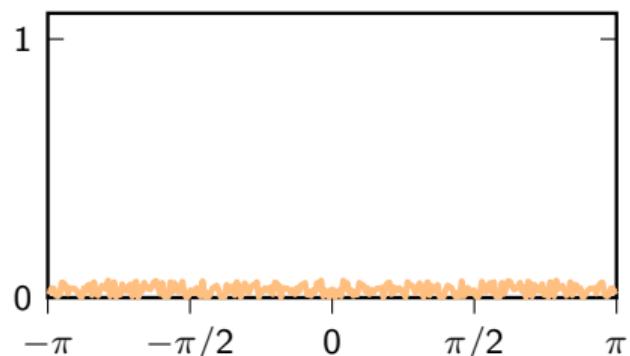
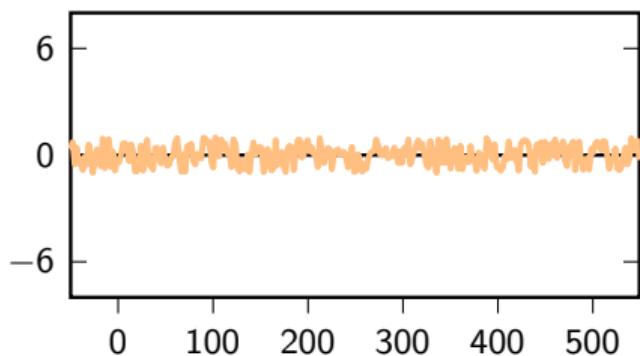
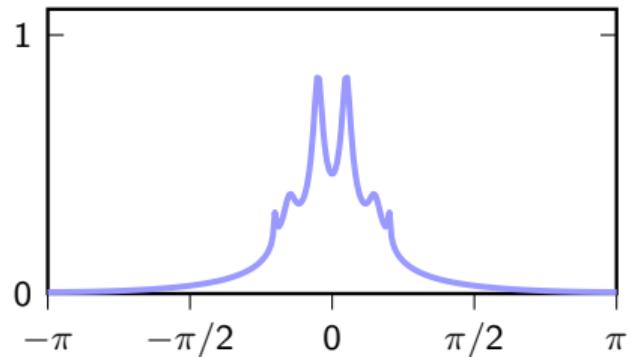
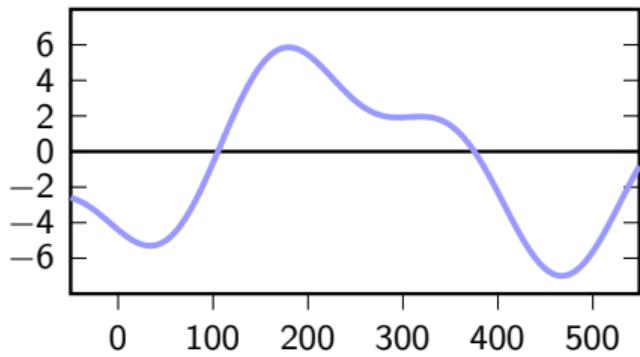
$$|H(e^{j\omega})| = \frac{1}{M} \left| \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} \right|$$



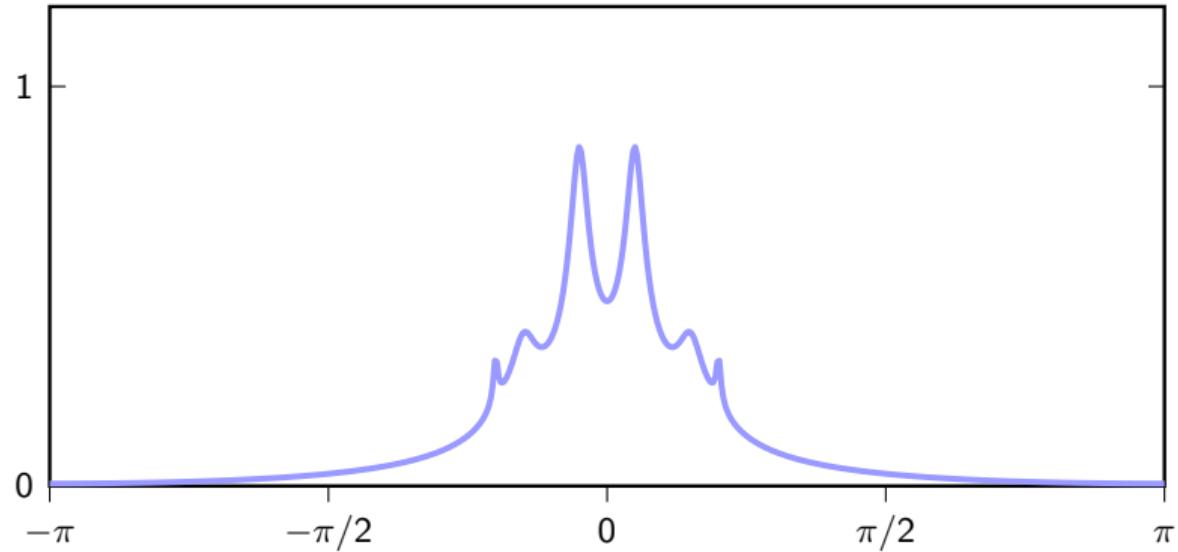
Denoising revisited



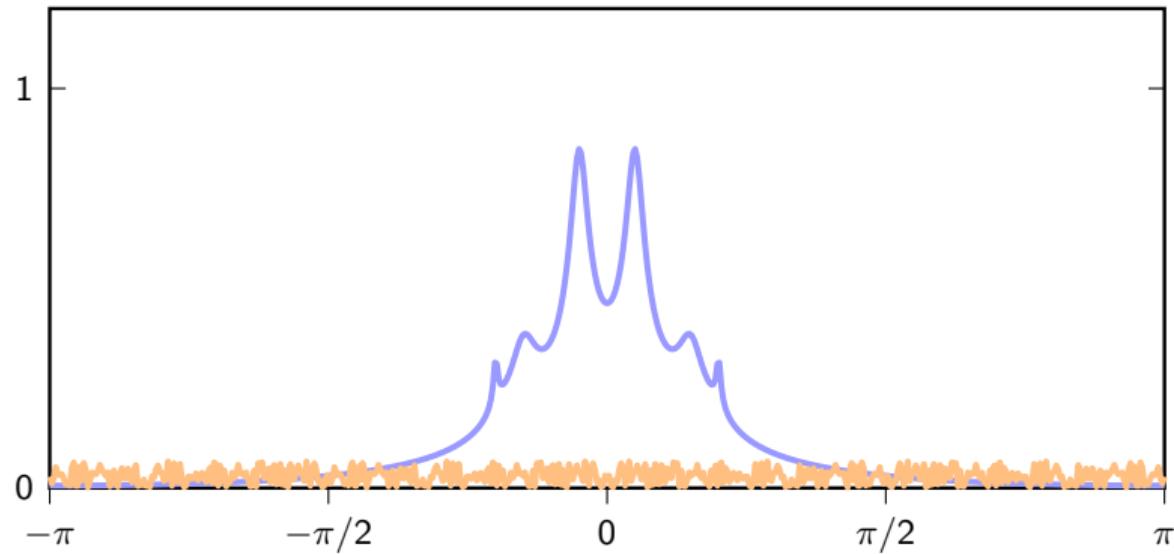
Denoising revisited



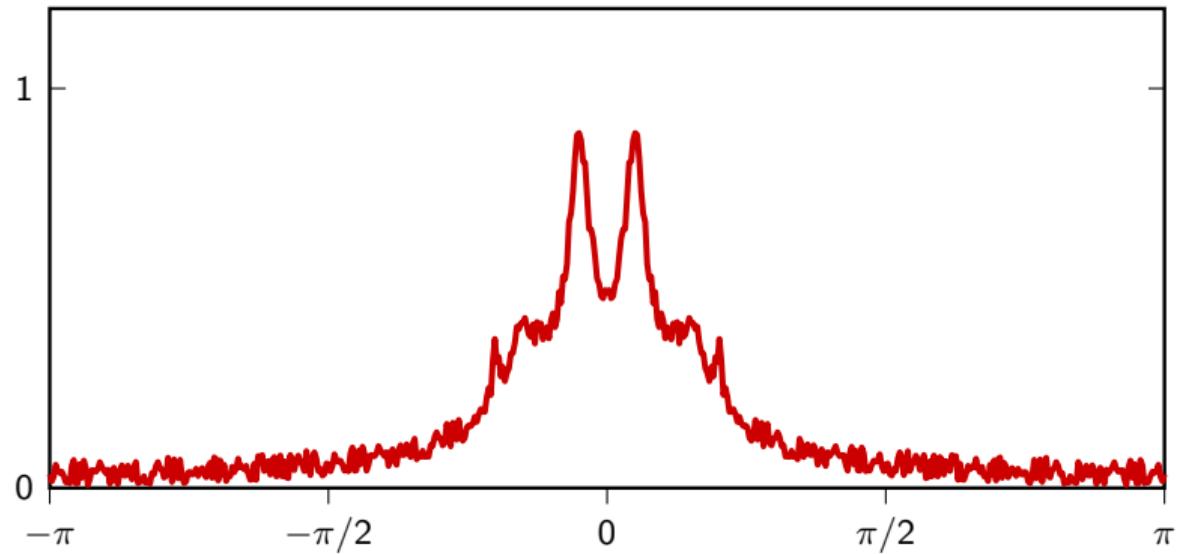
Denoising revisited



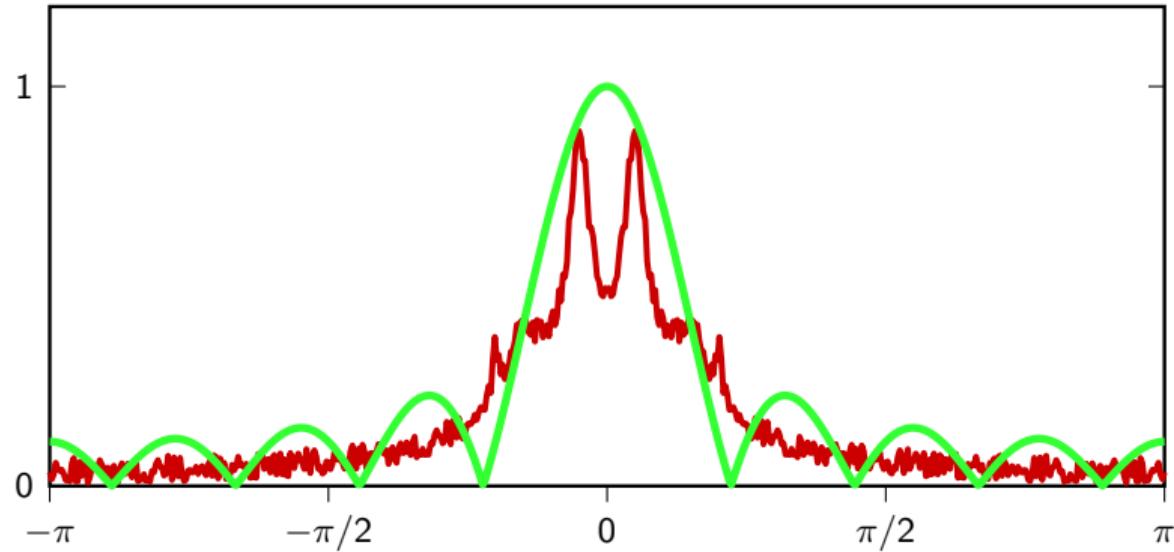
Denoising revisited



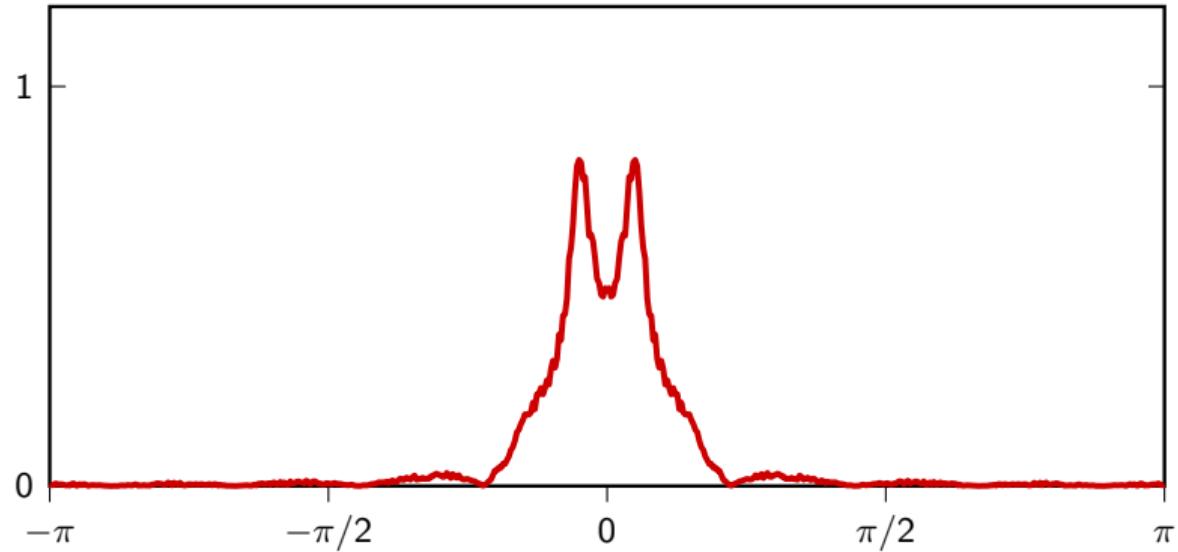
Denoising revisited



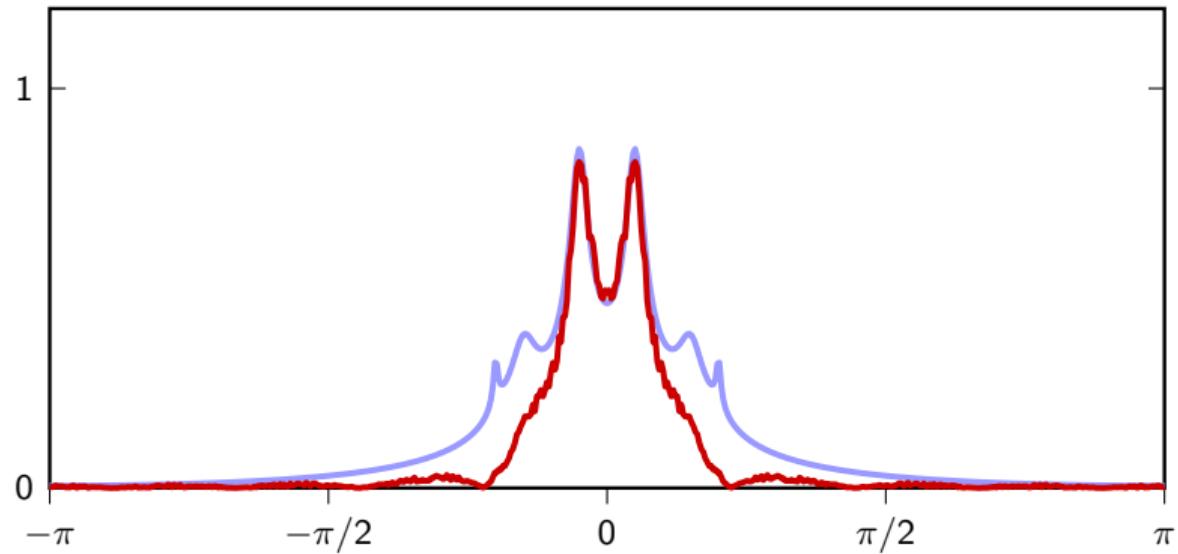
Denoising revisited



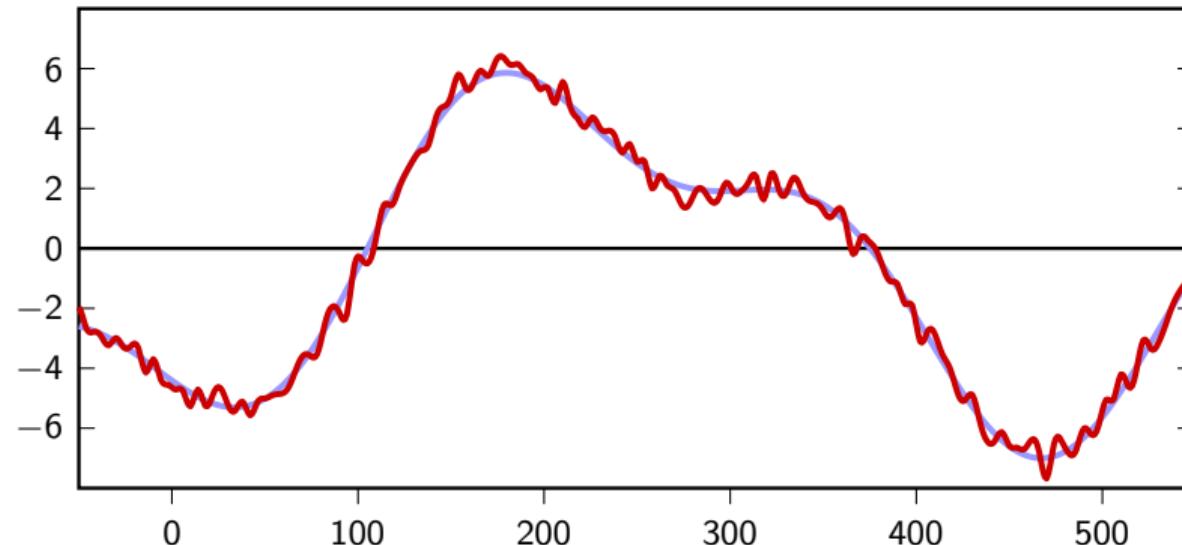
Denoising revisited



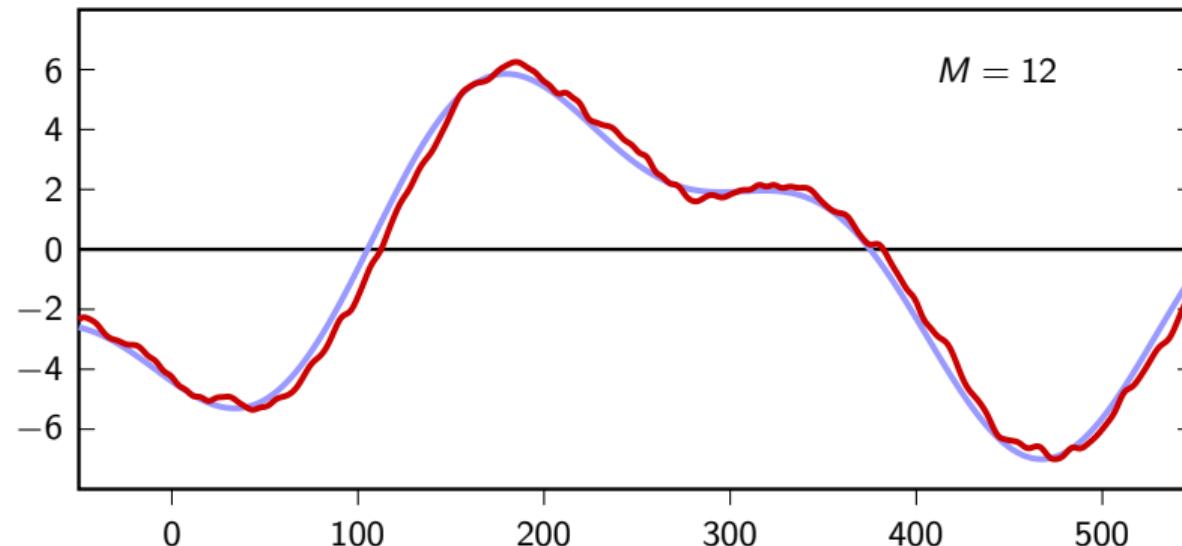
Denoising revisited



By the way, remember the time-domain analysis...



By the way, remember the time-domain analysis...



What about the phase?

Assume $|H(e^{j\omega})| = 1$

- ▶ zero phase: $\angle H(e^{j\omega}) = 0$
- ▶ linear phase: $\angle H(e^{j\omega}) = d\omega$
- ▶ nonlinear phase

What about the phase?

Assume $|H(e^{j\omega})| = 1$

- ▶ zero phase: $\angle H(e^{j\omega}) = 0$
- ▶ linear phase: $\angle H(e^{j\omega}) = d\omega$
- ▶ nonlinear phase

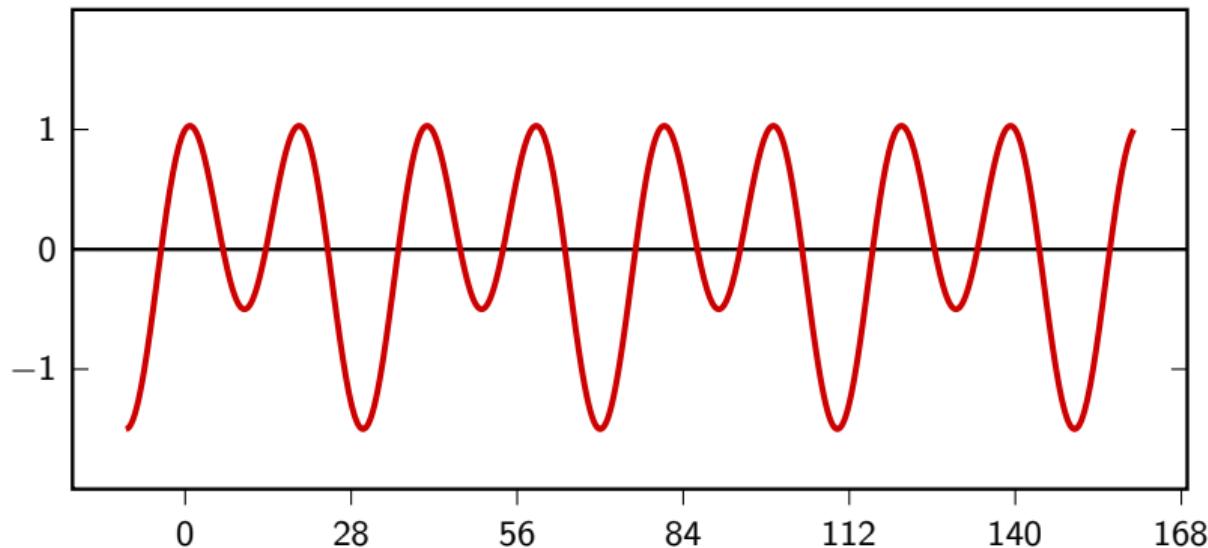
What about the phase?

Assume $|H(e^{j\omega})| = 1$

- ▶ zero phase: $\angle H(e^{j\omega}) = 0$
- ▶ linear phase: $\angle H(e^{j\omega}) = d\omega$
- ▶ nonlinear phase

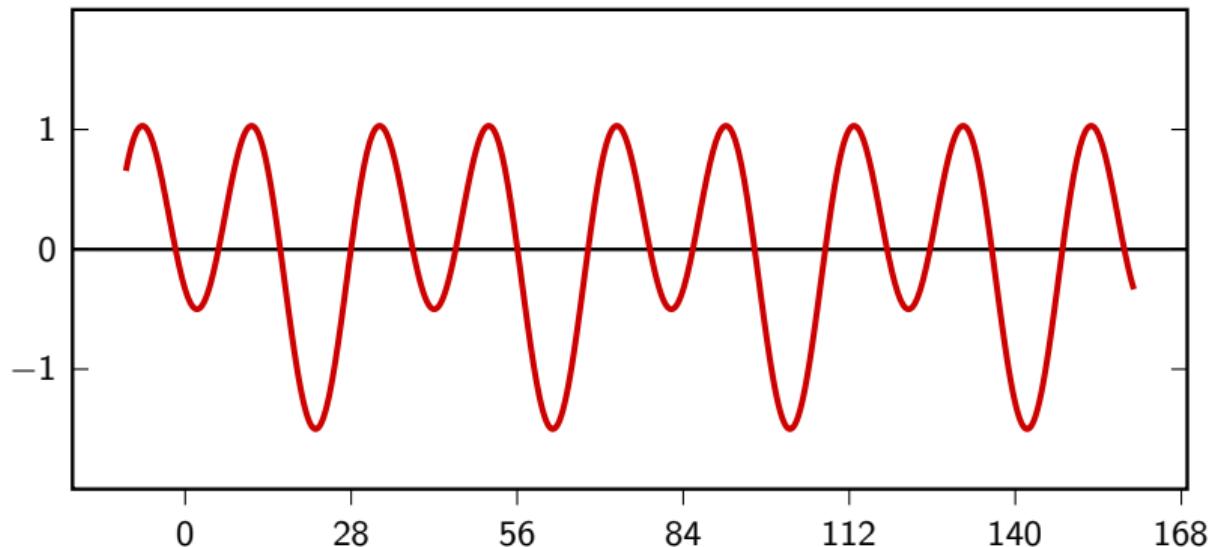
Phase and signal shape

$$x[n] = \frac{1}{2} \sin(\omega_0 n) + \cos(2\omega_0 n) \quad \omega_0 = \frac{2\pi}{40}$$



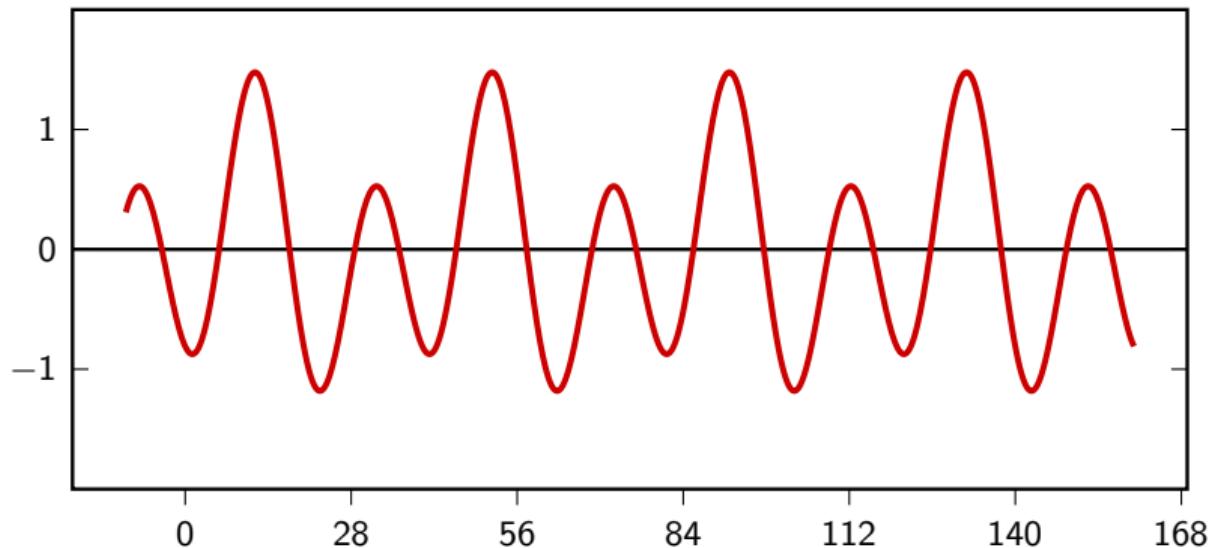
Phase and signal shape: linear phase

$$x[n] = \frac{1}{2} \sin(\omega_0 n + \theta_0) + \cos(2\omega_0 n + 2\theta_0) \quad \theta_0 = \frac{8\pi}{5}$$

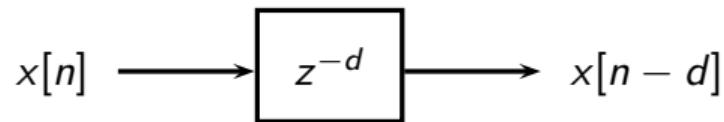


Phase and signal shape: nonlinear phase

$$x[n] = \frac{1}{2} \sin(\omega_0 n) + \cos(2\omega_0 n + 2\theta_0)$$

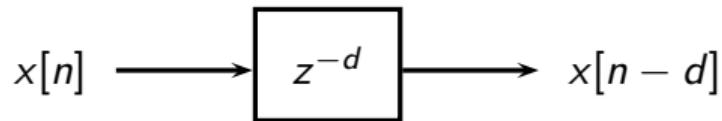


Linear phase



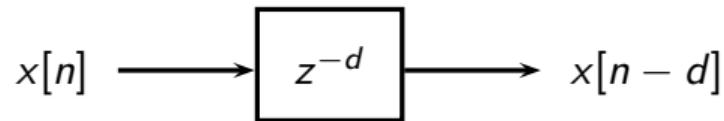
- ▶ $y[n] = x[n - d]$
- ▶ $Y(e^{j\omega}) = e^{-j\omega d} X(e^{j\omega})$
- ▶ $H(e^{j\omega}) = e^{-j\omega d}$
- ▶ linear phase term

Linear phase



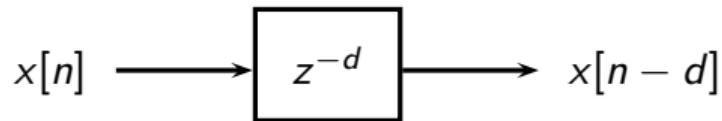
- ▶ $y[n] = x[n - d]$
- ▶ $Y(e^{j\omega}) = e^{-j\omega d} X(e^{j\omega})$
- ▶ $H(e^{j\omega}) = e^{-j\omega d}$
- ▶ linear phase term

Linear phase



- ▶ $y[n] = x[n - d]$
- ▶ $Y(e^{j\omega}) = e^{-j\omega d} X(e^{j\omega})$
- ▶ $H(e^{j\omega}) = e^{-j\omega d}$
- ▶ linear phase term

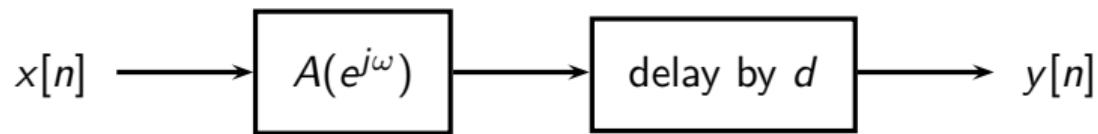
Linear phase



- ▶ $y[n] = x[n - d]$
- ▶ $Y(e^{j\omega}) = e^{-j\omega d} X(e^{j\omega})$
- ▶ $H(e^{j\omega}) = e^{-j\omega d}$
- ▶ linear phase term

Linear phase

In general, if $H(e^{j\omega}) = A(e^{j\omega})e^{-j\omega d}$, with $A(e^{j\omega}) \in \mathbb{R}$

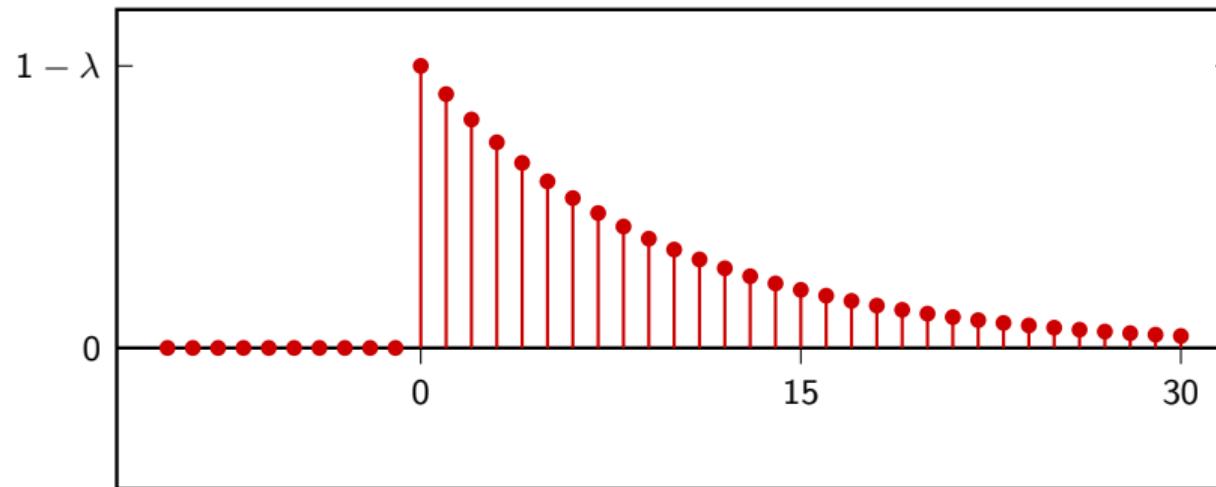


Moving Average is linear phase

$$H(e^{j\omega}) = \frac{1}{M} \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} e^{-j\frac{M-1}{2}\omega}$$

Leaky integrator revisited

$$h[n] = (1 - \lambda)\lambda^n u[n]$$



Leaky integrator revisited

$$H(e^{j\omega}) = \frac{1 - \lambda}{1 - \lambda e^{-j\omega}}$$

Finding magnitude and phase require a little algebra...

Leaky integrator revisited

Recall from complex algebra:

$$\frac{1}{a+jb} = \frac{a-jb}{a^2+b^2}$$

so that if $x = 1/(a+jb)$,

$$|x|^2 = \frac{1}{a^2+b^2}$$

$$\angle x = \tan^{-1} \left[-\frac{b}{a} \right]$$

Leaky integrator revisited

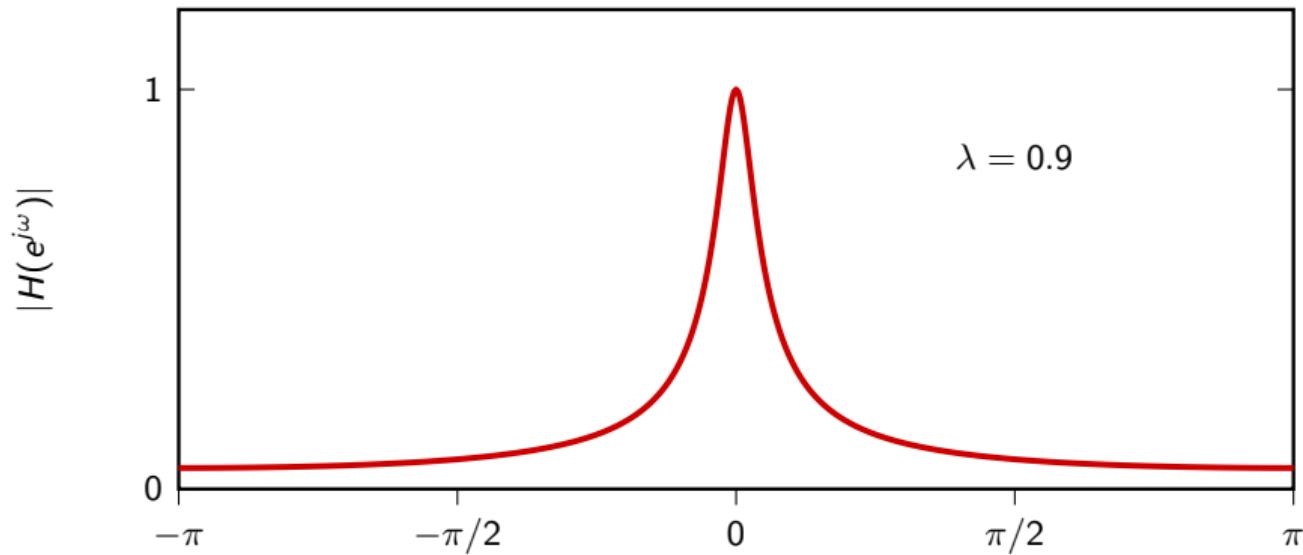
$$H(e^{j\omega}) = \frac{1 - \lambda}{(1 - \lambda \cos \omega) - j\lambda \sin \omega}$$

so that:

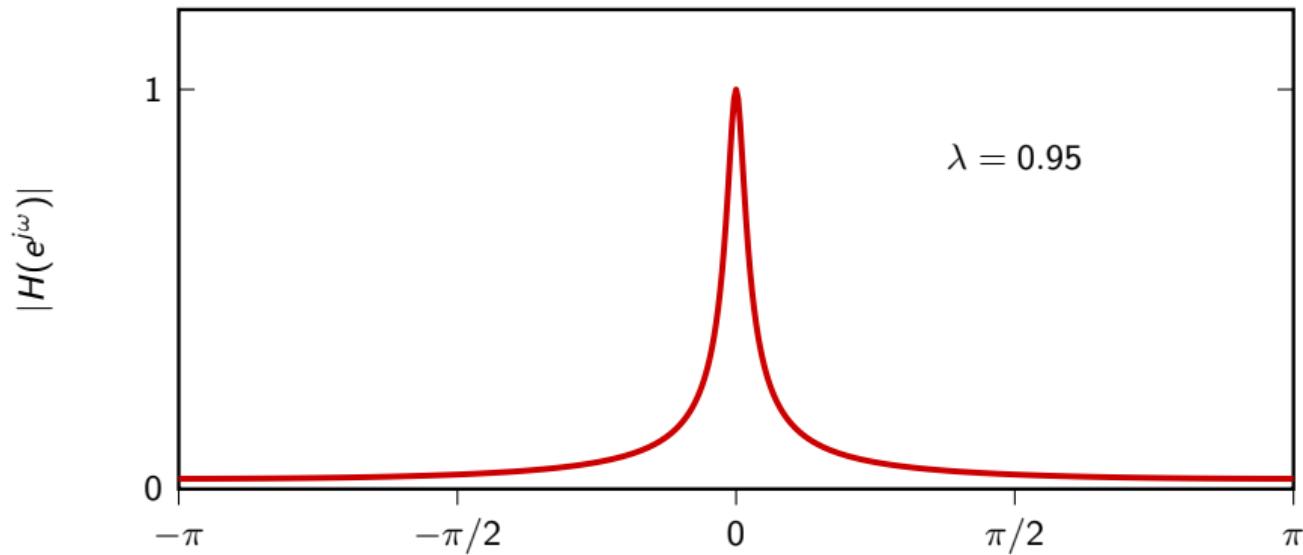
$$|H(e^{j\omega})|^2 = \frac{(1 - \lambda)^2}{1 - 2\lambda \cos \omega + \lambda^2}$$

$$\angle H(e^{j\omega}) = \tan^{-1} \left[\frac{\lambda \sin \omega}{1 - \lambda \cos \omega} \right]$$

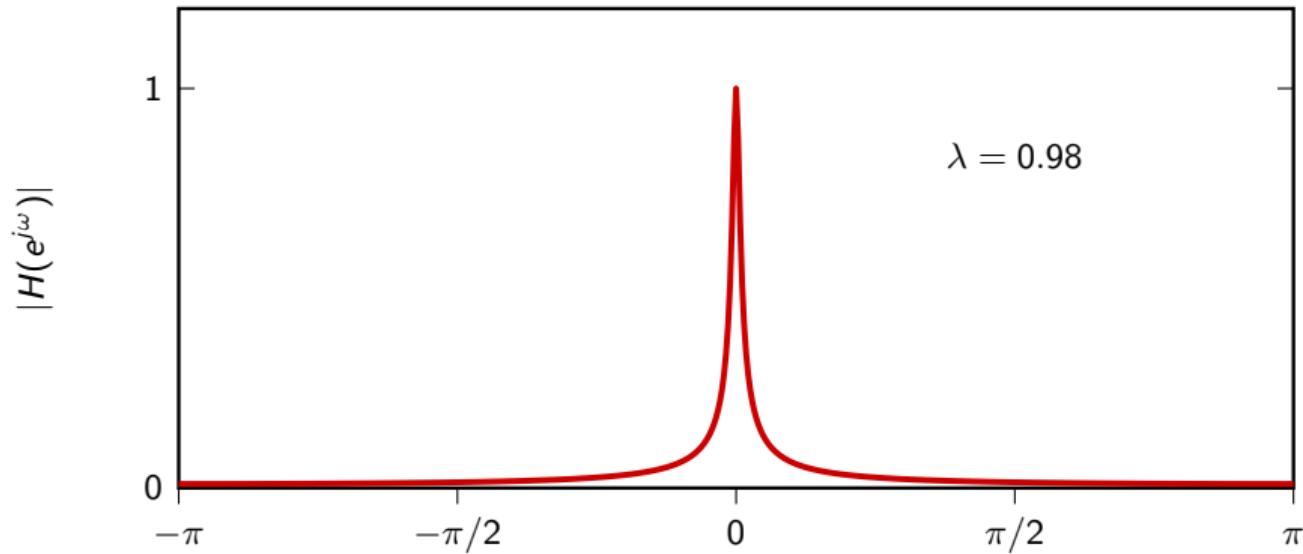
Leaky integrator, magnitude response



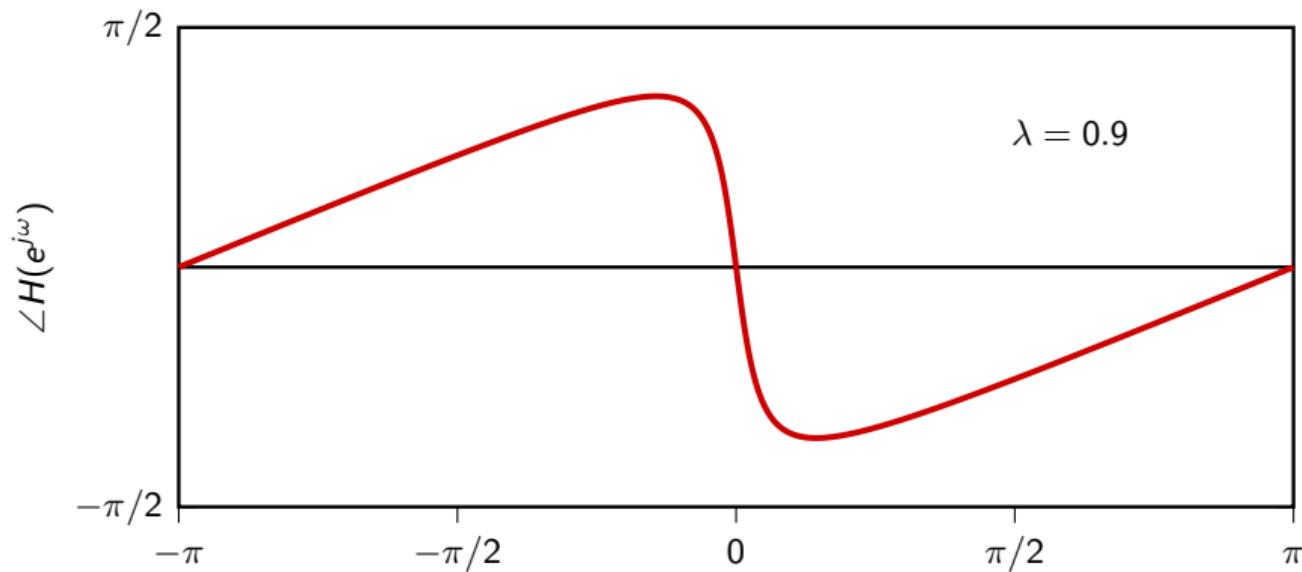
Leaky integrator, magnitude response



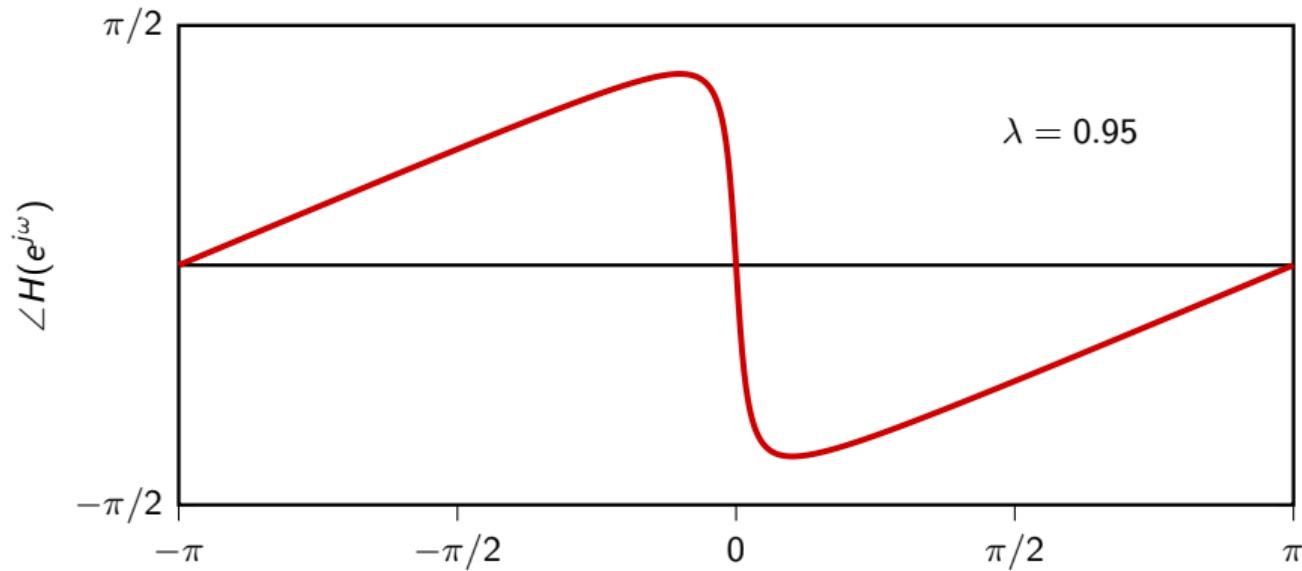
Leaky integrator, magnitude response



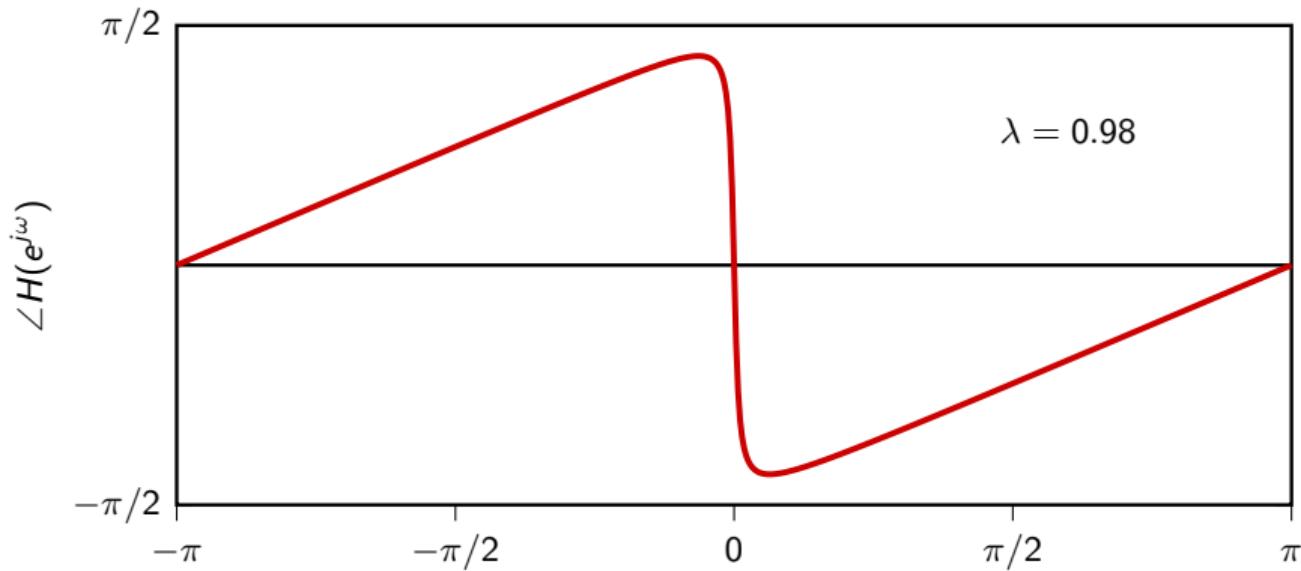
Leaky integrator, phase response



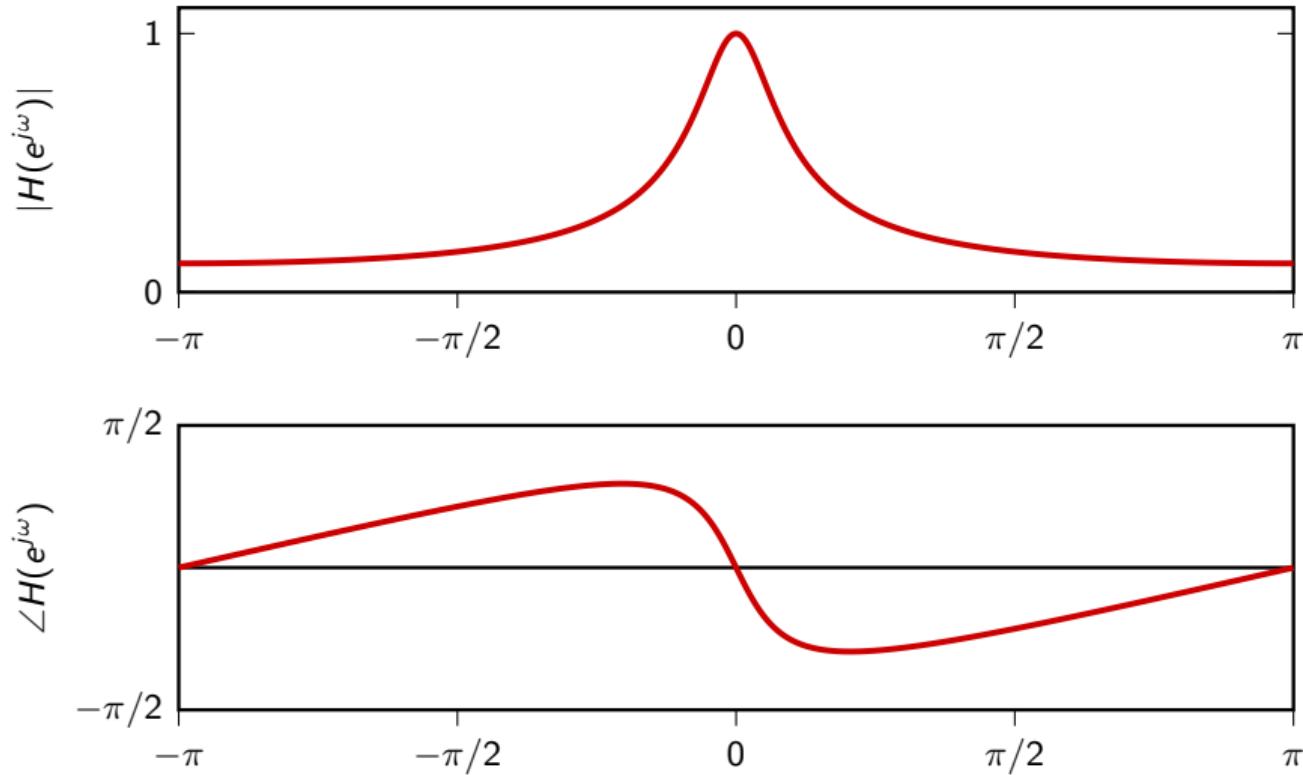
Leaky integrator, phase response



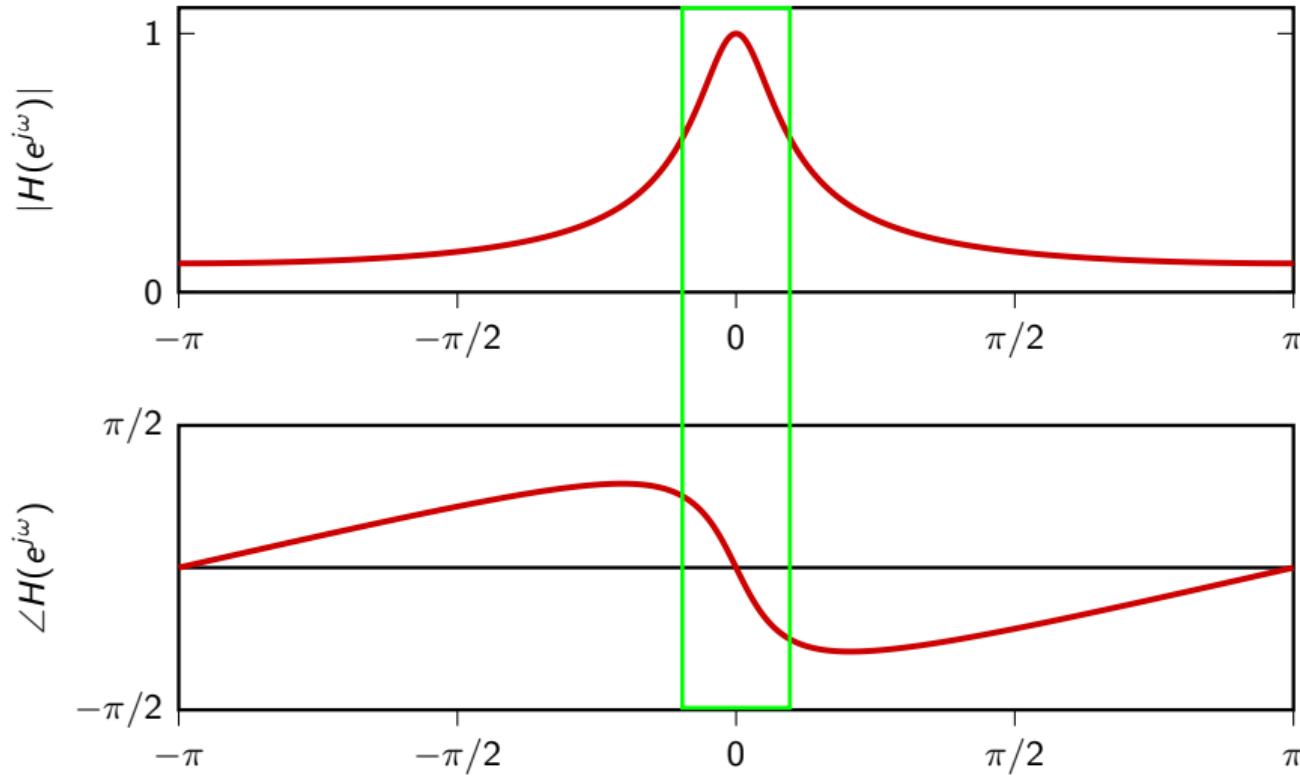
Leaky integrator, phase response



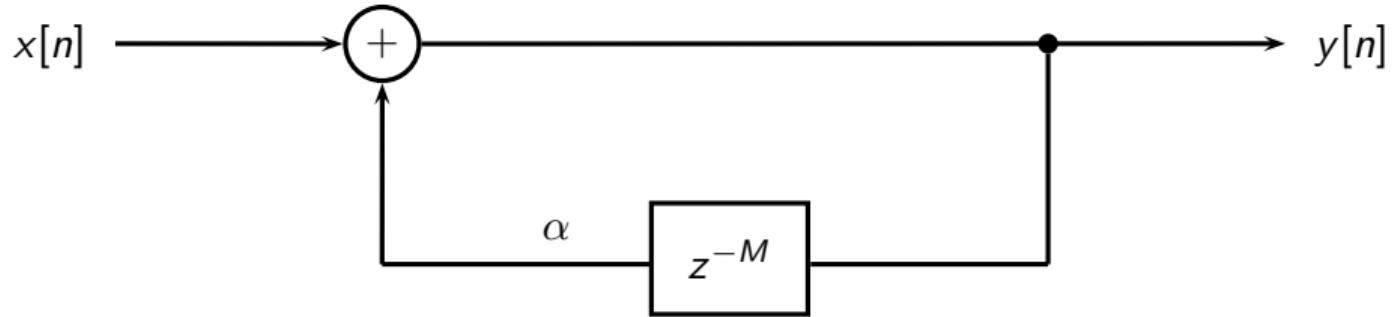
Phase is sufficiently linear where it matters



Phase is sufficiently linear where it matters



Karplus-Strong revisited, again!



$$y[n] = \alpha y[n - M] + x[n]$$

Karplus-Strong revisited

$$y[n] = \bar{x}[0], \bar{x}[1], \dots, \bar{x}[M-1], \alpha\bar{x}[0], \alpha\bar{x}[1], \dots, \alpha\bar{x}[M-1], \alpha^2\bar{x}[0], \alpha^2\bar{x}[1], \dots$$

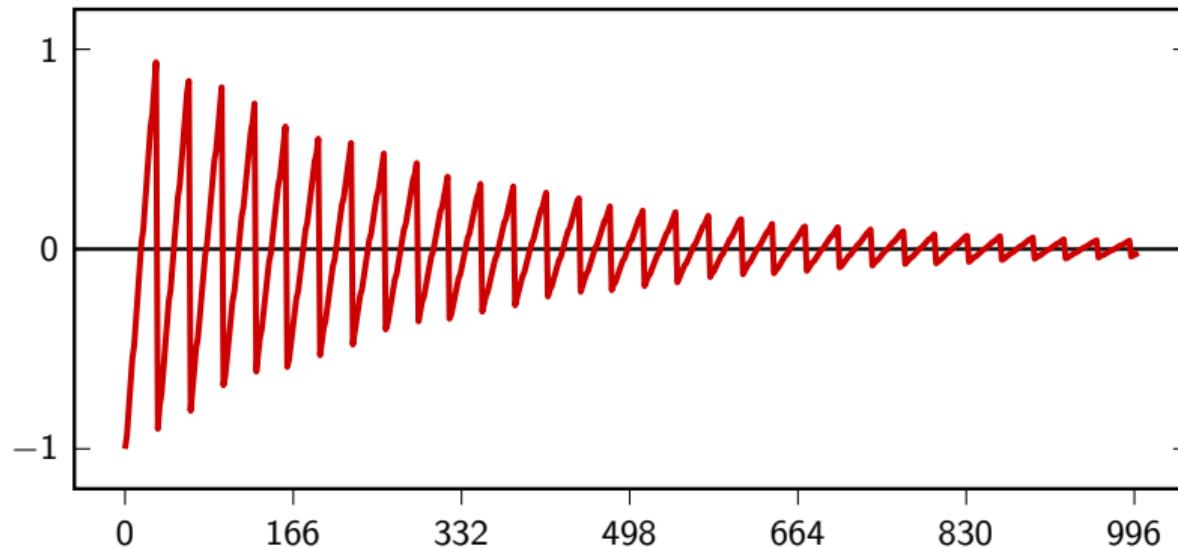
Karplus-Strong revisited

$$y[n] = \bar{x}[0], \bar{x}[1], \dots, \bar{x}[M-1], \alpha\bar{x}[0], \alpha\bar{x}[1], \dots, \alpha\bar{x}[M-1], \alpha^2\bar{x}[0], \alpha^2\bar{x}[1], \dots$$

1st period 2nd period ...

KS revisited: sawtooth signal

$$y[n] = \alpha^{\lfloor n/M \rfloor} \bar{x}[n \bmod M] u[n]$$



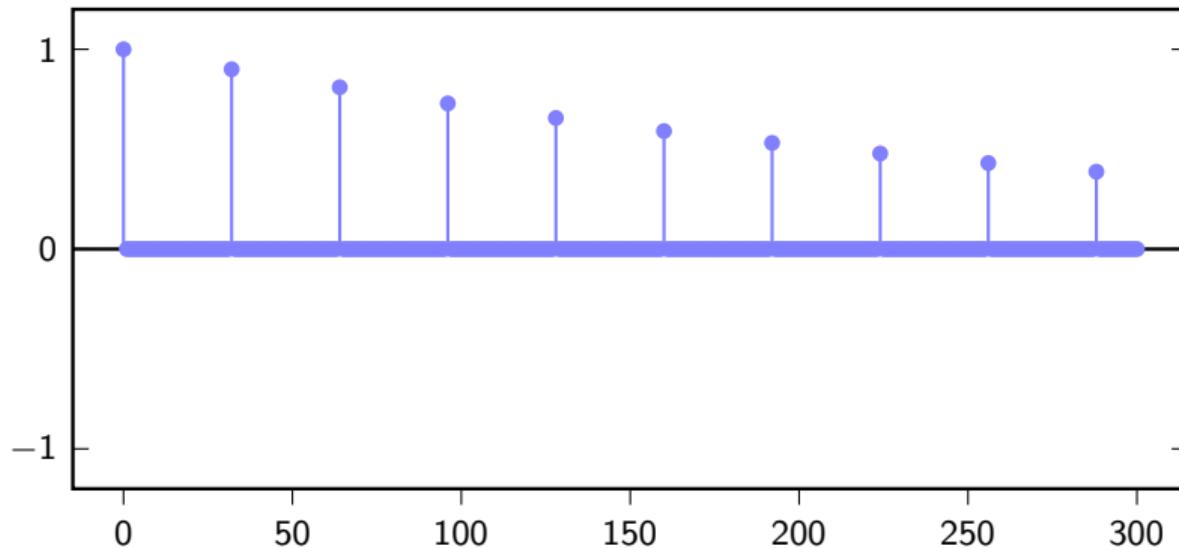
DTFT of KS signal, using the convolution theorem

key observation:

$$y[n] = \bar{x}[n] * w[n], \quad w[n] = \begin{cases} \alpha^k & \text{for } n = kM \\ 0 & \text{otherwise} \end{cases}$$

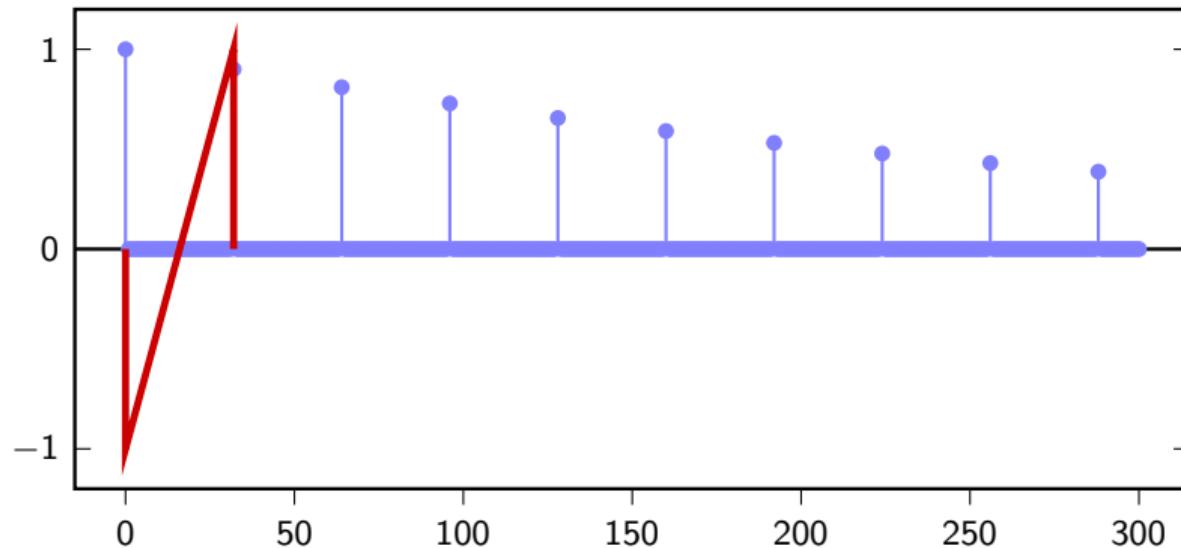
KS revisited

$$y[n] = \alpha^{\lfloor n/M \rfloor} \bar{x}[n \bmod M] u[n]$$



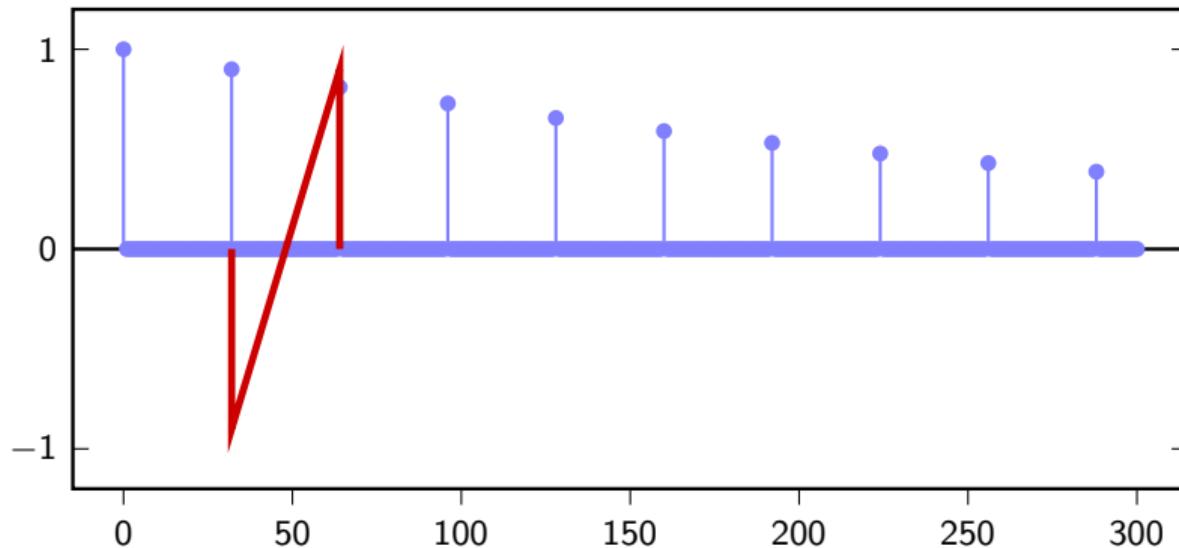
KS revisited

$$y[n] = \alpha^{\lfloor n/M \rfloor} \bar{x}[n \bmod M] u[n]$$



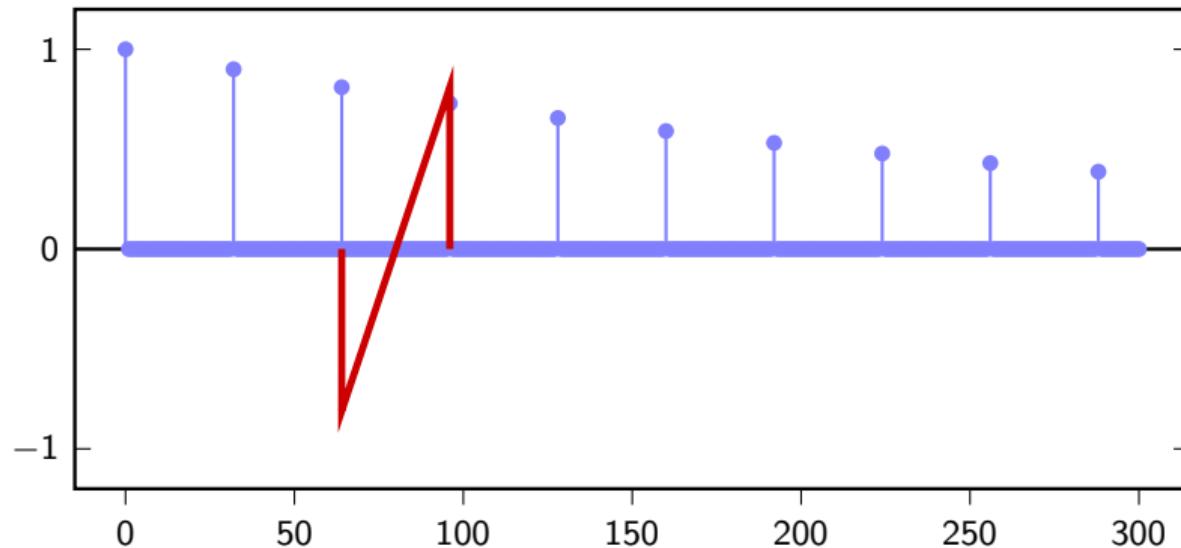
KS revisited

$$y[n] = \alpha^{\lfloor n/M \rfloor} \bar{x}[n \bmod M] u[n]$$



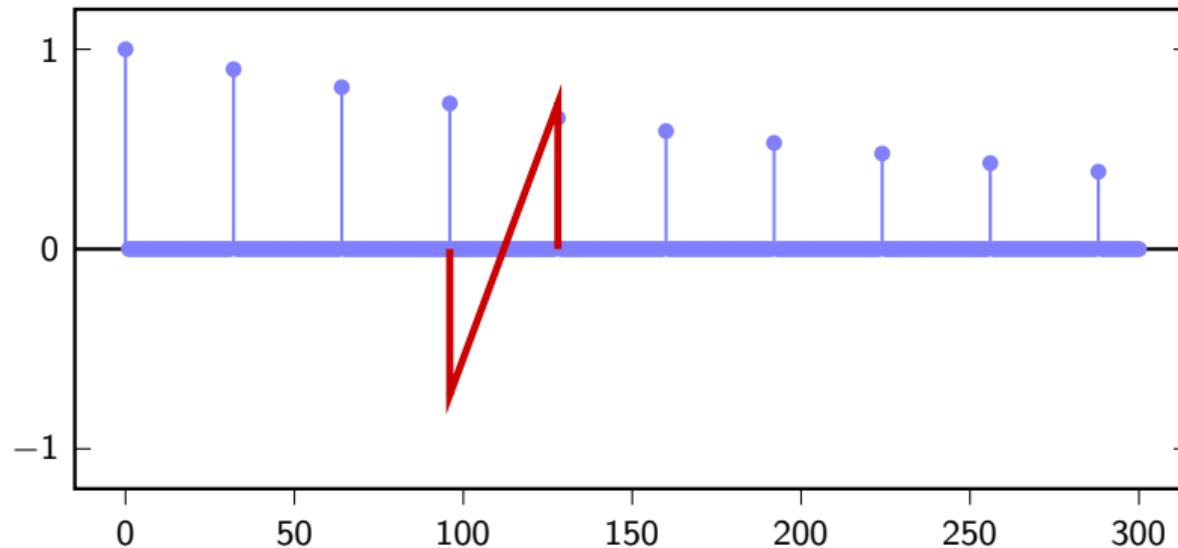
KS revisited

$$y[n] = \alpha^{\lfloor n/M \rfloor} \bar{x}[n \bmod M] u[n]$$



KS revisited

$$y[n] = \alpha^{\lfloor n/M \rfloor} \bar{x}[n \bmod M] u[n]$$



DTFT of KS signal, using the convolution theorem

key observation:

$$y[n] = \bar{x}[n] * w[n], \quad w[n] = \begin{cases} \alpha^k & \text{for } n = kM \\ 0 & \text{otherwise} \end{cases}$$

$$Y(e^{j\omega}) = \bar{X}(e^{j\omega})W(e^{j\omega})$$

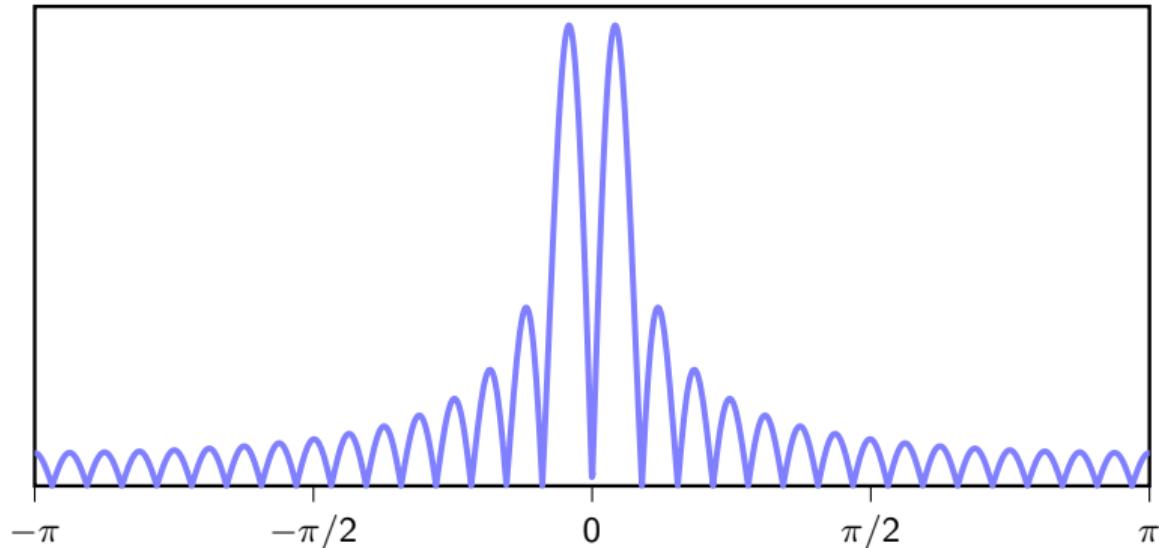
DTFT of KS signal, using the convolution theorem

$$\bar{X}(e^{j\omega}) = e^{-j\omega} \left(\frac{M+1}{M-1} \right) \frac{1 - e^{-j(M-1)\omega}}{(1 - e^{-j\omega})^2} - \frac{1 - e^{-j(M+1)\omega}}{(1 - e^{-j\omega})^2}$$

$$W(e^{j\omega}) = \frac{1}{1 - \alpha e^{-j\omega M}}$$

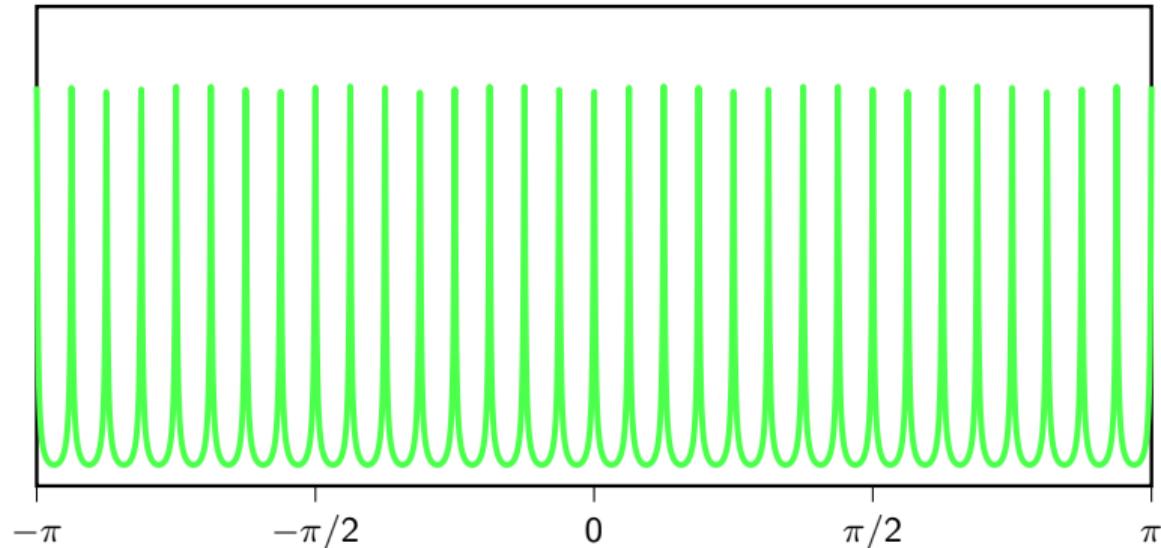
DTFT of KS

$$|\bar{X}(e^{j\omega})|$$



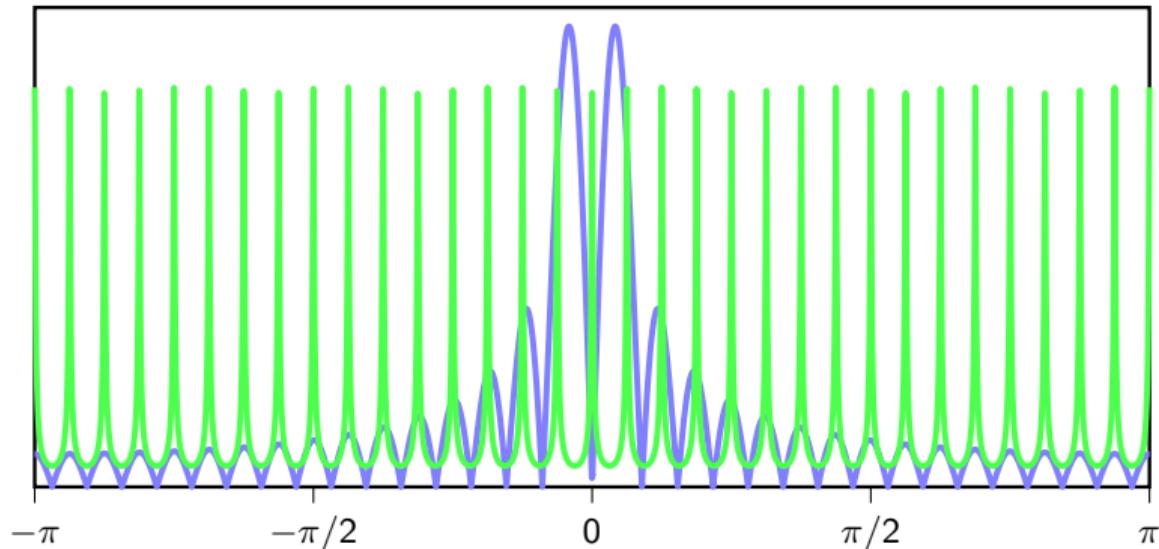
DTFT of KS

$$|W(e^{j\omega})|$$



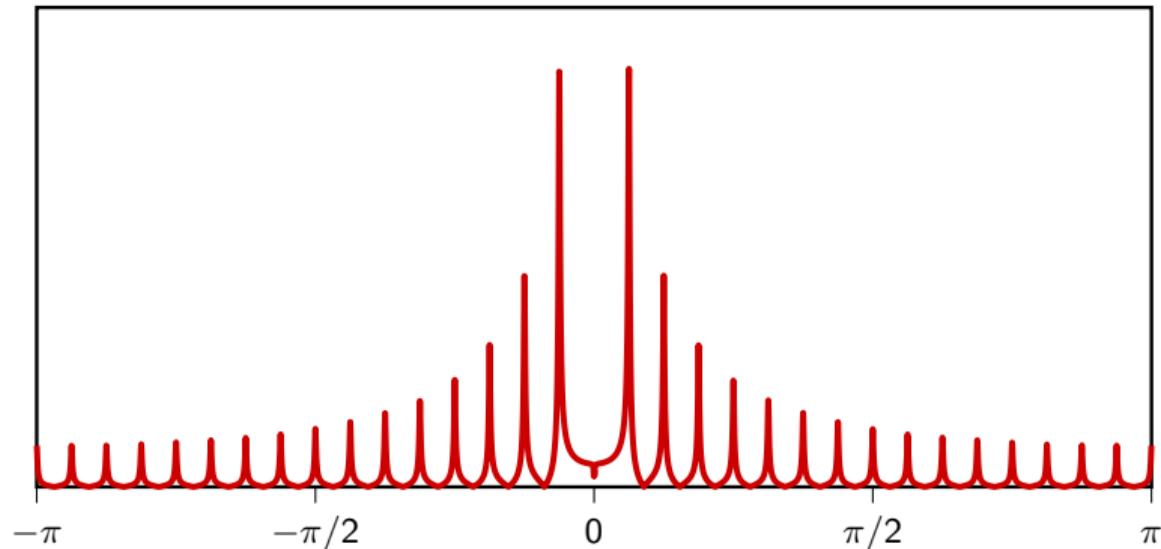
DTFT of KS

$$|Y(e^{j\omega})|$$



DTFT of KS

$$|Y(e^{j\omega})|$$



COM303: Digital Signal Processing

Lecture 10: Ideal Filters and Approximations

Overview:

- ▶ ideal filters
- ▶ approximating ideal filters

ideal filters

Overview:

- ▶ Filter classification in the frequency domain
- ▶ Ideal filters
- ▶ Demodulation revisited

Overview:

- ▶ Filter classification in the frequency domain
- ▶ Ideal filters
- ▶ Demodulation revisited

Overview:

- ▶ Filter classification in the frequency domain
- ▶ Ideal filters
- ▶ Demodulation revisited

Filter types according to magnitude response

- ▶ Lowpass
- ▶ Highpass
- ▶ Bandpass
- ▶ Allpass

Moving Average and Leaky Integrator are lowpass filters

Filter types according to magnitude response

- ▶ Lowpass
- ▶ Highpass
- ▶ Bandpass
- ▶ Allpass

Moving Average and Leaky Integrator are lowpass filters

Filter types according to magnitude response

- ▶ Lowpass
- ▶ Highpass
- ▶ Bandpass
- ▶ Allpass

Moving Average and Leaky Integrator are lowpass filters

Filter types according to magnitude response

- ▶ Lowpass
- ▶ Highpass
- ▶ Bandpass
- ▶ Allpass

Moving Average and Leaky Integrator are lowpass filters

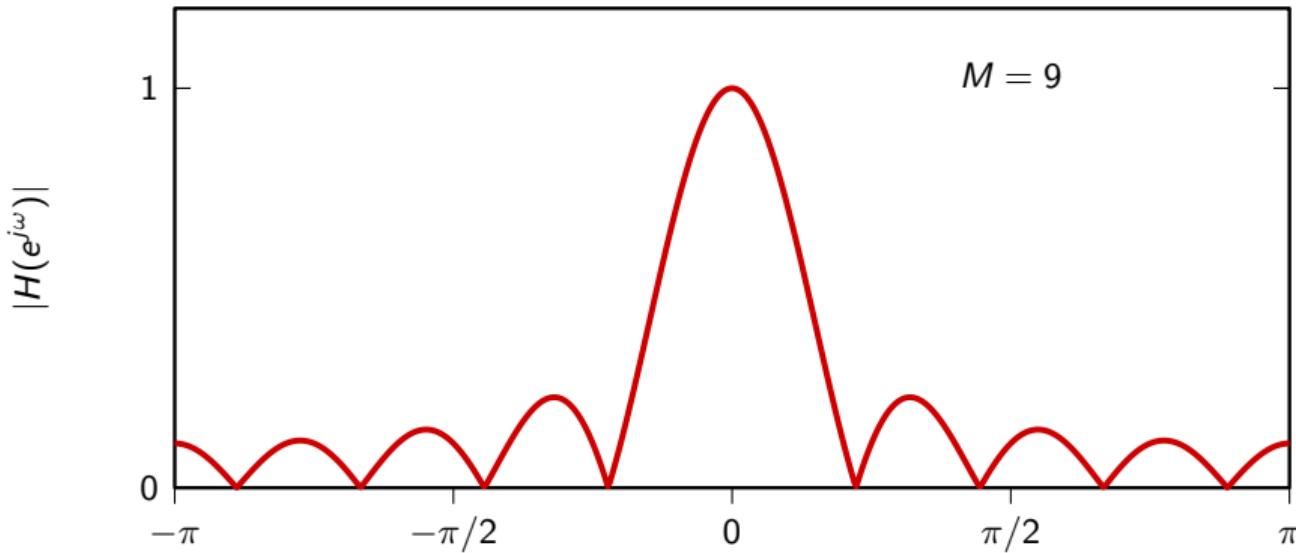
Filter types according to magnitude response

- ▶ Lowpass
- ▶ Highpass
- ▶ Bandpass
- ▶ Allpass

Moving Average and Leaky Integrator are lowpass filters

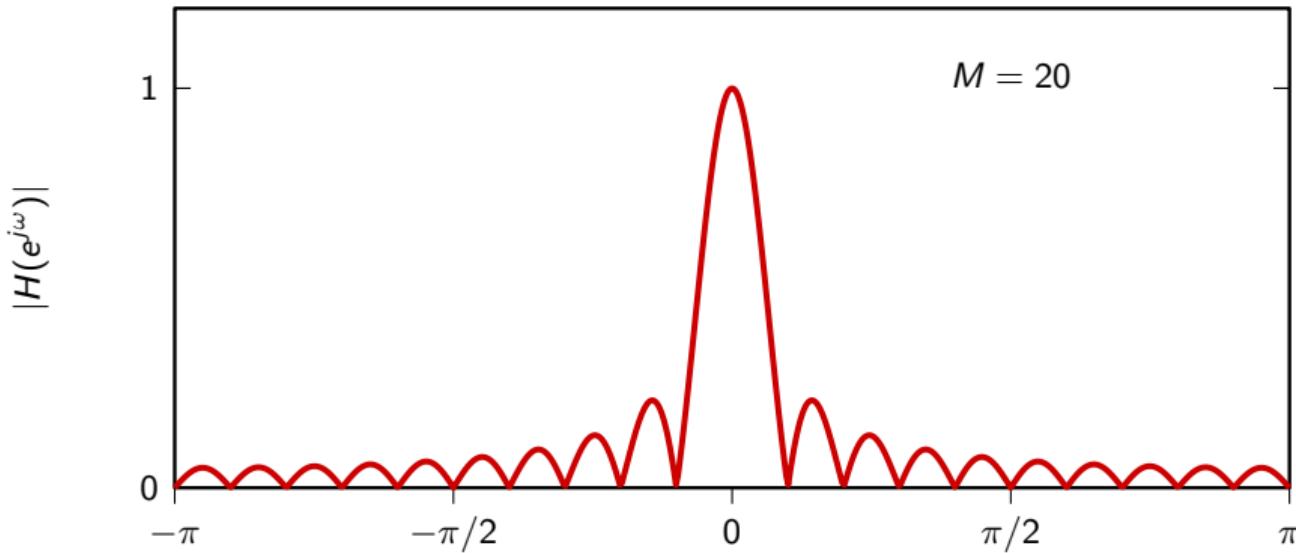
Moving Average, magnitude response

$$|H(e^{j\omega})| = \frac{1}{M} \left| \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} \right|$$



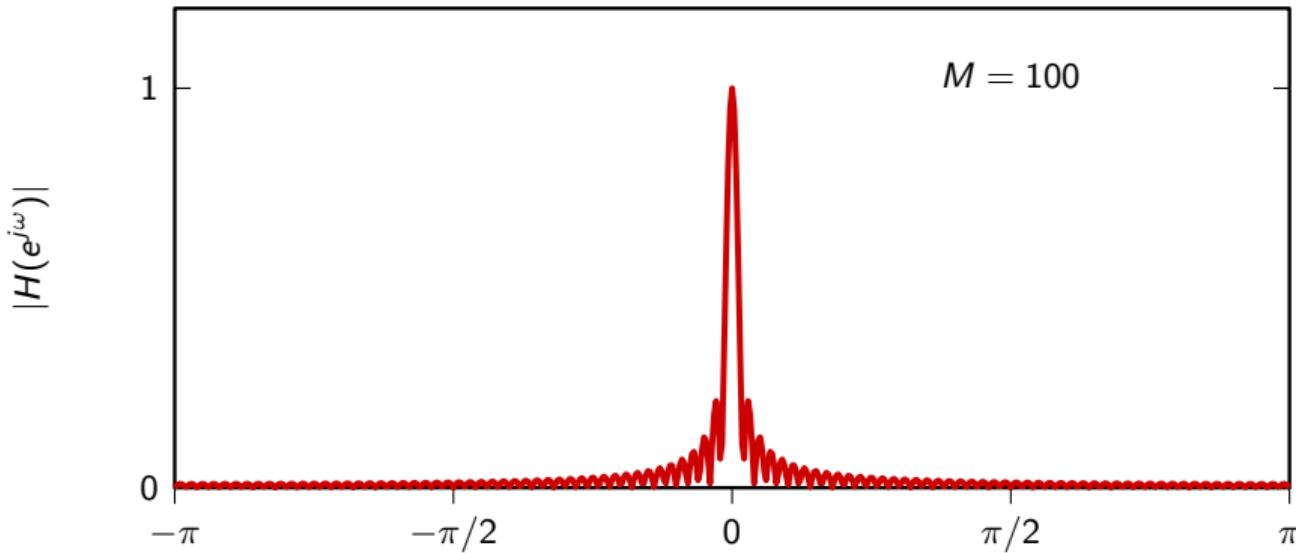
Moving Average, magnitude response

$$|H(e^{j\omega})| = \frac{1}{M} \left| \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} \right|$$



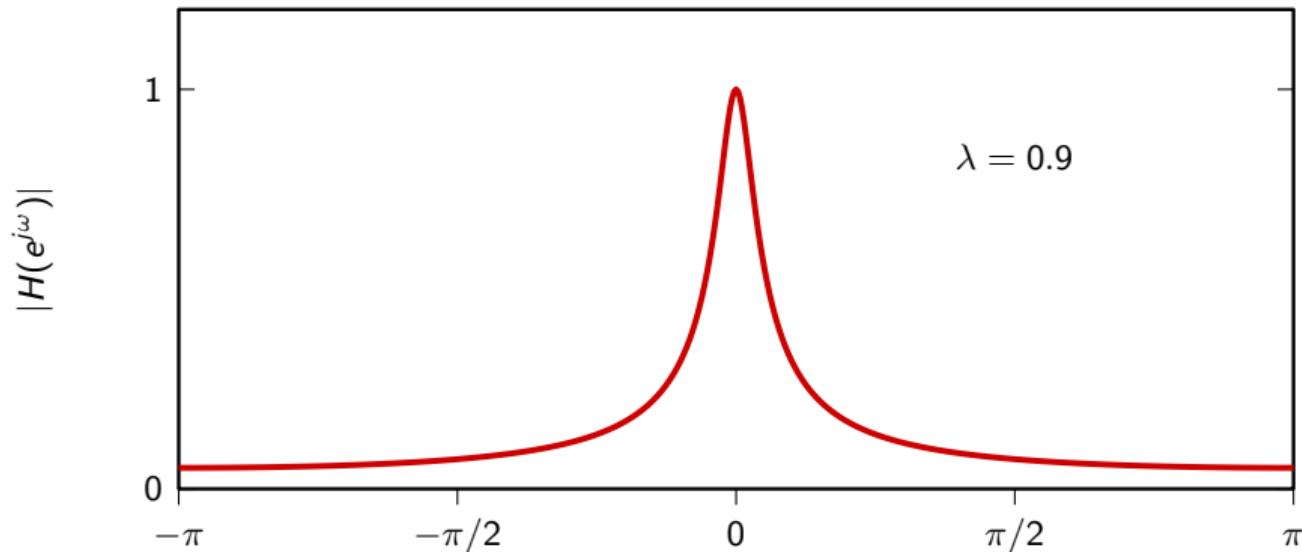
Moving Average, magnitude response

$$|H(e^{j\omega})| = \frac{1}{M} \left| \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} \right|$$



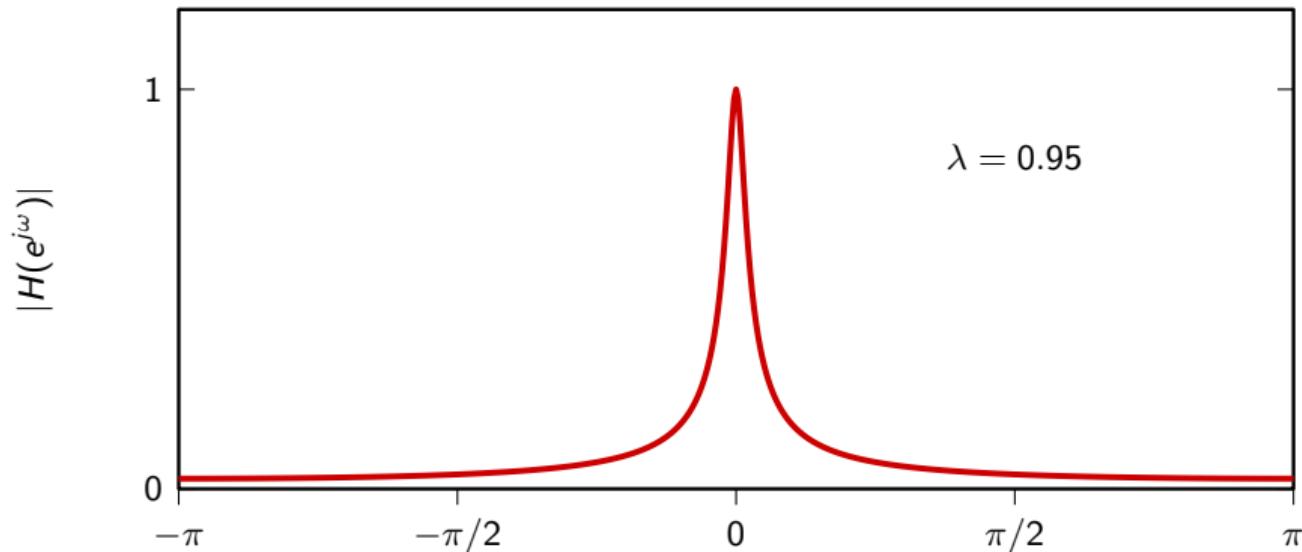
Leaky integrator, magnitude response

$$|H(e^{j\omega})| = \left| \frac{1-\lambda}{1-\lambda e^{-j\omega}} \right|$$



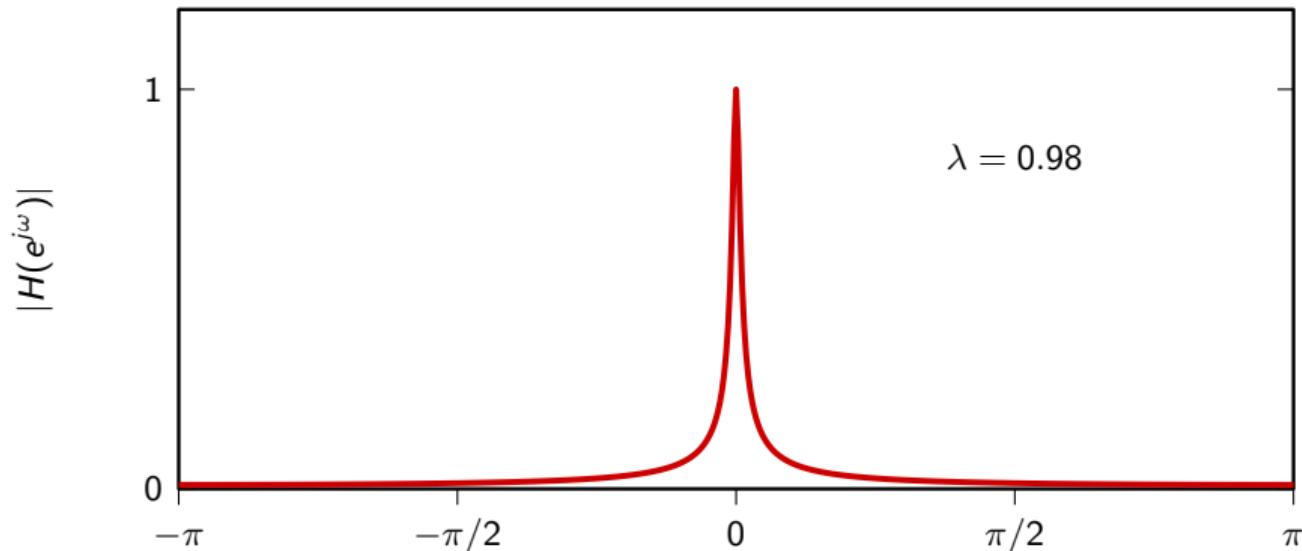
Leaky integrator, magnitude response

$$|H(e^{j\omega})| = \left| \frac{1-\lambda}{1-\lambda e^{-j\omega}} \right|$$



Leaky integrator, magnitude response

$$|H(e^{j\omega})| = \left| \frac{1-\lambda}{1-\lambda e^{-j\omega}} \right|$$



Filter types according to phase response

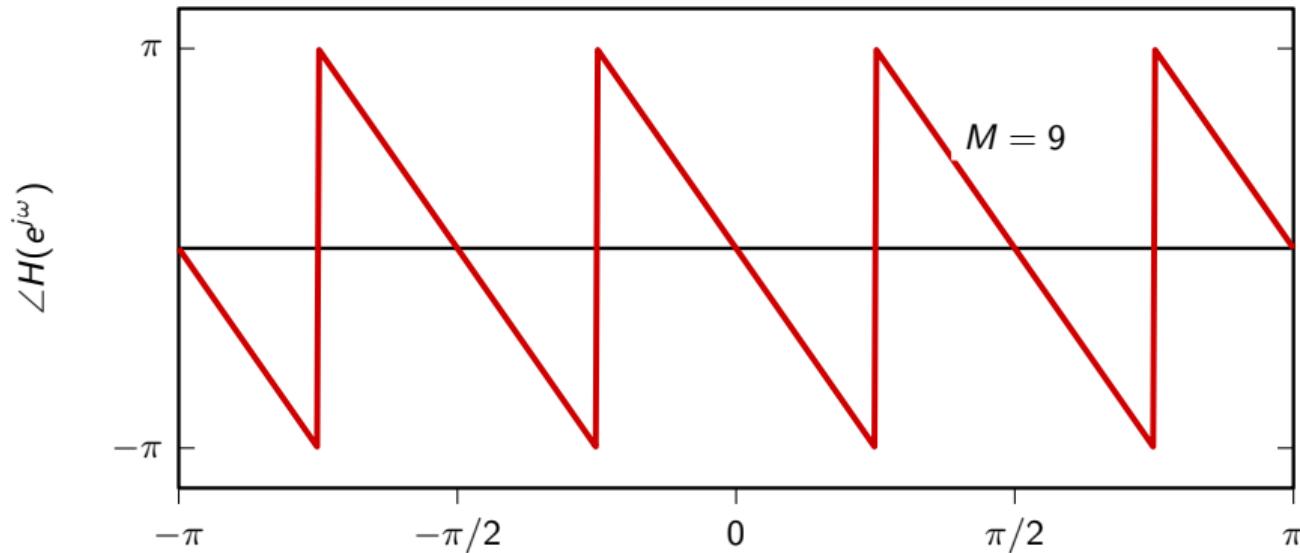
- ▶ Linear phase
- ▶ Nonlinear phase

Filter types according to phase response

- ▶ Linear phase
- ▶ Nonlinear phase

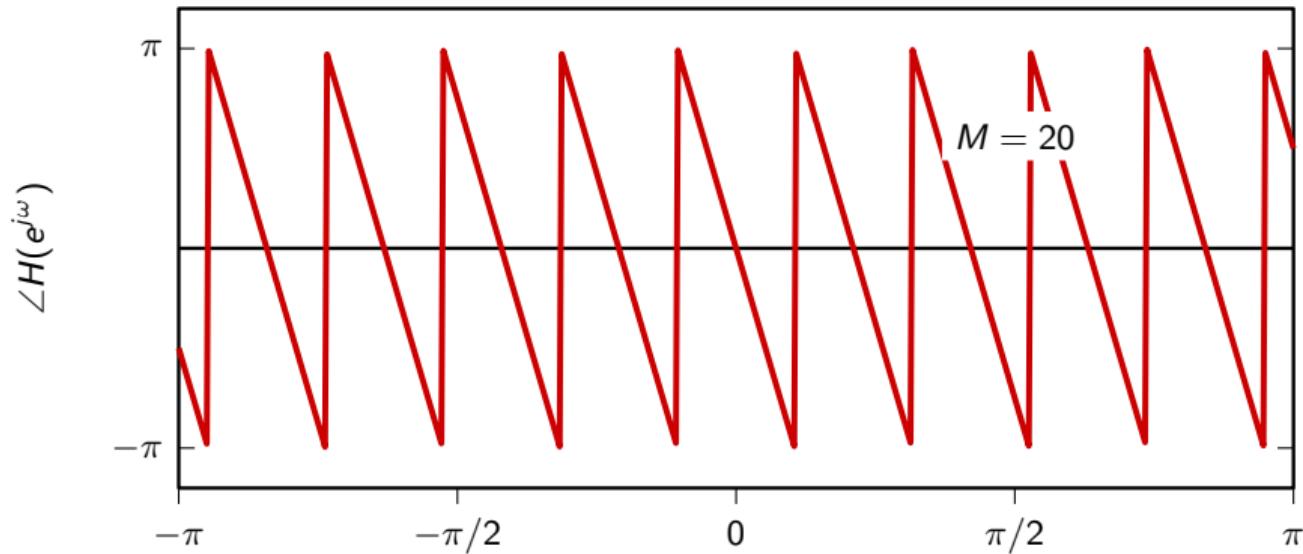
Moving Average, phase response

$$\angle H(e^{j\omega}) = -\frac{M-1}{2}\omega$$



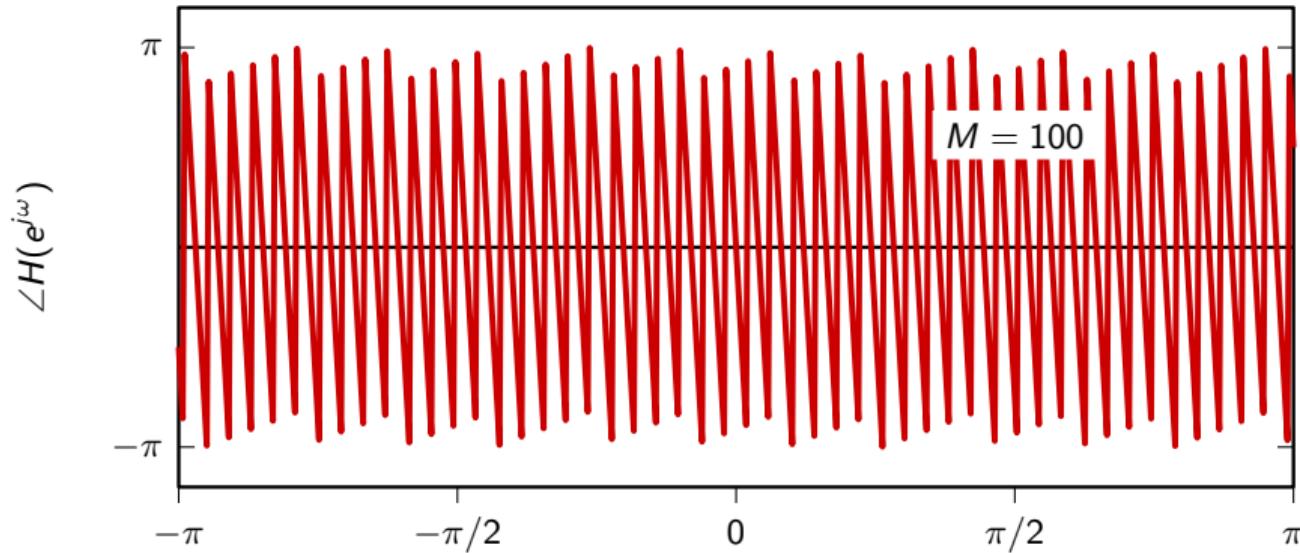
Moving Average, phase response

$$\angle H(e^{j\omega}) = -\frac{M-1}{2}\omega$$



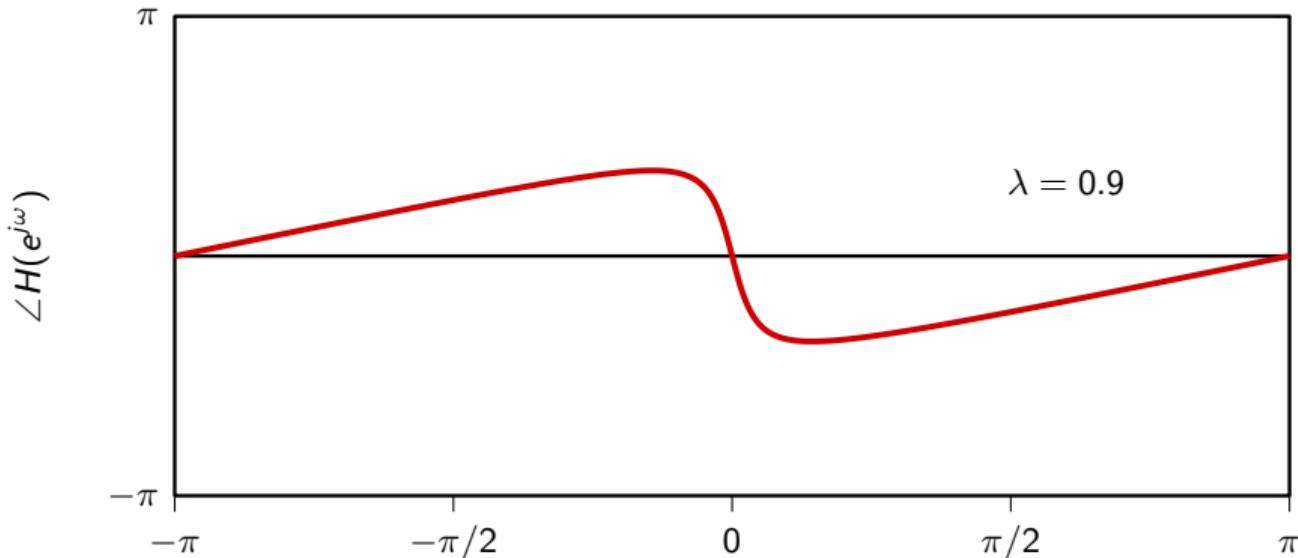
Moving Average, phase response

$$\angle H(e^{j\omega}) = -\frac{M-1}{2}\omega$$



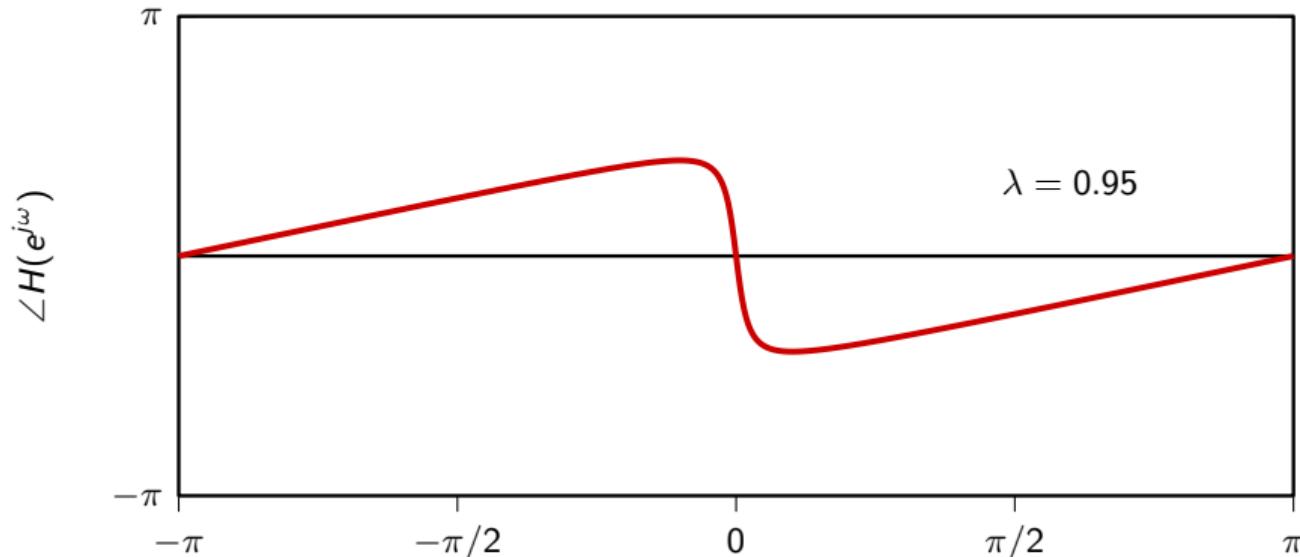
Leaky integrator, phase response

$$\angle H(e^{j\omega}) = \tan^{-1} \left[\frac{\lambda \sin \omega}{1 - \lambda \cos \omega} \right]$$



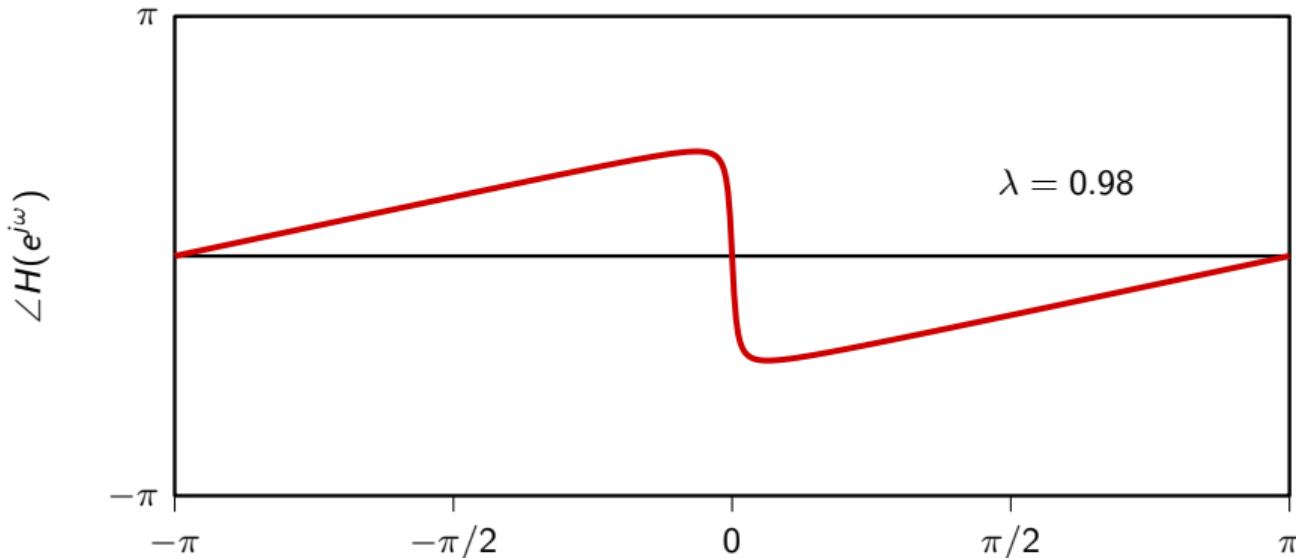
Leaky integrator, phase response

$$\angle H(e^{j\omega}) = \tan^{-1} \left[\frac{\lambda \sin \omega}{1 - \lambda \cos \omega} \right]$$

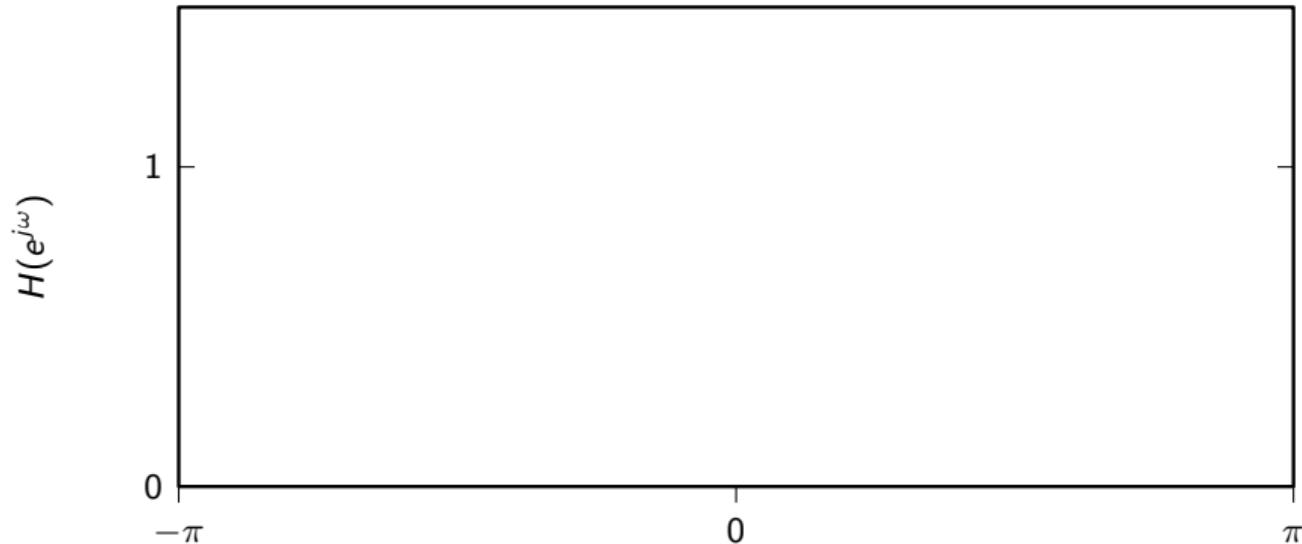


Leaky integrator, phase response

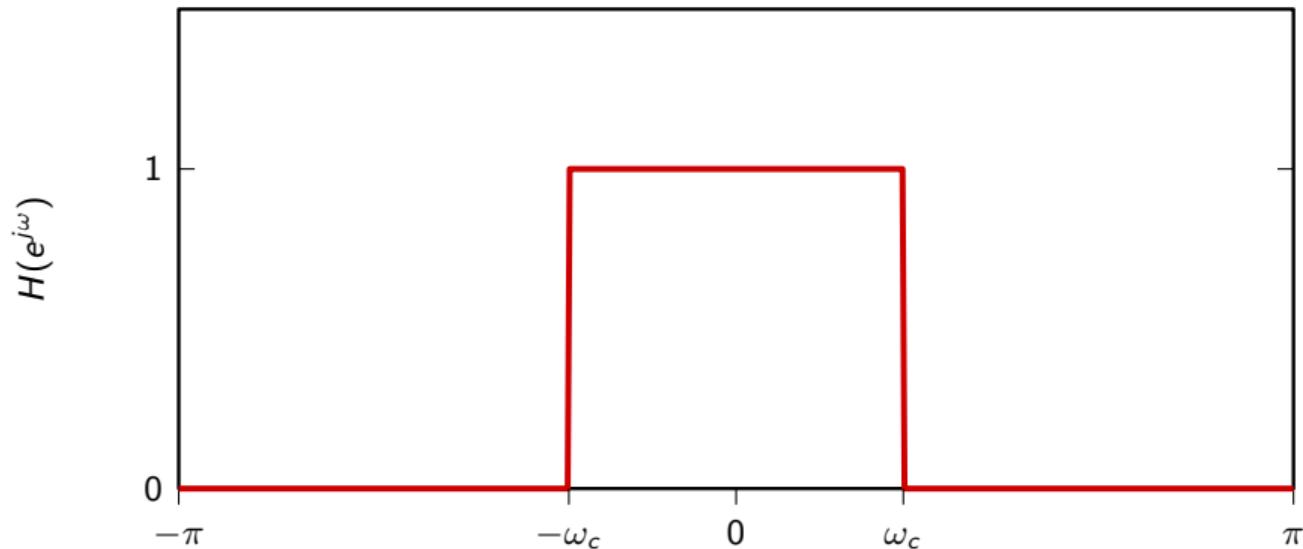
$$\angle H(e^{j\omega}) = \tan^{-1} \left[\frac{\lambda \sin \omega}{1 - \lambda \cos \omega} \right]$$



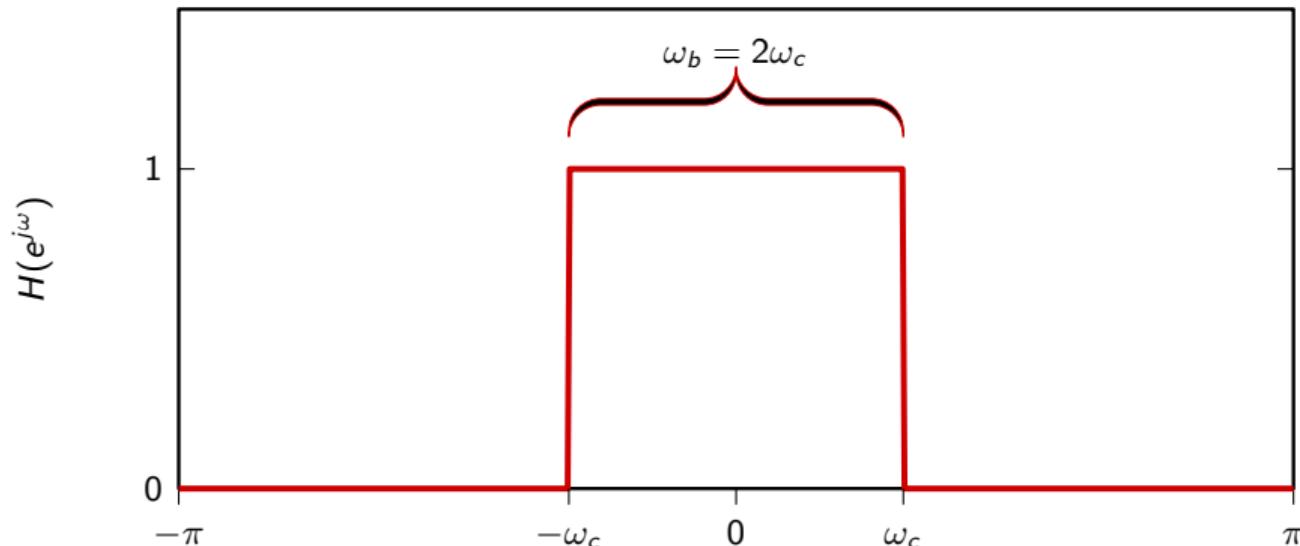
What is the best lowpass we can think of?



What is the best lowpass we can think of?



What is the best lowpass we can think of?



Ideal lowpass filter

$$H(e^{j\omega}) = \begin{cases} 1 & \text{for } |\omega| \leq \omega_c \\ 0 & \text{otherwise} \end{cases} \quad (2\pi\text{-periodicity implicit})$$

- ▶ perfectly flat passband
- ▶ infinite attenuation in stopband
- ▶ zero-phase (no delay)

Ideal lowpass filter

$$H(e^{j\omega}) = \begin{cases} 1 & \text{for } |\omega| \leq \omega_c \\ 0 & \text{otherwise} \end{cases} \quad (2\pi\text{-periodicity implicit})$$

- ▶ perfectly flat passband
- ▶ infinite attenuation in stopband
- ▶ zero-phase (no delay)

Ideal lowpass filter

$$H(e^{j\omega}) = \begin{cases} 1 & \text{for } |\omega| \leq \omega_c \\ 0 & \text{otherwise} \end{cases} \quad (2\pi\text{-periodicity implicit})$$

- ▶ perfectly flat passband
- ▶ infinite attenuation in stopband
- ▶ zero-phase (no delay)

Ideal lowpass filter: impulse response

$$\begin{aligned} h[n] &= \text{IDTFT} \{ H(e^{j\omega}) \} \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega}) e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{j\omega n} d\omega \\ &= \frac{1}{\pi n} \frac{e^{j\omega_c n} - e^{-j\omega_c n}}{2j} \\ &= \frac{\sin \omega_c n}{\pi n} \end{aligned}$$

Ideal lowpass filter: impulse response

$$\begin{aligned} h[n] &= \text{IDTFT} \{ H(e^{j\omega}) \} \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega}) e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{j\omega n} d\omega \\ &= \frac{1}{\pi n} \frac{e^{j\omega_c n} - e^{-j\omega_c n}}{2j} \\ &= \frac{\sin \omega_c n}{\pi n} \end{aligned}$$

Ideal lowpass filter: impulse response

$$\begin{aligned} h[n] &= \text{IDTFT} \{ H(e^{j\omega}) \} \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega}) e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{j\omega n} d\omega \\ &= \frac{1}{\pi n} \frac{e^{j\omega_c n} - e^{-j\omega_c n}}{2j} \\ &= \frac{\sin \omega_c n}{\pi n} \end{aligned}$$

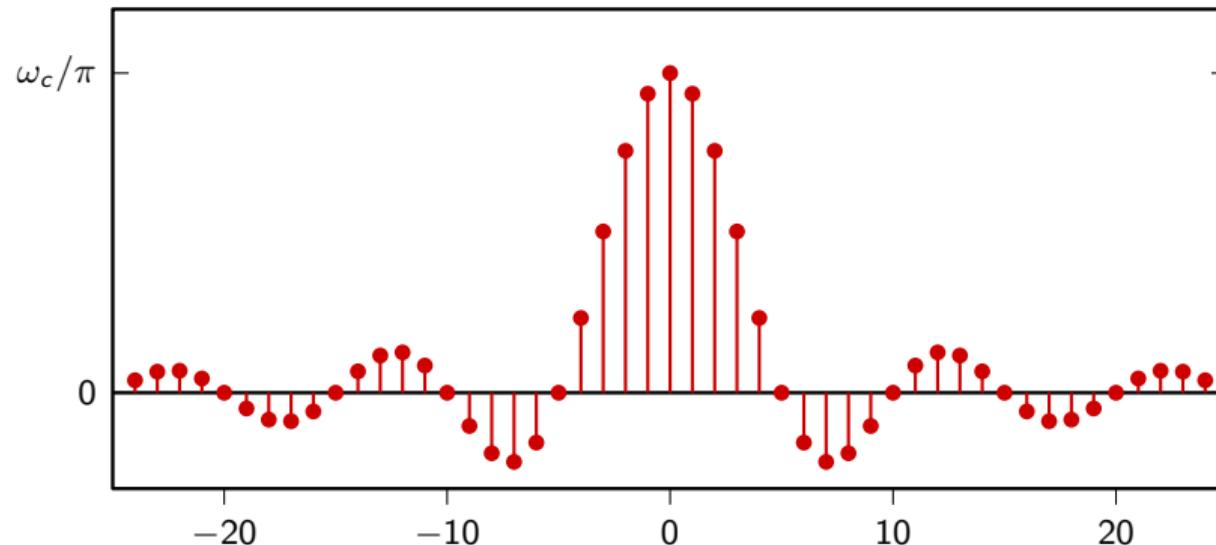
Ideal lowpass filter: impulse response

$$\begin{aligned} h[n] &= \text{IDTFT} \{ H(e^{j\omega}) \} \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega}) e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{j\omega n} d\omega \\ &= \frac{1}{\pi n} \frac{e^{j\omega_c n} - e^{-j\omega_c n}}{2j} \\ &= \frac{\sin \omega_c n}{\pi n} \end{aligned}$$

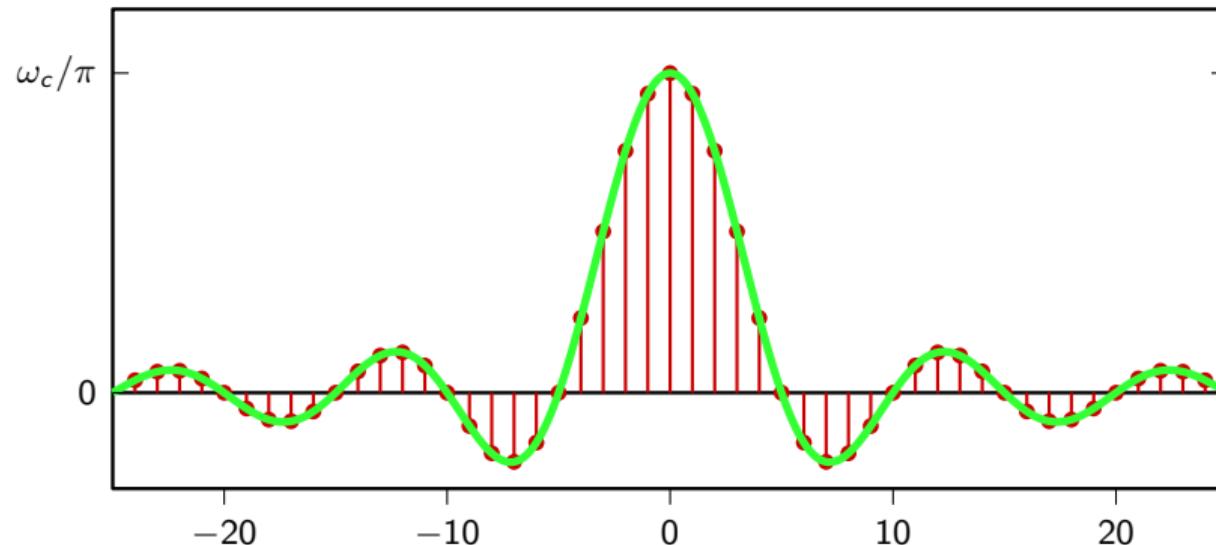
Ideal lowpass filter: impulse response

$$\begin{aligned} h[n] &= \text{IDTFT} \{ H(e^{j\omega}) \} \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega}) e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{j\omega n} d\omega \\ &= \frac{1}{\pi n} \frac{e^{j\omega_c n} - e^{-j\omega_c n}}{2j} \\ &= \frac{\sin \omega_c n}{\pi n} \end{aligned}$$

Ideal lowpass filter: impulse response



Ideal lowpass filter: impulse response



Ladies and Gentlemen: the Sinc and the Rect!

The sinc-rect pair:

$$\text{rect}(x) = \begin{cases} 1 & |x| \leq 1/2 \\ 0 & |x| > 1/2 \end{cases}$$

$$\text{sinc}(x) = \begin{cases} \frac{\sin(\pi x)}{\pi x} & x \neq 0 \\ 1 & x = 0 \end{cases}$$

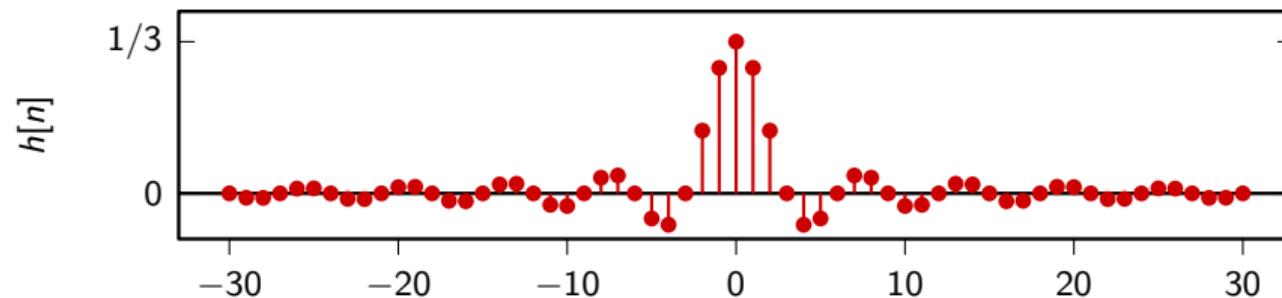
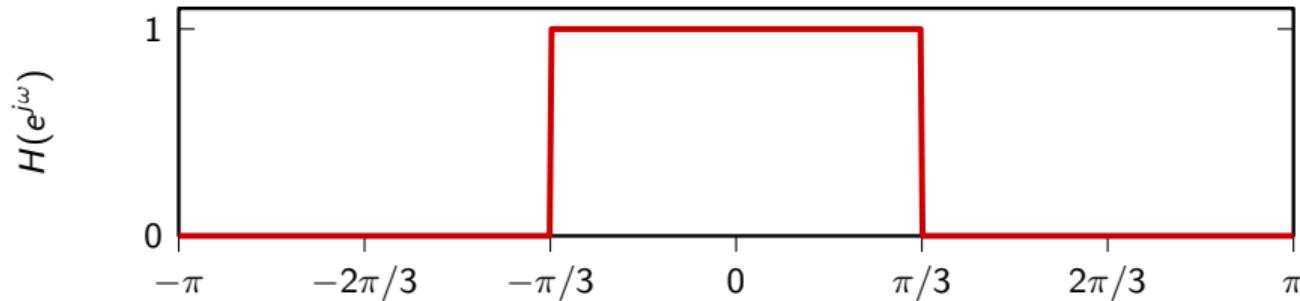
(note that $\text{sinc}(x) = 0$ when x is a nonzero integer)

The ideal lowpass in canonical form

$$\text{rect}\left(\frac{\omega}{2\omega_c}\right) \xleftrightarrow{\text{DTFT}} \frac{\omega_c}{\pi} \text{sinc}\left(\frac{\omega_c}{\pi}n\right)$$

Example

$$\omega_c = \pi/3: H(e^{j\omega}) = \text{rect}(3\omega/2\pi), h[n] = (1/3)\text{sinc}(n/3)$$



The bad news

- ▶ impulse response is infinite support, two-sided
 - ⇒ cannot compute the output in a finite amount of time
 - ⇒ that's why it's called "ideal"
- ▶ impulse response decays slowly in time
 - ⇒ we need a lot of samples for a good approximation

The bad news

- ▶ impulse response is infinite support, two-sided
 - ⇒ cannot compute the output in a finite amount of time
 - ⇒ that's why it's called "ideal"
- ▶ impulse response decays slowly in time
 - ⇒ we need a lot of samples for a good approximation

Little-known fact

- ▶ the sinc is not absolutely summable
- ▶ the ideal lowpass is not BIBO stable!
- ▶ example for $\omega_c = \pi/3$: $h[n] = (1/3) \operatorname{sinc}(n/3)$
- ▶ take $x[n] = \operatorname{sign}\{\operatorname{sinc}(-n/3)\}$ and

$$y[0] = (x * h)[0] = \frac{1}{3} \sum_{k=-\infty}^{\infty} |\operatorname{sinc}(k/3)| = \infty$$

Little-known fact

- ▶ the sinc is not absolutely summable
- ▶ the ideal lowpass is not BIBO stable!
- ▶ example for $\omega_c = \pi/3$: $h[n] = (1/3) \operatorname{sinc}(n/3)$
- ▶ take $x[n] = \operatorname{sign}\{\operatorname{sinc}(-n/3)\}$ and

$$y[0] = (x * h)[0] = \frac{1}{3} \sum_{k=-\infty}^{\infty} |\operatorname{sinc}(k/3)| = \infty$$

Little-known fact

- ▶ the sinc is not absolutely summable
- ▶ the ideal lowpass is not BIBO stable!
- ▶ example for $\omega_c = \pi/3$: $h[n] = (1/3) \operatorname{sinc}(n/3)$
- ▶ take $x[n] = \operatorname{sign}\{\operatorname{sinc}(-n/3)\}$ and

$$y[0] = (x * h)[0] = \frac{1}{3} \sum_{k=-\infty}^{\infty} |\operatorname{sinc}(k/3)| = \infty$$

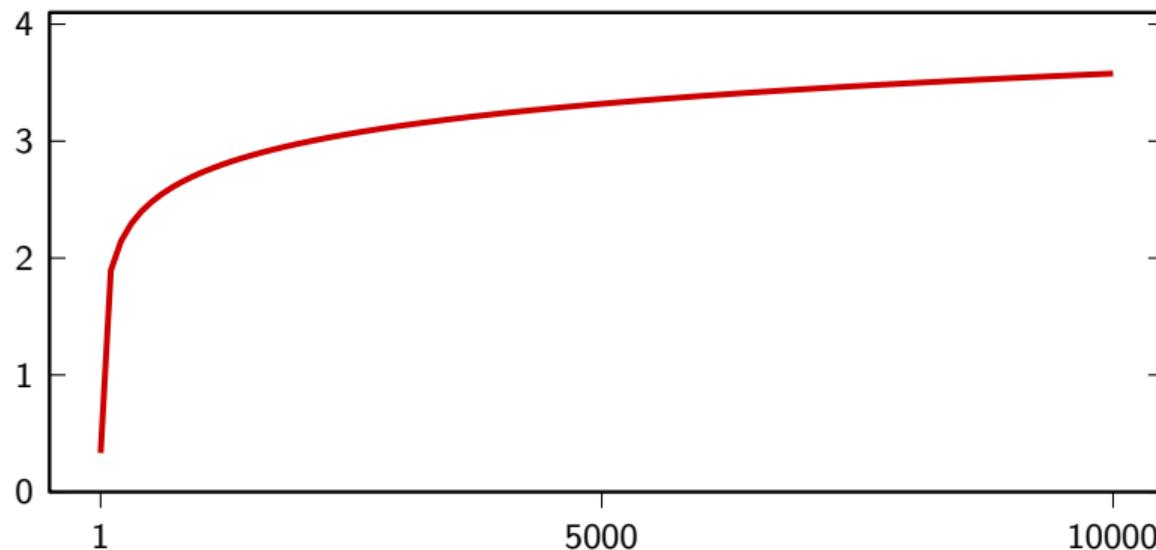
Little-known fact

- ▶ the sinc is not absolutely summable
- ▶ the ideal lowpass is not BIBO stable!
- ▶ example for $\omega_c = \pi/3$: $h[n] = (1/3) \operatorname{sinc}(n/3)$
- ▶ take $x[n] = \operatorname{sign}\{\operatorname{sinc}(-n/3)\}$ and

$$y[0] = (x * h)[0] = \frac{1}{3} \sum_{k=-\infty}^{\infty} |\operatorname{sinc}(k/3)| = \infty$$

Divergence is however very slow...

$$s(n) = (1/3) \sum_{k=-n}^n |\text{sinc}(k/3)|$$



for the mathematically-oriented:

integral criterion for convergence:

$$\sum_{n \in \mathbb{N}} f(n) < \infty \Leftrightarrow \int_1^{\infty} f(t) dt < \infty$$

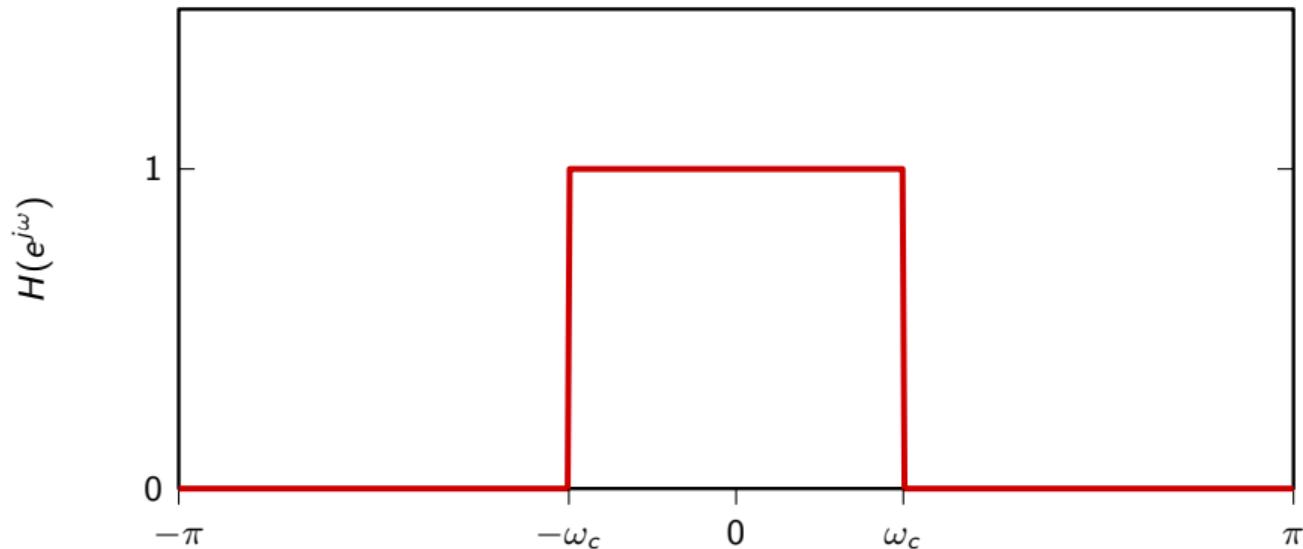
so:

$$\sum_{n \in \mathbb{N}} \left| \frac{\sin n}{n} \right| < \infty \Leftrightarrow \int_1^{\infty} \left| \frac{\sin t}{t} \right| dt < \infty$$

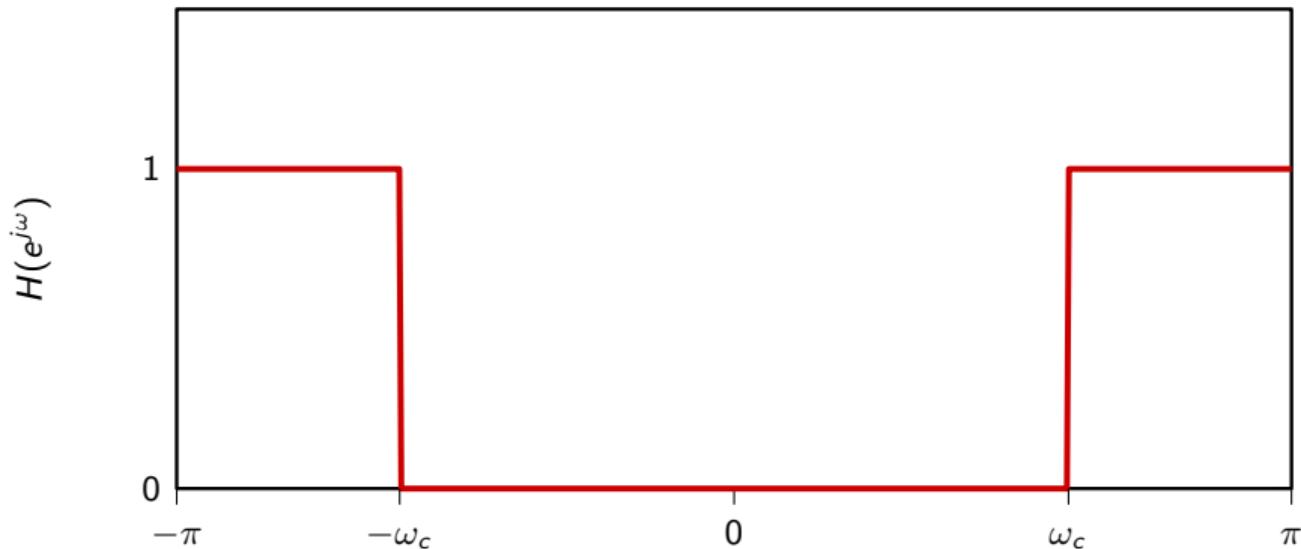
for the mathematically-oriented:

$$\begin{aligned} \int_1^\infty \left| \frac{\sin t}{t} \right| dt &\geq \int_\pi^\infty \left| \frac{\sin t}{t} \right| dt \\ &= \sum_{k=1}^{\infty} \int_{k\pi}^{(k+1)\pi} \left| \frac{\sin t}{t} \right| dt \\ &\geq \sum_{k=1}^{\infty} \int_{k\pi}^{(k+1)\pi} \frac{|\sin t|}{(k+1)\pi} dt \\ &= \frac{1}{\pi} \sum_{k=1}^{\infty} \frac{1}{k+1} \int_{k\pi}^{(k+1)\pi} |\sin t| dt \\ &= \frac{2}{\pi} \sum_{k=1}^{\infty} \frac{1}{k+1} = \infty \end{aligned}$$

From the ideal lowpass...



... to the ideal highpass



Ideal highpass filter

$$H_{hp}(e^{j\omega}) = \begin{cases} 1 & \text{for } \pi \geq |\omega| \geq \omega_c \\ 0 & \text{otherwise} \end{cases} \quad (2\pi\text{-periodicity implicit})$$

$$H_{hp}(e^{j\omega}) = 1 - H_{lp}(e^{j\omega})$$

$$h_{hp}[n] = \delta[n] - \frac{\omega_c}{\pi} \operatorname{sinc}\left(\frac{\omega_c}{\pi} n\right)$$

Ideal highpass filter

$$H_{hp}(e^{j\omega}) = \begin{cases} 1 & \text{for } \pi \geq |\omega| \geq \omega_c \\ 0 & \text{otherwise} \end{cases} \quad (2\pi\text{-periodicity implicit})$$

$$H_{hp}(e^{j\omega}) = 1 - H_{lp}(e^{j\omega})$$

$$h_{hp}[n] = \delta[n] - \frac{\omega_c}{\pi} \operatorname{sinc}\left(\frac{\omega_c}{\pi} n\right)$$

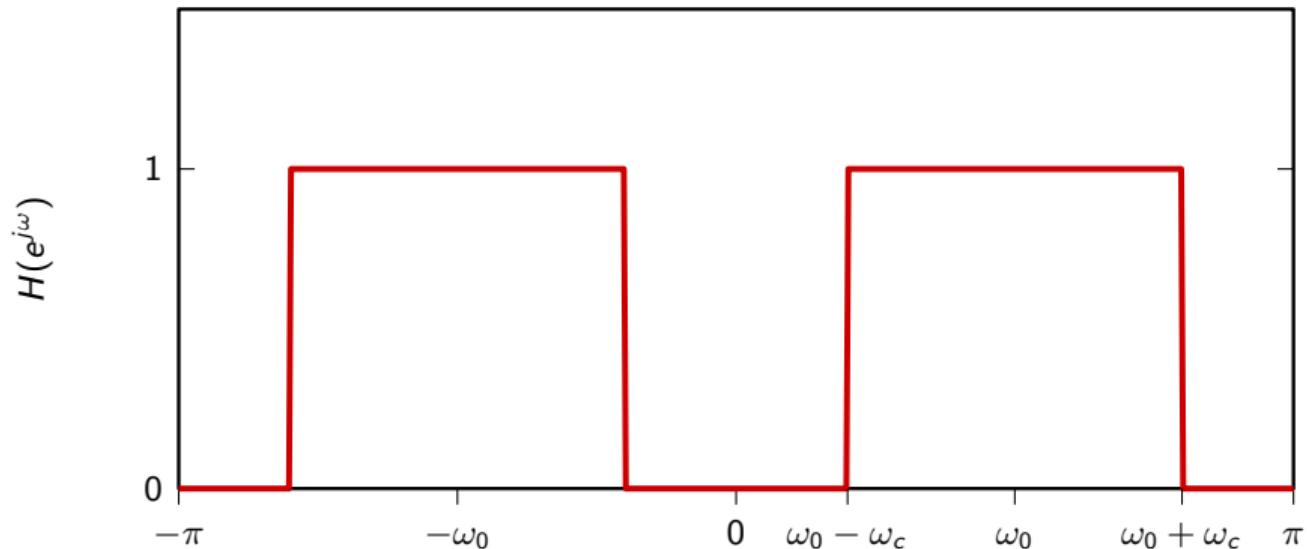
Ideal highpass filter

$$H_{hp}(e^{j\omega}) = \begin{cases} 1 & \text{for } \pi \geq |\omega| \geq \omega_c \\ 0 & \text{otherwise} \end{cases} \quad (2\pi\text{-periodicity implicit})$$

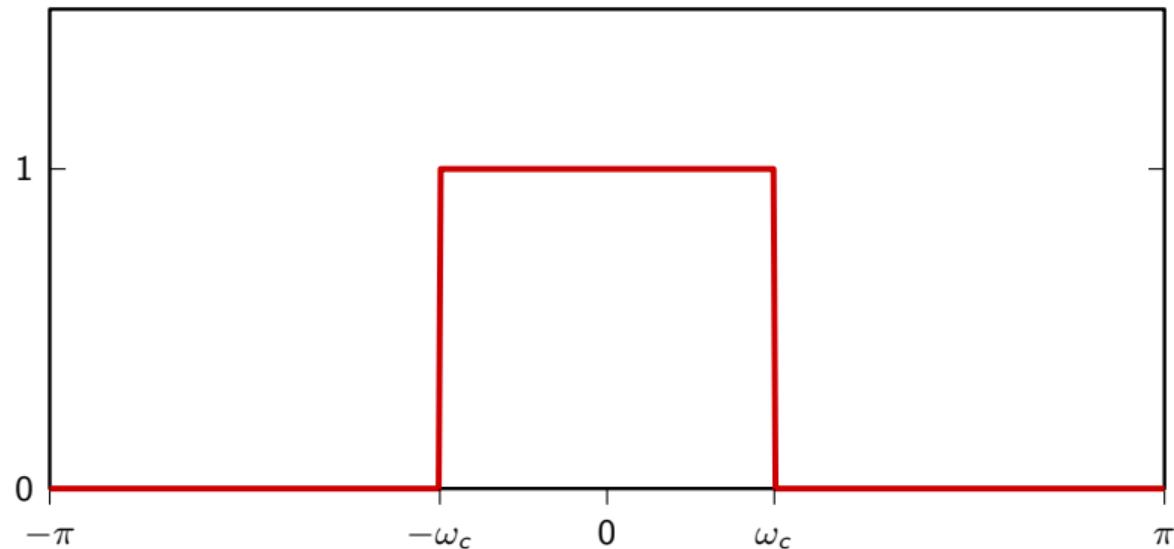
$$H_{hp}(e^{j\omega}) = 1 - H_{lp}(e^{j\omega})$$

$$h_{hp}[n] = \delta[n] - \frac{\omega_c}{\pi} \text{sinc}\left(\frac{\omega_c}{\pi} n\right)$$

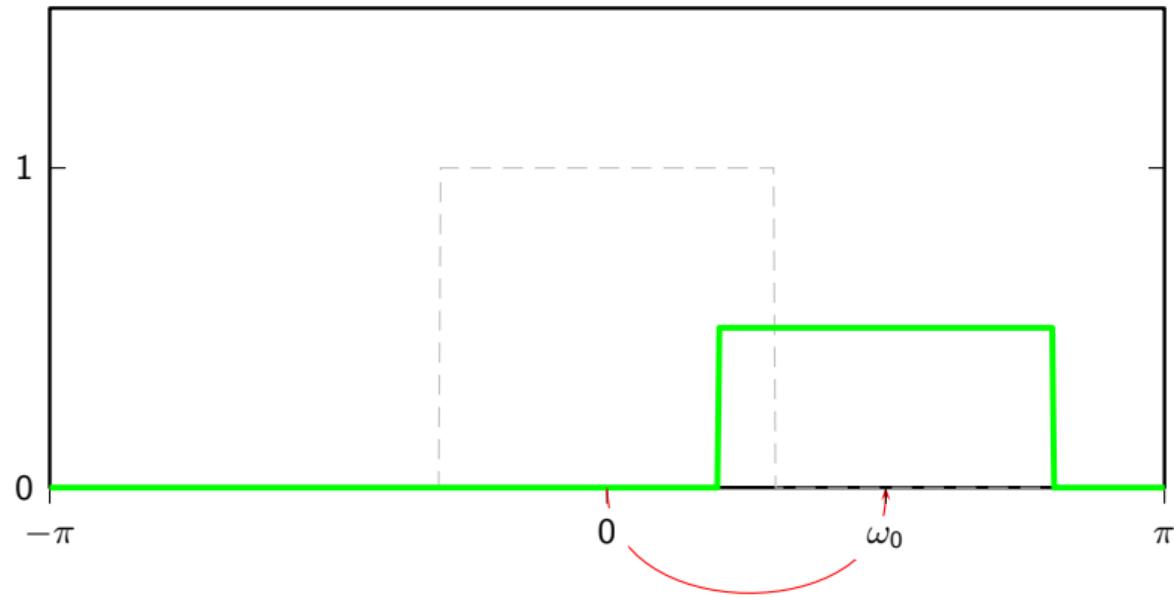
Ideal bandpass filter



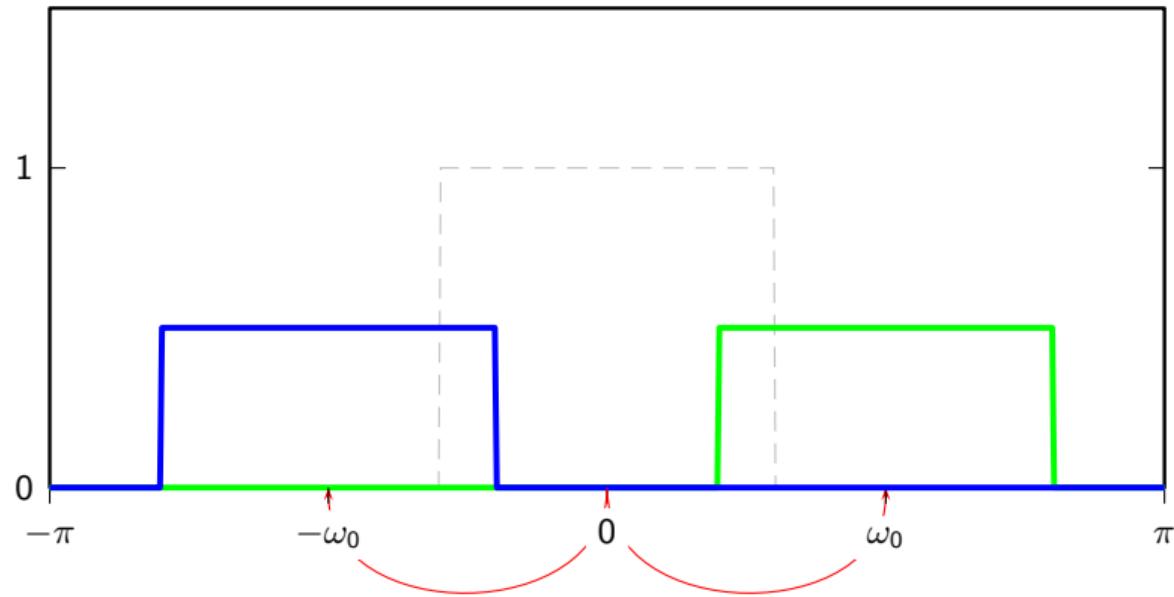
Ideal bandpass filter



Ideal bandpass filter



Ideal bandpass filter



Ideal bandpass filter

$$H_{bp}(e^{j\omega}) = \begin{cases} 1 & \text{for } |\omega \pm \omega_0| \leq \omega_c \\ 0 & \text{otherwise} \end{cases} \quad (2\pi\text{-periodicity implicit})$$

$$h_{bp}[n] = 2 \cos(\omega_0 n) \frac{\omega_c}{\pi} \operatorname{sinc}\left(\frac{\omega_c}{\pi} n\right)$$

Ideal bandpass filter

$$H_{bp}(e^{j\omega}) = \begin{cases} 1 & \text{for } |\omega \pm \omega_0| \leq \omega_c \\ 0 & \text{otherwise} \end{cases} \quad (2\pi\text{-periodicity implicit})$$

$$h_{bp}[n] = 2 \cos(\omega_0 n) \frac{\omega_c}{\pi} \operatorname{sinc}\left(\frac{\omega_c}{\pi} n\right)$$

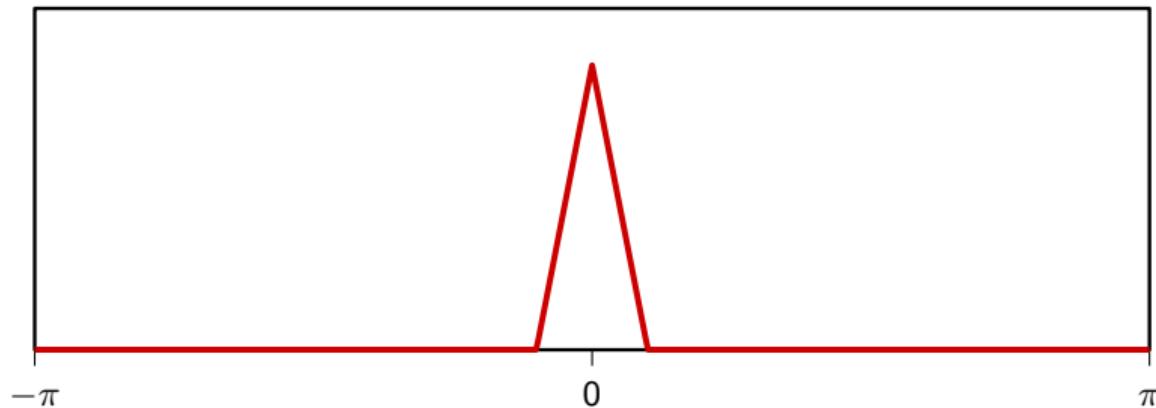
Demodulation revisited

remember the classic demodulation scheme:

- ▶ apply sinusoidal modulation to $x[n]$: $y[n] = x[n] \cos \omega_0 n$
- ▶ demodulate by multiplying by the carrier $x'[n] = y[n] \cos \omega_0 n$
- ▶ demodulated signal contains unwanted high-frequency components

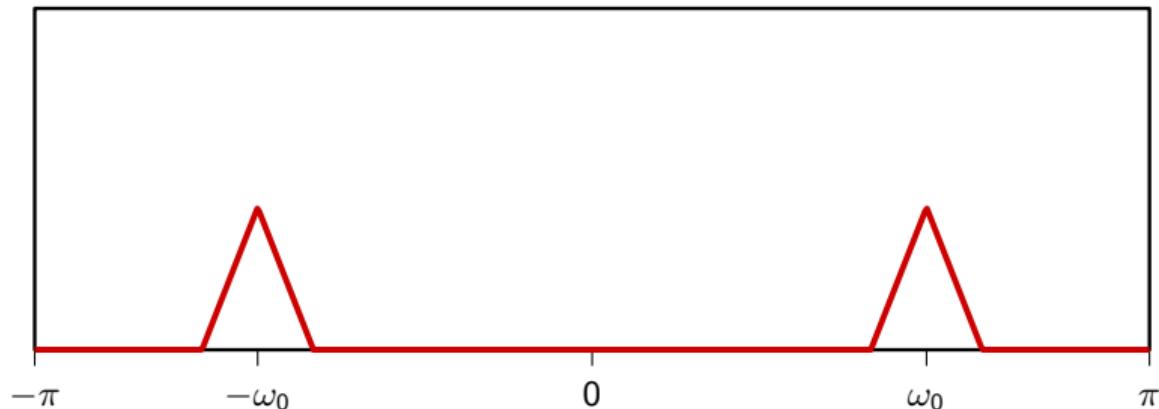
Demodulation revisited

$$X(e^{j\omega})$$



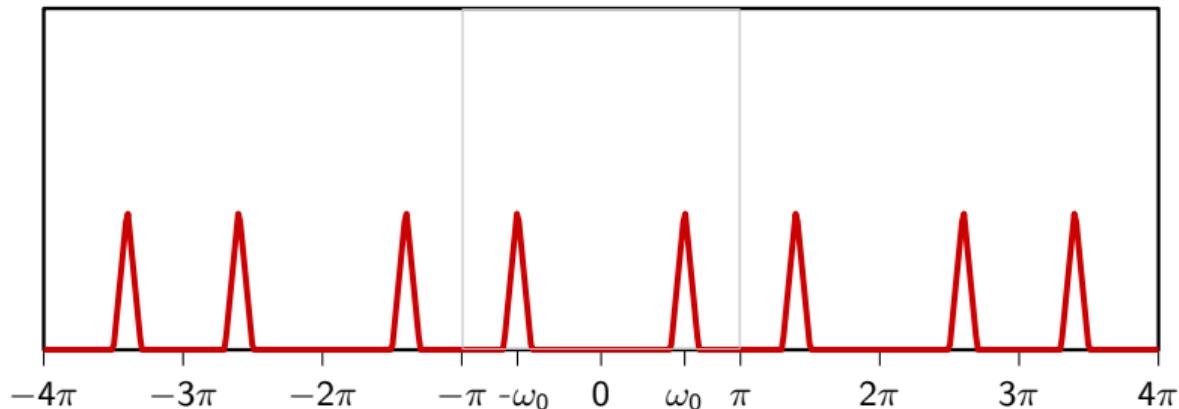
Demodulation revisited

$$Y(e^{j\omega})$$



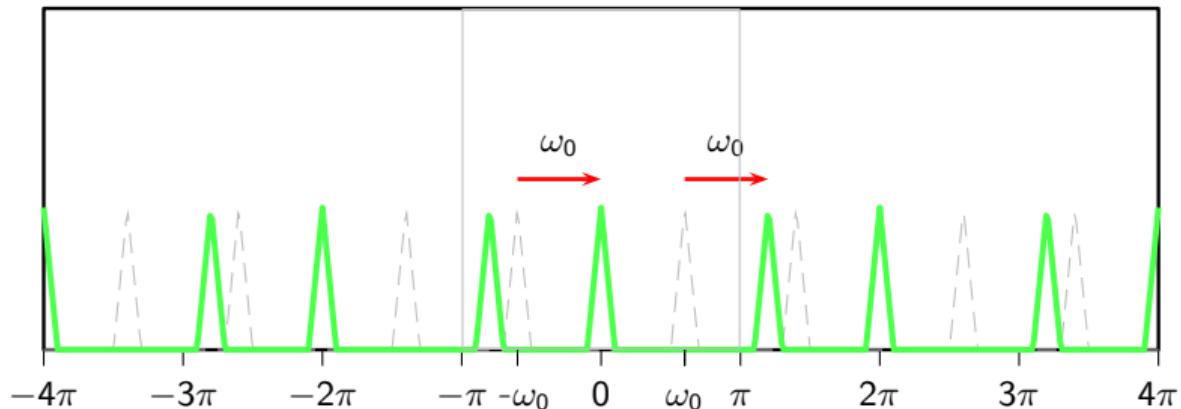
Demodulation revisited

$$Y(e^{j\omega})$$



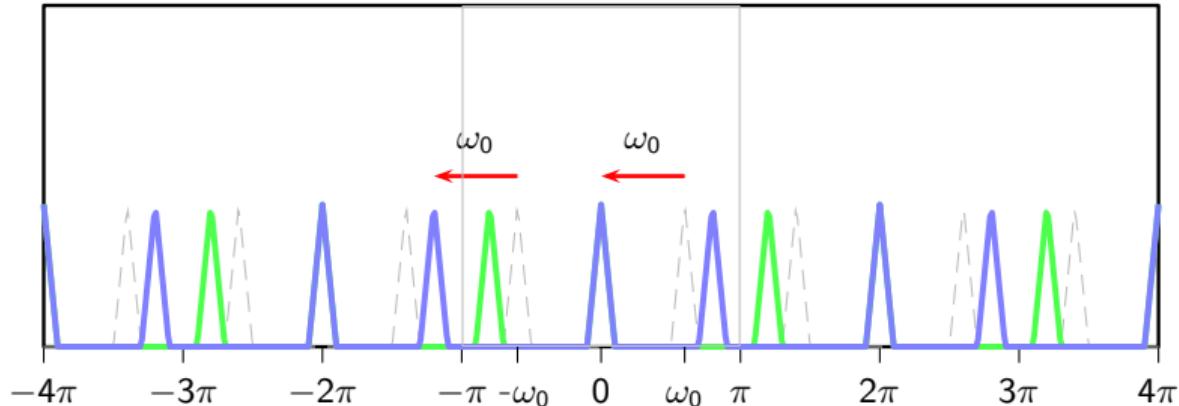
Demodulation revisited

$$X'(e^{j\omega})$$



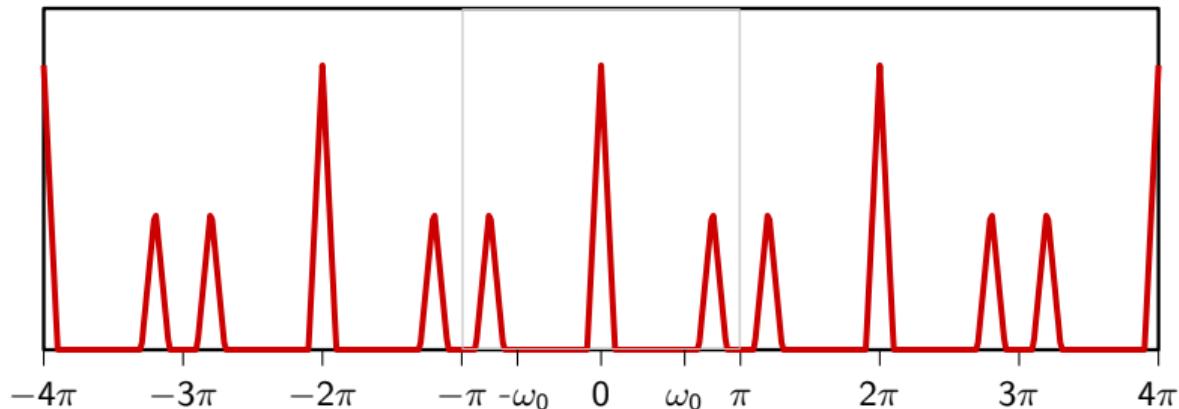
Demodulation revisited

$$X'(e^{j\omega})$$



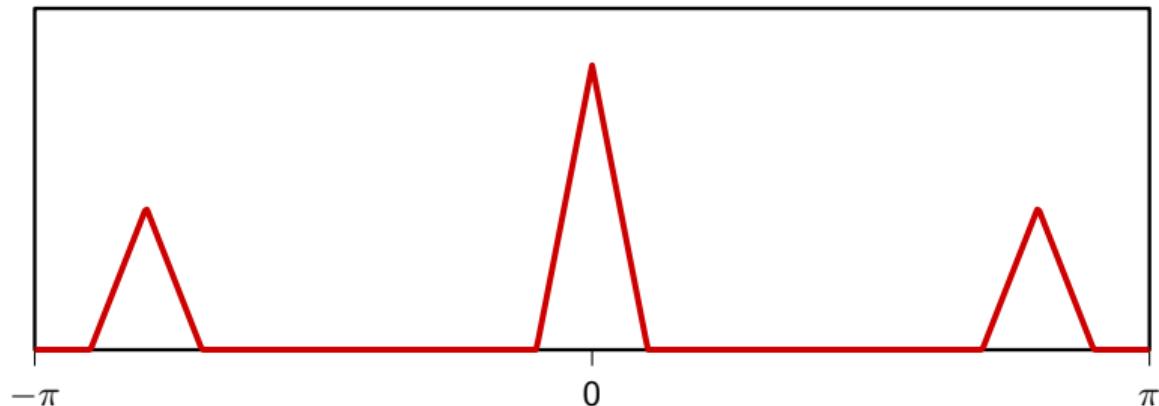
Demodulation revisited

$$X'(e^{j\omega})$$



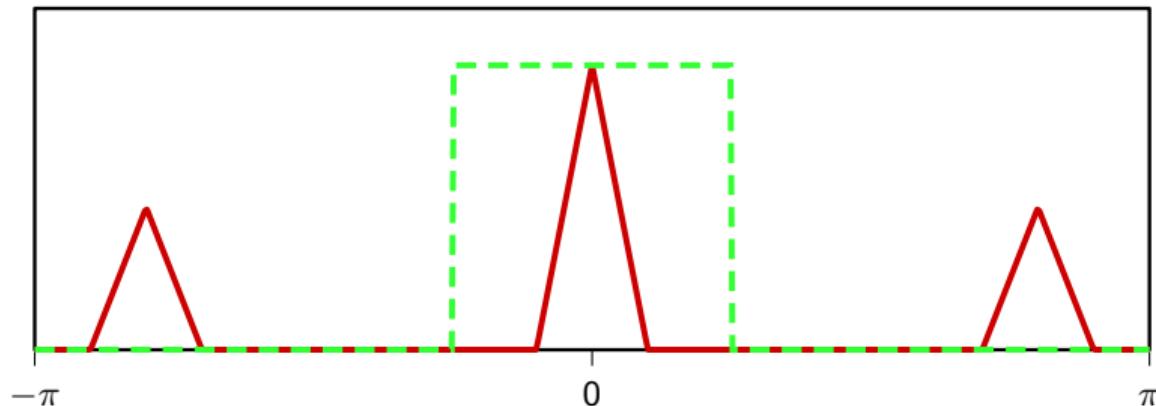
Solution: lowpass filtering

$$X'(e^{j\omega})$$

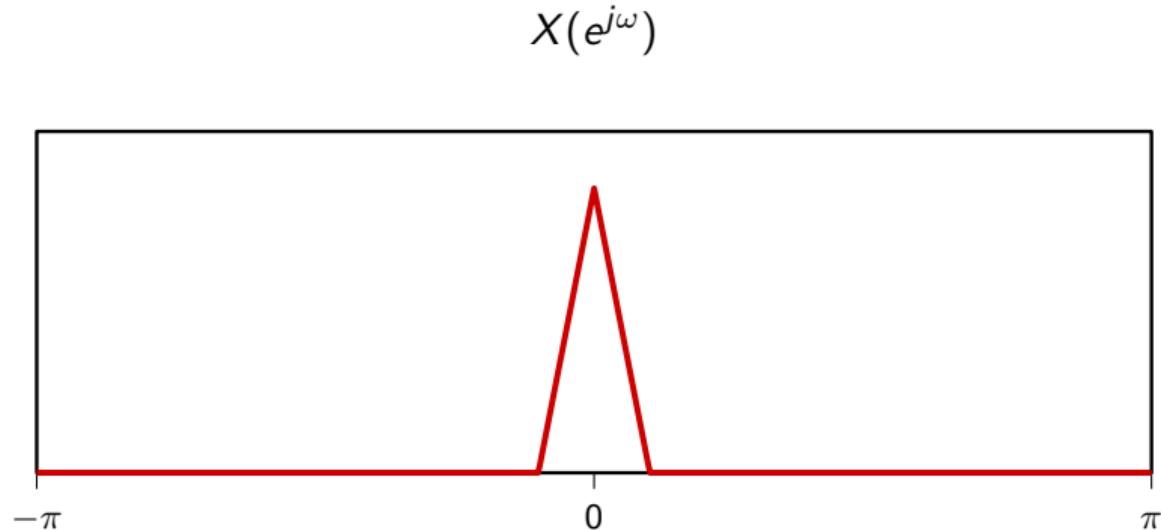


Solution: lowpass filtering

$$X'(e^{j\omega})$$



Solution: lowpass filtering



approximations of ideal filters

Overview:

- ▶ Impulse truncation
- ▶ Window method
- ▶ Frequency sampling

Overview:

- ▶ Impulse truncation
- ▶ Window method
- ▶ Frequency sampling

Overview:

- ▶ Impulse truncation
- ▶ Window method
- ▶ Frequency sampling

How can we approximate an ideal lowpass?

Idea #1:

- ▶ pick ω_c
- ▶ compute ideal impulse response $h[n]$
- ▶ truncate $h[n]$ to a finite-support $\hat{h}[n]$
- ▶ $\hat{h}[n]$ defines an FIR filter

How can we approximate an ideal lowpass?

Idea #1:

- ▶ pick ω_c
- ▶ compute ideal impulse response $h[n]$
- ▶ truncate $h[n]$ to a finite-support $\hat{h}[n]$
- ▶ $\hat{h}[n]$ defines an FIR filter

How can we approximate an ideal lowpass?

Idea #1:

- ▶ pick ω_c
- ▶ compute ideal impulse response $h[n]$
- ▶ truncate $h[n]$ to a finite-support $\hat{h}[n]$
- ▶ $\hat{h}[n]$ defines an FIR filter

How can we approximate an ideal lowpass?

Idea #1:

- ▶ pick ω_c
- ▶ compute ideal impulse response $h[n]$
- ▶ truncate $h[n]$ to a finite-support $\hat{h}[n]$
- ▶ $\hat{h}[n]$ defines an FIR filter

Approximation by truncation

FIR approximation of length $M = 2N + 1$:

$$\hat{h}[n] = \begin{cases} \frac{\omega_c}{\pi} \operatorname{sinc}\left(\frac{\omega_c}{\pi} n\right) & |n| \leq N \\ 0 & \text{otherwise} \end{cases}$$

Why it could be a good idea

$$\begin{aligned}\text{MSE} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega}) - \hat{H}(e^{j\omega})|^2 d\omega \\ &= \|H(e^{j\omega}) - \hat{H}(e^{j\omega})\|^2 \\ &= \|h[n] - \hat{h}[n]\|^2 \\ &= \sum_{n=-\infty}^{\infty} |h[n] - \hat{h}[n]|^2\end{aligned}$$

MSE is minimized by symmetric impulse truncation around zero

Why it could be a good idea

$$\begin{aligned}\text{MSE} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega}) - \hat{H}(e^{j\omega})|^2 d\omega \\ &= \|H(e^{j\omega}) - \hat{H}(e^{j\omega})\|^2 \\ &= \|h[n] - \hat{h}[n]\|^2 \\ &= \sum_{n=-\infty}^{\infty} |h[n] - \hat{h}[n]|^2\end{aligned}$$

MSE is minimized by symmetric impulse truncation around zero

Why it could be a good idea

$$\begin{aligned}\text{MSE} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega}) - \hat{H}(e^{j\omega})|^2 d\omega \\ &= \|H(e^{j\omega}) - \hat{H}(e^{j\omega})\|^2 \\ &= \|h[n] - \hat{h}[n]\|^2 \\ &= \sum_{n=-\infty}^{\infty} |h[n] - \hat{h}[n]|^2\end{aligned}$$

MSE is minimized by symmetric impulse truncation around zero

Why it could be a good idea

$$\begin{aligned}\text{MSE} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega}) - \hat{H}(e^{j\omega})|^2 d\omega \\ &= \|H(e^{j\omega}) - \hat{H}(e^{j\omega})\|^2 \\ &= \|h[n] - \hat{h}[n]\|^2 \\ &= \sum_{n=-\infty}^{\infty} |h[n] - \hat{h}[n]|^2\end{aligned}$$

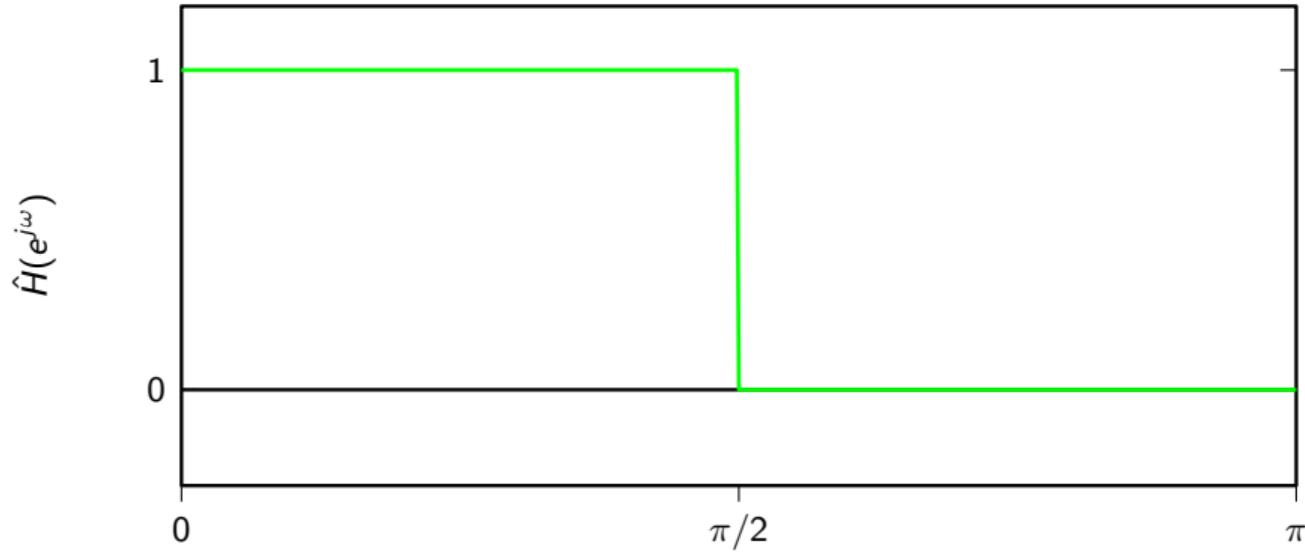
MSE is minimized by symmetric impulse truncation around zero

Why it could be a good idea

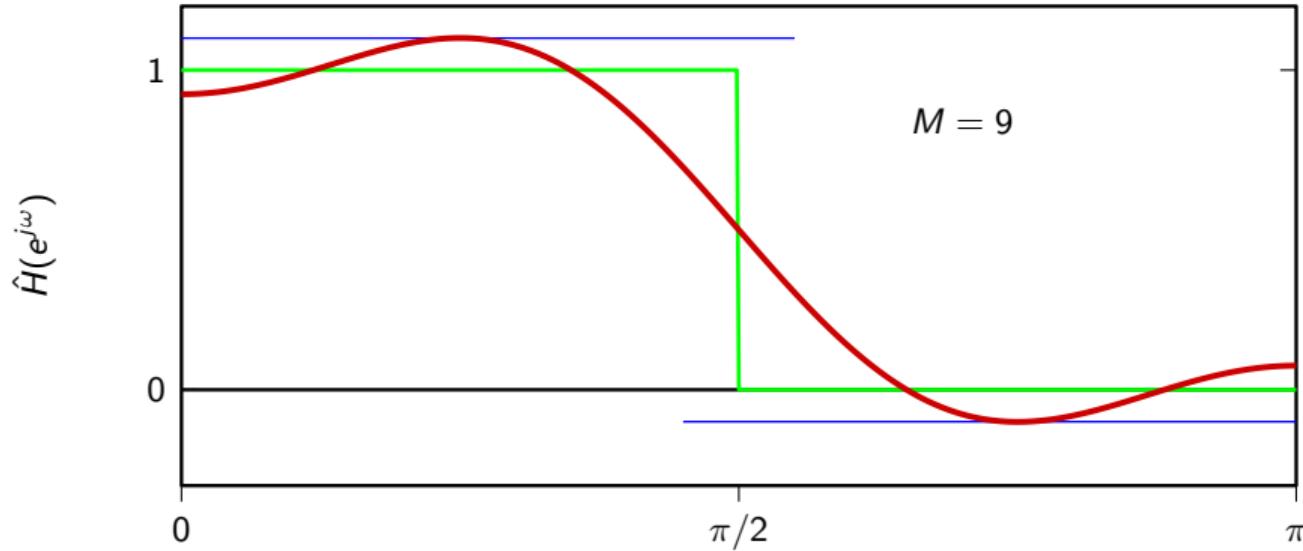
$$\begin{aligned}\text{MSE} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega}) - \hat{H}(e^{j\omega})|^2 d\omega \\ &= \|H(e^{j\omega}) - \hat{H}(e^{j\omega})\|^2 \\ &= \|h[n] - \hat{h}[n]\|^2 \\ &= \sum_{n=-\infty}^{\infty} |h[n] - \hat{h}[n]|^2\end{aligned}$$

MSE is minimized by symmetric impulse truncation around zero

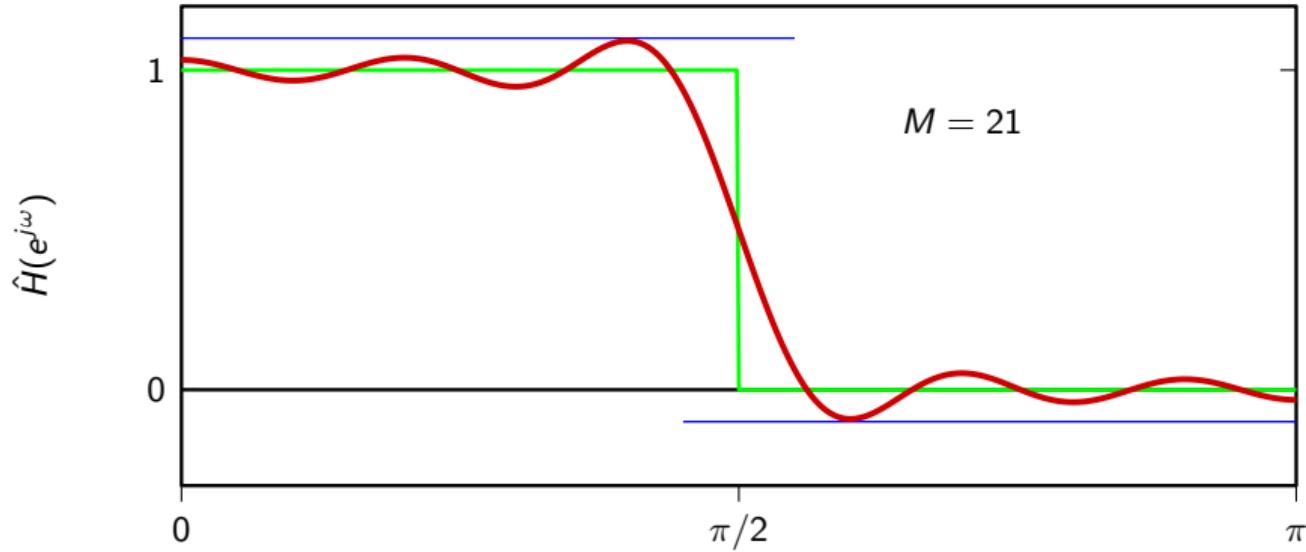
Why it's not such a good idea



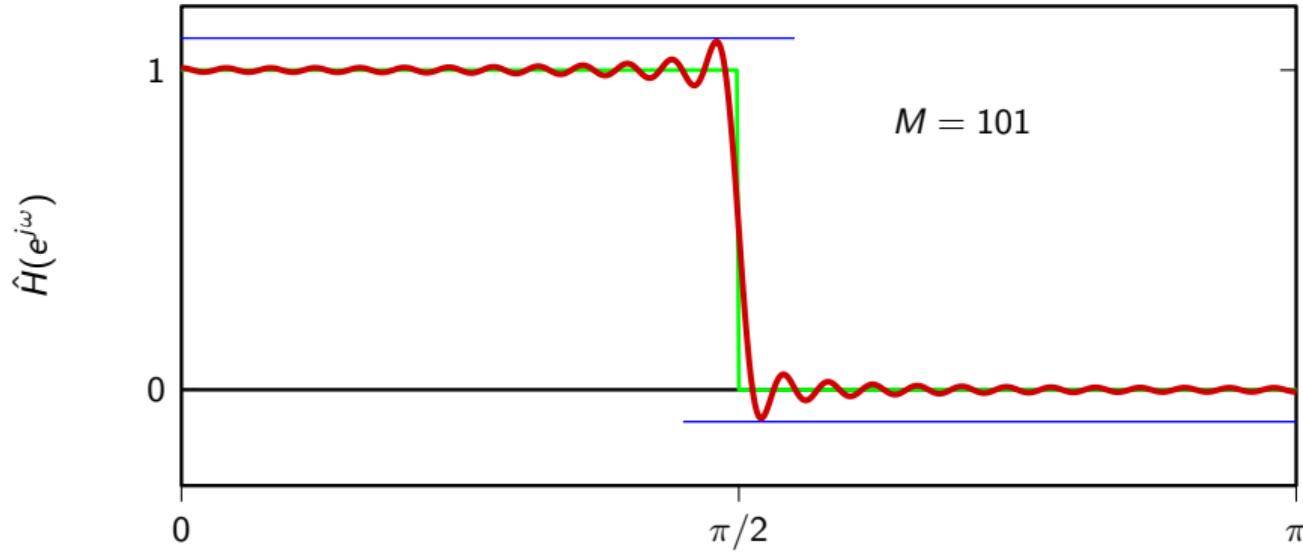
Why it's not such a good idea



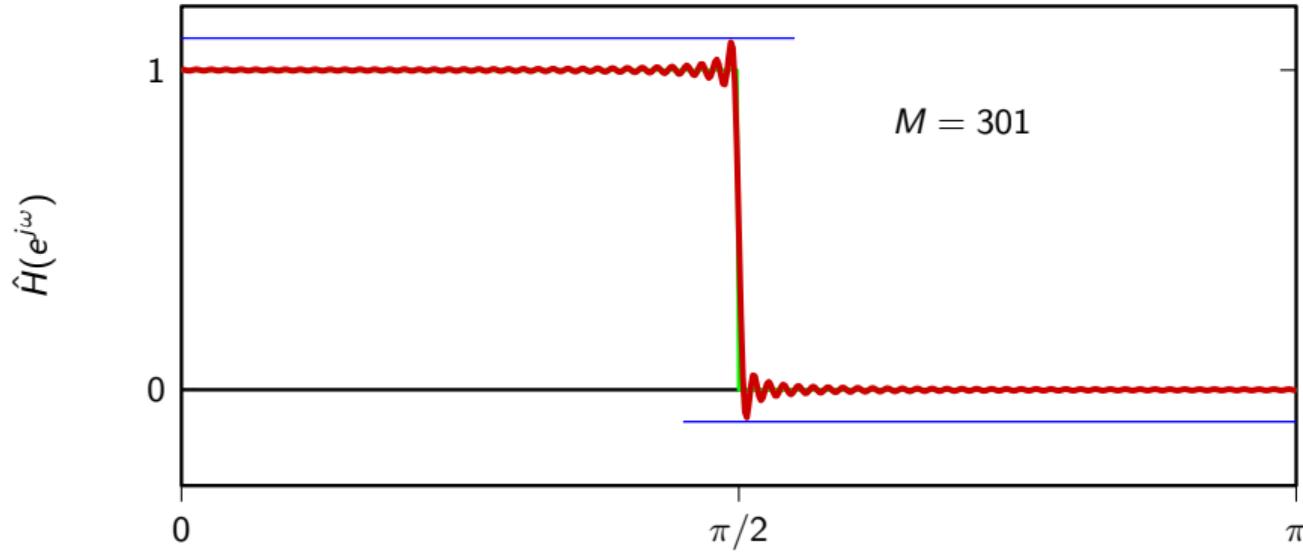
Why it's not such a good idea



Why it's not such a good idea



Why it's not such a good idea



The Gibbs phenomenon

The maximum error around the cutoff frequency
is around 9% of the height of the jump
regardless of N

Understanding the Gibbs phenomenon

$$\hat{h}[n] = h[n] w[n]$$

$$w[n] = \begin{cases} 1 & |n| \leq N \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{H}(e^{j\omega}) = ?$$

Understanding the Gibbs phenomenon

$$\hat{h}[n] = h[n] w[n]$$

$$w[n] = \begin{cases} 1 & |n| \leq N \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{H}(e^{j\omega}) = ?$$

The convolution and modulation theorems

$$\text{DTFT } \{(x * y)[n]\} = X(e^{j\omega}) Y(e^{j\omega})$$

$$\text{DTFT } \{x[n] y[n]\} = (X * Y)(e^{j\omega})$$

The convolution and modulation theorems

$$\text{DTFT} \{(x * y)[n]\} = X(e^{j\omega}) Y(e^{j\omega})$$

$$\text{DTFT} \{x[n] y[n]\} = (X * Y)(e^{j\omega})$$

Convolution of DTFTs

in $\ell_2(\mathbb{Z})$:

$$(x * y)[n] = \langle x^*[k], y[n - k] \rangle$$

$$= \sum_{k=-\infty}^{\infty} x[k]y[n - k]$$

in $L_2([-\pi, \pi])$:

$$(X * Y)(e^{j\omega}) = \langle X^*(e^{j\sigma}), Y(e^{j(\omega-\sigma)}) \rangle$$

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) Y(e^{j(\omega-\sigma)}) d\sigma$$

Convolution of DTFTs

in $\ell_2(\mathbb{Z})$:

$$(x * y)[n] = \langle x^*[k], y[n - k] \rangle$$

$$= \sum_{k=-\infty}^{\infty} x[k]y[n - k]$$

in $L_2([-\pi, \pi])$:

$$(X * Y)(e^{j\omega}) = \langle X^*(e^{j\sigma}), Y(e^{j(\omega-\sigma)}) \rangle$$

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) Y(e^{j(\omega-\sigma)}) d\sigma$$

Modulation theorem: proof

$$\begin{aligned}\text{IDTFT} \{(X * Y)(e^{j\omega})\} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} (X * Y)(e^{j\omega}) e^{j\omega n} d\omega \\&= \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) Y(e^{j(\omega-\sigma)}) e^{j\omega n} d\sigma d\omega \\&= \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) Y(e^{j(\omega-\sigma)}) e^{j\sigma n} e^{j(\omega-\sigma)n} d\sigma d\omega \\&= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) e^{j\sigma n} d\sigma \frac{1}{2\pi} \int_{-\pi}^{\pi} Y(e^{j(\omega-\sigma)}) e^{j(\omega-\sigma)n} d\omega \\&= x[n] y[n]\end{aligned}$$

Modulation theorem: proof

$$\begin{aligned}\text{IDTFT} \{(X * Y)(e^{j\omega})\} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} (X * Y)(e^{j\omega}) e^{j\omega n} d\omega \\&= \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) Y(e^{j(\omega-\sigma)}) e^{j\omega n} d\sigma d\omega \\&= \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) Y(e^{j(\omega-\sigma)}) e^{j\sigma n} e^{j(\omega-\sigma)n} d\sigma d\omega \\&= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) e^{j\sigma n} d\sigma \frac{1}{2\pi} \int_{-\pi}^{\pi} Y(e^{j(\omega-\sigma)}) e^{j(\omega-\sigma)n} d\omega \\&= x[n] y[n]\end{aligned}$$

Modulation theorem: proof

$$\begin{aligned}\text{IDTFT} \{(X * Y)(e^{j\omega})\} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} (X * Y)(e^{j\omega}) e^{j\omega n} d\omega \\&= \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) Y(e^{j(\omega-\sigma)}) e^{j\omega n} d\sigma d\omega \\&= \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) Y(e^{j(\omega-\sigma)}) e^{j\sigma n} e^{j(\omega-\sigma)n} d\sigma d\omega \\&= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) e^{j\sigma n} d\sigma \frac{1}{2\pi} \int_{-\pi}^{\pi} Y(e^{j(\omega-\sigma)}) e^{j(\omega-\sigma)n} d\omega \\&= x[n] y[n]\end{aligned}$$

Modulation theorem: proof

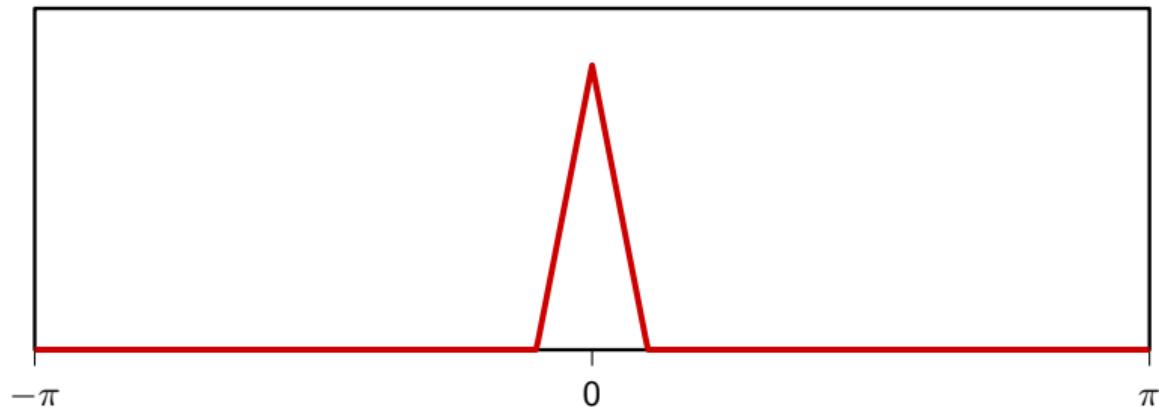
$$\begin{aligned}\text{IDTFT} \{(X * Y)(e^{j\omega})\} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} (X * Y)(e^{j\omega}) e^{j\omega n} d\omega \\&= \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) Y(e^{j(\omega-\sigma)}) e^{j\omega n} d\sigma d\omega \\&= \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) Y(e^{j(\omega-\sigma)}) e^{j\sigma n} e^{j(\omega-\sigma)n} d\sigma d\omega \\&= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) e^{j\sigma n} d\sigma \frac{1}{2\pi} \int_{-\pi}^{\pi} Y(e^{j(\omega-\sigma)}) e^{j(\omega-\sigma)n} d\omega \\&= x[n] y[n]\end{aligned}$$

Modulation theorem: proof

$$\begin{aligned}\text{IDTFT} \{(X * Y)(e^{j\omega})\} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} (X * Y)(e^{j\omega}) e^{j\omega n} d\omega \\&= \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) Y(e^{j(\omega-\sigma)}) e^{j\omega n} d\sigma d\omega \\&= \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) Y(e^{j(\omega-\sigma)}) e^{j\sigma n} e^{j(\omega-\sigma)n} d\sigma d\omega \\&= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) e^{j\sigma n} d\sigma \frac{1}{2\pi} \int_{-\pi}^{\pi} Y(e^{j(\omega-\sigma)}) e^{j(\omega-\sigma)n} d\omega \\&= x[n] y[n]\end{aligned}$$

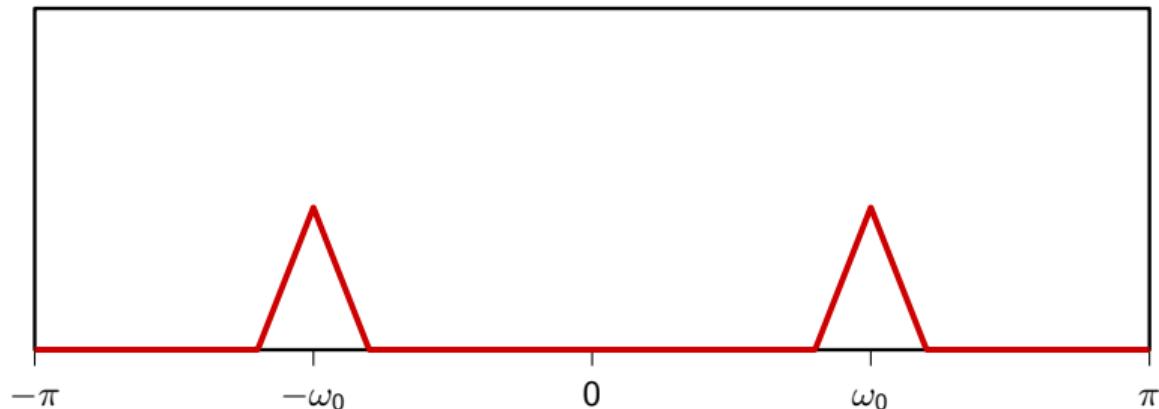
Aside: sinusoidal modulation revisited

$$X(e^{j\omega})$$



Aside: sinusoidal modulation revisited

$$Y(e^{j\omega}) = \text{DTFT} \{x[n] \cos(\omega_0 n)\}$$



Aside: sinusoidal modulation revisited

$$\begin{aligned}\text{DTFT } \{x[n] \cos \omega_0 n\} &= X(e^{j\omega}) * \frac{1}{2}[\tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0)] \\ &= \frac{1}{4\pi} \int_{-\pi}^{\pi} X(e^{j(\omega-\sigma)}) \tilde{\delta}(\sigma - \omega_0) d\sigma + \frac{1}{4\pi} \int_{-\pi}^{\pi} X(e^{j(\omega-\sigma)}) \tilde{\delta}(\sigma + \omega_0) d\sigma \\ &= \frac{1}{2} [X(e^{j(\omega-\omega_0)}) + X(e^{j(\omega+\omega_0)})]\end{aligned}$$

Aside: sinusoidal modulation revisited

$$\begin{aligned}\text{DTFT } \{x[n] \cos \omega_0 n\} &= X(e^{j\omega}) * \frac{1}{2}[\tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0)] \\ &= \frac{1}{4\pi} \int_{-\pi}^{\pi} X(e^{j(\omega-\sigma)}) \tilde{\delta}(\sigma - \omega_0) d\sigma + \frac{1}{4\pi} \int_{-\pi}^{\pi} X(e^{j(\omega-\sigma)}) \tilde{\delta}(\sigma + \omega_0) d\sigma \\ &= \frac{1}{2} [X(e^{j(\omega-\omega_0)}) + X(e^{j(\omega+\omega_0)})]\end{aligned}$$

Aside: sinusoidal modulation revisited

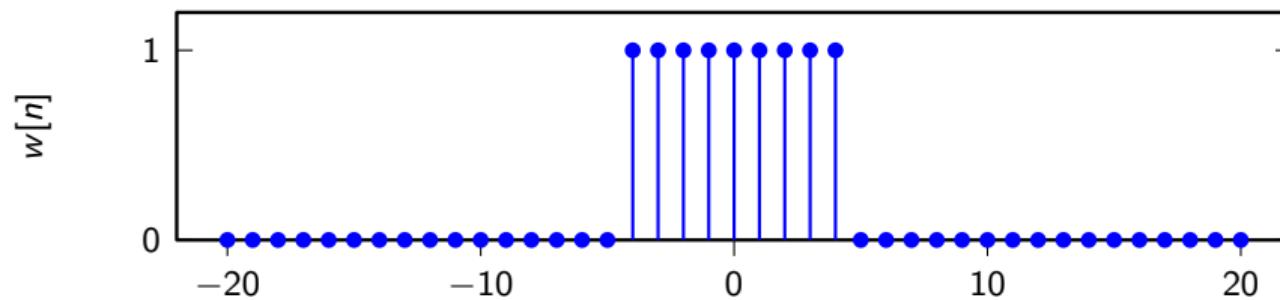
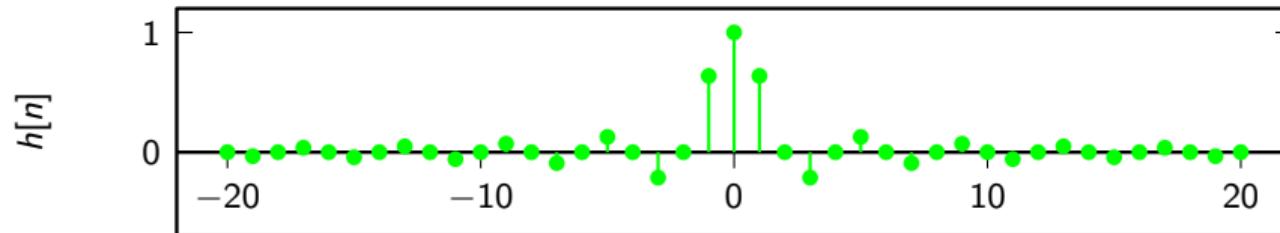
$$\begin{aligned}\text{DTFT } \{x[n] \cos \omega_0 n\} &= X(e^{j\omega}) * \frac{1}{2}[\tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0)] \\ &= \frac{1}{4\pi} \int_{-\pi}^{\pi} X(e^{j(\omega-\sigma)}) \tilde{\delta}(\sigma - \omega_0) d\sigma + \frac{1}{4\pi} \int_{-\pi}^{\pi} X(e^{j(\omega-\sigma)}) \tilde{\delta}(\sigma + \omega_0) d\sigma \\ &= \frac{1}{2} [X(e^{j(\omega-\omega_0)}) + X(e^{j(\omega+\omega_0)})]\end{aligned}$$

Back to the Gibbs phenomenon

The maximum error around the cutoff frequency
is around 9% of the height of the jump
regardless of N

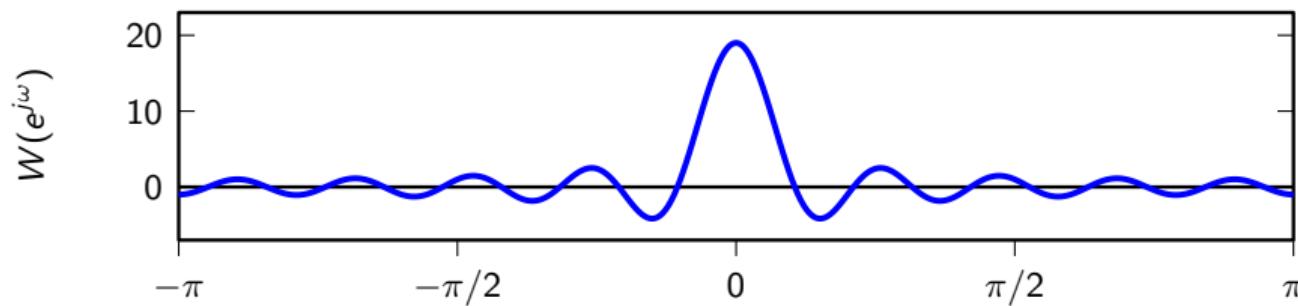
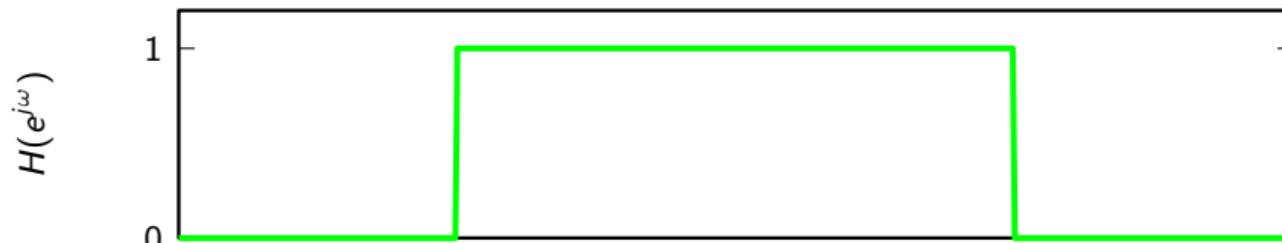
Understanding the Gibbs phenomenon

$$\hat{h}[n] = h[n] w[n]$$



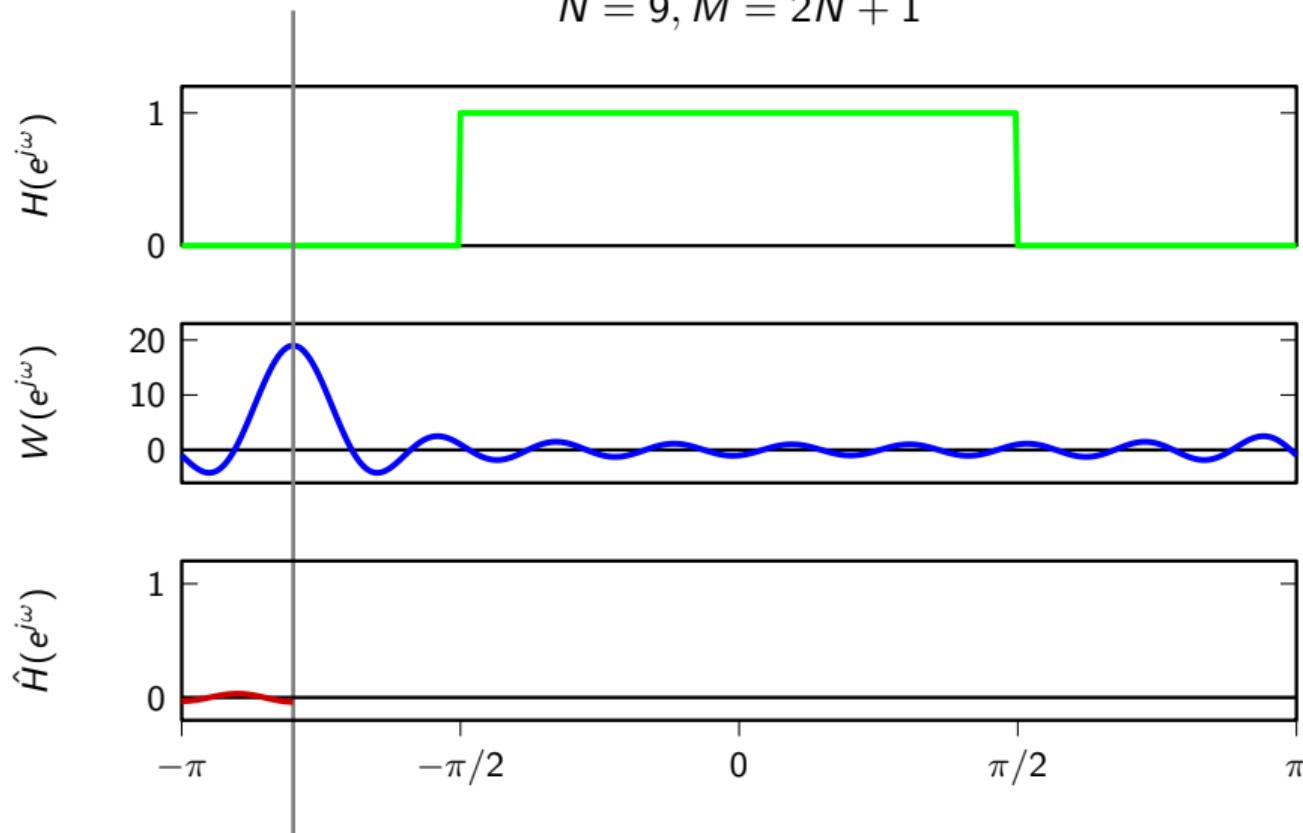
Understanding the Gibbs phenomenon

$$\hat{H}(e^{j\omega}) = (H * W)(e^{j\omega}), \quad W(e^{j\omega}) = \sin(\omega M/2) / \sin(\omega/2)$$



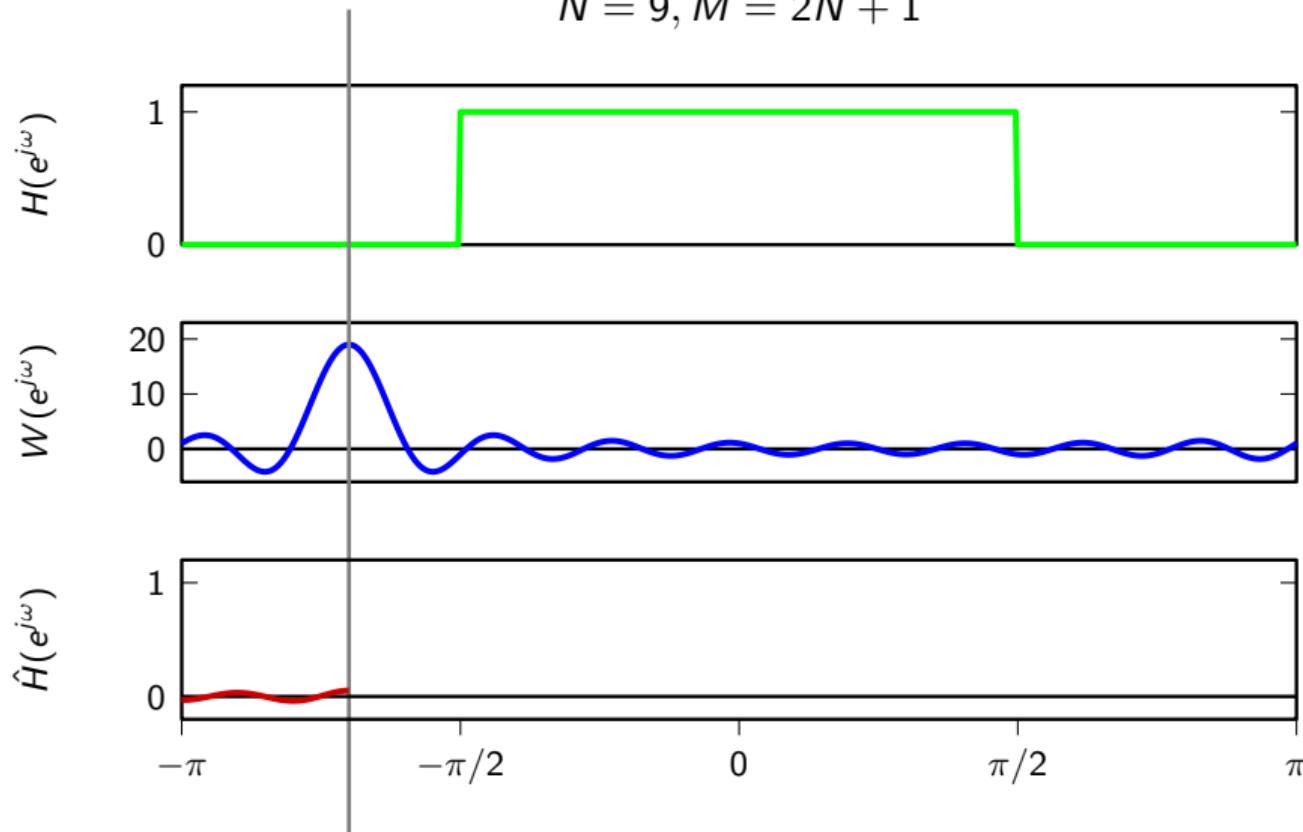
Implicit frequency-domain convolution

$$N = 9, M = 2N + 1$$



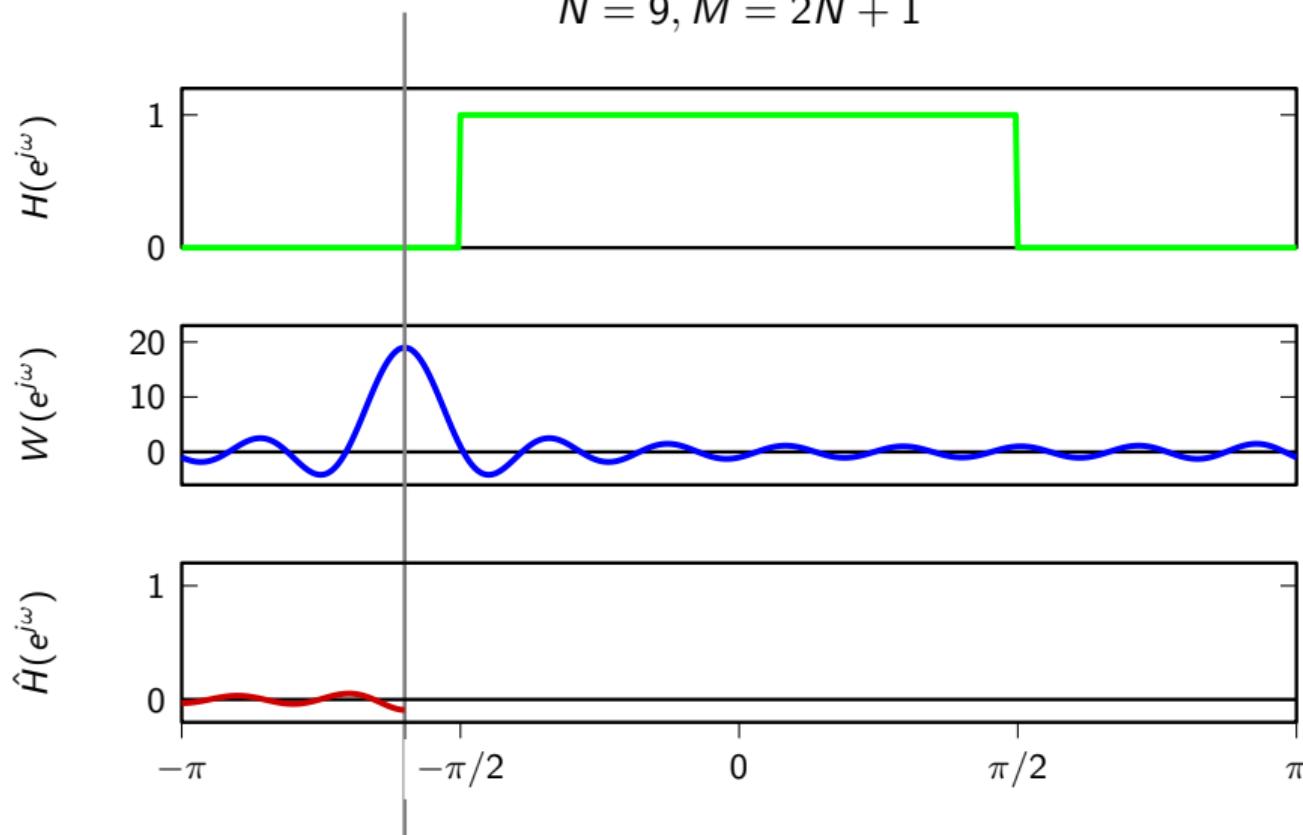
Implicit frequency-domain convolution

$$N = 9, M = 2N + 1$$

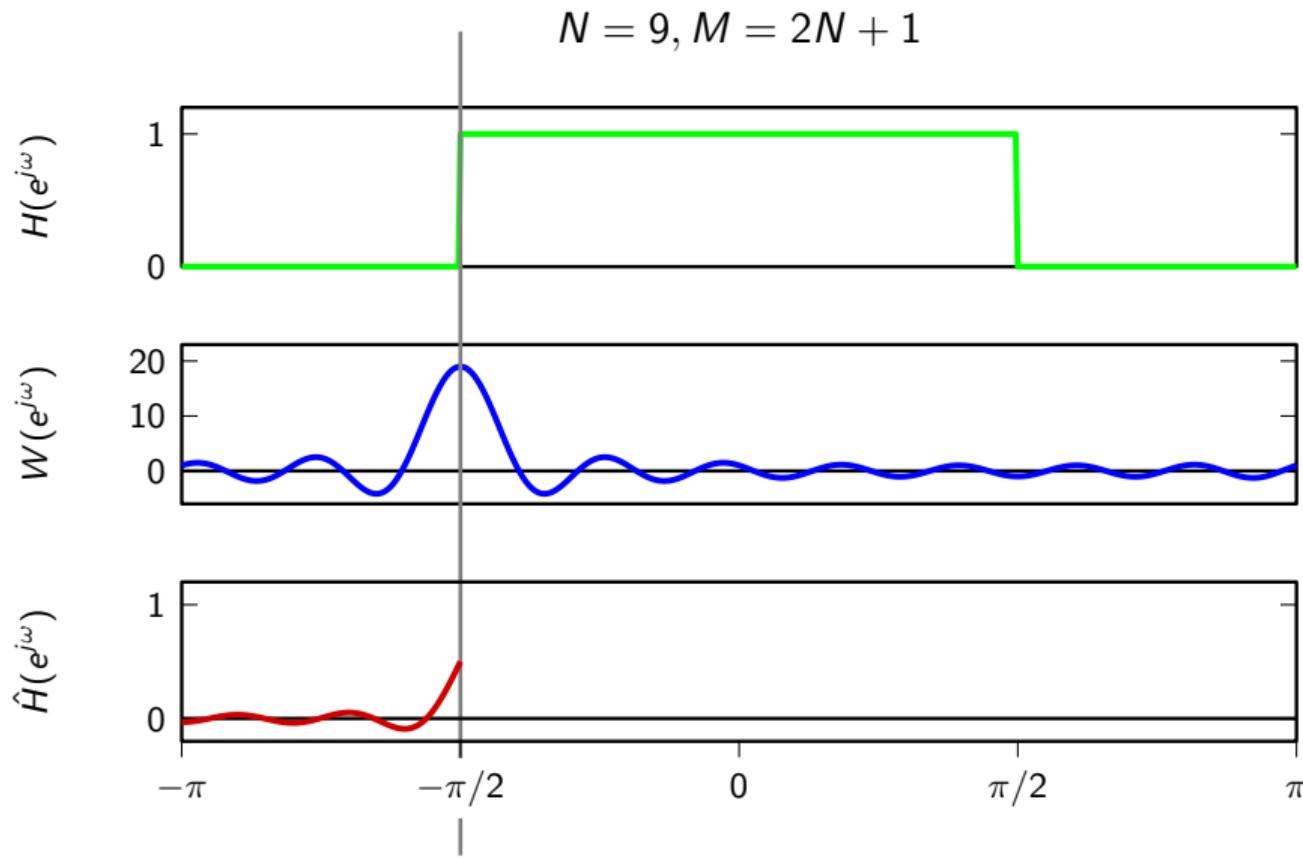


Implicit frequency-domain convolution

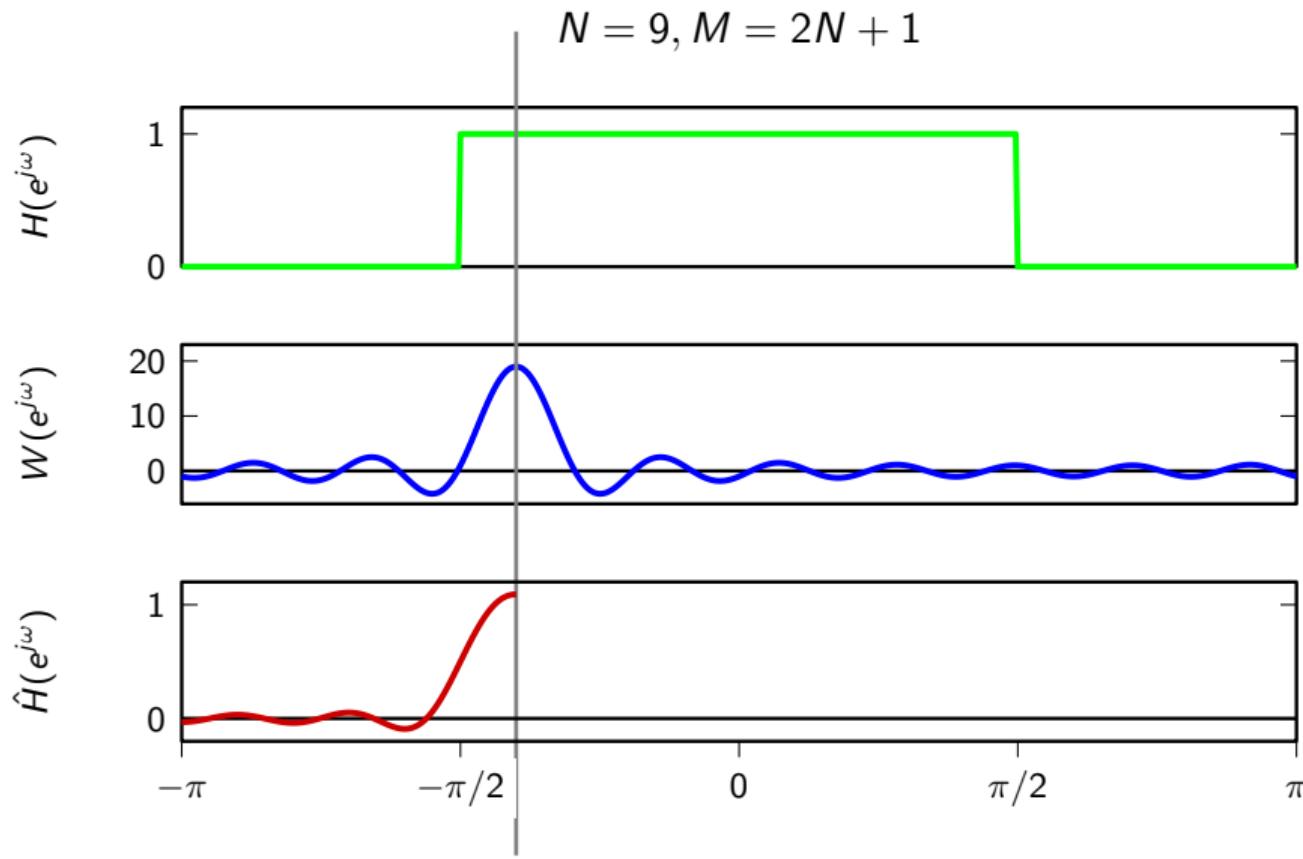
$$N = 9, M = 2N + 1$$



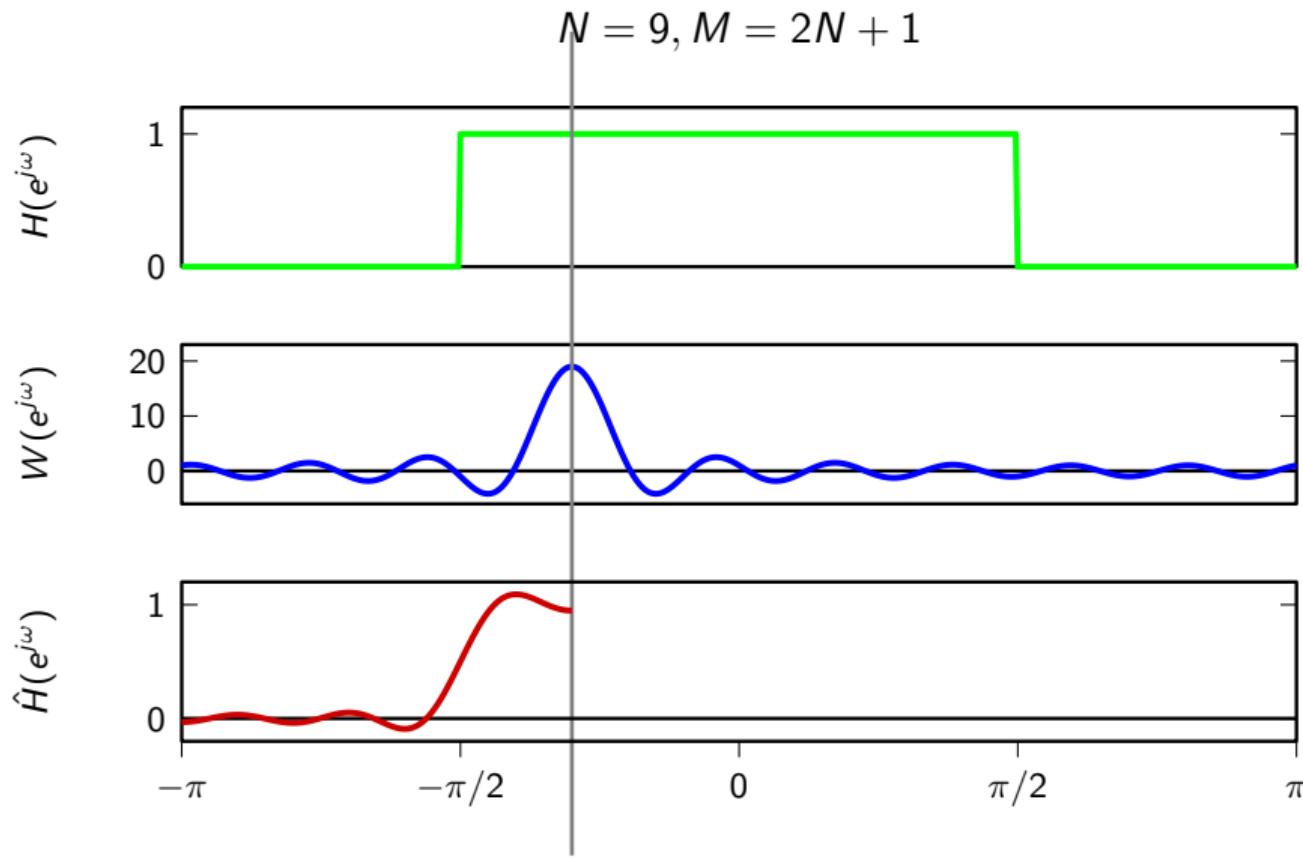
Implicit frequency-domain convolution



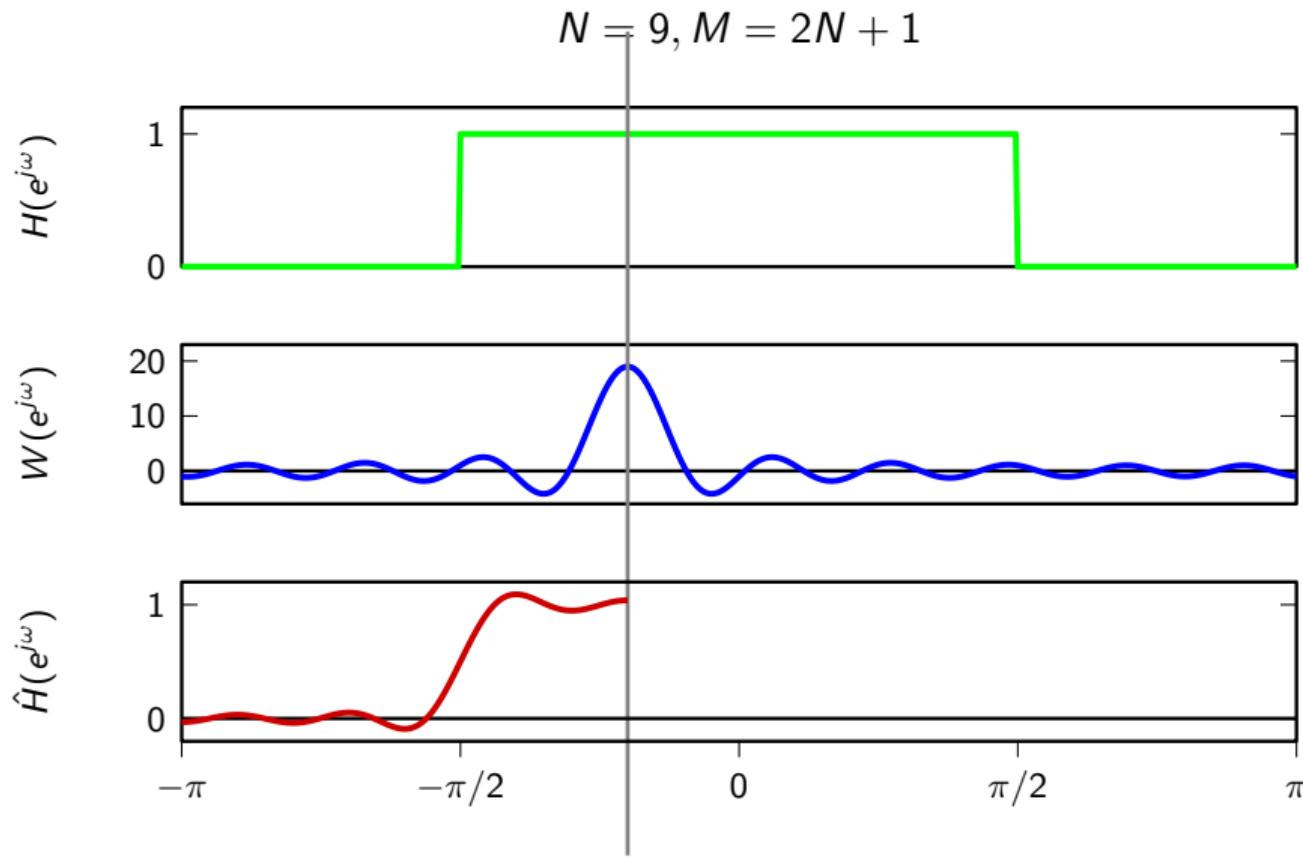
Implicit frequency-domain convolution



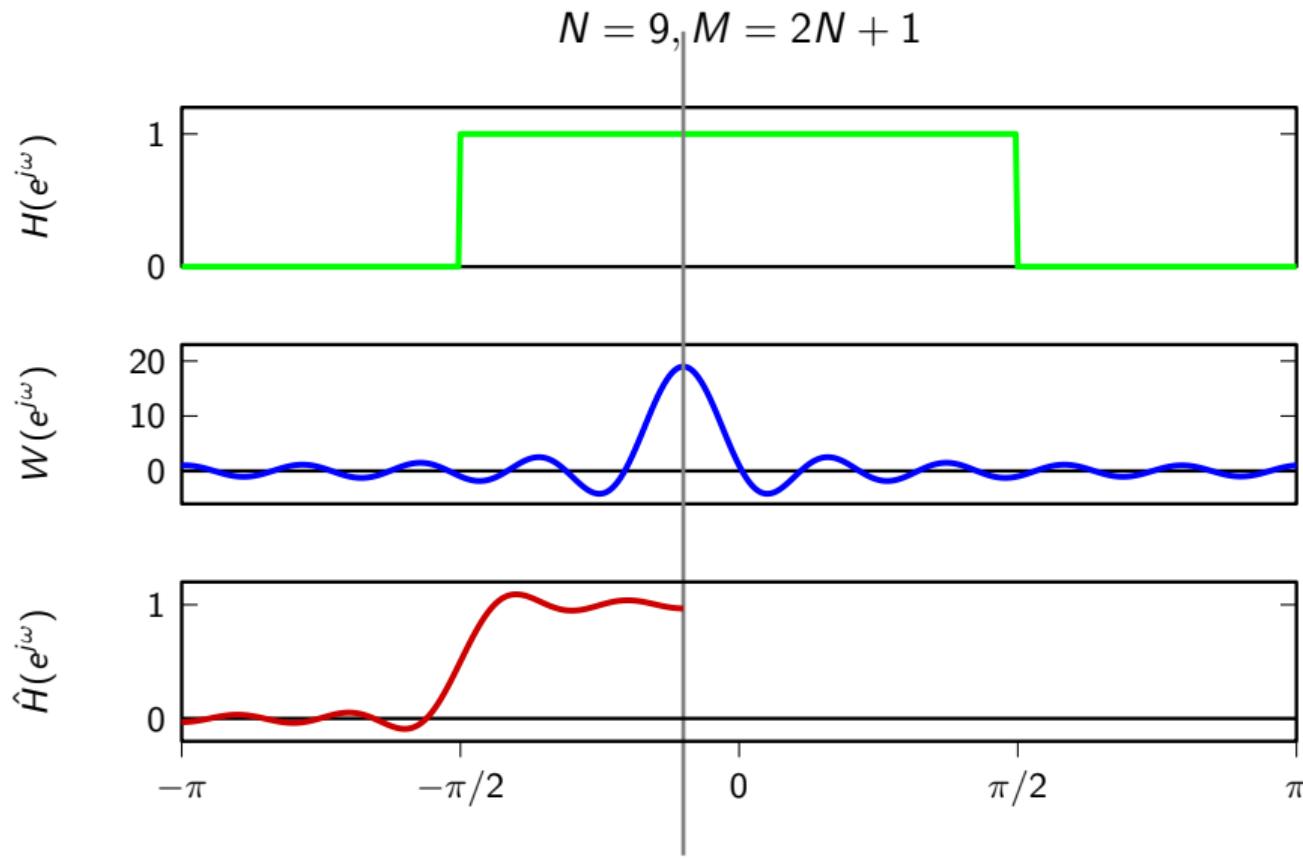
Implicit frequency-domain convolution



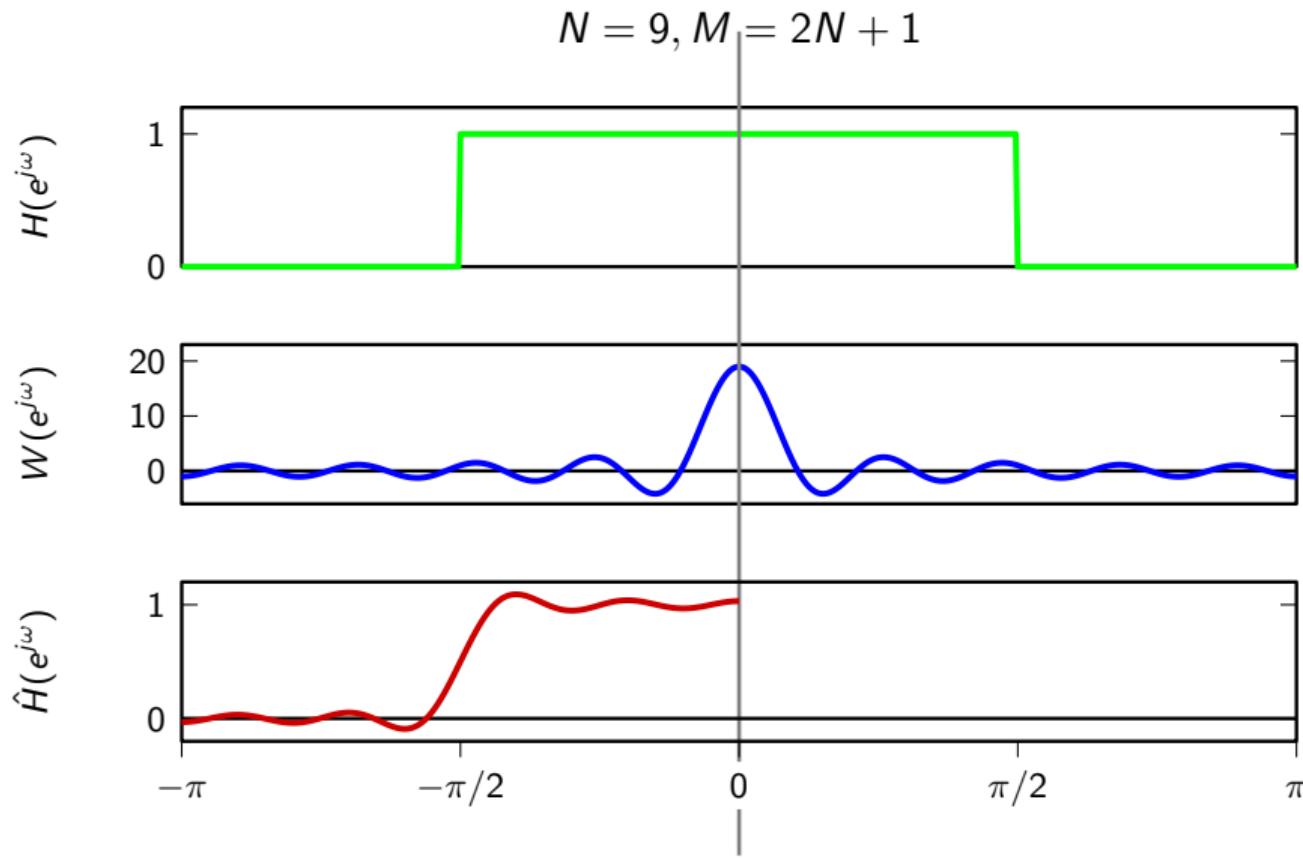
Implicit frequency-domain convolution



Implicit frequency-domain convolution

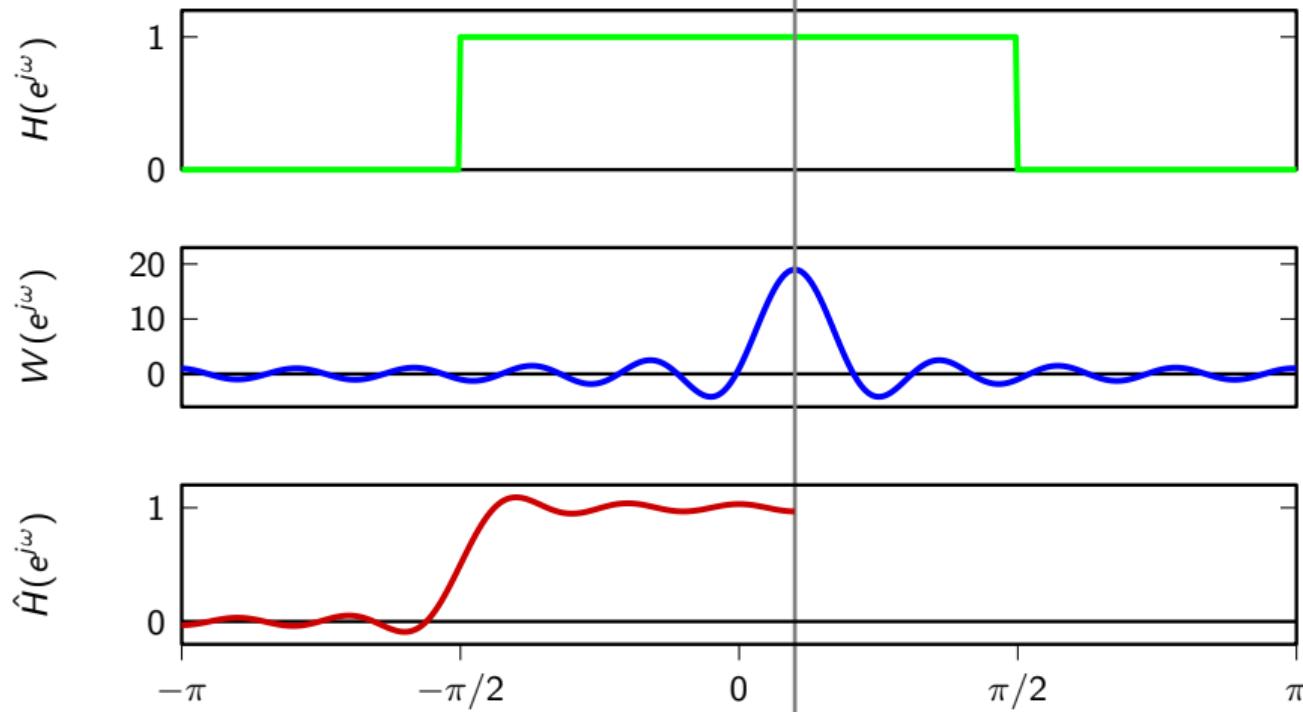


Implicit frequency-domain convolution

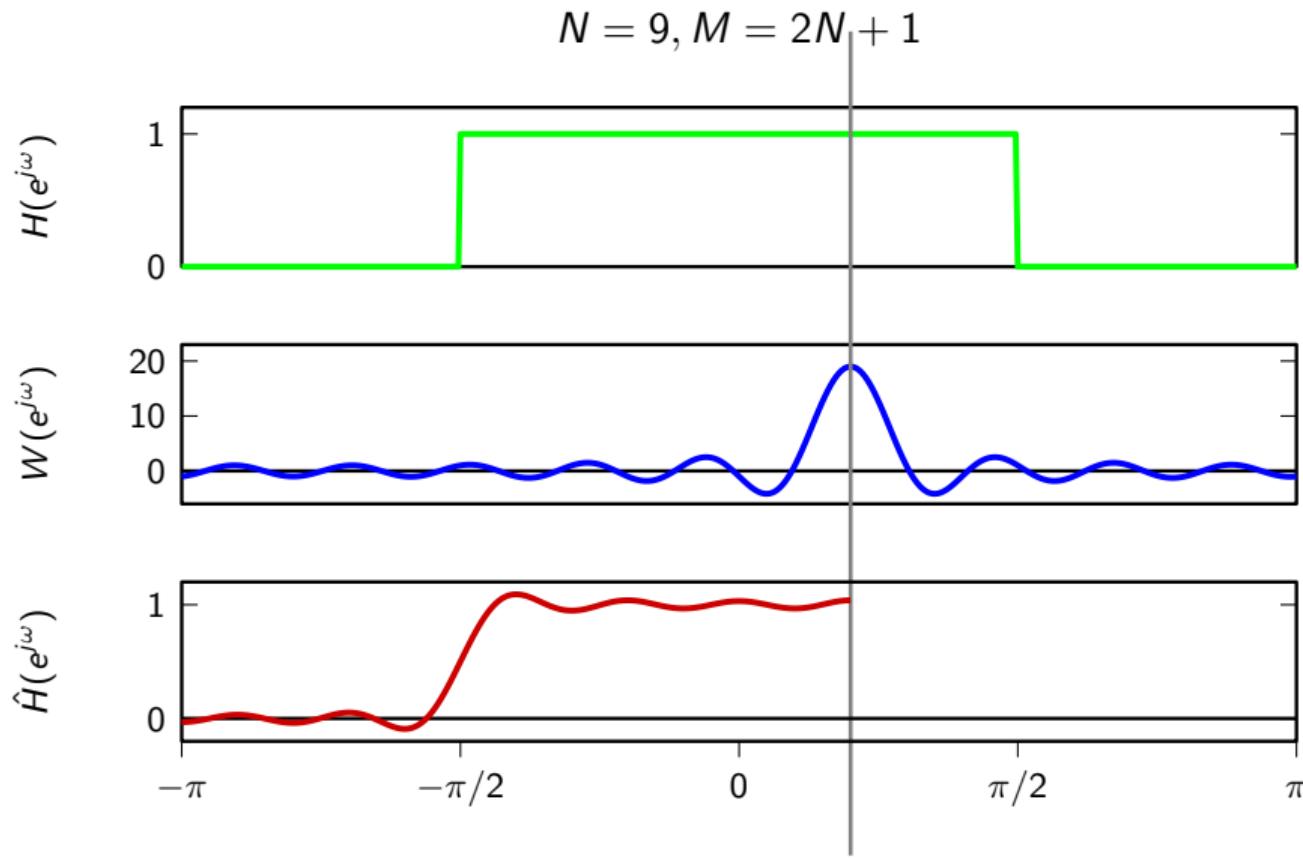


Implicit frequency-domain convolution

$$N = 9, M = 2N + 1$$

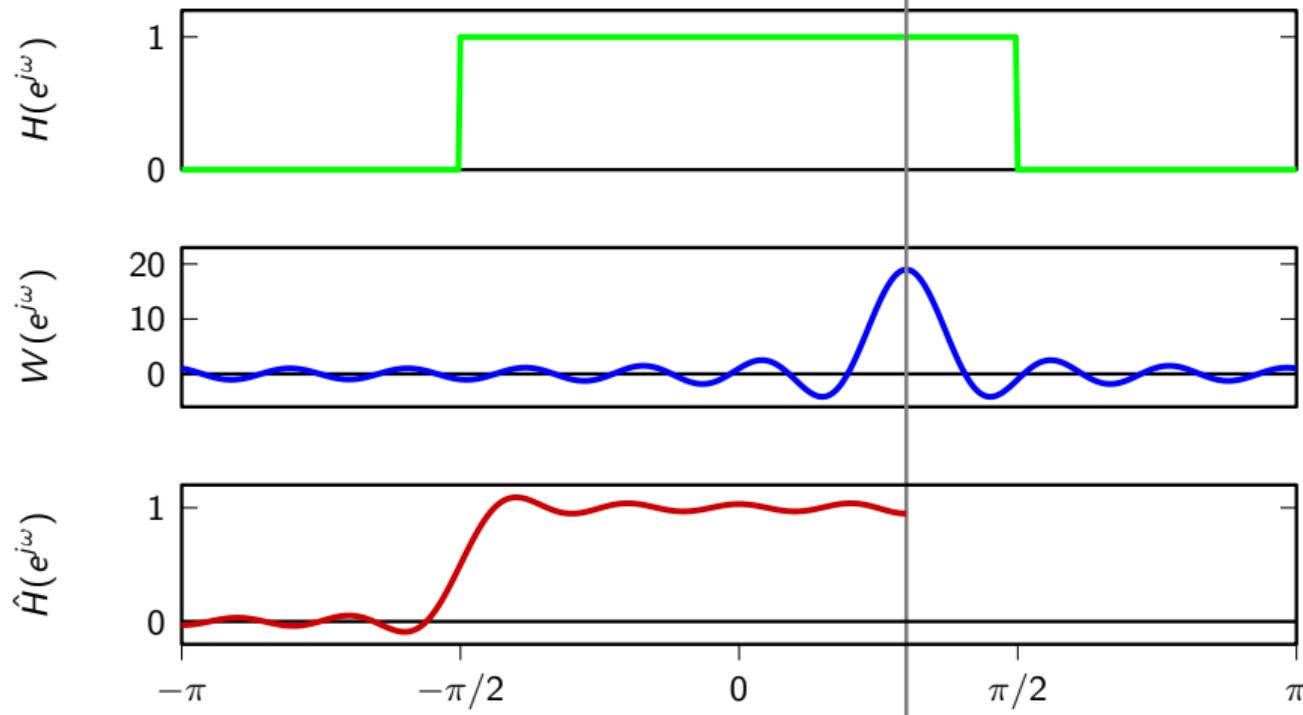


Implicit frequency-domain convolution



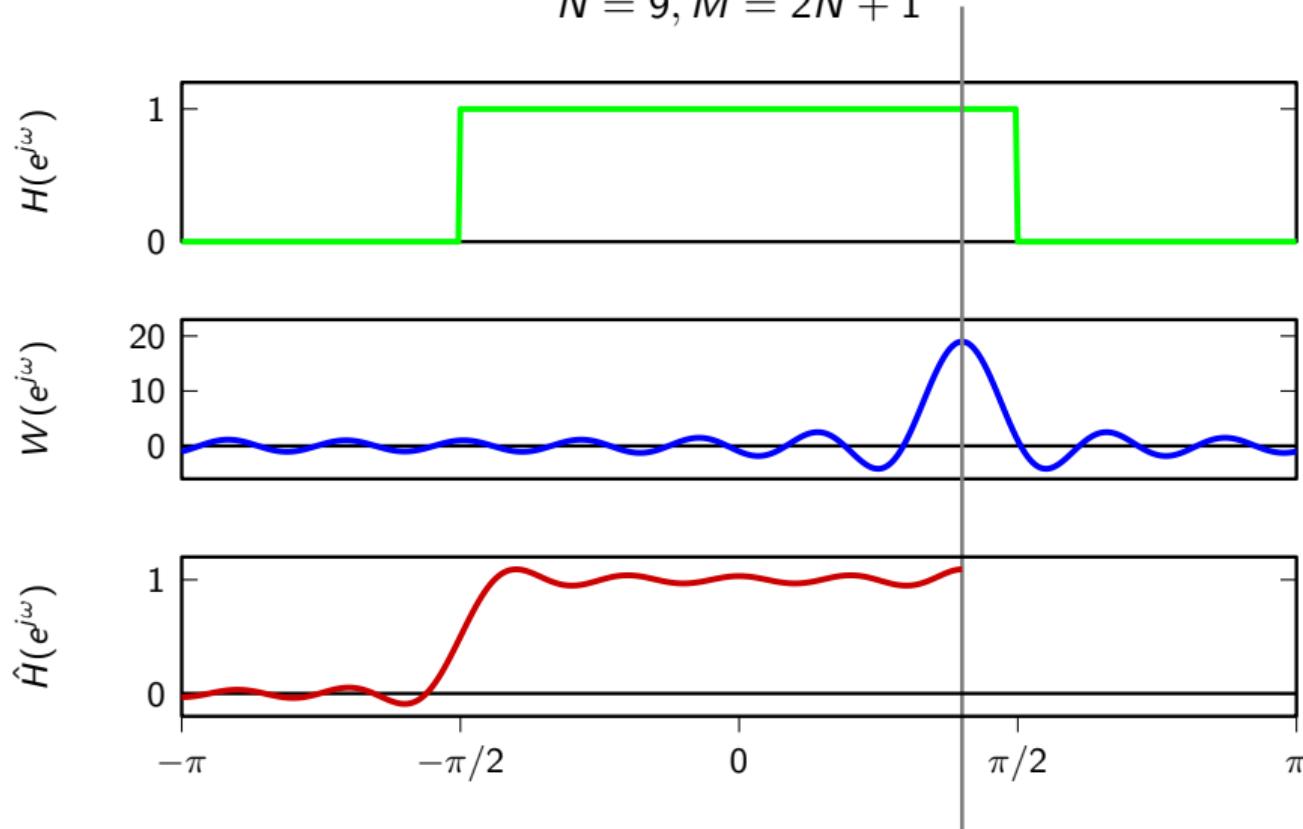
Implicit frequency-domain convolution

$$N = 9, M = 2N + 1$$



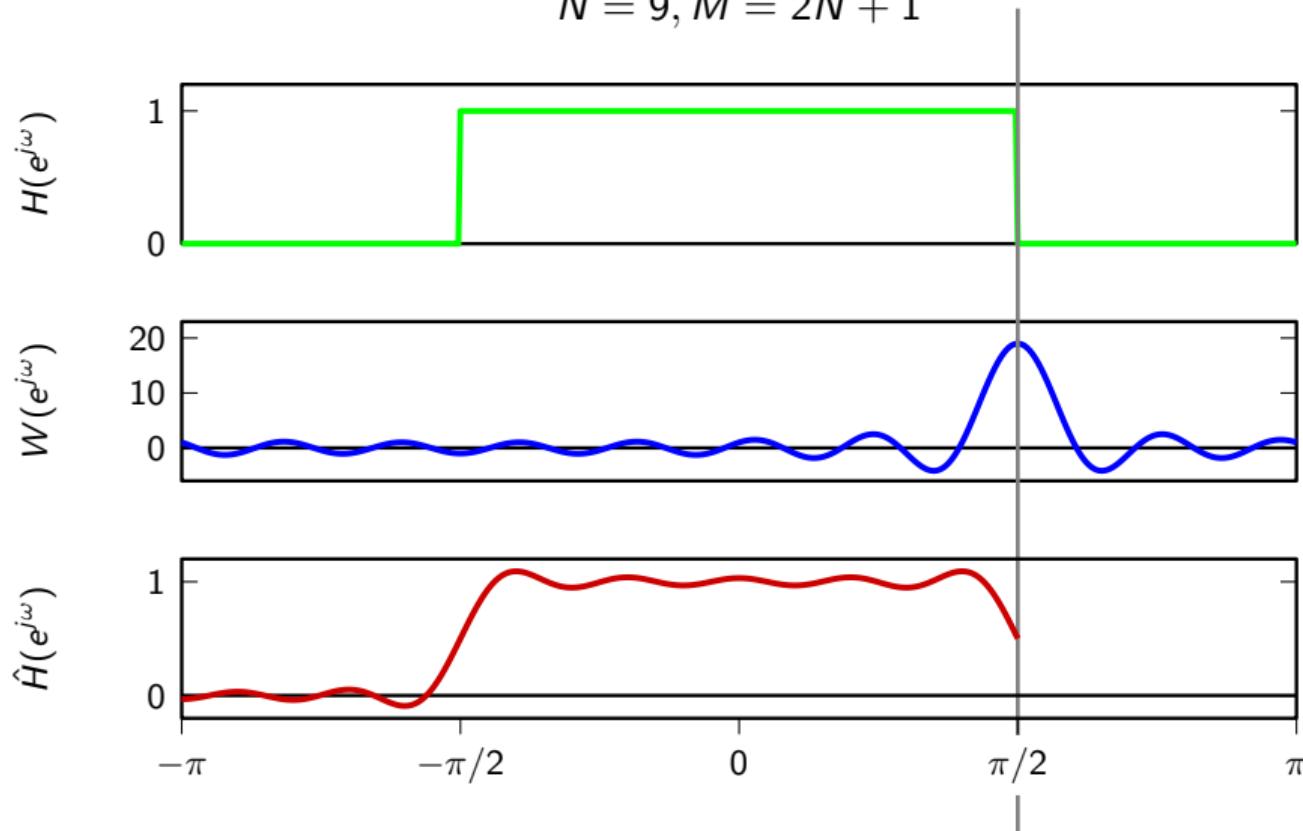
Implicit frequency-domain convolution

$$N = 9, M = 2N + 1$$



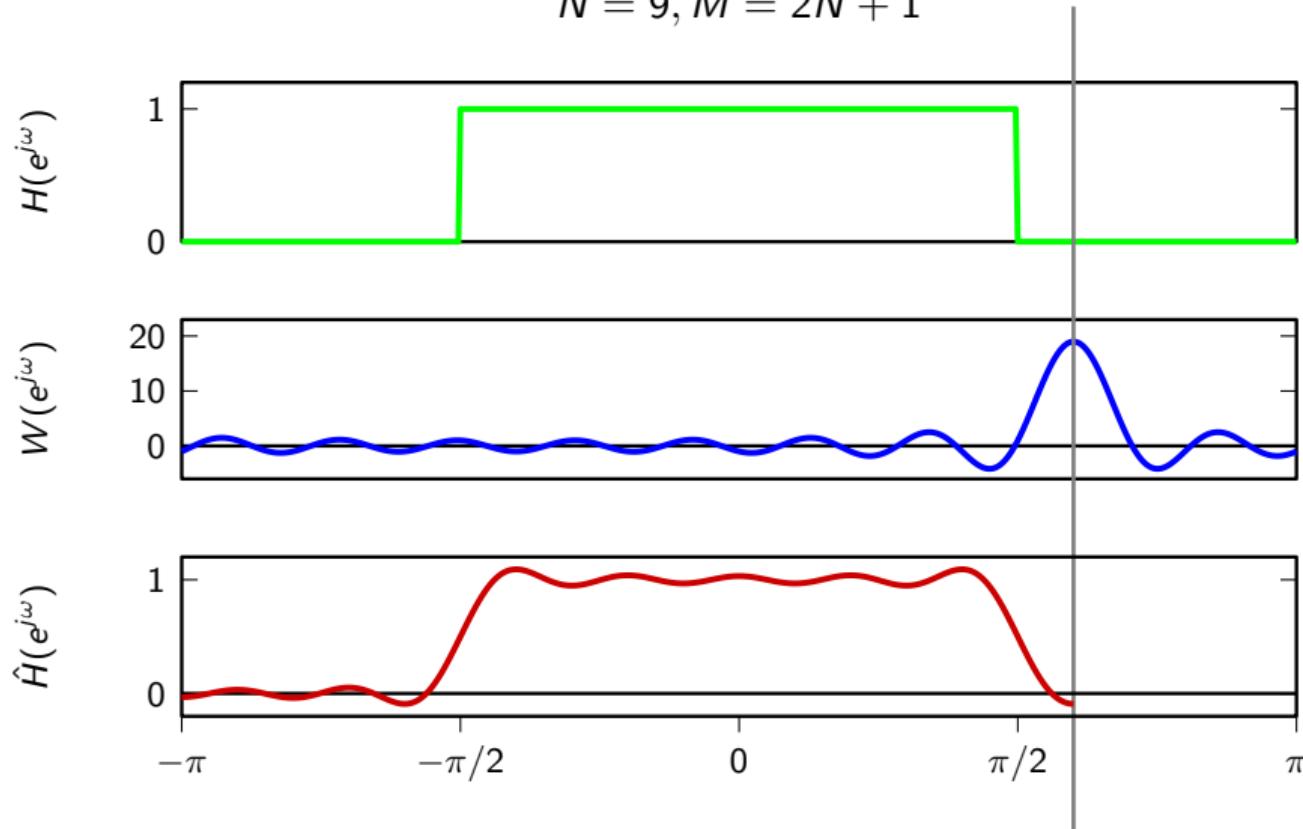
Implicit frequency-domain convolution

$$N = 9, M = 2N + 1$$



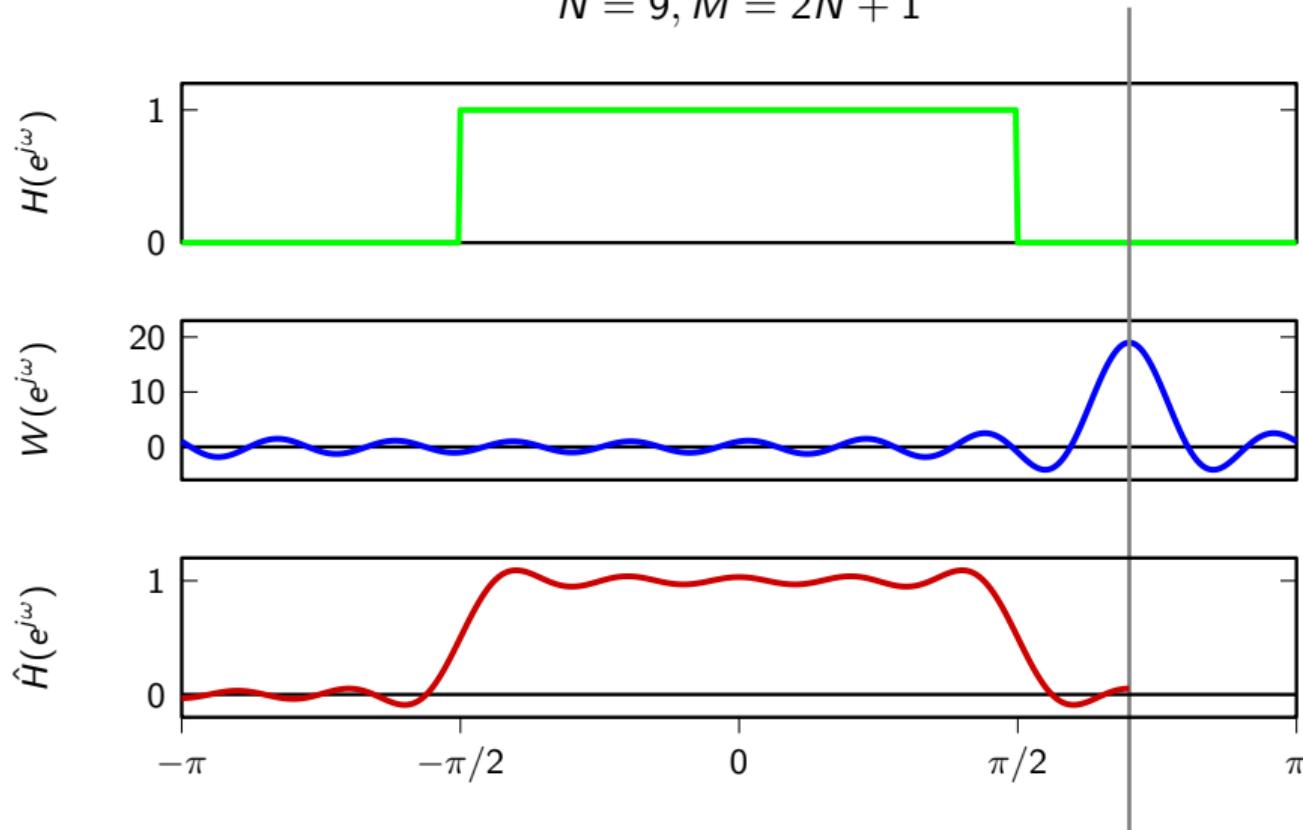
Implicit frequency-domain convolution

$$N = 9, M = 2N + 1$$



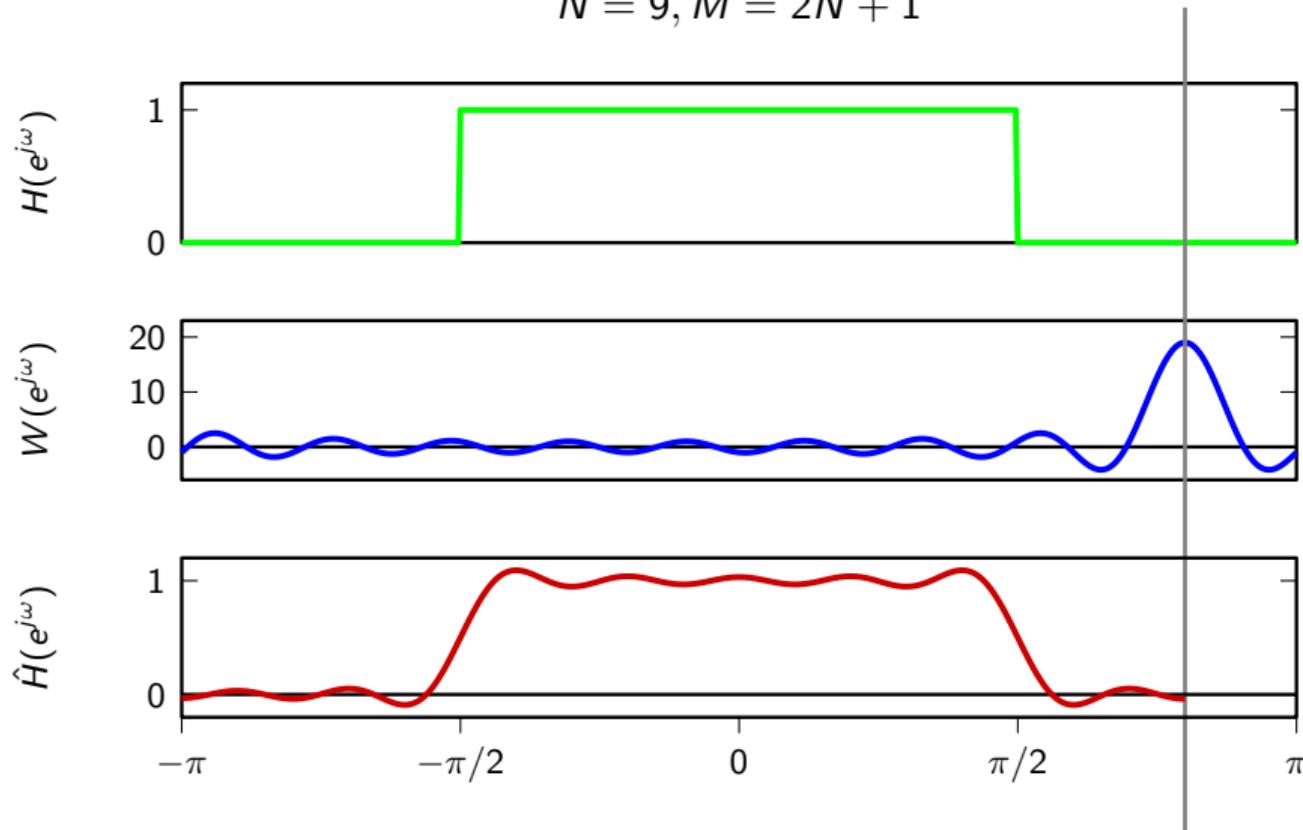
Implicit frequency-domain convolution

$$N = 9, M = 2N + 1$$



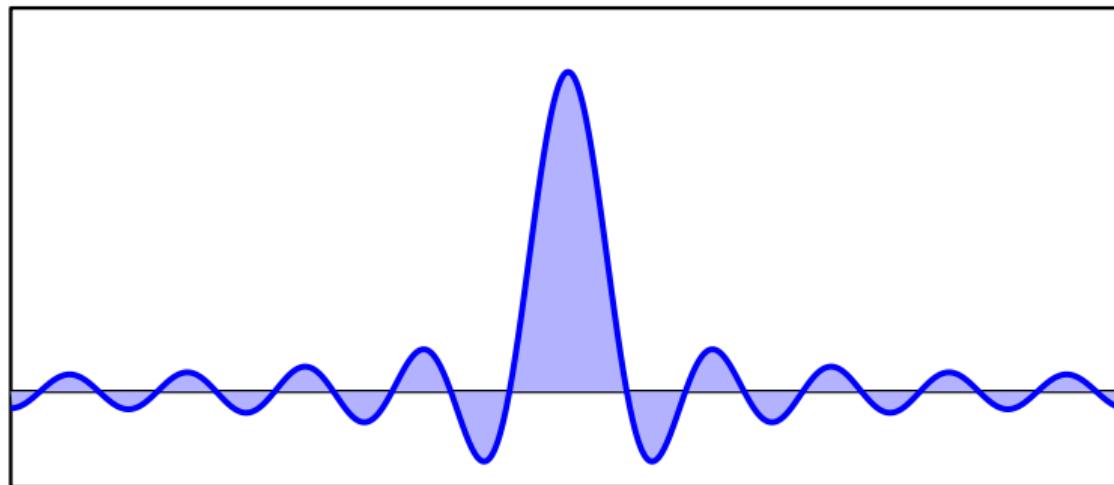
Implicit frequency-domain convolution

$$N = 9, M = 2N + 1$$



Quantifying the Gibbs overshoot – 1

Observation 1: integral of window's transform is independent of N :



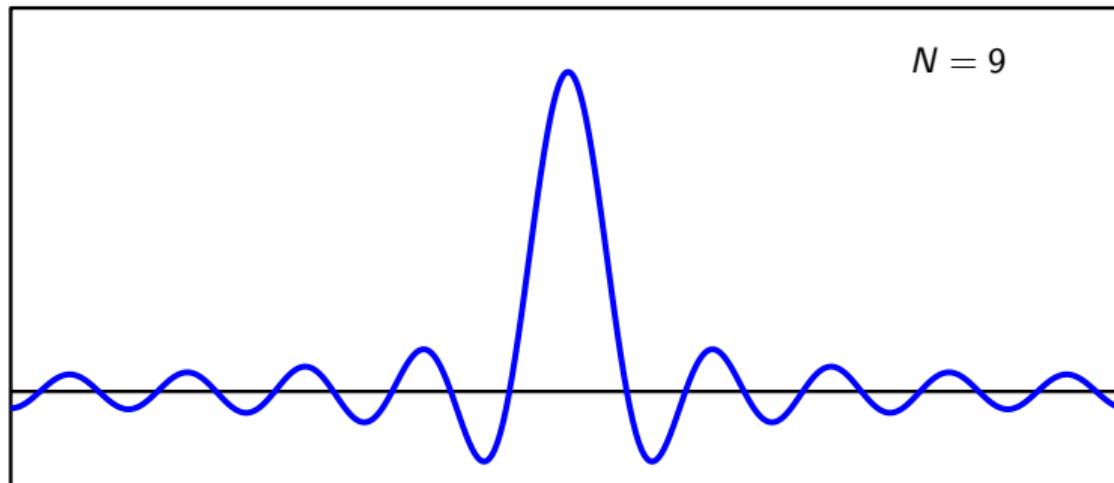
Quantifying the Gibbs overshoot – 1

Observation 1: integral of window's transform is independent of N :

$$\begin{aligned} \int_{-\pi}^{\pi} W(e^{j\omega}) &= \int_{-\pi}^{\pi} \sum_{n=-N}^N e^{-j\omega n} d\omega \\ &= \sum_{n=-N}^N \int_{-\pi}^{\pi} e^{-j\omega n} d\omega \\ &= 2\pi \end{aligned}$$

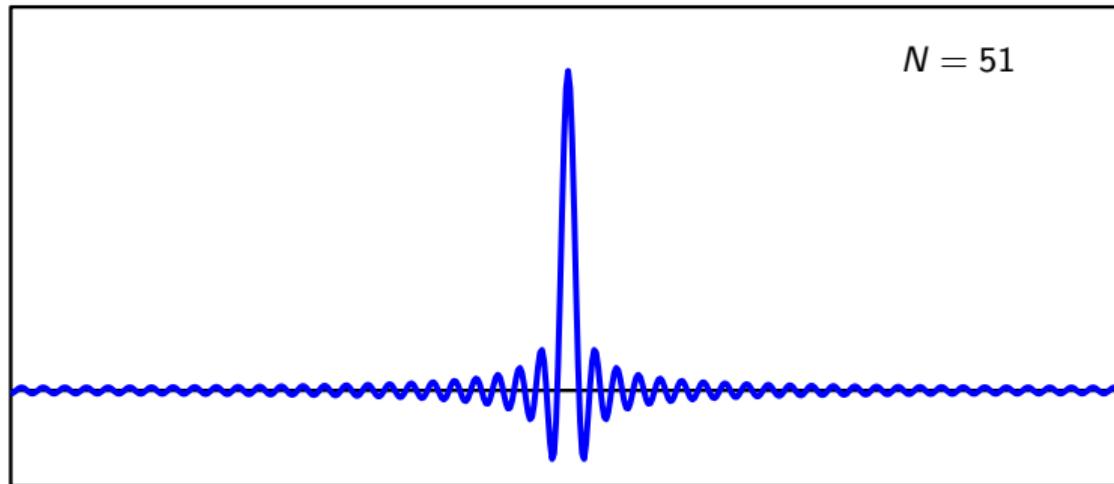
Quantifying the Gibbs overshoot – 2

For large N , the area is concentrated around the midpoint:



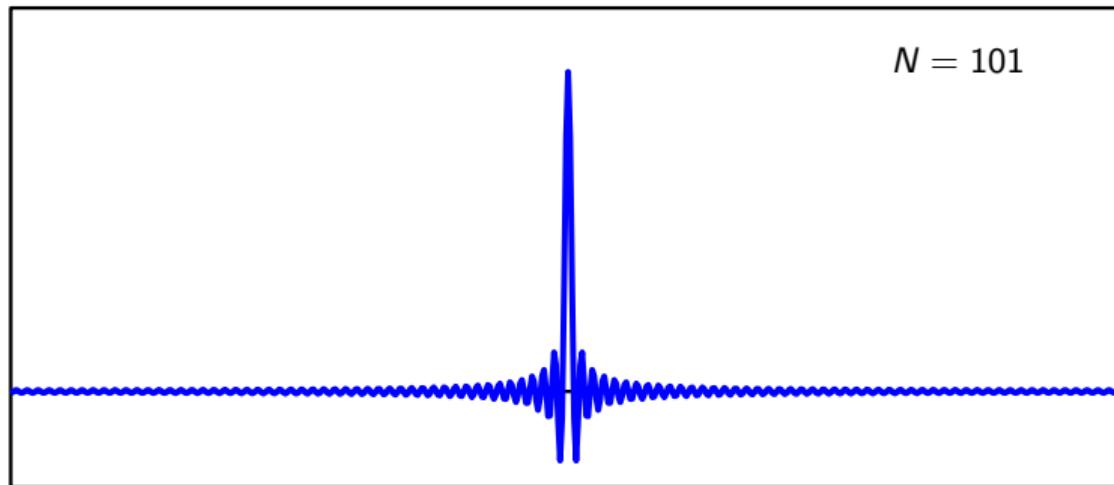
Quantifying the Gibbs overshoot – 2

For large N , the area is concentrated around the midpoint:



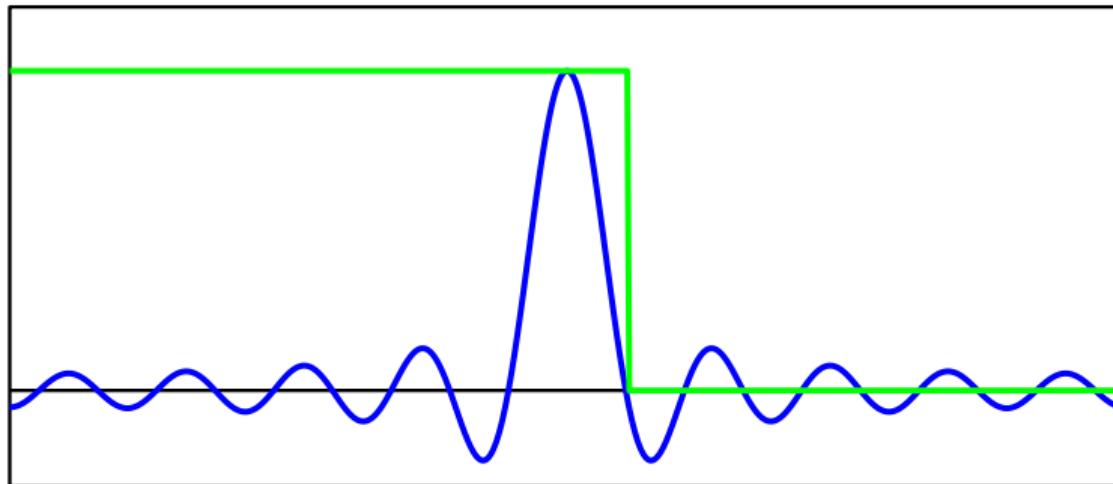
Quantifying the Gibbs overshoot – 2

For large N , the area is concentrated around the midpoint:



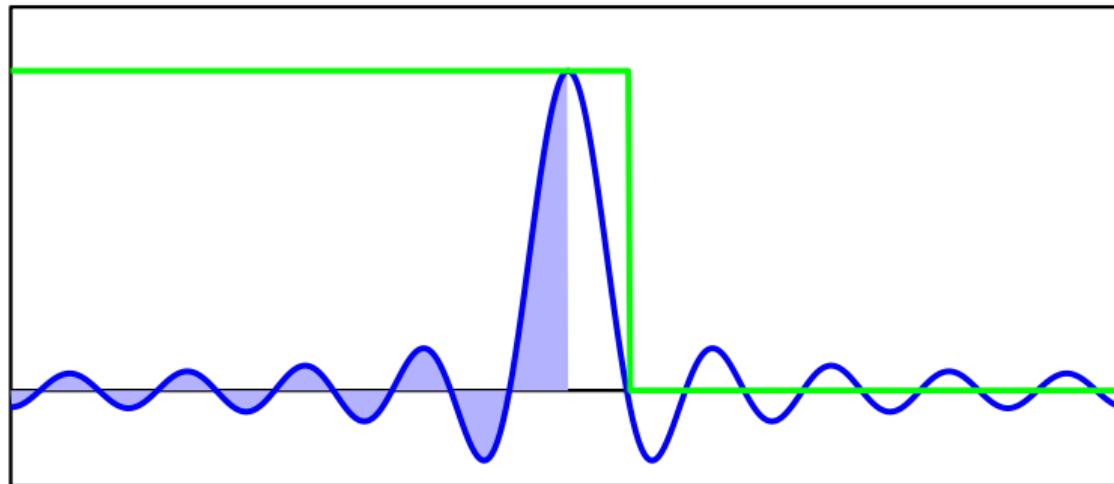
Quantifying the Gibbs overshoot – 3

maximum value of the convolution integral:



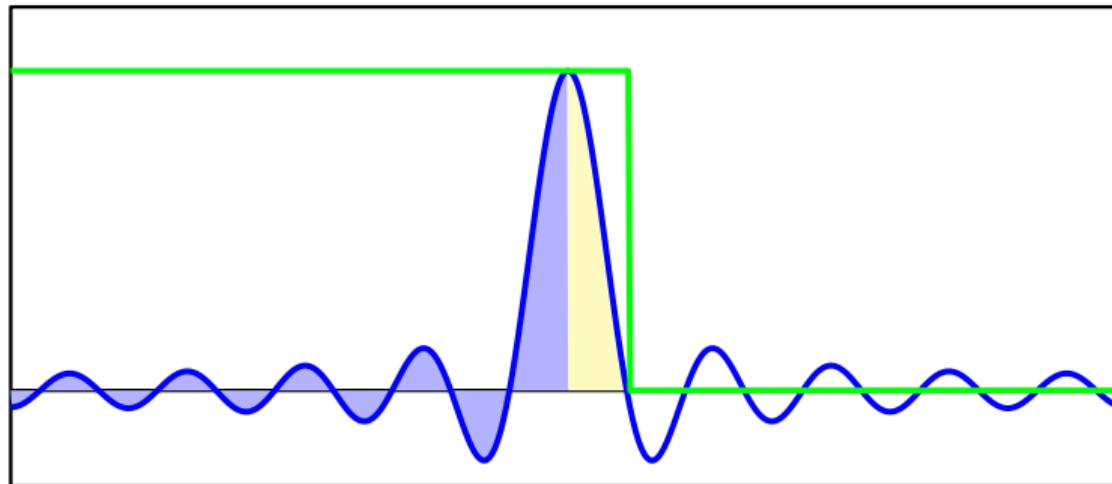
Quantifying the Gibbs overshoot – 3

maximum value of the convolution integral: A



Quantifying the Gibbs overshoot – 3

maximum value of the convolution integral: $A + B$



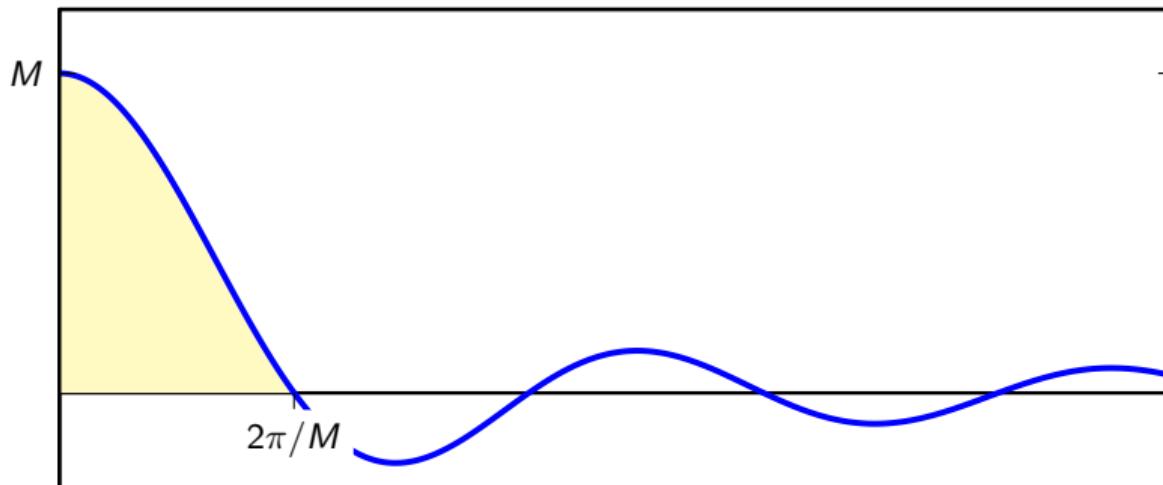
Quantifying the Gibbs overshoot – 4

For large N , A is basically half the total area:

$$A \approx \frac{1}{2} \int_{-\pi}^{\pi} W(e^{j\omega}) = \pi$$

Quantifying the Gibbs overshoot – 5

$$W(e^{j\omega}) = \sin(\omega M/2) / \sin(\omega/2), \quad M = 2N + 1$$

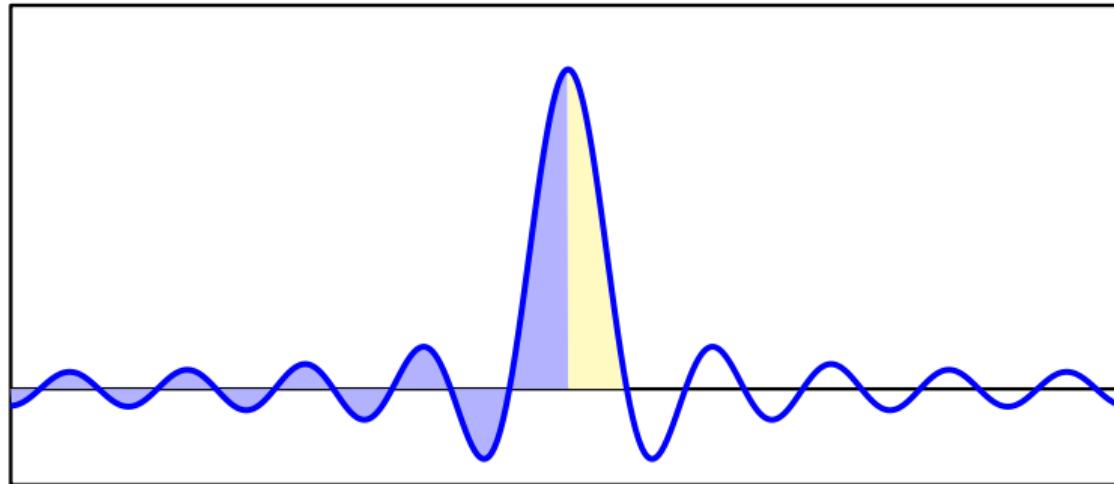


Quantifying the Gibbs overshoot – 5

$$\begin{aligned} \int_0^{2\pi/M} W(e^{j\omega}) &= \int_0^{2\pi/M} \frac{\sin(\omega M/2)}{\sin(\omega/2)} d\omega \\ &= \frac{2}{M} \int_0^{\pi} \frac{\sin t}{\sin(t/M)} dt \quad (t = M\omega/2) \\ &\approx \frac{2}{M} \int_0^{\pi} \frac{\sin t}{(t/M)} dt \quad (\sin t/M \approx t/M \text{ for } M \text{ large}) \\ &= 2 \int_0^{\pi} \frac{\sin t}{t} dt \\ &\approx 2\pi \times 0.589 \quad \text{independent of } M! \end{aligned}$$

Quantifying the Gibbs overshoot – 6

maximum value of the convolution: $(A + B)/2\pi$



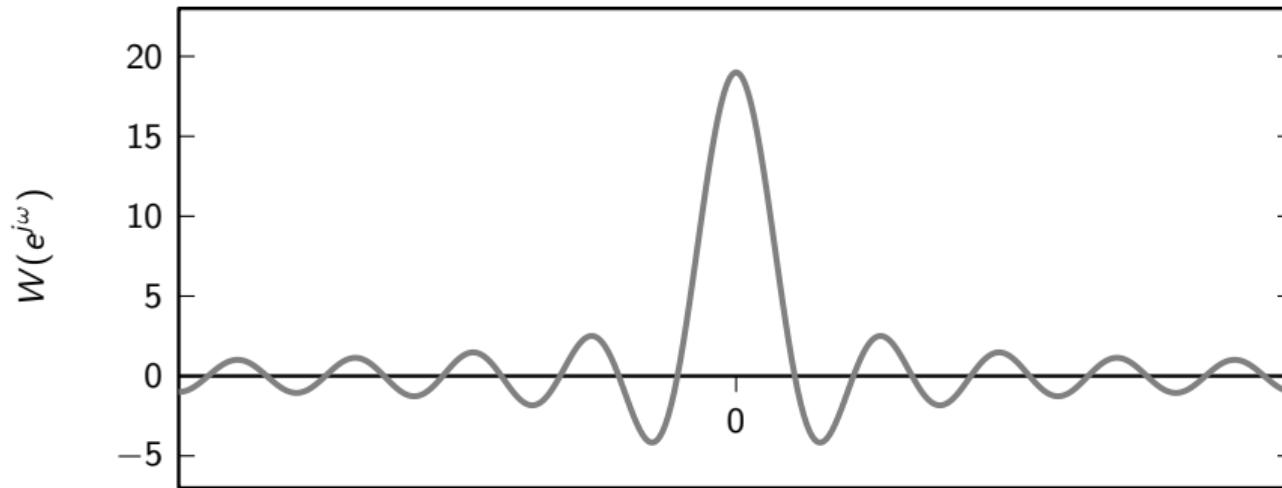
Quantifying the Gibbs overshoot – 6

$$A \approx \frac{1}{2} \int_{-\pi}^{\pi} W(e^{j\omega}) = \pi$$

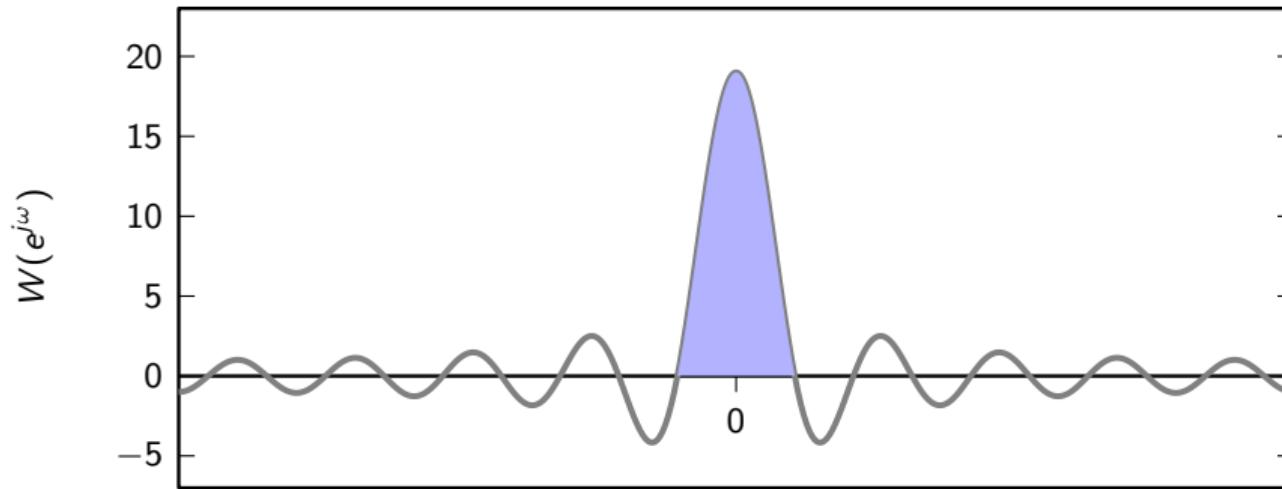
$$B \approx 2\pi \times 0.589$$

$$(A + B)/2\pi \approx 1.09$$

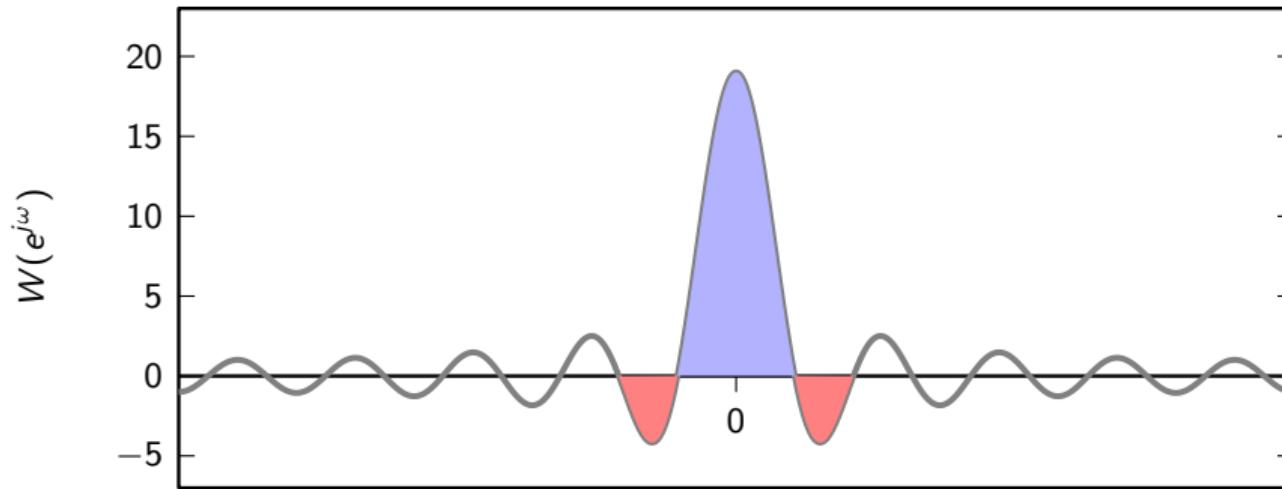
Mainlobe and sidelobes



Mainlobe and sidelobes



Mainlobe and sidelobes



What if we change the window?

We want:

- ▶ narrow mainlobe so that transition is sharp
- ▶ small sidelobe so Gibbs error is small
- ▶ short window so FIR is efficient

What if we change the window?

We want:

- ▶ narrow mainlobe so that transition is sharp
- ▶ small sidelobe so Gibbs error is small
- ▶ short window so FIR is efficient

What if we change the window?

We want:

- ▶ narrow mainlobe so that transition is sharp
- ▶ small sidelobe so Gibbs error is small
- ▶ short window so FIR is efficient

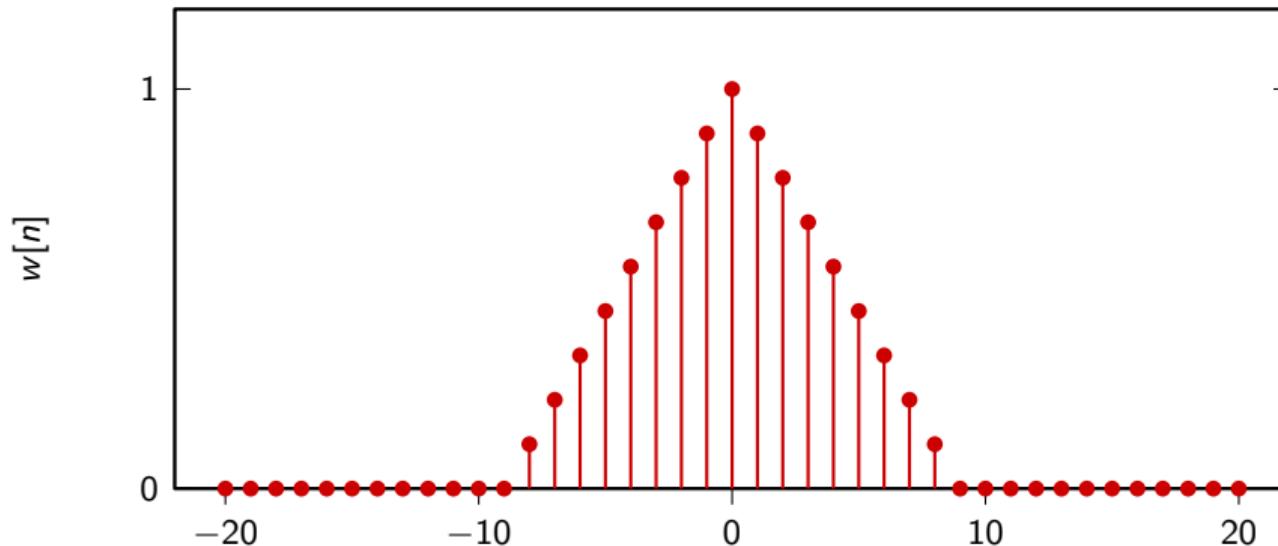
What if we change the window?

We want:

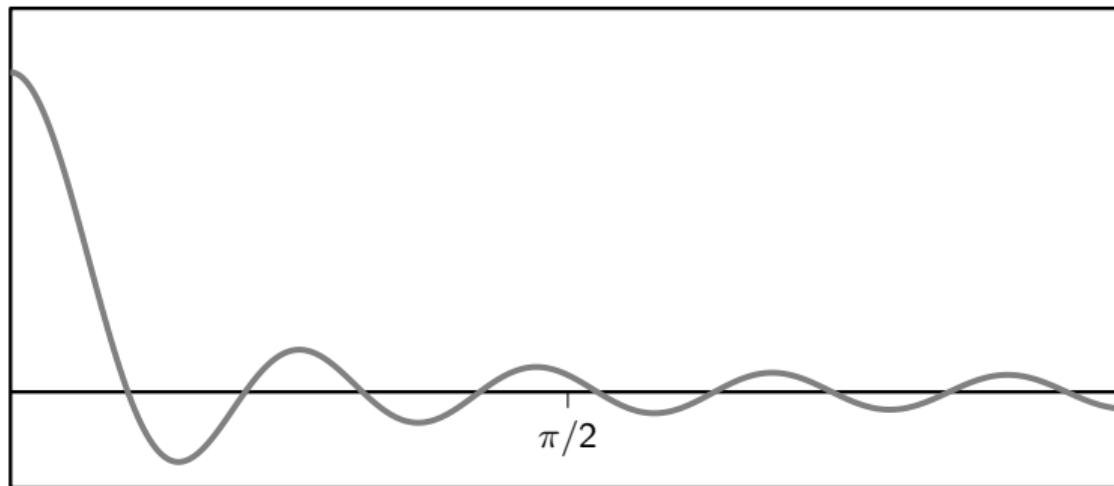
- ▶ narrow mainlobe so that transition is sharp
- ▶ small sidelobe so Gibbs error is small
- ▶ short window so FIR is efficient

very conflicting requirements!

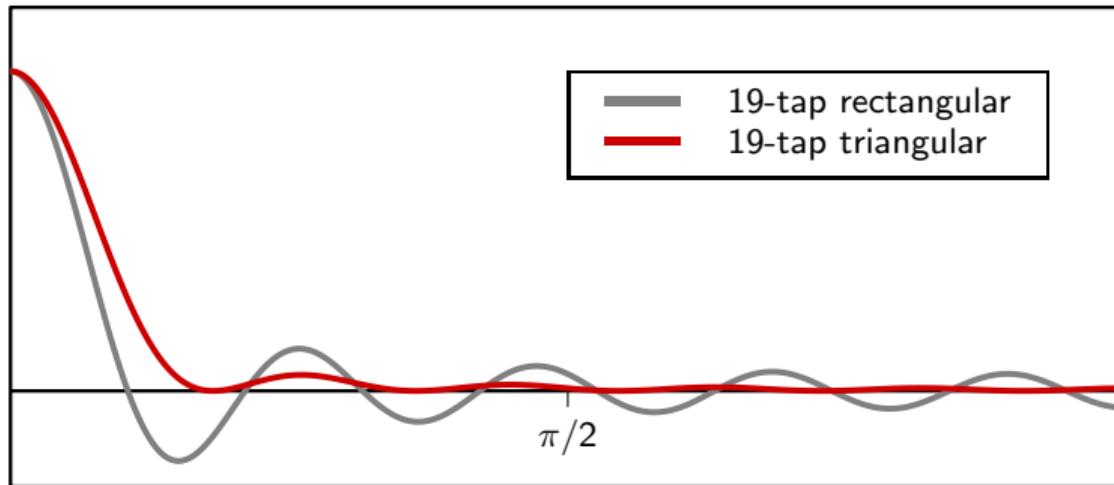
Triangular window



Rectangular vs Triangular Window



Rectangular vs Triangular Window



Window method: pros and cons

Pros:

- ▶ extremely simple
- ▶ minimizes MSE

Cons:

- ▶ can't control max error (Gibbs)
- ▶ must know the impulse response (not easy for arbitrary frequency responses)

Frequency sampling

Idea #2:

- ▶ draw desired frequency response $H(e^{j\omega})$
- ▶ take M samples $S[k] = H(e^{j\omega_k})$, $\omega_k = (2\pi/M)k$
- ▶ compute inverse DFT of the $S[k]$ values
- ▶ use $s[n]$ to build impulse response $\hat{h}[n]$

Frequency sampling

Idea #2:

- ▶ draw desired frequency response $H(e^{j\omega})$
- ▶ take M samples $S[k] = H(e^{j\omega_k})$, $\omega_k = (2\pi/M)k$
- ▶ compute inverse DFT of the $S[k]$ values
- ▶ use $s[n]$ to build impulse response $\hat{h}[n]$

Frequency sampling

Idea #2:

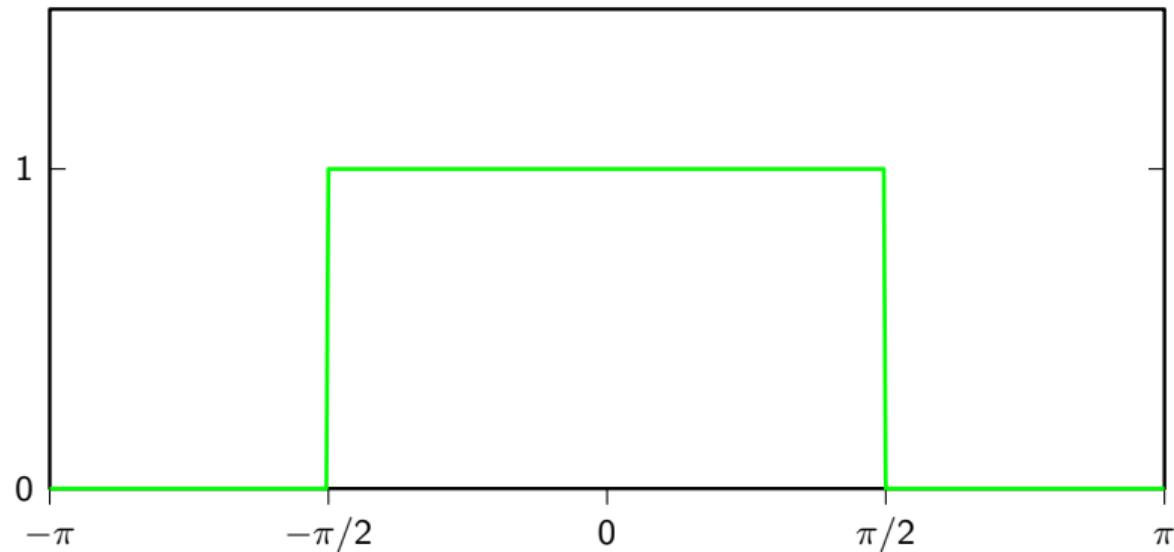
- ▶ draw desired frequency response $H(e^{j\omega})$
- ▶ take M samples $S[k] = H(e^{j\omega_k})$, $\omega_k = (2\pi/M)k$
- ▶ compute inverse DFT of the $S[k]$ values
- ▶ use $s[n]$ to build impulse response $\hat{h}[n]$

Frequency sampling

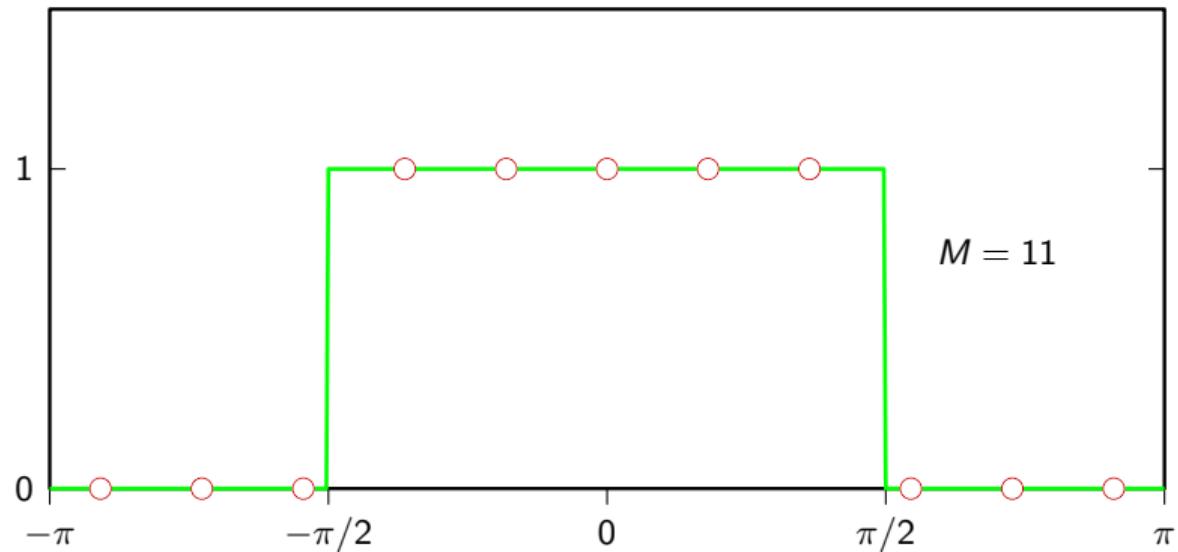
Idea #2:

- ▶ draw desired frequency response $H(e^{j\omega})$
- ▶ take M samples $S[k] = H(e^{j\omega_k})$, $\omega_k = (2\pi/M)k$
- ▶ compute inverse DFT of the $S[k]$ values
- ▶ use $s[n]$ to build impulse response $\hat{h}[n]$

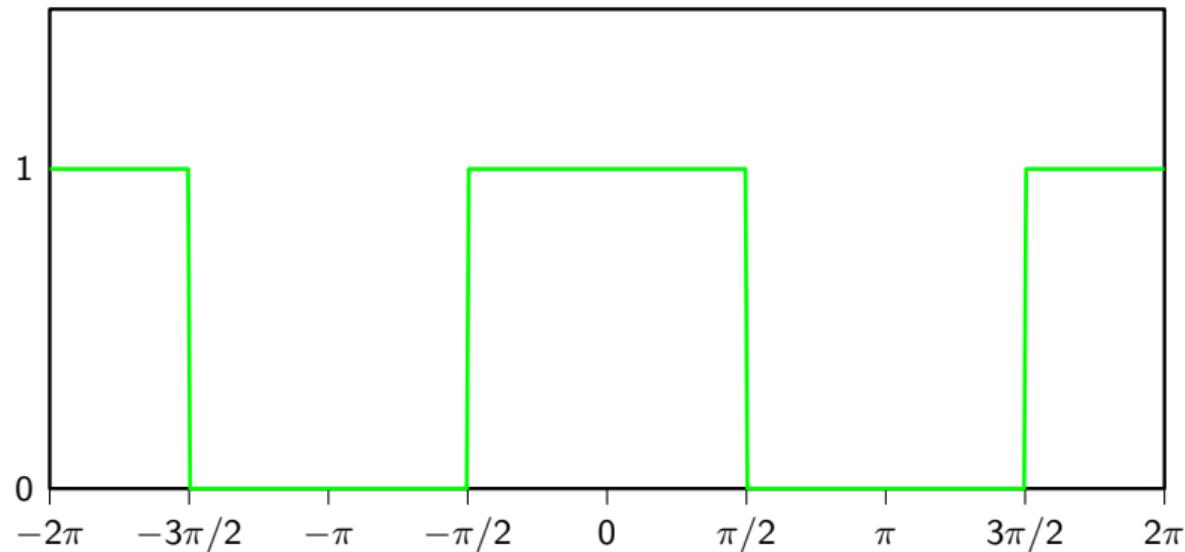
Frequency sampling: desired response



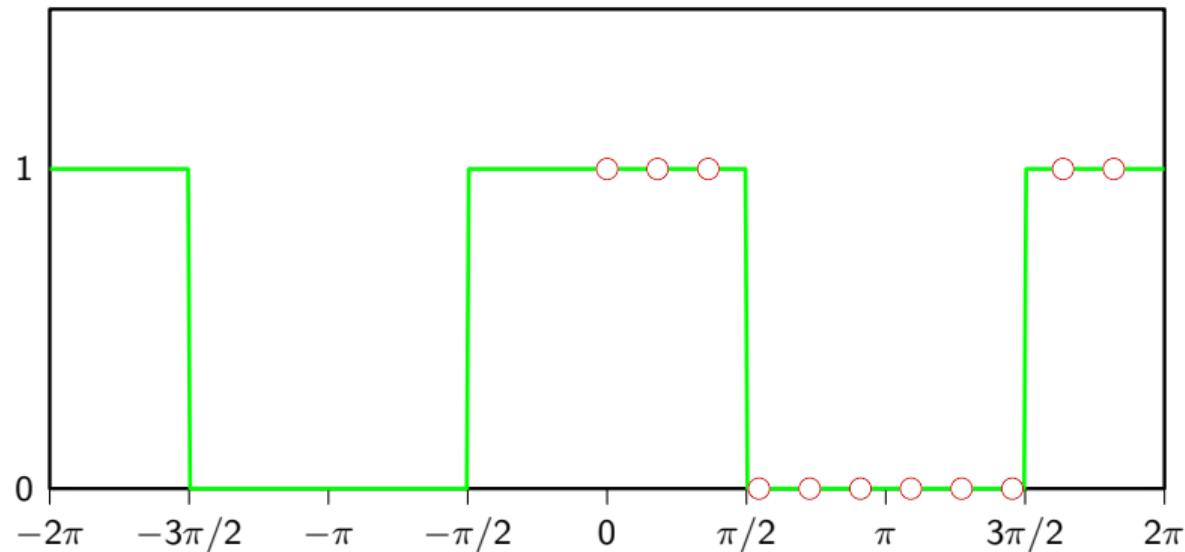
Frequency sampling: desired response



Frequency sampling: from DTFT to DFT

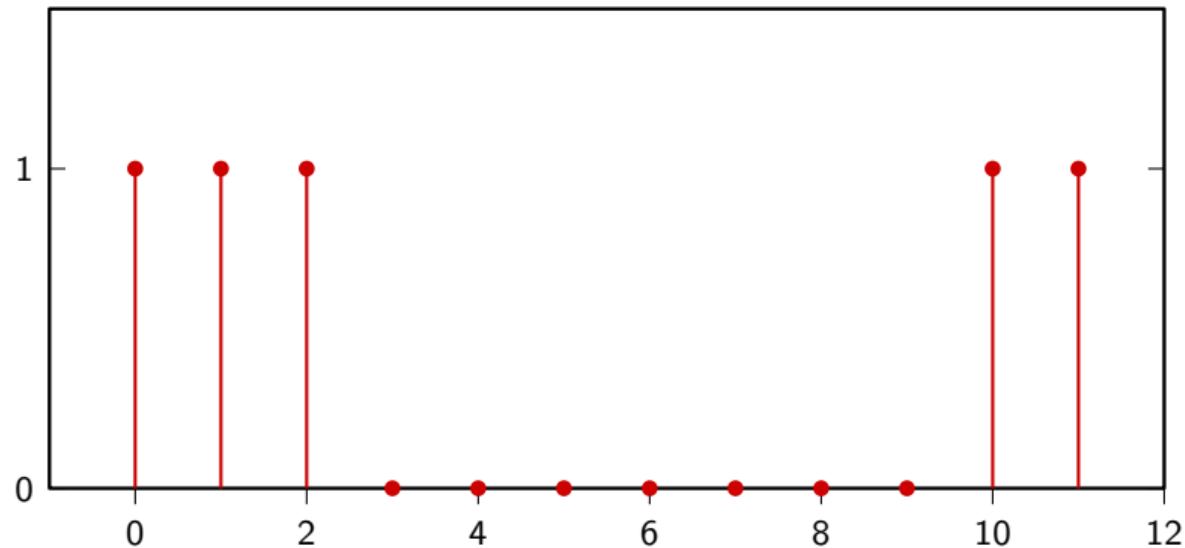


Frequency sampling: from DTFT to DFT



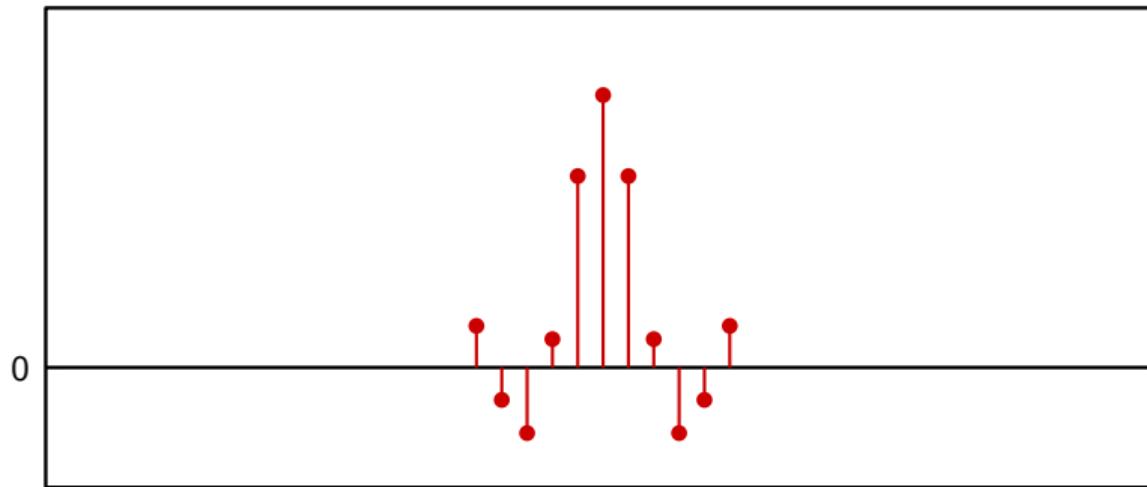
Frequency sampling: DFT samples

$S[k]$



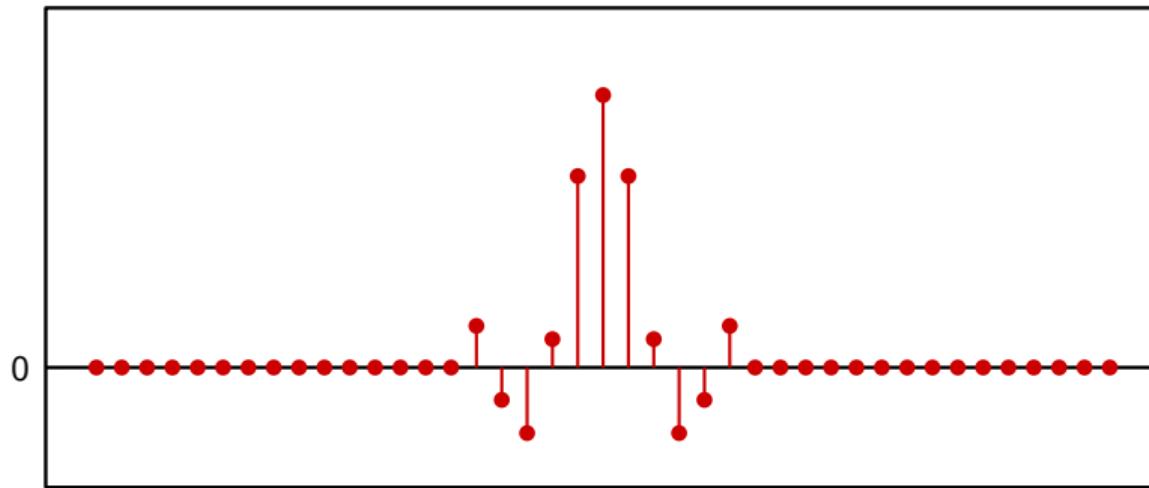
Frequency sampling: impulse response from IDFT

$s[n]$



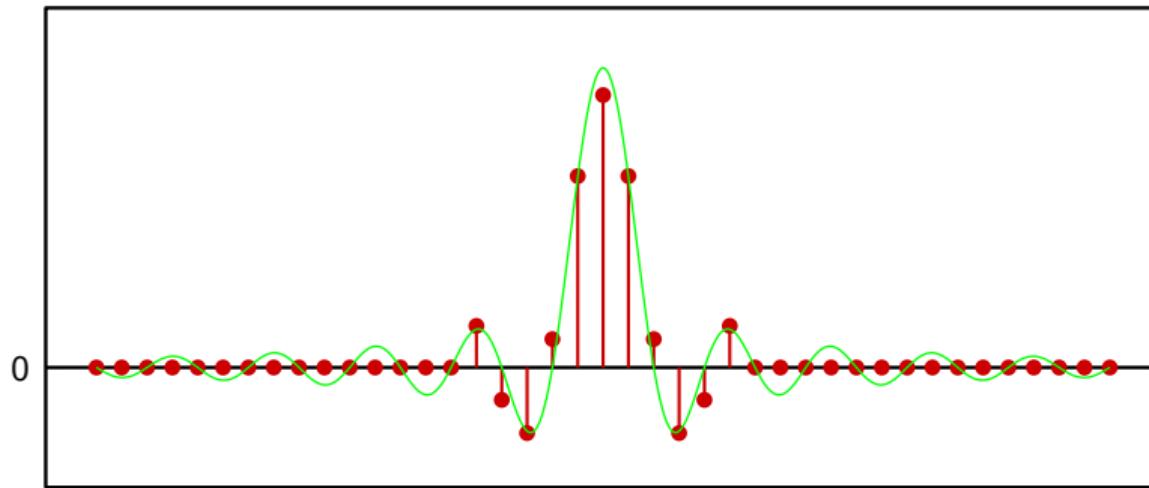
Frequency sampling: impulse response from IDFT

$$\hat{h}[n]$$



Frequency sampling: impulse response from IDFT

$$\hat{h}[n]$$



Frequency sampling: what happens in the time domain

$$\hat{h}[n] = \begin{cases} s[n] & \text{if } 0 \leq n < M \\ 0 & \text{otherwise} \end{cases}$$

$$s[n] = \text{IDFT}\{S[k]\}$$

$$S[k] = H(e^{j\frac{2\pi}{M}k})$$

Frequency sampling: what happens in the time domain

$$\hat{h}[n] = \begin{cases} s[n] & \text{if } 0 \leq n < M \\ 0 & \text{otherwise} \end{cases}$$

$$s[n] = \text{IDFT} \{ S[k] \}$$

$$S[k] = H(e^{j\frac{2\pi}{M}k})$$

Frequency sampling: what happens in the time domain

$$\hat{h}[n] = \begin{cases} s[n] & \text{if } 0 \leq n < M \\ 0 & \text{otherwise} \end{cases}$$

$$s[n] = \text{IDFT} \{ S[k] \}$$

$$S[k] = H(e^{j\frac{2\pi}{M}k})$$

Frequency sampling: what happens in the time domain

$$\begin{aligned}s[n] &= \frac{1}{M} \sum_{k=0}^{M-1} S[k] e^{j \frac{2\pi}{M} nk} \\&= \frac{1}{M} \sum_{k=0}^{M-1} H(e^{j \frac{2\pi}{M} k}) e^{j \frac{2\pi}{M} nk} \\&= \frac{1}{M} \sum_{k=0}^{M-1} \left(\sum_{m=-\infty}^{\infty} h[m] e^{-j \frac{2\pi}{M} k m} \right) e^{j \frac{2\pi}{M} nk} \\&= \sum_{m=-\infty}^{\infty} h[m] \frac{1}{M} \sum_{k=0}^{M-1} e^{-j \frac{2\pi}{M} (m-n) k}\end{aligned}$$

Frequency sampling: what happens in the time domain

$$\begin{aligned}s[n] &= \frac{1}{M} \sum_{k=0}^{M-1} S[k] e^{j\frac{2\pi}{M}nk} \\&= \frac{1}{M} \sum_{k=0}^{M-1} H(e^{j\frac{2\pi}{M}k}) e^{j\frac{2\pi}{M}nk} \\&= \frac{1}{M} \sum_{k=0}^{M-1} \left(\sum_{m=-\infty}^{\infty} h[m] e^{-j\frac{2\pi}{M}k} \right) e^{j\frac{2\pi}{M}nk} \\&= \sum_{m=-\infty}^{\infty} h[m] \frac{1}{M} \sum_{k=0}^{M-1} e^{-j\frac{2\pi}{M}(m-n)k}\end{aligned}$$

Frequency sampling: what happens in the time domain

$$\begin{aligned}s[n] &= \frac{1}{M} \sum_{k=0}^{M-1} S[k] e^{j \frac{2\pi}{M} nk} \\&= \frac{1}{M} \sum_{k=0}^{M-1} H(e^{j \frac{2\pi}{M} k}) e^{j \frac{2\pi}{M} nk} \\&= \frac{1}{M} \sum_{k=0}^{M-1} \left(\sum_{m=-\infty}^{\infty} h[m] e^{-j \frac{2\pi}{M} k m} \right) e^{j \frac{2\pi}{M} nk} \\&= \sum_{m=-\infty}^{\infty} h[m] \frac{1}{M} \sum_{k=0}^{M-1} e^{-j \frac{2\pi}{M} (m-n) k}\end{aligned}$$

a familiar result

$$\sum_{k=0}^{M-1} e^{-j\frac{2\pi}{M}(m-n)k} = \begin{cases} M & \text{if } m - n \text{ multiple of } M \\ 0 & \text{otherwise} \end{cases}$$

Frequency sampling: what happens in the time domain

$$\begin{aligned}s[n] &= \sum_{m=-\infty}^{\infty} h[m] \delta[(m - n) \bmod M] \\&= \sum_{m=-\infty}^{\infty} h[n + mM]\end{aligned}$$

sampling in the frequency domain results in periodization in the time domain

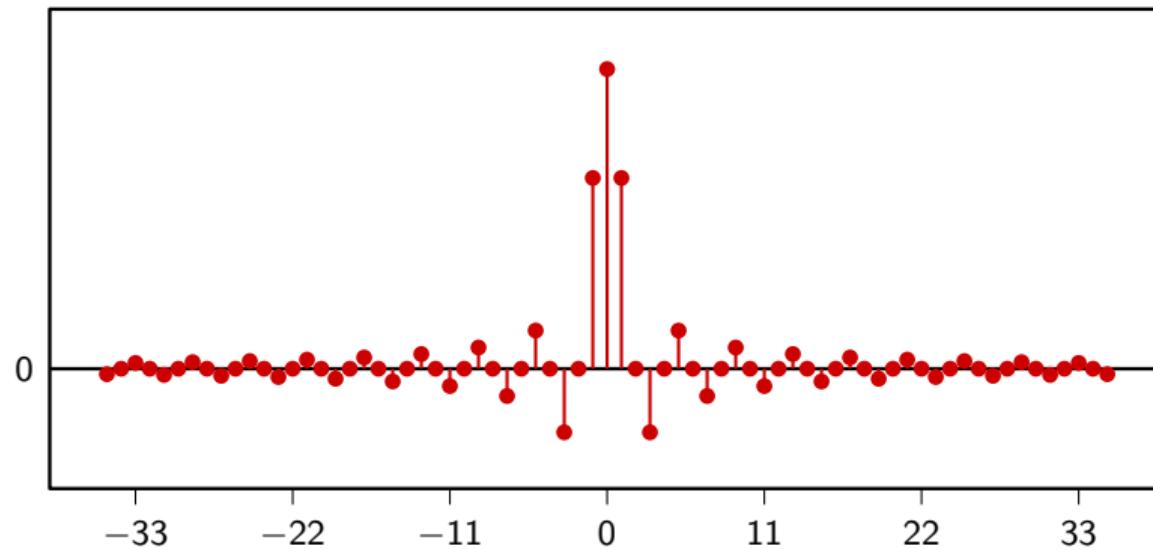
Frequency sampling: what happens in the time domain

$$\begin{aligned}s[n] &= \sum_{m=-\infty}^{\infty} h[m] \delta[(m - n) \bmod M] \\&= \sum_{m=-\infty}^{\infty} h[n + mM]\end{aligned}$$

sampling in the frequency domain results in periodization in the time domain

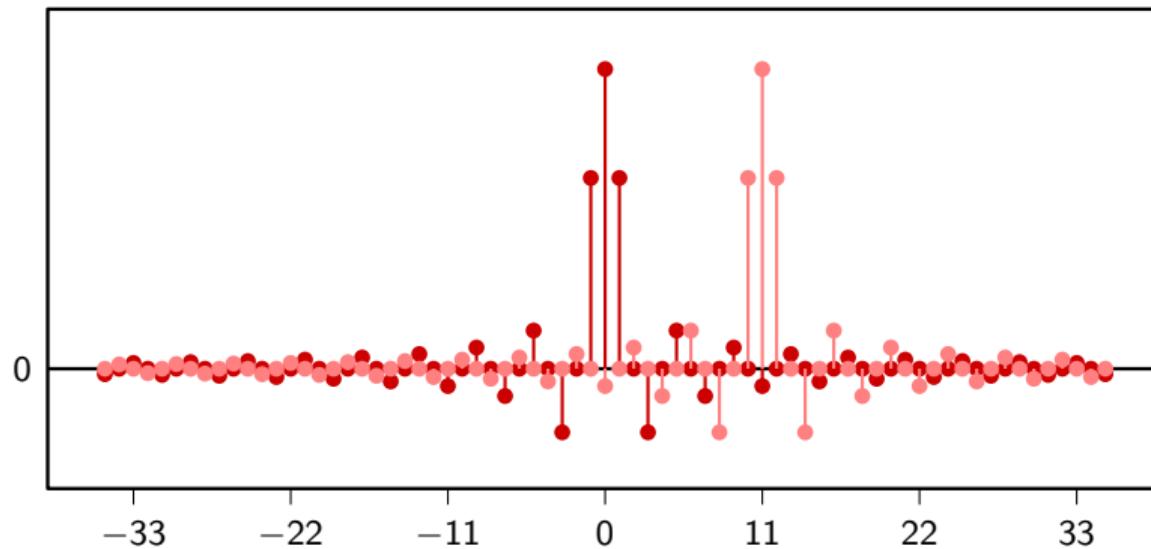
Frequency sampling: impulse response from IDFT

$h[n]$



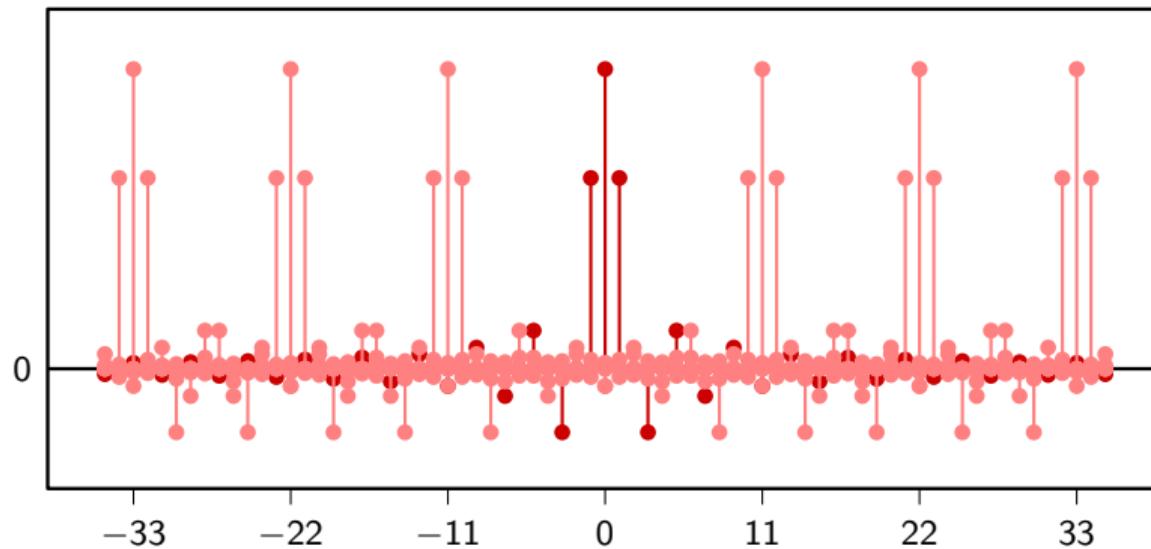
Frequency sampling: impulse response from IDFT

$h[n]$



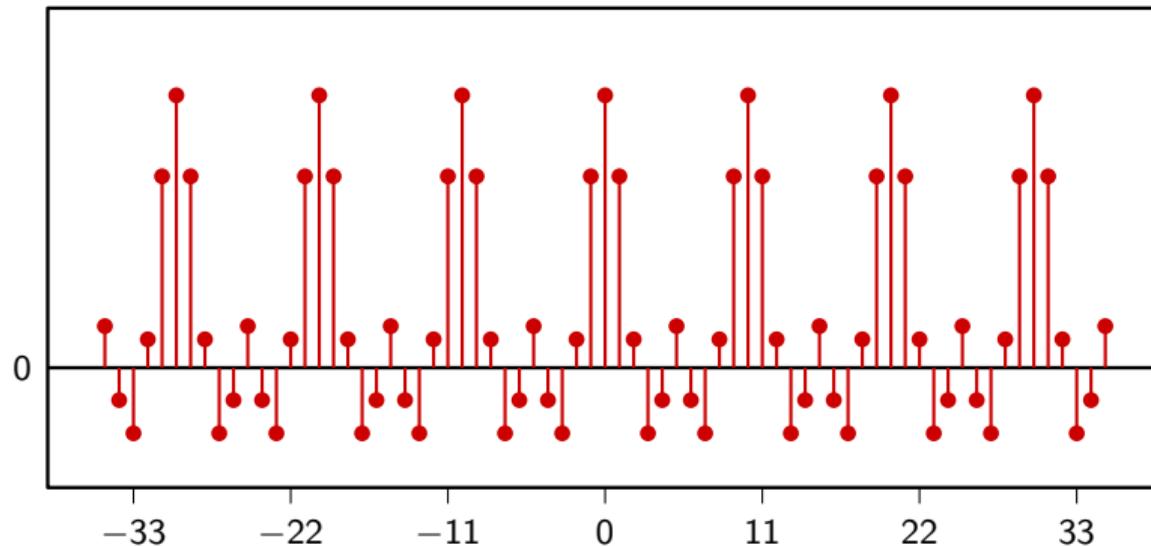
Frequency sampling: impulse response from IDFT

$$\sum_{m=-\infty}^{\infty} h[n + mM]$$

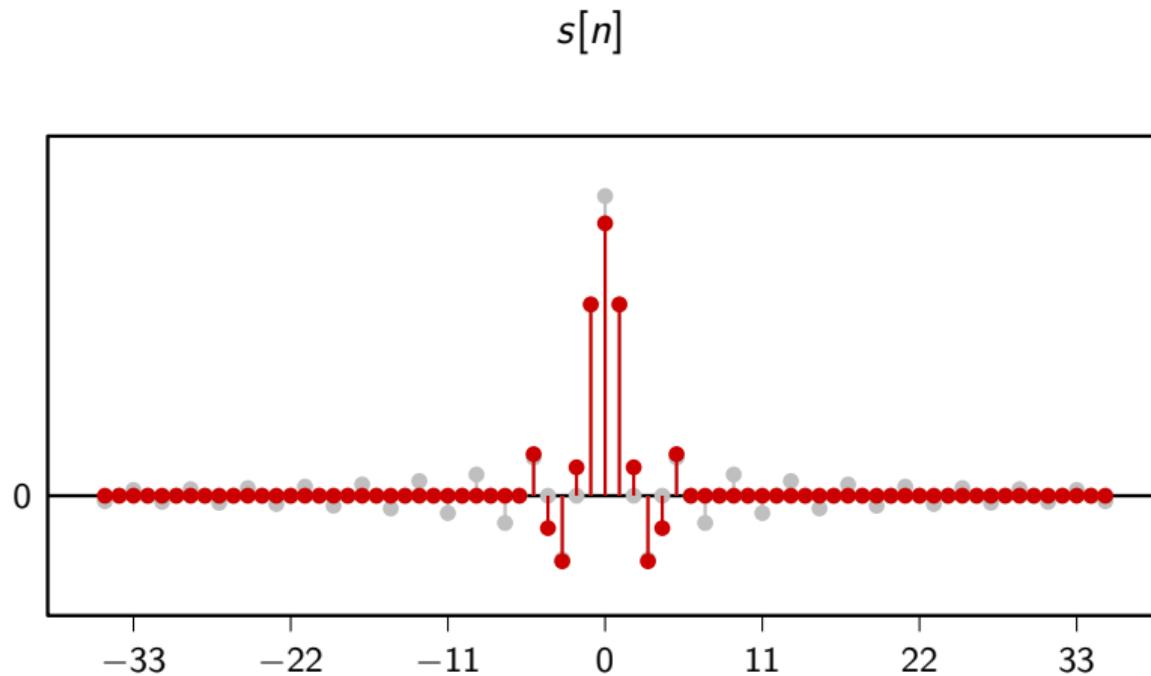


Frequency sampling: impulse response from IDFT

$$\sum_{m=-\infty}^{\infty} h[n + mM]$$



Frequency sampling: impulse response from IDFT



Frequency sampling: what happens in the frequency domain

what is the frequency response $\hat{H}(e^{j\omega})$?

- ▶ $s[n]$: length- M signal
- ▶ $\hat{h}[n]$: finite-support infinite sequence from $s[n]$
- ▶ DFT coefficients $S[k]$ are known
- ▶ use the DFT to DTFT result for finite-support sequences: Lagrangian interpolation

Frequency sampling: what happens in the frequency domain

what is the frequency response $\hat{H}(e^{j\omega})$?

- ▶ $s[n]$: length- M signal
- ▶ $\hat{h}[n]$: finite-support infinite sequence from $s[n]$
- ▶ DFT coefficients $S[k]$ are known
- ▶ use the DFT to DTFT result for finite-support sequences: Lagrangian interpolation

Frequency sampling: what happens in the frequency domain

what is the frequency response $\hat{H}(e^{j\omega})$?

- ▶ $s[n]$: length- M signal
- ▶ $\hat{h}[n]$: finite-support infinite sequence from $s[n]$
- ▶ DFT coefficients $S[k]$ are known
- ▶ use the DFT to DTFT result for finite-support sequences: Lagrangian interpolation

Frequency sampling: what happens in the frequency domain

what is the frequency response $\hat{H}(e^{j\omega})$?

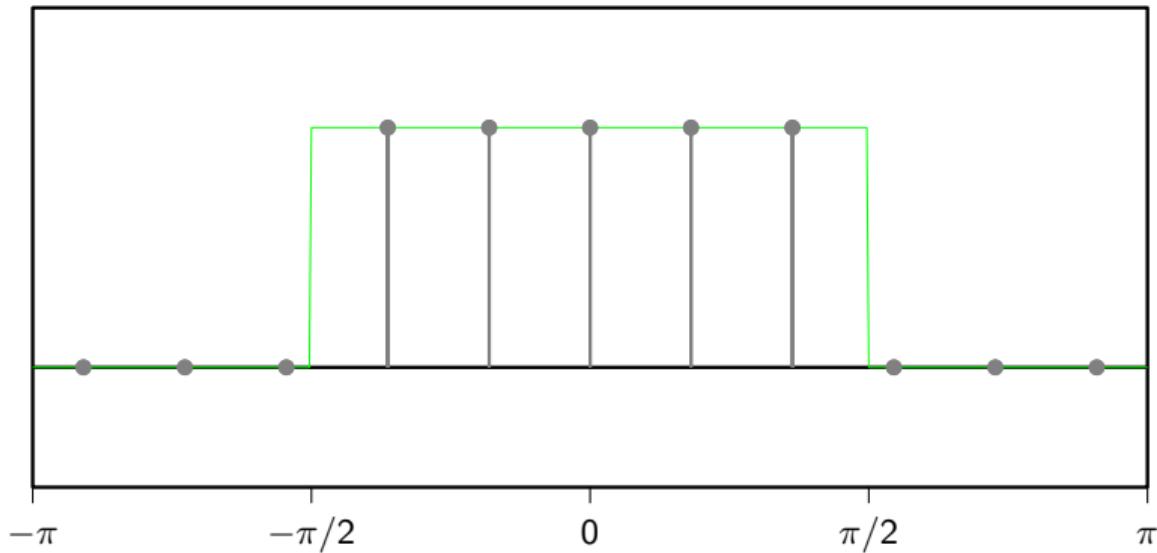
- ▶ $s[n]$: length- M signal
- ▶ $\hat{h}[n]$: finite-support infinite sequence from $s[n]$
- ▶ DFT coefficients $S[k]$ are known
- ▶ use the DFT to DTFT result for finite-support sequences: Lagrangian interpolation

DTFT of finite-support signals

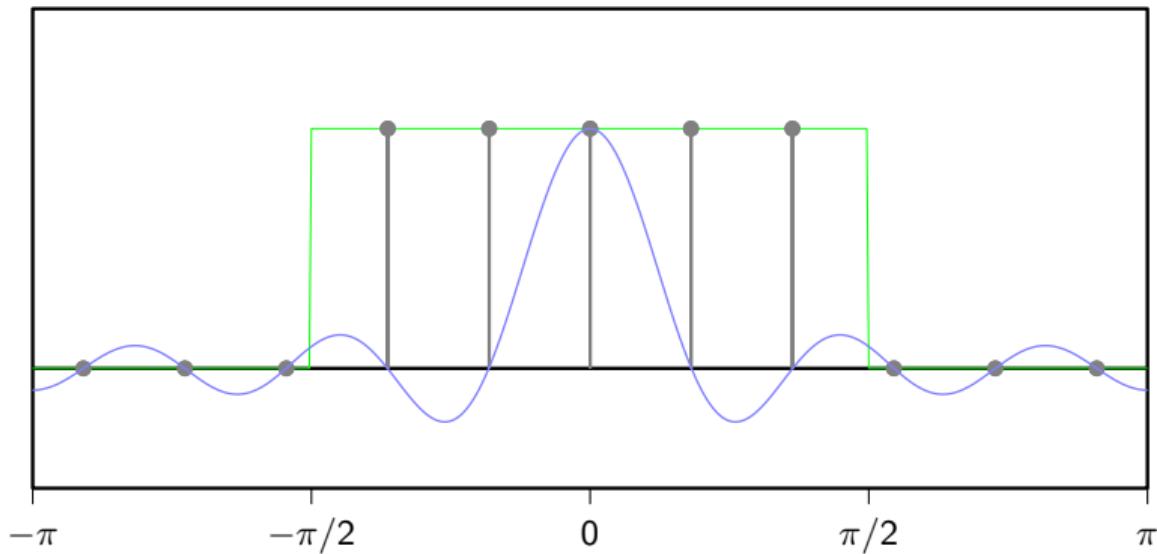
$$\hat{H}(e^{j\omega}) = \sum_{k=0}^{M-1} S[k] \Lambda(\omega - \frac{2\pi}{M} k)$$

with $\Lambda(\omega) = \frac{1}{M} \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} e^{-j\frac{\omega}{2}(M-1)}$: smooth interpolation of DFT values.

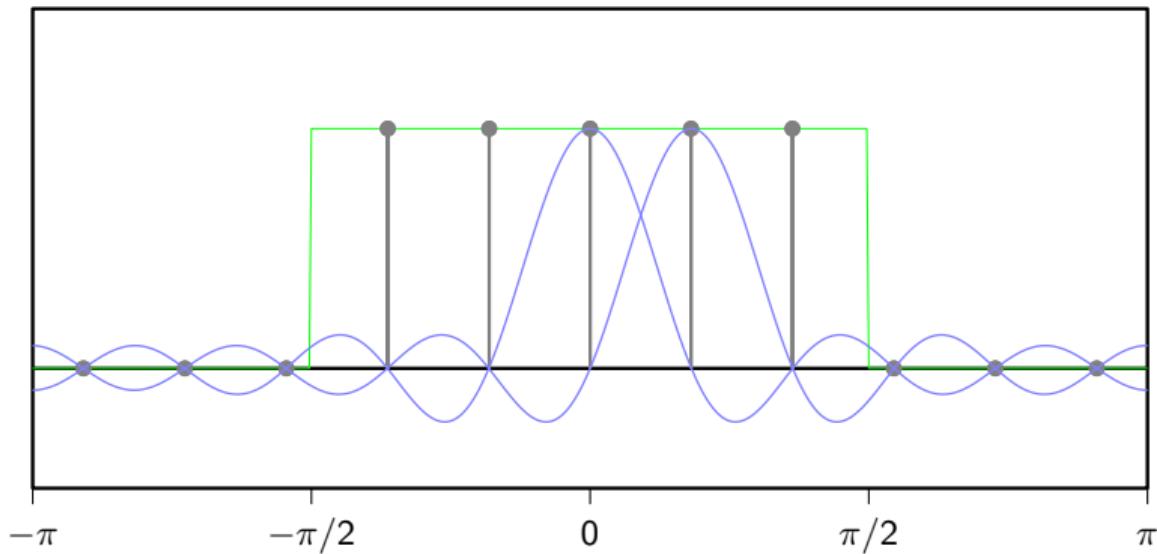
Frequency sampling: frequency response



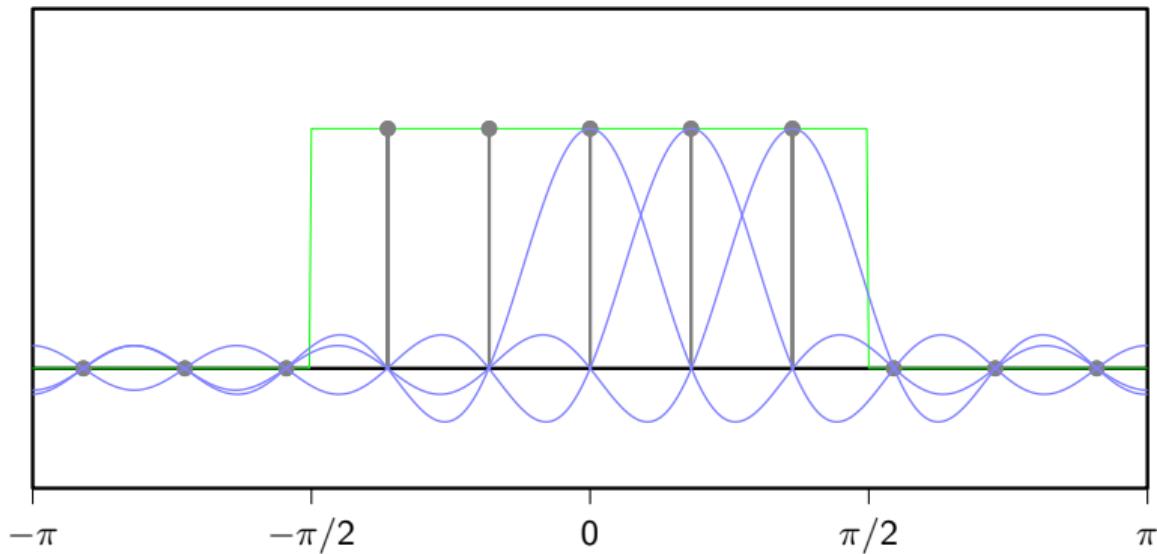
Frequency sampling: frequency response



Frequency sampling: frequency response

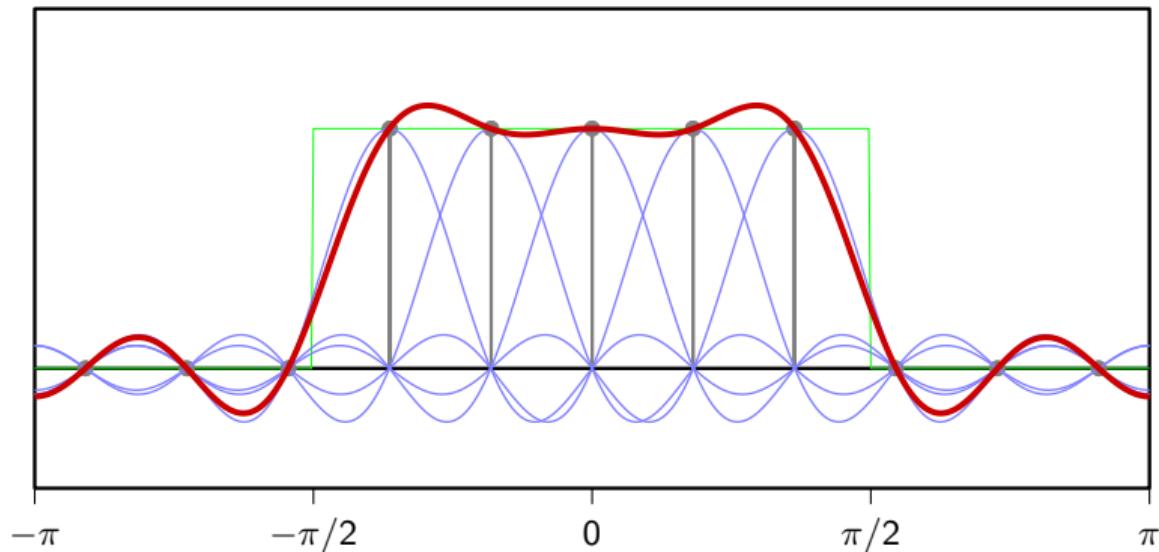


Frequency sampling: frequency response



Frequency sampling: frequency response

$$\hat{H}(e^{j\omega})$$



Frequency sampling: pros and cons

Pros:

- ▶ simple
- ▶ works with arbitrary frequency responses

Cons:

- ▶ can't control max error

COM303: Digital Signal Processing

Lecture 11: z -transform, filter structures and design examples

Overview

- ▶ the z -transform
- ▶ filtering algorithms and filter structures
- ▶ intuitive filter design
- ▶ two more ideal filters

Overview

- ▶ the z -transform
- ▶ filtering algorithms and filter structures
- ▶ intuitive filter design
- ▶ two more ideal filters

Overview

- ▶ the z -transform
- ▶ filtering algorithms and filter structures
- ▶ intuitive filter design
- ▶ two more ideal filters

Overview

- ▶ the z -transform
- ▶ filtering algorithms and filter structures
- ▶ intuitive filter design
- ▶ two more ideal filters

the *z*-transform

Overview:

- ▶ Constant-Coefficient Difference Equations
- ▶ The z -transform
- ▶ System transfer function
- ▶ Region of convergence and system stability

Overview:

- ▶ Constant-Coefficient Difference Equations
- ▶ The z -transform
- ▶ System transfer function
- ▶ Region of convergence and system stability

Overview:

- ▶ Constant-Coefficient Difference Equations
- ▶ The z -transform
- ▶ System transfer function
- ▶ Region of convergence and system stability

Overview:

- ▶ Constant-Coefficient Difference Equations
- ▶ The z -transform
- ▶ System transfer function
- ▶ Region of convergence and system stability

The most general LTI systems

- ▶ ideal filters cannot be implemented
- ▶ what is the most general, realizable LTI transformation?
 - linearity: only sums and multiplications
 - time-invariance: only multiplications by constants
 - realizability: only finite number of past and future samples

The most general LTI systems

- ▶ ideal filters cannot be implemented
- ▶ what is the most general, realizable LTI transformation?
 - linearity: only sums and multiplications
 - time-invariance: only multiplications by constants
 - realizability: only finite number of past and future samples

The most general LTI systems

- ▶ ideal filters cannot be implemented
- ▶ what is the most general, realizable LTI transformation?
 - linearity: only sums and multiplications
 - time-invariance: only multiplications by constants
 - realizability: only finite number of past and future samples

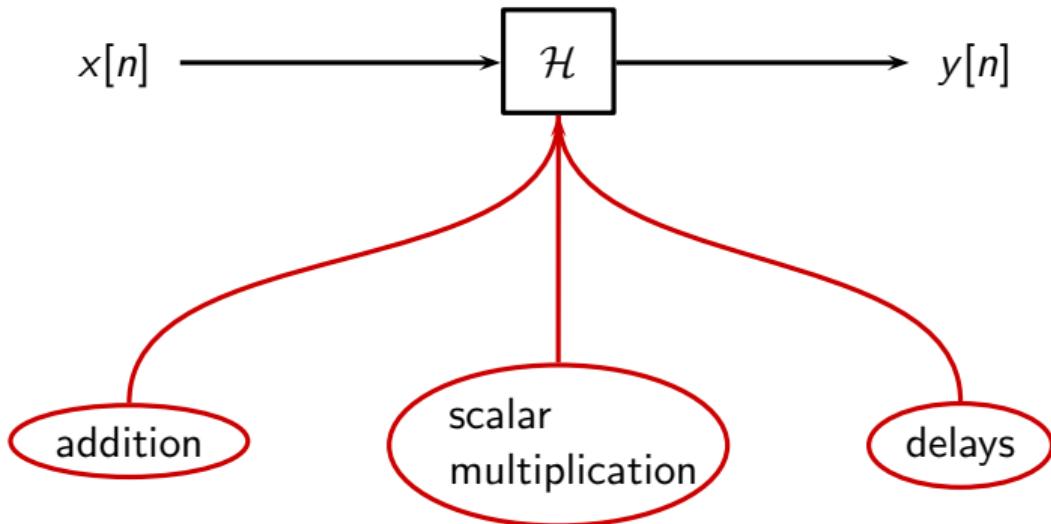
The most general LTI systems

- ▶ ideal filters cannot be implemented
- ▶ what is the most general, realizable LTI transformation?
 - linearity: only sums and multiplications
 - time-invariance: only multiplications by constants
 - realizability: only finite number of past and future samples

The most general LTI systems

- ▶ ideal filters cannot be implemented
- ▶ what is the most general, realizable LTI transformation?
 - linearity: only sums and multiplications
 - time-invariance: only multiplications by constants
 - realizability: only finite number of past and future samples

Linear, time-invariant systems



Constant-Coefficient Difference Equation

$$\sum_{k=0}^N a_k y[n-k] = \sum_{k=0}^M b_k x[n-k]$$

- ▶ uses $M + 1$ input and N output values
- ▶ $a_0 = 1$ (otherwise renormalize)
- ▶ completely specified by $M + N + 1$ scalar coefficients

Constant-Coefficient Difference Equation

$$\sum_{k=0}^N a_k y[n-k] = \sum_{k=0}^M b_k x[n-k]$$

- ▶ uses $M + 1$ input and N output values
- ▶ $a_0 = 1$ (otherwise renormalize)
- ▶ completely specified by $M + N + 1$ scalar coefficients

Constant-Coefficient Difference Equation

$$\sum_{k=0}^N a_k y[n-k] = \sum_{k=0}^M b_k x[n-k]$$

- ▶ uses $M + 1$ input and N output values
- ▶ $a_0 = 1$ (otherwise renormalize)
- ▶ completely specified by $M + N + 1$ scalar coefficients

Constant-Coefficient Difference Equation

$$\sum_{k=0}^N a_k y[n-k] = \sum_{k=0}^M b_k x[n-k]$$

- ▶ uses $M + 1$ input and N output values
- ▶ $a_0 = 1$ (otherwise renormalize)
- ▶ completely specified by $M + N + 1$ scalar coefficients

Constant-Coefficient Difference Equation

Examples:

- ▶ moving average:

$$y[n] = (1/4)x[n] + (1/4)x[n - 1] + (1/4)x[n - 2] + (1/4)x[n - 3]$$

- ▶ leaky integrator:

$$y[n] = \lambda y[n - 1] + (1 - \lambda)x[n]$$

Constant-Coefficient Difference Equation

Examples:

- ▶ moving average:

$$y[n] = (1/4)x[n] + (1/4)x[n - 1] + (1/4)x[n - 2] + (1/4)x[n - 3]$$

- ▶ leaky integrator:

$$y[n] = \lambda y[n - 1] + (1 - \lambda)x[n]$$

Constant-Coefficient Difference Equation

$$y[n] = \sum_{k=0}^M b_k x[n - k] - \sum_{k=1}^N a_k y[n - k]$$

- ▶ causal formulation
- ▶ provides an *algorithm* to compute each output value
- ▶ how do we compute the frequency response? Well, it's the DTFT of...
- ▶ how do we compute the impulse response?

Constant-Coefficient Difference Equation

$$y[n] = \sum_{k=0}^M b_k x[n - k] - \sum_{k=1}^N a_k y[n - k]$$

- ▶ causal formulation
- ▶ provides an *algorithm* to compute each output value
- ▶ how do we compute the frequency response? Well, it's the DTFT of...
- ▶ how do we compute the impulse response?

Constant-Coefficient Difference Equation

$$y[n] = \sum_{k=0}^M b_k x[n - k] - \sum_{k=1}^N a_k y[n - k]$$

- ▶ causal formulation
- ▶ provides an *algorithm* to compute each output value
- ▶ how do we compute the frequency response? Well, it's the DTFT of...
- ▶ how do we compute the impulse response?

Constant-Coefficient Difference Equation

$$y[n] = \sum_{k=0}^M b_k x[n - k] - \sum_{k=1}^N a_k y[n - k]$$

- ▶ causal formulation
- ▶ provides an *algorithm* to compute each output value
- ▶ how do we compute the frequency response? Well, it's the DTFT of...
- ▶ how do we compute the impulse response?

Constant-Coefficient Difference Equation

$$y[n] = \sum_{k=0}^M b_k x[n-k] - \sum_{k=1}^N a_k y[n-k]$$

- ▶ causal formulation
- ▶ provides an *algorithm* to compute each output value
- ▶ how do we compute the frequency response? Well, it's the DTFT of...
- ▶ how do we compute the impulse response?

Apparently unrelated topic: Polynomial Multiplication

$$\begin{aligned}(1 + 3t + 2t^2)(2 + t - t^2 + 4t^3) &= 2 + \quad t - \quad t^2 + 4t^3 \\ &\quad + 6t + 3t^2 - 3t^3 + 12t^4 \\ &\quad + 4t^2 + 2t^3 - 2t^4 \quad + 8t^5 \\ &= 2 + 7t + 6t^2 + 3t^3 + 10t^4 + 8t^5\end{aligned}$$

Apparently unrelated topic: Polynomial Multiplication

$$\begin{aligned}(1 + 3t + 2t^2)(2 + t - t^2 + 4t^3) &= 2 + t - t^2 + 4t^3 \\&\quad + 6t + 3t^2 - 3t^3 + 12t^4 \\&\quad + 4t^2 + 2t^3 - 2t^4 + 8t^5 \\&= 2 + 7t + 6t^2 + 3t^3 + 10t^4 + 8t^5\end{aligned}$$

Apparently unrelated topic: Polynomial Multiplication

$$\begin{aligned}(1 + 3t + 2t^2)(2 + t - t^2 + 4t^3) &= 2 + \quad t - \quad t^2 + 4t^3 \\ &\quad + 6t + 3t^2 - 3t^3 + 12t^4 \\ &\quad + 4t^2 + 2t^3 - 2t^4 \quad + 8t^5 \\ &= 2 + 7t + 6t^2 + 3t^3 + 10t^4 + 8t^5\end{aligned}$$

Now try this...

$$[\dots \ 1 \ 3 \ 2 \ 0 \ 0 \ \dots] * [\dots \ 2 \ 1 \ -1 \ 4 \ 0 \ \dots] =$$
$$[\dots \ 2 \ 7 \ 6 \ 3 \ 10 \ 8 \ 0 \ \dots]$$

Polynomial multiplication and convolution are the same

$$p(t) = p_0 + p_1 t + \dots + p_P t^P$$

$$q(t) = q_0 + q_1 t + \dots + q_Q t^Q$$

$$r(t) = p(t) \cdot q(t) = \sum_{n=0}^{P+Q} r_n t^n$$

$$r_n = \sum_{k=\max\{0, n-Q\}}^{\min\{n, P\}} p_k q_{n-k}, \quad 0 \leq n \leq P+Q$$

Polynomial multiplication and convolution are the same

$$p(t) = p_0 + p_1 t + \dots + p_P t^P$$

$$q(t) = q_0 + q_1 t + \dots + q_Q t^Q$$

$$r(t) = p(t) \cdot q(t) = \sum_{n=0}^{P+Q} r_n t^n$$

$$r_n = \sum_{k=\max\{0, n-Q\}}^{\min\{n, P\}} p_k q_{n-k}, \quad 0 \leq n \leq P+Q$$

Polynomial multiplication and convolution are the same

$$p(t) = p_0 + p_1 t + \dots + p_P t^P$$

$$q(t) = q_0 + q_1 t + \dots + q_Q t^Q$$

$$r(t) = p(t) \cdot q(t) = \sum_{n=0}^{P+Q} r_n t^n$$

$$r_n = \sum_{k=\max\{0, n-Q\}}^{\min\{n, P\}} p_k q_{n-k}, \quad 0 \leq n \leq P + Q$$

Polynomial multiplication and convolution are the same

Better still: assume $p_n = 0$ for $n \notin [0, P]$ and $q_n = 0$ for $n \notin [0, Q]$:

$$r_n = \sum_{k=-\infty}^{\infty} p_k q_{n-k},$$

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

Polynomial multiplication and convolution are the same

Better still: assume $p_n = 0$ for $n \notin [0, P]$ and $q_n = 0$ for $n \notin [0, Q]$:

$$r_n = \sum_{k=-\infty}^{\infty} p_k q_{n-k},$$

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

The z-transform

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n}, \quad z \in \mathbb{C}$$

- ▶ associate a power series (i.e. a polynomial) to a sequence
- ▶ for us mostly a *formal operator*...
- ▶ ...but also as the extension of the DTFT to the whole complex plane:

$$X(z)|_{z=e^{j\omega}} = \text{DTFT}\{x[n]\}$$

- ▶ (and now the notation $X(e^{j\omega})$ should make more sense)

The z -transform

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n}, \quad z \in \mathbb{C}$$

- ▶ associate a power series (i.e. a polynomial) to a sequence
- ▶ for us mostly a *formal operator*...
- ▶ ...but also as the extension of the DTFT to the whole complex plane:

$$X(z)|_{z=e^{j\omega}} = \text{DTFT}\{x[n]\}$$

- ▶ (and now the notation $X(e^{j\omega})$ should make more sense)

The z -transform

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n}, \quad z \in \mathbb{C}$$

- ▶ associate a power series (i.e. a polynomial) to a sequence
- ▶ for us mostly a *formal operator*...
- ▶ ...but also as the extension of the DTFT to the whole complex plane:

$$X(z)|_{z=e^{j\omega}} = \text{DTFT}\{x[n]\}$$

- ▶ (and now the notation $X(e^{j\omega})$ should make more sense)

The z -transform

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n}, \quad z \in \mathbb{C}$$

- ▶ associate a power series (i.e. a polynomial) to a sequence
- ▶ for us mostly a *formal operator*...
- ▶ ...but also as the extension of the DTFT to the whole complex plane:

$$X(z)|_{z=e^{j\omega}} = \text{DTFT}\{x[n]\}$$

- ▶ (and now the notation $X(e^{j\omega})$ should make more sense)

The z -transform

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n}, \quad z \in \mathbb{C}$$

- ▶ associate a power series (i.e. a polynomial) to a sequence
- ▶ for us mostly a *formal operator*...
- ▶ ...but also as the extension of the DTFT to the whole complex plane:

$$X(z)|_{z=e^{j\omega}} = \text{DTFT}\{x[n]\}$$

- ▶ (and now the notation $X(e^{j\omega})$ should make more sense)

Convergence

the z -transform is a power *series*

we should (and will) be concerned about its convergence

but not for now...

Convergence

the z -transform is a power *series*

we should (and will) be concerned about its convergence

but not for now...

Convergence

the z -transform is a power *series*

we should (and will) be concerned about its convergence

but not for now...

Key properties

linearity:

$$\mathcal{Z}\{\alpha x[n] + \beta y[n]\} = \alpha X(z) + \beta Y(z)$$

time shift:

$$\mathcal{Z}\{x[n - N]\} = z^{-N}X(z)$$

convolution:

$$\mathcal{Z}\{h[n] * x[n]\} = H(z)X(z)$$

Key properties

linearity:

$$\mathcal{Z}\{\alpha x[n] + \beta y[n]\} = \alpha X(z) + \beta Y(z)$$

time shift:

$$\mathcal{Z}\{x[n - N]\} = z^{-N}X(z)$$

convolution:

$$\mathcal{Z}\{h[n] * x[n]\} = H(z)X(z)$$

Key properties

linearity:

$$\mathcal{Z}\{\alpha x[n] + \beta y[n]\} = \alpha X(z) + \beta Y(z)$$

time shift:

$$\mathcal{Z}\{x[n - N]\} = z^{-N}X(z)$$

convolution:

$$\mathcal{Z}\{h[n] * x[n]\} = H(z)X(z)$$

Convolution in the z -domain

Consider an LTI system with impulse response $h[n]$

$$y[n] = h[n] * x[n]$$

$$\mathcal{Z}\{y[n]\} = \mathcal{Z}\{h[n] * x[n]\}$$

$$Y(z) = H(z)X(z)$$

$H(z)$ is the *transfer function* of the system

Convolution in the z -domain

Consider an LTI system with impulse response $h[n]$

$$y[n] = h[n] * x[n]$$

$$\mathcal{Z}\{y[n]\} = \mathcal{Z}\{h[n] * x[n]\}$$

$$Y(z) = H(z)X(z)$$

$H(z)$ is the *transfer function* of the system

Transfer function

- ▶ the transfer function is the z -transform of the impulse response
- ▶ by setting $z = e^{j\omega}$ in $H(z)$ we get the frequency response

Transfer function

- ▶ the transfer function is the z -transform of the impulse response
- ▶ by setting $z = e^{j\omega}$ in $H(z)$ we get the frequency response

Now let's go back to where we started...

$$y[n] = \sum_{k=0}^M b_k x[n-k] - \sum_{k=1}^N a_k y[n-k]$$

- ▶ causal formulation
- ▶ provides an *algorithm* to compute each output value
- ▶ the frequency response is the DTFT of the impulse response
- ▶ how do we compute the impulse response from the CCDE?

Applying the z -transform to CCDE's

$$\sum_{k=0}^N a_k y[n-k] = \sum_{k=0}^M b_k x[n-k]$$

$$Y(z) \sum_{k=0}^N a_k z^{-k} = X(z) \sum_{k=0}^M b_k z^{-k}$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^M b_k z^{-k}}{1 + \sum_{k=1}^N a_k z^{-k}}$$

Applying the z -transform to CCDE's

$$\sum_{k=0}^N a_k y[n-k] = \sum_{k=0}^M b_k x[n-k]$$

$$Y(z) \sum_{k=0}^N a_k z^{-k} = X(z) \sum_{k=0}^M b_k z^{-k}$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^M b_k z^{-k}}{1 + \sum_{k=1}^N a_k z^{-k}}$$

Applying the z -transform to CCDE's

$$\sum_{k=0}^N a_k y[n-k] = \sum_{k=0}^M b_k x[n-k]$$

$$Y(z) \sum_{k=0}^N a_k z^{-k} = X(z) \sum_{k=0}^M b_k z^{-k}$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^M b_k z^{-k}}{1 + \sum_{k=1}^N a_k z^{-k}}$$

Rational Transfer Function

- ▶ we can obtain the transfer function of an LTI directly from the CCDE coefficients!
- ▶ the transfer function is a ratio of polynomials
- ▶ this ASSUMES that everything converges...

Rational Transfer Function

- ▶ we can obtain the transfer function of an LTI directly from the CCDE coefficients!
- ▶ the transfer function is a ratio of polynomials
- ▶ this ASSUMES that everything converges...

Rational Transfer Function

- ▶ we can obtain the transfer function of an LTI directly from the CCDE coefficients!
- ▶ the transfer function is a ratio of polynomials
- ▶ this ASSUMES that everything converges...

Rational Transfer Function

$$H(z) = \frac{\sum_{k=0}^M b_k z^{-k}}{1 + \sum_{k=1}^N a_k z^{-k}}$$

- ▶ feedforward part
- ▶ feedback part

Rational Transfer Function

$$H(z) = \frac{\sum_{k=0}^M b_k z^{-k}}{1 + \sum_{k=1}^N a_k z^{-k}}$$

- ▶ feedforward part
- ▶ feedback part

Leaky Integrator revisited

► CCDE: $y[n] = (1 - \lambda)x[n] + \lambda y[n - 1]$

► impulse response: $h[n] = (1 - \lambda)\lambda^n u[n]$

► transfer function from impulse response

$$H(z) = (1 - \lambda) \sum_{n=0}^{\infty} \lambda^n z^{-n} = \frac{(1 - \lambda)}{1 - \lambda z^{-1}}$$

► transfer function from CCDE:

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

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

Leaky Integrator revisited

- ▶ CCDE: $y[n] = (1 - \lambda)x[n] + \lambda y[n - 1]$
- ▶ impulse response: $h[n] = (1 - \lambda)\lambda^n u[n]$
- ▶ transfer function from impulse response

$$H(z) = (1 - \lambda) \sum_{n=0}^{\infty} \lambda^n z^{-n} = \frac{(1 - \lambda)}{1 - \lambda z^{-1}}$$

- ▶ transfer function from CCDE:

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

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

Leaky Integrator revisited

- ▶ CCDE: $y[n] = (1 - \lambda)x[n] + \lambda y[n - 1]$
- ▶ impulse response: $h[n] = (1 - \lambda)\lambda^n u[n]$
- ▶ transfer function from impulse response

$$H(z) = (1 - \lambda) \sum_{n=0}^{\infty} \lambda^n z^{-n} = \frac{(1 - \lambda)}{1 - \lambda z^{-1}}$$

- ▶ transfer function from CCDE:

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

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

Leaky Integrator revisited

- ▶ CCDE: $y[n] = (1 - \lambda)x[n] + \lambda y[n - 1]$
- ▶ impulse response: $h[n] = (1 - \lambda)\lambda^n u[n]$
- ▶ transfer function from impulse response

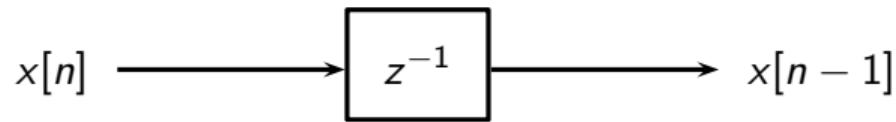
$$H(z) = (1 - \lambda) \sum_{n=0}^{\infty} \lambda^n z^{-n} = \frac{(1 - \lambda)}{1 - \lambda z^{-1}}$$

- ▶ transfer function from CCDE:

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

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

BTW, remember the delay block?



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

now the notation should make more sense!

Existence and region of convergence (ROC)

ROC is defined by the absolute convergence of the power series:

$$z \in \text{ROC}\{X(z)\} \iff \sum_{n=-\infty}^{\infty} |x[n]z^{-n}| < \infty$$

How can we determine the ROC?

- ▶ ROC depends on the values of $x[n]$
- ▶ for rational transfer function we can use indirect methods
- ▶ we don't care about convergence in zero and infinity

Existence and region of convergence (ROC)

ROC is defined by the absolute convergence of the power series:

$$z \in \text{ROC}\{X(z)\} \iff \sum_{n=-\infty}^{\infty} |x[n]z^{-n}| < \infty$$

How can we determine the ROC?

- ▶ ROC depends on the values of $x[n]$
- ▶ for rational transfer function we can use indirect methods
- ▶ we don't care about convergence in zero and infinity

Region of convergence (ROC)

observation #1:

for finite-support signals, the z -transform converges everywhere (except in 0 and/or ∞)

$$X(z) = \sum_{n=-M}^N x[n]z^{-n}$$

Region of convergence (ROC)

observation #2:

the region of convergence has circular symmetry: set $z = ae^{j\theta}$:

$$\sum_{n=-\infty}^{\infty} |x[n]z^{-n}| < \infty \iff \sum_{n=-\infty}^{\infty} |x[n]| |a^{-n}| < \infty$$

Region of convergence (ROC)

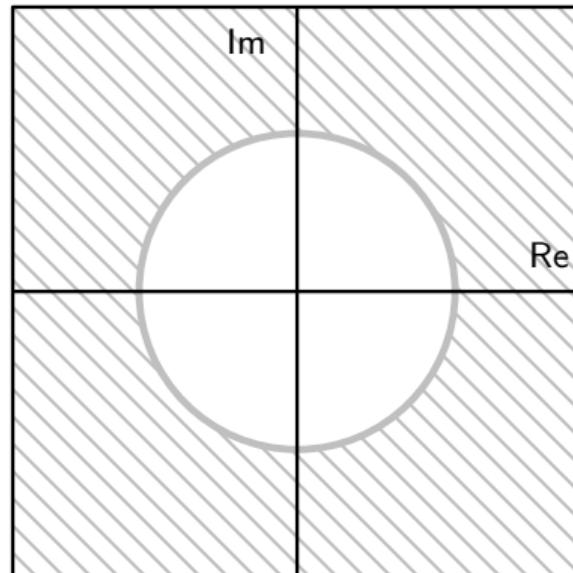
observation #3:

for causal sequences, the ROC extends from a circle to infinity:

assume $z_0 \in \text{ROC}$ and $|z_1| > |z_0|$:

$$\sum_{n=0}^{\infty} |x[n] z_1^{-n}| = \sum_{n=0}^{\infty} \frac{|x[n]|}{|z_1^n|} \leq \sum_{n=0}^{\infty} \frac{|x[n]|}{|z_0^n|} \leq \infty$$

ROC shape for causal sequences



Region of convergence (ROC)

so where are the convergence problems?

in general, difficult question; but we're only interested in rational transfer functions!

Region of convergence (ROC)

so where are the convergence problems?

in general, difficult question; but we're only interested in rational transfer functions!

ROC for causal systems

Consider the transfer function for an LTI system:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \dots + b_M z^{-M}}{1 + a_1 z^{-1} + \dots + a_N z^{-N}}$$

It can always be factored as:

$$H(z) = b_0 \frac{\prod_{n=1}^M (1 - z_n z^{-1})}{\prod_{n=1}^N (1 - p_n z^{-1})}$$

ROC for causal systems

- ▶ z_n 's: zeros of the transfer function
- ▶ p_n 's: poles of the transfer function
- ▶ only trouble spots for ROC are the poles

ROC for causal systems

- ▶ z_n 's: zeros of the transfer function
- ▶ p_n 's: poles of the transfer function
- ▶ only trouble spots for ROC are the poles

ROC for causal systems

- ▶ z_n 's: zeros of the transfer function
- ▶ p_n 's: poles of the transfer function
- ▶ only trouble spots for ROC are the poles

ROC for causal systems

We know:

- ▶ ROC extends outwards
- ▶ ROC cannot include poles

ROC extends outwards from a circle touching the largest-magnitude pole

ROC for causal systems

We know:

- ▶ ROC extends outwards
- ▶ ROC cannot include poles

ROC extends outwards from a circle touching the largest-magnitude pole

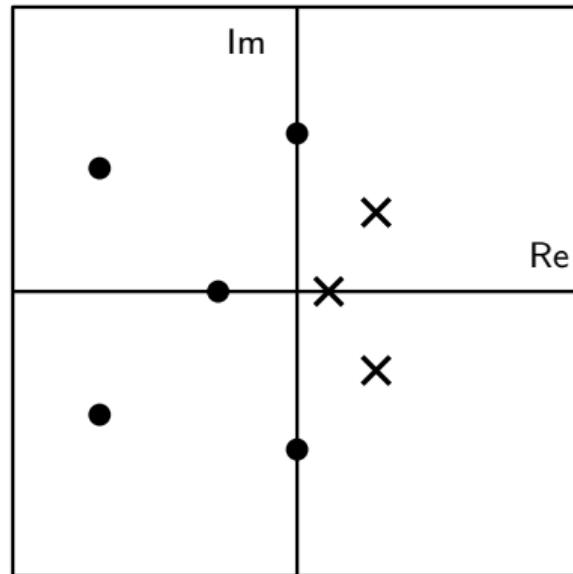
ROC for causal systems

We know:

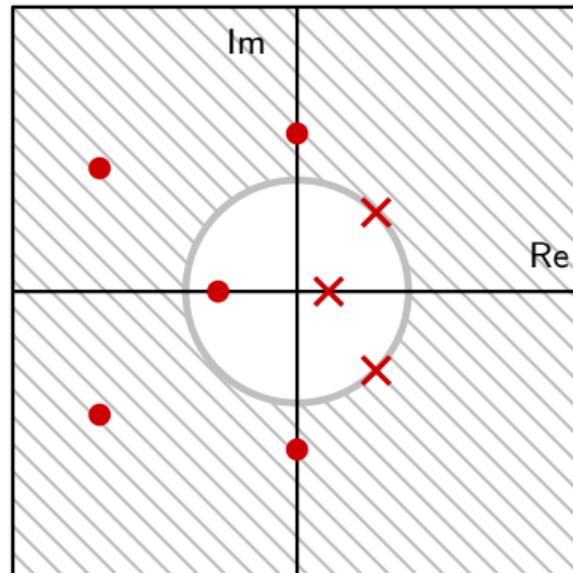
- ▶ ROC extends outwards
- ▶ ROC cannot include poles

ROC extends outwards from a circle touching the largest-magnitude pole

ROC for causal systems



ROC for causal systems



ROC for causal systems - Proof (sketch)

- ▶ $H(z) = B(z)(1/A(z))$
- ▶ with no loss of generality, assume $B(z) = 1$
- ▶ assume all poles distinct
- ▶ use partial fraction decomposition:

$$H(z) = \prod_{k=0}^{N-1} \frac{1}{(1 - p_k z^{-1})} = \sum_{k=0}^{N-1} \frac{c_k}{(1 - p_k z^{-1})}$$

ROC for causal systems - Proof (sketch)

- ▶ $H(z) = B(z)(1/A(z))$
- ▶ with no loss of generality, assume $B(z) = 1$
- ▶ assume all poles distinct
- ▶ use partial fraction decomposition:

$$H(z) = \prod_{k=0}^{N-1} \frac{1}{(1 - p_k z^{-1})} = \sum_{k=0}^{N-1} \frac{c_k}{(1 - p_k z^{-1})}$$

ROC for causal systems - Proof (sketch)

- ▶ $H(z) = B(z)(1/A(z))$
- ▶ with no loss of generality, assume $B(z) = 1$
- ▶ assume all poles distinct
- ▶ use partial fraction decomposition:

$$H(z) = \prod_{k=0}^{N-1} \frac{1}{(1 - p_k z^{-1})} = \sum_{k=0}^{N-1} \frac{c_k}{(1 - p_k z^{-1})}$$

ROC for causal systems - Proof (sketch)

- ▶ $H(z) = B(z)(1/A(z))$
- ▶ with no loss of generality, assume $B(z) = 1$
- ▶ assume all poles distinct
- ▶ use partial fraction decomposition:

$$H(z) = \prod_{k=0}^{N-1} \frac{1}{(1 - p_k z^{-1})} = \sum_{k=0}^{N-1} \frac{c_k}{(1 - p_k z^{-1})}$$

ROC for causal systems - Proof (sketch)

Example:

$$\begin{aligned}\frac{1}{1 - 5z^{-1} + 6z^{-2}} &= \frac{1}{(1 - 2z^{-1})(1 - 3z^{-1})} \\ &= \frac{c_0}{1 - 2z^{-1}} + \frac{c_1}{1 - 3z^{-1}}\end{aligned}$$

$$c_0 + c_1 = 1$$

$$2c_0 + 3c_1 = 0$$

$$\frac{1}{1 - 5z^{-1} + 6z^{-2}} = \frac{-2}{1 - 2z^{-1}} + \frac{3}{1 - 3z^{-1}}$$

ROC for causal systems - Proof (sketch)

Example:

$$\begin{aligned}\frac{1}{1 - 5z^{-1} + 6z^{-2}} &= \frac{1}{(1 - 2z^{-1})(1 - 3z^{-1})} \\ &= \frac{c_0}{1 - 2z^{-1}} + \frac{c_1}{1 - 3z^{-1}}\end{aligned}$$

$$c_0 + c_1 = 1$$

$$2c_0 + 3c_1 = 0$$

$$\frac{1}{1 - 5z^{-1} + 6z^{-2}} = \frac{-2}{1 - 2z^{-1}} + \frac{3}{1 - 3z^{-1}}$$

ROC for causal systems - Proof (sketch)

Example:

$$\begin{aligned}\frac{1}{1 - 5z^{-1} + 6z^{-2}} &= \frac{1}{(1 - 2z^{-1})(1 - 3z^{-1})} \\ &= \frac{c_0}{1 - 2z^{-1}} + \frac{c_1}{1 - 3z^{-1}}\end{aligned}$$

$$c_0 + c_1 = 1$$

$$2c_0 + 3c_1 = 0$$

$$\frac{1}{1 - 5z^{-1} + 6z^{-2}} = \frac{-2}{1 - 2z^{-1}} + \frac{3}{1 - 3z^{-1}}$$

ROC for causal systems - Proof (sketch)

Example:

$$\begin{aligned}\frac{1}{1 - 5z^{-1} + 6z^{-2}} &= \frac{1}{(1 - 2z^{-1})(1 - 3z^{-1})} \\ &= \frac{c_0}{1 - 2z^{-1}} + \frac{c_1}{1 - 3z^{-1}}\end{aligned}$$

$$c_0 + c_1 = 1$$

$$2c_0 + 3c_1 = 0$$

$$\frac{1}{1 - 5z^{-1} + 6z^{-2}} = \frac{-2}{1 - 2z^{-1}} + \frac{3}{1 - 3z^{-1}}$$

ROC for causal systems - Proof (sketch)

$$H(z) = \prod_{k=0}^{N-1} \frac{1}{(1 - p_k z^{-1})} = \sum_{k=0}^{N-1} \frac{c_k}{(1 - p_k z^{-1})}$$

- ▶ remember the leaky integrator...
- ▶ each term corresponds to an exponential sequence:

$$\mathcal{Z}\{p_k^n u[n]\} = \frac{1}{(1 - p_k z^{-1})}$$

- ▶ the ROC for each term is $|z| > |p_k|$
- ▶ intersection of all ROCs is $|z| > |p_{\max}|$

ROC for causal systems - Proof (sketch)

$$H(z) = \prod_{k=0}^{N-1} \frac{1}{(1 - p_k z^{-1})} = \sum_{k=0}^{N-1} \frac{c_k}{(1 - p_k z^{-1})}$$

- ▶ remember the leaky integrator...
- ▶ each term corresponds to an exponential sequence:

$$\mathcal{Z}\{p_k^n u[n]\} = \frac{1}{(1 - p_k z^{-1})}$$

- ▶ the ROC for each term is $|z| > |p_k|$
- ▶ intersection of all ROCs is $|z| > |p_{\max}|$

ROC for causal systems - Proof (sketch)

$$H(z) = \prod_{k=0}^{N-1} \frac{1}{(1 - p_k z^{-1})} = \sum_{k=0}^{N-1} \frac{c_k}{(1 - p_k z^{-1})}$$

- ▶ remember the leaky integrator...
- ▶ each term corresponds to an exponential sequence:

$$\mathcal{Z}\{p_k^n u[n]\} = \frac{1}{(1 - p_k z^{-1})}$$

- ▶ the ROC for each term is $|z| > |p_k|$
- ▶ intersection of all ROCs is $|z| > |p_{\max}|$

ROC for causal systems - Proof (sketch)

$$H(z) = \prod_{k=0}^{N-1} \frac{1}{(1 - p_k z^{-1})} = \sum_{k=0}^{N-1} \frac{c_k}{(1 - p_k z^{-1})}$$

- ▶ remember the leaky integrator...
- ▶ each term corresponds to an exponential sequence:

$$\mathcal{Z}\{p_k^n u[n]\} = \frac{1}{(1 - p_k z^{-1})}$$

- ▶ the ROC for each term is $|z| > |p_k|$
- ▶ intersection of all ROCs is $|z| > |p_{\max}|$

ROC for causal systems - Proof (sketch)

- ▶ same for multiple poles, just more tedious

$$\mathcal{Z}\{np^n u[n]\} = \frac{pz^{-1}}{(1-pz^{-1})^2}, \quad \text{ROC: } |z| > |p|$$

- ▶ all LTI impulse responses are linear combinations of weighted exponential sequences
- ▶ we could use an inverse z-transform to obtain $h[n]$

ROC for causal systems - Proof (sketch)

- ▶ same for multiple poles, just more tedious

$$\mathcal{Z}\{np^n u[n]\} = \frac{pz^{-1}}{(1-pz^{-1})^2}, \quad \text{ROC: } |z| > |p|$$

- ▶ all LTI impulse responses are linear combinations of weighed exponential sequences
- ▶ we could use an inverse z-transform to obtain $h[n]$

ROC for causal systems - Proof (sketch)

- ▶ same for multiple poles, just more tedious

$$\mathcal{Z}\{np^n u[n]\} = \frac{pz^{-1}}{(1-pz^{-1})^2}, \quad \text{ROC: } |z| > |p|$$

- ▶ all LTI impulse responses are linear combinations of weighed exponential sequences
- ▶ we could use an inverse z-transform to obtain $h[n]$

System stability criterion

Consider a filter with impulse response $h[n]$

- ▶ BIBO stability $\iff \sum_{n=-\infty}^{\infty} |h[n]| < \infty$
- ▶ $1 \in \text{ROC} \iff \sum_{n=-\infty}^{\infty} |h[n]| < \infty$ because $H(z)$ converges absolutely in $z = 1$

a causal system is stable if and only if ROC includes the unit circle!

System stability criterion

Consider a filter with impulse response $h[n]$

- ▶ BIBO stability $\iff \sum_{n=-\infty}^{\infty} |h[n]| < \infty$
- ▶ $1 \in \text{ROC} \iff \sum_{n=-\infty}^{\infty} |h[n]| < \infty$ because $H(z)$ converges absolutely in $z = 1$

a causal system is stable if and only if ROC includes the unit circle!

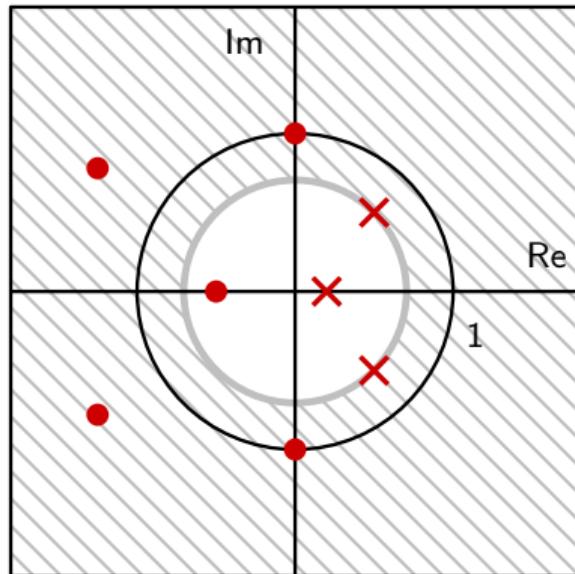
System stability criterion

Consider a filter with impulse response $h[n]$

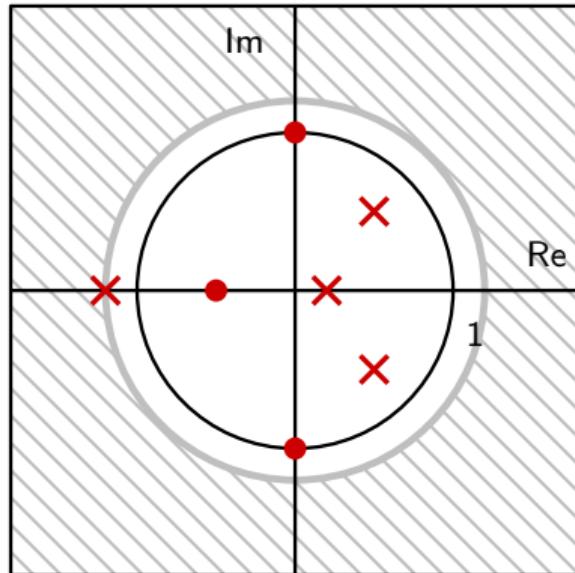
- ▶ BIBO stability $\iff \sum_{n=-\infty}^{\infty} |h[n]| < \infty$
- ▶ $1 \in \text{ROC} \iff \sum_{n=-\infty}^{\infty} |h[n]| < \infty$ because $H(z)$ converges absolutely in $z = 1$

a causal system is stable if and only if ROC includes the unit circle!

Stable system



Unstable system



Common confusion...

$$y[n] = 2y[n - 1] + x[n] \quad (\text{unstable!})$$

apply z -transform as a formal operator:

$$H(z) = Y(z)/X(z) = \frac{1}{1 - 2z^{-1}}$$

$H(1) = -1 < \infty$, so system is stable?

Common confusion...

$$y[n] = 2y[n - 1] + x[n] \quad (\text{unstable!})$$

apply z -transform as a formal operator:

$$H(z) = Y(z)/X(z) = \frac{1}{1 - 2z^{-1}}$$

$H(1) = -1 < \infty$, so system is stable?

Common confusion...

$$y[n] = 2y[n - 1] + x[n] \quad (\text{unstable!})$$

apply z -transform as a formal operator:

$$H(z) = Y(z)/X(z) = \frac{1}{1 - 2z^{-1}}$$

$$H(1) = -1 < \infty, \text{ so system is stable?}$$

Common confusion clarified

ROC depends on $h[n]$, NOT on *formal* value of $H(z)$:

- ▶ $h[n] = 2^n u[n]$
- ▶ to apply the z -transform operator we *assume* to be in the ROC
- ▶ the region of convergence is $|z| > 2$ because

$$\sum_{n=0}^{\infty} a^n z^{-n} = \lim_{N \rightarrow \infty} \frac{1 - (a/z)^N}{1 - (a/z)} = \begin{cases} \frac{1}{1 - az^{-1}} & \text{if } |z| > |a| \\ \infty & \text{otherwise} \end{cases}$$

In other words...

the function $\frac{1}{1 - az^{-1}}$ is defined for all $z \in \mathbb{C} \setminus \{a\}$

BUT

it is the z -transform of $a^n u[n]$ *only* for $|z| > |a|$

Rational Transfer Function

for which values of z does a rational $H(z)$ exist?

- ▶ option 1: compute $h[n]$ explicitly and find ROC for the power series $\sum h[n]z^{-n}$
- ▶ option 2: derive ROC indirectly:
 - ROC is circular symmetric
 - ROC extends outwards for causal sequences
 - ROC cannot include poles

Estimating the frequency response from the pole-zero plot

The “circus tent” method:

- ▶ magnitude of z transform is like a rubber sheet over the complex plane
- ▶ zeros glue the sheet to the ground
- ▶ poles are like ... poles, pushing it up
- ▶ frequency response (in magnitude) is sheet profile around the unit circle

Estimating the frequency response from the pole-zero plot

The “circus tent” method:

- ▶ magnitude of z transform is like a rubber sheet over the complex plane
- ▶ zeros glue the sheet to the ground
- ▶ poles are like ... poles, pushing it up
- ▶ frequency response (in magnitude) is sheet profile around the unit circle

Estimating the frequency response from the pole-zero plot

The “circus tent” method:

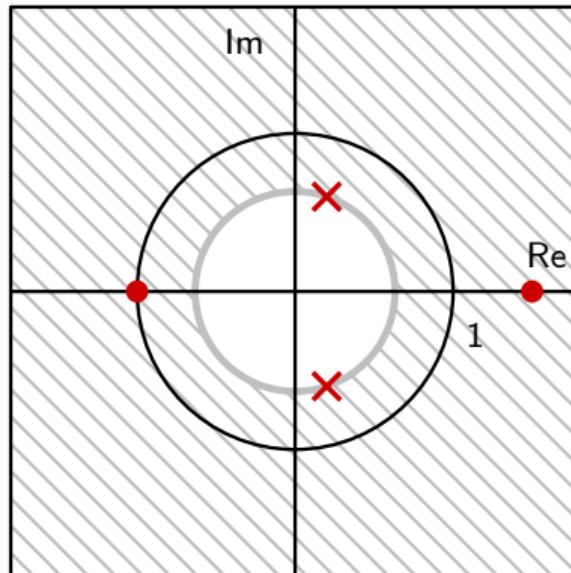
- ▶ magnitude of z transform is like a rubber sheet over the complex plane
- ▶ zeros glue the sheet to the ground
- ▶ poles are like ... poles, pushing it up
- ▶ frequency response (in magnitude) is sheet profile around the unit circle

Estimating the frequency response from the pole-zero plot

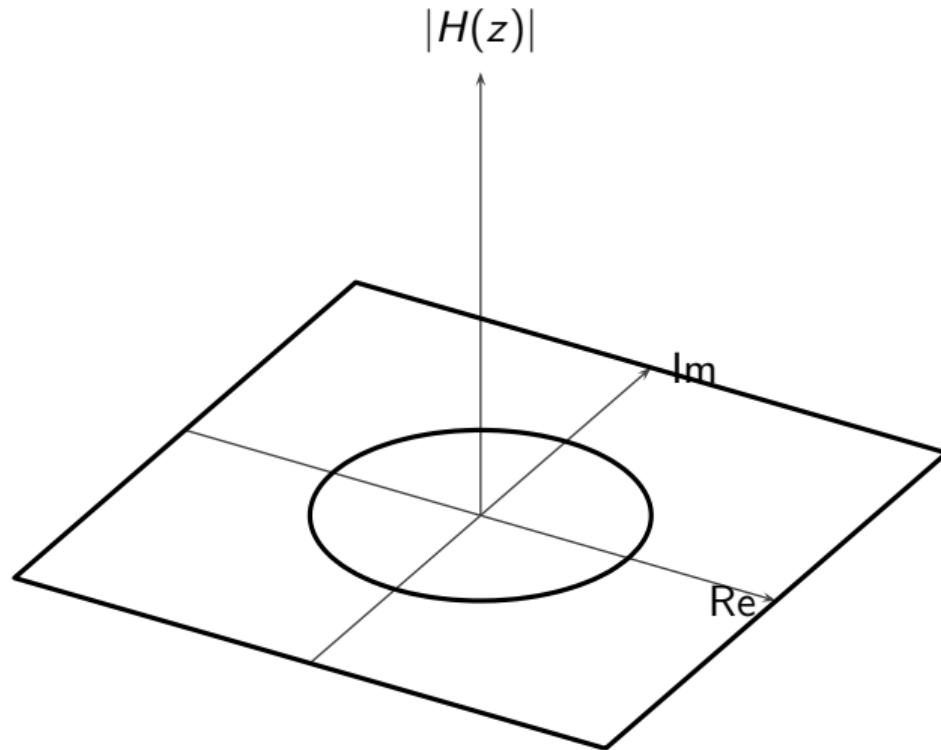
The “circus tent” method:

- ▶ magnitude of z transform is like a rubber sheet over the complex plane
- ▶ zeros glue the sheet to the ground
- ▶ poles are like ... poles, pushing it up
- ▶ frequency response (in magnitude) is sheet profile around the unit circle

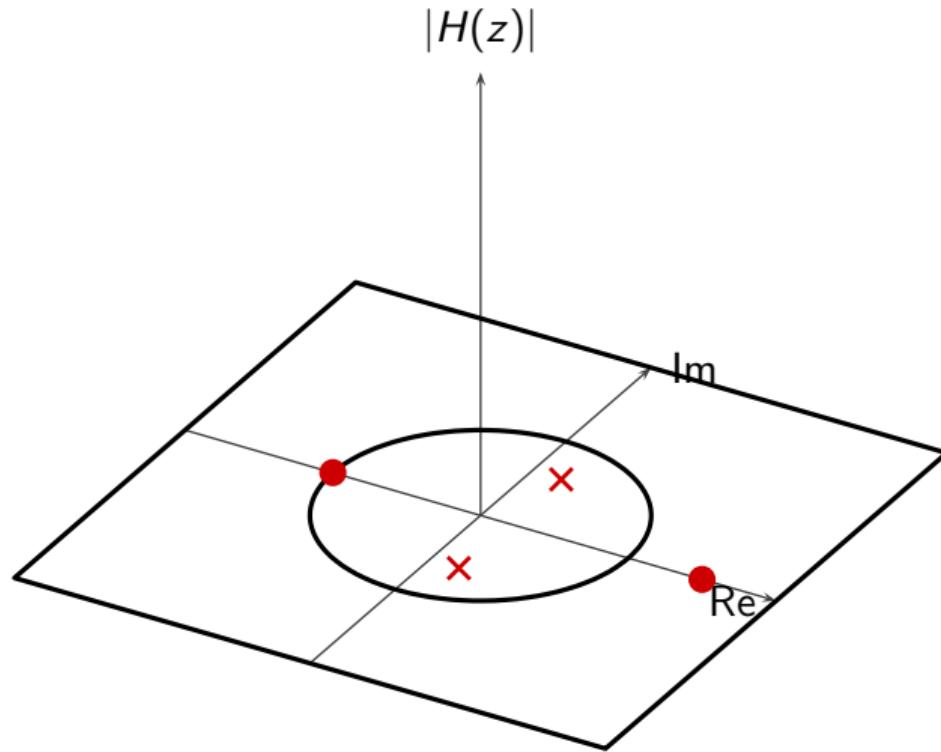
Estimating the frequency response



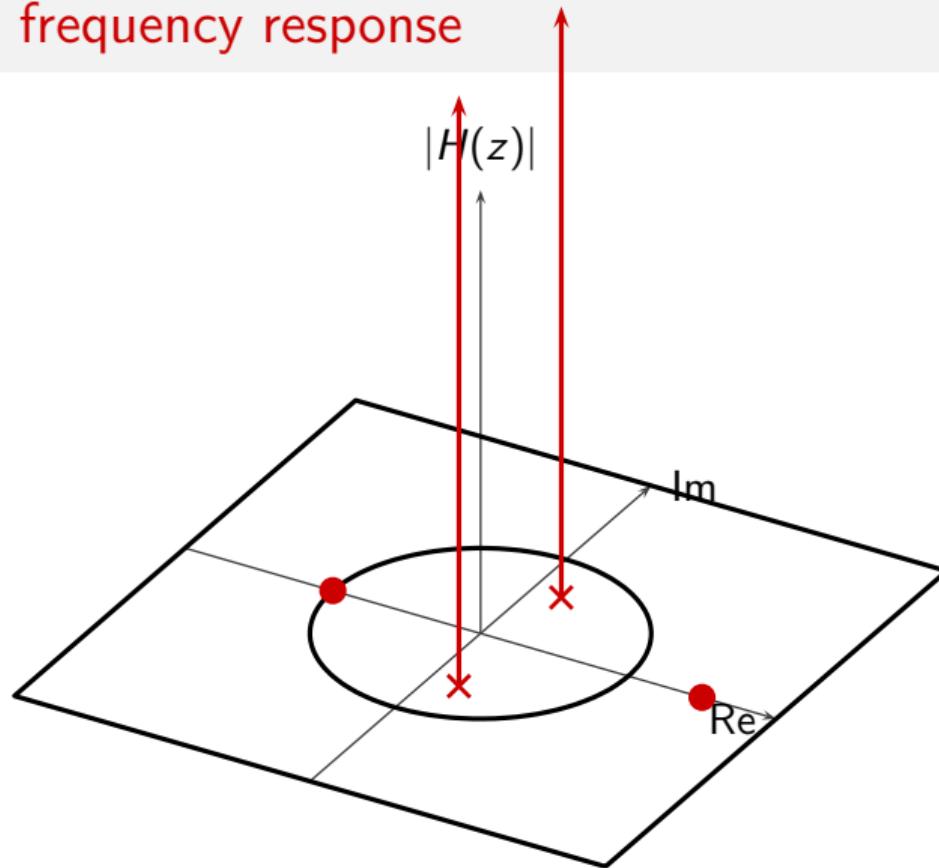
Estimating the frequency response



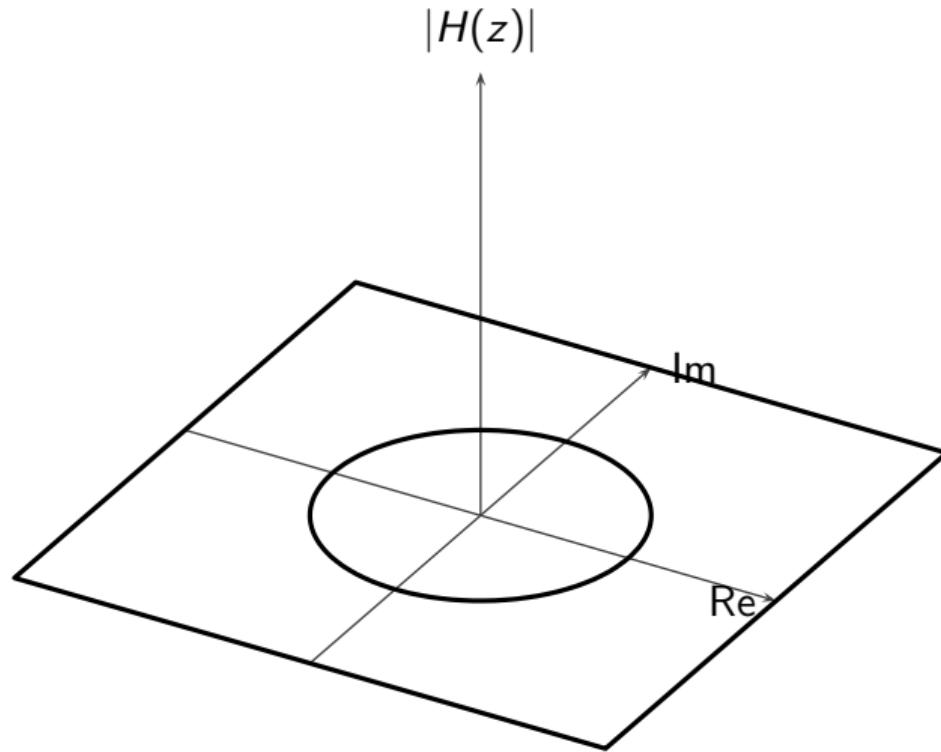
Estimating the frequency response



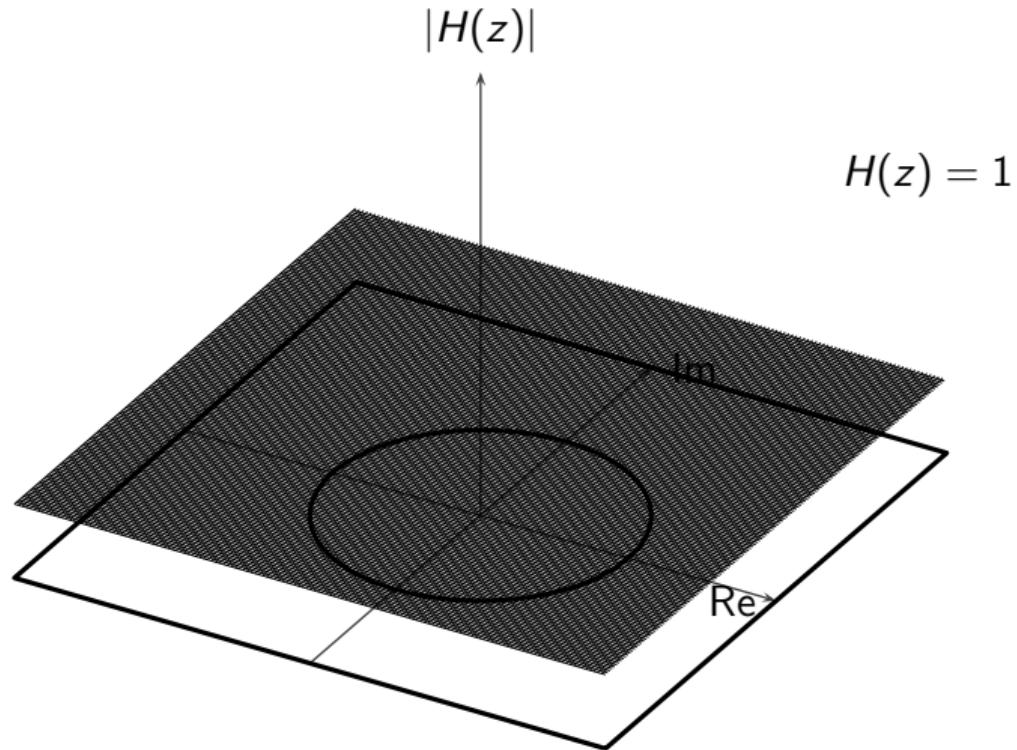
Estimating the frequency response



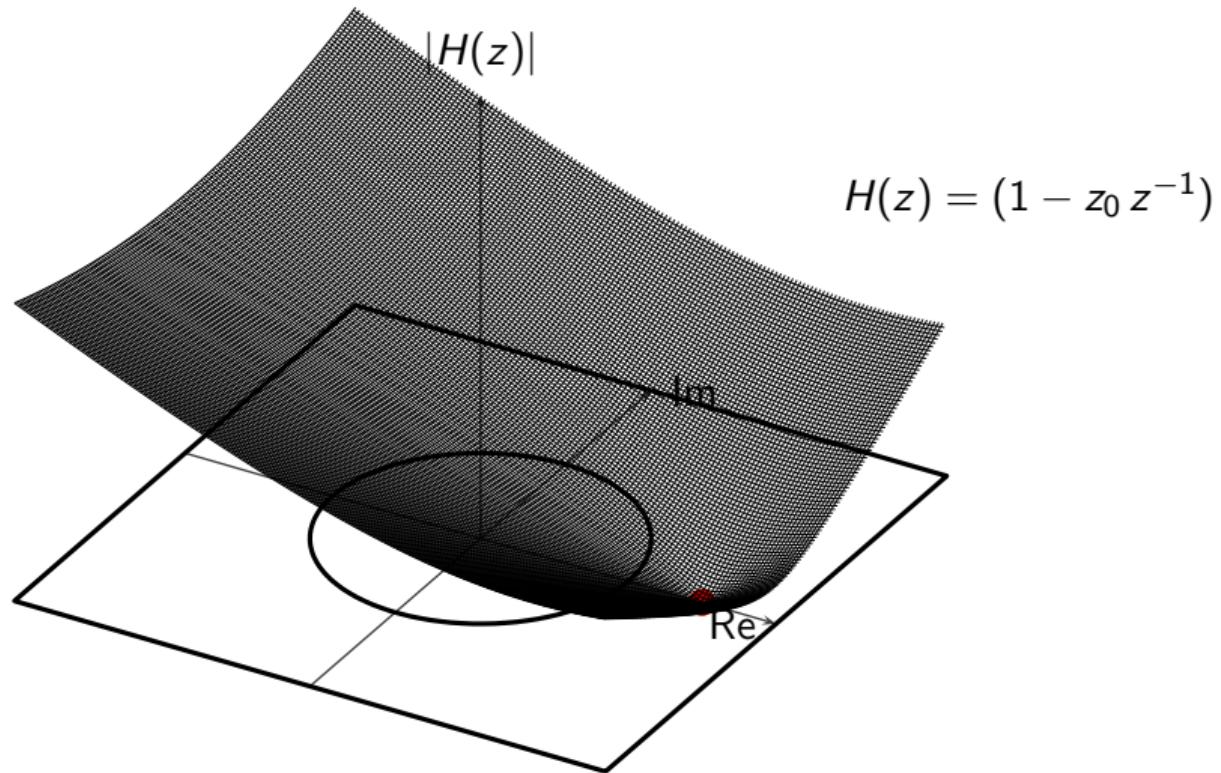
Estimating the frequency response



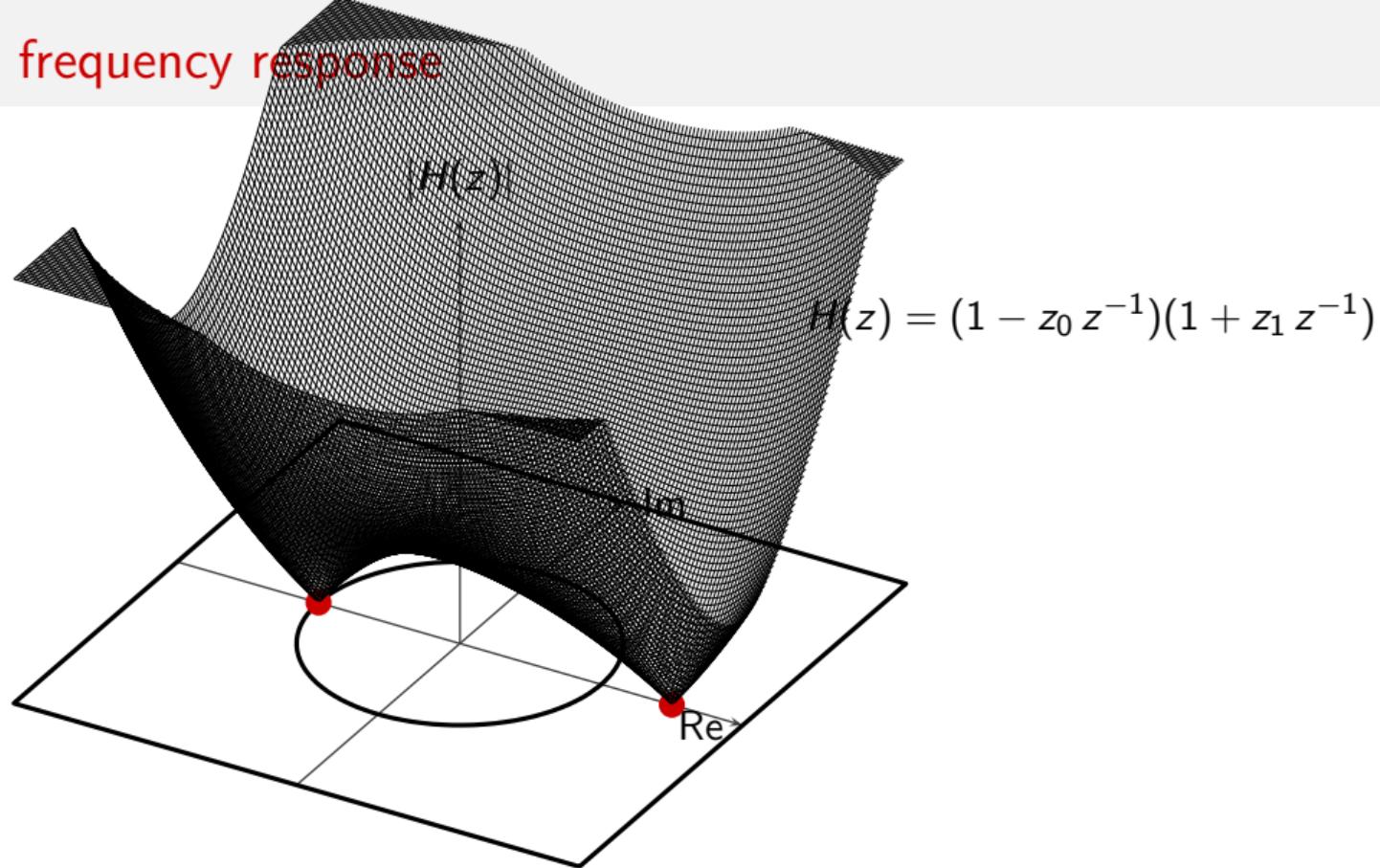
Estimating the frequency response



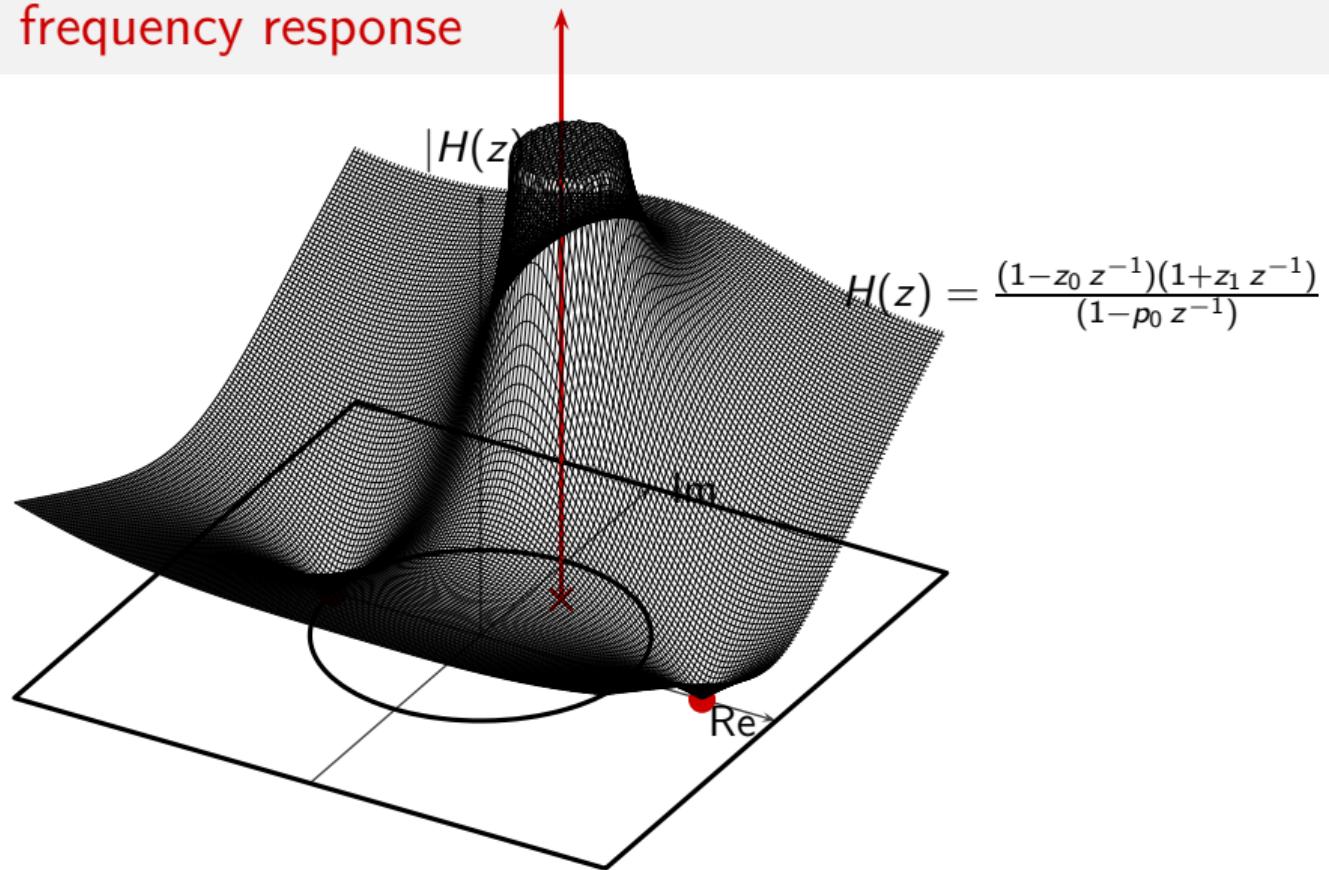
Estimating the frequency response



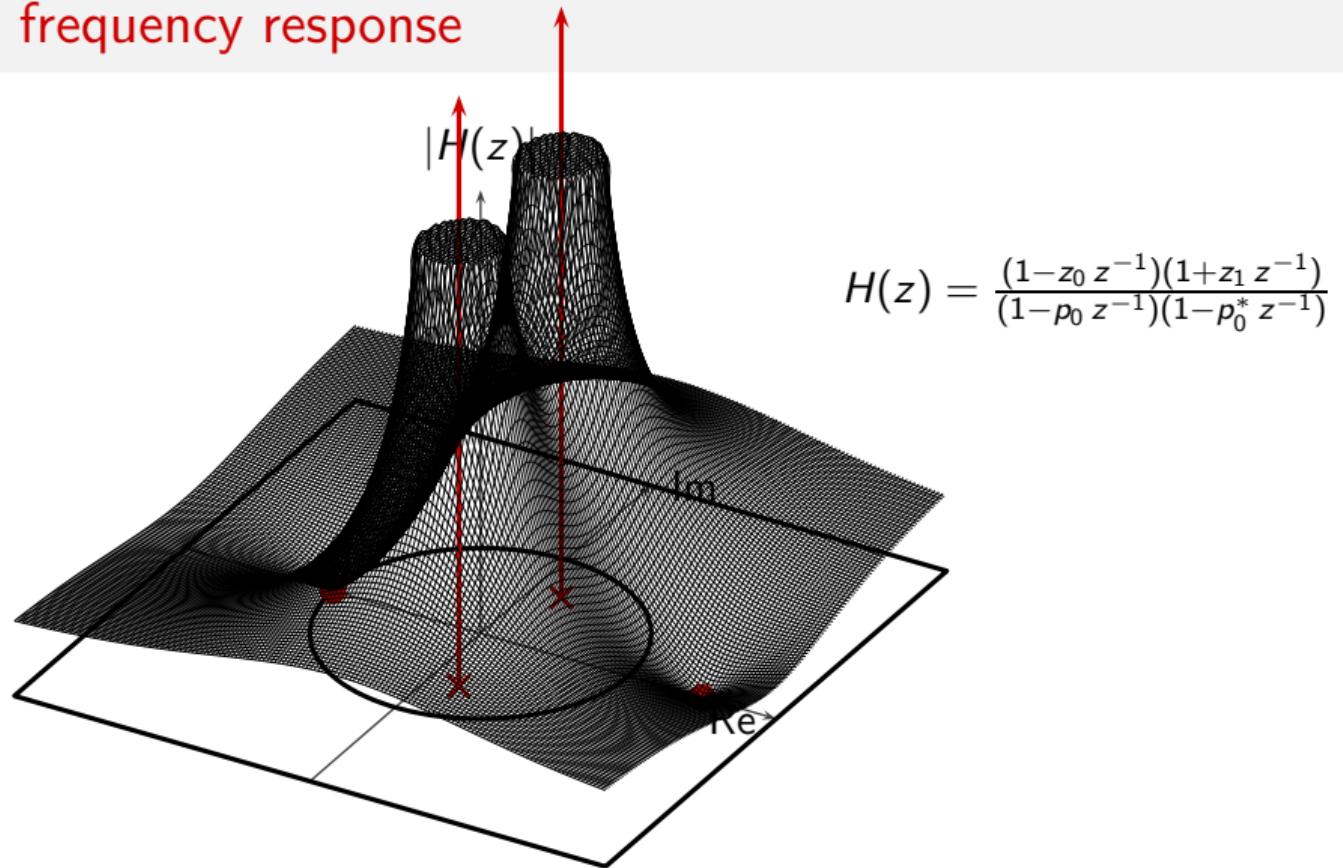
Estimating the frequency response



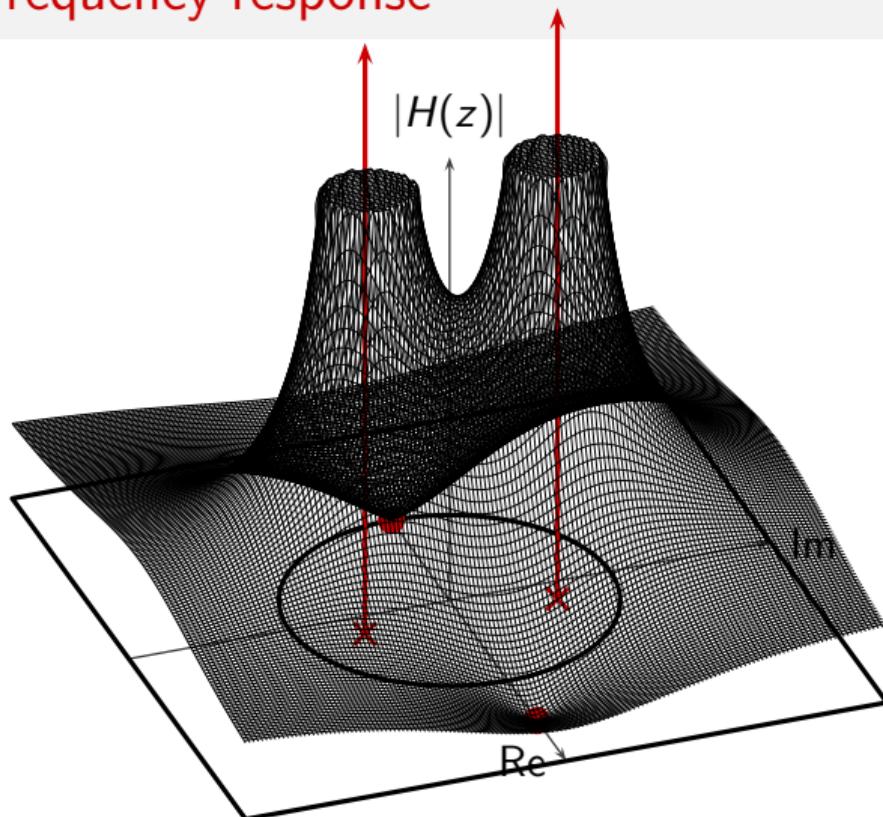
Estimating the frequency response



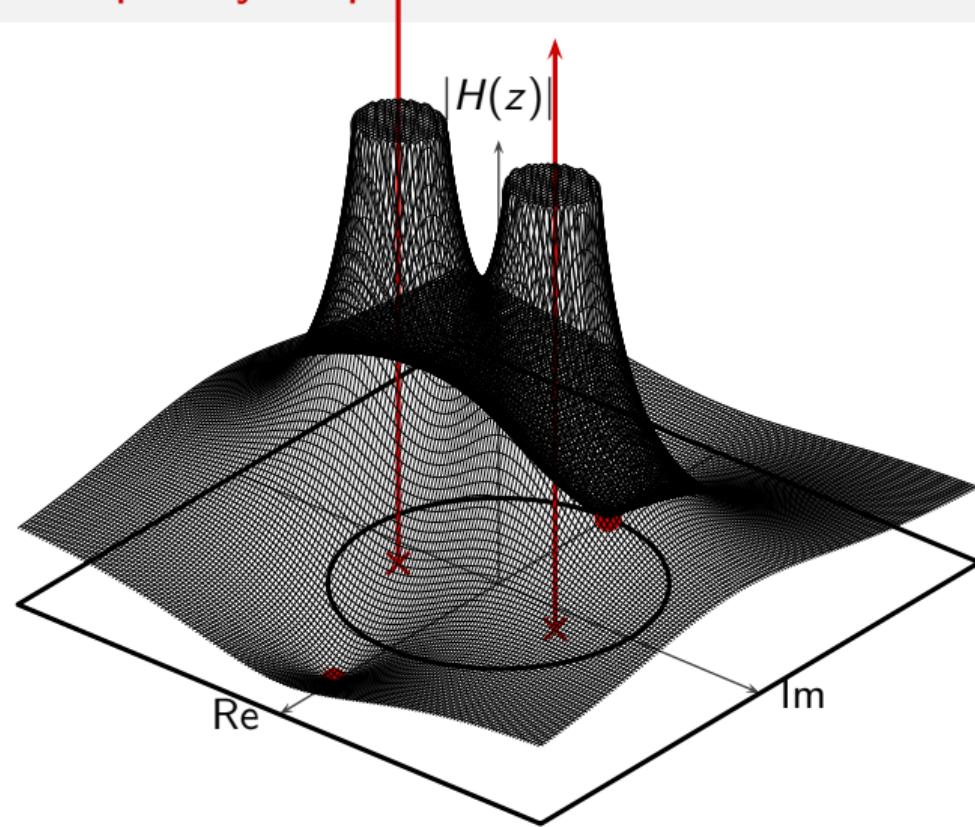
Estimating the frequency response



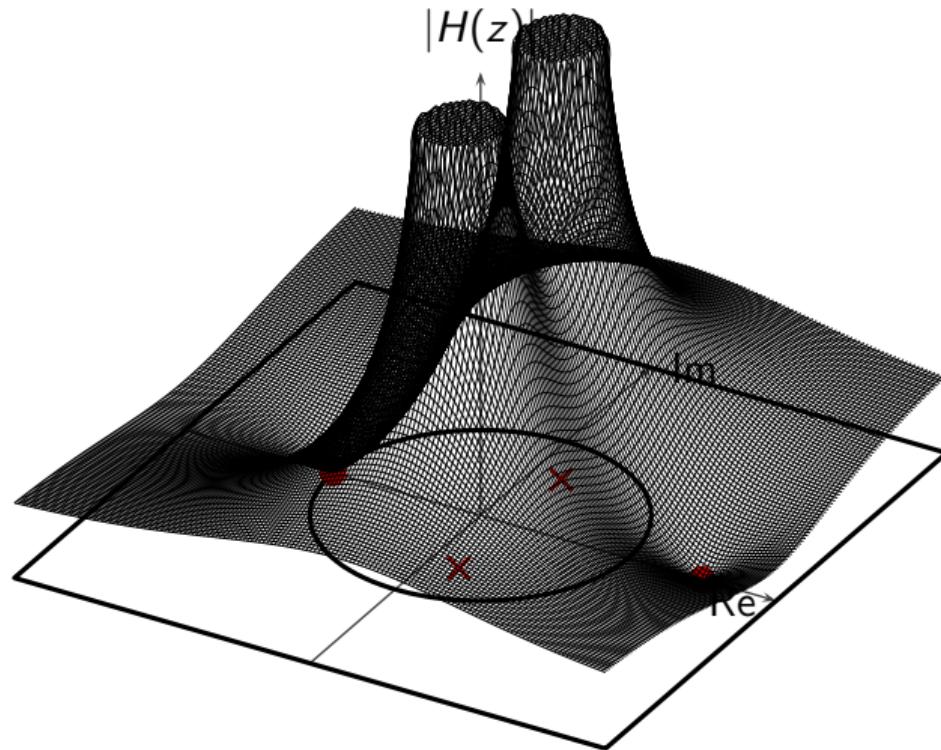
Estimating the frequency response



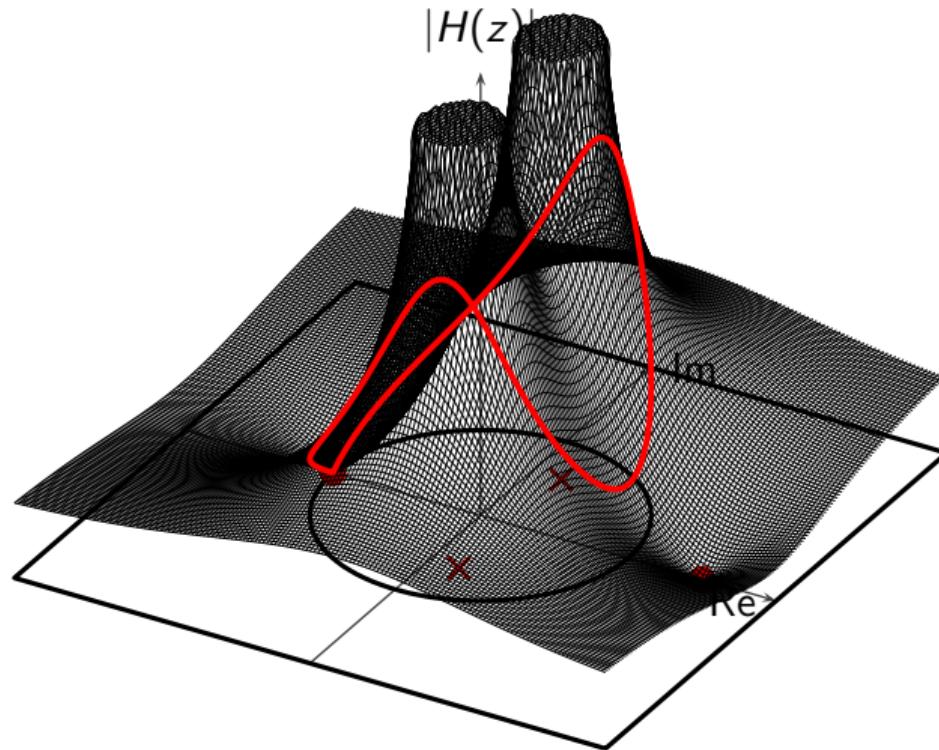
Estimating the frequency response



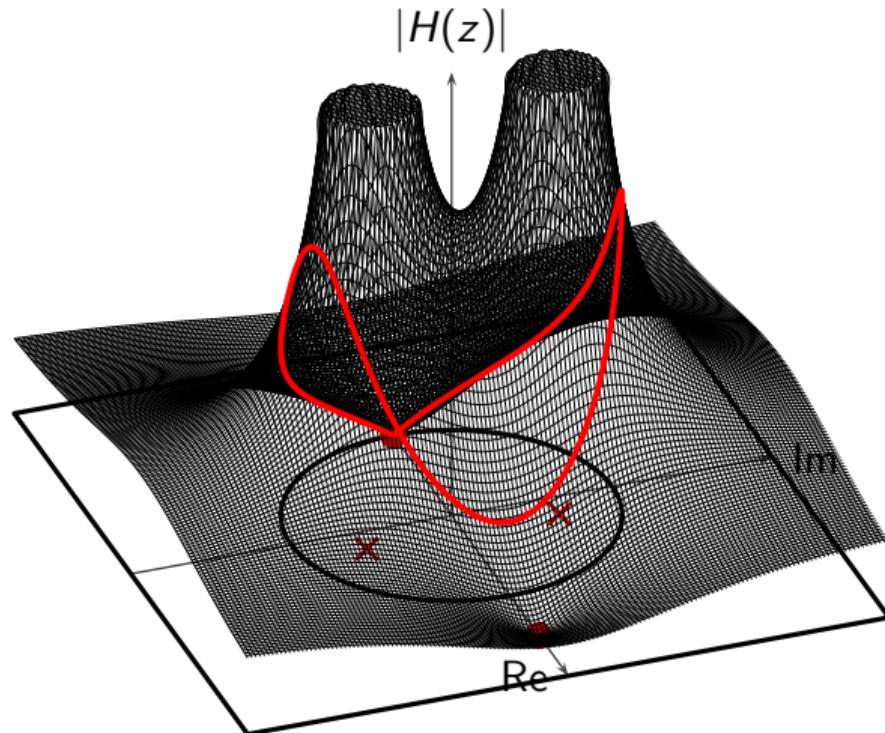
Estimating the frequency response



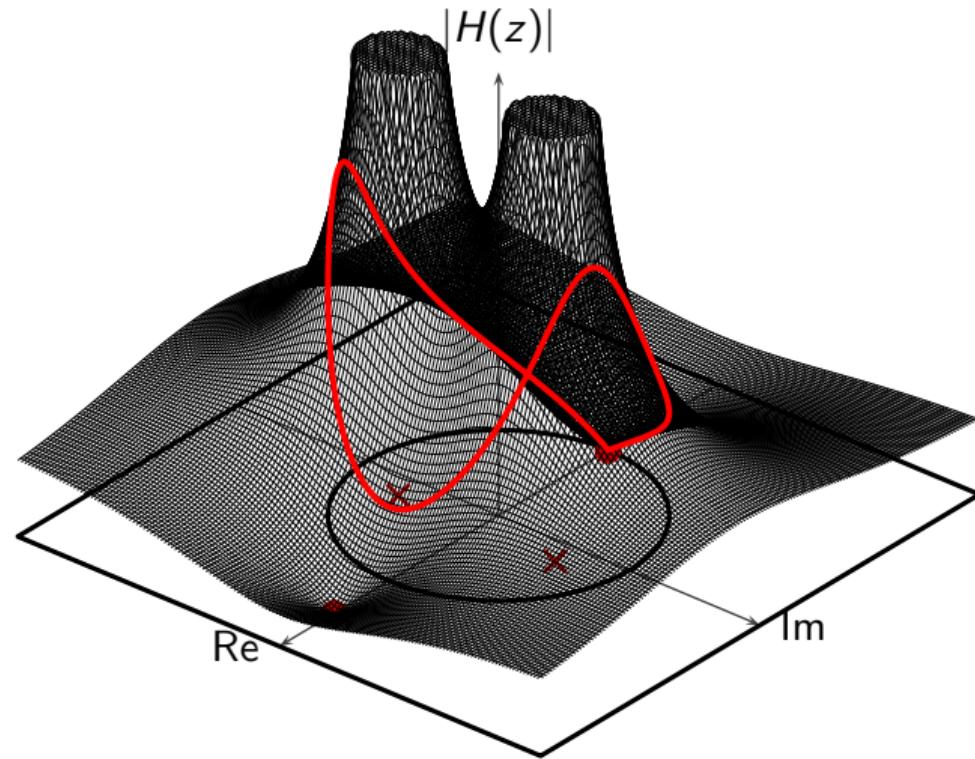
Estimating the frequency response



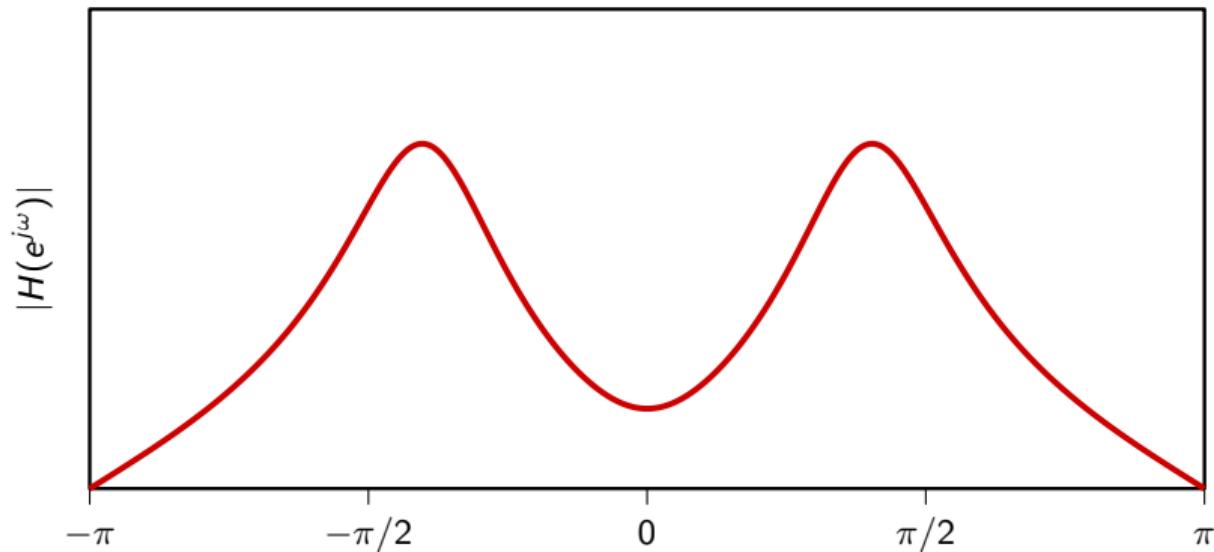
Estimating the frequency response



Estimating the frequency response



Estimating the frequency response



filtering algorithms and structures

Overview:

- ▶ Algorithms for CCDE's
- ▶ Block diagram
- ▶ Real-time processing

An old friend

```
class Leaky:  
    def __init__(self, lmb):  
        self.lmb=lmb  
        self.y=0  
  
    def compute(self, x):  
        self.y = self.lmb * self.y + (1 - self.lmb) * x  
        return self.y
```

Testing the code

```
>>> from leaky import Leaky  
>>> li = Leaky(0.95)  
>>> for v in [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]:  
>>>     print(li.compute(v), end=' ')  
  
0.0, 0.0, 0.0, 0.0, 0.0500000000000000, 0.0475000000000000,  
0.0451250000000000, 0.0428687500000000, 0.0407253125000000,  
0.038689046875000, 0.0367545945312500  
>>>
```

Key points

- ▶ we need a “memory cell” to store previous output
- ▶ we need to initialize the storage before first use
- ▶ we need 2 multiplications and one addition per output sample

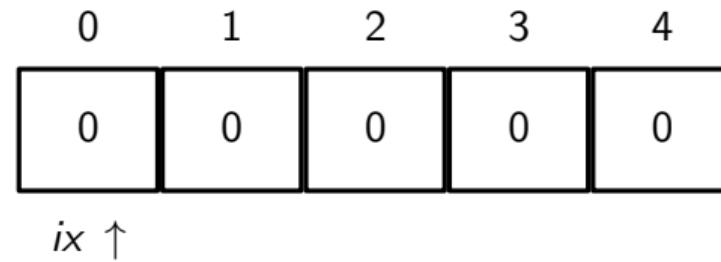
Another old friend

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n - k]$$

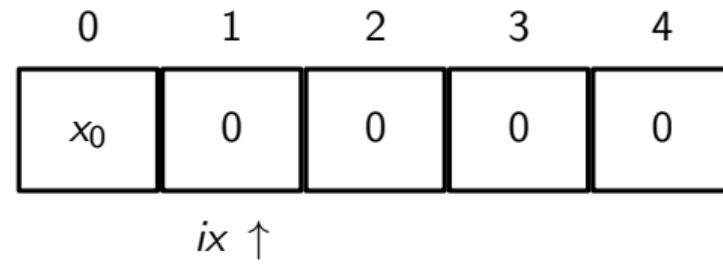
Another old friend

```
class MA:  
    def __init__(self, M):  
        self.M = M  
        self.buf = [0.0] * M  
        self.ix = 0  
  
    def compute(self, x):  
        self.buf[self.ix] = x  
        self.ix = (self.ix + 1) % self.M  
        res = 0.0  
        for v in self.buf:  
            res += v  
        return res / float(self.M)
```

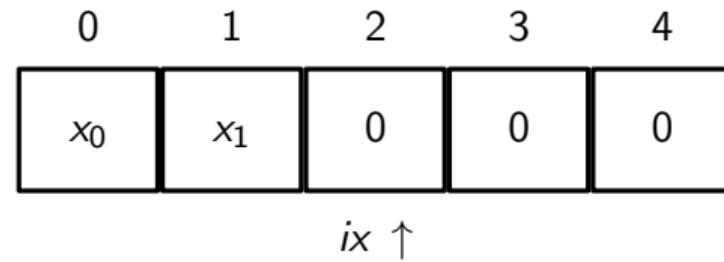
The circular buffer



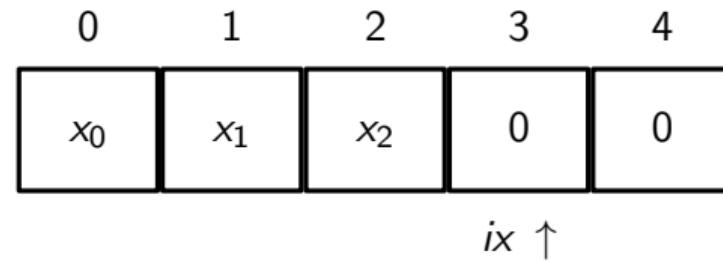
The circular buffer



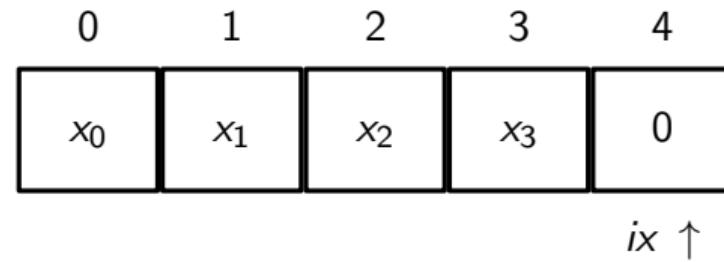
The circular buffer



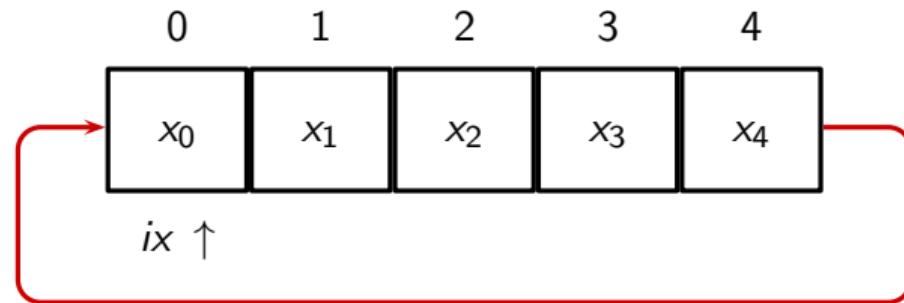
The circular buffer



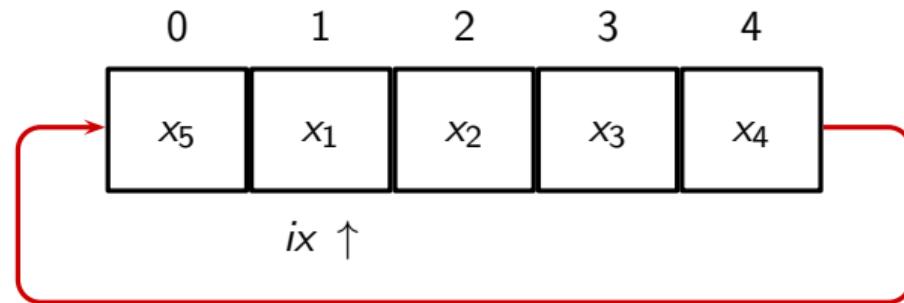
The circular buffer



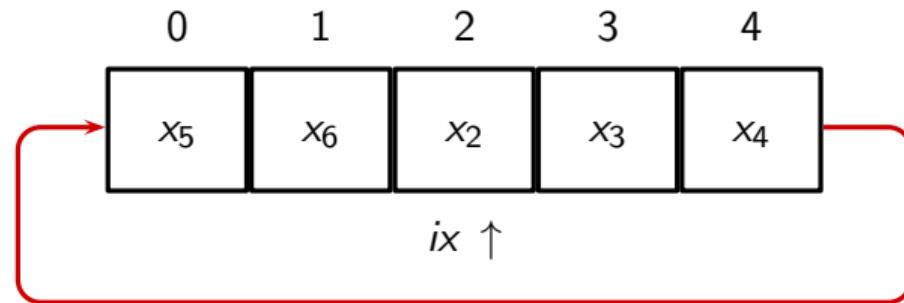
The circular buffer



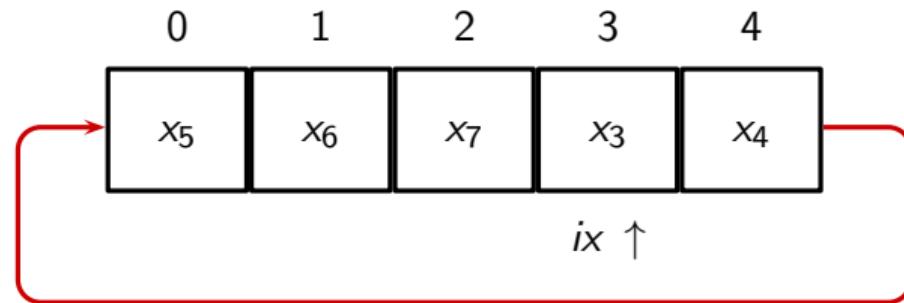
The circular buffer



The circular buffer



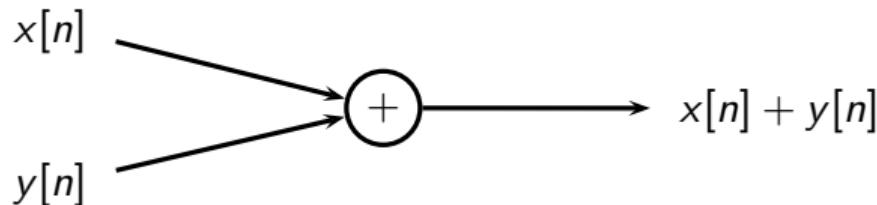
The circular buffer



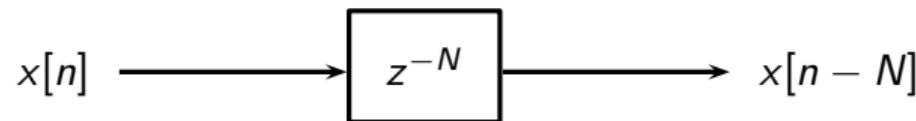
Key points

- ▶ we now need M memory cells to store previous input values
- ▶ we need to initialize the storage before first use
- ▶ we need 1 division and M additions per output sample

We can abstract from the implementation

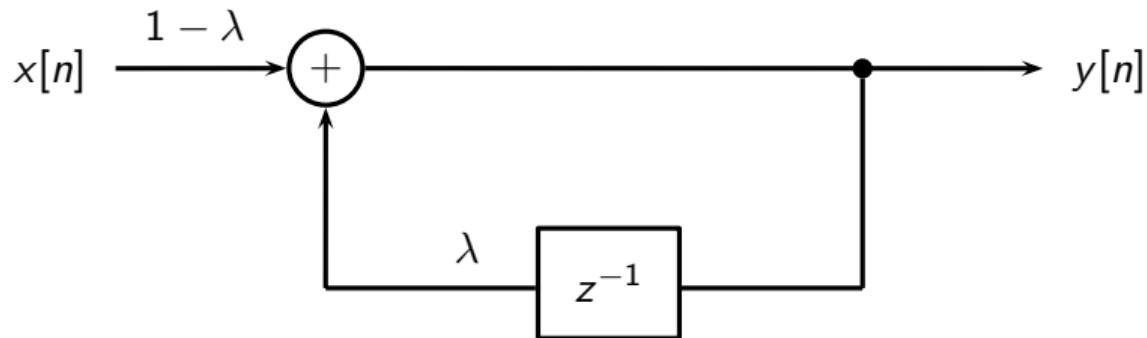


$$x[n] \xrightarrow{\alpha} \alpha x[n]$$



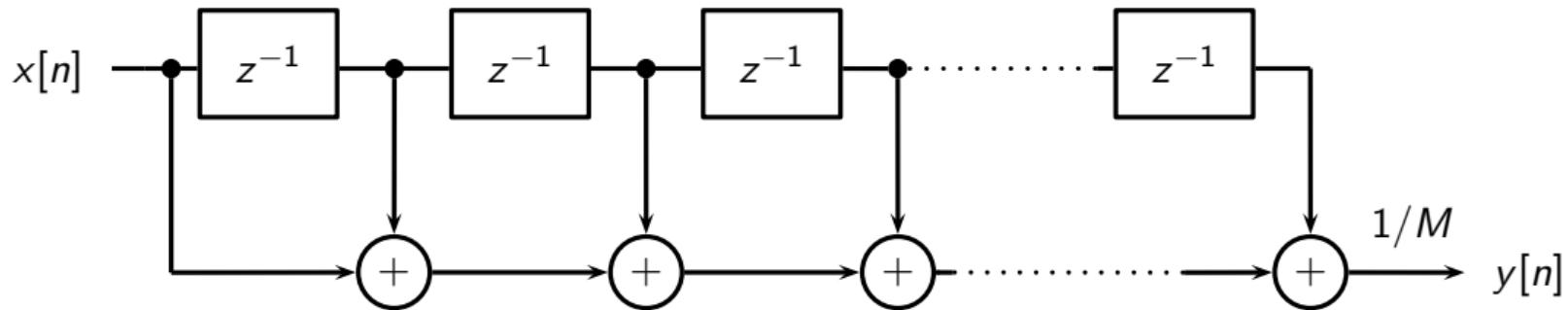
Leaky Integrator

$$y[n] = \lambda y[n - 1] + (1 - \lambda)x[n]$$



Moving Average

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n - k]$$



The second-order section

$$y[n] + a_1y[n - 1] + a_2y[n - 2] = b_0x[n] + b_1x[n - 1] + b_2x[n - 2]$$

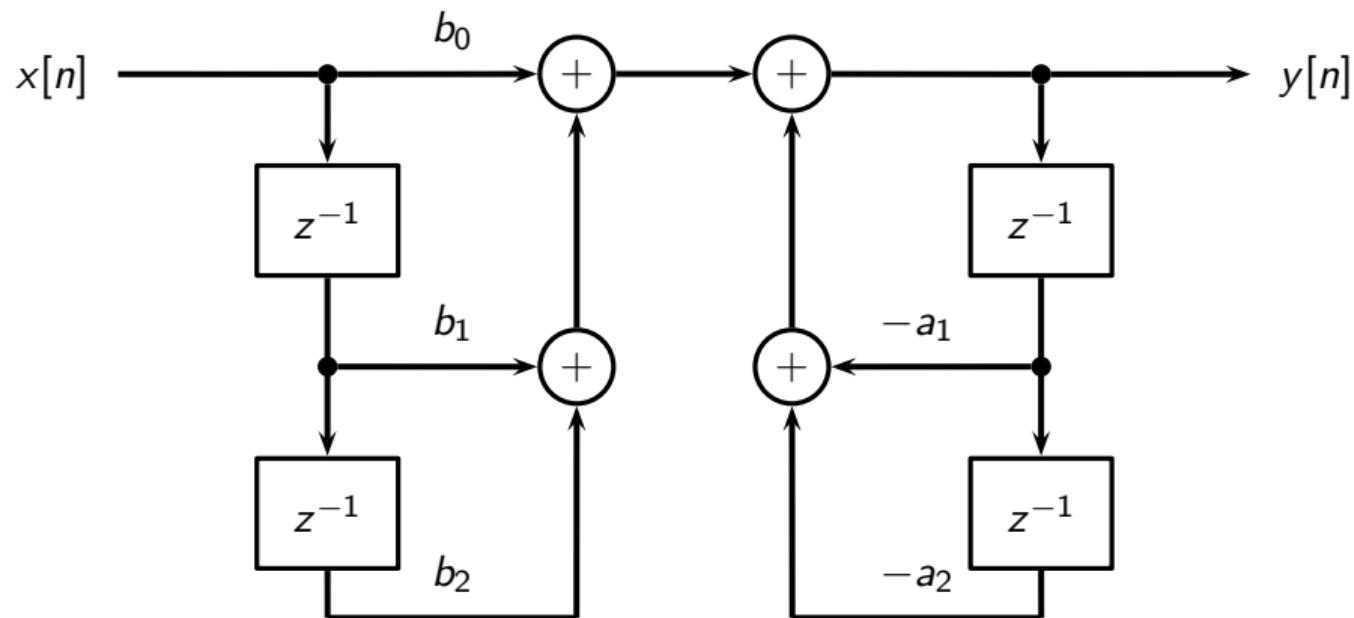
$$H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{1 + a_1z^{-1} + a_2z^{-2}} = \frac{B(z)}{A(z)}$$

The second-order section

$$y[n] + a_1y[n - 1] + a_2y[n - 2] = b_0x[n] + b_1x[n - 1] + b_2x[n - 2]$$

$$H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{1 + a_1z^{-1} + a_2z^{-2}} = \frac{B(z)}{A(z)}$$

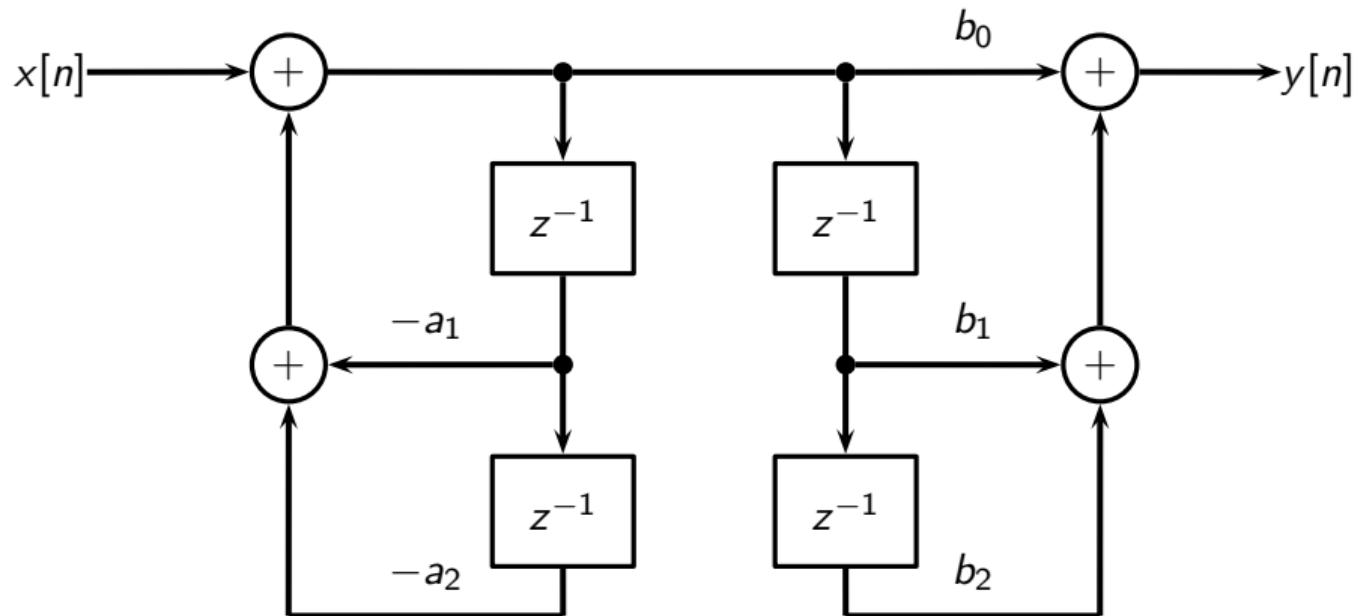
Second-order section, direct form I



$$B(z)$$

$$1/A(z)$$

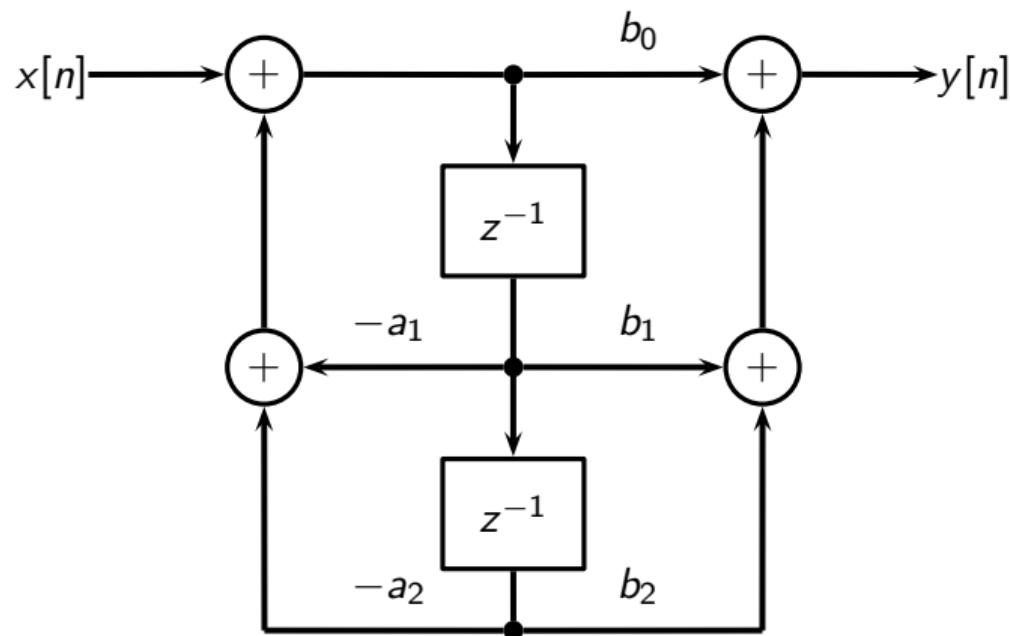
Second-order section, direct form I, inverted order



$$1/A(z)$$

$$B(z)$$

Second-order section, direct form II



intuitive filter design

Overview:

- ▶ General problem
- ▶ “Intuitive” IIR design

Simple, useful filters

- ▶ many signal processing problems can be solved using simple filters
- ▶ we have seen simple lowpass filters already (Moving Average, Leaky Integrator)
- ▶ simple (low order) transfer functions allow for intuitive design and tuning

Simple lowpass

- ▶ let only low frequencies pass
- ▶ used to remove high frequency components (e.g. noise)
- ▶ useful in audio, communication, control systems
- ▶ we know a simple answer: leaky integrator

Simple lowpass

- ▶ let only low frequencies pass
- ▶ used to remove high frequency components (e.g. noise)
- ▶ useful in audio, communication, control systems
- ▶ we know a simple answer: leaky integrator

Simple lowpass

- ▶ let only low frequencies pass
- ▶ used to remove high frequency components (e.g. noise)
- ▶ useful in audio, communication, control systems
- ▶ we know a simple answer: leaky integrator

Simple lowpass

- ▶ let only low frequencies pass
- ▶ used to remove high frequency components (e.g. noise)
- ▶ useful in audio, communication, control systems
- ▶ we know a simple answer: leaky integrator

Leaky Integrator

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

Leaky Integrator

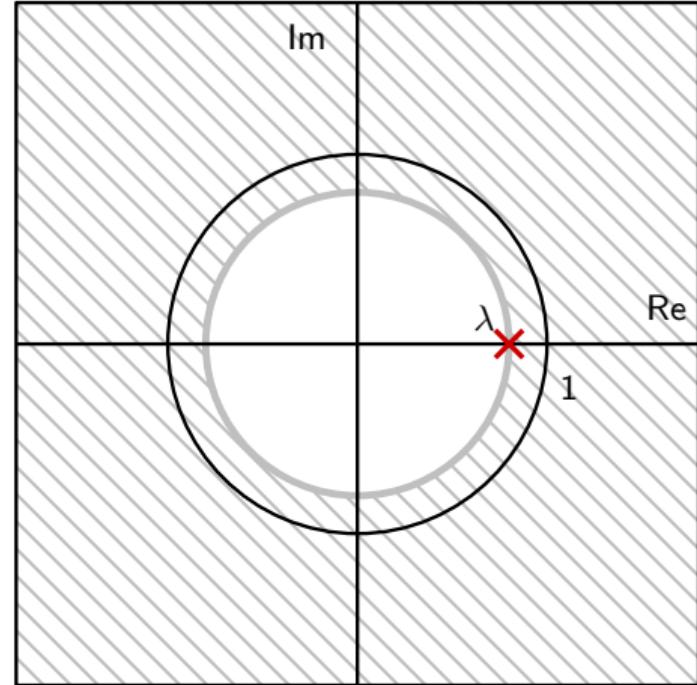
$$H(z) = \frac{(1 - \lambda)}{1 - \lambda z^{-1}}$$

$$y[n] = (1 - \lambda)x[n] + \lambda y[n - 1]$$

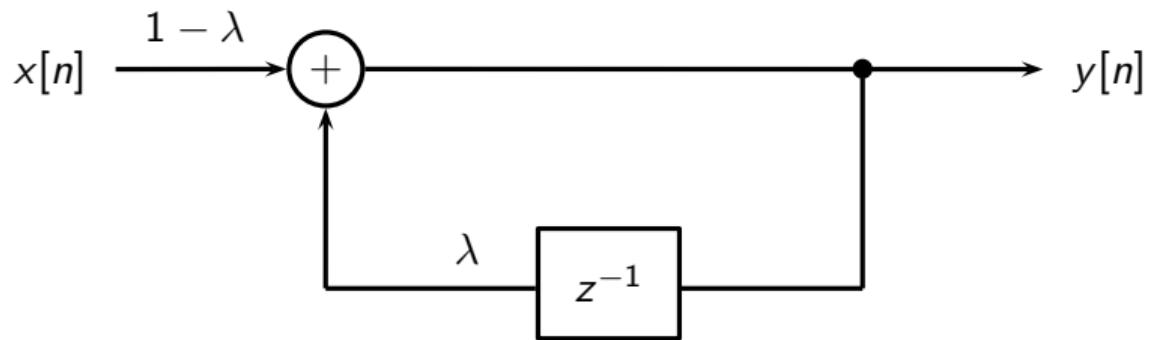
Leaky Integrator

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

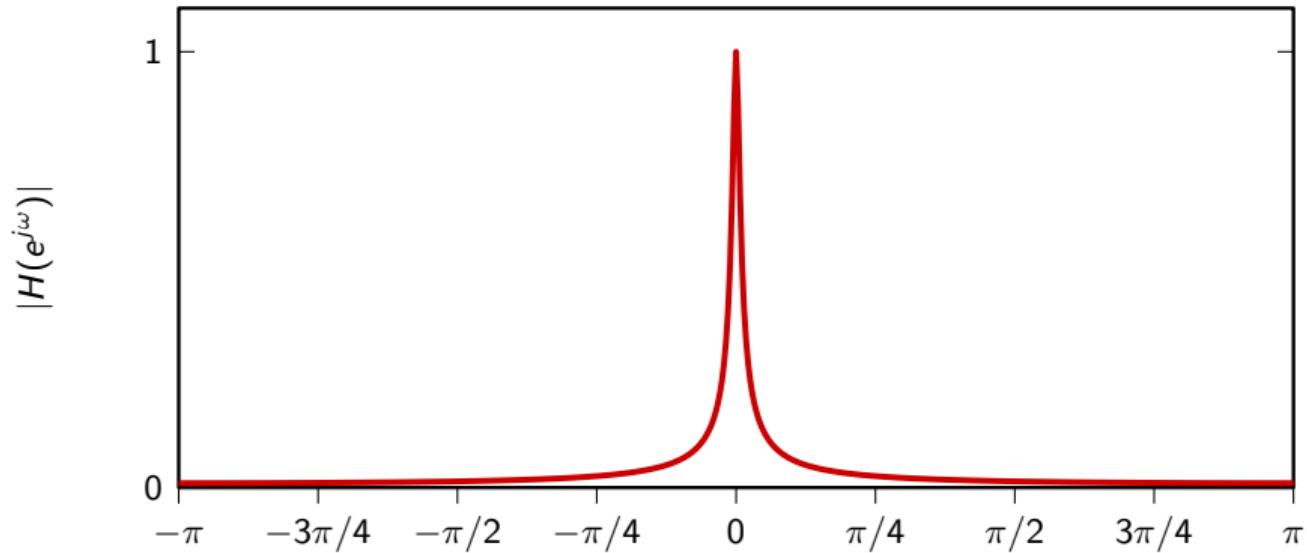
$$y[n] = (1 - \lambda)x[n] + \lambda y[n - 1]$$



Leaky Integrator, filter structure



Leaky Integrator, $\lambda = 0.98$



Resonator

- ▶ a resonator is a narrow bandpass filter
- ▶ used to detect the presence of a sinusoid of a given frequency
- ▶ useful in communication systems and telephony (DTMF)
- ▶ idea: shift the passband of the Leaky Integrator!

Resonator

- ▶ a resonator is a narrow bandpass filter
- ▶ used to detect the presence of a sinusoid of a given frequency
- ▶ useful in communication systems and telephony (DTMF)
- ▶ idea: shift the passband of the Leaky Integrator!

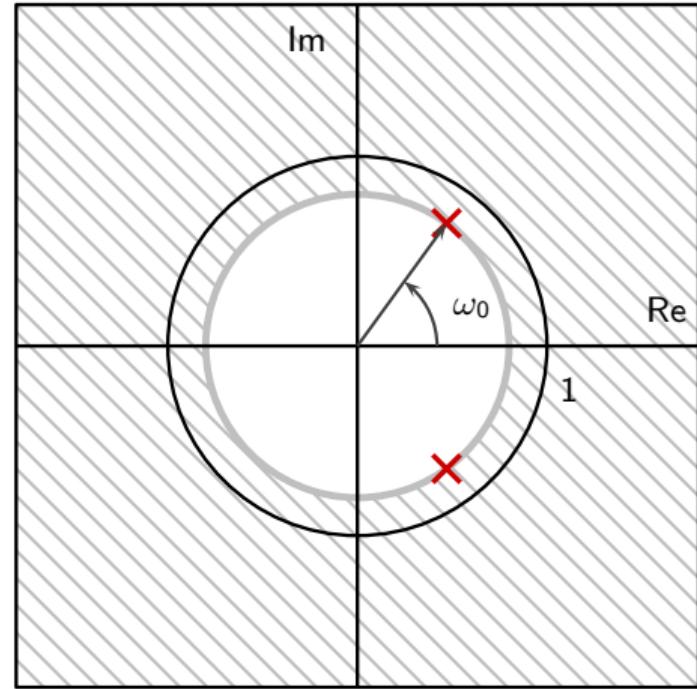
Resonator

- ▶ a resonator is a narrow bandpass filter
- ▶ used to detect the presence of a sinusoid of a given frequency
- ▶ useful in communication systems and telephony (DTMF)
- ▶ idea: shift the passband of the Leaky Integrator!

Resonator

- ▶ a resonator is a narrow bandpass filter
- ▶ used to detect the presence of a sinusoid of a given frequency
- ▶ useful in communication systems and telephony (DTMF)
- ▶ idea: shift the passband of the Leaky Integrator!

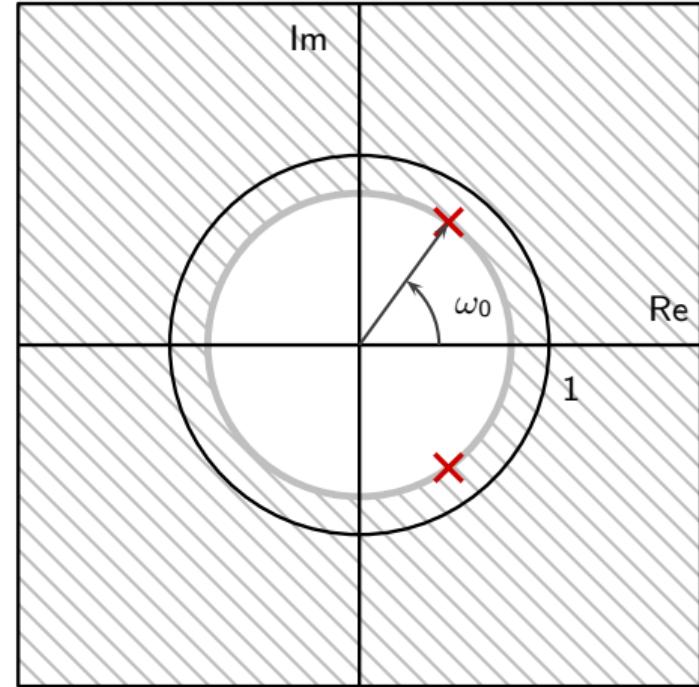
Resonator



Resonator

$$H(z) = \frac{G_0}{(1 - pz^{-1})(1 - p^*z^{-1})}$$

$$p = \lambda e^{j\omega_0}$$

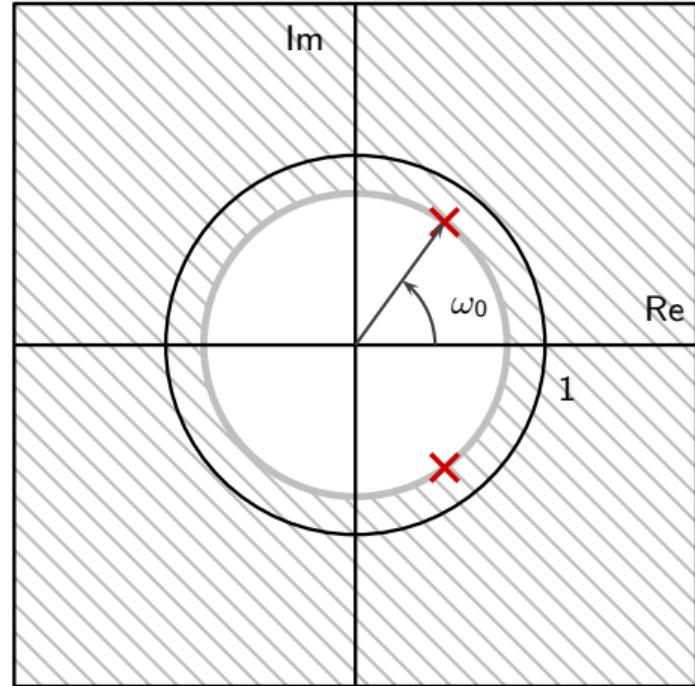


Resonator

$$H(z) = \frac{G_0}{(1 - pz^{-1})(1 - p^*z^{-1})}$$

$$p = \lambda e^{j\omega_0}$$

$$y[n] = G_0x[n] - a_1y[n-1] - a_2y[n-2]$$



Resonator

$$\begin{aligned} H(z) &= \frac{G_0}{(1 - pz^{-1})(1 - p^*z^{-1})}, \quad p = \lambda e^{j\omega_0} \\ &= \frac{G_0}{1 - 2\Re\{p\} z^{-1} + |p|^2 z^{-2}} \\ &= \frac{G_0}{1 - 2\lambda \cos \omega_0 z^{-1} + |\lambda|^2 z^{-2}} \end{aligned}$$

$$a_1 = 2\lambda \cos \omega_0$$

$$a_2 = -|\lambda|^2$$

Resonator

$$\begin{aligned} H(z) &= \frac{G_0}{(1 - pz^{-1})(1 - p^*z^{-1})}, \quad p = \lambda e^{j\omega_0} \\ &= \frac{G_0}{1 - 2\Re\{p\} z^{-1} + |p|^2 z^{-2}} \\ &= \frac{G_0}{1 - 2\lambda \cos \omega_0 z^{-1} + |\lambda|^2 z^{-2}} \end{aligned}$$

$$a_1 = 2\lambda \cos \omega_0$$

$$a_2 = -|\lambda|^2$$

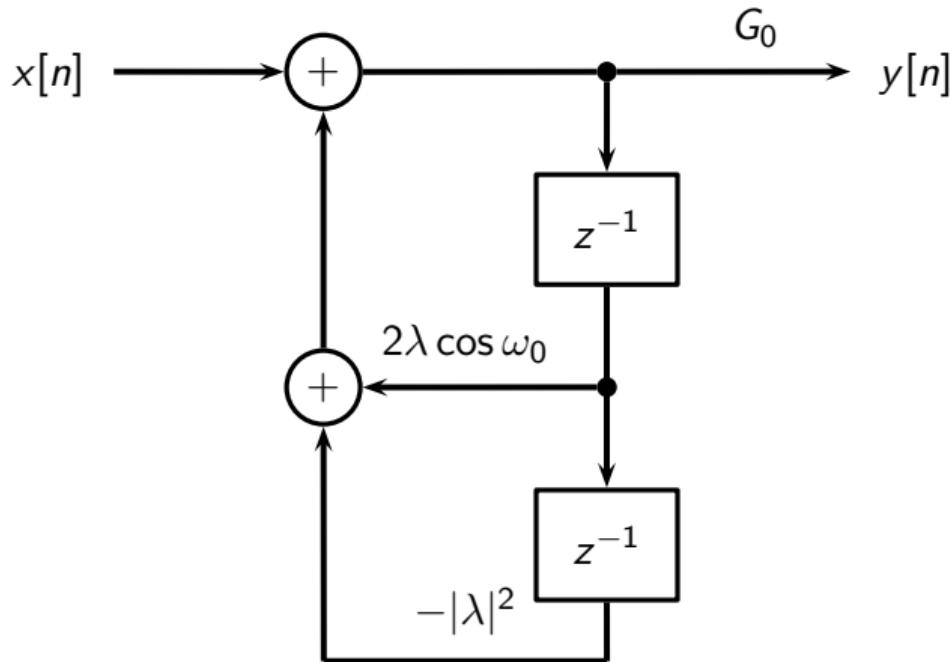
Resonator

$$\begin{aligned} H(z) &= \frac{G_0}{(1 - pz^{-1})(1 - p^*z^{-1})}, \quad p = \lambda e^{j\omega_0} \\ &= \frac{G_0}{1 - 2\Re\{p\} z^{-1} + |p|^2 z^{-2}} \\ &= \frac{G_0}{1 - 2\lambda \cos \omega_0 z^{-1} + |\lambda|^2 z^{-2}} \end{aligned}$$

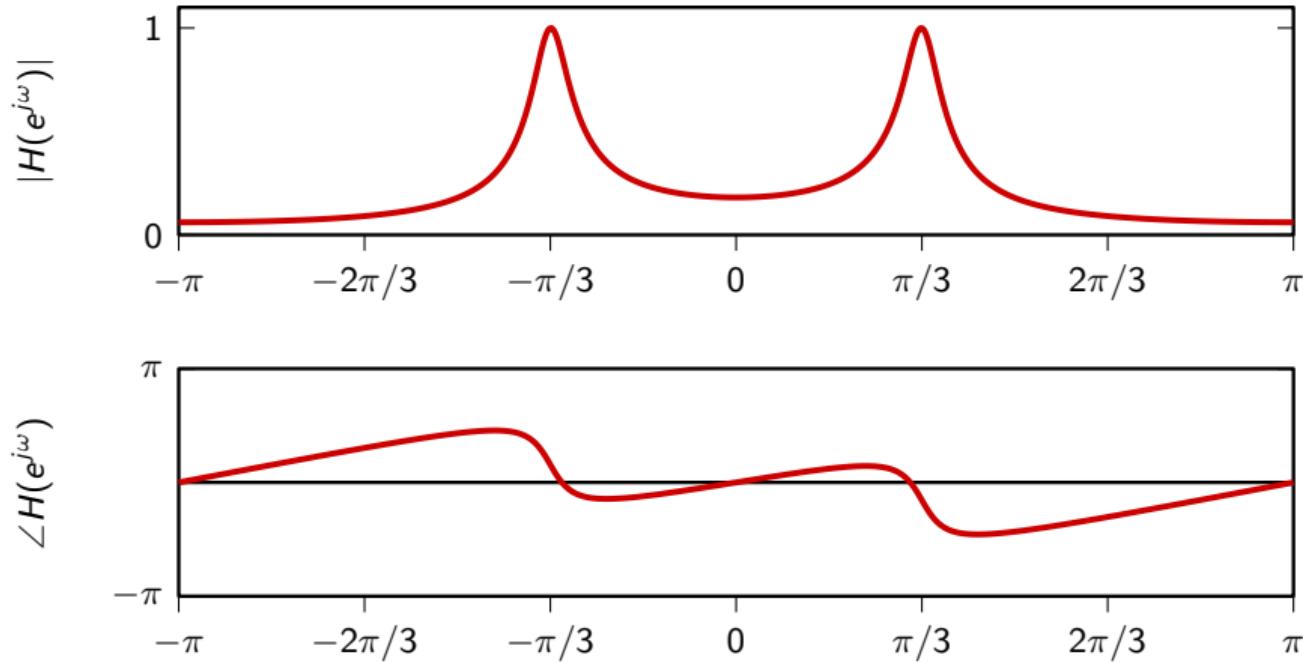
$$a_1 = 2\lambda \cos \omega_0$$

$$a_2 = -|\lambda|^2$$

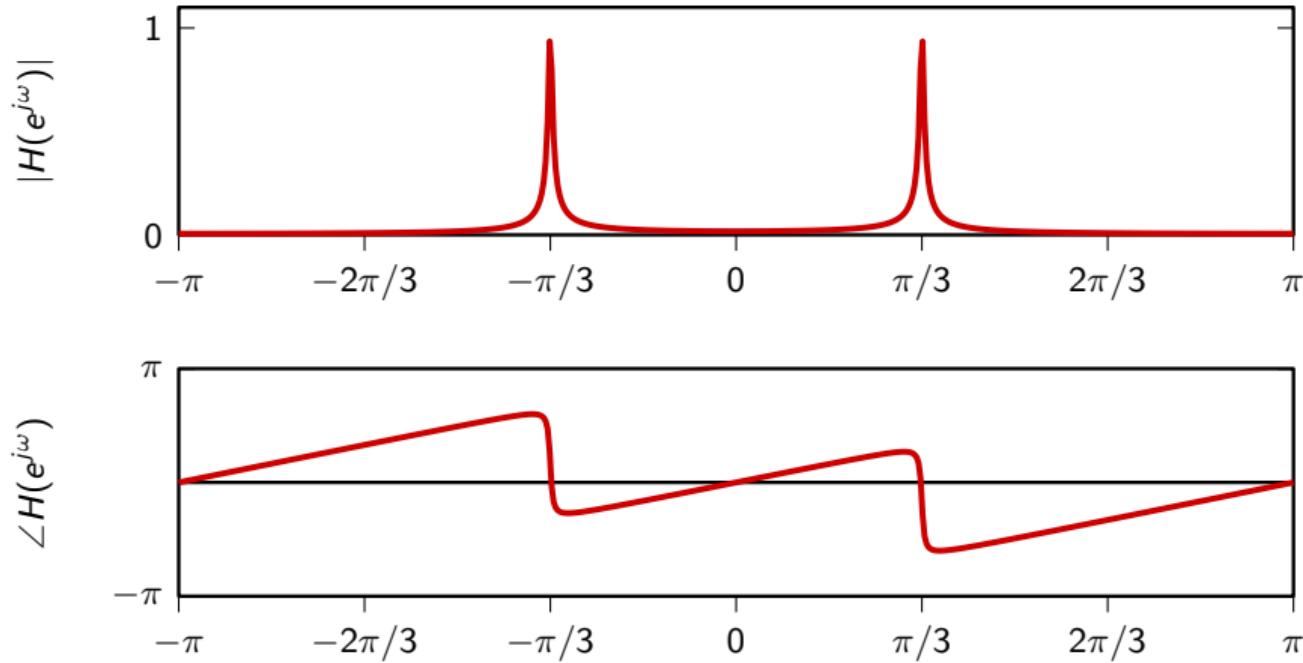
Resonator, filter structure



Resonator, $\lambda = 0.9, \omega_0 = \pi/3$



Resonator, $\lambda = 0.99$, $\omega_0 = \pi/3$



DC removal

- ▶ a DC-balanced signal has zero sum: $\lim_{N \rightarrow \infty} \sum_{n=-N}^N x[n] = 0$
i.e. there is no Direct Current component
- ▶ its DTFT value at zero is zero
- ▶ we want to remove the DC bias from a non zero-centered signal
- ▶ we want to kill the frequency component at $\omega = 0$

DC removal

- ▶ a DC-balanced signal has zero sum: $\lim_{N \rightarrow \infty} \sum_{n=-N}^N x[n] = 0$
i.e. there is no Direct Current component
- ▶ its DTFT value at zero is zero
- ▶ we want to remove the DC bias from a non zero-centered signal
- ▶ we want to kill the frequency component at $\omega = 0$

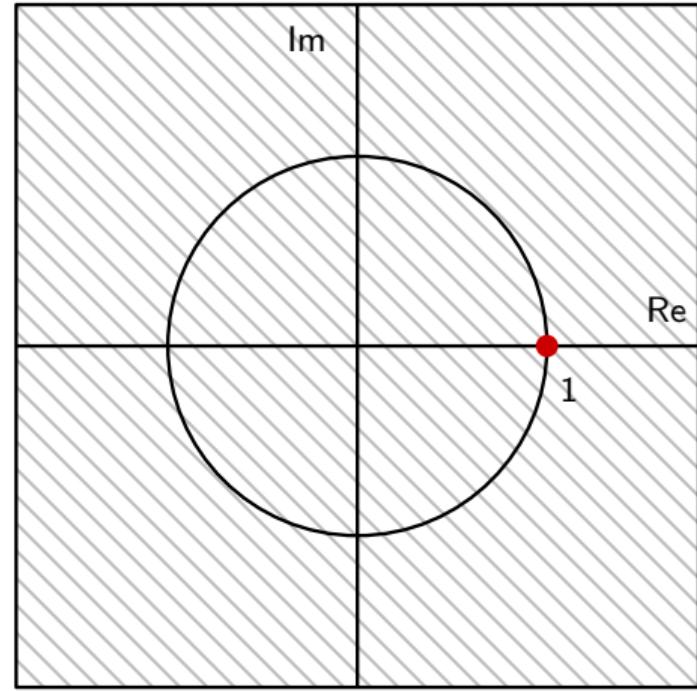
DC removal

- ▶ a DC-balanced signal has zero sum: $\lim_{N \rightarrow \infty} \sum_{n=-N}^N x[n] = 0$
i.e. there is no Direct Current component
- ▶ its DTFT value at zero is zero
- ▶ we want to remove the DC bias from a non zero-centered signal
- ▶ we want to kill the frequency component at $\omega = 0$

DC removal

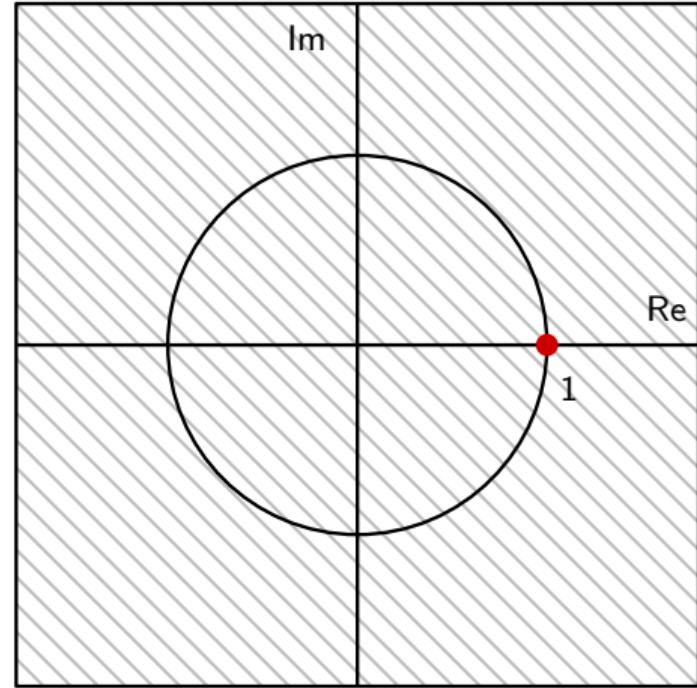
- ▶ a DC-balanced signal has zero sum: $\lim_{N \rightarrow \infty} \sum_{n=-N}^N x[n] = 0$
i.e. there is no Direct Current component
- ▶ its DTFT value at zero is zero
- ▶ we want to remove the DC bias from a non zero-centered signal
- ▶ we want to kill the frequency component at $\omega = 0$

DC removal



DC removal

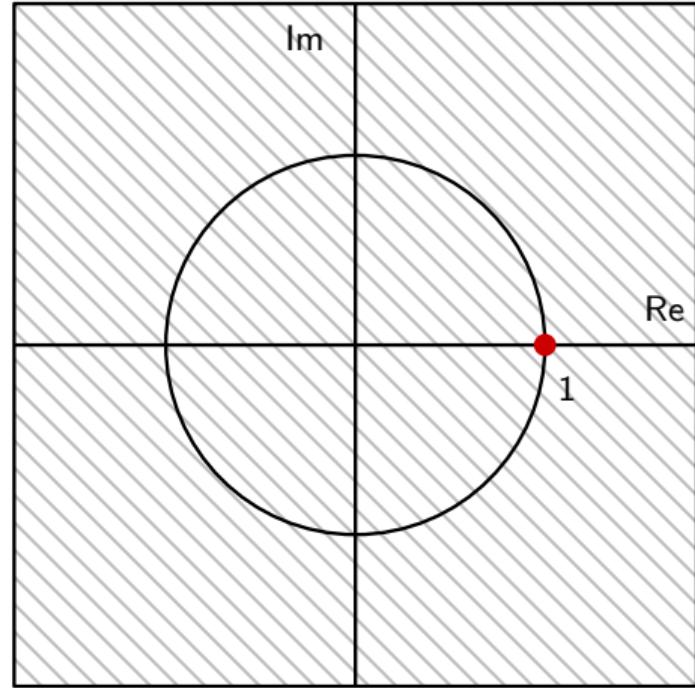
$$H(z) = 1 - z^{-1}$$



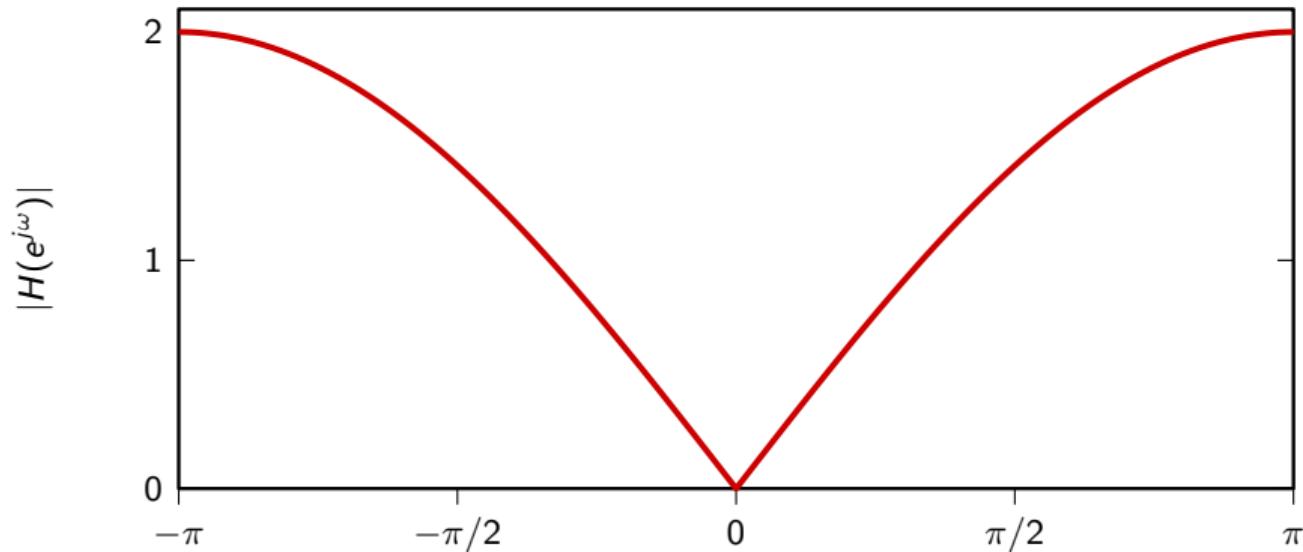
DC removal

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

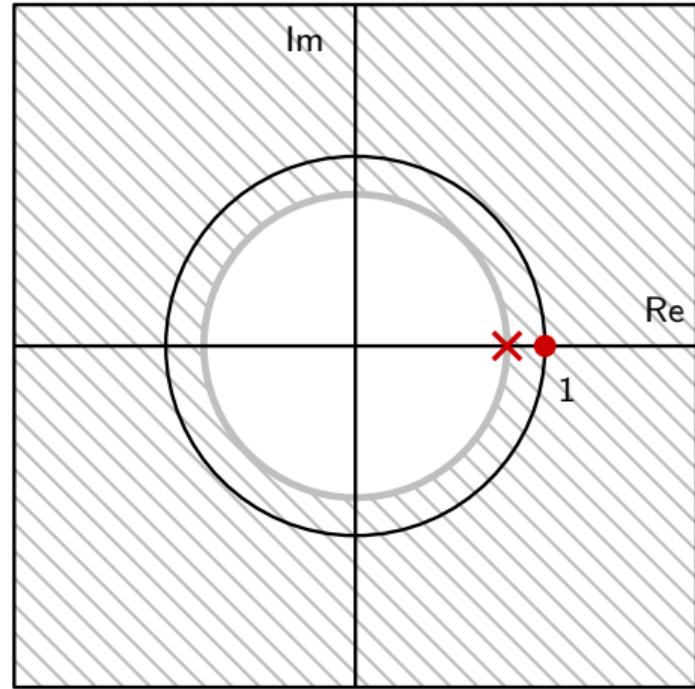
$$y[n] = x[n] - x[n - 1]$$



DC notch

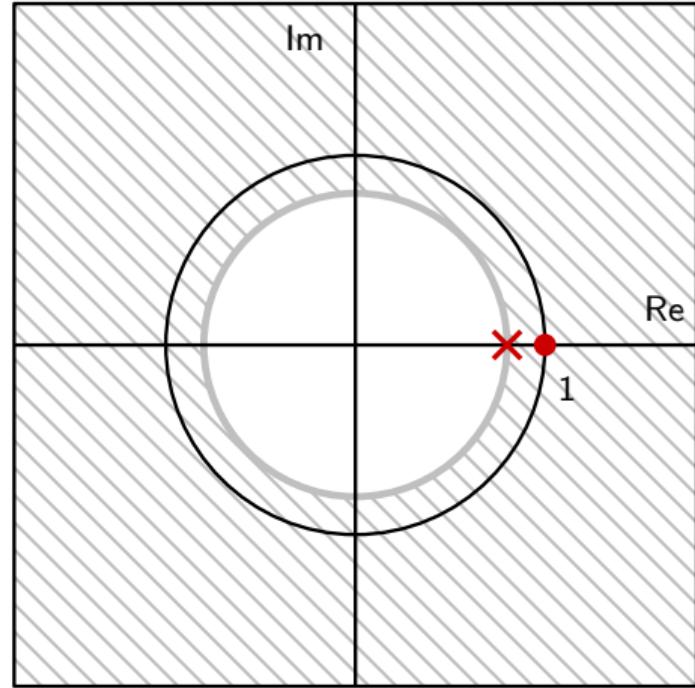


DC removal, improved



DC removal, improved

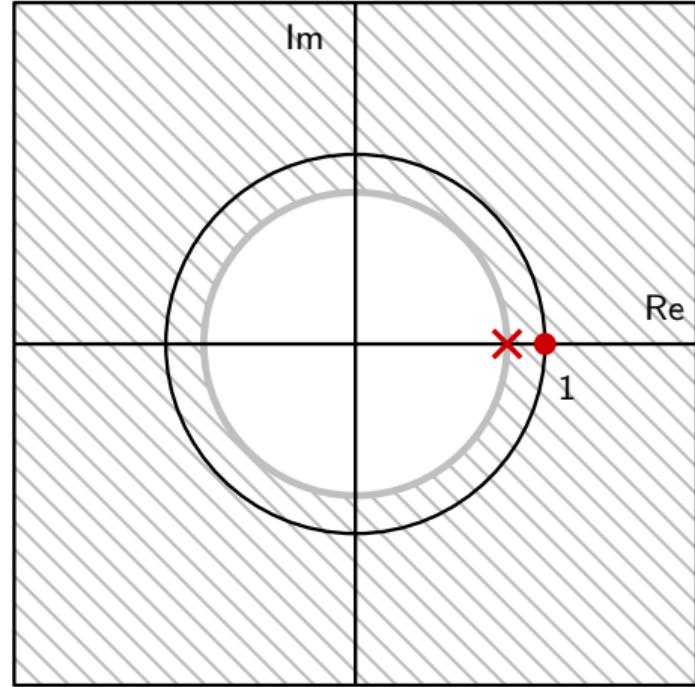
$$H(z) = \frac{1 - z^{-1}}{1 - \lambda z^{-1}}$$



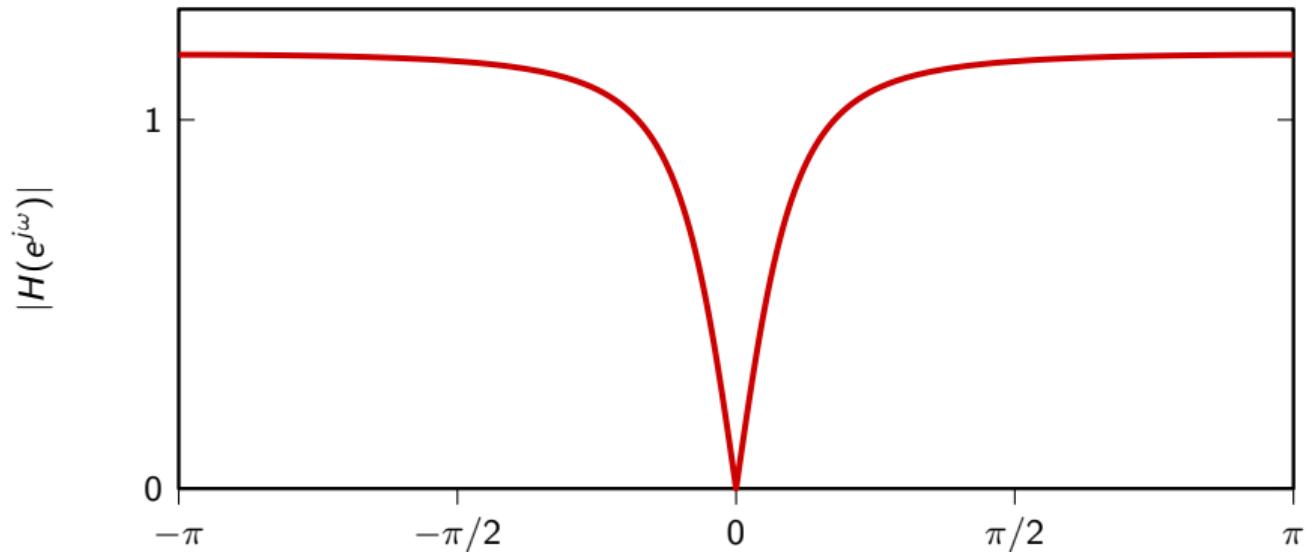
DC removal, improved

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

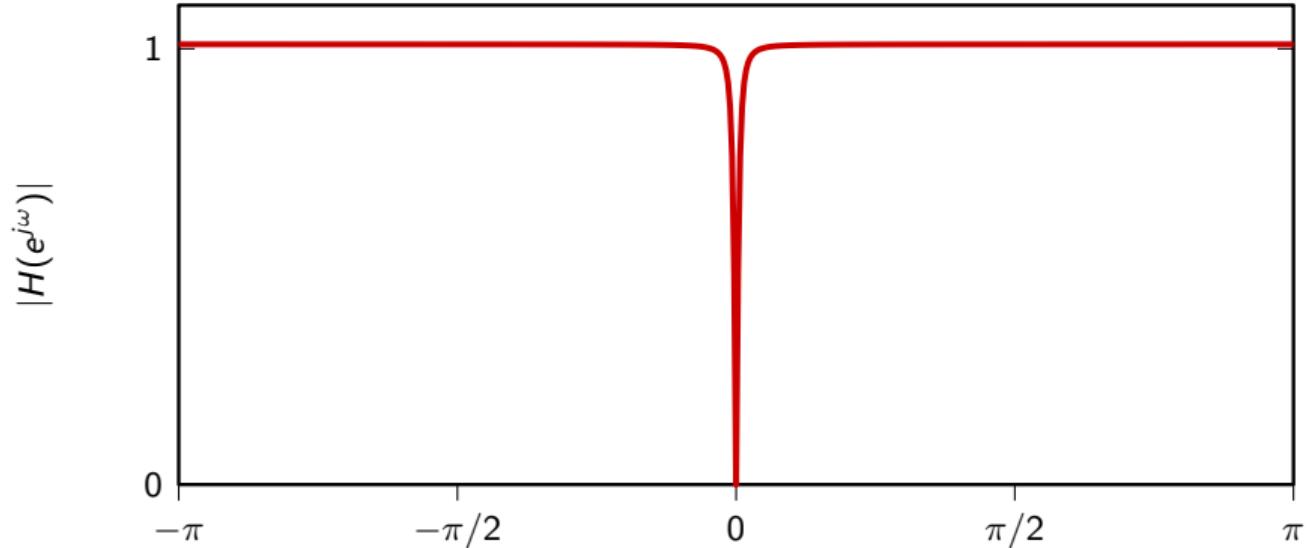
$$y[n] = \lambda y[n-1] + x[n] - x[n-1]$$



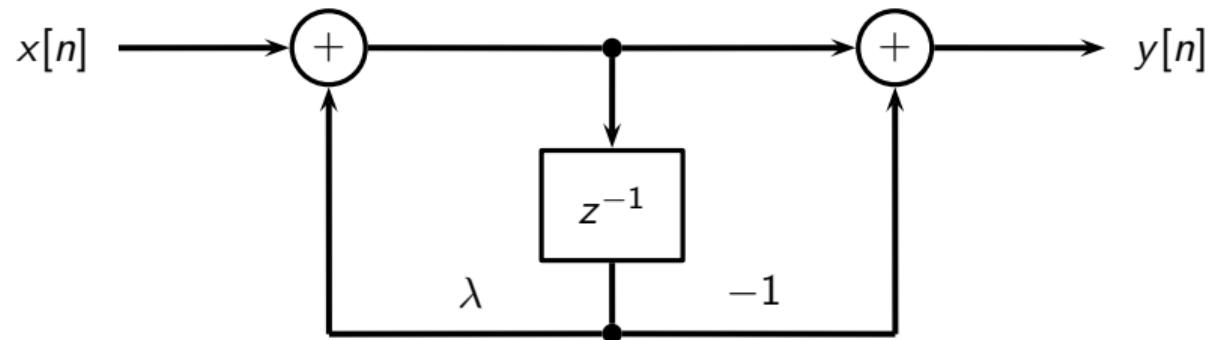
DC notch, $\lambda = 0.7$



DC notch, $\lambda = 0.98$



DC notch, filter structure



Hum removal

- ▶ similar to DC removal but we want to remove a specific nonzero frequency
- ▶ very useful for musicians: amplifiers for electric guitars pick up the hum from the electric mains (50Hz in Europe and 60Hz in North America)
- ▶ we need to tune the hum removal according to country

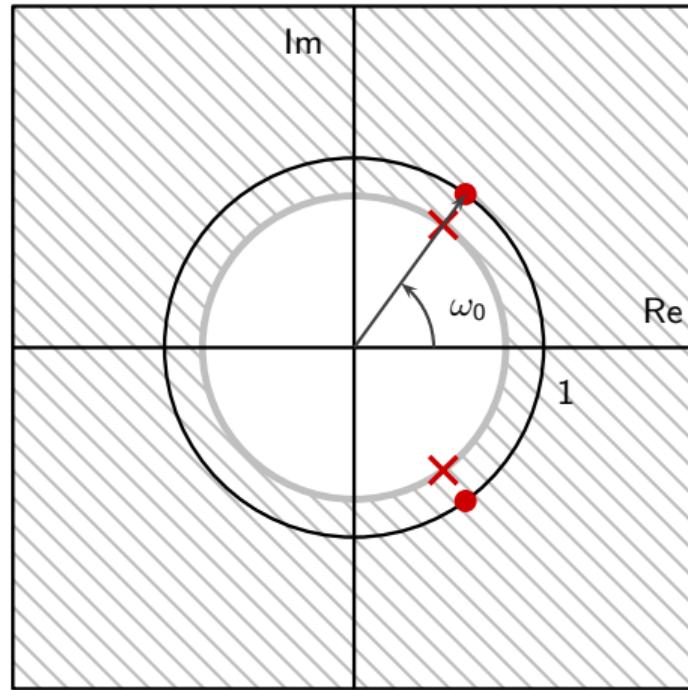
Hum removal

- ▶ similar to DC removal but we want to remove a specific nonzero frequency
- ▶ very useful for musicians: amplifiers for electric guitars pick up the hum from the electric mains (50Hz in Europe and 60Hz in North America)
- ▶ we need to tune the hum removal according to country

Hum removal

- ▶ similar to DC removal but we want to remove a specific nonzero frequency
- ▶ very useful for musicians: amplifiers for electric guitars pick up the hum from the electric mains (50Hz in Europe and 60Hz in North America)
- ▶ we need to tune the hum removal according to country

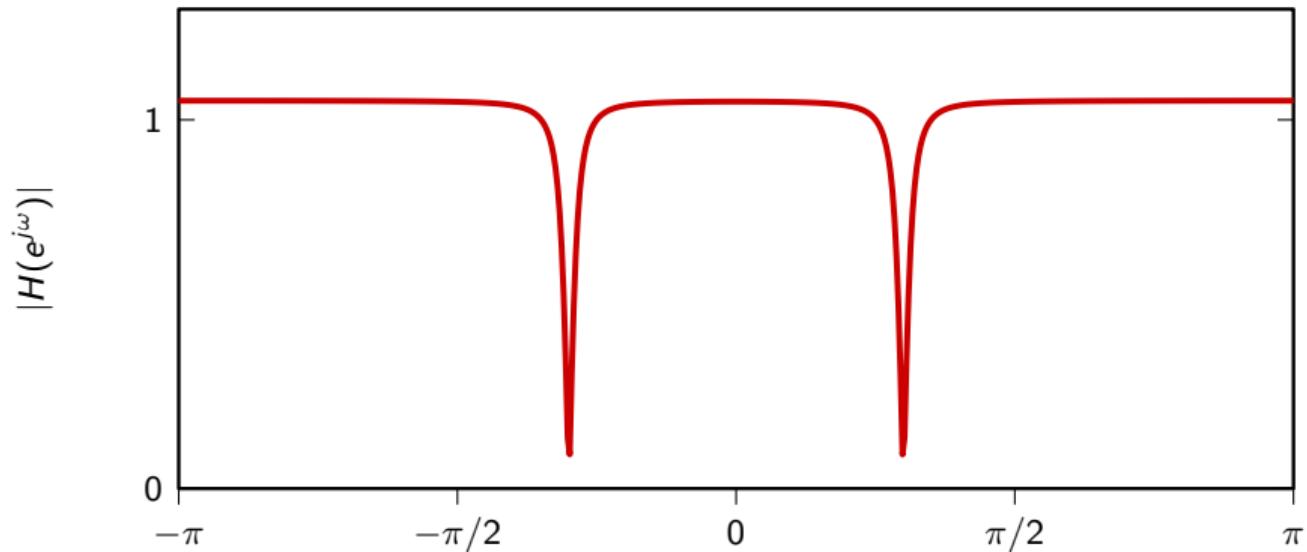
Hum removal



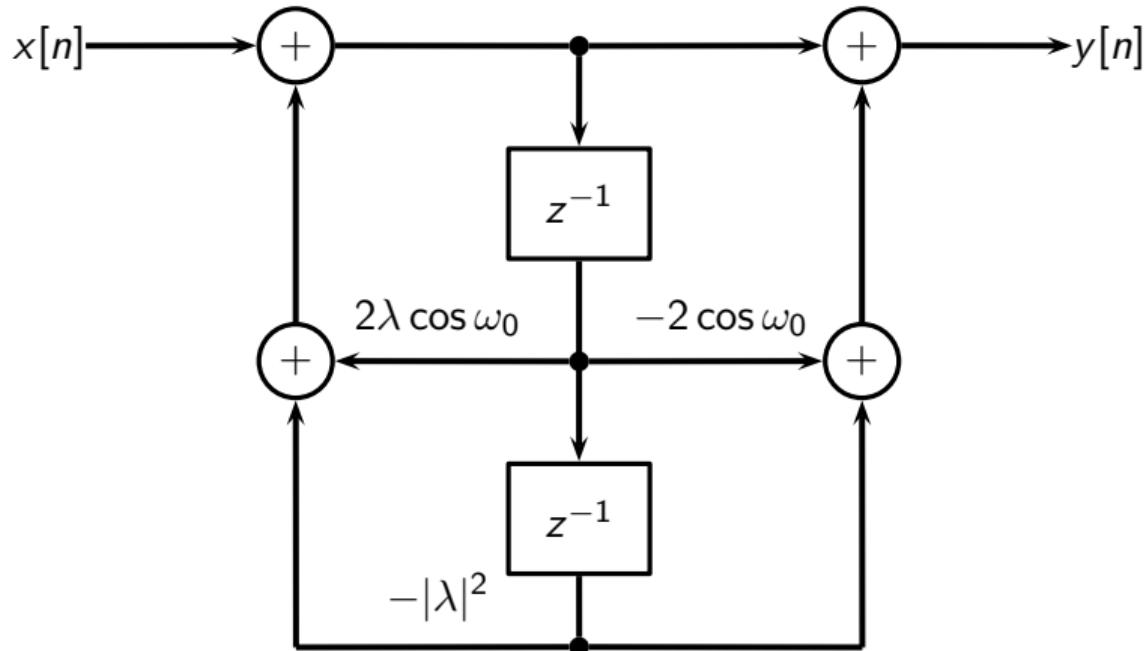
Hum removal

$$H(z) = \frac{(1 - e^{j\omega_0} z^{-1})(1 - e^{-j\omega_0} z^{-1})}{(1 - \lambda e^{j\omega_0} z^{-1})(1 - \lambda e^{-j\omega_0} z^{-1})}$$

Hum removal, $\lambda = 0.95$



Hum removal, filter structure



COM303: Digital Signal Processing

Lecture 12: Filter design

Overview

- ▶ filter design: problem statement
- ▶ IIR design
- ▶ two more ideal filters

filter design

The filter design problem

You are given a set of requirements:

- ▶ frequency response: passband(s) and stopband(s)
- ▶ phase: overall delay, linearity
- ▶ some limit on computational resources and/or numerical precision

You must determine N , M , a_k 's and b_k 's in

$$H(z) = \frac{b_0 + b_1 z^{-1} + \dots + b_{M-1} z^{-M}}{1 + a_1 z^{-1} + \dots + a_{N-1} z^{-N}}$$

in order to best fulfill the requirements

The filter design problem

You are given a set of requirements:

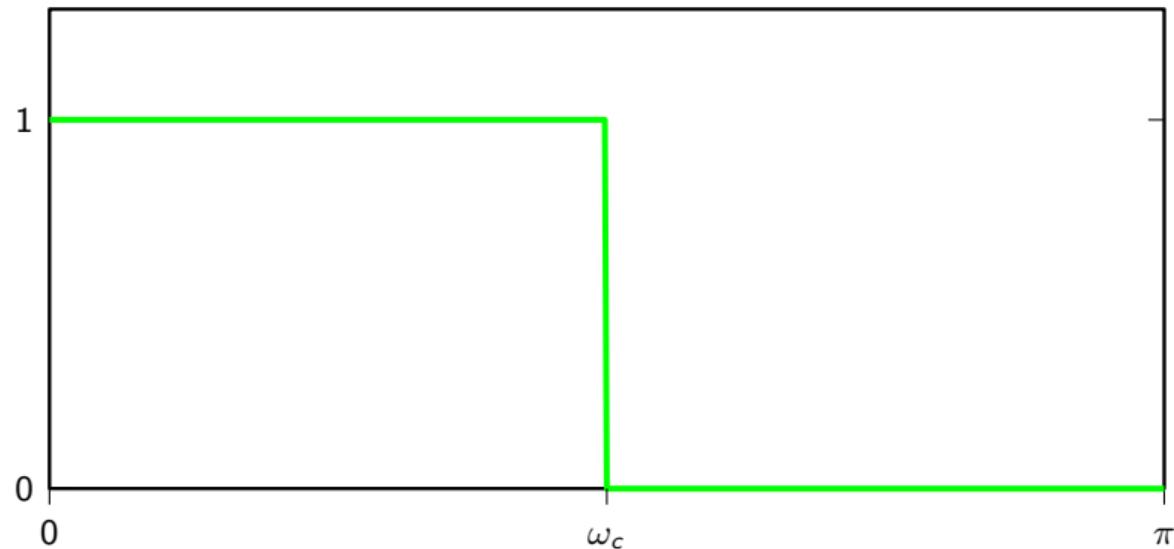
- ▶ frequency response: passband(s) and stopband(s)
- ▶ phase: overall delay, linearity
- ▶ some limit on computational resources and/or numerical precision

You must determine N , M , a_k 's and b_k 's in

$$H(z) = \frac{b_0 + b_1 z^{-1} + \dots + b_{M-1} z^{-M}}{1 + a_1 z^{-1} + \dots + a_{N-1} z^{-N}}$$

in order to best fulfill the requirements

Example: lowpass specs



Practical limitations

- ▶ passband/stopband transitions cannot be infinitely sharp
⇒ use *transition bands*
- ▶ magnitude response cannot be constant over an interval
⇒ *specify magnitude tolerances over bands*
- ▶ in general:
 - smaller transition bands ⇒ higher filter order
 - smaller error tolerances ⇒ higher filter order
 - higher filter order ⇒ more expensive, larger delay

Practical limitations

- ▶ passband/stopband transitions cannot be infinitely sharp
⇒ use *transition bands*
- ▶ magnitude response cannot be constant over an interval
⇒ *specify magnitude tolerances over bands*
- ▶ in general:
 - smaller transition bands ⇒ higher filter order
 - smaller error tolerances ⇒ higher filter order
 - higher filter order ⇒ more expensive, larger delay

Practical limitations

- ▶ passband/stopband transitions cannot be infinitely sharp
⇒ use *transition bands*
- ▶ magnitude response cannot be constant over an interval
⇒ specify magnitude *tolerances* over bands
- ▶ in general:
 - smaller transition bands ⇒ higher filter order
 - smaller error tolerances ⇒ higher filter order
 - higher filter order ⇒ more expensive, larger delay

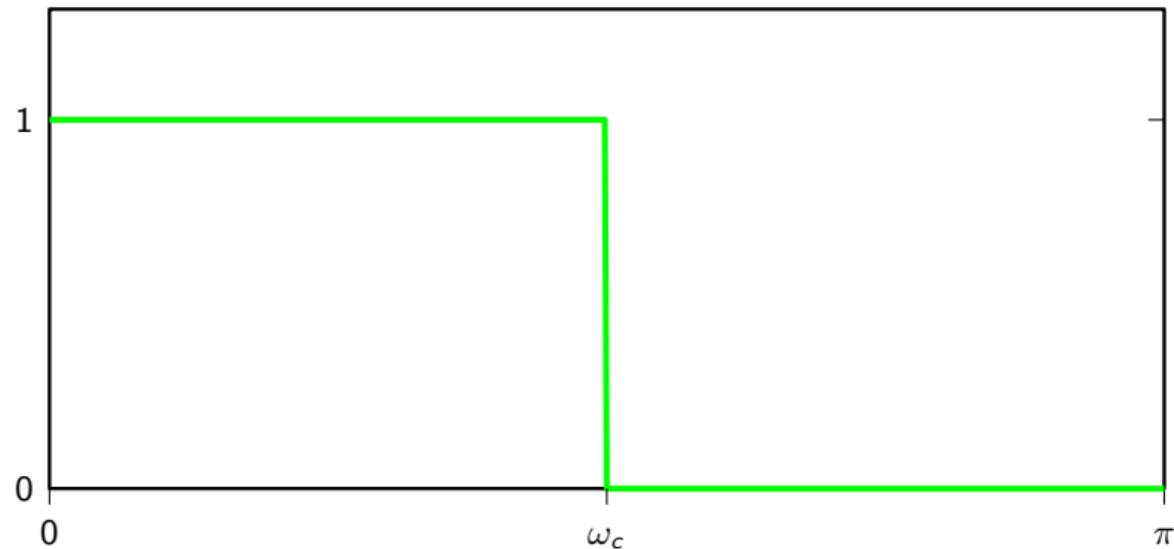
Practical limitations

- ▶ passband/stopband transitions cannot be infinitely sharp
⇒ use *transition bands*
- ▶ magnitude response cannot be constant over an interval
⇒ specify magnitude *tolerances* over bands
- ▶ in general:
 - smaller transition bands ⇒ higher filter order
 - smaller error tolerances ⇒ higher filter order
 - higher filter order ⇒ more expensive, larger delay

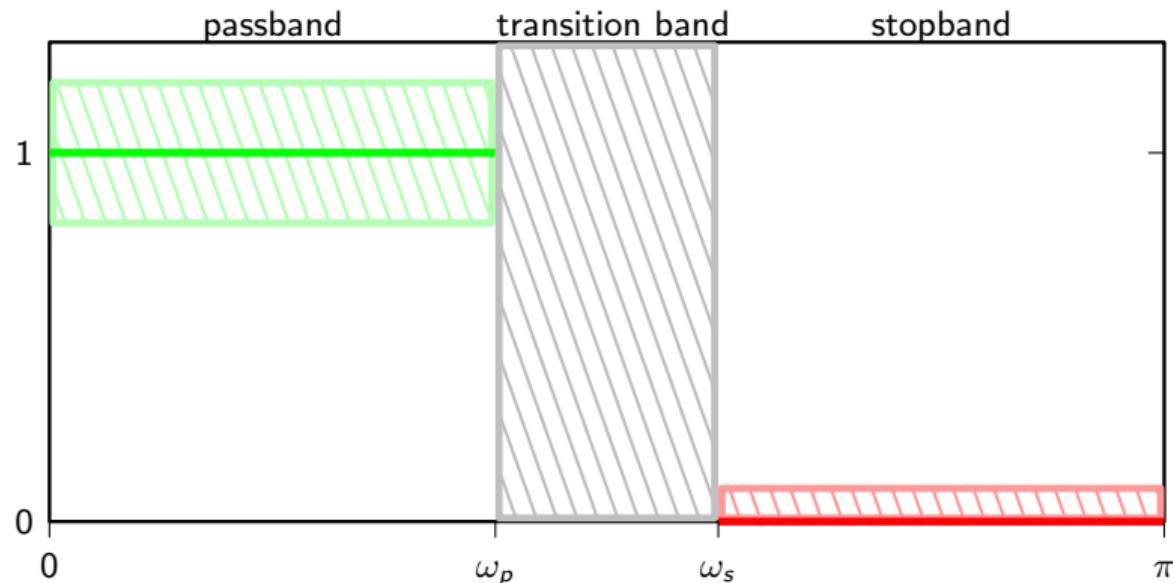
Practical limitations

- ▶ passband/stopband transitions cannot be infinitely sharp
⇒ use *transition bands*
- ▶ magnitude response cannot be constant over an interval
⇒ specify magnitude *tolerances* over bands
- ▶ in general:
 - smaller transition bands ⇒ higher filter order
 - smaller error tolerances ⇒ higher filter order
 - higher filter order ⇒ more expensive, larger delay

Example: lowpass specs



Realistic specs



Why we can't have a “vertical” transition

$H(z) = \frac{B(z)}{A(z)}$ is a rational function with $A, B \in C^\infty$

polynomial rational functions cannot have jump discontinuities

Why we can't have a flat response

$$H(z) = \frac{B(z)}{A(z)}, \quad \text{with } A \text{ and } B \text{ polynomials}$$

Why we can't have a flat response

$$H(z) = \frac{B(z)}{A(z)}, \quad \text{with } A \text{ and } B \text{ polynomials}$$

- $H(e^{j\omega}) = c$ over an interval $\Rightarrow B(z) - cA(z) = 0$ over an interval
 $\Rightarrow B(z) - cA(z)$ has an infinite number of roots
 $\Rightarrow B(z) - cA(z) = 0$ for all values of z
 $\Rightarrow H(e^{j\omega}) = c$ over the entire $[-\pi, \pi]$ interval.

Why we can't have a flat response

$$H(z) = \frac{B(z)}{A(z)}, \quad \text{with } A \text{ and } B \text{ polynomials}$$

$$\begin{aligned}
 H(e^{j\omega}) = c \text{ over an interval} &\Rightarrow B(z) - cA(z) = 0 \text{ over an interval} \\
 &\Rightarrow B(z) - cA(z) \text{ has an infinite number of roots} \\
 &\Rightarrow B(z) - cA(z) = 0 \text{ for all values of } z \\
 &\Rightarrow H(e^{j\omega}) = c \text{ over the entire } [-\pi, \pi] \text{ interval.}
 \end{aligned}$$

Why we can't have a flat response

$$H(z) = \frac{B(z)}{A(z)}, \quad \text{with } A \text{ and } B \text{ polynomials}$$

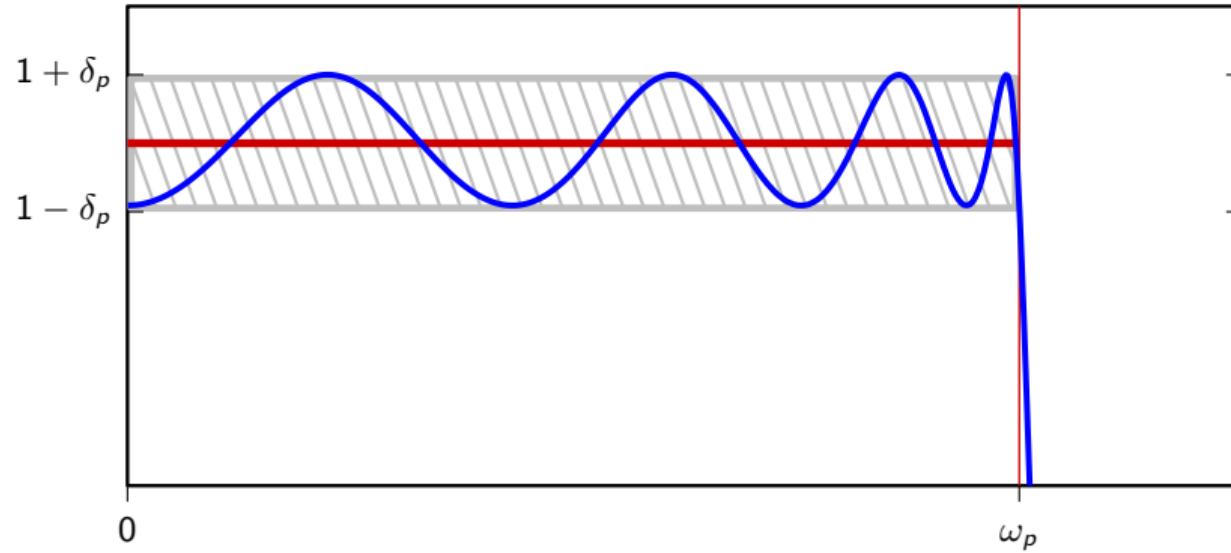
$$\begin{aligned}
 H(e^{j\omega}) = c \text{ over an interval} &\Rightarrow B(z) - cA(z) = 0 \text{ over an interval} \\
 &\Rightarrow B(z) - cA(z) \text{ has an infinite number of roots} \\
 &\Rightarrow B(z) - cA(z) = 0 \text{ for all values of } z \\
 &\Rightarrow H(e^{j\omega}) = c \text{ over the entire } [-\pi, \pi] \text{ interval.}
 \end{aligned}$$

Why we can't have a flat response

$$H(z) = \frac{B(z)}{A(z)}, \quad \text{with } A \text{ and } B \text{ polynomials}$$

- $H(e^{j\omega}) = c$ over an interval \Rightarrow $B(z) - cA(z) = 0$ over an interval
 \Rightarrow $B(z) - cA(z)$ has an infinite number of roots
 \Rightarrow $B(z) - cA(z) = 0$ for all values of z
 \Rightarrow $H(e^{j\omega}) = c$ over the entire $[-\pi, \pi]$ interval.

Important case: equiripple error



The big questions

- ▶ IIR or FIR?
- ▶ how to determine the coefficients?
- ▶ how to evaluate the performance?

IIRs: pros and cons

Pros:

- ▶ computationally efficient
- ▶ strong attenuation easy
- ▶ “natural sounding” in audio applications

Cons:

- ▶ stability and numerical precision issues
- ▶ limit cycles
- ▶ difficult to design for arbitrary response
- ▶ nonlinear phase

FIRs: pros and cons

Pros:

- ▶ can be designed with linear phase
- ▶ always stable
- ▶ numerically precise
- ▶ optimal design techniques exist
- ▶ efficient in multirate processing

Cons:

- ▶ computationally much more expensive
- ▶ because of length, significant delay (hard to use in live audio)

The design methods

- ▶ finding N , M , a_k 's and b_k 's from specs is a hard nonlinear problem
- ▶ established methods:
 - IIR: conversion of analog design
 - FIR: optimal minimax filter design

The design methods

- ▶ finding N , M , a_k 's and b_k 's from specs is a hard nonlinear problem
- ▶ established methods:
 - IIR: conversion of analog design
 - FIR: optimal minimax filter design

The design methods

- ▶ finding N , M , a_k 's and b_k 's from specs is a hard nonlinear problem
- ▶ established methods:
 - IIR: conversion of analog design
 - FIR: optimal minimax filter design

The design methods

- ▶ finding N , M , a_k 's and b_k 's from specs is a hard nonlinear problem
- ▶ established methods:
 - IIR: conversion of analog design
 - FIR: optimal minimax filter design

IIR design

IIR: conversion of analog design

Filter design was an established art long before digital processing appeared

- ▶ lots of nice analog filters exist
- ▶ methods exist to “translate” the analog design into a rational transfer function
- ▶ most numerical packages (Matlab, Numpy, etc.) provide ready-made routines
- ▶ design involves specifying some parameters and testing that the specs are fulfilled

IIR: conversion of analog design

Filter design was an established art long before digital processing appeared

- ▶ lots of nice analog filters exist
- ▶ methods exist to “translate” the analog design into a rational transfer function
- ▶ most numerical packages (Matlab, Numpy, etc.) provide ready-made routines
- ▶ design involves specifying some parameters and testing that the specs are fulfilled

IIR: conversion of analog design

Filter design was an established art long before digital processing appeared

- ▶ lots of nice analog filters exist
- ▶ methods exist to “translate” the analog design into a rational transfer function
- ▶ most numerical packages (Matlab, Numpy, etc.) provide ready-made routines
- ▶ design involves specifying some parameters and testing that the specs are fulfilled

IIR: conversion of analog design

Filter design was an established art long before digital processing appeared

- ▶ lots of nice analog filters exist
- ▶ methods exist to “translate” the analog design into a rational transfer function
- ▶ most numerical packages (Matlab, Numpy, etc.) provide ready-made routines
- ▶ design involves specifying some parameters and testing that the specs are fulfilled

Butterworth lowpass

Magnitude response:

- ▶ maximally flat
- ▶ monotonic over $[0, \pi]$

Design parameters:

- ▶ order N (N poles and N zeros)
- ▶ cutoff frequency

Design test criterion:

- ▶ width of transition band
- ▶ passband error

Butterworth lowpass

Magnitude response:

- ▶ maximally flat
- ▶ monotonic over $[0, \pi]$

Design parameters:

- ▶ order N (N poles and N zeros)
- ▶ cutoff frequency

Design test criterion:

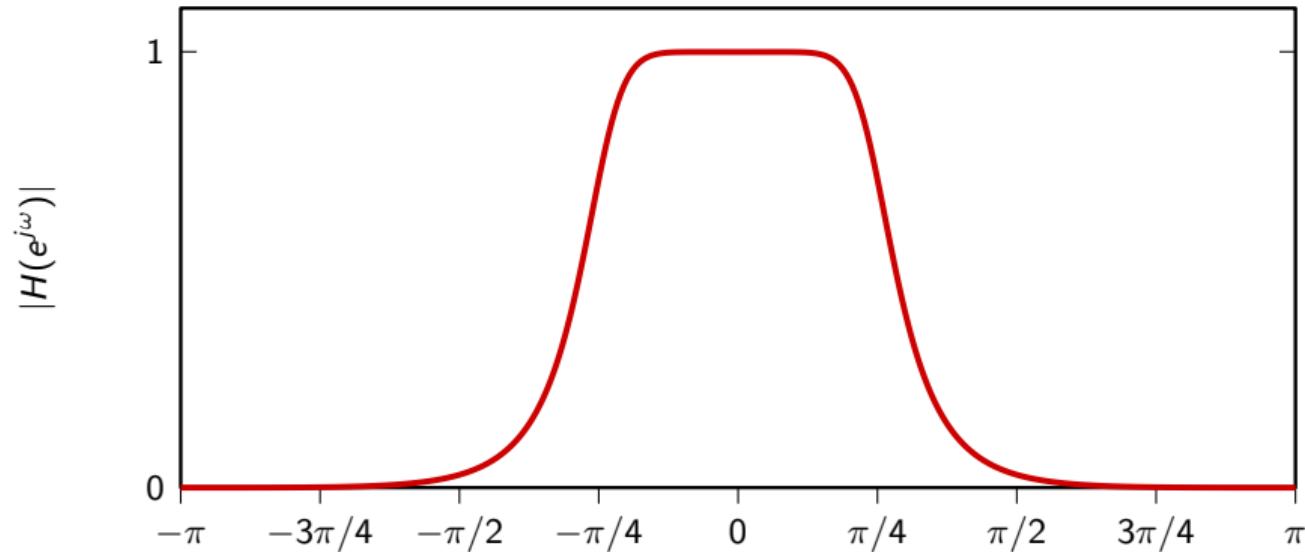
- ▶ width of transition band
- ▶ passband error

Butterworth lowpass in NumPy

```
import scipy.signal as sp  
  
b, a = sp.butter(4, 0.25)  
  
wb, Hb = sp.freqz(b, a, 1024);  
plt.plot(wb/np.pi, np.abs(Hb));
```

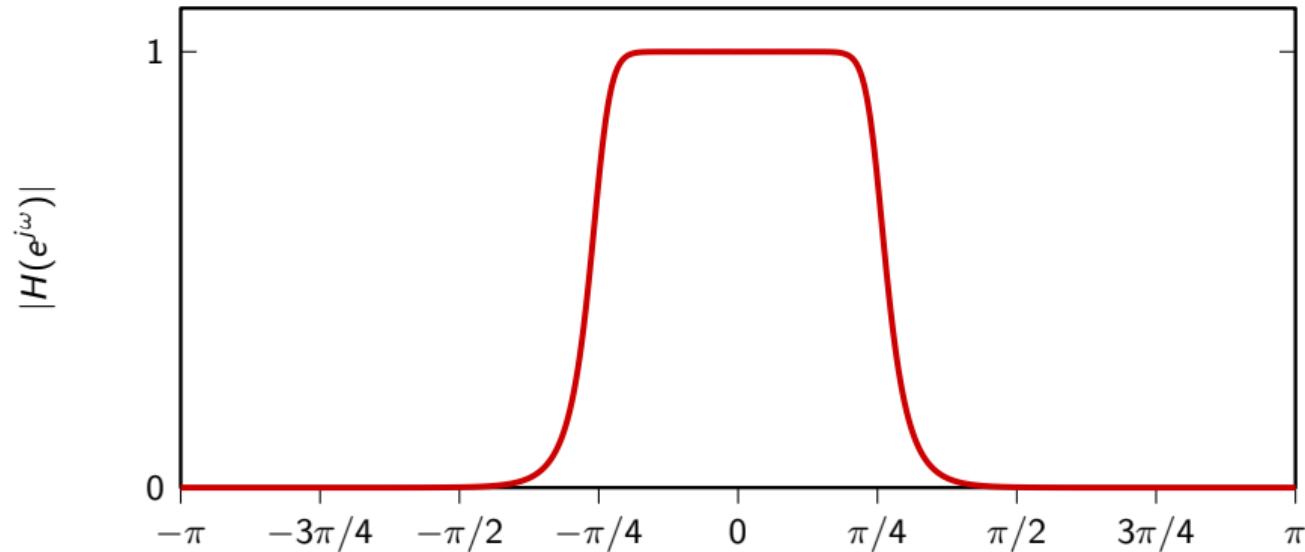
Butterworth lowpass example

$$N = 4, \omega_c = \pi/4$$



Butterworth lowpass example

$$N = 8, \omega_c = \pi/4$$



Chebyshev lowpass

Magnitude response:

- ▶ equiripple in passband
- ▶ monotonic in stopband
- ▶ (or vice-versa)

Design parameters:

- ▶ order N (N poles and N zeros)
- ▶ passband max error
- ▶ cutoff frequency

Design test criterion:

- ▶ width of transition band
- ▶ stopband error

Chebyshev lowpass

Magnitude response:

- ▶ equiripple in passband
- ▶ monotonic in stopband
- ▶ (or vice-versa)

Design parameters:

- ▶ order N (N poles and N zeros)
- ▶ passband max error
- ▶ cutoff frequency

Design test criterion:

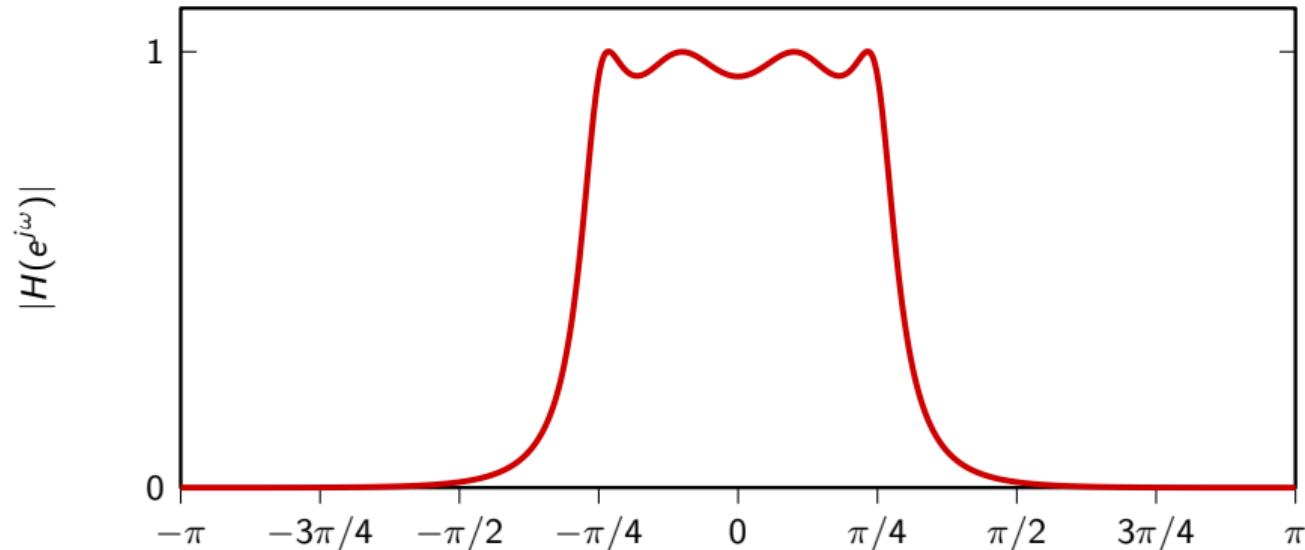
- ▶ width of transition band
- ▶ stopband error

Chebyshev lowpass in NumPy

```
b, a = sp.cheby1(4, .12, 0.25)
```

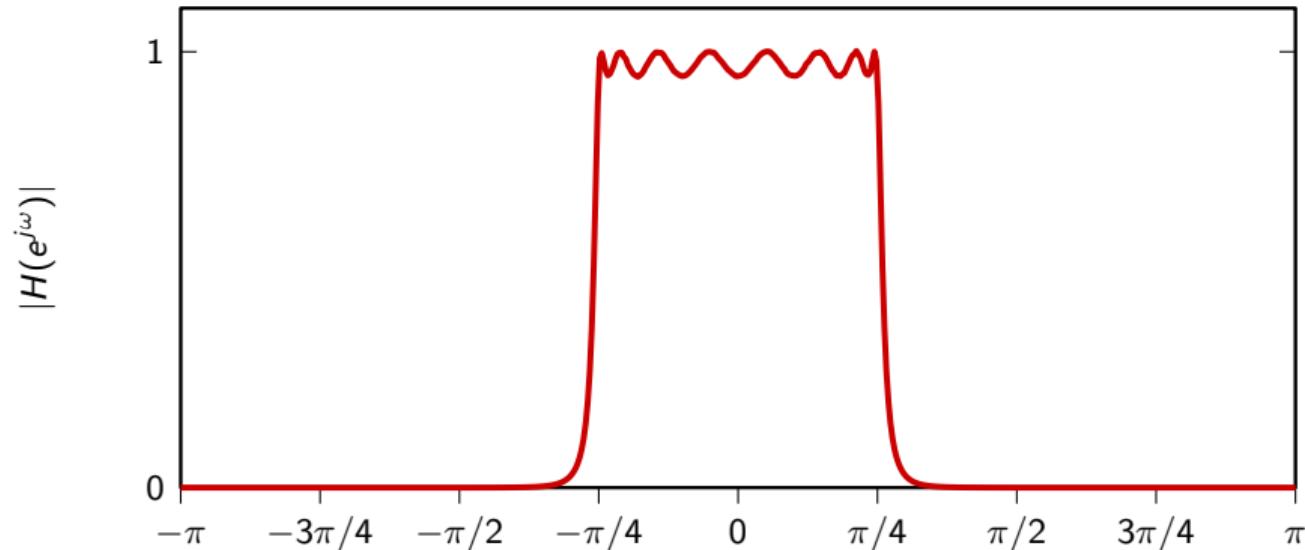
Chebyshev lowpass example

$$N = 4, \omega_c = \pi/4, e_{\max} = 12\%$$



Chebyshev lowpass example

$$N = 8, \omega_c = \pi/4, e_{\max} = 12\%$$



Elliptic lowpass

Magnitude response:

- ▶ equiripple in passband and stopband

Design parameters:

- ▶ order N
- ▶ cutoff frequency
- ▶ passband max error
- ▶ stopband min attenuation

Design test criterion:

- ▶ width of transition band

Elliptic lowpass

Magnitude response:

- ▶ equiripple in passband and stopband

Design parameters:

- ▶ order N
- ▶ cutoff frequency
- ▶ passband max error
- ▶ stopband min attenuation

Design test criterion:

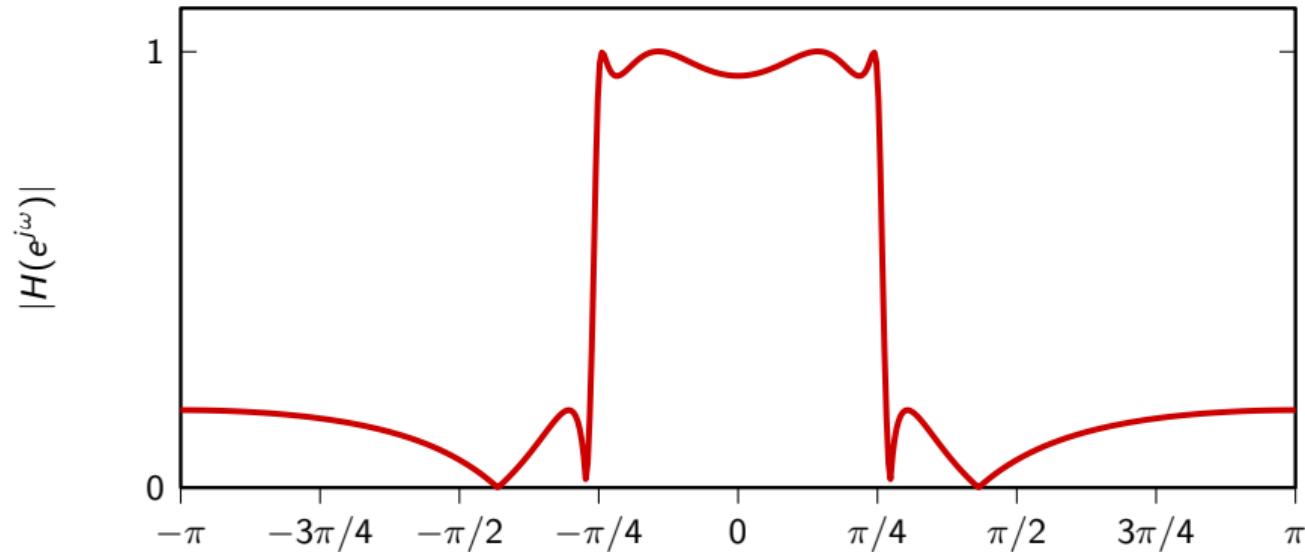
- ▶ width of transition band

Elliptic lowpass in NumPy

```
b, a = sp.ellip(4, .1, 50, 0.25)
```

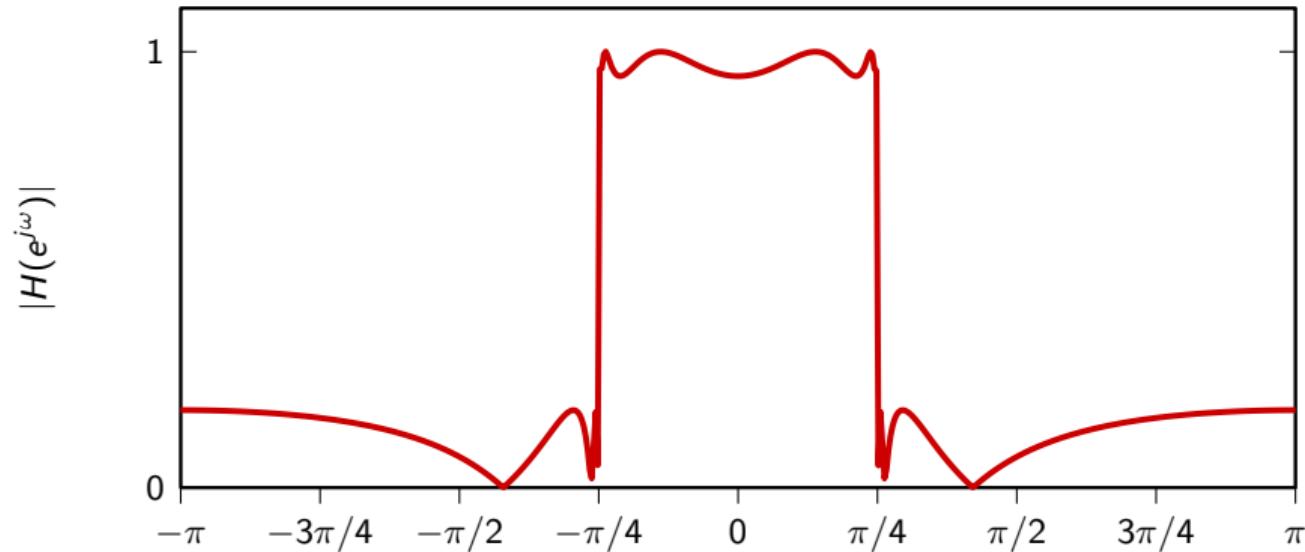
Elliptic lowpass example

$$N = 4, \omega_c = \pi/4, e_{\max} = 12\%, \text{att}_{\min} = 0.03$$



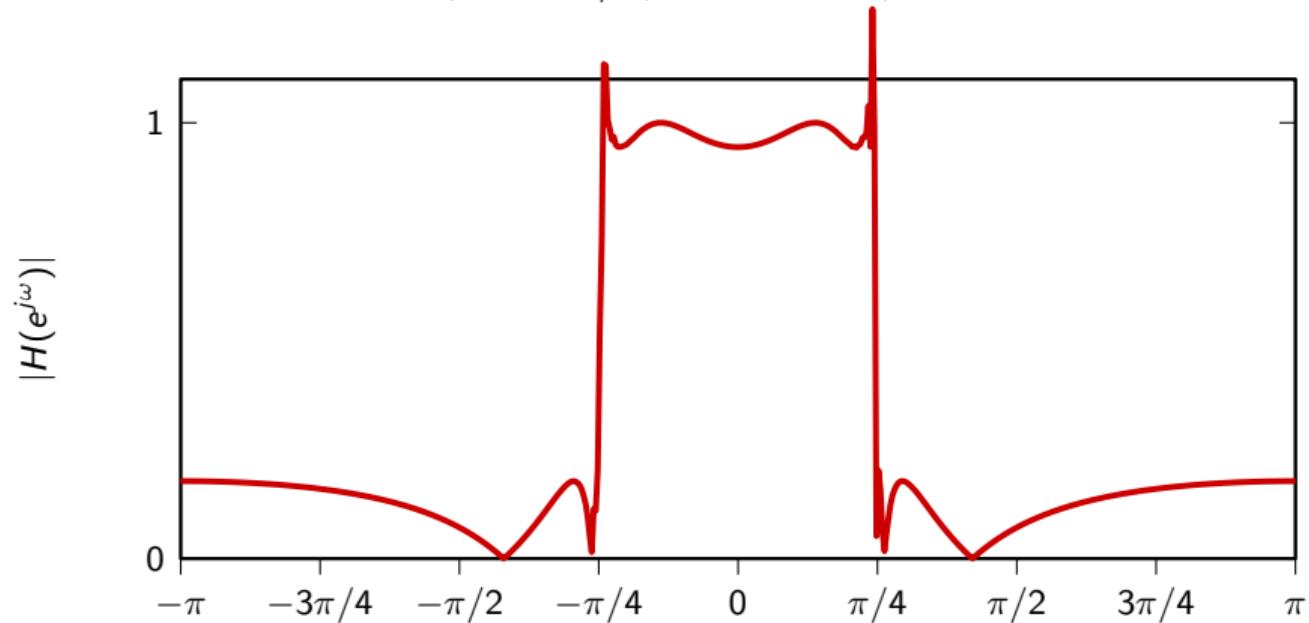
Elliptic lowpass example

$$N = 6, \omega_c = \pi/4, e_{\max} = 12\%, \text{att}_{\min} = 0.03$$



Elliptic lowpass example

$N = 8, \omega_c = \pi/4, e_{\max} = 12\%, \text{att}_{\min} = 0.03$



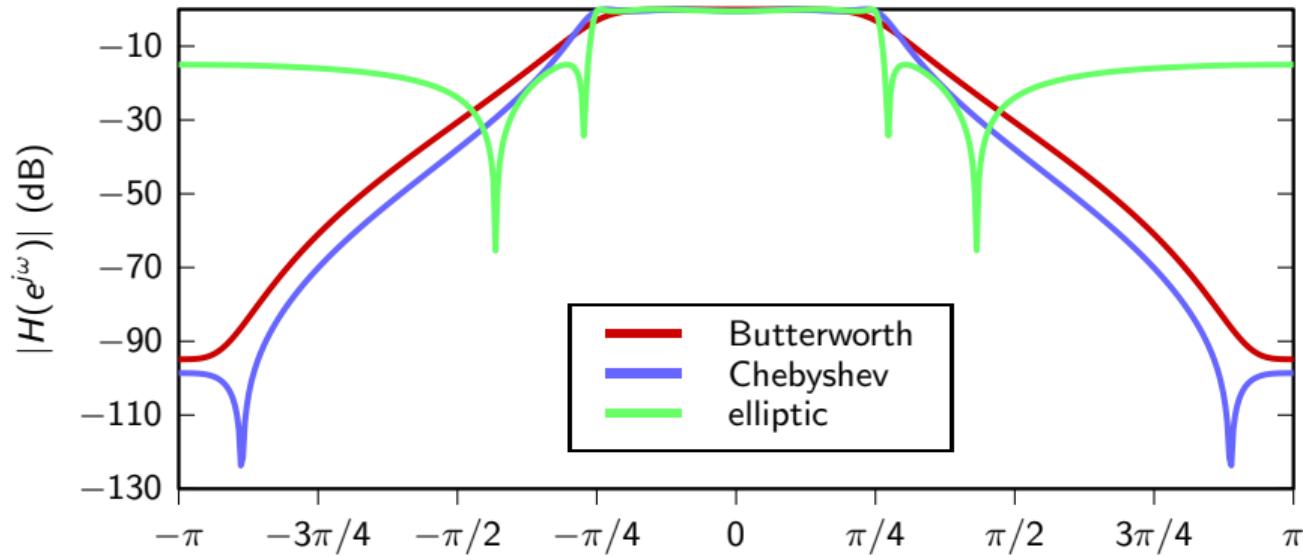
Magnitude response in decibels

- ▶ filter max passband magnitude G
- ▶ filter attenuation expressed in decibels as:

$$A_{\text{dB}} = 20 \log_{10}(|H(e^{j\omega})|/G)$$

- ▶ useful to compare attenuations between filters

4-th order lowpass comparison



Qualitative comparison

For a given order N

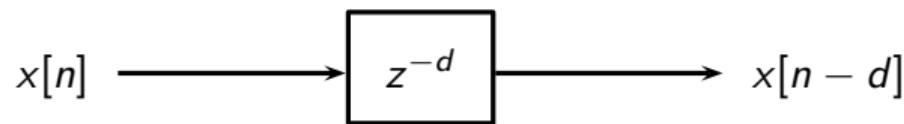
- ▶ sharpness of transition band: Elliptic > Chebyshev > Butterworth
- ▶ phase distortion: Butterworth < Chebyshev < Elliptic
- ▶ pasband ripples Butterworth < Chebyshev < Elliptic
- ▶ stopband attenuation: Elliptic > Chebyshev > Butterworth

a couple more ideal filters

Overview

- ▶ the fractional delay
- ▶ the Hilbert filter

consider a simple delay...

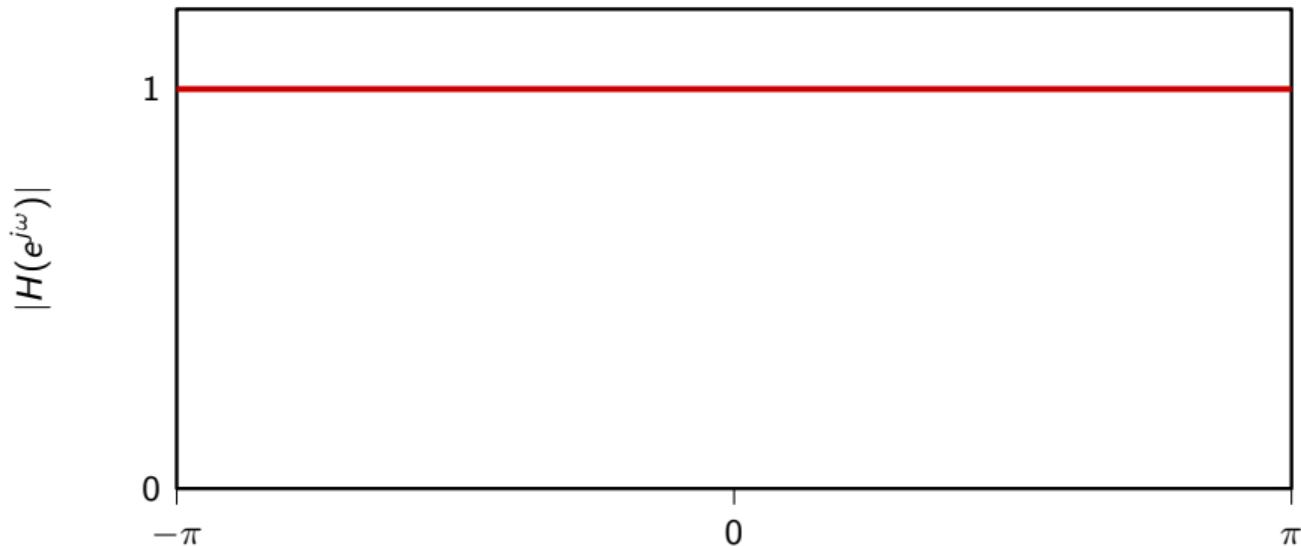


$$H(e^{j\omega}) = e^{-j\omega d} \quad d \in \mathbb{Z}$$

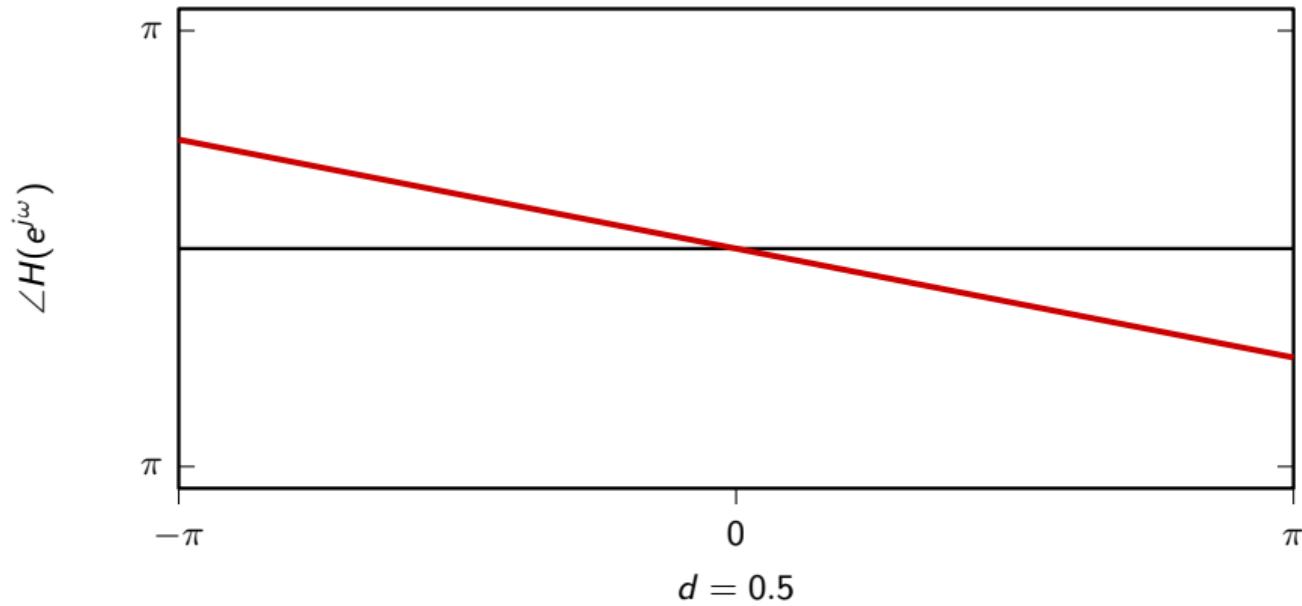
question

what happens if, in $H(e^{j\omega})$ we use a non-integer $d \in \mathbb{R}$?

Fractional delay: magnitude response



Fractional delay: phase response



impulse response

$$\begin{aligned} h[n] &= \text{IDTFT} \left\{ e^{-j\omega d} \right\} \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-j\omega d} e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega(n-d)} d\omega \\ &= \frac{1}{\pi(n-d)} \frac{e^{j\pi(n-d)} - e^{-j\pi(n-d)}}{2j} \\ &= \frac{\sin \pi(n-d)}{\pi(n-d)} \\ &= \text{sinc}(n-d) \end{aligned}$$

impulse response

$$\begin{aligned} h[n] &= \text{IDTFT} \left\{ e^{-j\omega d} \right\} \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-j\omega d} e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega(n-d)} d\omega \\ &= \frac{1}{\pi(n-d)} \frac{e^{j\pi(n-d)} - e^{-j\pi(n-d)}}{2j} \\ &= \frac{\sin \pi(n-d)}{\pi(n-d)} \\ &= \text{sinc}(n-d) \end{aligned}$$

impulse response

$$\begin{aligned} h[n] &= \text{IDTFT} \left\{ e^{-j\omega d} \right\} \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-j\omega d} e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega(n-d)} d\omega \\ &= \frac{1}{\pi(n-d)} \frac{e^{j\pi(n-d)} - e^{-j\pi(n-d)}}{2j} \\ &= \frac{\sin \pi(n-d)}{\pi(n-d)} \\ &= \text{sinc}(n-d) \end{aligned}$$

impulse response

$$\begin{aligned} h[n] &= \text{IDTFT} \left\{ e^{-j\omega d} \right\} \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-j\omega d} e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega(n-d)} d\omega \\ &= \frac{1}{\pi(n-d)} \frac{e^{j\pi(n-d)} - e^{-j\pi(n-d)}}{2j} \\ &= \frac{\sin \pi(n-d)}{\pi(n-d)} \\ &= \text{sinc}(n-d) \end{aligned}$$

impulse response

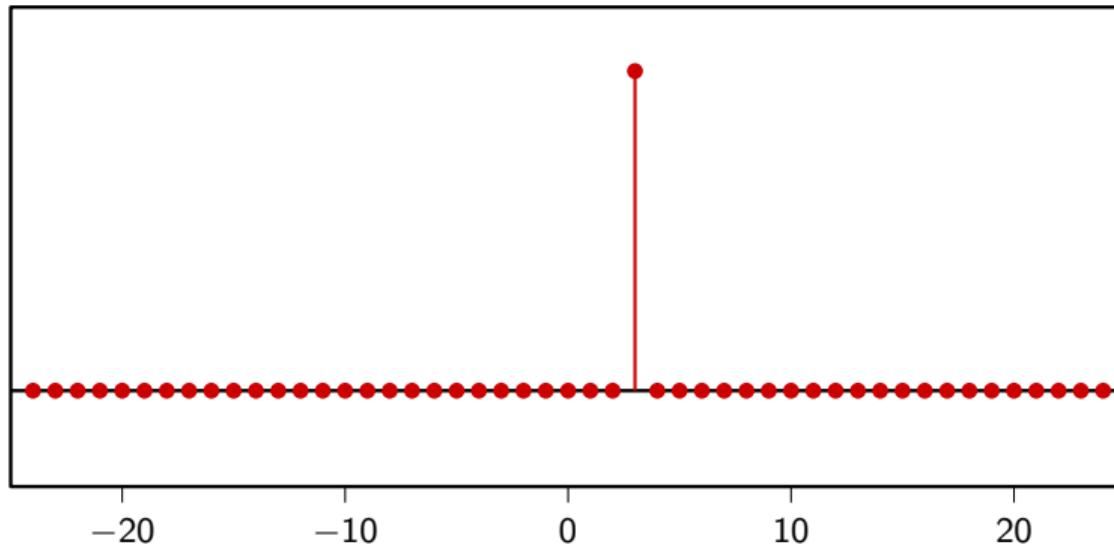
$$\begin{aligned} h[n] &= \text{IDTFT} \left\{ e^{-j\omega d} \right\} \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-j\omega d} e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega(n-d)} d\omega \\ &= \frac{1}{\pi(n-d)} \frac{e^{j\pi(n-d)} - e^{-j\pi(n-d)}}{2j} \\ &= \frac{\sin \pi(n-d)}{\pi(n-d)} \\ &= \text{sinc}(n-d) \end{aligned}$$

impulse response

$$\begin{aligned} h[n] &= \text{IDTFT} \left\{ e^{-j\omega d} \right\} \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-j\omega d} e^{j\omega n} d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega(n-d)} d\omega \\ &= \frac{1}{\pi(n-d)} \frac{e^{j\pi(n-d)} - e^{-j\pi(n-d)}}{2j} \\ &= \frac{\sin \pi(n-d)}{\pi(n-d)} \\ &= \text{sinc}(n-d) \end{aligned}$$

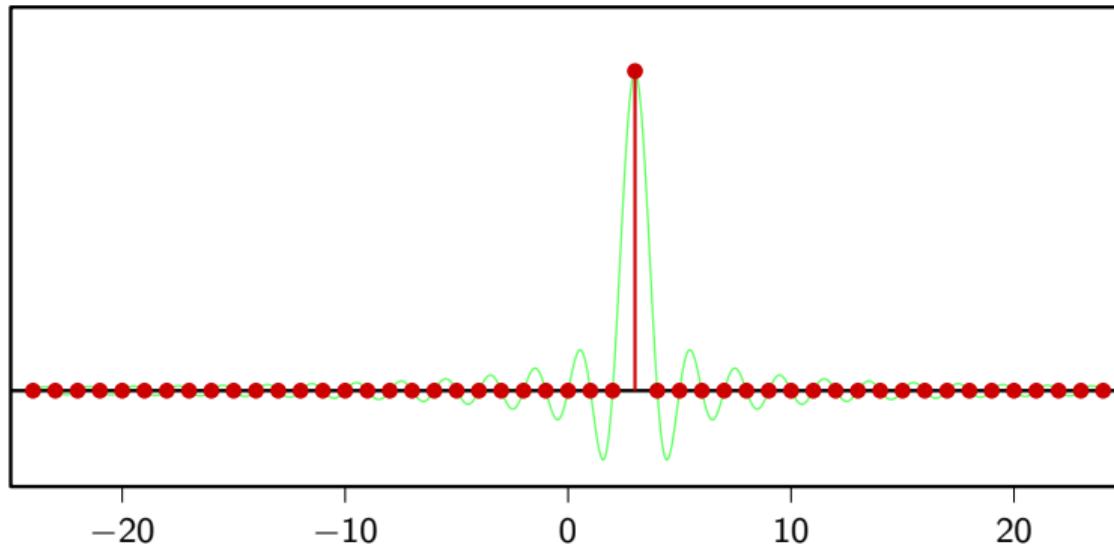
Fractional delay: impulse response

$$d = 3$$



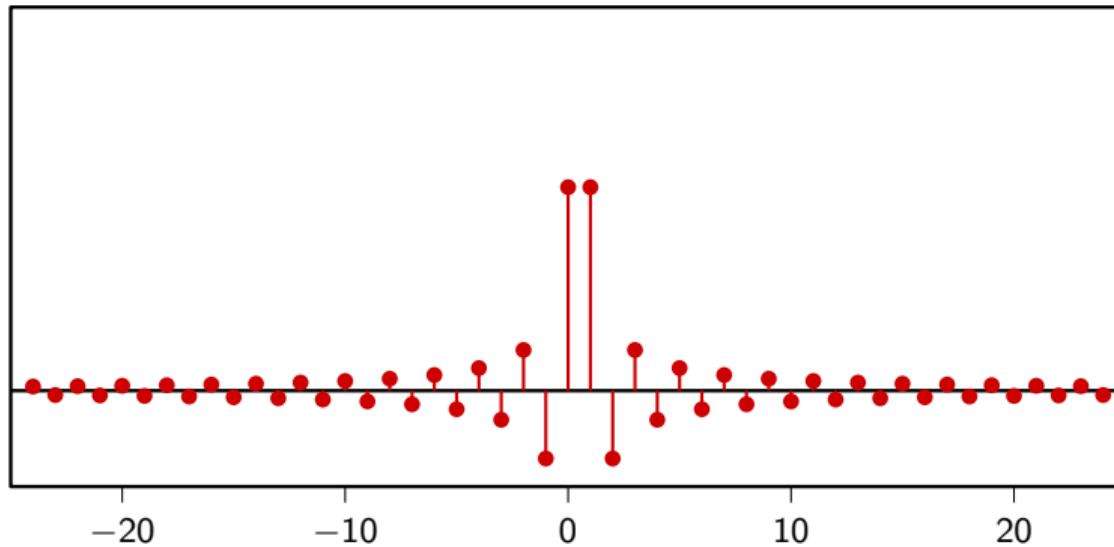
Fractional delay: impulse response

$$d = 3$$



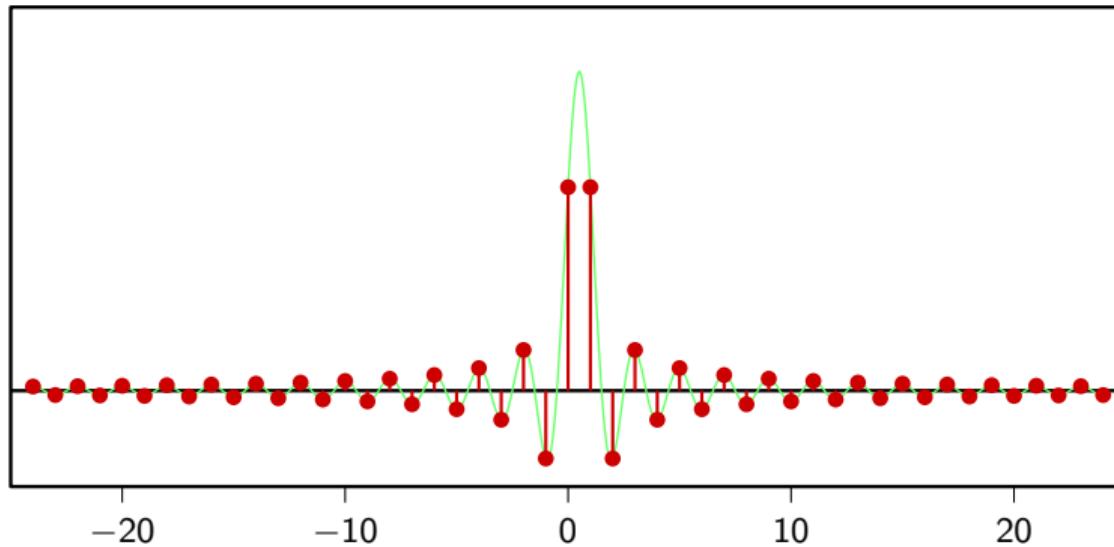
Fractional delay: impulse response

$$d = 0.5$$



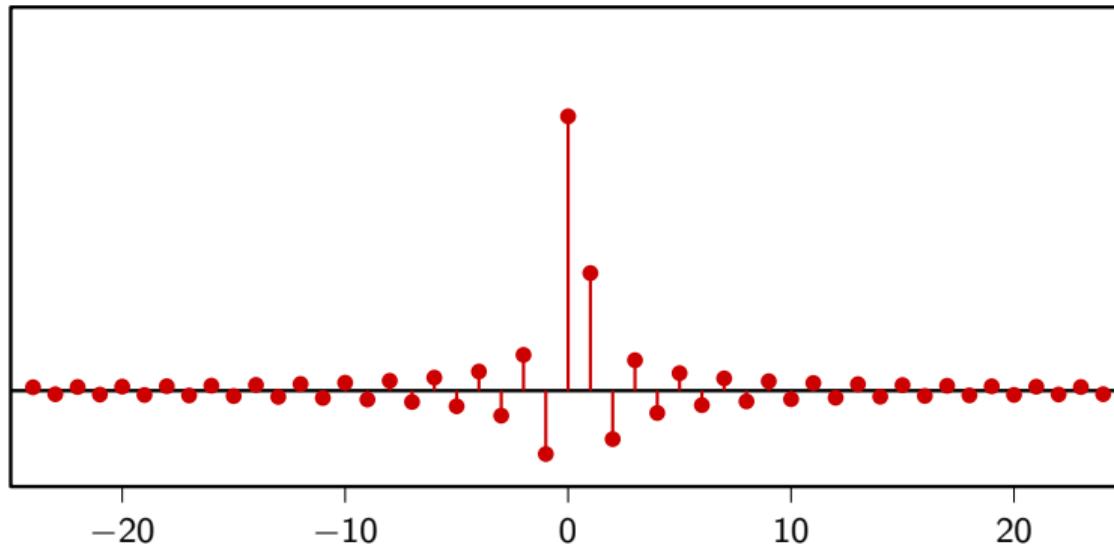
Fractional delay: impulse response

$$d = 0.5$$



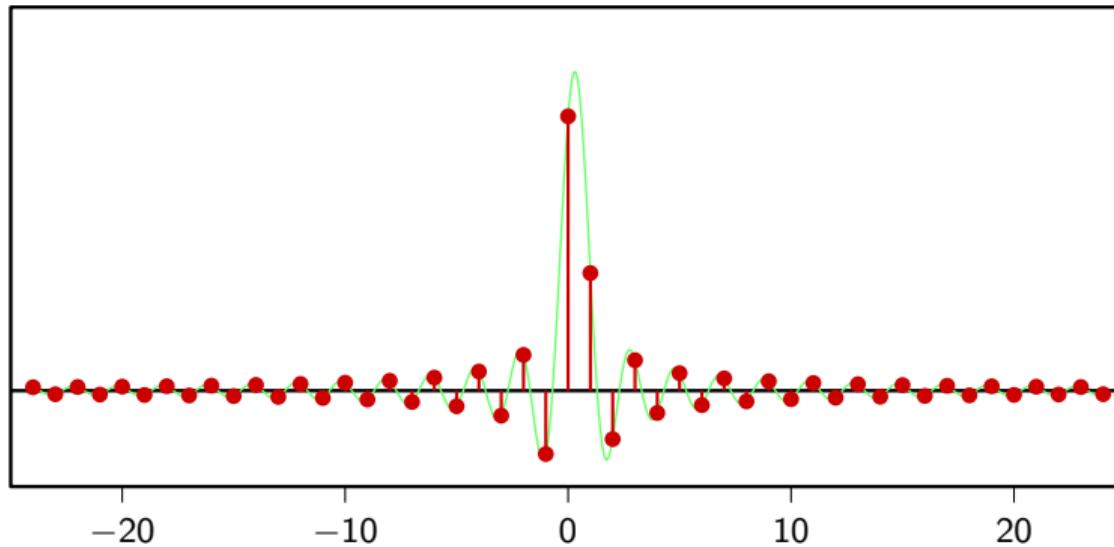
Fractional delay: impulse response

$$d = 0.3$$



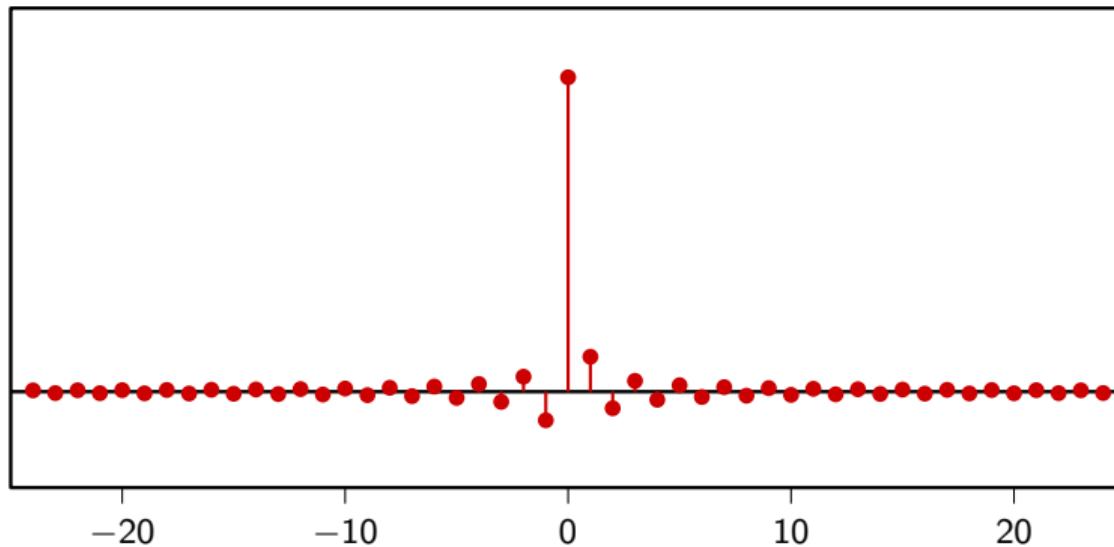
Fractional delay: impulse response

$$d = 0.3$$



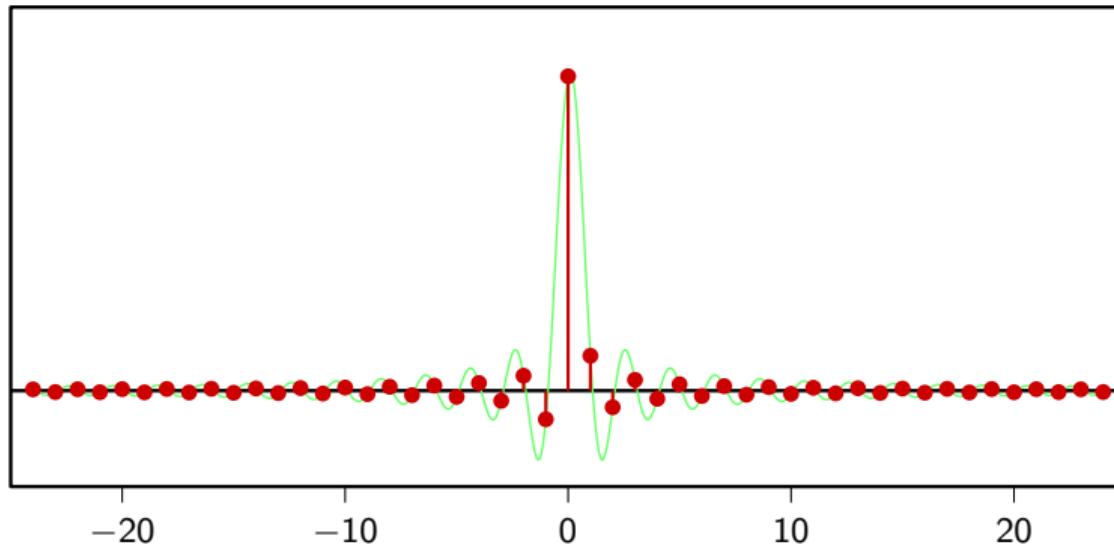
Fractional delay: impulse response

$$d = 0.1$$



Fractional delay: impulse response

$$d = 0.1$$

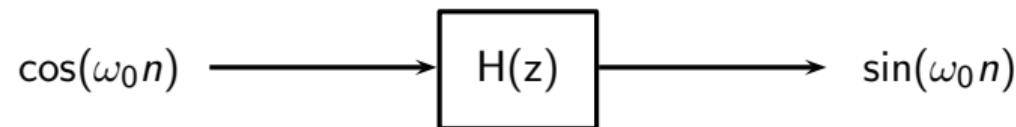


fractional delay

- ▶ fractional delay computes “in-between” values for samples
- ▶ it is an ideal filter!
- ▶ often approximated with local interpolation
- ▶ all will be clear when we study the sampling theorem

the Hilbert filter

a quirky machine



can we build such a thing?

in the frequency domain

$$\text{DTFT } \{2 \cos(\omega_0 n)\} = \tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0)$$

$$\text{DTFT } \{2 \sin(\omega_0 n)\} = -j\tilde{\delta}(\omega - \omega_0) + j\tilde{\delta}(\omega + \omega_0)$$

$$H(e^{j\omega})[\tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0)] = -j\tilde{\delta}(\omega - \omega_0) + j\tilde{\delta}(\omega + \omega_0)$$

$$\begin{cases} H(e^{j\omega_0}) &= -j \\ H(e^{-j\omega_0}) &= +j \end{cases}$$

in the frequency domain

$$\text{DTFT } \{2 \cos(\omega_0 n)\} = \tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0)$$

$$\text{DTFT } \{2 \sin(\omega_0 n)\} = -j\tilde{\delta}(\omega - \omega_0) + j\tilde{\delta}(\omega + \omega_0)$$

$$H(e^{j\omega})[\tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0)] = -j\tilde{\delta}(\omega - \omega_0) + j\tilde{\delta}(\omega + \omega_0)$$

$$\begin{cases} H(e^{j\omega_0}) &= -j \\ H(e^{-j\omega_0}) &= +j \end{cases}$$

in the frequency domain

$$\text{DTFT } \{2 \cos(\omega_0 n)\} = \tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0)$$

$$\text{DTFT } \{2 \sin(\omega_0 n)\} = -j\tilde{\delta}(\omega - \omega_0) + j\tilde{\delta}(\omega + \omega_0)$$

$$H(e^{j\omega})[\tilde{\delta}(\omega - \omega_0) + \tilde{\delta}(\omega + \omega_0)] = -j\tilde{\delta}(\omega - \omega_0) + j\tilde{\delta}(\omega + \omega_0)$$

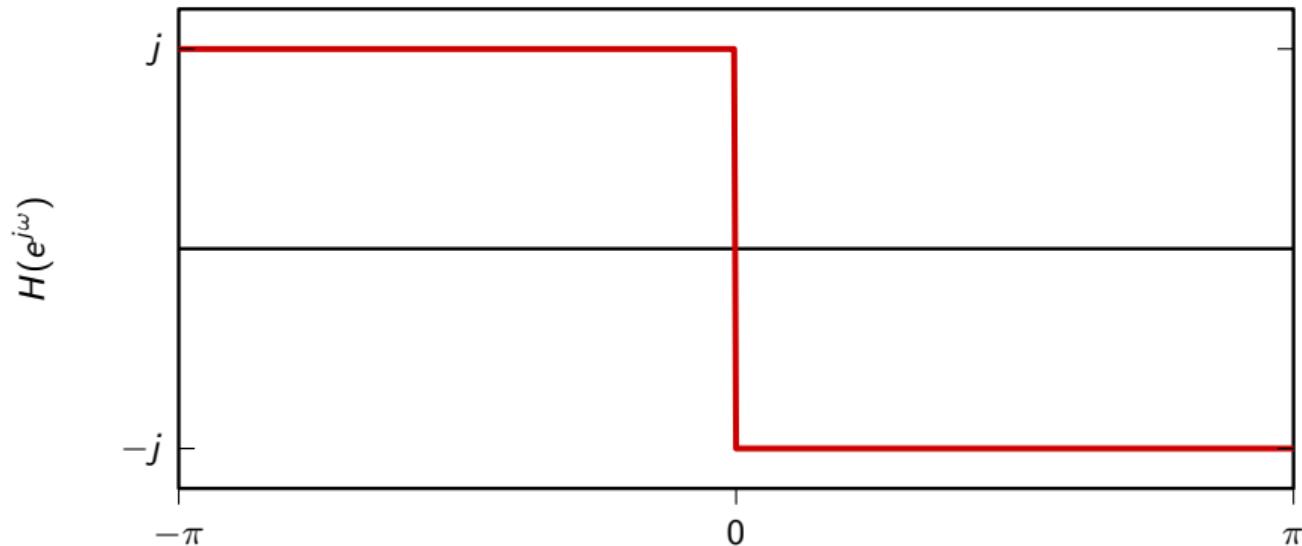
$$\begin{cases} H(e^{j\omega_0}) &= -j \\ H(e^{-j\omega_0}) &= +j \end{cases}$$

in the frequency domain

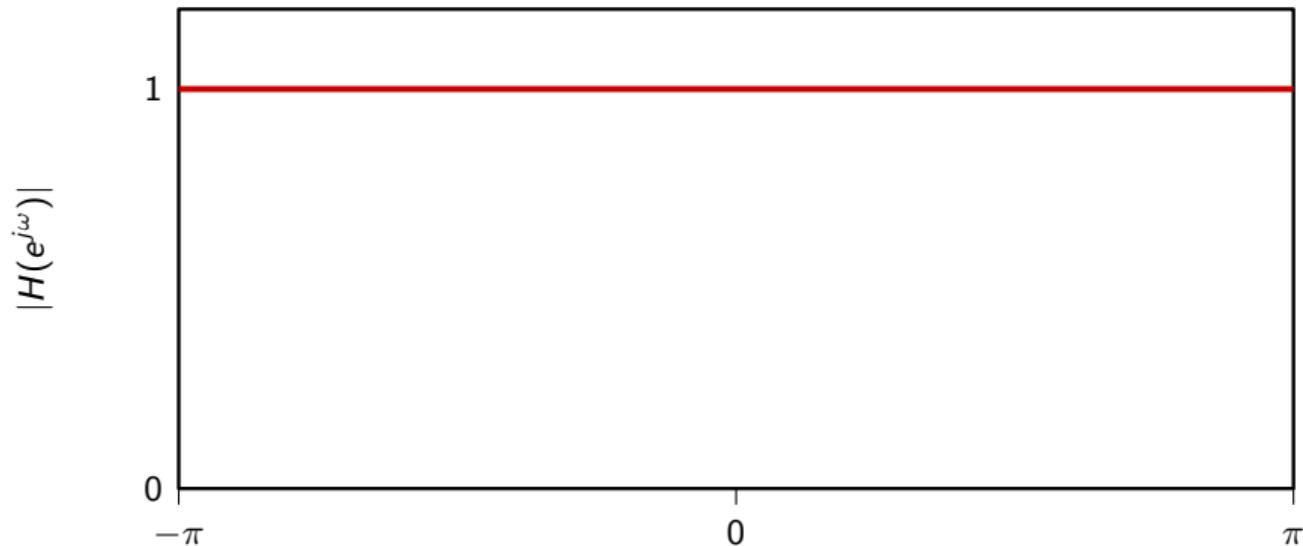
for the machine to work at all frequencies:

$$H(e^{j\omega_0}) = \begin{cases} -j & \text{for } 0 \leq \omega < \pi \\ +j & \text{for } -\pi \leq \omega < 0 \end{cases} \quad (2\pi\text{-periodic})$$

Hilbert filter



Hilbert filter is an allpass



impulse response

$$\begin{aligned} h[n] &= \frac{1}{2\pi} \int_{-\pi}^0 j e^{j\omega n} d\omega + \frac{1}{2\pi} \int_0^\pi -j e^{j\omega n} d\omega \\ &= \frac{1}{2\pi n} [1 - e^{-j\pi n} - (e^{j\pi n} - 1)] \\ &= \frac{1}{\pi n} [1 - \cos(\pi n)] \\ &= \begin{cases} \frac{2}{\pi n} & n \text{ odd} \\ 0 & n \text{ even} \end{cases} \end{aligned}$$

impulse response

$$\begin{aligned} h[n] &= \frac{1}{2\pi} \int_{-\pi}^0 j e^{j\omega n} d\omega + \frac{1}{2\pi} \int_0^\pi -j e^{j\omega n} d\omega \\ &= \frac{1}{2\pi n} [1 - e^{-j\pi n} - (e^{j\pi n} - 1)] \\ &= \frac{1}{\pi n} [1 - \cos(\pi n)] \\ &= \begin{cases} \frac{2}{\pi n} & n \text{ odd} \\ 0 & n \text{ even} \end{cases} \end{aligned}$$

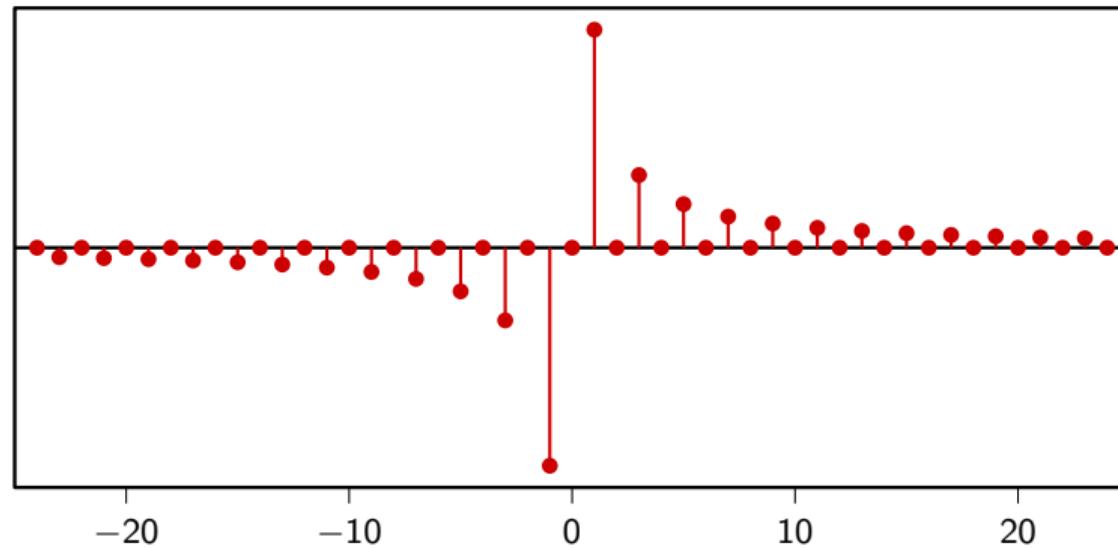
impulse response

$$\begin{aligned} h[n] &= \frac{1}{2\pi} \int_{-\pi}^0 j e^{j\omega n} d\omega + \frac{1}{2\pi} \int_0^\pi -j e^{j\omega n} d\omega \\ &= \frac{1}{2\pi n} [1 - e^{-j\pi n} - (e^{j\pi n} - 1)] \\ &= \frac{1}{\pi n} [1 - \cos(\pi n)] \\ &= \begin{cases} \frac{2}{\pi n} & n \text{ odd} \\ 0 & n \text{ even} \end{cases} \end{aligned}$$

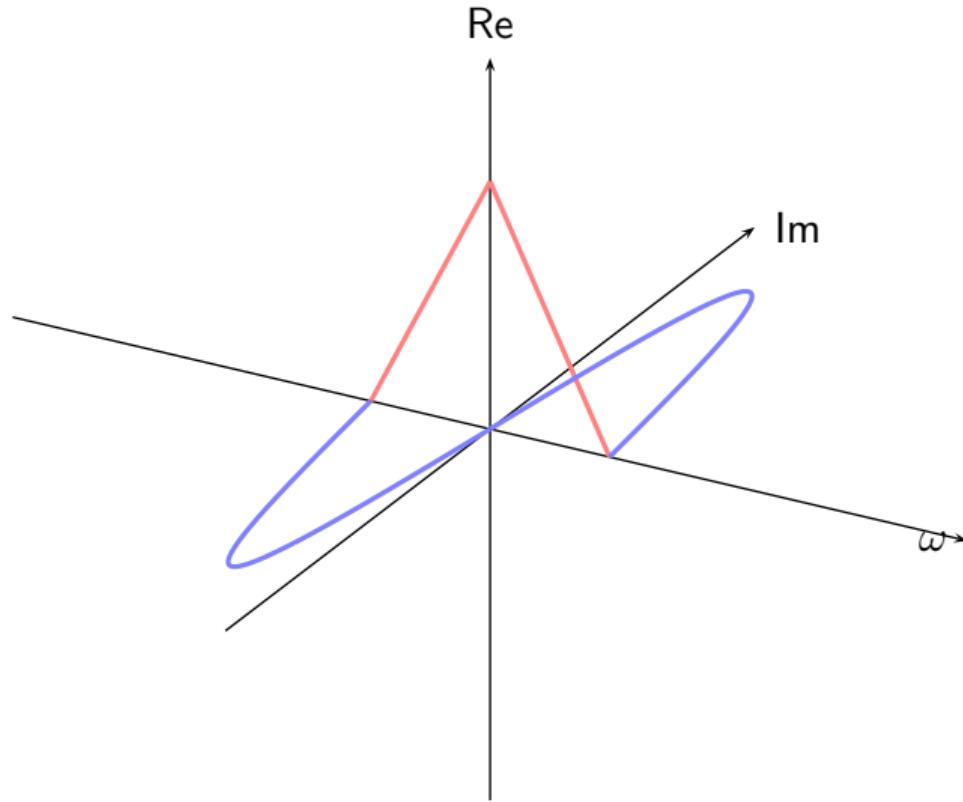
impulse response

$$\begin{aligned} h[n] &= \frac{1}{2\pi} \int_{-\pi}^0 j e^{j\omega n} d\omega + \frac{1}{2\pi} \int_0^\pi -j e^{j\omega n} d\omega \\ &= \frac{1}{2\pi n} [1 - e^{-j\pi n} - (e^{j\pi n} - 1)] \\ &= \frac{1}{\pi n} [1 - \cos(\pi n)] \\ &= \begin{cases} \frac{2}{\pi n} & n \text{ odd} \\ 0 & n \text{ even} \end{cases} \end{aligned}$$

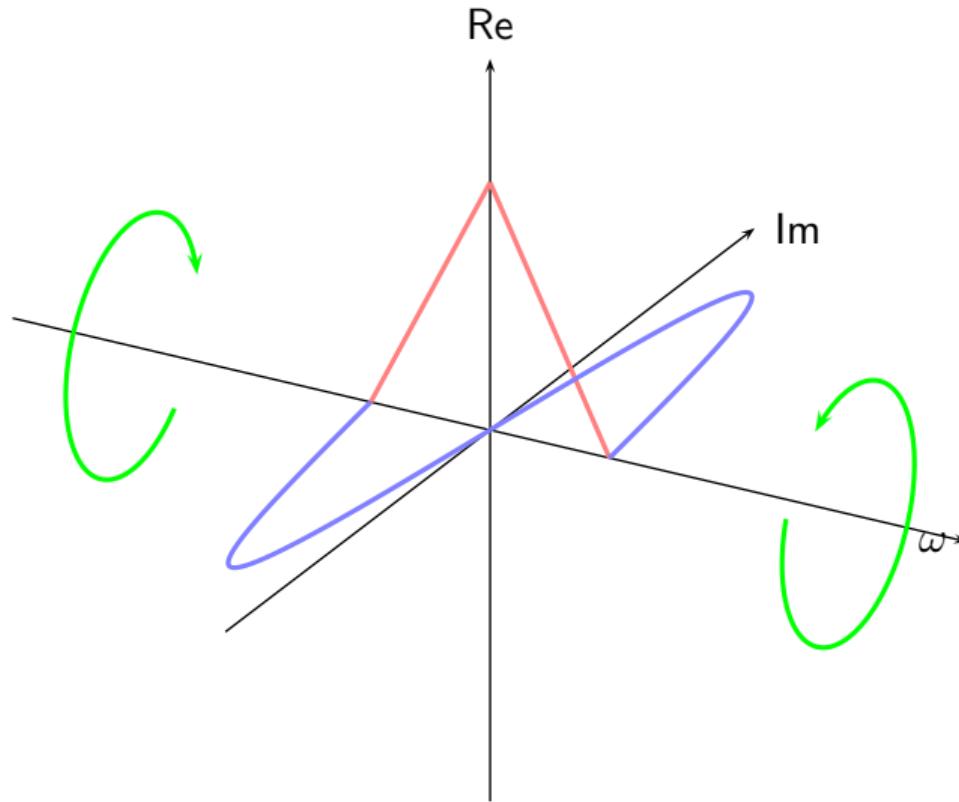
Hilbert filter



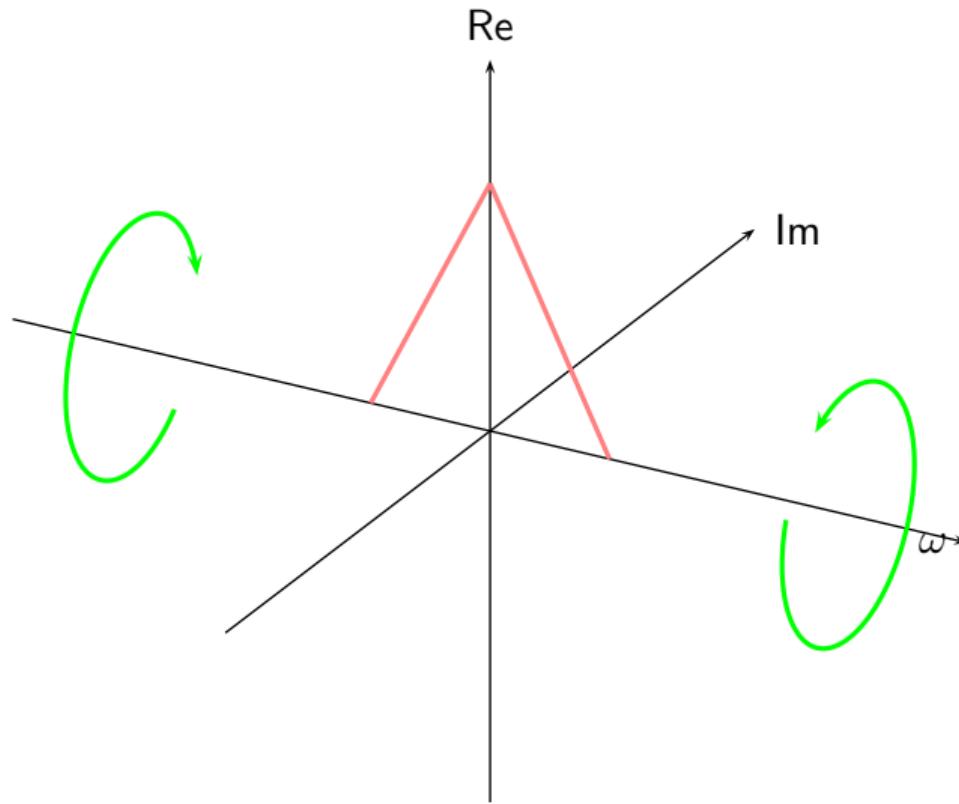
what does the Hilbert filter do?



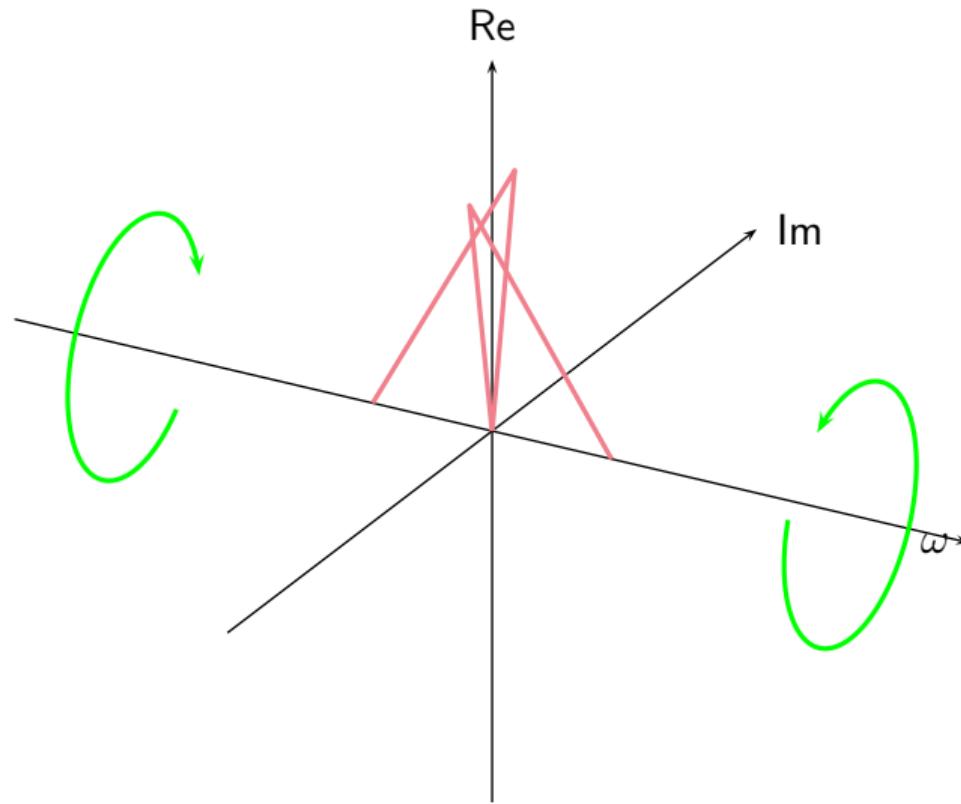
what does the Hilbert filter do?



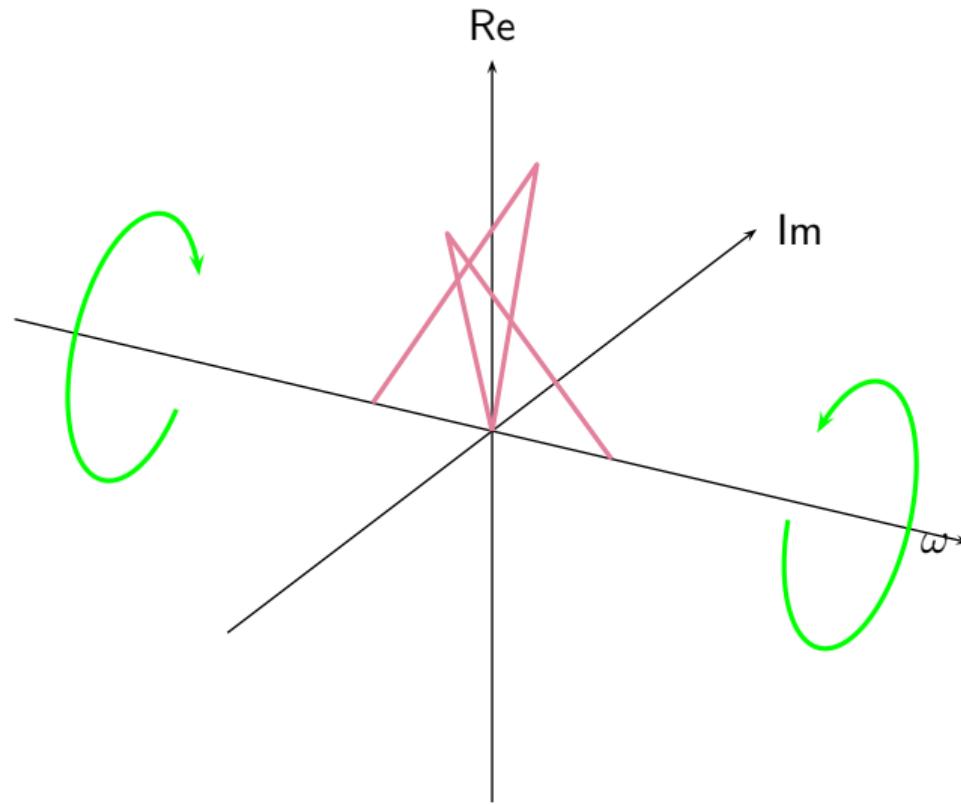
effect of the Hilbert filter (real part)



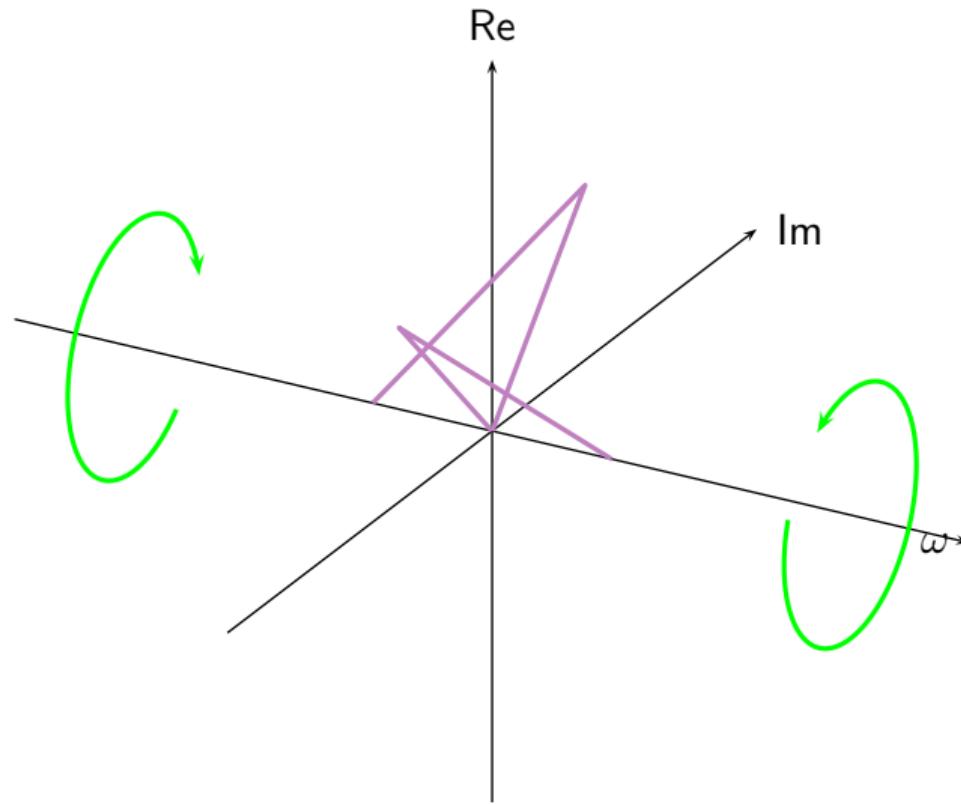
effect of the Hilbert filter (real part)



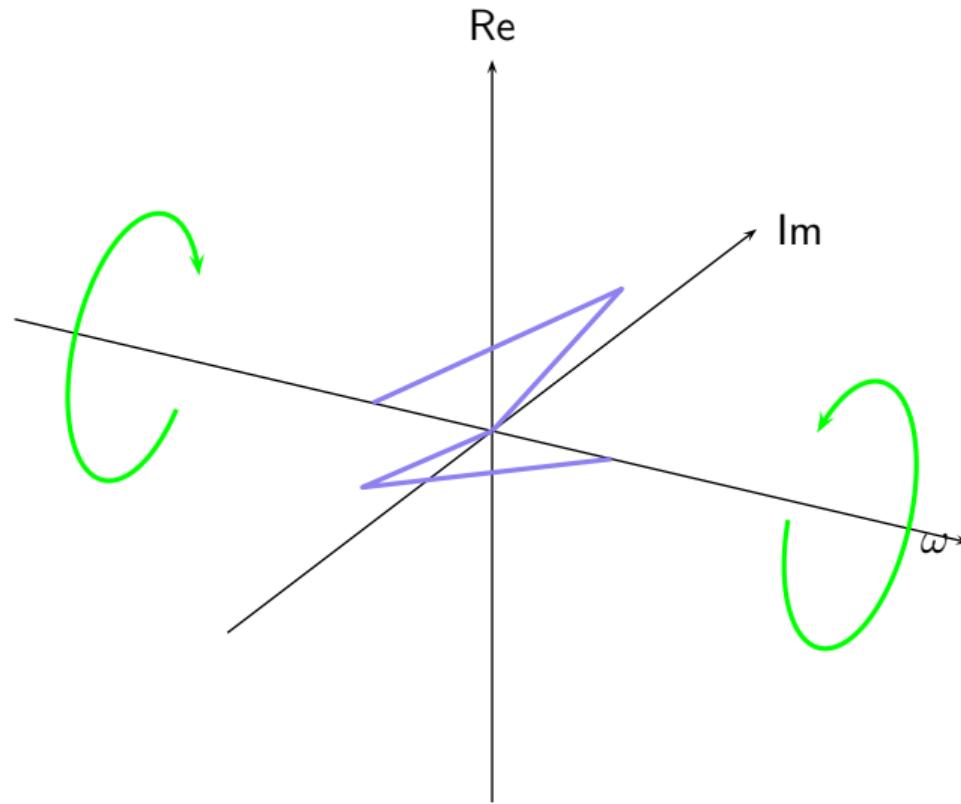
effect of the Hilbert filter (real part)



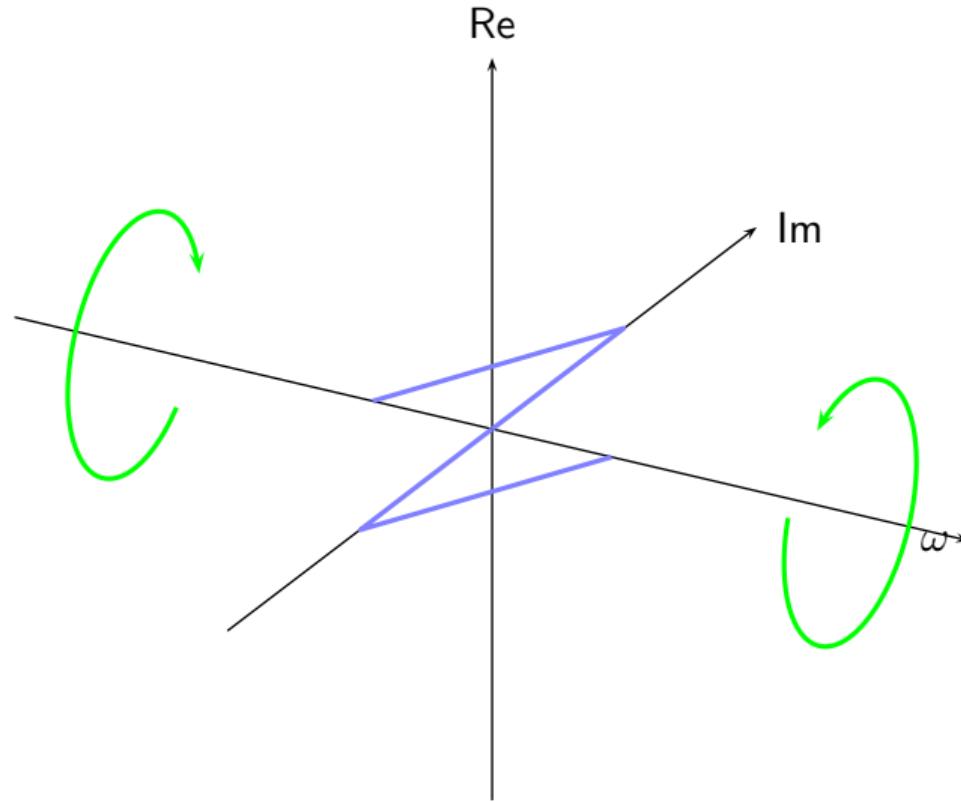
effect of the Hilbert filter (real part)



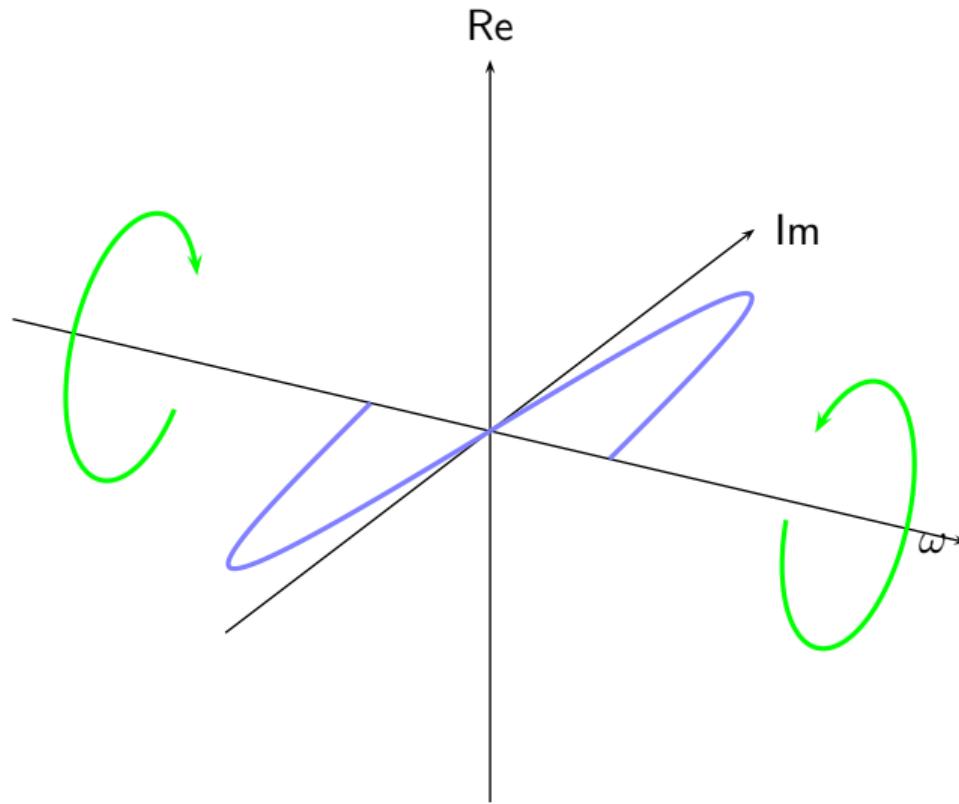
effect of the Hilbert filter (real part)



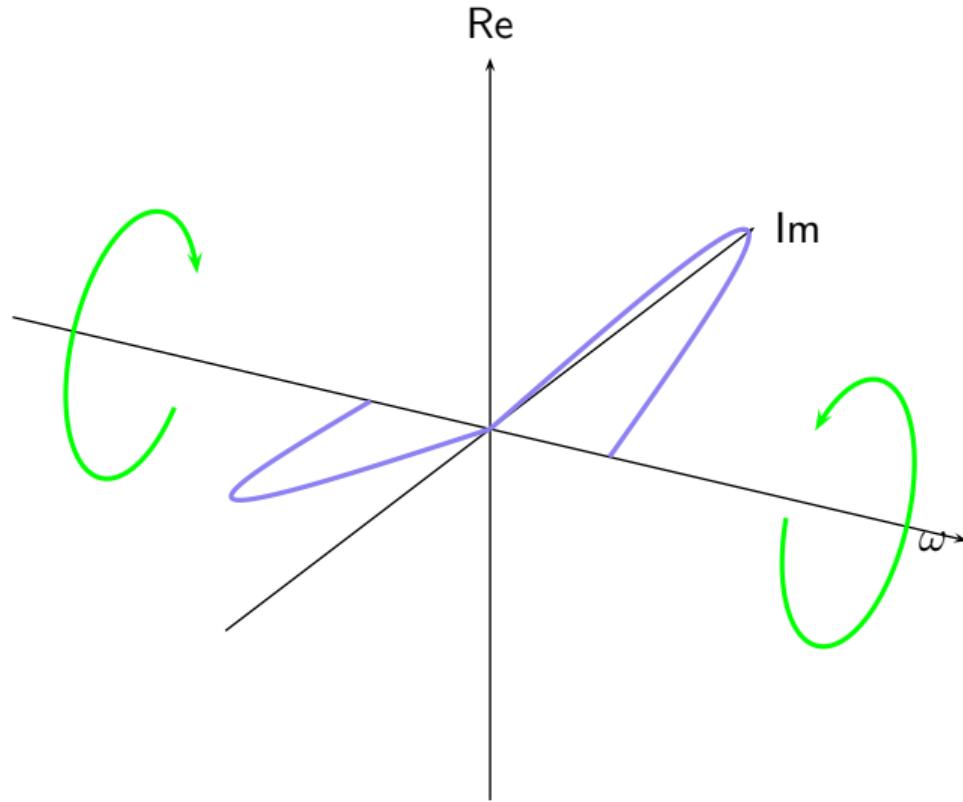
effect of the Hilbert filter (real part)



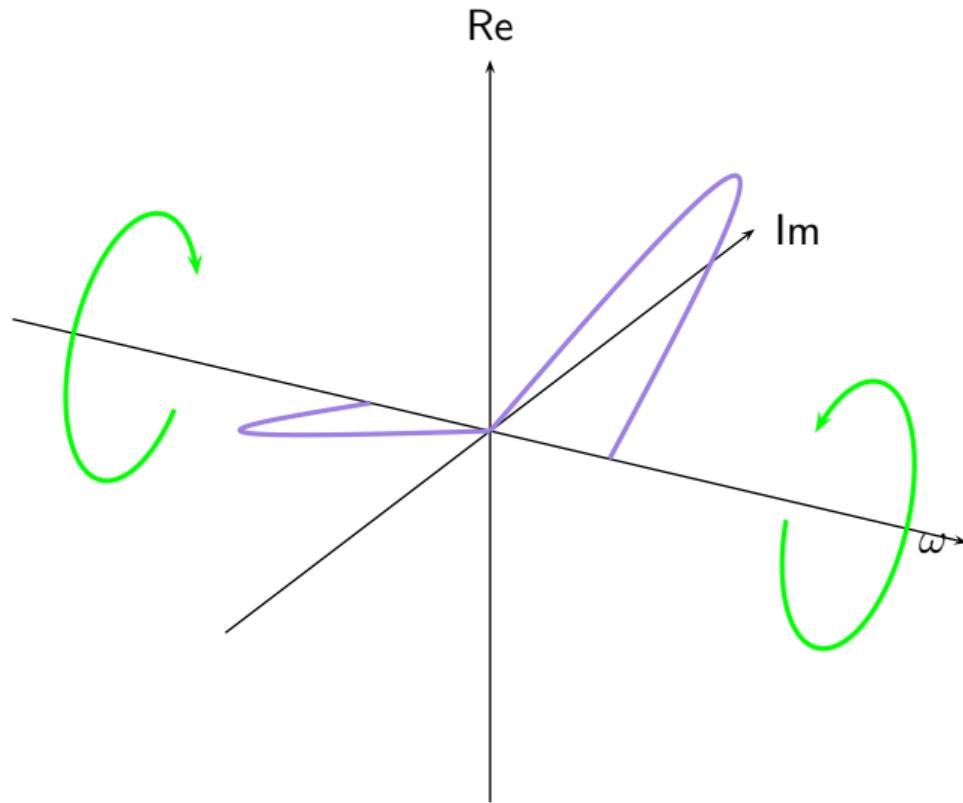
effect of the Hilbert filter (imaginary part)



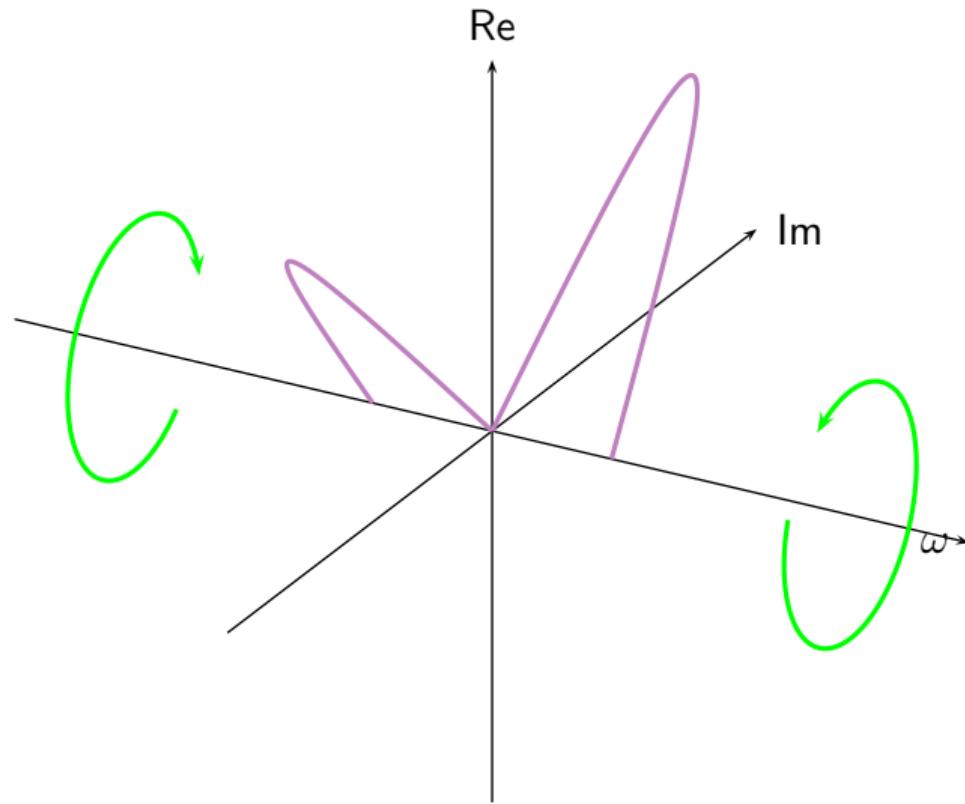
effect of the Hilbert filter (imaginary part)



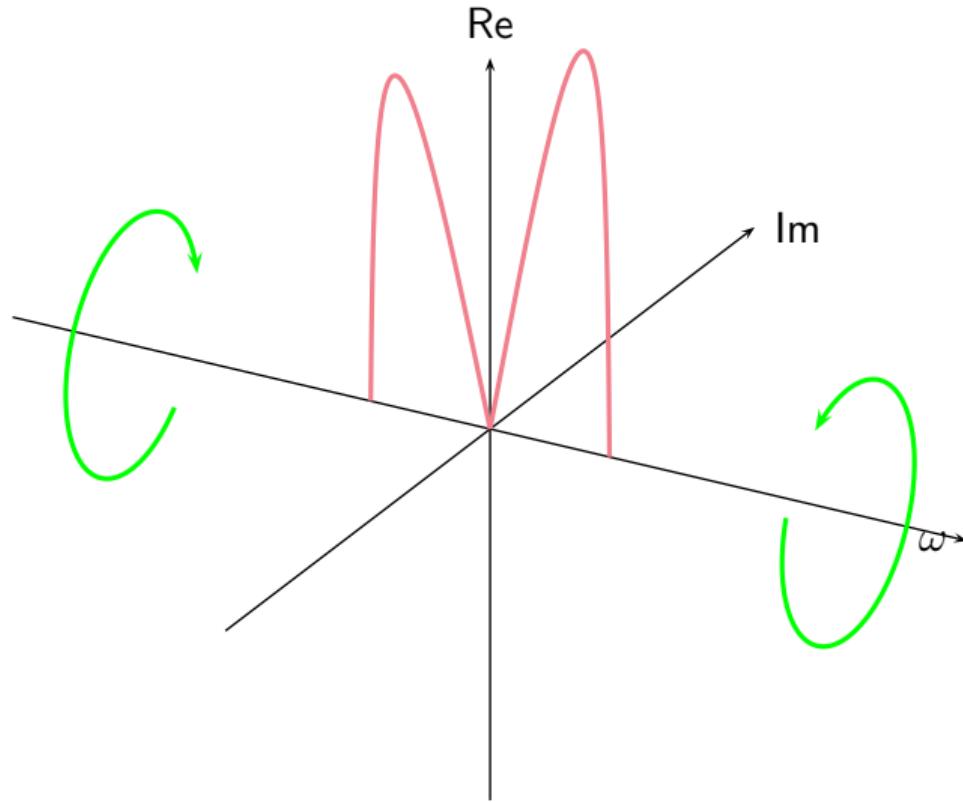
effect of the Hilbert filter (imaginary part)



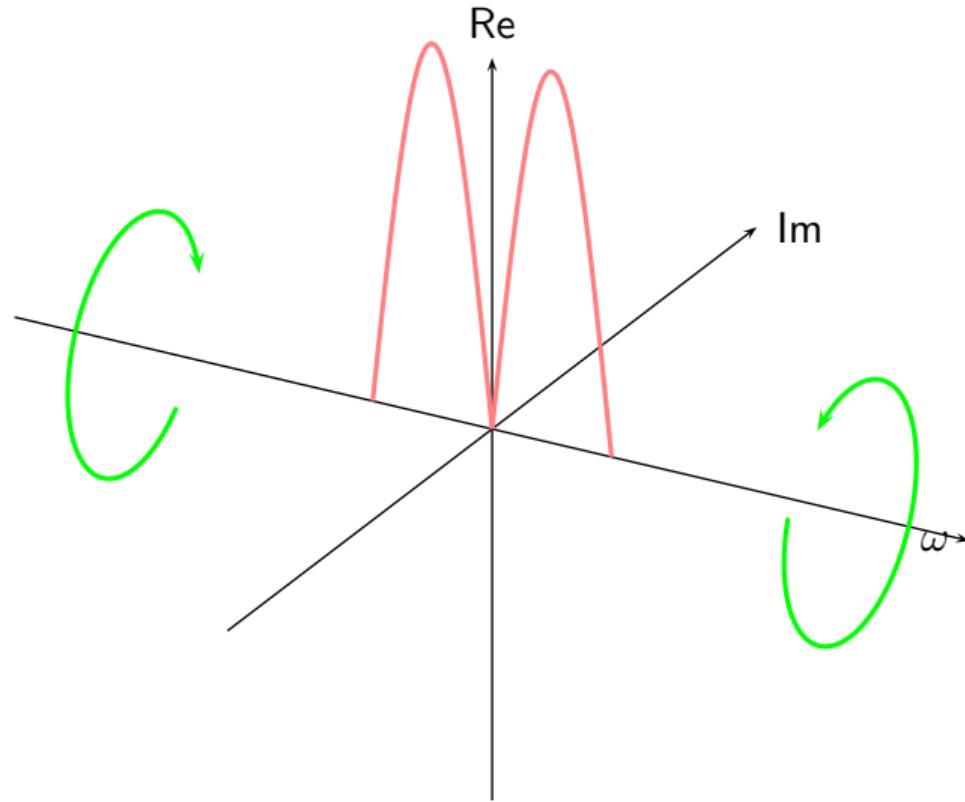
effect of the Hilbert filter (imaginary part)



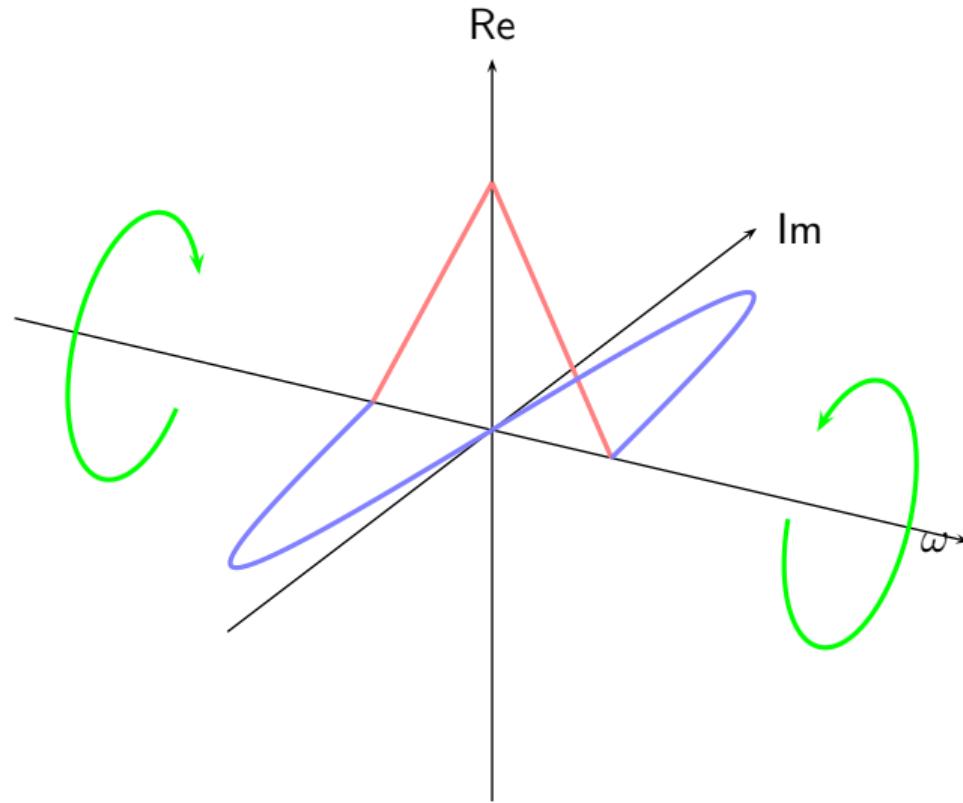
effect of the Hilbert filter (imaginary part)



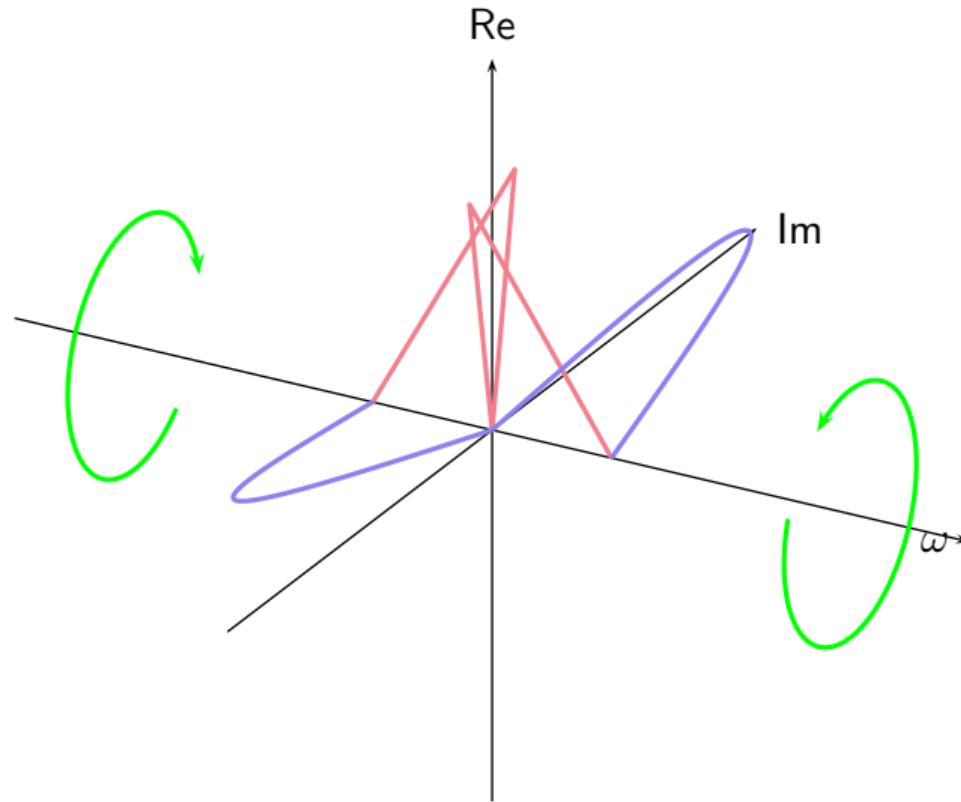
effect of the Hilbert filter (imaginary part)



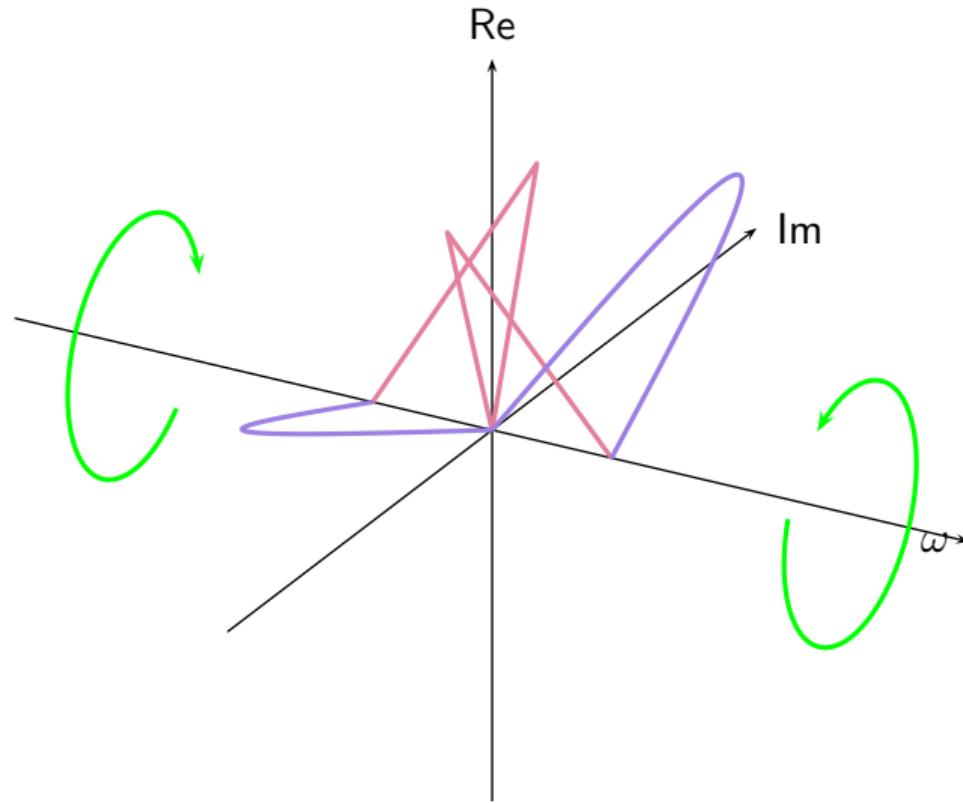
effect of the Hilbert filter



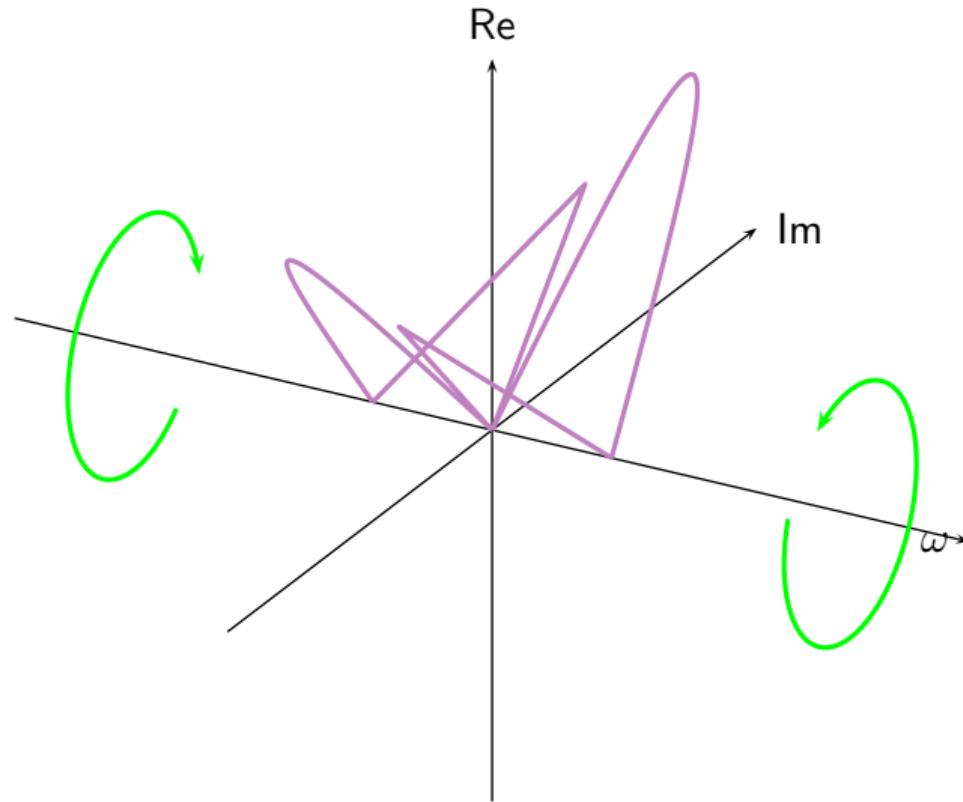
effect of the Hilbert filter



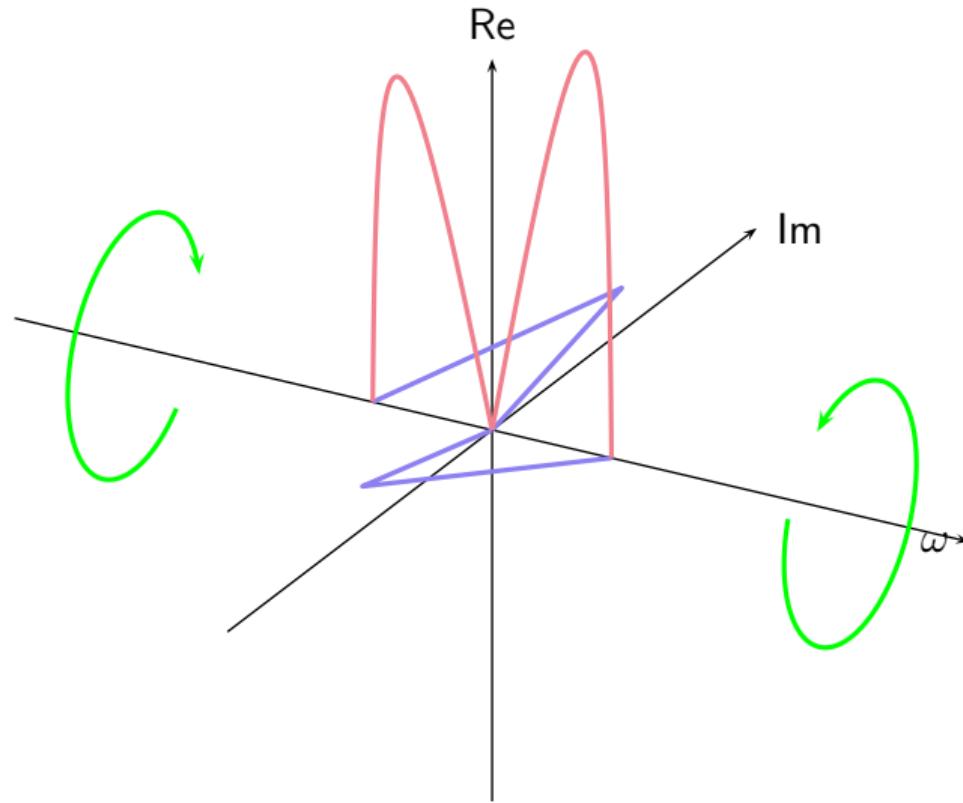
effect of the Hilbert filter



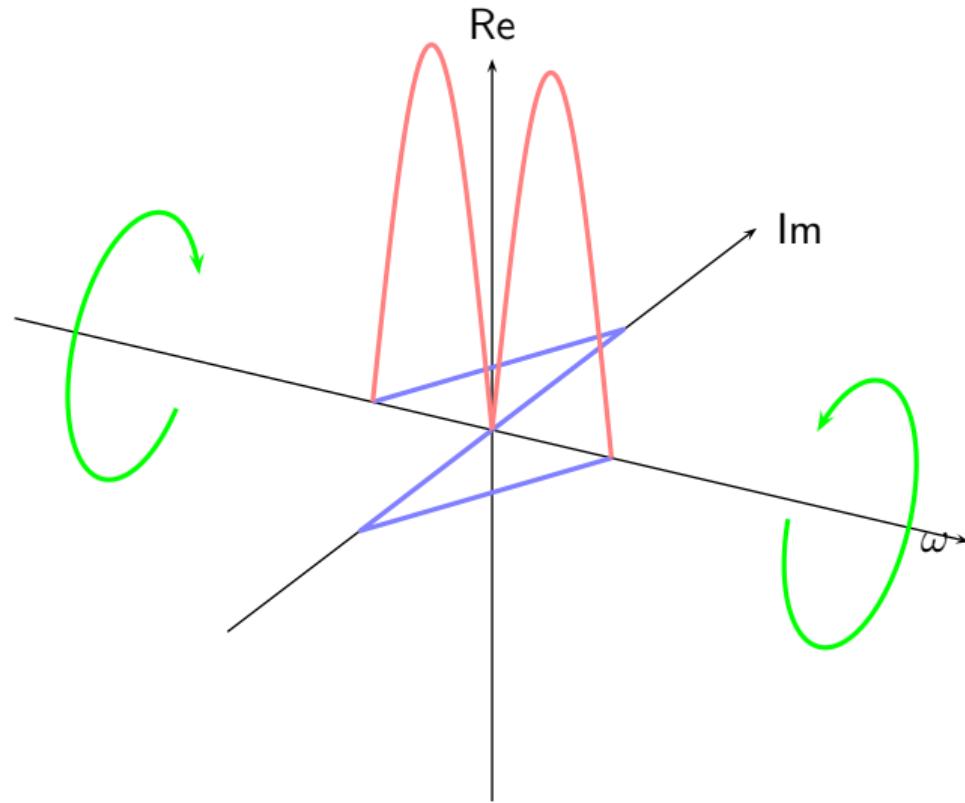
effect of the Hilbert filter



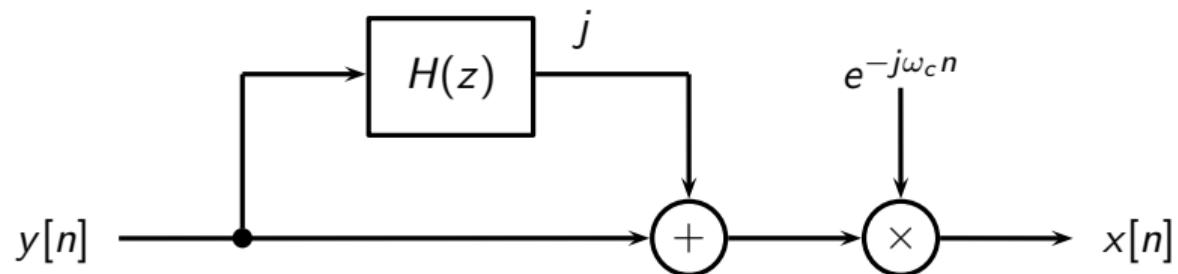
effect of the Hilbert filter



effect of the Hilbert filter

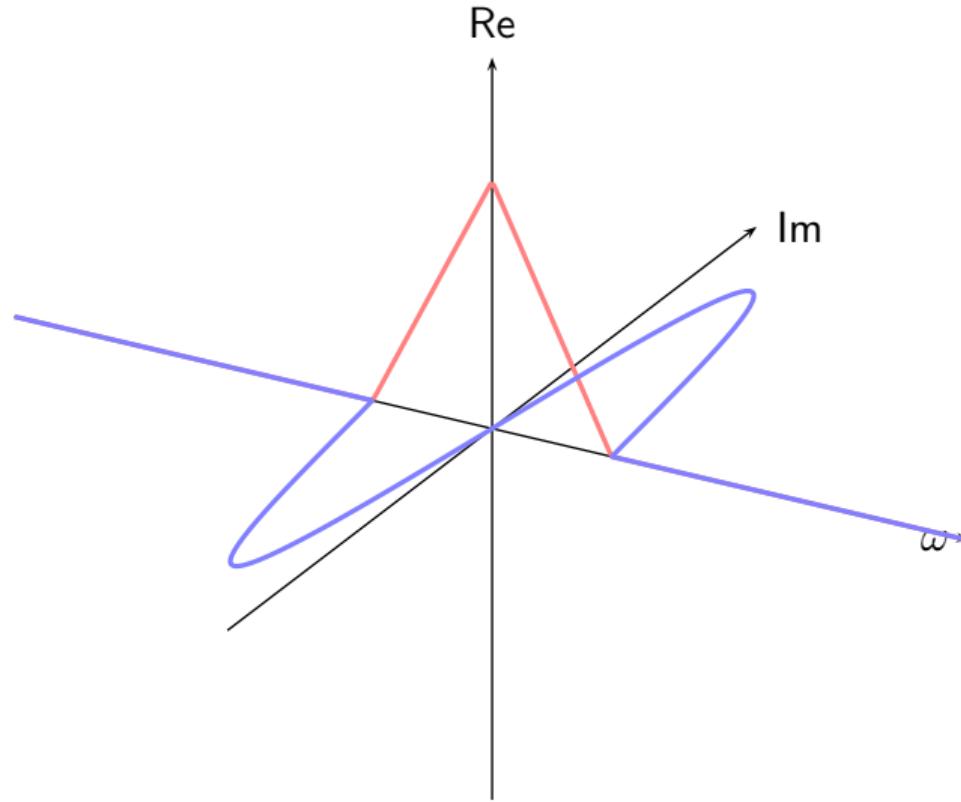


Hilbert demodulation



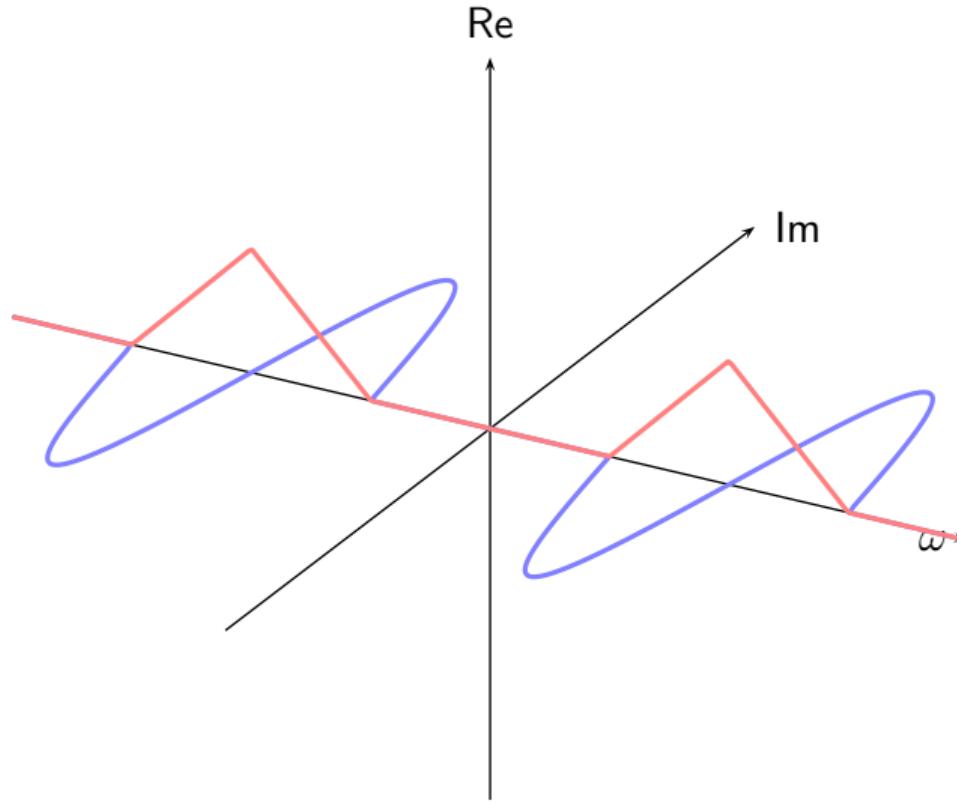
Hilbert demodulation

$x[n]$



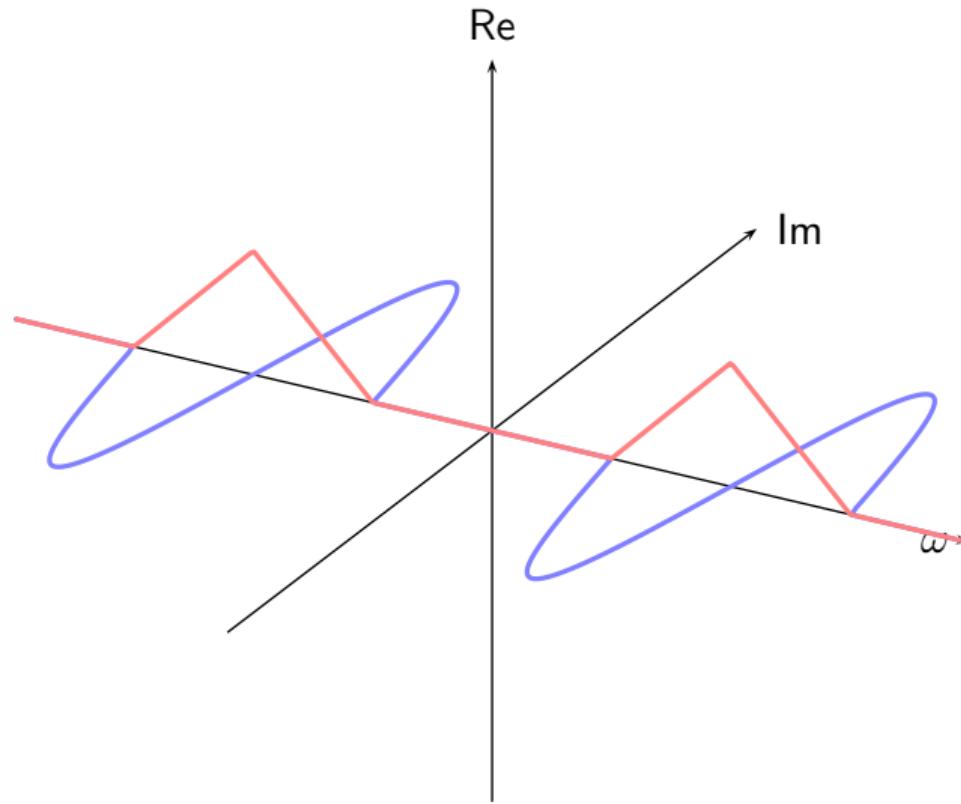
Hilbert demodulation

$$x[n] \cos(\omega_0 n) = y[n]$$



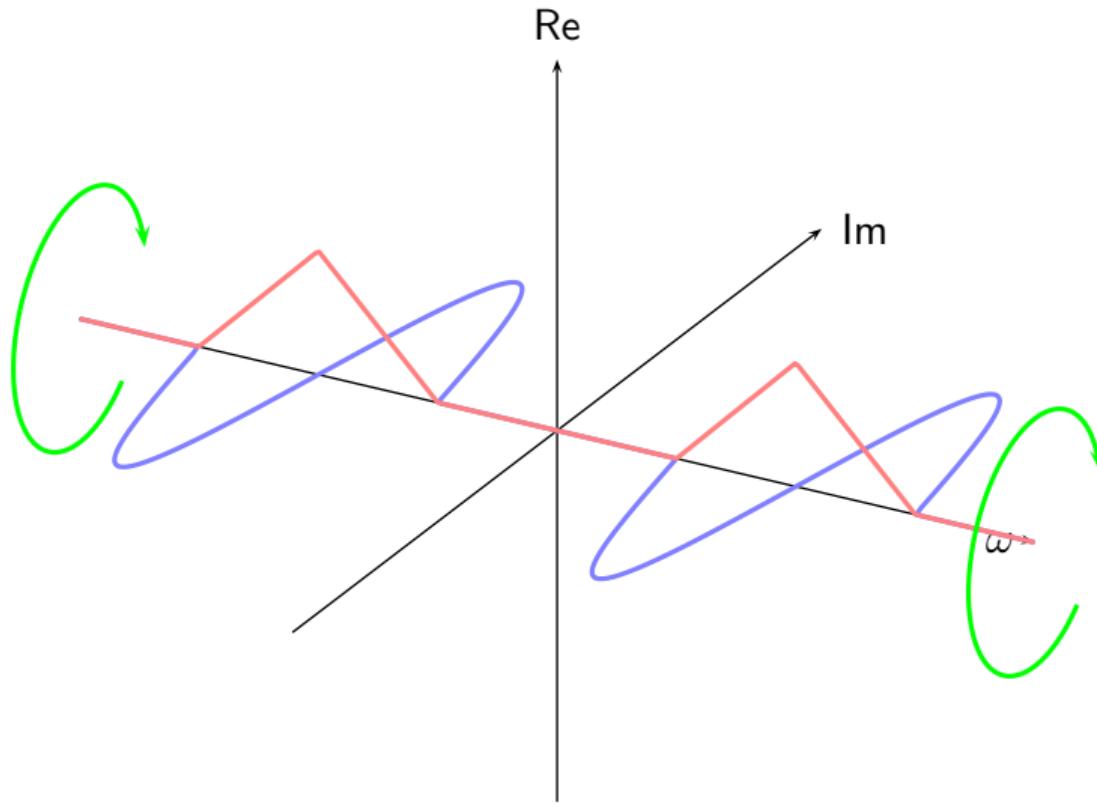
Hilbert demodulation: $jy[n] * h[n]$

$y[n]$



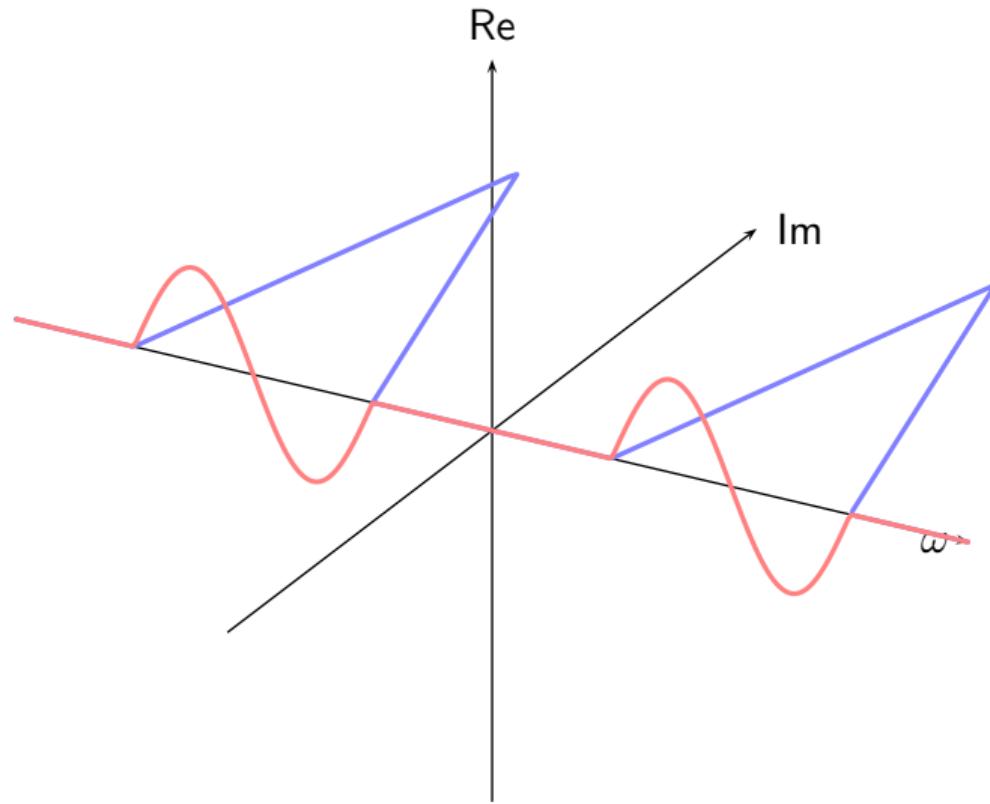
Hilbert demodulation: $jy[n] * h[n]$

$jy[n]$



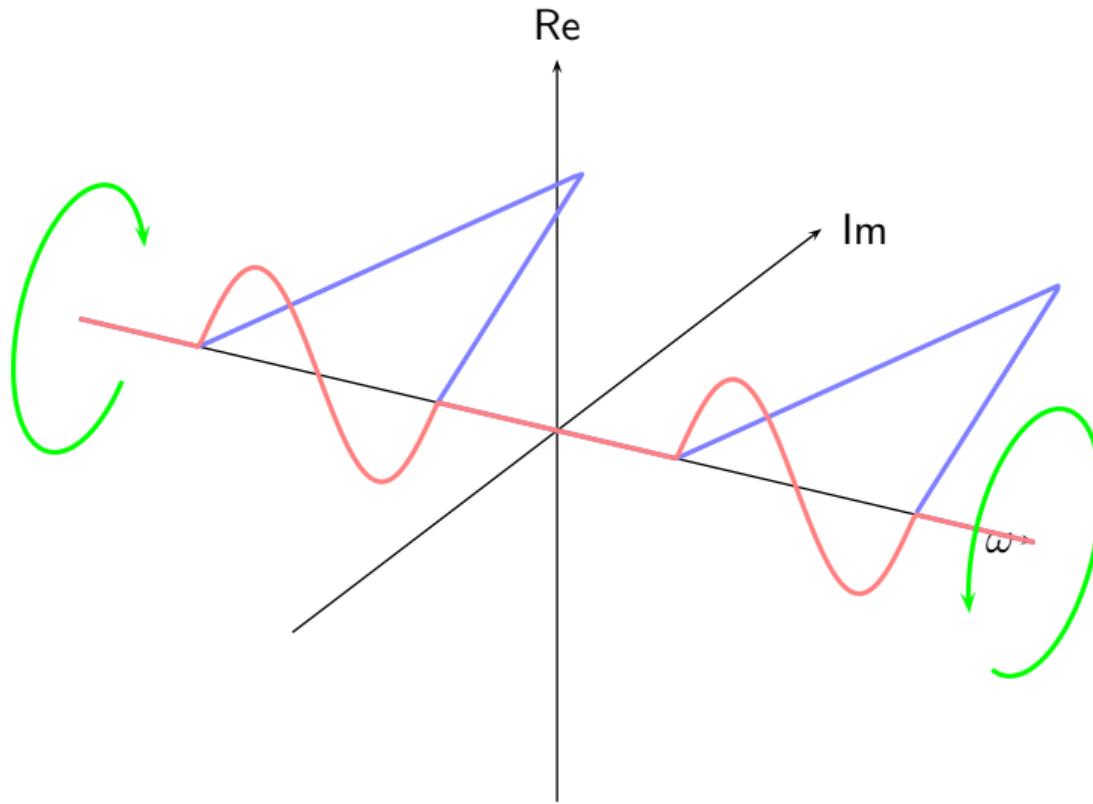
Hilbert demodulation: $jy[n] * h[n]$

$jy[n]$



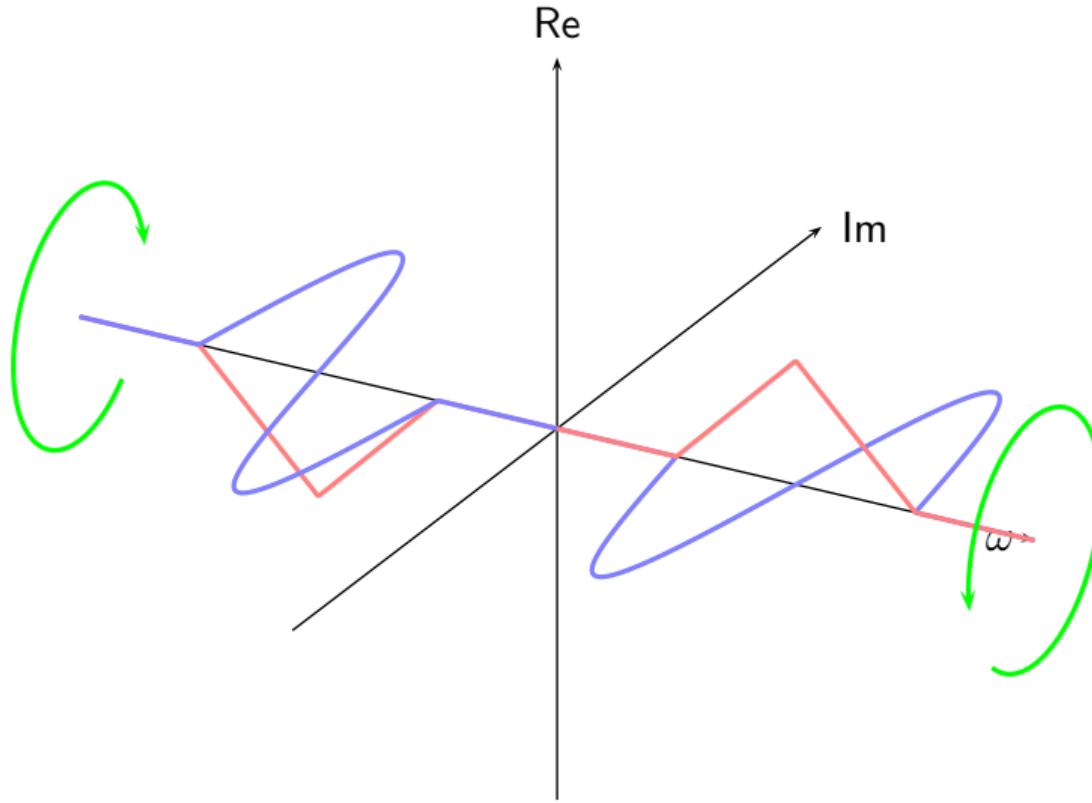
Hilbert demodulation: $jy[n] * h[n]$

$jy[n] * h[n]$

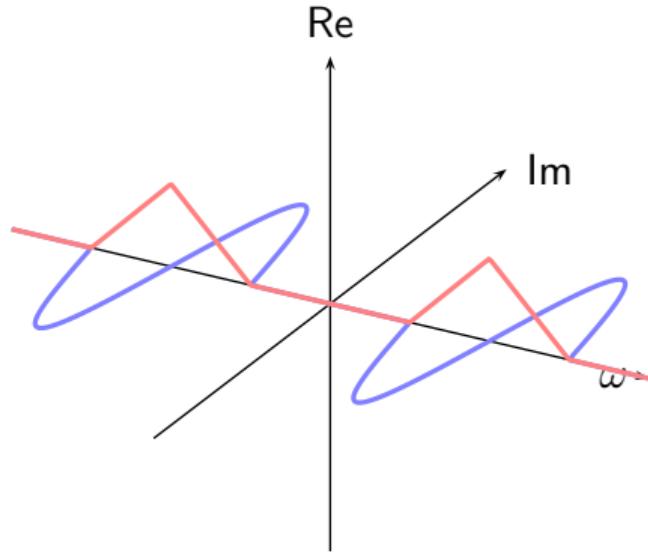


Hilbert demodulation: $jy[n] * h[n]$

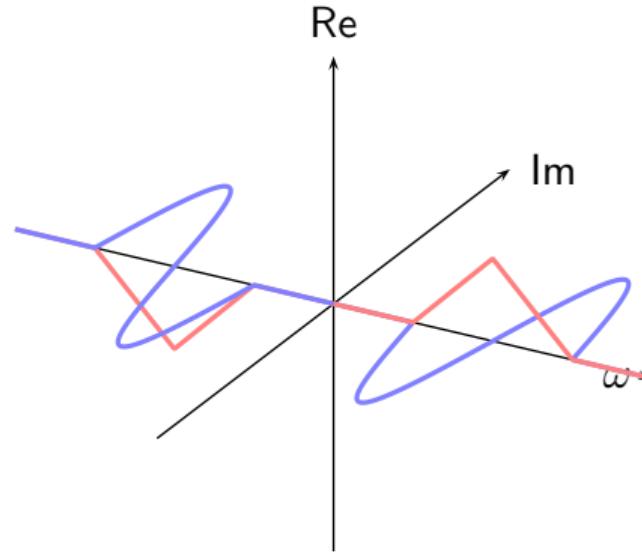
$jy[n] * h[n]$



Hilbert demodulation: $jy[n] * h[n] + y[n]$

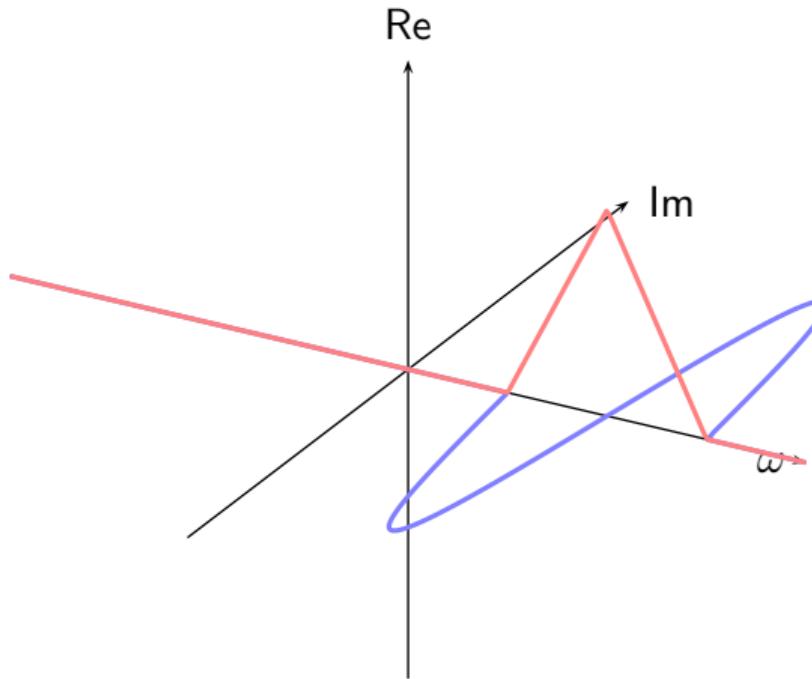


$y[n]$

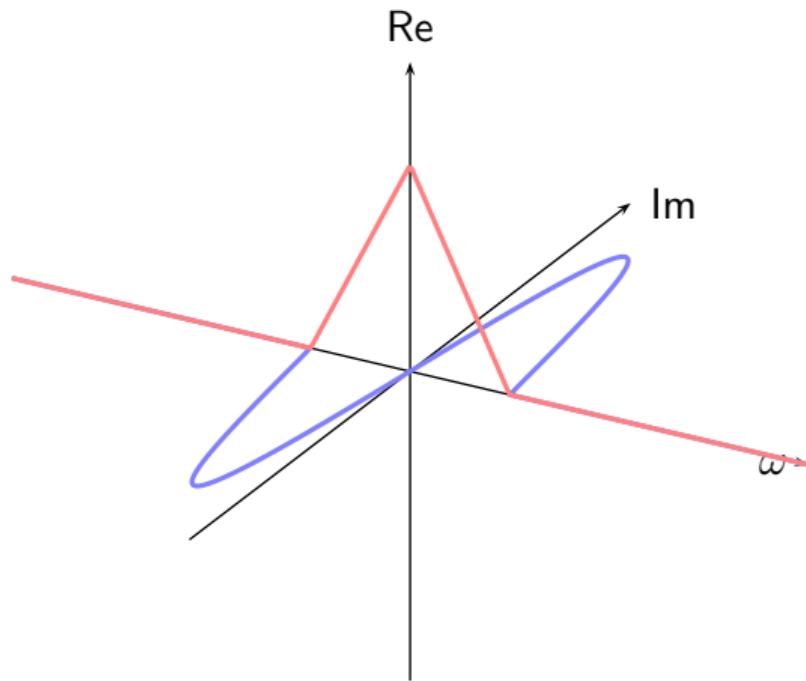


$jy[n] * h[n]$

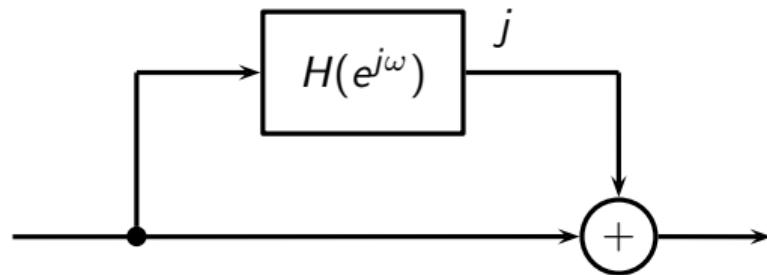
$$\text{Hilbert demodulation: } jy[n] * h[n] + y[n] = x[n]e^{j\omega_0 n}$$



Hilbert demodulation: $(jy[n] * h[n] + y[n])e^{-j\omega_0 n}$



Hilbert demodulator extracts the positive frequencies



$$G(e^{j\omega}) = 1 + jH(e^{j\omega}) = \begin{cases} 2 & 0 \leq \omega < \pi \\ 0 & -\pi \leq \omega < 0 \end{cases}$$

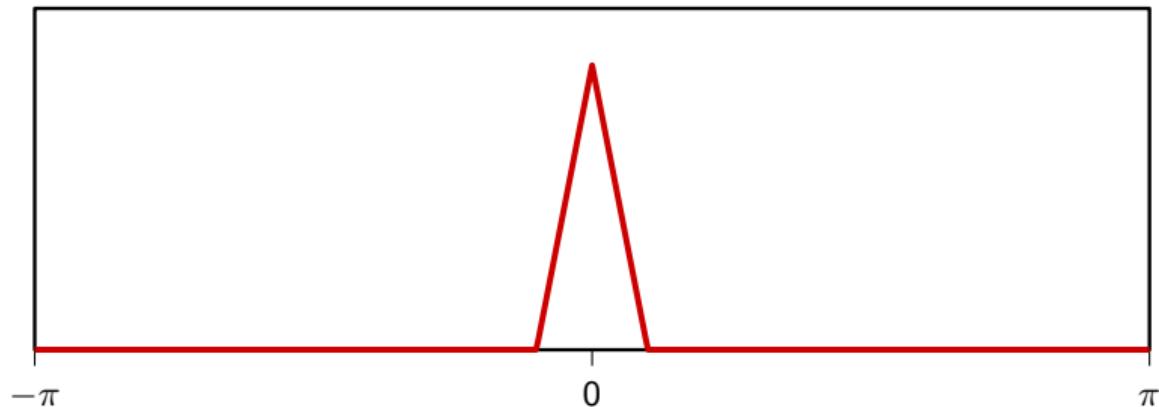
Classic Demodulation

remember the classic demodulation scheme:

- ▶ apply sinusoidal modulation to $x[n]$: $y[n] = x[n] \cos \omega_0 n$
- ▶ demodulate by multiplying by the carrier $x'[n] = y[n] \cos \omega_0 n$
- ▶ remove unwanted high-frequency components via lowpass filtering

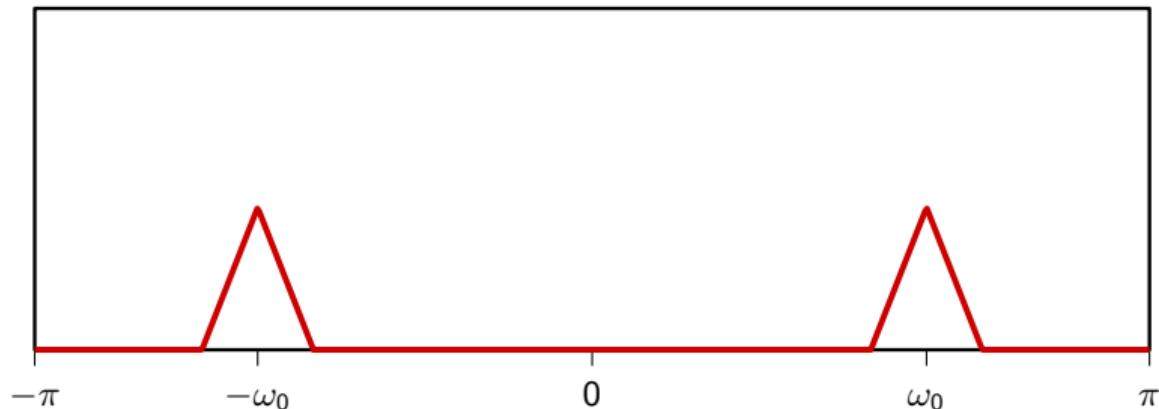
Classic Demodulation

$$X(e^{j\omega})$$



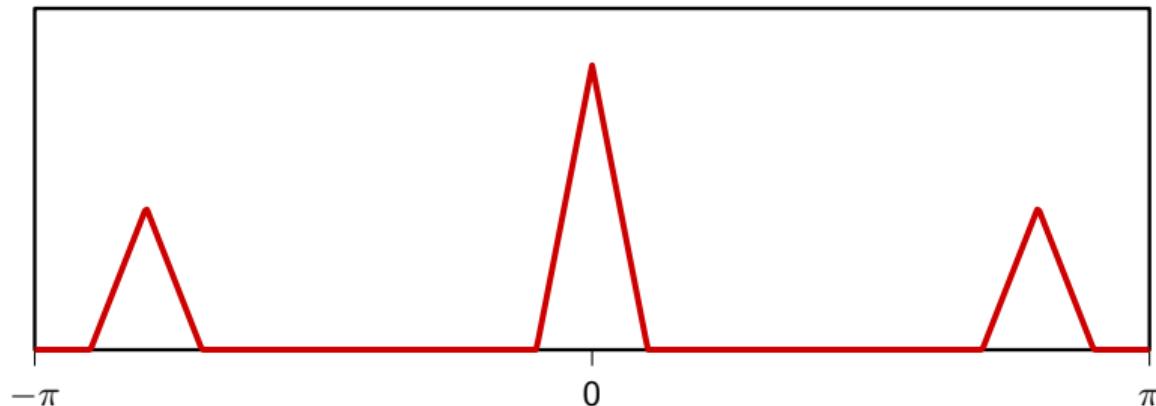
Classic Demodulation

$$Y(e^{j\omega})$$



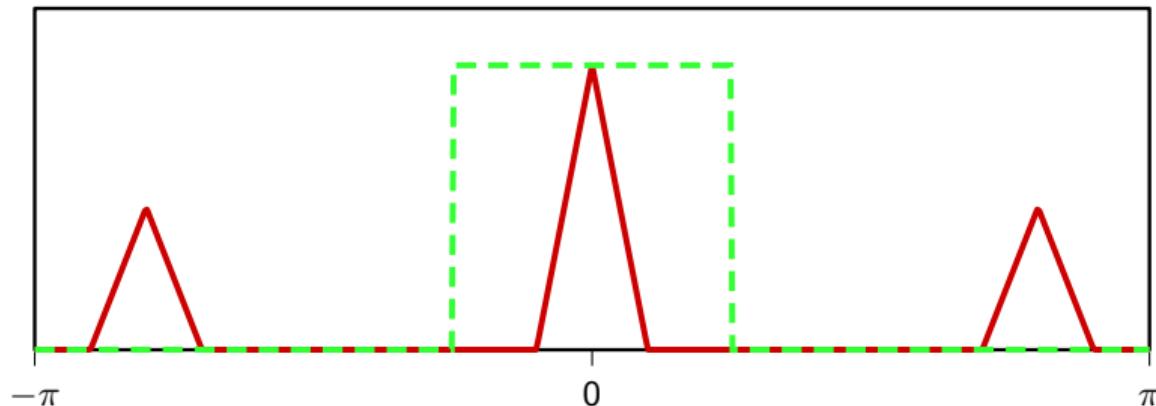
Classic Demodulation

$$X'(e^{j\omega})$$



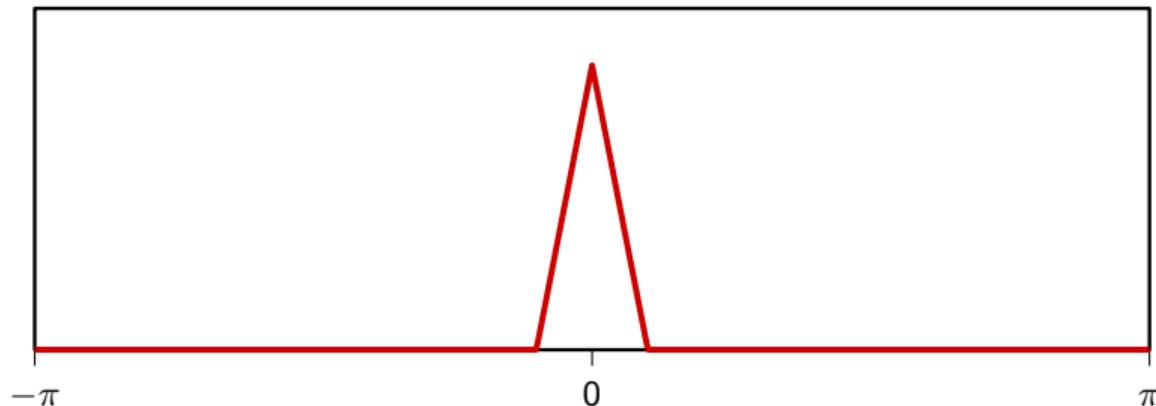
Classic Demodulation

$$X'(e^{j\omega})$$



Classic Demodulation

$$X(e^{j\omega})$$



Hilbert vs Classic Demodulation

- ▶ no need to know the bandwidth of the signal
- ▶ same filter for all modulation frequencies
- ▶ good FIR approximations for the Hilbert filter exist!

COM303: Digital Signal Processing

Lecture 13: Optimal FIR Filter design

Overview

- ▶ linear phase FIR
- ▶ the Parks-McClellan algorithm

Overview

- ▶ linear phase FIR
- ▶ the Parks-McClellan algorithm

FIR vs IIR

IIRs: pros and cons

Pros:

- ▶ computationally efficient
- ▶ strong attenuation easy
- ▶ “natural sounding” in audio applications

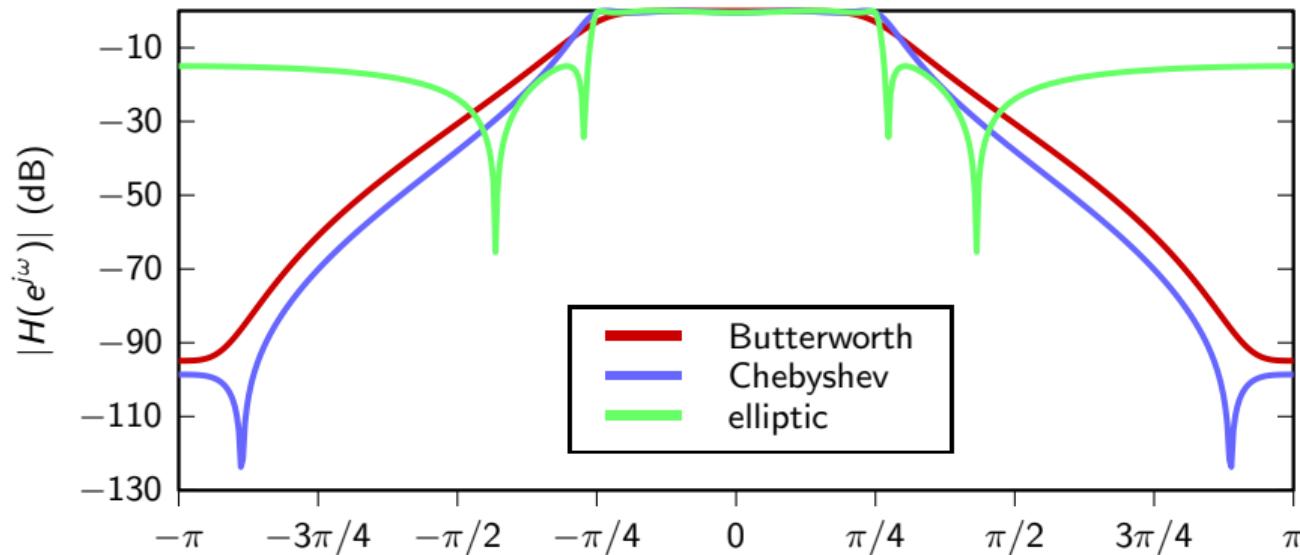
Cons:

- ▶ stability and numerical precision issues
- ▶ difficult to design for arbitrary response
- ▶ nonlinear phase

IIR design method

- ▶ based on analog filter design
- ▶ ready-made numerical tools (e.g. `b,a = sp.cheby1(4, .12, 0.25)`)
- ▶ play with order to meet the specs
- ▶ standard families: Butterworth, Chebyshev, Elliptic

4-th order IIR lowpass comparison



all filters require 9 multiplications per output sample

FIRs: pros and cons

Pros:

- ▶ always stable
- ▶ numerically precise implementations
- ▶ can be designed with linear phase
- ▶ optimal design techniques exist

Cons:

- ▶ computationally much more expensive
- ▶ because of length, significant delay (hard to use in live audio)

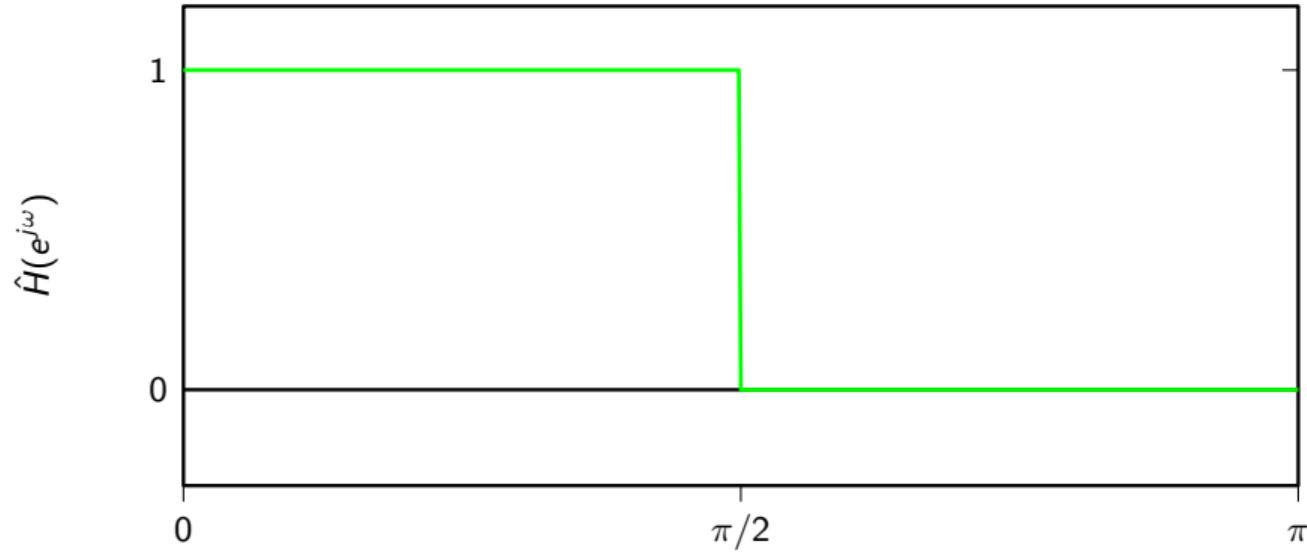
FIR design

Previous FIR designs

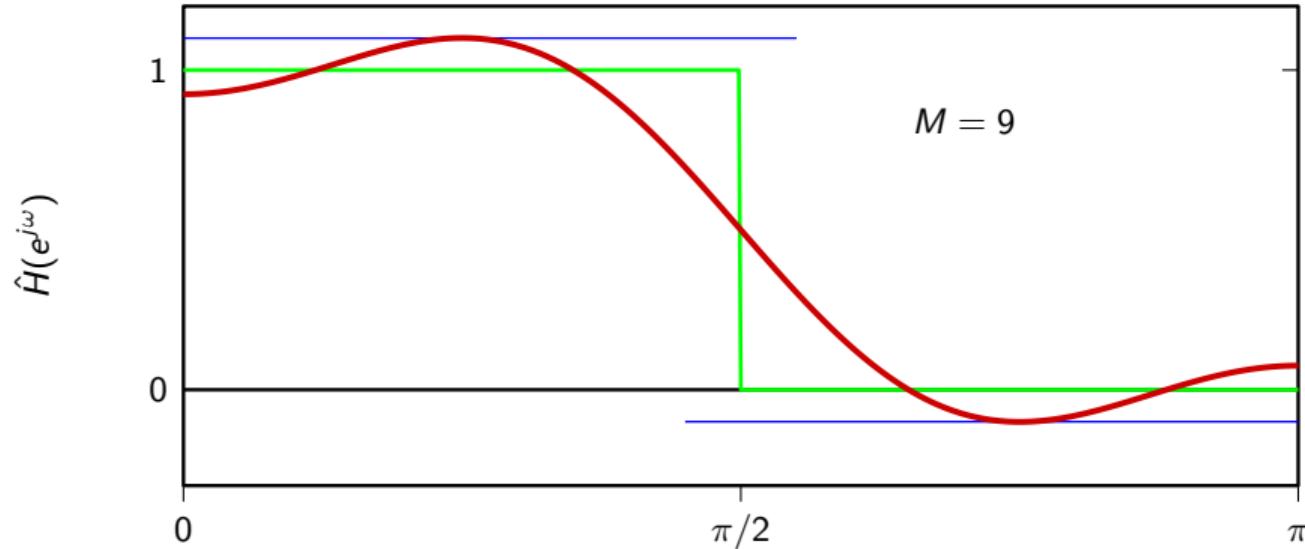
- ▶ impulse truncation, window method
- ▶ frequency sampling

design could not control the maximum error

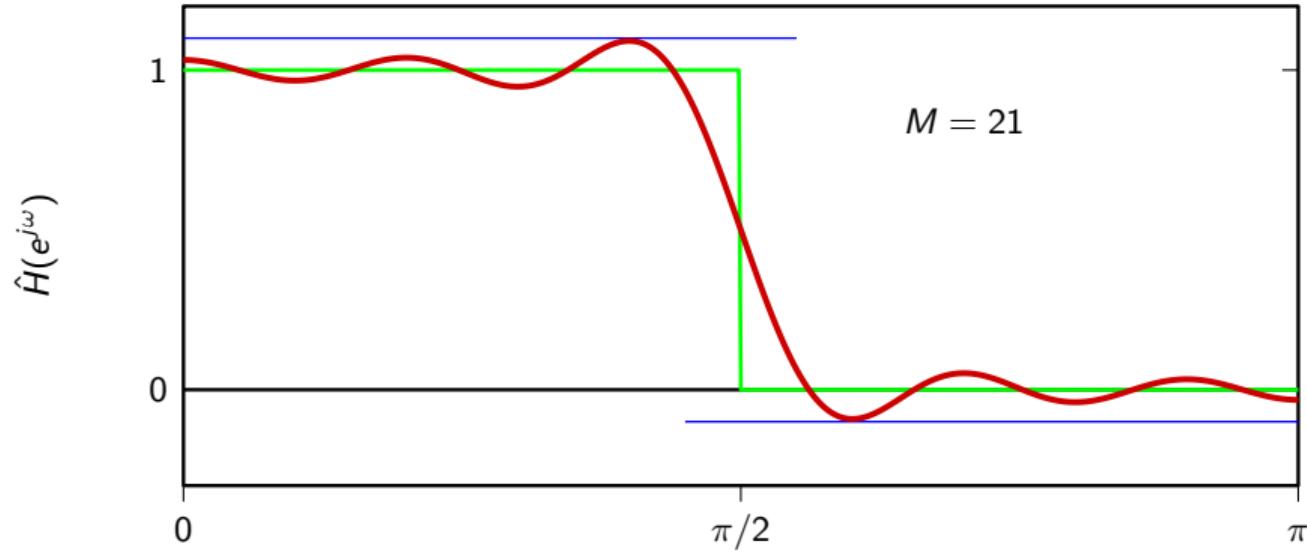
The Gibbs phenomenon



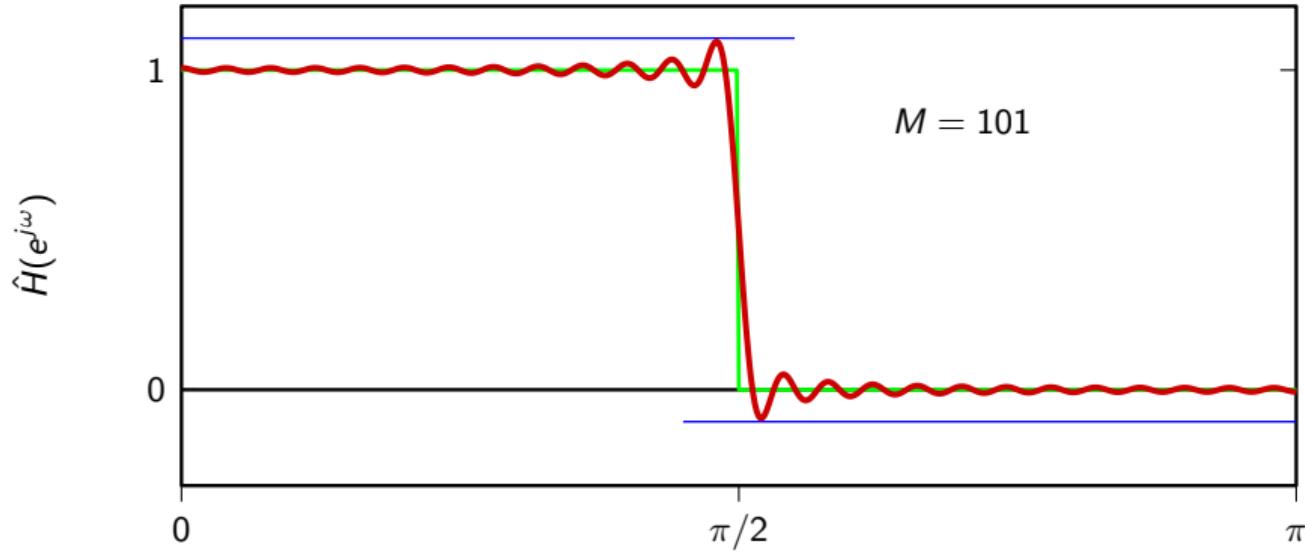
The Gibbs phenomenon



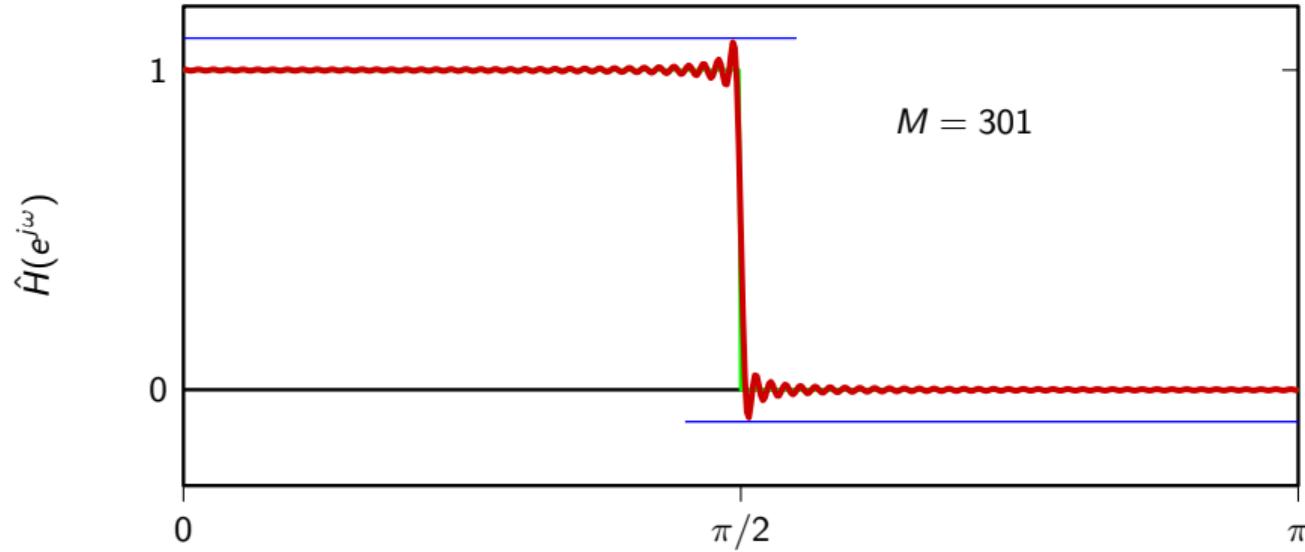
The Gibbs phenomenon



The Gibbs phenomenon



The Gibbs phenomenon



FIR: optimal minimax design

FIR filters are a digital signal processing “exclusivity”.

In the 1970s Parks and McClellan developed an algorithm to design optimal FIR filters:

- ▶ linear phase
- ▶ equiripple error in passband and stopband

algorithm proceeds by minimizing the maximum error in passband and stopband

FIR: optimal minimax design

FIR filters are a digital signal processing “exclusivity”.

In the 1970s Parks and McClellan developed an algorithm to design optimal FIR filters:

- ▶ linear phase
- ▶ equiripple error in passband and stopband

algorithm proceeds by minimizing the maximum error in passband and stopband

FIR: optimal minimax design

FIR filters are a digital signal processing “exclusivity”.

In the 1970s Parks and McClellan developed an algorithm to design optimal FIR filters:

- ▶ linear phase
- ▶ equiripple error in passband and stopband

algorithm proceeds by minimizing the maximum error in passband and stopband

FIR: optimal minimax design

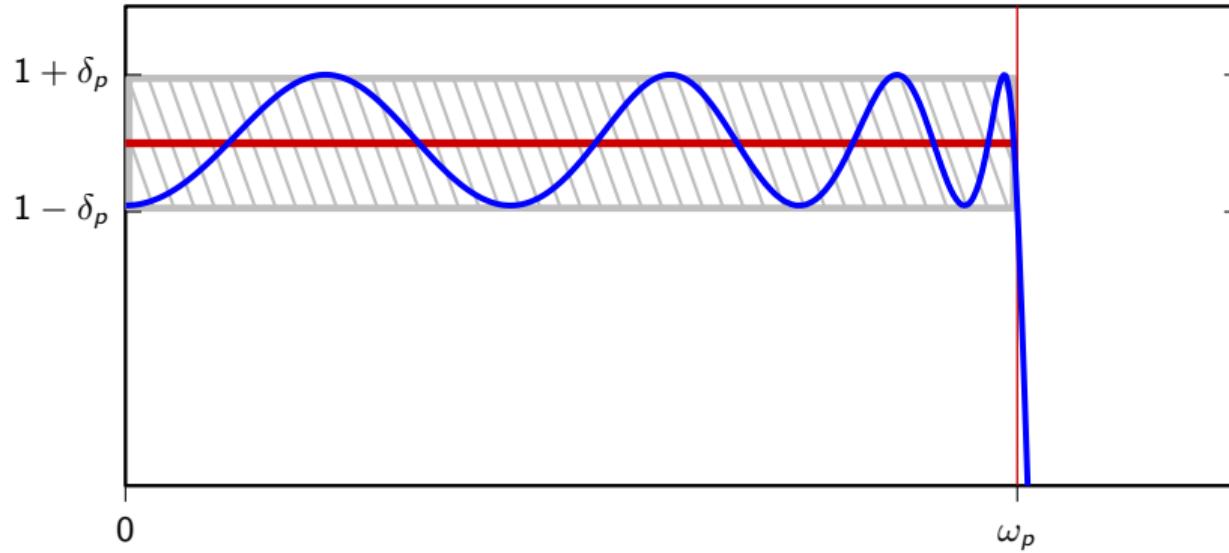
FIR filters are a digital signal processing “exclusivity”.

In the 1970s Parks and McClellan developed an algorithm to design optimal FIR filters:

- ▶ linear phase
- ▶ equiripple error in passband and stopband

algorithm proceeds by **minimizing** the **maximum** error in passband and stopband

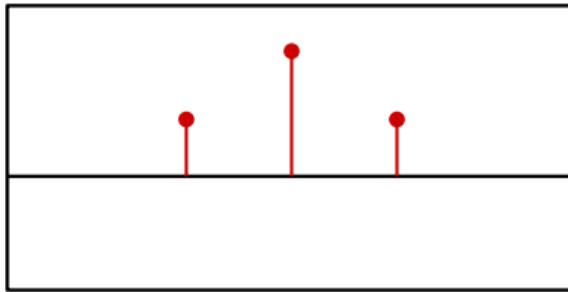
Optimal FIR will have equiripple error



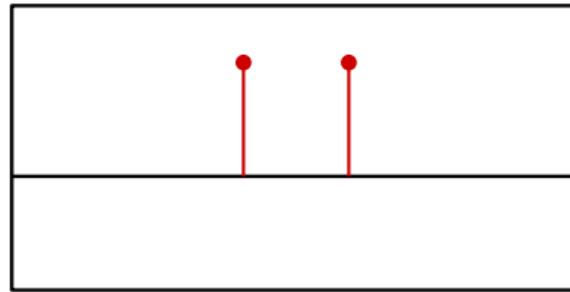
Linear phase in FIRs

Symmetric or antisymmetric impulse responses have linear phase

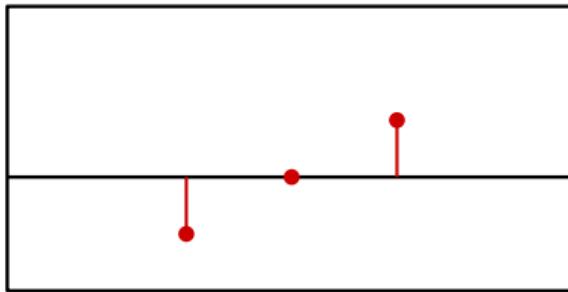
Type I



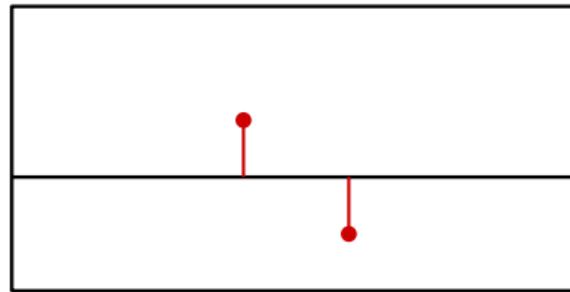
Type II



Type III



Type IV



Linear phase (Type I)

filter length is **odd**: $M = 2L + 1$

$$h[L + n] = h[L - n]$$

zero-centered filter:

$$h_d[n] = h[n + L]$$

$$h_d[n] = h_d[-n]$$

Linear phase (Type I)

filter length is **odd**: $M = 2L + 1$

$$h[L + n] = h[L - n]$$

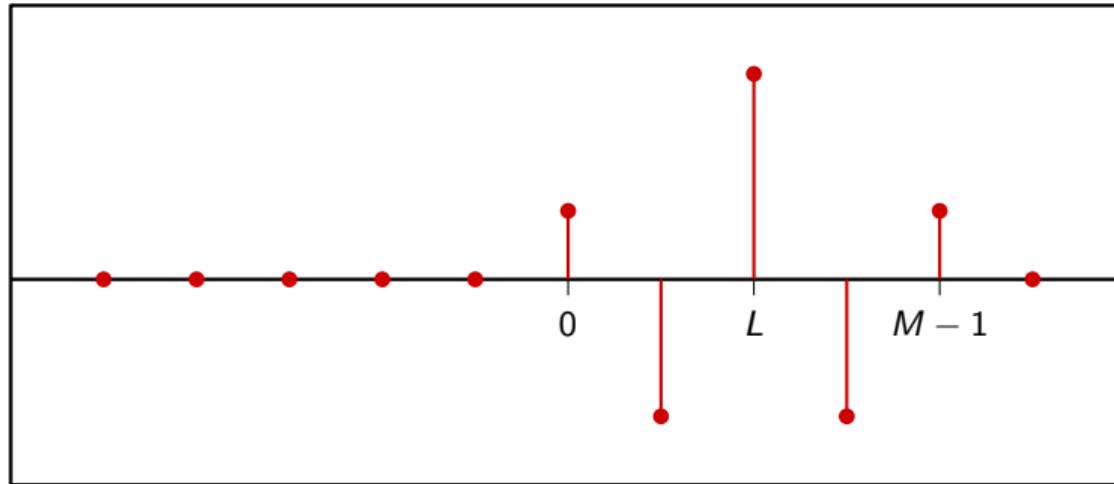
zero-centered filter:

$$h_d[n] = h[n + L]$$

$$h_d[n] = h_d[-n]$$

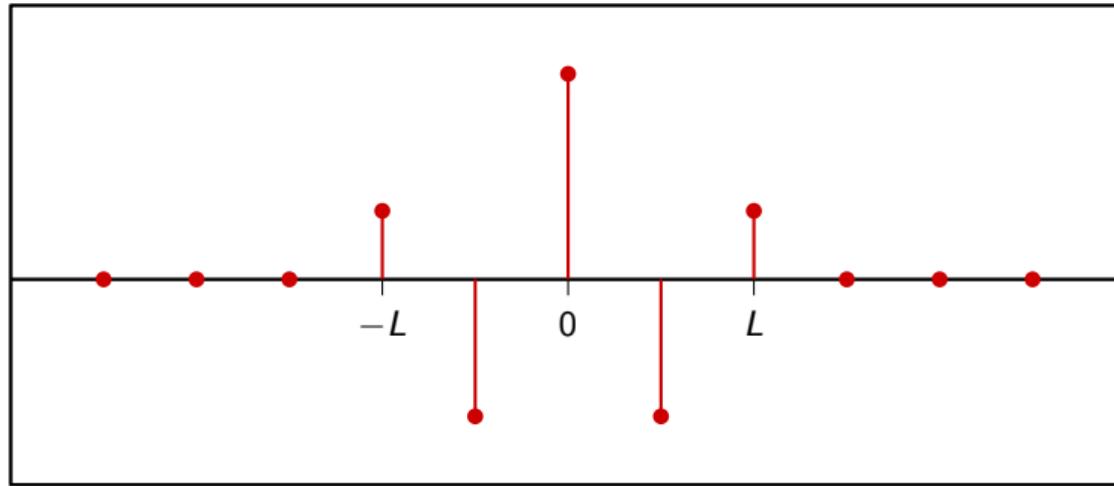
Linear phase (Type I)

$h[n]$



Linear phase (Type I)

$h_d[n]$



Linear phase (Type I)

$$\begin{aligned} H_d(z) &= \sum_{n=-L}^L h_d[n]z^{-n} \\ &= h_d[0] + \sum_{n=1}^L h_d[n](z^n + z^{-n}) \end{aligned}$$

$$\begin{aligned} H_d(e^{j\omega}) &= h_d[0] + \sum_{n=1}^L h_d[n](e^{j\omega n} + e^{-j\omega n}) \\ &= h_d[0] + 2 \sum_{n=1}^L h_d[n] \cos \omega n \quad \in \mathbb{R} \end{aligned}$$

Linear phase (Type I)

$$\begin{aligned} H_d(z) &= \sum_{n=-L}^L h_d[n]z^{-n} \\ &= h_d[0] + \sum_{n=1}^L h_d[n](z^n + z^{-n}) \end{aligned}$$

$$\begin{aligned} H_d(e^{j\omega}) &= h_d[0] + \sum_{n=1}^L h_d[n](e^{j\omega n} + e^{-j\omega n}) \\ &= h_d[0] + 2 \sum_{n=1}^L h_d[n] \cos \omega n \quad \in \mathbb{R} \end{aligned}$$

Linear phase (Type I)

$$\begin{aligned} H_d(z) &= \sum_{n=-L}^L h_d[n]z^{-n} \\ &= h_d[0] + \sum_{n=1}^L h_d[n](z^n + z^{-n}) \end{aligned}$$

$$\begin{aligned} H_d(e^{j\omega}) &= h_d[0] + \sum_{n=1}^L h_d[n](e^{j\omega n} + e^{-j\omega n}) \\ &= h_d[0] + 2 \sum_{n=1}^L h_d[n] \cos \omega n \quad \in \mathbb{R} \end{aligned}$$

Linear phase (Type I)

$$\begin{aligned} H_d(z) &= \sum_{n=-L}^L h_d[n]z^{-n} \\ &= h_d[0] + \sum_{n=1}^L h_d[n](z^n + z^{-n}) \end{aligned}$$

$$\begin{aligned} H_d(e^{j\omega}) &= h_d[0] + \sum_{n=1}^L h_d[n](e^{j\omega n} + e^{-j\omega n}) \\ &= h_d[0] + 2 \sum_{n=1}^L h_d[n] \cos \omega n \quad \in \mathbb{R} \end{aligned}$$

Linear phase (Type I)

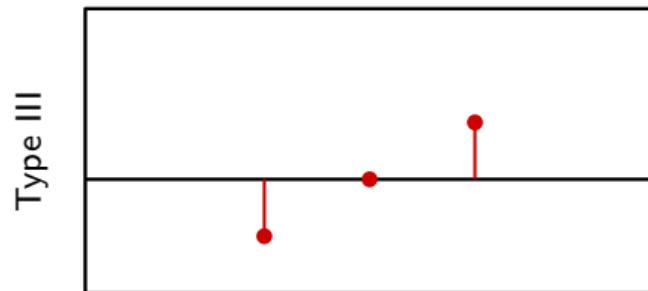
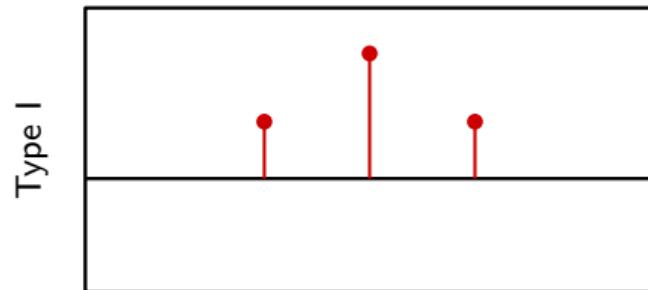
$$H(z) = z^{-L} H_d(z)$$

$$H(e^{j\omega}) = \left[h[L] + 2 \sum_{n=1}^L h[n+L] \cos n\omega \right] e^{-j\omega L}$$

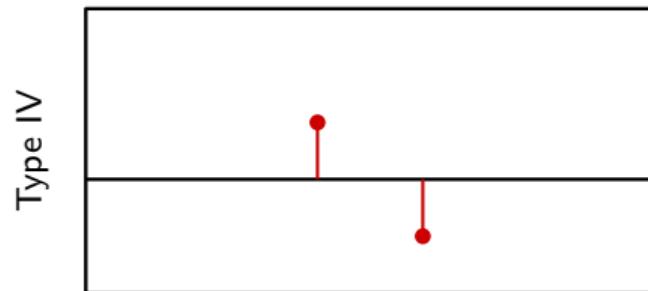
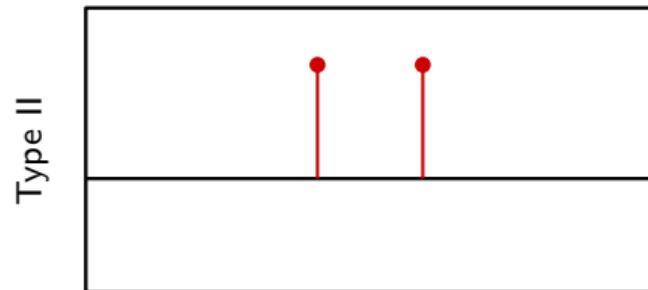
Linear Phase FIR Filters

- ▶ L : number of points with a “companion”
- ▶ even-length FIRs: $M = 2L$ taps
- ▶ odd-length FIRs: $M = 2L + 1$ taps
- ▶ delay equal to half-length: $C = (M - 1)/2$
- ▶ delay is non-integer for even-length filters!

FIR types ($L = 1$)



$$M = 2L + 1 = 3$$



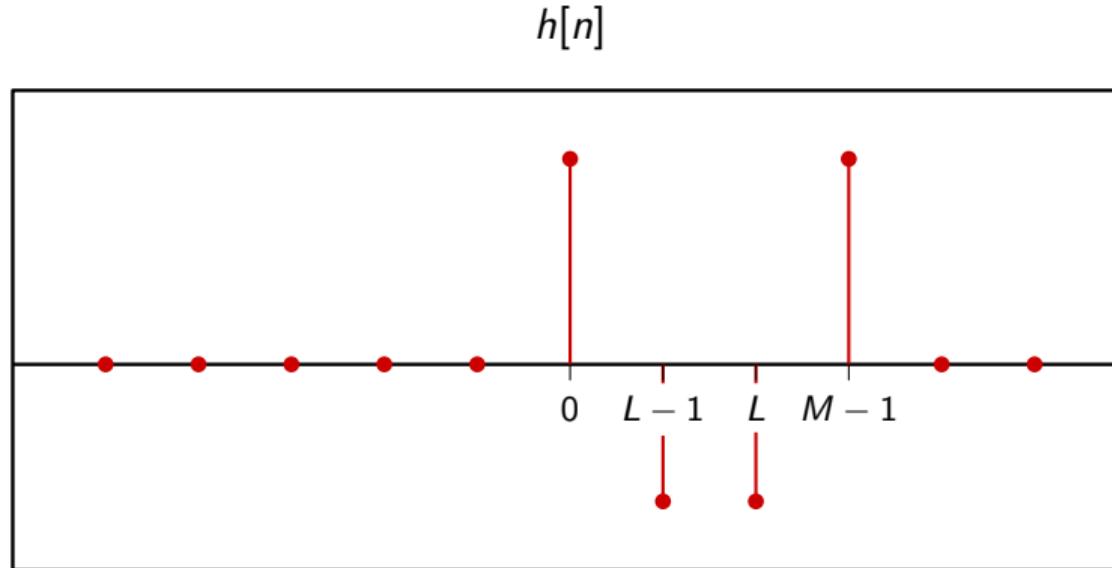
$$M = 2L = 2$$

Linear phase (Type II)

filter length is **even**: $M = 2L$

$$h[n] = h[2L - 1 - n]$$

Linear phase (Type II)



Linear phase (Type II)

$$\begin{aligned} H(z) &= h[0] + h[1]z^{-1} + \dots + h[L-1]z^{-L+1} + \\ &\quad h[2L-1]z^{-2L+1} + h[2L-2]z^{-2L+2} + \dots + h[L]z^{-L} \\ &= h[0] + h[1]z^{-1} + \dots + h[L-1]z^{-L+1} + \\ &\quad h[0]z^{-2L+1} + h[1]z^{-2L+2} + \dots + h[L-1]z^{-L} \\ &= \sum_{n=0}^{L-1} h[n](z^{-n} + z^{-2L+1+n}) \end{aligned}$$

Linear phase (Type II)

$$\begin{aligned} H(z) &= h[0] + h[1]z^{-1} + \dots + h[L-1]z^{-L+1} + \\ &\quad h[2L-1]z^{-2L+1} + h[2L-2]z^{-2L+2} + \dots + h[L]z^{-L} \\ &= h[0] + h[1]z^{-1} + \dots + h[L-1]z^{-L+1} + \\ &\quad h[0]z^{-2L+1} + h[1]z^{-2L+2} + \dots + h[L-1]z^{-L} \\ &= \sum_{n=0}^{L-1} h[n](z^{-n} + z^{-2L+1+n}) \end{aligned}$$

Linear phase (Type II)

$$\begin{aligned} H(z) &= h[0] + h[1]z^{-1} + \dots + h[L-1]z^{-L+1} + \\ &\quad h[2L-1]z^{-2L+1} + h[2L-2]z^{-2L+2} + \dots + h[L]z^{-L} \\ &= h[0] + h[1]z^{-1} + \dots + h[L-1]z^{-L+1} + \\ &\quad h[0]z^{-2L+1} + h[1]z^{-2L+2} + \dots + h[L-1]z^{-L} \\ &= \sum_{n=0}^{L-1} h[n](z^{-n} + z^{-2L+1+n}) \end{aligned}$$

Linear phase (Type II)

$$C = (M - 1)/2 = (2L - 1)/2 = L - 1/2 \quad (\text{non-integer!})$$

$$H(z) = \sum_{n=0}^{L-1} h[n](z^{-n} + z^{-2C+n})$$

$$= z^{-C} \sum_{n=0}^{L-1} h[n](z^{(C-n)} + z^{-(C-n)})$$

Linear phase (Type II)

$$C = (M - 1)/2 = (2L - 1)/2 = L - 1/2 \quad (\text{non-integer!})$$

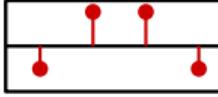
$$\begin{aligned} H(z) &= \sum_{n=0}^{L-1} h[n](z^{-n} + z^{-2C+n}) \\ &= z^{-C} \sum_{n=0}^{L-1} h[n](z^{(C-n)} + z^{-(C-n)}) \end{aligned}$$

Linear phase (Type II)

$$H(e^{j\omega}) = \left[2 \sum_{n=0}^{L-1} h[n] \cos(\omega(C - n)) \right] e^{-j\omega C}$$

$$C = L - \frac{1}{2}$$

Linear Phase FIR Filters

type	length	sym.	delay	zeros	
I	odd	S	integer		
II	even	S	non-int.		
III	odd	A	integer		
IV	even	A	non-int.		

Zero locations (all types)

- ▶ FIRs have only zeros
- ▶ $h[n] \in \mathbb{R} \Rightarrow$ if z_0 is a zero, so is z_0^*

Zero locations (Type I)

$$H(z) = z^{-L} \left[h[0] + \sum_{n=1}^L h[n](z^n + z^{-n}) \right]$$

$$H(z^{-1}) = z^L \left[h[0] + \sum_{n=1}^L h[n](z^n + z^{-n}) \right]$$

$$H(z^{-1}) = z^{2L} H(z)$$

if z_0 is a zero, so is $1/z_0$

Zero locations (Type I)

$$H(z) = z^{-L} \left[h[0] + \sum_{n=1}^L h[n](z^n + z^{-n}) \right]$$

$$H(z^{-1}) = z^L \left[h[0] + \sum_{n=1}^L h[n](z^n + z^{-n}) \right]$$

$$H(z^{-1}) = z^{2L} H(z)$$

if z_0 is a zero, so is $1/z_0$

Zero locations (Type I)

$$H(z) = z^{-L} \left[h[0] + \sum_{n=1}^L h[n](z^n + z^{-n}) \right]$$

$$H(z^{-1}) = z^L \left[h[0] + \sum_{n=1}^L h[n](z^n + z^{-n}) \right]$$

$$H(z^{-1}) = z^{2L} H(z)$$

if z_0 is a zero, so is $1/z_0$

Zero locations

if z_0 is a zero, so is $1/z_0$

this is valid for all FIR types (easy to prove)

Zero locations

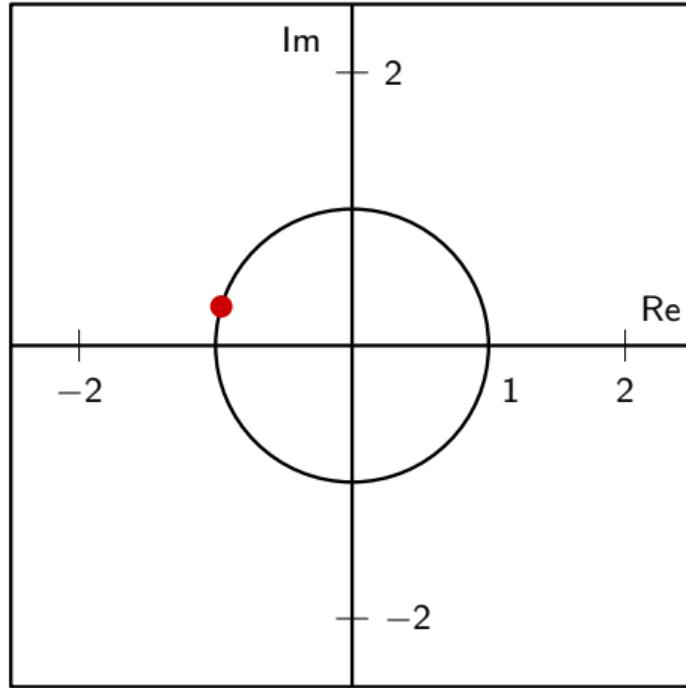
if z_0 is a zero, so is $1/z_0$

this is valid for all FIR types (easy to prove)

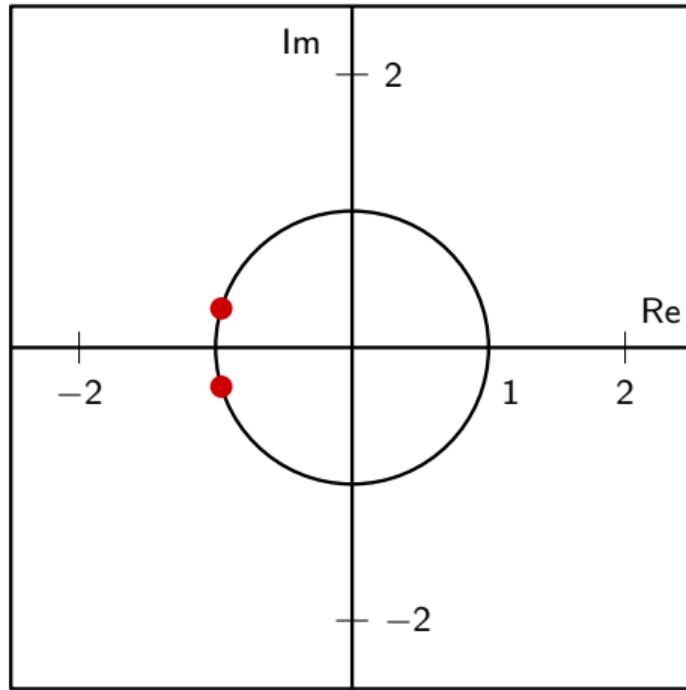
Zero locations (all types)

- ▶ if z_0 is a zero, so is z_0^*
- ▶ if z_0 is a zero, so is $1/z_0$
- ▶ if $z_0 = \rho e^{j\theta}$ is a zero so are:
 - $\rho e^{j\theta}$
 - $(1/\rho) e^{j\theta}$
 - $\rho e^{-j\theta}$
 - $(1/\rho) e^{-j\theta}$

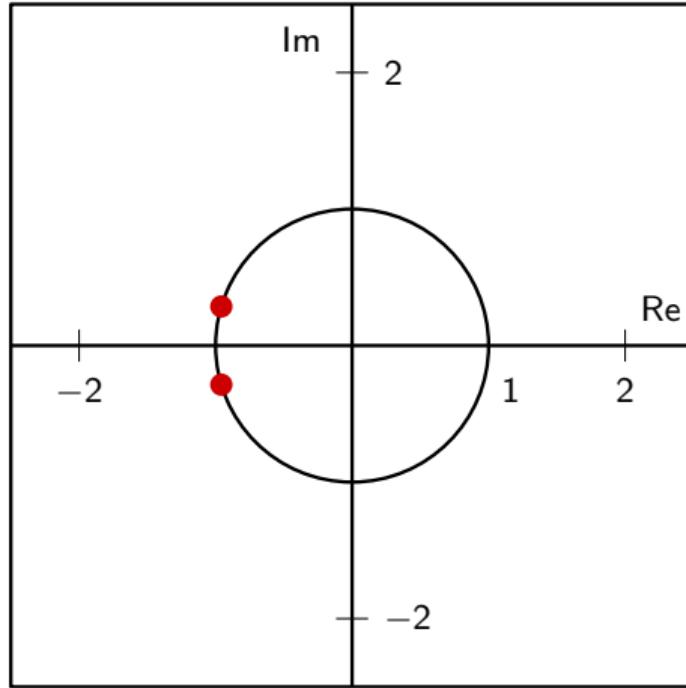
Typical zero plot for linear-phase FIR



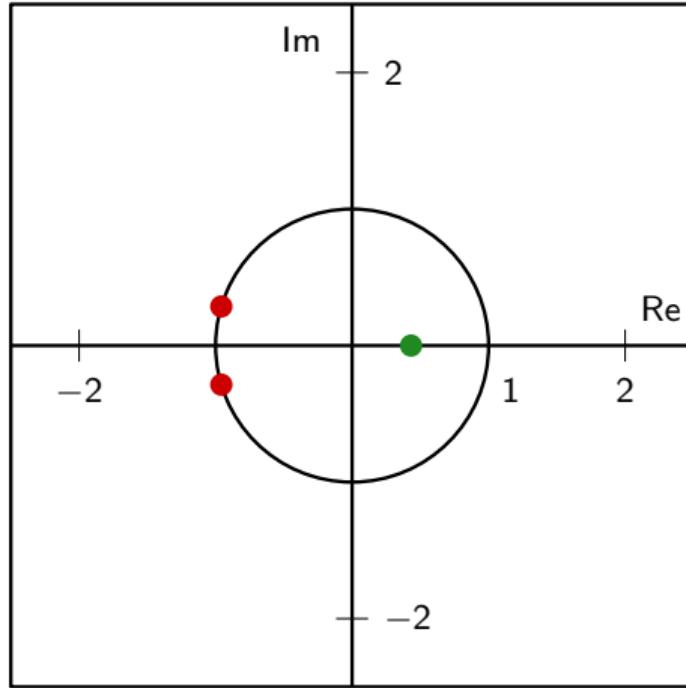
Typical zero plot for linear-phase FIR



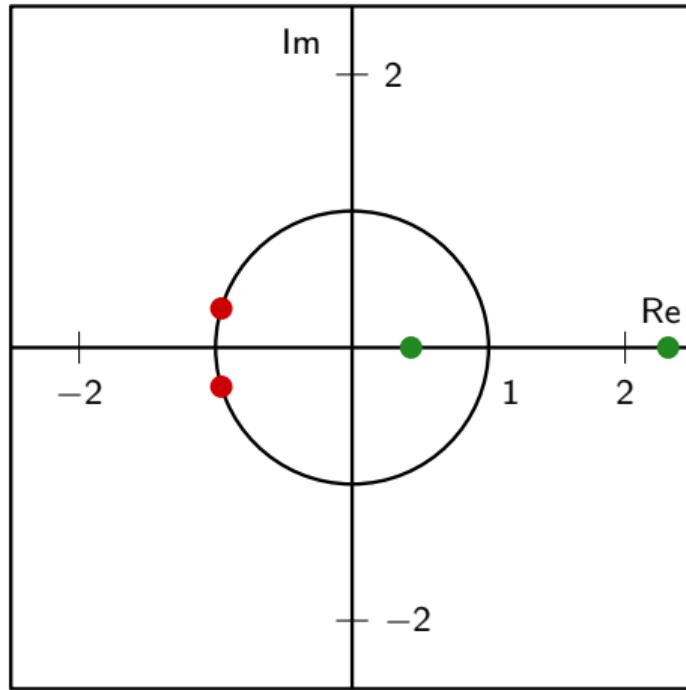
Typical zero plot for linear-phase FIR



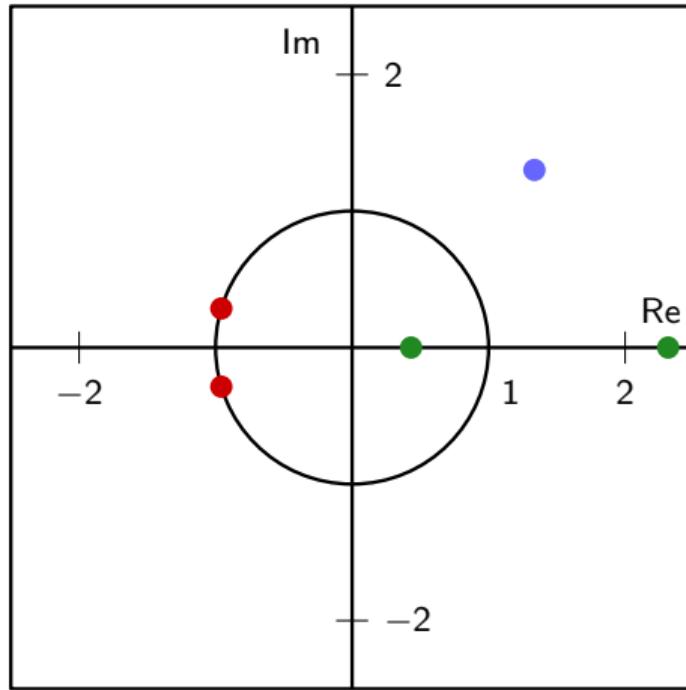
Typical zero plot for linear-phase FIR



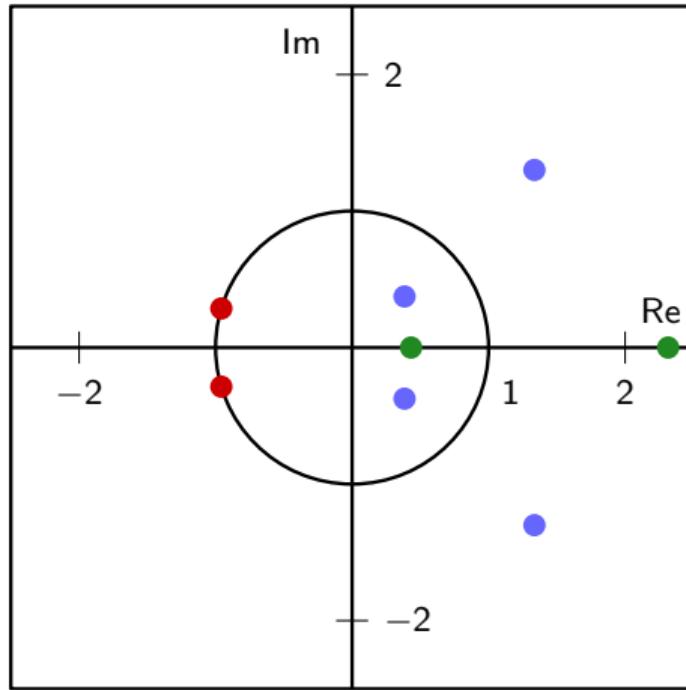
Typical zero plot for linear-phase FIR



Typical zero plot for linear-phase FIR



Typical zero plot for linear-phase FIR



Zero locations (Type II)

$$H(z) = z^{-C} \sum_{n=0}^{L-1} h[n] (z^{(C-n)} + z^{-(C-n)})$$

$$H(z^{-1}) = z^C \sum_{n=0}^{L-1} h[n] (z^{(C-n)} + z^{-(C-n)})$$

$$H(z^{-1}) = z^{2C} H(z)$$

Zero locations (Type II)

$$H(z) = z^{-C} \sum_{n=0}^{L-1} h[n] (z^{(C-n)} + z^{-(C-n)})$$

$$H(z^{-1}) = z^C \sum_{n=0}^{L-1} h[n] (z^{(C-n)} + z^{-(C-n)})$$

$$H(z^{-1}) = z^{2C} H(z)$$

Zero locations (Type II)

$$H(z) = z^{-C} \sum_{n=0}^{L-1} h[n] (z^{(C-n)} + z^{-(C-n)})$$

$$H(z^{-1}) = z^C \sum_{n=0}^{L-1} h[n] (z^{(C-n)} + z^{-(C-n)})$$

$$H(z^{-1}) = z^{2C} H(z)$$

Zero locations (Type II)

$$C = L - 1/2$$

$$\begin{aligned} H(z^{-1}) &= z^{2C} H(z) \\ &= z^{2L-1} H(z) \end{aligned}$$

$$\begin{aligned} H(-1) &= (-1)^{2L-1} H(-1) = -H(-1) \\ H(-1) &= 0 \end{aligned}$$

Zero locations (Type II)

$$C = L - 1/2$$

$$\begin{aligned} H(z^{-1}) &= z^{2C} H(z) \\ &= z^{2L-1} H(z) \end{aligned}$$

$$\begin{aligned} H(-1) &= (-1)^{2L-1} H(-1) = -H(-1) \\ H(-1) &= 0 \end{aligned}$$

Zero locations (Type II)

$$C = L - 1/2$$

$$\begin{aligned} H(z^{-1}) &= z^{2C} H(z) \\ &= z^{2L-1} H(z) \end{aligned}$$

$$\begin{aligned} H(-1) &= (-1)^{2L-1} H(-1) = -H(-1) \\ H(-1) &= 0 \end{aligned}$$

Zero locations (Type II)

$$C = L - 1/2$$

$$\begin{aligned} H(z^{-1}) &= z^{2C} H(z) \\ &= z^{2L-1} H(z) \end{aligned}$$

$$\begin{aligned} H(-1) &= (-1)^{2L-1} H(-1) = -H(-1) \\ H(-1) &= 0 \end{aligned}$$

Zero locations (Type II)

type-II FIRs always have a zero at $\omega = \pi$

Zero locations (Type III)

$$H(z) = z^{-L} \left[\sum_{n=1}^L h[n] (z^n - z^{-n}) \right]$$

$$\begin{aligned} H(z^{-1}) &= z^L \left[\sum_{n=1}^L h[n] (-z^n + z^{-n}) \right] \\ &= -z^{2L} H(z) \end{aligned}$$

$$H(1) = -H(1) \implies H(1) = 0$$

$$H(-1) = -H(-1) \implies H(-1) = 0$$

Zero locations (Type III)

$$H(z) = z^{-L} \left[\sum_{n=1}^L h[n](z^n - z^{-n}) \right]$$

$$\begin{aligned} H(z^{-1}) &= z^L \left[\sum_{n=1}^L h[n](-z^n + z^{-n}) \right] \\ &= -z^{2L} H(z) \end{aligned}$$

$$H(1) = -H(1) \implies H(1) = 0$$

$$H(-1) = -H(-1) \implies H(-1) = 0$$

Zero locations (Type III)

$$H(z) = z^{-L} \left[\sum_{n=1}^L h[n] (z^n - z^{-n}) \right]$$

$$\begin{aligned} H(z^{-1}) &= z^L \left[\sum_{n=1}^L h[n] (-z^n + z^{-n}) \right] \\ &= -z^{2L} H(z) \end{aligned}$$

$$H(1) = -H(1) \implies H(1) = 0$$

$$H(-1) = -H(-1) \implies H(-1) = 0$$

Zero locations (Type III)

$$H(z) = z^{-L} \left[\sum_{n=1}^L h[n] (z^n - z^{-n}) \right]$$

$$\begin{aligned} H(z^{-1}) &= z^L \left[\sum_{n=1}^L h[n] (-z^n + z^{-n}) \right] \\ &= -z^{2L} H(z) \end{aligned}$$

$$H(1) = -H(1) \implies H(1) = 0$$

$$H(-1) = -H(-1) \implies H(-1) = 0$$

Zero locations (Type III)

$$H(z) = z^{-L} \left[\sum_{n=1}^L h[n](z^n - z^{-n}) \right]$$

$$\begin{aligned} H(z^{-1}) &= z^L \left[\sum_{n=1}^L h[n](-z^n + z^{-n}) \right] \\ &= -z^{2L} H(z) \end{aligned}$$

$$H(1) = -H(1) \implies H(1) = 0$$

$$H(-1) = -H(-1) \implies H(-1) = 0$$

Zero locations (Type III)

type-III FIRs always have a zero at $\omega = 0$ and $\omega = \pi$

Zero locations (Type IV)

$$C = L - 1/2$$

$$H(z) = z^{-C} \sum_{n=0}^{L-1} h[n] (z^{(C-n)} - z^{-(C-n)})$$

$$\begin{aligned} H(z^{-1}) &= z^C \left[\sum_{n=0}^{L-1} h[n] (-z^{(C-n)} + z^{-(C-n)}) \right] \\ &= -z^{2C} H(z) \end{aligned}$$

$$H(1) = -H(1) \implies H(1) = 0$$

Zero locations (Type IV)

$$C = L - 1/2$$

$$H(z) = z^{-C} \sum_{n=0}^{L-1} h[n] (z^{(C-n)} - z^{-(C-n)})$$

$$\begin{aligned} H(z^{-1}) &= z^C \left[\sum_{n=0}^{L-1} h[n] (-z^{(C-n)} + z^{-(C-n)}) \right] \\ &= -z^{2C} H(z) \end{aligned}$$

$$H(1) = -H(1) \implies H(1) = 0$$

Zero locations (Type IV)

$$C = L - 1/2$$

$$H(z) = z^{-C} \sum_{n=0}^{L-1} h[n] (z^{(C-n)} - z^{-(C-n)})$$

$$\begin{aligned} H(z^{-1}) &= z^C \left[\sum_{n=0}^{L-1} h[n] (-z^{(C-n)} + z^{-(C-n)}) \right] \\ &= -z^{2C} H(z) \end{aligned}$$

$$H(1) = -H(1) \implies H(1) = 0$$

Zero locations (Type IV)

$$C = L - 1/2$$

$$H(z) = z^{-C} \sum_{n=0}^{L-1} h[n] (z^{(C-n)} - z^{-(C-n)})$$

$$\begin{aligned} H(z^{-1}) &= z^C \left[\sum_{n=0}^{L-1} h[n] (-z^{(C-n)} + z^{-(C-n)}) \right] \\ &= -z^{2C} H(z) \end{aligned}$$

$$H(1) = -H(1) \implies H(1) = 0$$

Zero locations (Type IV)

$$C = L - 1/2$$

$$H(z) = z^{-C} \sum_{n=0}^{L-1} h[n] (z^{(C-n)} - z^{-(C-n)})$$

$$\begin{aligned} H(z^{-1}) &= z^C \left[\sum_{n=0}^{L-1} h[n] (-z^{(C-n)} + z^{-(C-n)}) \right] \\ &= -z^{2C} H(z) \end{aligned}$$

$$H(1) = -H(1) \implies H(1) = 0$$

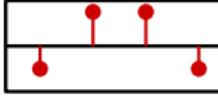
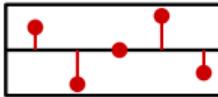
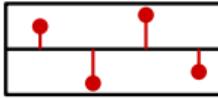
Zero locations (Type III)

type-IV FIRs always have a zero at $\omega = 0$

Zero locations

Filter Type	Relation	Constraint on Zeros
Type I	$H(z^{-1}) = z^{M-1}H(z)$	No constraints
Type II	$H(z^{-1}) = z^{M-1}H(z)$	Zero at $z = -1$ (i.e. $\omega = \pi$)
Type III	$H(z^{-1}) = -z^{M-1}H(z)$	Zeros at $z = \pm 1$ (i.e. at $\omega = \pi, \omega = 0$)
Type IV	$H(z^{-1}) = -z^{M-1}H(z)$	Zero at $z = 1$ (i.e. $\omega = 0$)

Linear Phase FIR Filters

type	length	sym.	delay	zeros	
I	odd	S	integer		
II	even	S	non-int.	$\pm\pi$	
III	odd	A	integer	$0, \pm\pi$	
IV	even	A	non-int.	0	

optimal FIR filter design

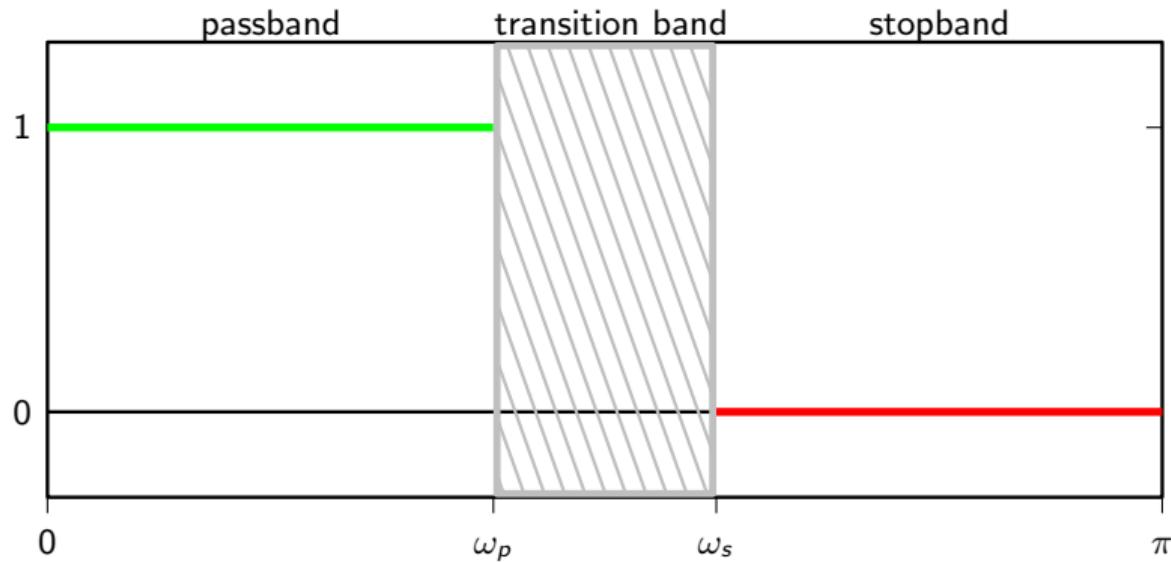
How do we design linear-phase FIRs?

answer: with the Parks-McClellan algorithm

let's work with an example:

- ▶ type I
- ▶ zero phase (work with $H_d(z)$)
- ▶ lowpass characteristic

Remember the realistic specs



Setting up the problem

Intuition #1: z -transform a finite-degree polynomial in z

$$H_d(z) = h_d[0] + \sum_{n=1}^L h_d[n](z^n + z^{-n}) = Q_M(z)$$

Setting up the problem

Intuition #2: Fourier transform also a finite-degree polynomial

$$H_d(e^{j\omega}) = h_d[0] + 2 \sum_{n=1}^L h_d[n] \cos \omega n$$

$$\cos 2\omega = 2 \cos^2 \omega - 1$$

$$\cos 3\omega = 4 \cos^3 \omega - 3 \cos \omega$$

$$\cos 4\omega = \dots$$

$$H_d(e^{j\omega}) = P_L(x)|_{x=\cos \omega}$$

Setting up the problem

Intuition #2: Fourier transform also a finite-degree polynomial

$$H_d(e^{j\omega}) = h_d[0] + 2 \sum_{n=1}^L h_d[n] \cos \omega n$$

$$\cos 2\omega = 2 \cos^2 \omega - 1$$

$$\cos 3\omega = 4 \cos^3 \omega - 3 \cos \omega$$

$$\cos 4\omega = \dots$$

$$H_d(e^{j\omega}) = P_L(x)|_{x=\cos \omega}$$

Setting up the problem

Intuition #2: Fourier transform also a finite-degree polynomial

$$H_d(e^{j\omega}) = h_d[0] + 2 \sum_{n=1}^L h_d[n] \cos \omega n$$

$$\cos 2\omega = 2 \cos^2 \omega - 1$$

$$\cos 3\omega = 4 \cos^3 \omega - 3 \cos \omega$$

$$\cos 4\omega = \dots$$

$$H_d(e^{j\omega}) = P_L(x)|_{x=\cos \omega}$$

Setting up the problem

Intuition #3: we want

$$P_L(x) \approx D(x)$$

filter design becomes polynomial fitting!

Finding the polynomial

$$H_d(e^{j\omega}) = h_d[0] + 2 \sum_{n=1}^L h_d[n] \cos \omega n$$

Step 1: Chebyshev polynomials

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_2(x) = 2x^2 - 1$$

...

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$$

Step 1: Chebyshev polynomials

fundamental property:

$$T_n(\cos \omega) = \cos n\omega$$

Step 1: Chebyshev polynomials

$$H_d(e^{j\omega}) = h_d[0] + \sum_{n=1}^L 2h_d[n] \cos n\omega$$

$$P(x) = h_d[0] + \sum_{n=1}^L 2h_d[n] T_n(x) \Big|_{x=\cos\omega}$$

$$H_d(e^{j\omega}) = P(x) \Big|_{x=\cos\omega}$$

Step 1: Chebyshev polynomials

$$H_d(e^{j\omega}) = h_d[0] + \sum_{n=1}^L 2h_d[n] \cos n\omega$$

$$P(x) = h_d[0] + \sum_{n=1}^L 2h_d[n] T_n(x) \Big|_{x=\cos \omega}$$

$$H_d(e^{j\omega}) = P(x) \Big|_{x=\cos \omega}$$

Step 1: Chebyshev polynomials

$$H_d(e^{j\omega}) = h_d[0] + \sum_{n=1}^L 2h_d[n] \cos n\omega$$

$$P(x) = h_d[0] + \sum_{n=1}^L 2h_d[n] T_n(x) \Big|_{x=\cos\omega}$$

$$H_d(e^{j\omega}) = P(x) \Big|_{x=\cos\omega}$$

Example for a 7-tap filter

$$H_d(e^{j\omega}) = a + 2b \cos \omega + 2c \cos 2\omega + 2d \cos 3\omega$$

$$= a T_0(\cos \omega) + 2b T_1(\cos \omega) + 2c T_2(\cos \omega) + 2d T_3(\cos \omega)$$

$$= a + 2b \cos \omega + 2c(2 \cos^2 \omega - 1) + 2d(4 \cos^3 \omega - 3 \cos \omega)$$

$$= (a - 2c) + (2b - 6d) \cos \omega + 4c \cos^2 \omega + 8d \cos^3 \omega$$

$$= [(a - 2c) + (2b - 6d)x + 4c x^2 + 8d x^3]_{x=\cos \omega}$$

Example for a 7-tap filter

$$H_d(e^{j\omega}) = a + 2b \cos \omega + 2c \cos 2\omega + 2d \cos 3\omega$$

$$= a T_0(\cos \omega) + 2b T_1(\cos \omega) + 2c T_2(\cos \omega) + 2d T_3(\cos \omega)$$

$$= a + 2b \cos \omega + 2c(2 \cos^2 \omega - 1) + 2d(4 \cos^3 \omega - 3 \cos \omega)$$

$$= (a - 2c) + (2b - 6d) \cos \omega + 4c \cos^2 \omega + 8d \cos^3 \omega$$

$$= [(a - 2c) + (2b - 6d)x + 4c x^2 + 8d x^3]_{x=\cos \omega}$$

Example for a 7-tap filter

$$H_d(e^{j\omega}) = a + 2b \cos \omega + 2c \cos 2\omega + 2d \cos 3\omega$$

$$= a T_0(\cos \omega) + 2b T_1(\cos \omega) + 2c T_2(\cos \omega) + 2d T_3(\cos \omega)$$

$$= a + 2b \cos \omega + 2c(2 \cos^2 \omega - 1) + 2d(4 \cos^3 \omega - 3 \cos \omega)$$

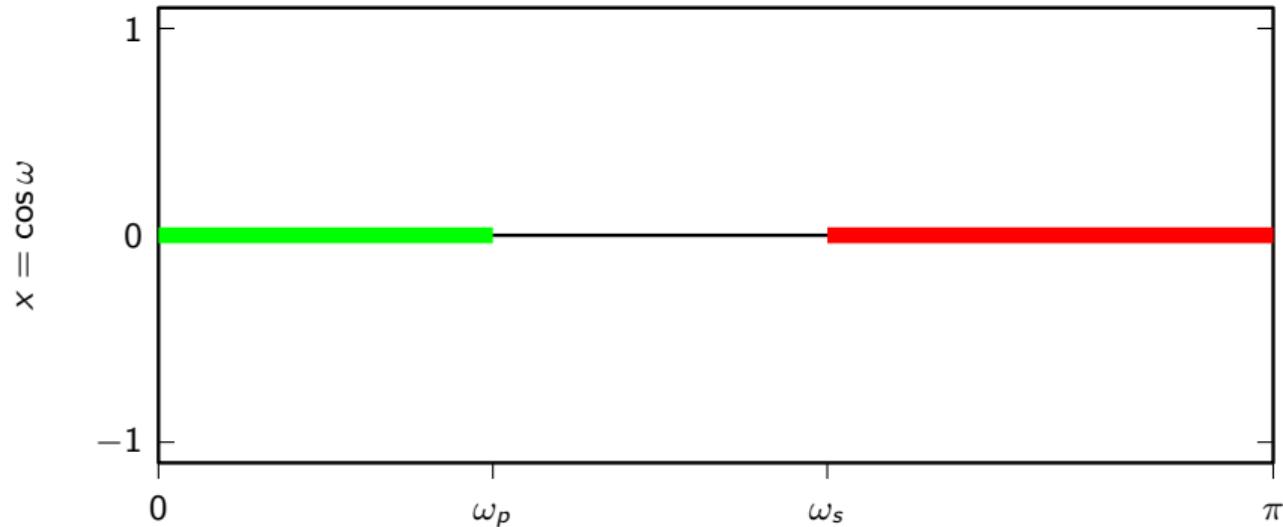
$$= (a - 2c) + (2b - 6d) \cos \omega + 4c \cos^2 \omega + 8d \cos^3 \omega$$

$$= [(a - 2c) + (2b - 6d)x + 4c x^2 + 8d x^3]_{x=\cos \omega}$$

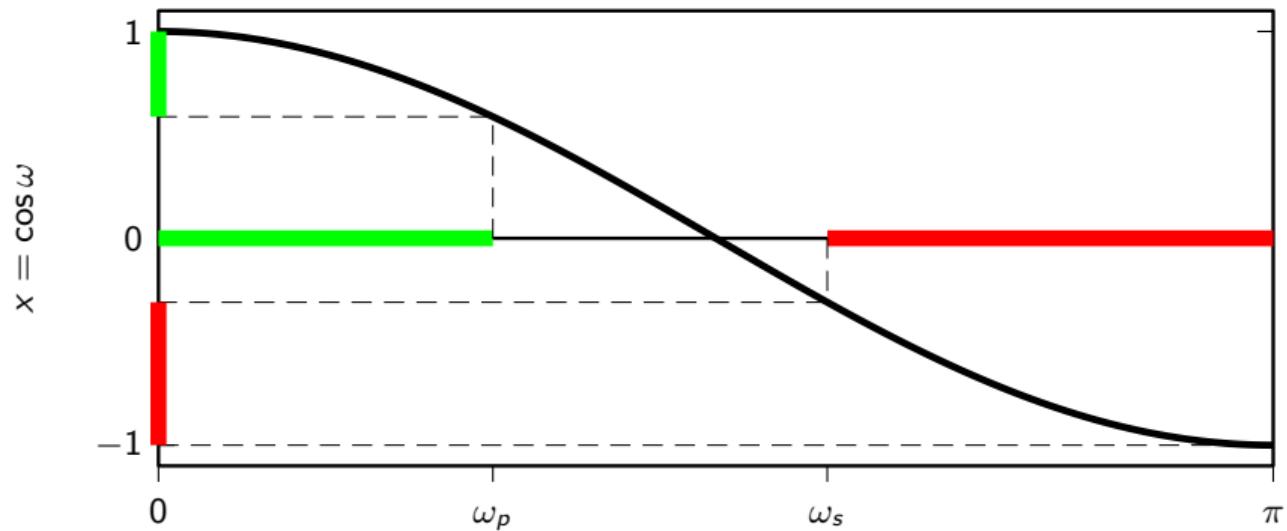
Example for a 7-tap filter

$$\begin{aligned}H_d(e^{j\omega}) &= a + 2b \cos \omega + 2c \cos 2\omega + 2d \cos 3\omega \\&= a T_0(\cos \omega) + 2b T_1(\cos \omega) + 2c T_2(\cos \omega) + 2d T_3(\cos \omega) \\&= a + 2b \cos \omega + 2c(2 \cos^2 \omega - 1) + 2d(4 \cos^3 \omega - 3 \cos \omega) \\&= (a - 2c) + (2b - 6d) \cos \omega + 4c \cos^2 \omega + 8d \cos^3 \omega \\&= [(a - 2c) + (2b - 6d)x + 4c x^2 + 8d x^3]_{x=\cos \omega}\end{aligned}$$

Step 2: Convert the specs



Step 2: Convert the specs



Step 2: Convert the specs

If $x = \cos \omega$

$$I_p = [0, \omega_p] \rightarrow I'_p = [\cos \omega_p, 1]$$

$$I_s = [\omega_p, \pi] \rightarrow I'_s = [-1, \cos \omega_s]$$

Step 2: Convert the specs

If $x = \cos \omega$

$$I_p = [0, \omega_p] \rightarrow I'_p = [\cos \omega_p, 1]$$

$$I_s = [\omega_p, \pi] \rightarrow I'_s = [-1, \cos \omega_s]$$

Step 2: Convert the specs

We want

$$P(x) \approx 1 \quad \text{for } x \in I'_p$$

$$P(x) \approx 0 \quad \text{for } x \in I'_s$$

Global error function

$$E(x) = P(x) - D(x)$$

with

$$D(x) = \begin{cases} 1 & \text{for } x \in I'_p \\ 0 & \text{for } x \in I'_s \end{cases}$$

Step 2: Convert the specs

We want

$$P(x) \approx 1 \quad \text{for } x \in I'_p$$

$$P(x) \approx 0 \quad \text{for } x \in I'_s$$

Global error function

$$E(x) = P(x) - D(x)$$

with

$$D(x) = \begin{cases} 1 & \text{for } x \in I'_p \\ 0 & \text{for } x \in I'_s \end{cases}$$

We could try this...

standard fitting of a degree- L polynomial:

- ▶ pick $L + 1$ points over the two intervals
- ▶ build the Vandermode matrix

$$\mathbf{A} = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^L \\ 1 & x_1 & x_1^2 & \dots & x_1^L \\ \vdots & & & & \\ 1 & x_L & x_L^2 & \dots & x_L^L \end{bmatrix}$$

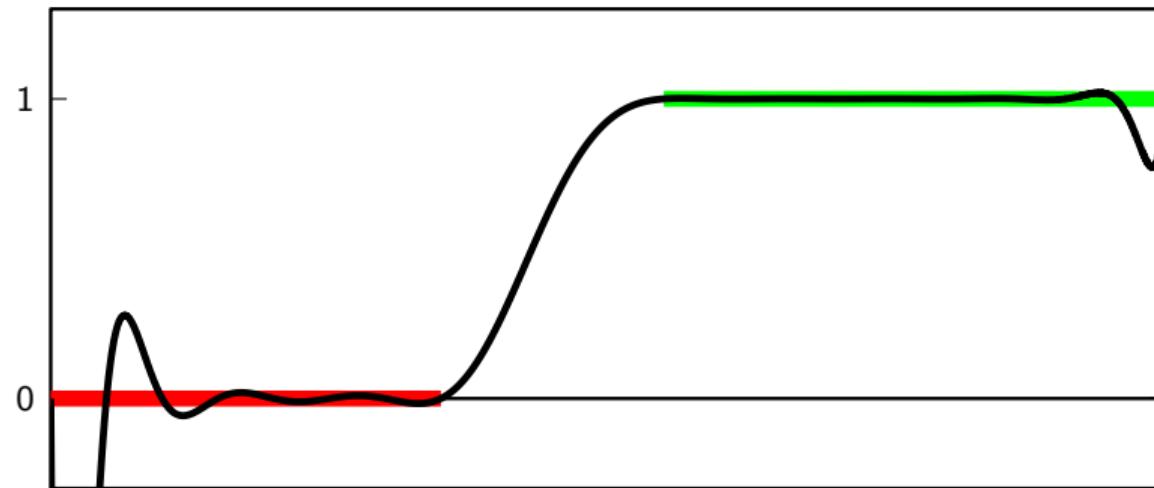
- ▶ solve the interpolation problem

$$\mathbf{Ap} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

... but it wouldn't work

- ▶ (direct methods numerically unstable)
- ▶ interpolation minimizes the MSE but not the maximum error

max error vs MSE



Brilliant idea: minimize max error

$$E = \min_{P(x)} \max_{x \in I'_p \cup I'_s} \{|P(x) - D(x)|\}$$

Alternation Theorem

$P(x)$ is the minimax approximation to $D(x)$ if and only if $P(x) - D(x)$ alternates $L + 2$ times between $+E$ and $-E$ in $I'_p \cup I'_s$

Why Alternation Theorem is key

- ▶ check candidates: if $P(x)$ satisfies the AT, we're done
- ▶ leads to a numerical algorithm to find $P(x)$: the Remez Exchange

Step 3: the Remez Algorithm

suppose we *knew* the positions of the alternations; then we could solve

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^L & \epsilon \\ 1 & x_1 & x_1^2 & \dots & x_1^L & -\epsilon \\ & & \vdots & & & \\ 1 & x_L & x_L^2 & \dots & x_L^L & (-1)^L \epsilon \end{bmatrix} \mathbf{p} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

and find both the polynomial coefficients and E

Step 3: the Remez Algorithm

obviously we don't know the positions of the alternations; but we can start with a guess

- ▶ solve the system of equation for the guessed x_i ;
- ▶ check if the solution satisfies the alternation theorem; if so, we're done
- ▶ otherwise, find the extrema of the error and use the locations as new guess; repeat

Example

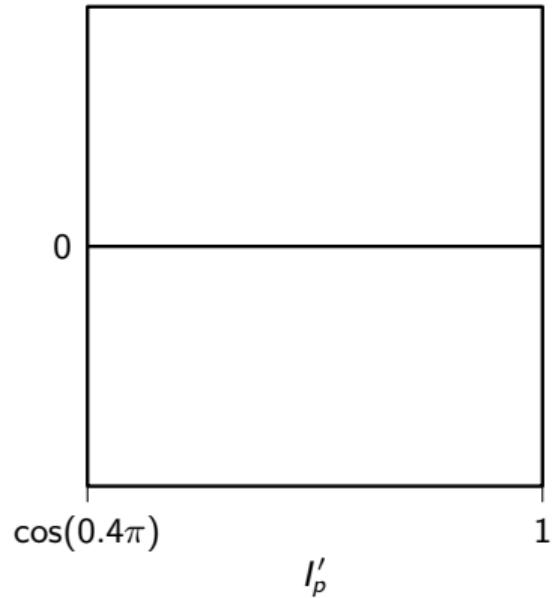
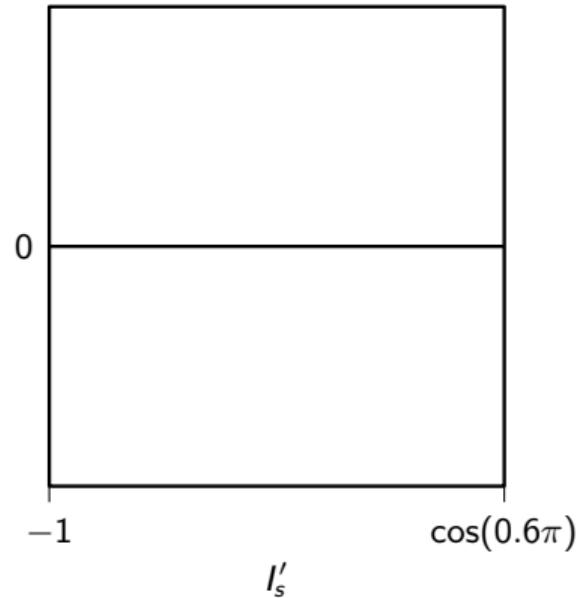
- ▶ $M = 9$ ($L = 4$)
- ▶ $\omega_p = 0.4\pi$
- ▶ $\omega_s = 0.6\pi$

- ▶ we need at least $L + 2 = 6$ alternations
- ▶ 2 alternations always at band edges (otherwise specs not fulfilled)
- ▶ guess the other 4 and apply remez

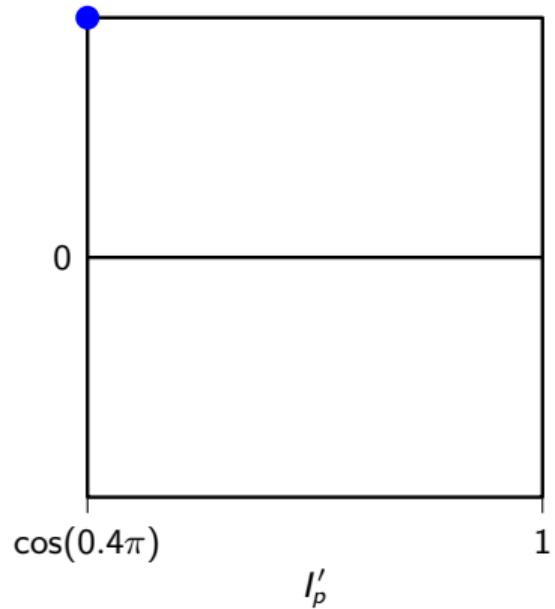
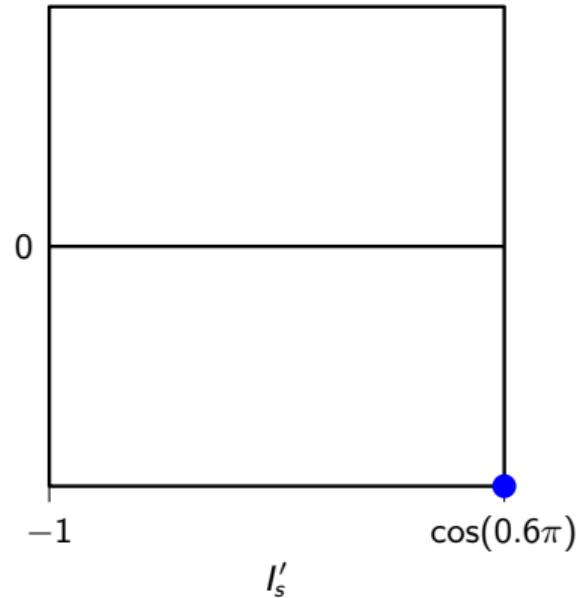
Example

- ▶ $M = 9$ ($L = 4$)
- ▶ $\omega_p = 0.4\pi$
- ▶ $\omega_s = 0.6\pi$
- ▶ we need at least $L + 2 = 6$ alternations
- ▶ 2 alternations always at band edges (otherwise specs not fulfilled)
- ▶ guess the other 4 and apply remez

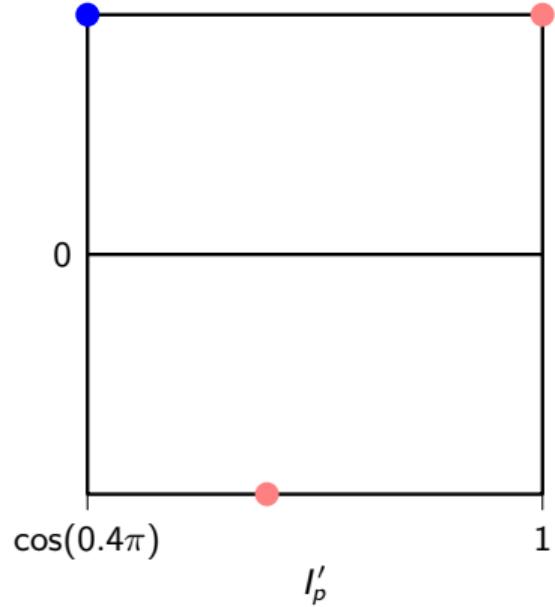
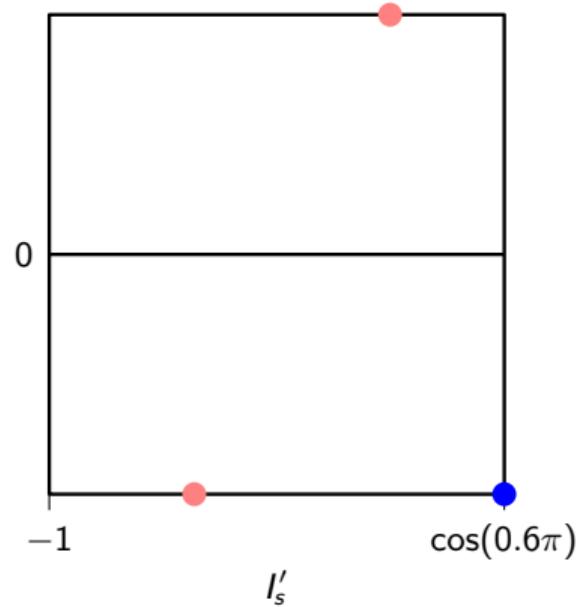
Remez exchange algorithm



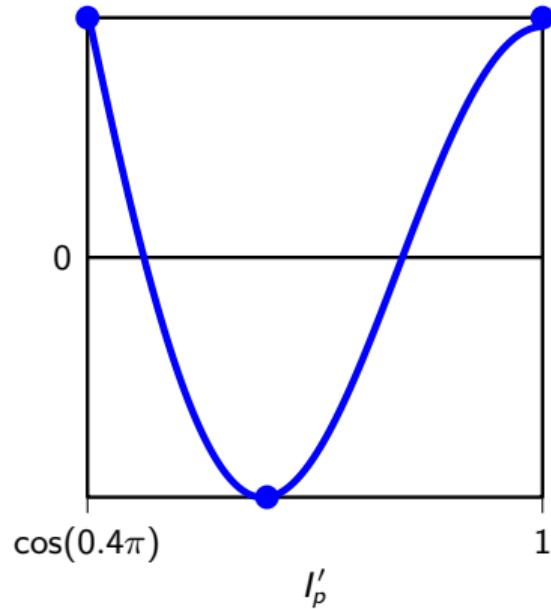
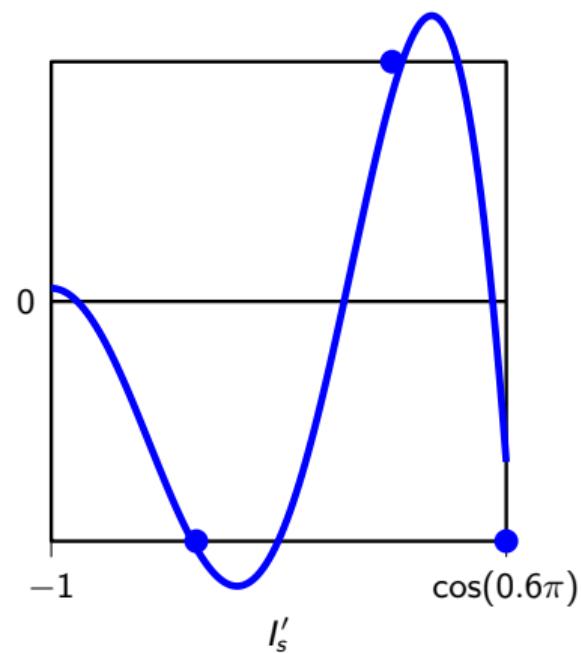
Remez exchange algorithm



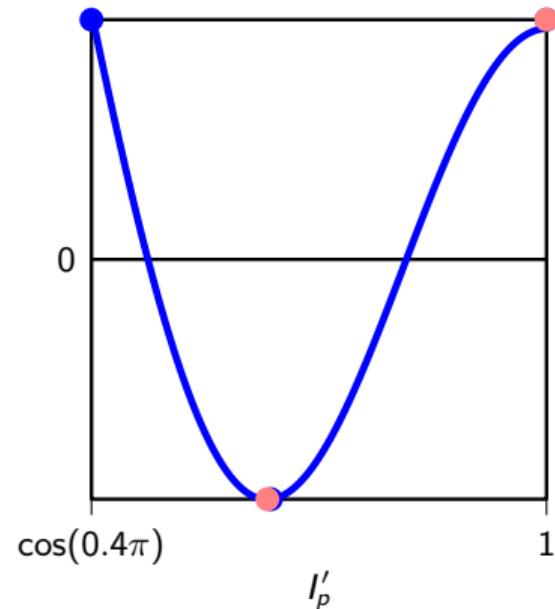
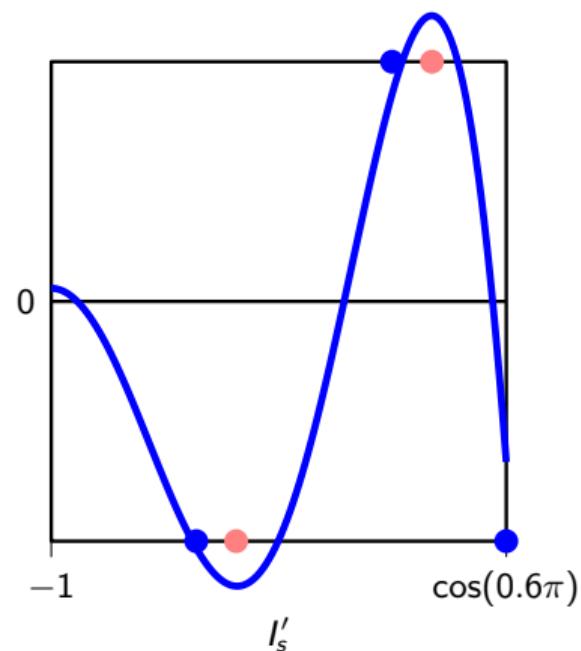
Remez exchange algorithm



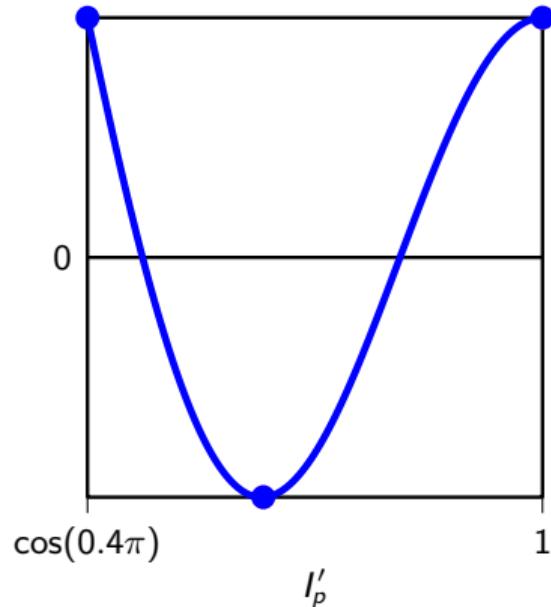
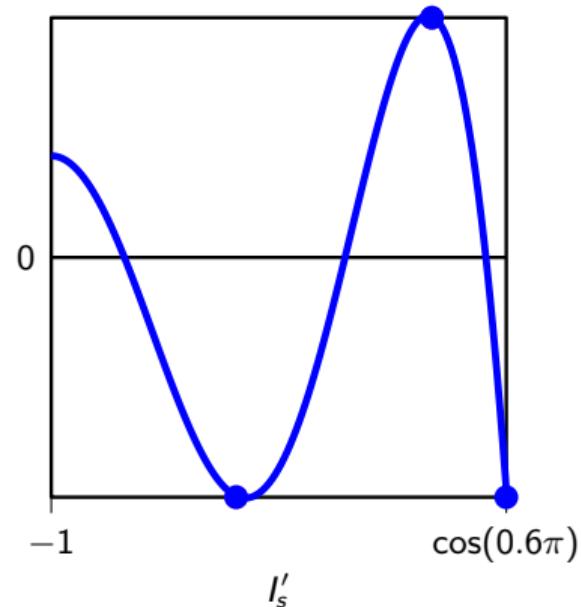
Remez exchange algorithm



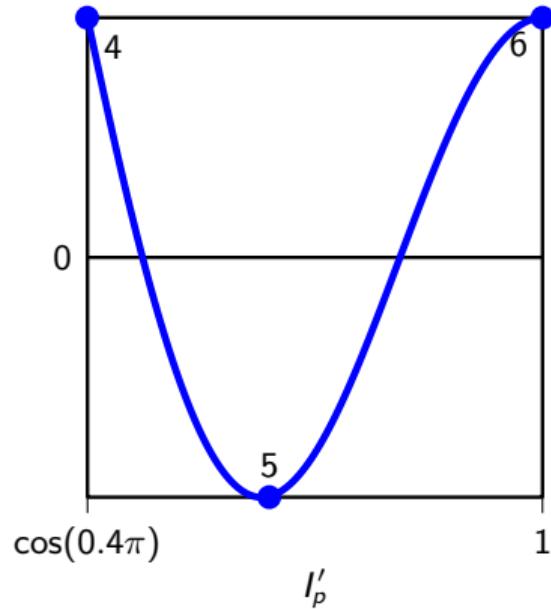
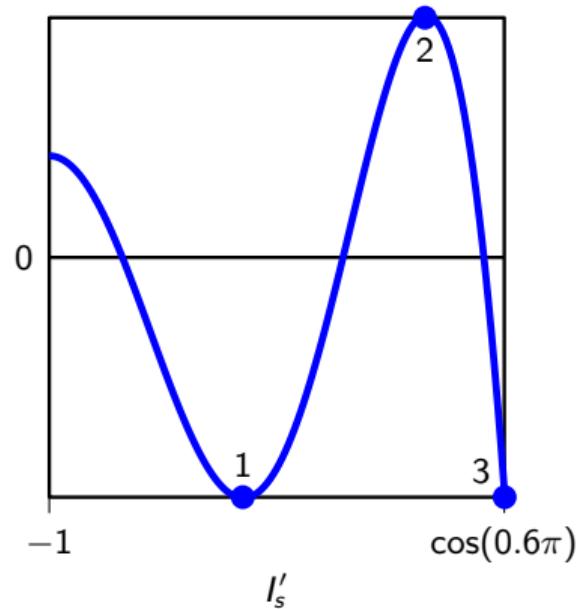
Remez exchange algorithm



Remez exchange algorithm

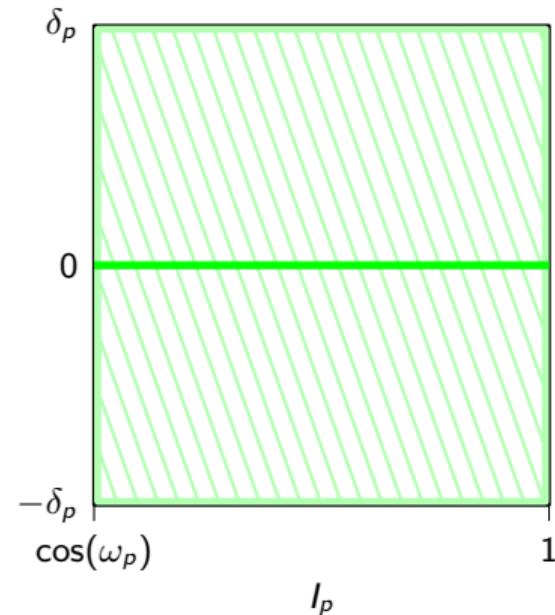
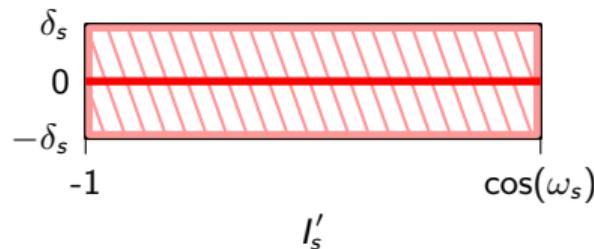


Passband and Stopband Error



Tuning the error

generally, we want to pay more attention to the error in stopband or passband



Goal: fit $E(x)$ within the boxes.

Tuning the error

The Alternation Theorem works also with a weighting function:

$$W(x) = \begin{cases} 1 & \text{for } x \in I'_p \\ \delta_p/\delta_s & \text{for } x \in I'_s \end{cases}$$

The updated minimization problem:

$$\min \max_{x \in I'_p \cup I'_s} \{|W(x)[P(x) - D(x)]|\}$$

Tuning the error

The Alternation Theorem works also with a weighting function:

$$W(x) = \begin{cases} 1 & \text{for } x \in I'_p \\ \delta_p/\delta_s & \text{for } x \in I'_s \end{cases}$$

The updated minimization problem:

$$\min \max_{x \in I'_p \cup I'_s} \{|W(x)[P(x) - D(x)]|\}$$

Parks-McClellan Algorithm; the full recipe for lowpass

User data:

Parks-McClellan Algorithm; the full recipe for lowpass

User data:

- ▶ filter length $M = 2L + 1$

Parks-McClellan Algorithm; the full recipe for lowpass

User data:

- ▶ filter length $M = 2L + 1$
- ▶ ω_p and ω_s

Parks-McClellan Algorithm; the full recipe for lowpass

User data:

- ▶ filter length $M = 2L + 1$
- ▶ ω_p and ω_s
- ▶ stopband -to-passband tolerance ratio δ_s/δ_p

Parks-McClellan Algorithm; the full recipe for lowpass

User data:

- ▶ filter length $M = 2L + 1$
- ▶ ω_p and ω_s
- ▶ stopband -to-passband tolerance ratio δ_s/δ_p

Run Parks-McClellan algorithm; obtain:

- ▶ M filter coefficients

Parks-McClellan Algorithm; the full recipe for lowpass

User data:

- ▶ filter length $M = 2L + 1$
- ▶ ω_p and ω_s
- ▶ stopband -to-passband tolerance ratio δ_s/δ_p

Run Parks-McClellan algorithm; obtain:

- ▶ M filter coefficients
- ▶ stopband and passband tolerances δ_s and δ_p

Parks-McClellan Algorithm; the full recipe for lowpass

User data:

- ▶ filter length $M = 2L + 1$
- ▶ ω_p and ω_s
- ▶ stopband -to-passband tolerance ratio δ_s/δ_p

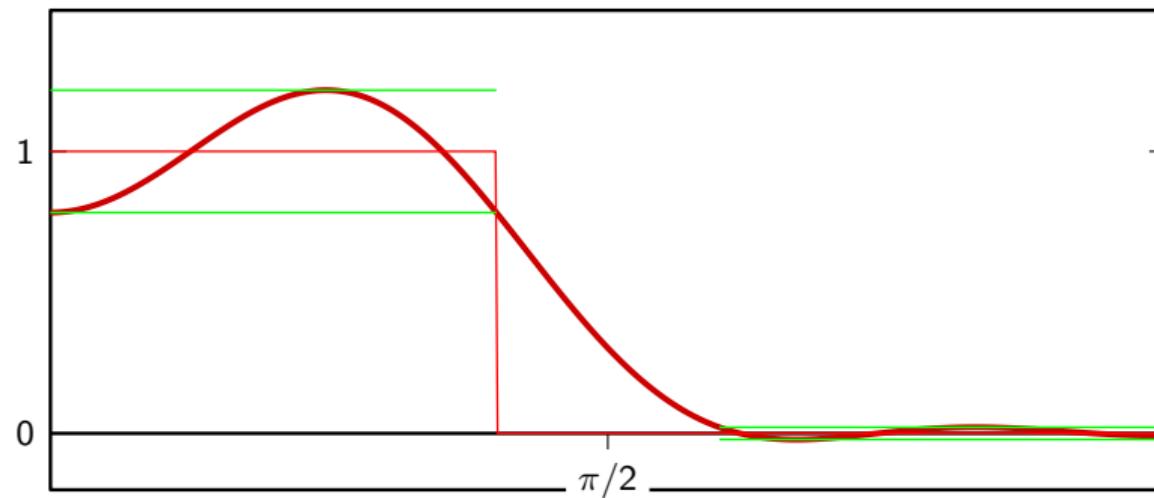
Run Parks-McClellan algorithm; obtain:

- ▶ M filter coefficients
- ▶ stopband and passband tolerances δ_s and δ_p
- ▶ If error too big, increase M and retry.

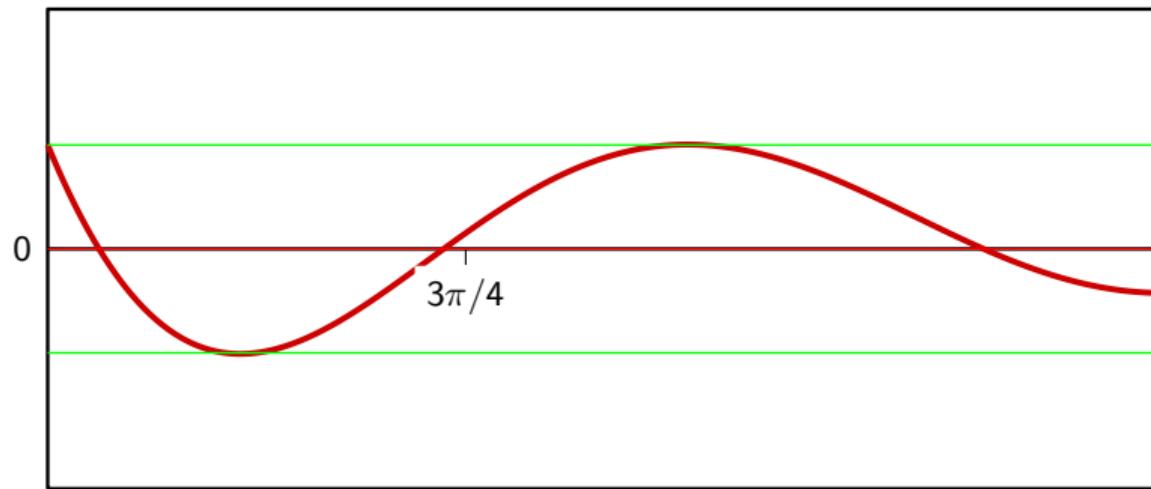
Example revisited

- ▶ $M = 9$ ($L = 4$)
- ▶ $\omega_p = 0.4\pi$
- ▶ $\omega_s = 0.6\pi$
- ▶ $\delta_s/\delta_p = 1/10$

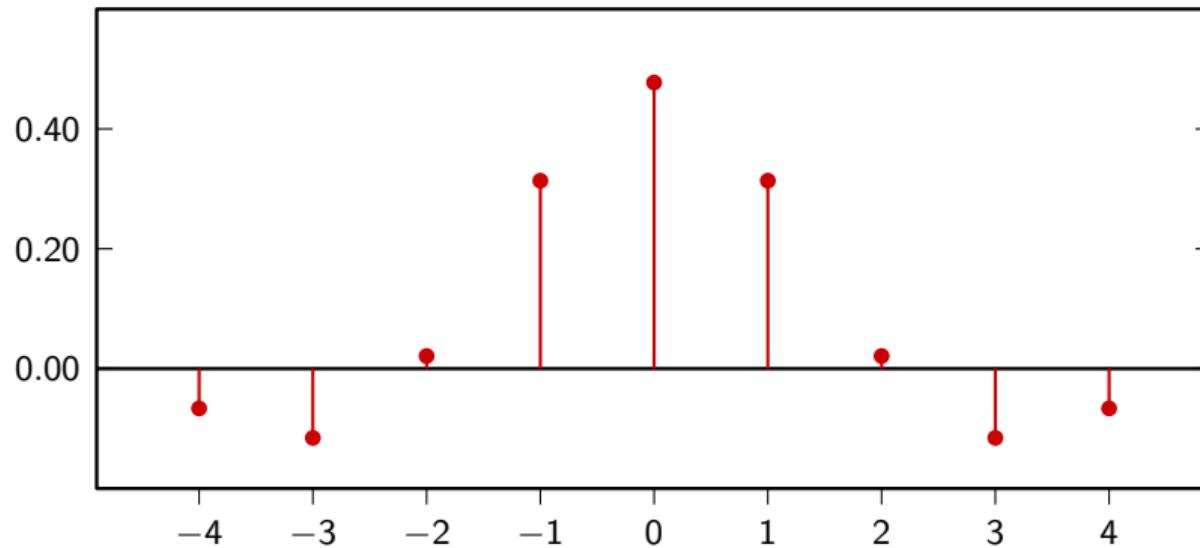
Final Result



Final Result (stopband)



Final Result (Impulse Response)



Minimax lowpass filter (recap)

Magnitude response:

- ▶ equiripple in passband and stopband

Design parameters:

- ▶ order N (number of taps)
- ▶ passband edge ω_p
- ▶ stopband edge ω_s
- ▶ ratio of passband to stopband error δ_p/δ_s

Design test criterion:

- ▶ passband max error
- ▶ stopband max error

Minimax lowpass filter (recap)

Magnitude response:

- ▶ equiripple in passband and stopband

Design parameters:

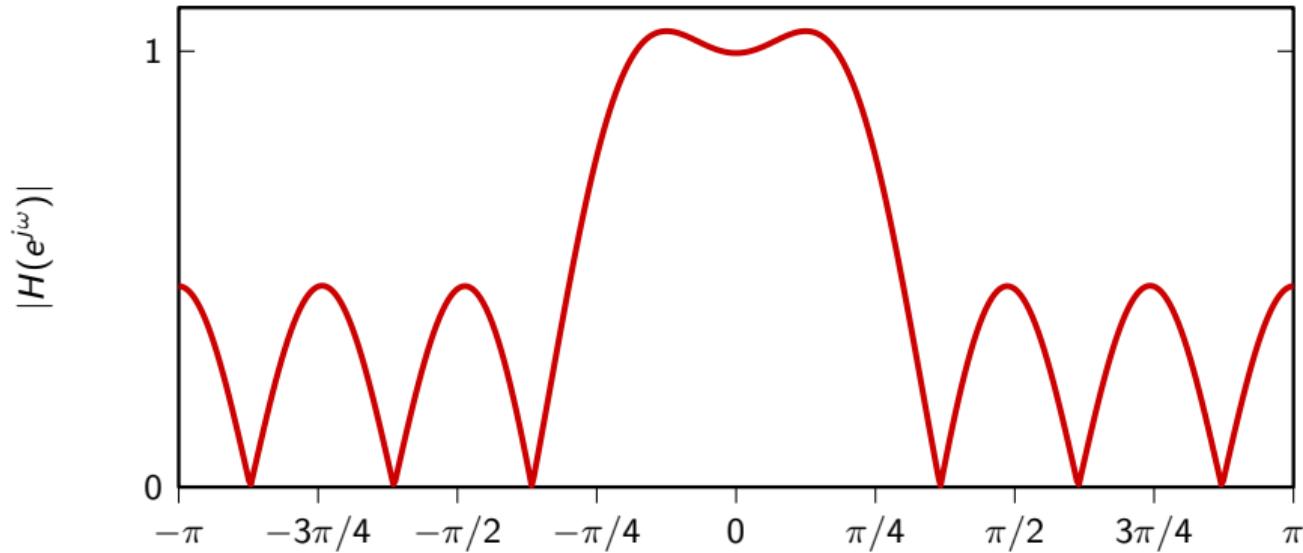
- ▶ order N (number of taps)
- ▶ passband edge ω_p
- ▶ stopband edge ω_s
- ▶ ratio of passband to stopband error δ_p/δ_s

Design test criterion:

- ▶ passband max error
- ▶ stopband max error

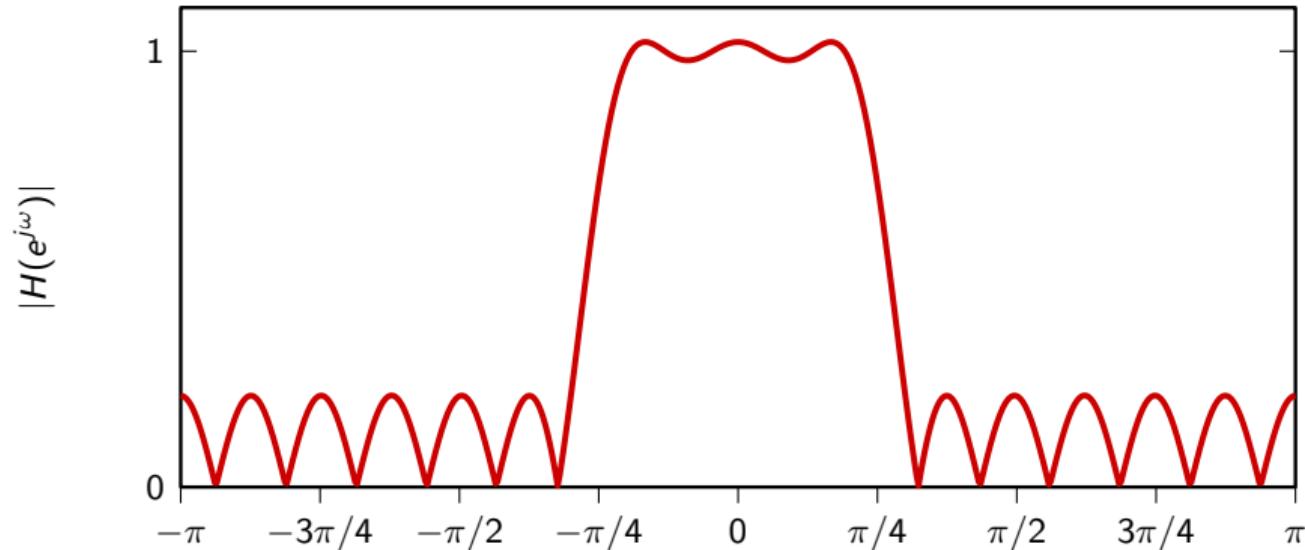
Minimax lowpass example

$$N = 9, \omega_p = 0.2\pi, \omega_s = 0.3\pi, \delta_p/\delta_s = 10$$



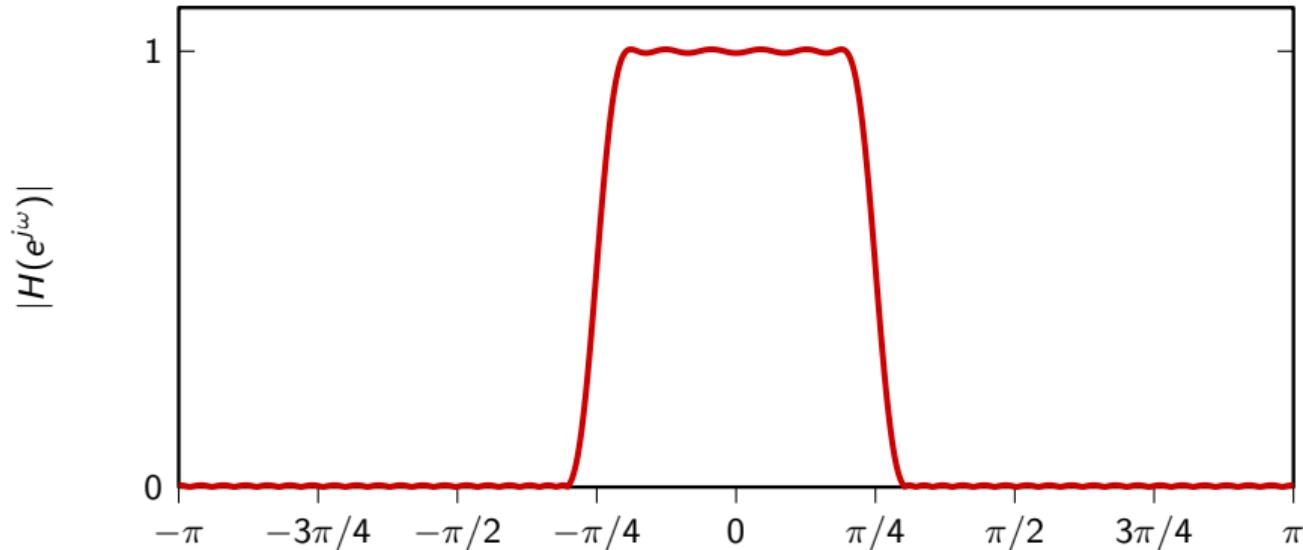
Minimax lowpass example

$$N = 19, \omega_p = 0.2\pi, \omega_s = 0.3\pi, \delta_p/\delta_s = 10$$



Minimax lowpass example

$$N = 51, \omega_p = 0.2\pi, \omega_s = 0.3\pi, \delta_p/\delta_s = 1$$



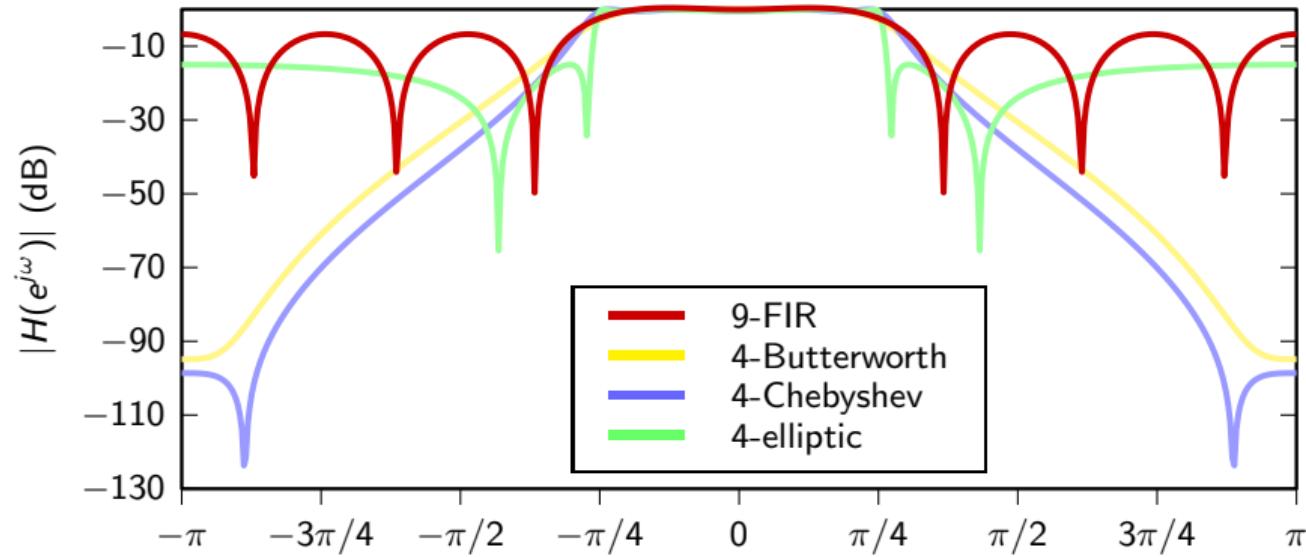
Magnitude response in decibels

- ▶ filter max passband magnitude G
- ▶ filter attenuation expressed in decibels as:

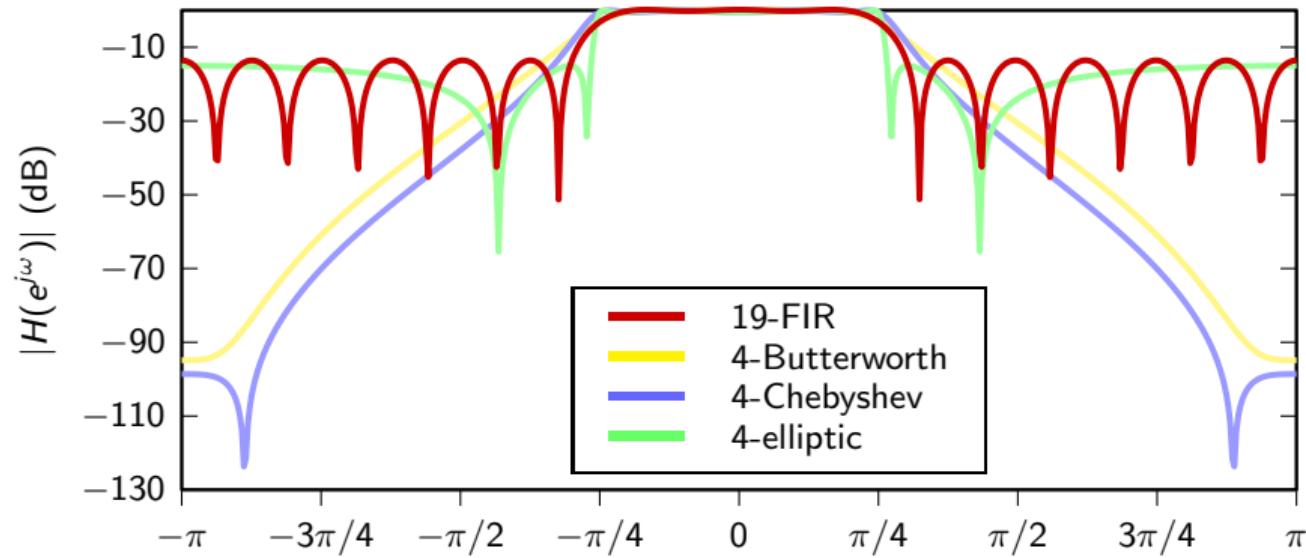
$$A_{\text{dB}} = 20 \log_{10}(|H(e^{j\omega})|/G)$$

- ▶ useful to compare attenuations between filters

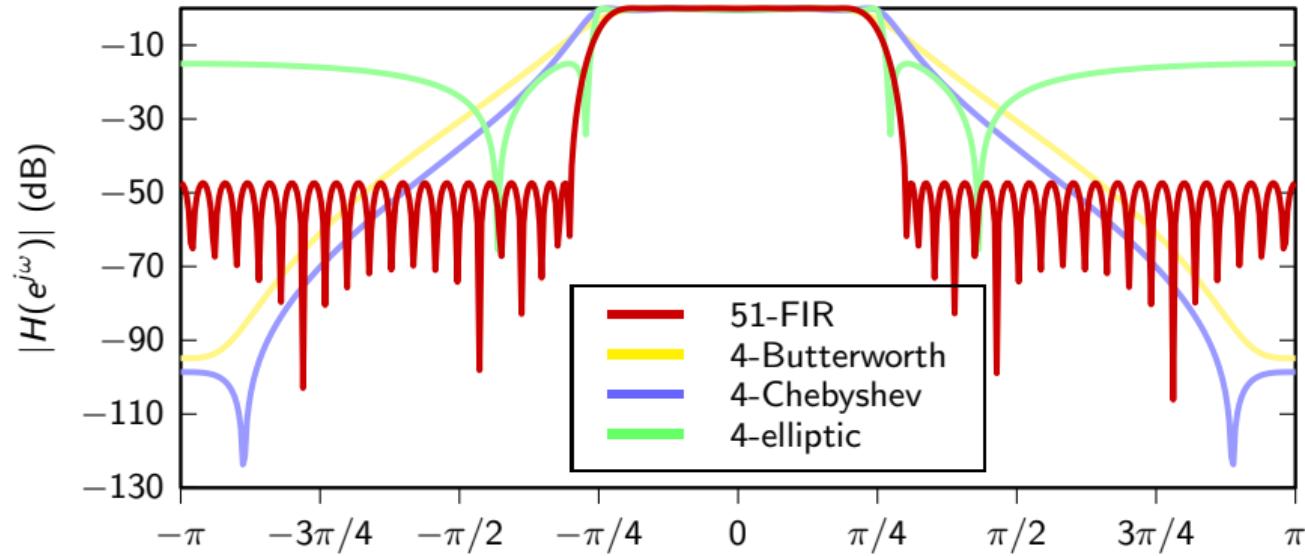
Lowpass comparison, $\omega_c = \pi/4$, log scale



Lowpass comparison, $\omega_c = \pi/4$, log scale



Lowpass comparison, $\omega_c = \pi/4$, log scale



Life beyond lowpass

The IIR and FIR methods we just described can be used to design more general filter types than lowpass, with only minor modifications

- ▶ IIR bandpass and highpass can be obtain by modulating the lowpass response
- ▶ optimal FIR bandpass and highpass can be designed by the Parks-McClellan algorithm
- ▶ optimal FIR can also be designed with piecewise linear magnitude response
- ▶ the literature on filter design is vast: this is just the tip of the iceberg!

Life beyond lowpass

The IIR and FIR methods we just described can be used to design more general filter types than lowpass, with only minor modifications

- ▶ IIR bandpass and highpass can be obtain by modulating the lowpass response
- ▶ optimal FIR bandpass and highpass can be designed by the Parks-McClellan algorithm
- ▶ optimal FIR can also be designed with piecewise linear magnitude response
- ▶ the literature on filter design is vast: this is just the tip of the iceberg!

Life beyond lowpass

The IIR and FIR methods we just described can be used to design more general filter types than lowpass, with only minor modifications

- ▶ IIR bandpass and highpass can be obtained by modulating the lowpass response
- ▶ optimal FIR bandpass and highpass can be designed by the Parks-McClellan algorithm
- ▶ optimal FIR can also be designed with piecewise linear magnitude response
- ▶ the literature on filter design is vast: this is just the tip of the iceberg!

Life beyond lowpass

The IIR and FIR methods we just described can be used to design more general filter types than lowpass, with only minor modifications

- ▶ IIR bandpass and highpass can be obtain by modulating the lowpass response
- ▶ optimal FIR bandpass and highpass can be designed by the Parks-McClellan algorithm
- ▶ optimal FIR can also be designed with piecewise linear magnitude response
- ▶ the literature on filter design is vast: this is just the tip of the iceberg!

Play with the demo!

Play with the interactive minimax filter design demo

<https://github.com/prandoni/COM303/>

COM303: Digital Signal Processing

Lecture 14: Real-time signal processing

Summary:

- ▶ I/O and DMA
 - ▶ multiple buffering
 - ▶ implementation framework
 - ▶ some guitar effects

Summary:

- ▶ I/O and DMA
- ▶ multiple buffering
- ▶ implementation framework
- ▶ some guitar effects

Summary:

- ▶ I/O and DMA
- ▶ multiple buffering
- ▶ implementation framework
- ▶ some guitar effects

Summary:

- ▶ I/O and DMA
- ▶ multiple buffering
- ▶ implementation framework
- ▶ some guitar effects

Real-time processing

Everything works in sync with a *system clock* of period T_s :

- ▶ “record” a value $x_i[n]$
- ▶ process the value in a causal filter
- ▶ “play” the output $x_o[n]$

everything needs to happen in at most T_s seconds!

Real-time processing

Everything works in sync with a *system clock* of period T_s :

- ▶ “record” a value $x_i[n]$
- ▶ process the value in a causal filter
- ▶ “play” the output $x_o[n]$

everything needs to happen in at most T_s seconds!

Real-time processing

Everything works in sync with a *system clock* of period T_s :

- ▶ “record” a value $x_i[n]$
- ▶ process the value in a causal filter
- ▶ “play” the output $x_o[n]$

everything needs to happen in at most T_s seconds!

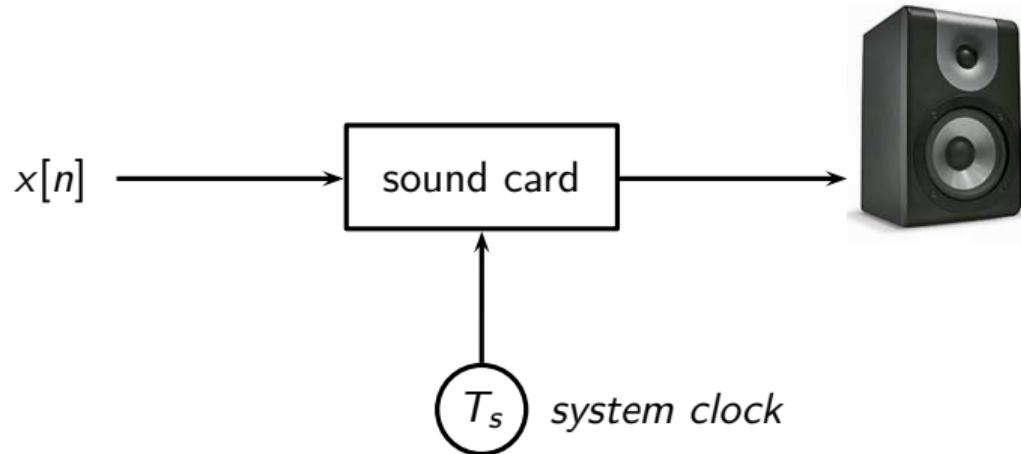
Real-time processing

Everything works in sync with a *system clock* of period T_s :

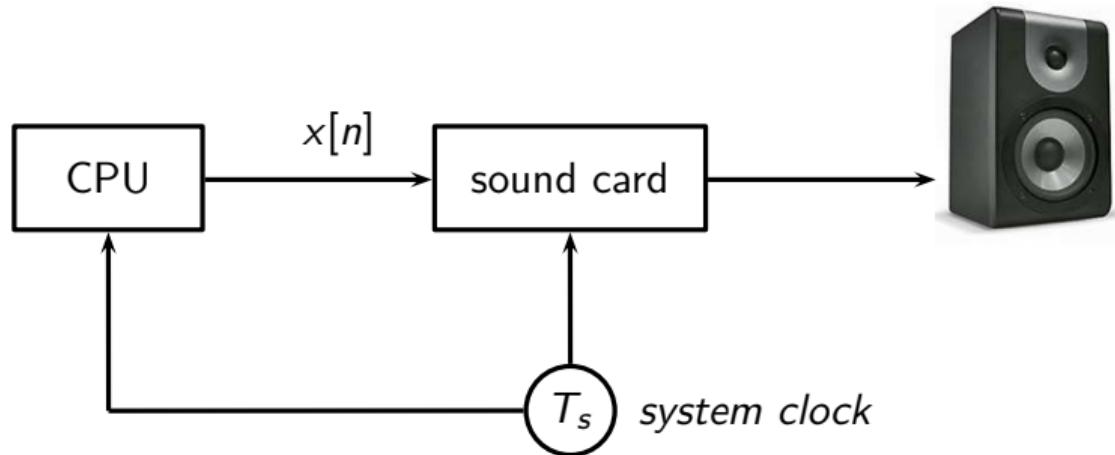
- ▶ “record” a value $x_i[n]$
- ▶ process the value in a causal filter
- ▶ “play” the output $x_o[n]$

everything needs to happen in at most T_s seconds!

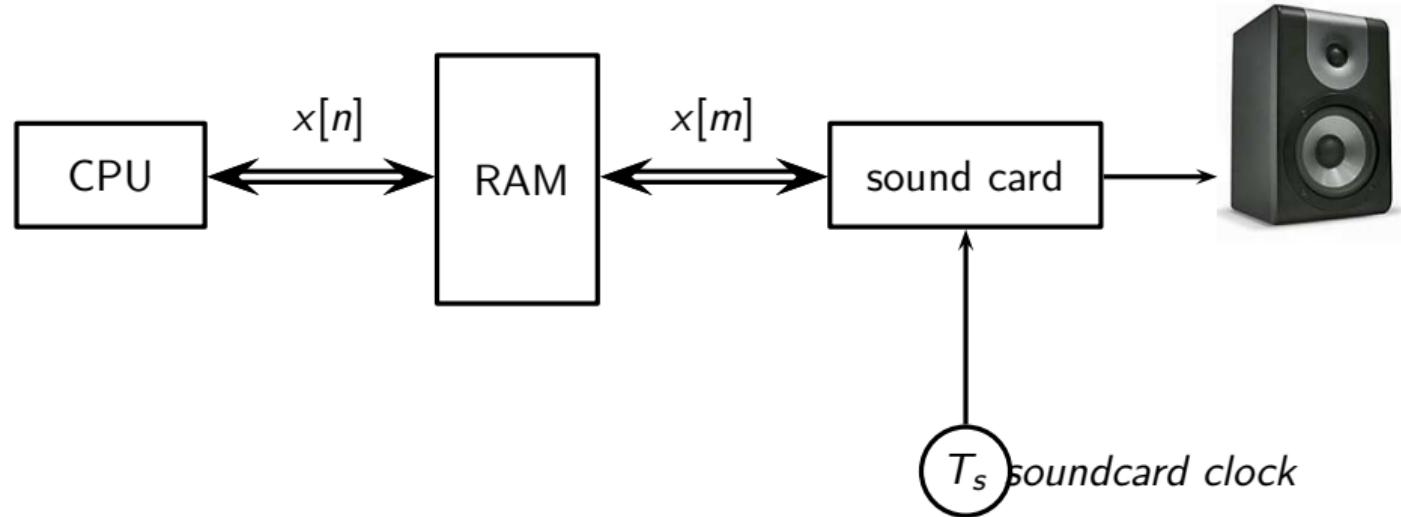
Playing a sound



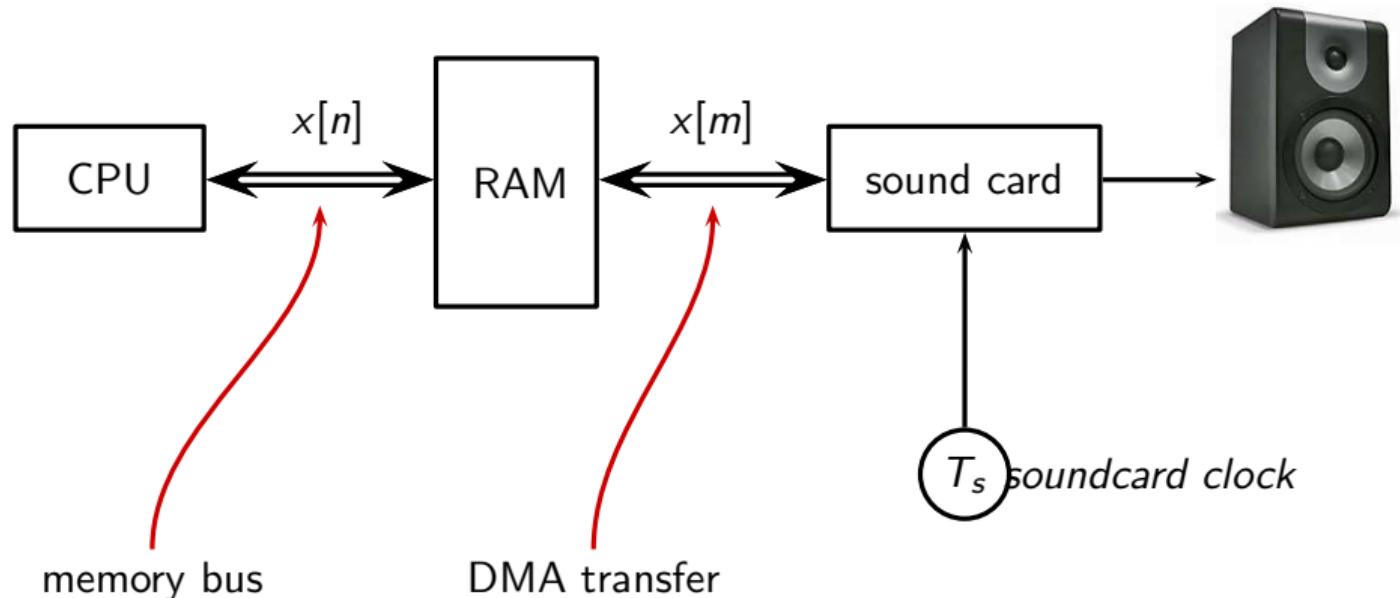
On dedicated hardware...



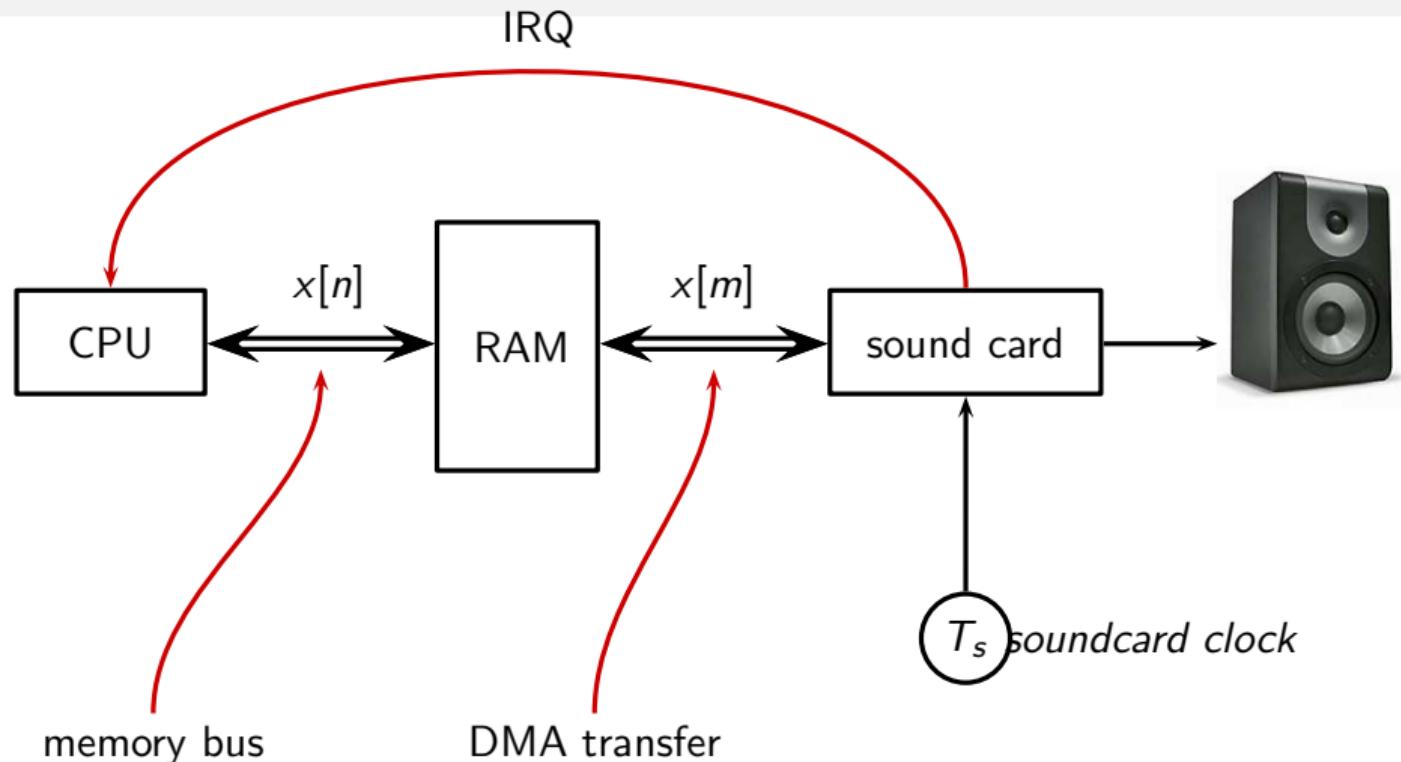
On a PC...



On a PC...



On a PC...



Buffering

- ▶ interrupt for each sample would be too much overhead
- ▶ soundcard consumes sample in buffers
- ▶ soundcard notifies when buffer used up
- ▶ CPU can fill a buffer in less time than soundcard can empty it

buffering introduces delay!

Buffering

- ▶ interrupt for each sample would be too much overhead
- ▶ soundcard consumes sample in buffers
- ▶ soundcard notifies when buffer used up
- ▶ CPU can fill a buffer in less time than soundcard can empty it

buffering introduces delay!

Buffering

- ▶ interrupt for each sample would be too much overhead
- ▶ soundcard consumes sample in buffers
- ▶ soundcard notifies when buffer used up
- ▶ CPU can fill a buffer in less time than soundcard can empty it

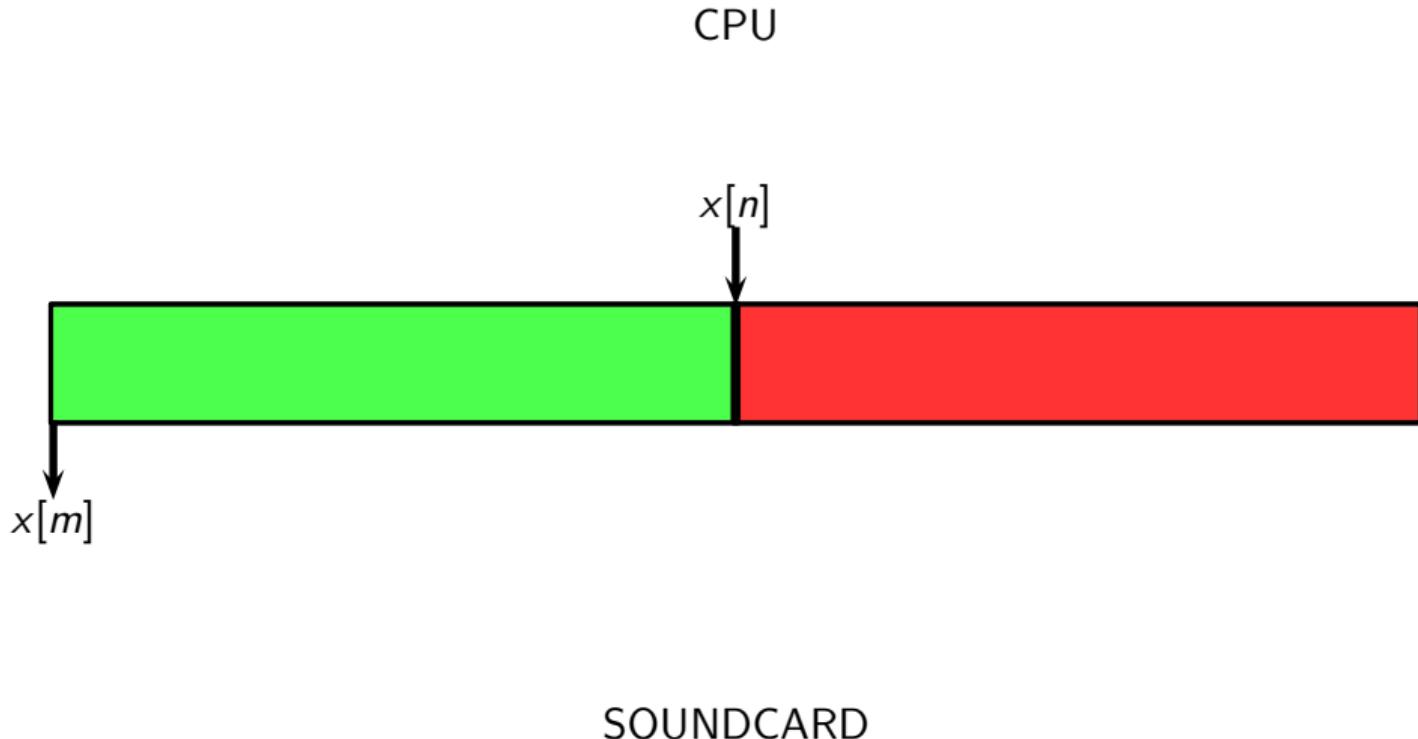
buffering introduces delay!

Buffering

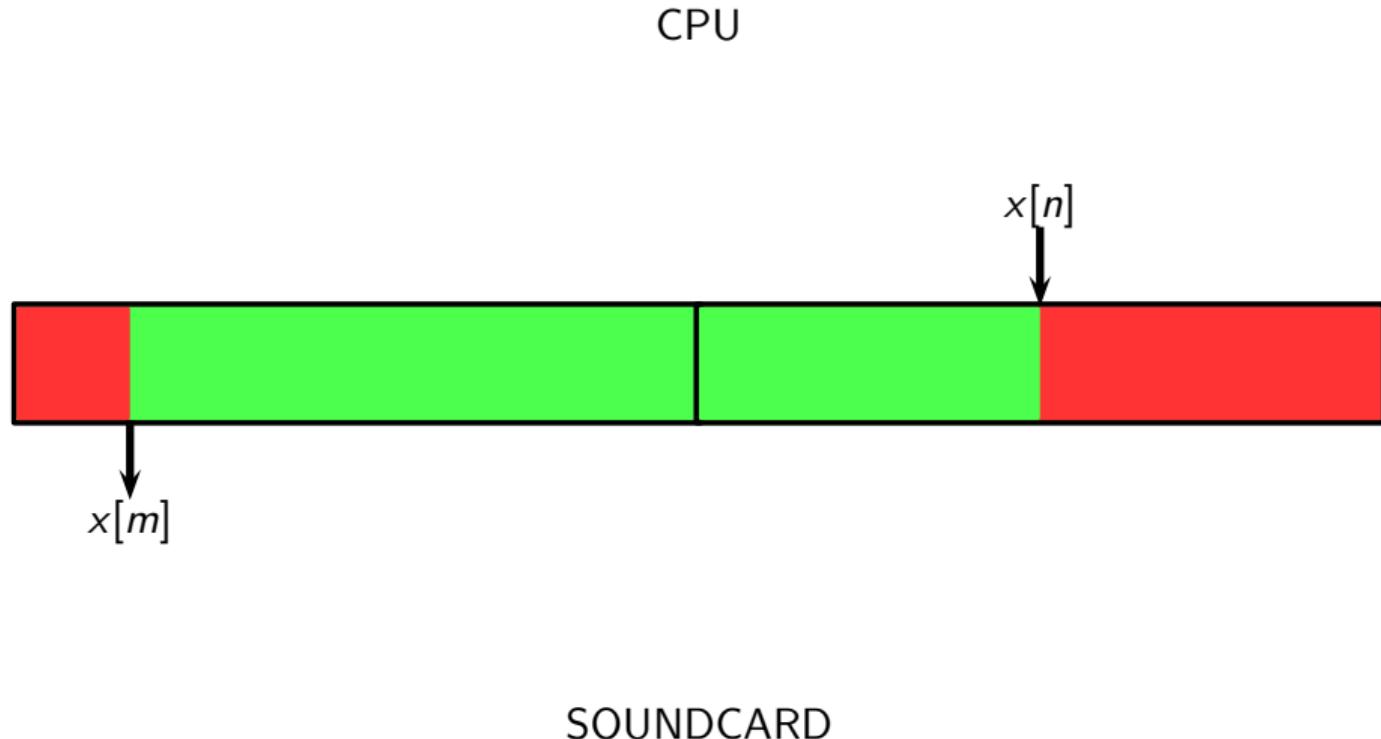
- ▶ interrupt for each sample would be too much overhead
- ▶ soundcard consumes sample in buffers
- ▶ soundcard notifies when buffer used up
- ▶ CPU can fill a buffer in less time than soundcard can empty it

buffering introduces delay!

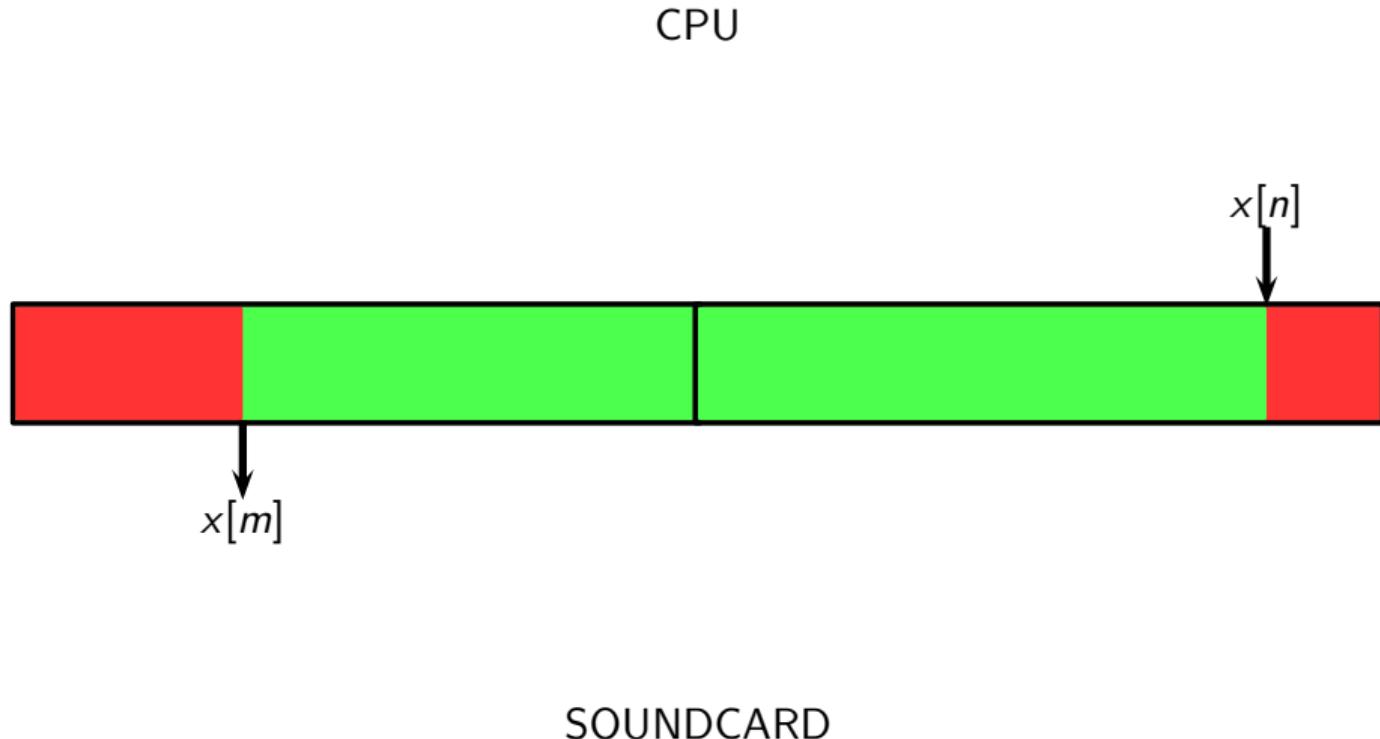
Example: double buffering (output)



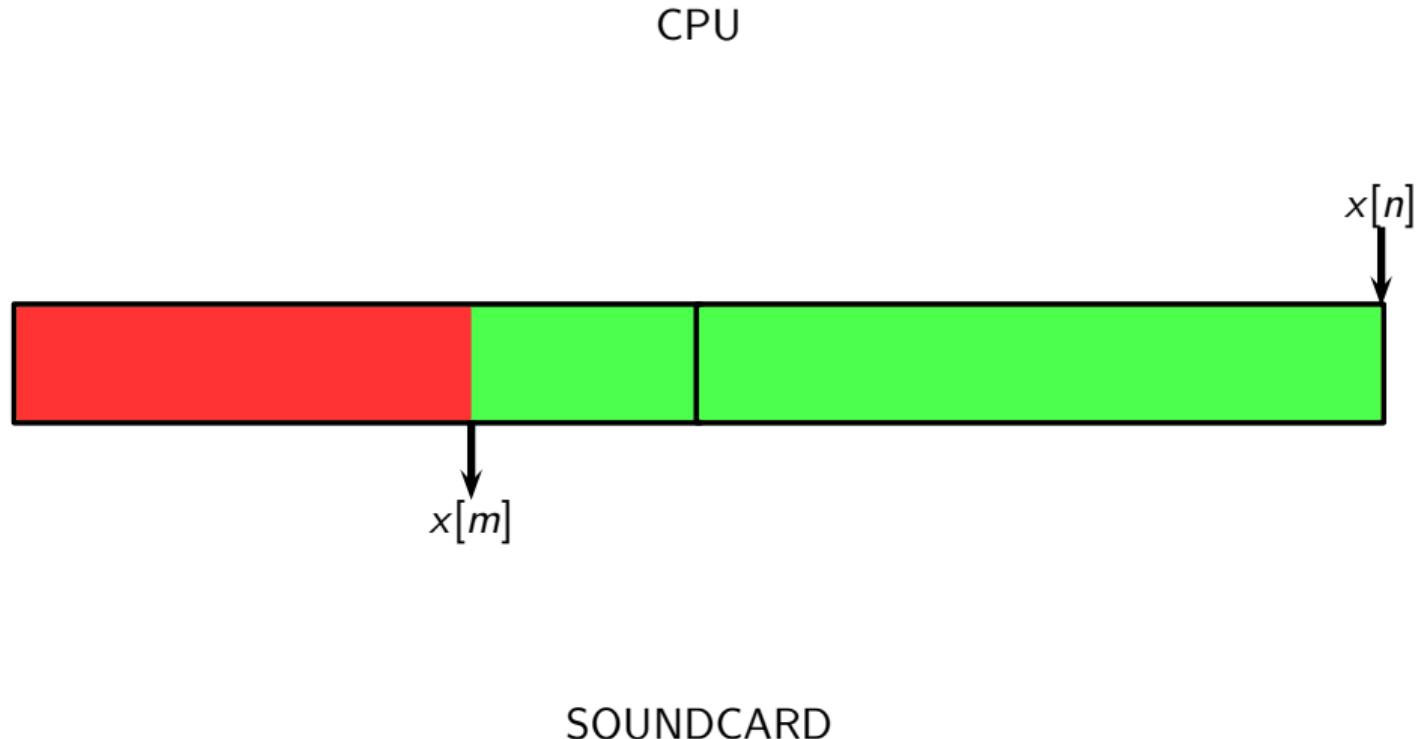
Example: double buffering (output)



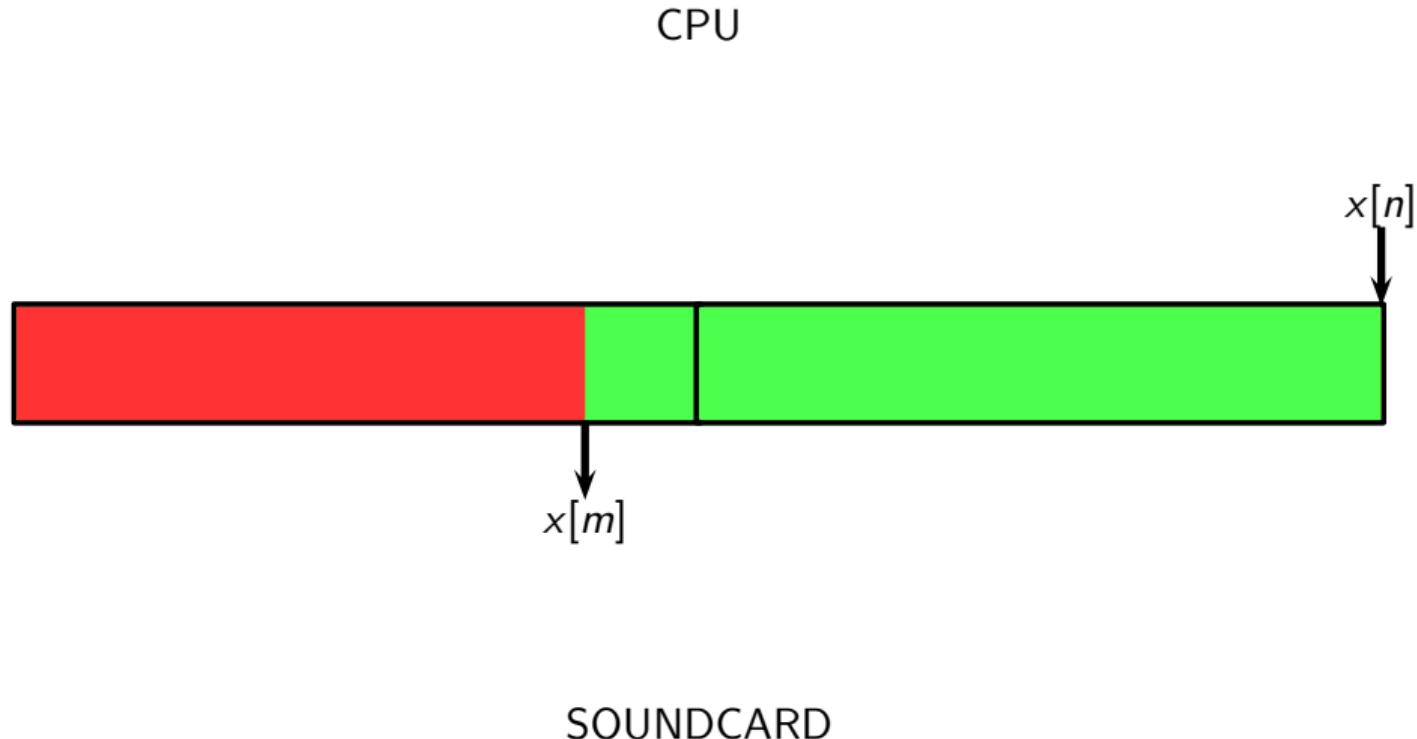
Example: double buffering (output)



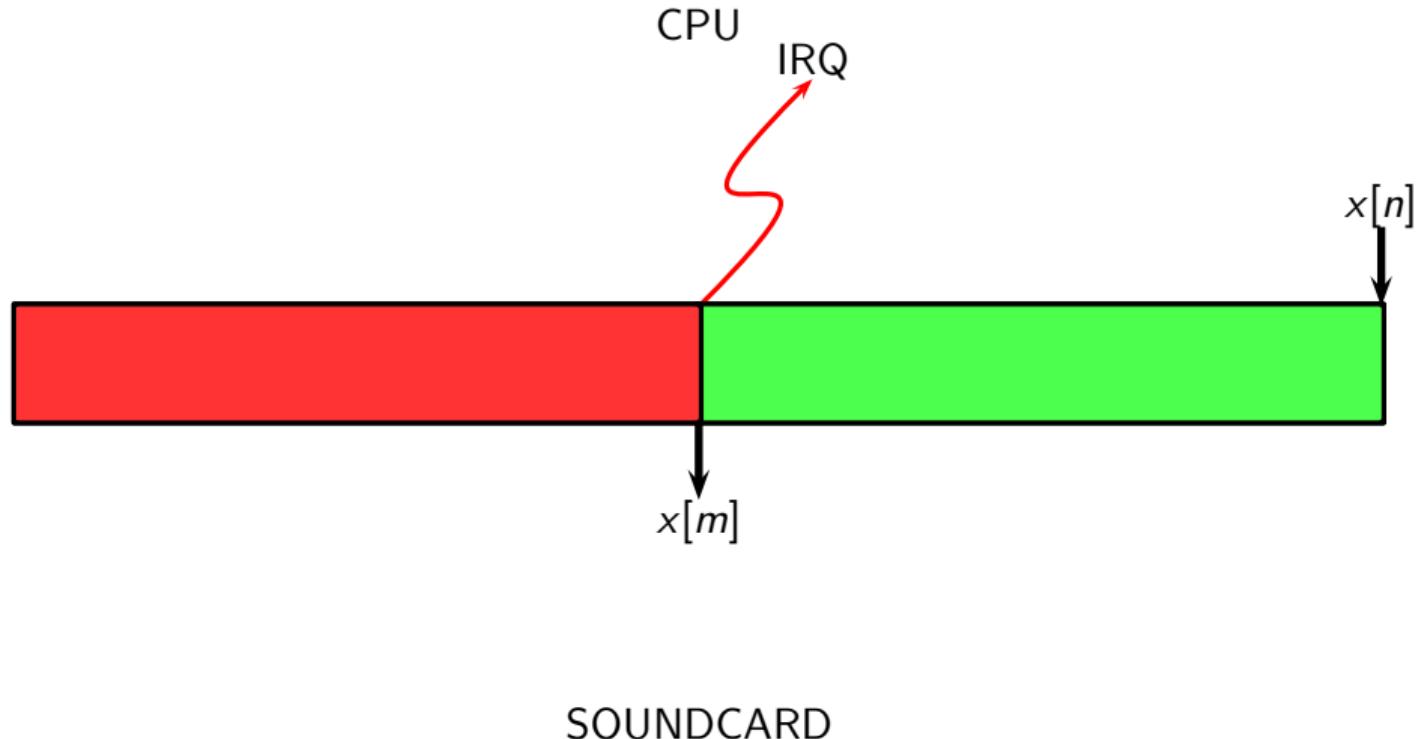
Example: double buffering (output)



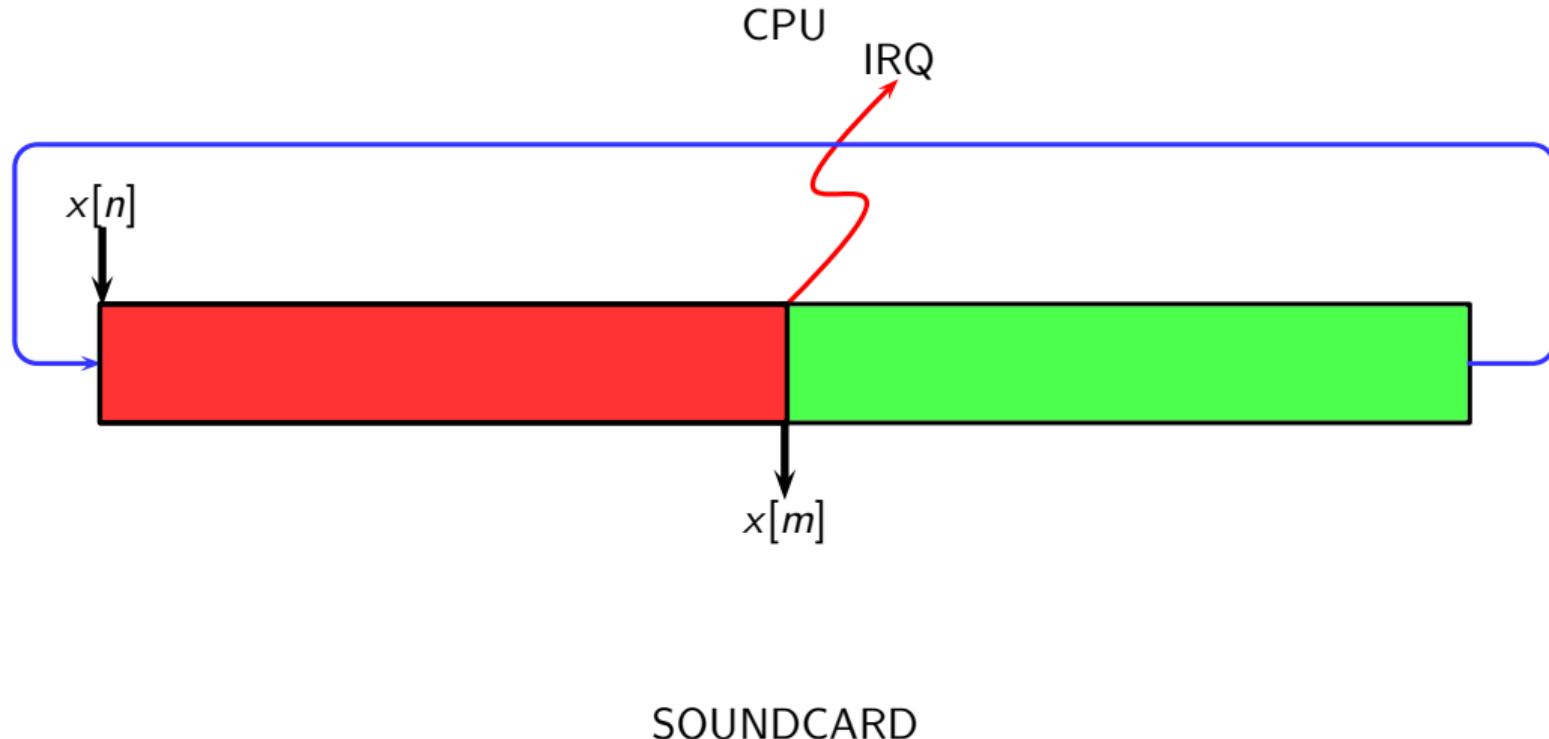
Example: double buffering (output)



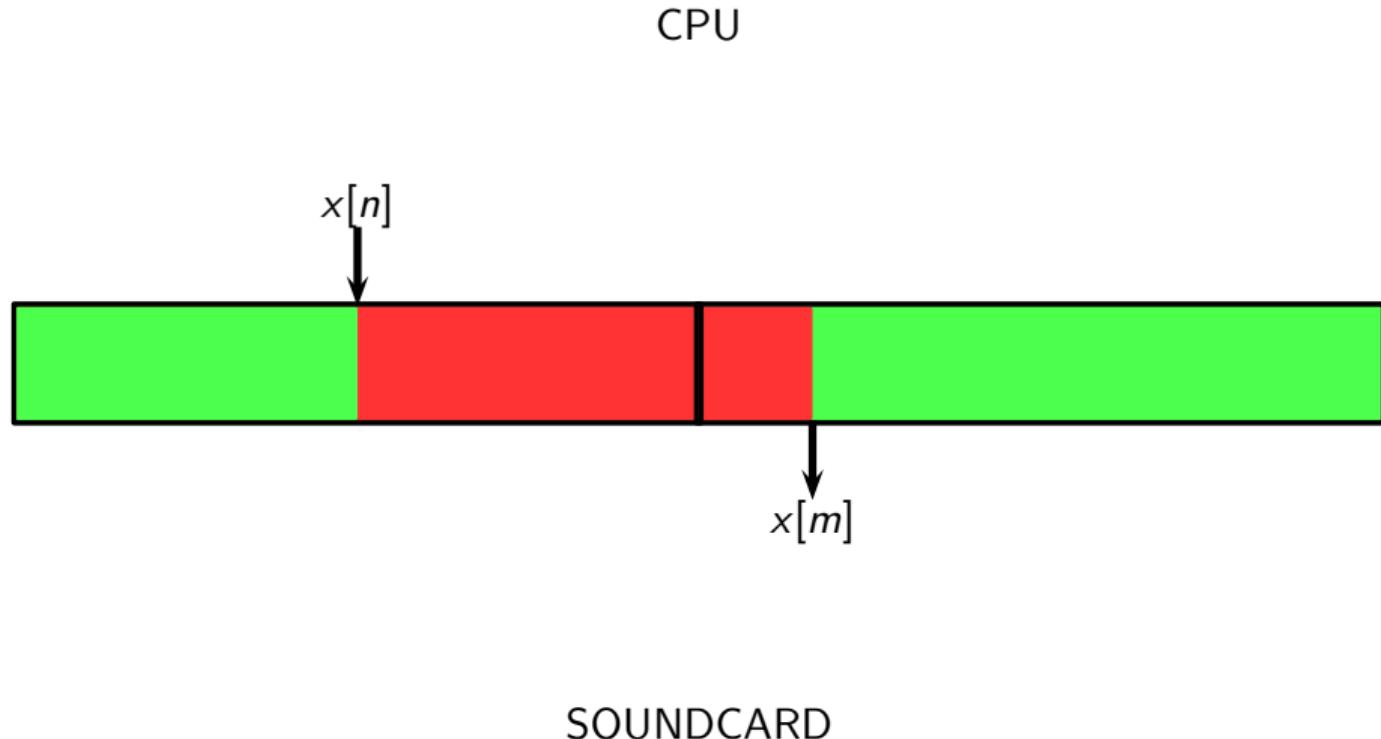
Example: double buffering (output)



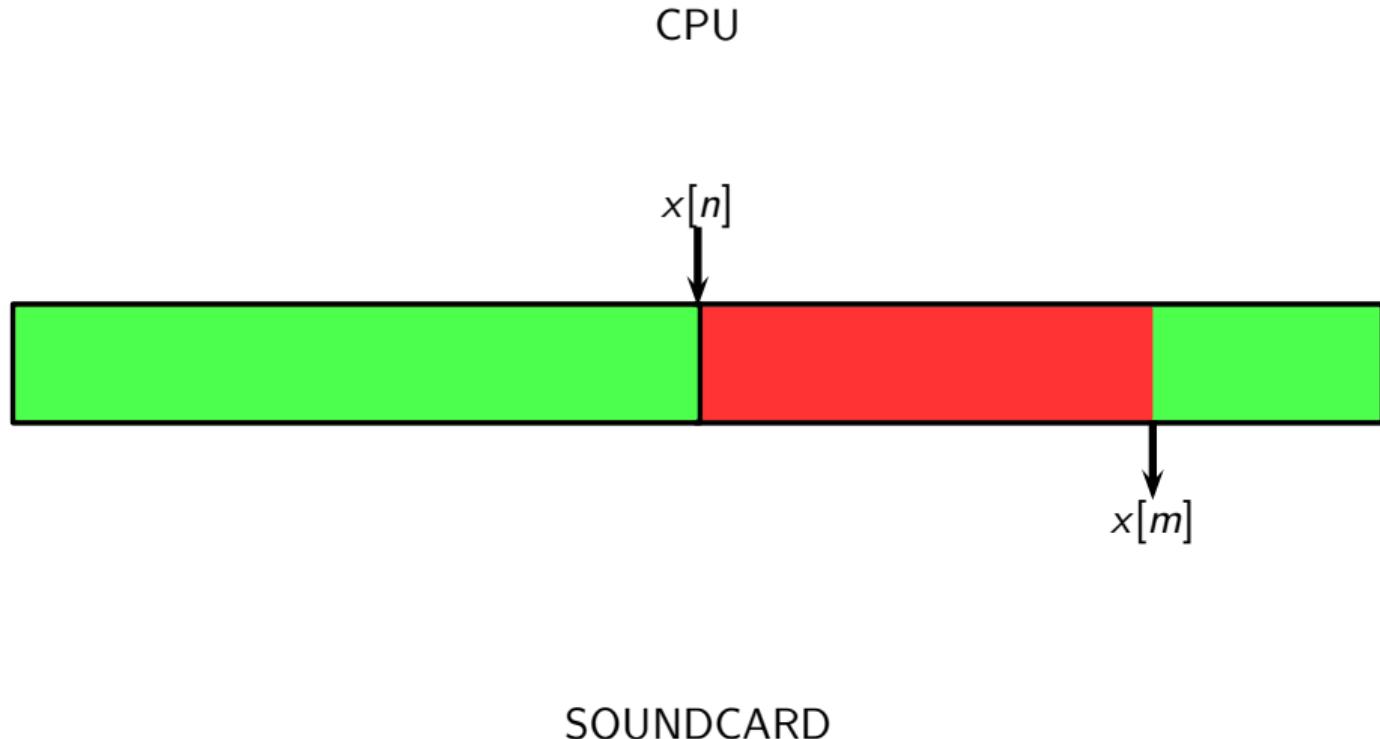
Example: double buffering (output)



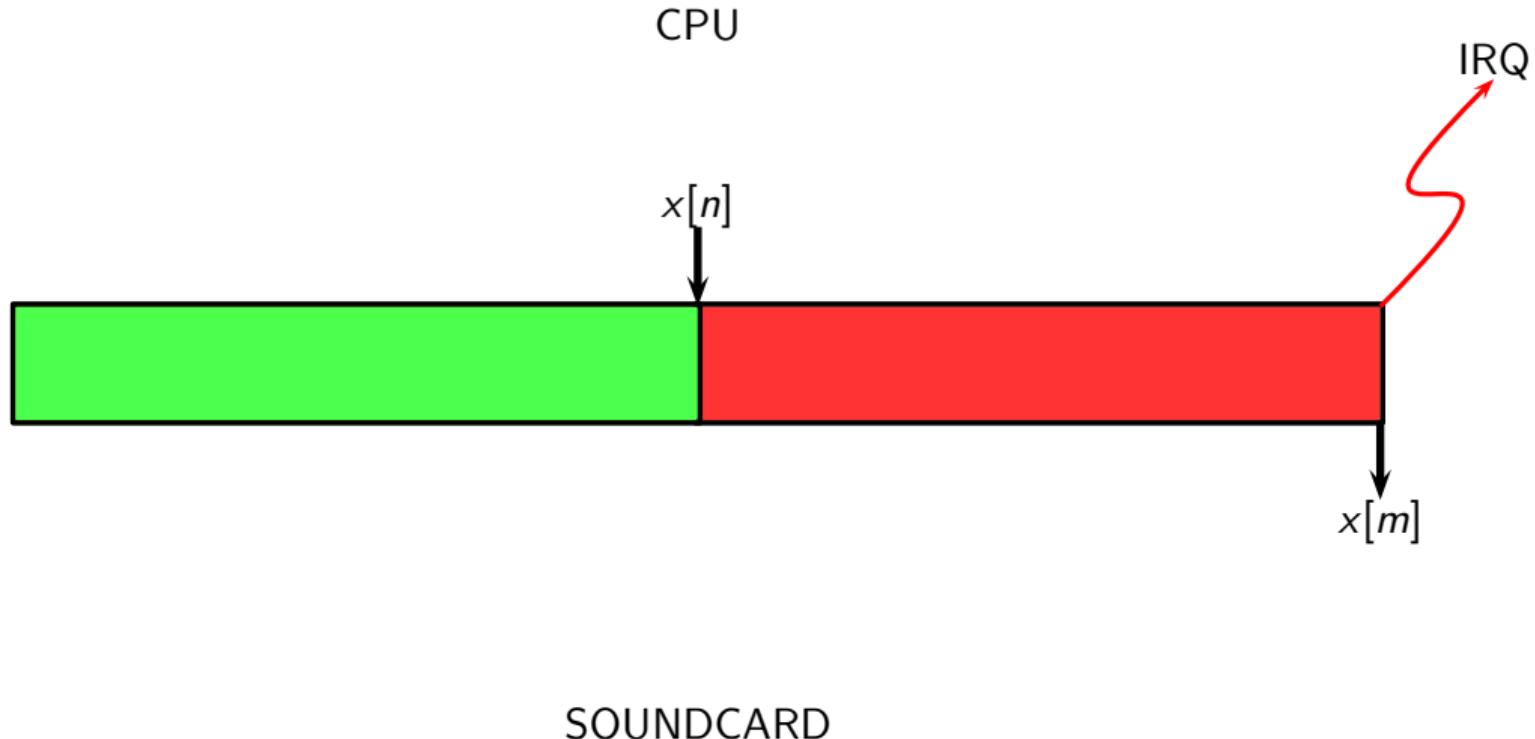
Example: double buffering (output)



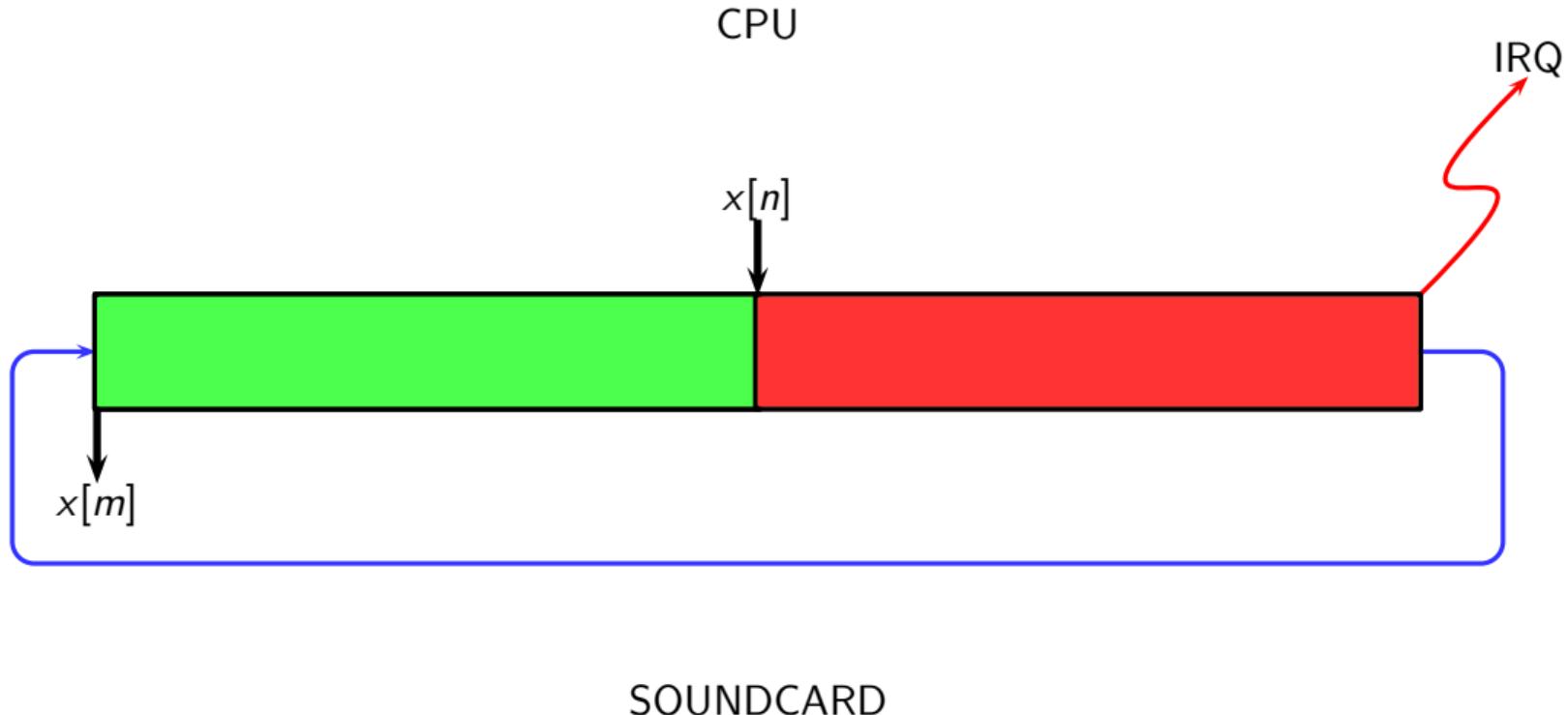
Example: double buffering (output)



Example: double buffering (output)



Example: double buffering (output)



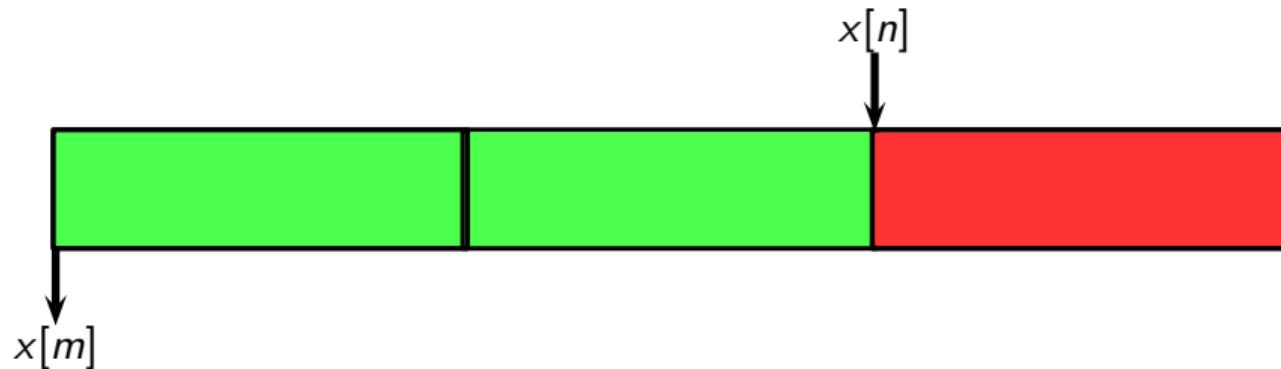
Example: double buffering

- ▶ double buffering introduces a delay $d = T_s \times \frac{L}{2}$ seconds
- ▶ if CPU doesn't fill the buffer fast enough: **underflow**

Example: double buffering

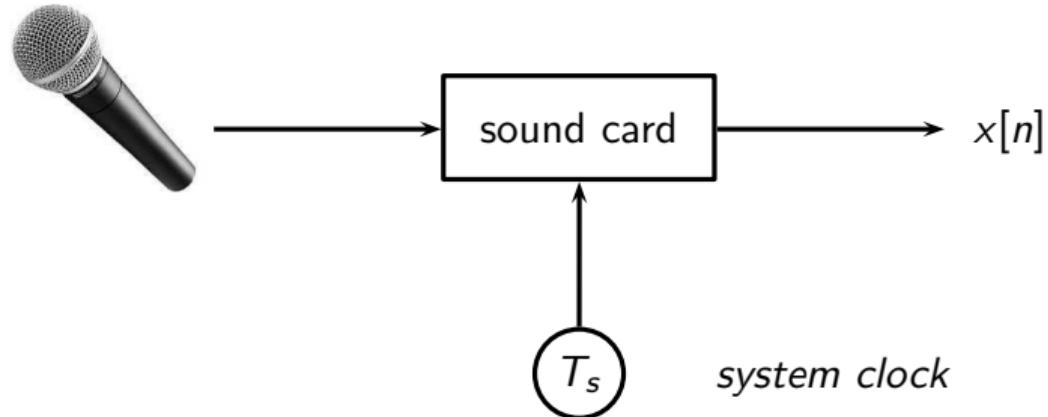
- ▶ double buffering introduces a delay $d = T_s \times \frac{L}{2}$ seconds
- ▶ if CPU doesn't fill the buffer fast enough: **underflow**

Multiple buffering

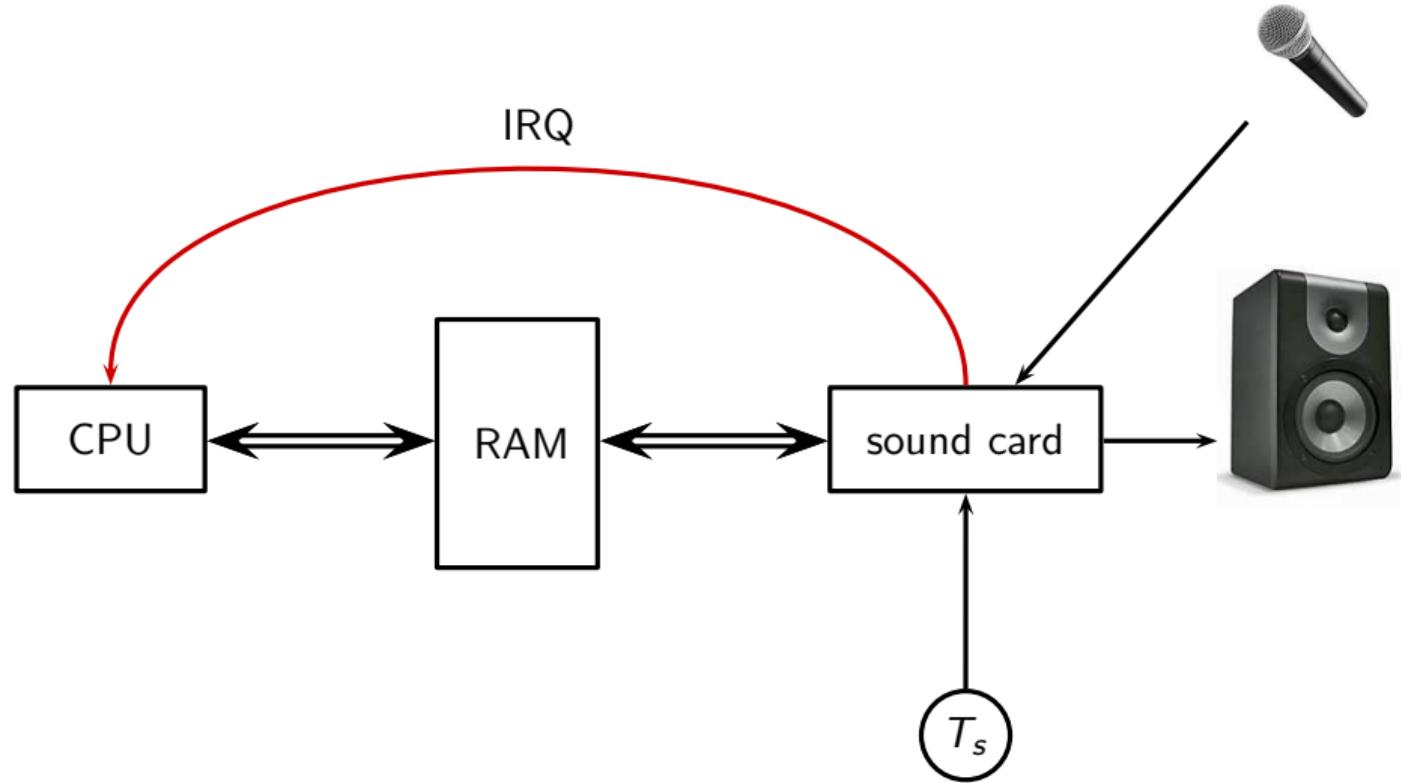


- ▶ call the CPU more often (balance load)
- ▶ keep reasonable underflow protection

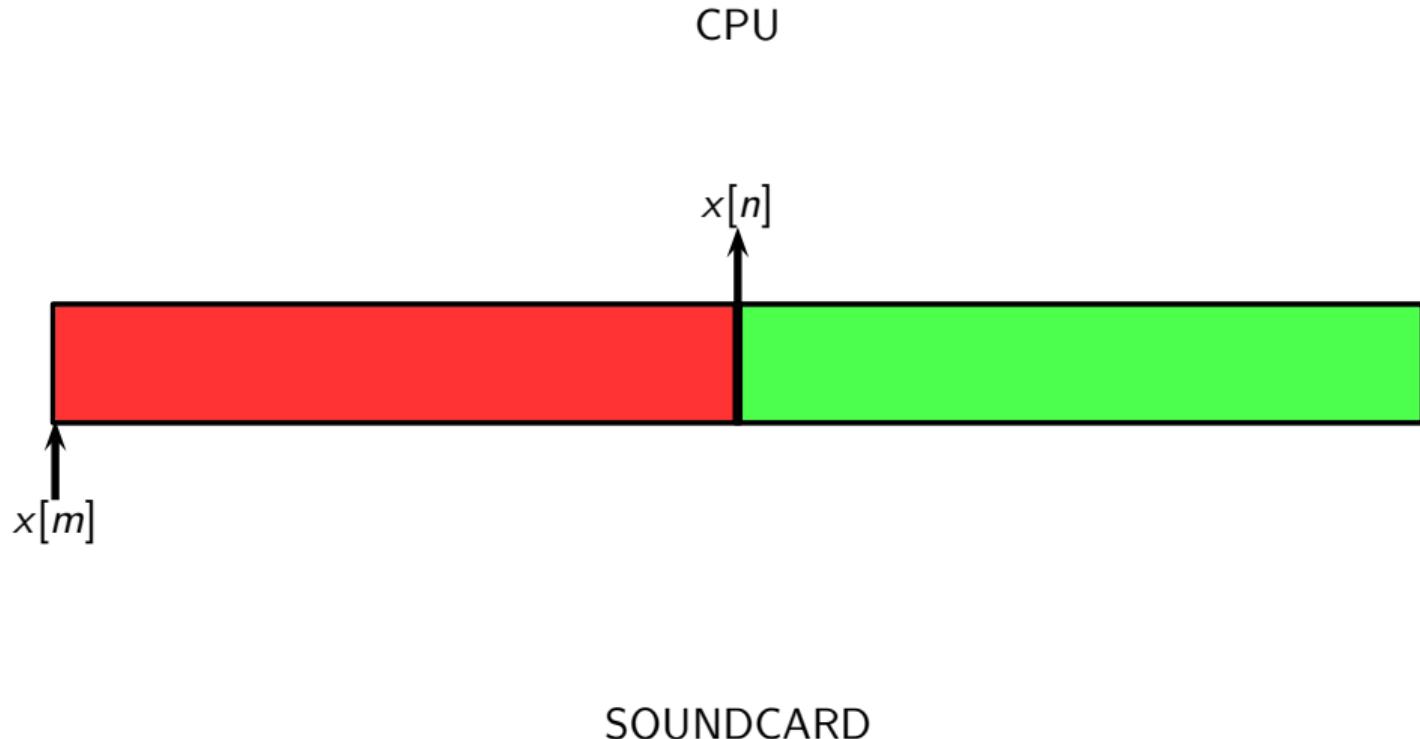
What about the input?



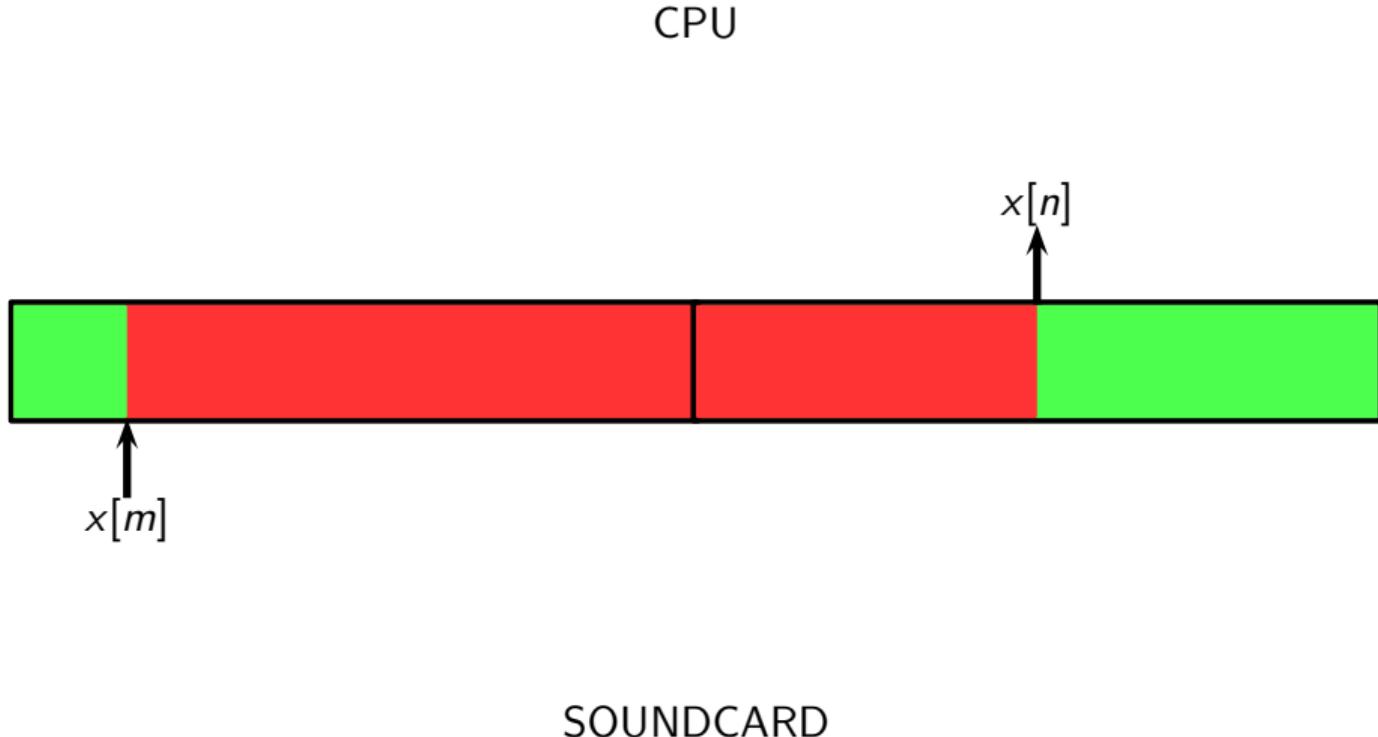
On a PC...



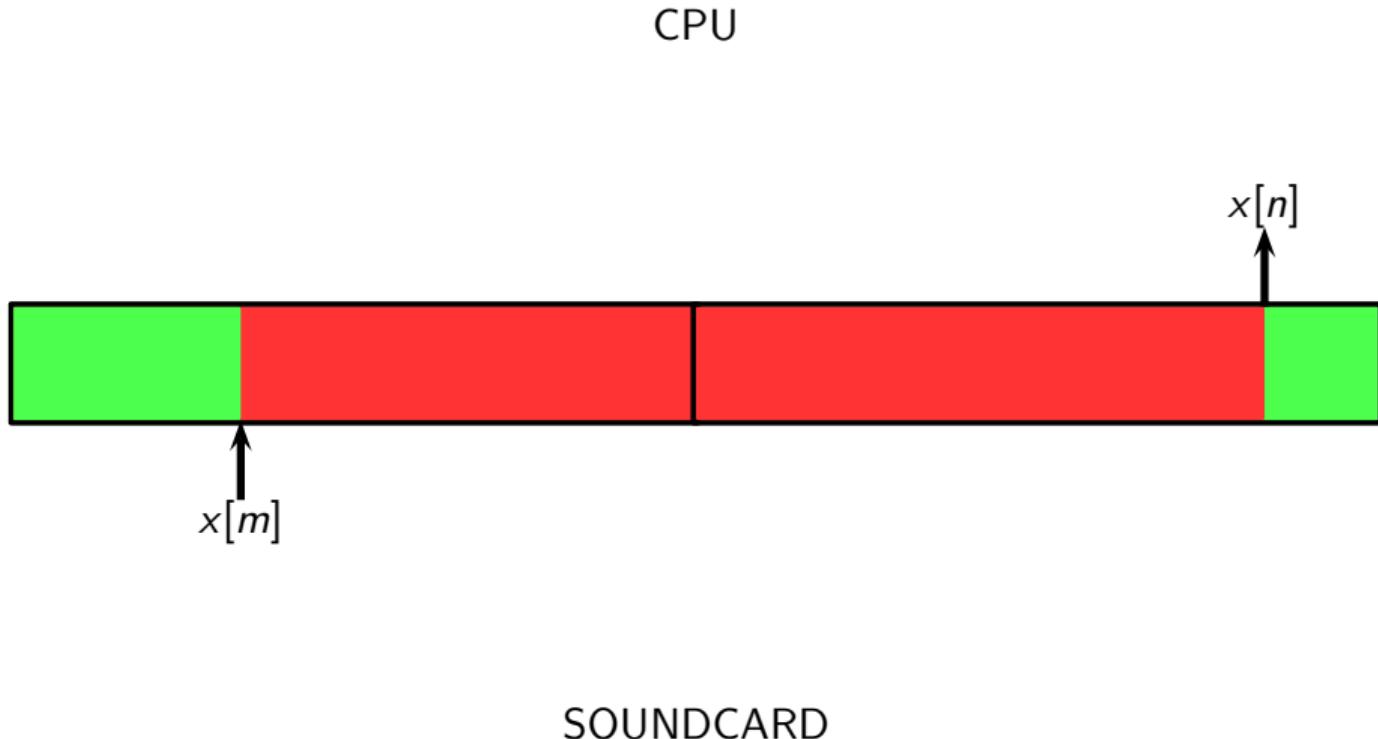
Example: double buffering (input)



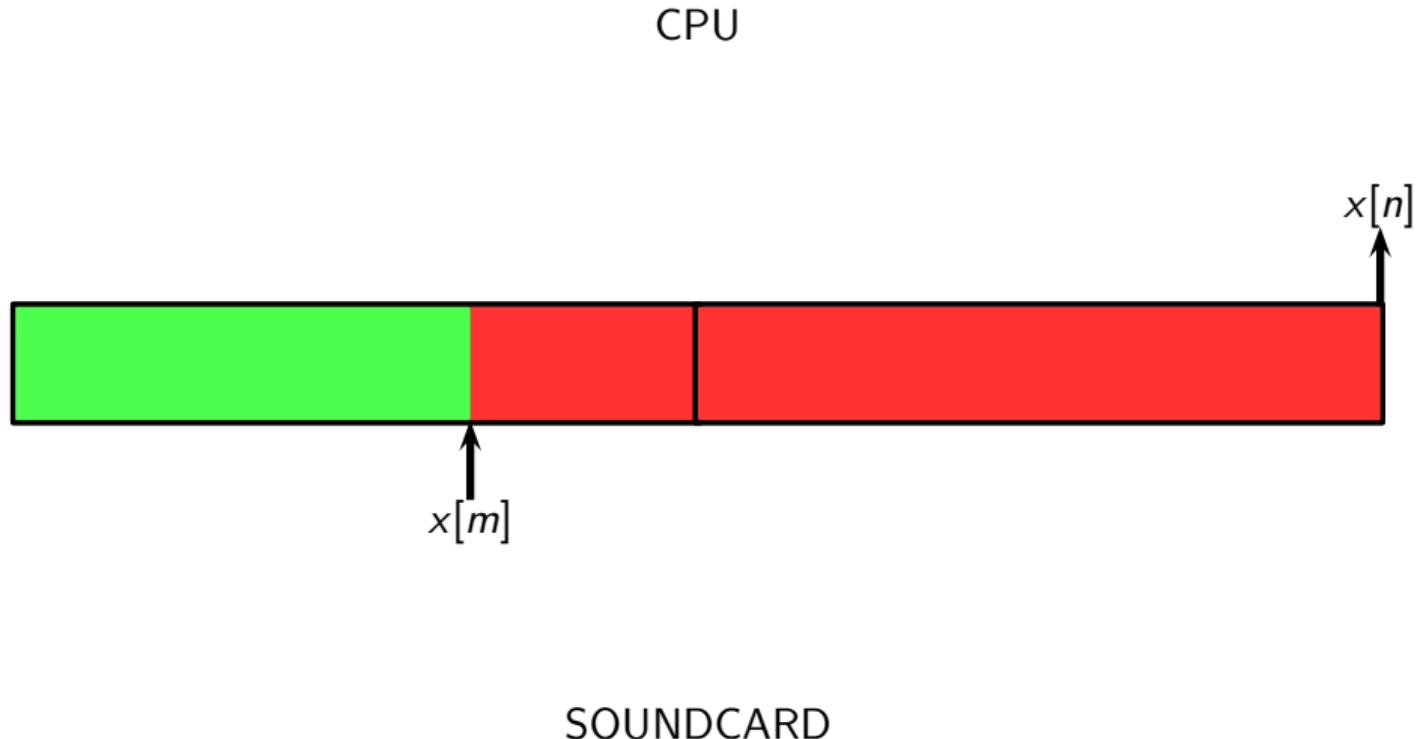
Example: double buffering (input)



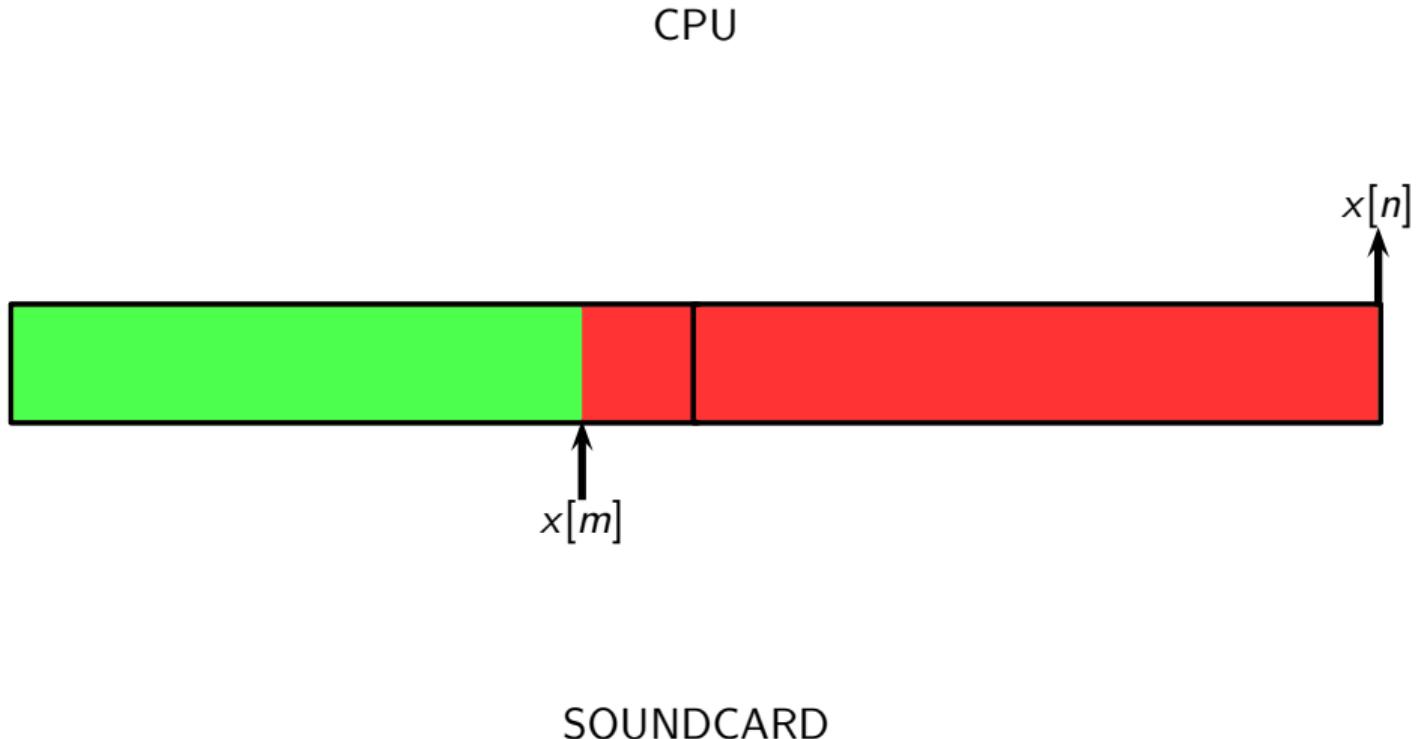
Example: double buffering (input)



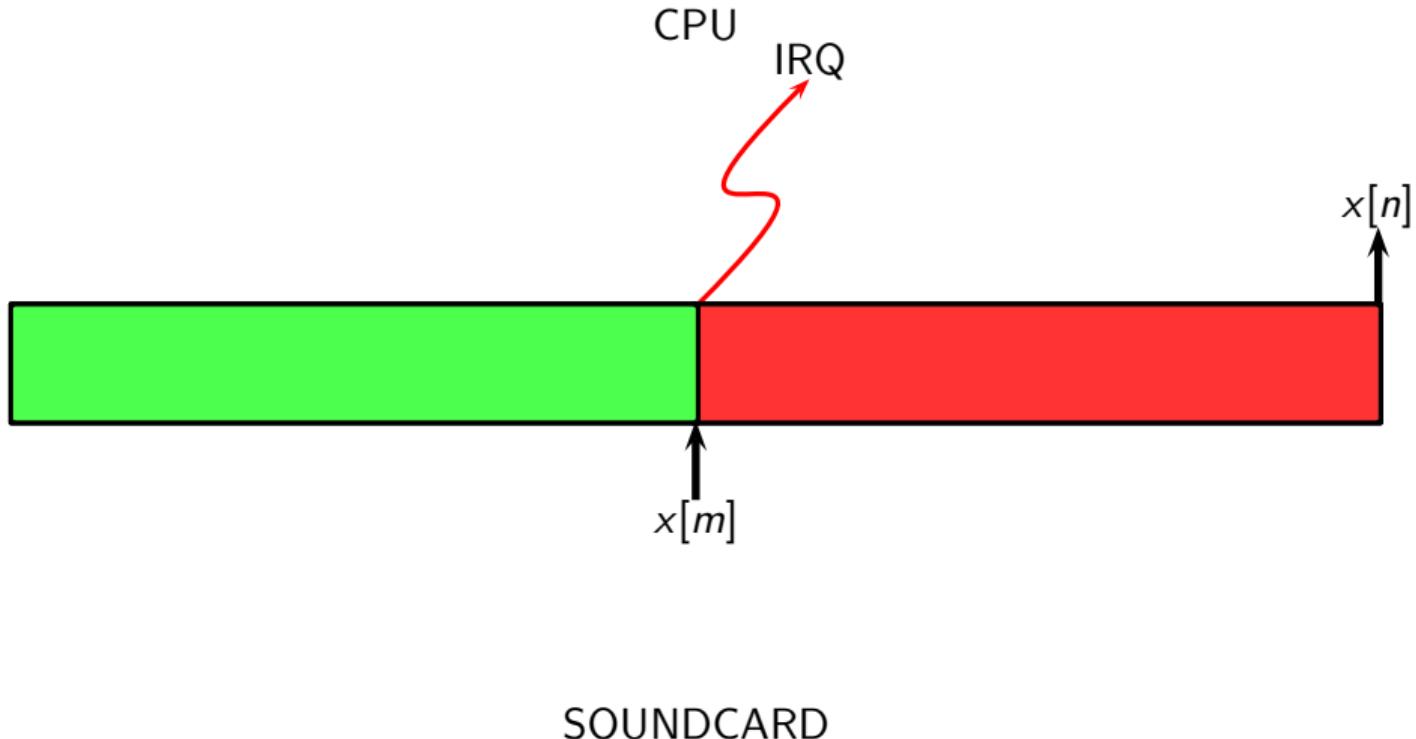
Example: double buffering (input)



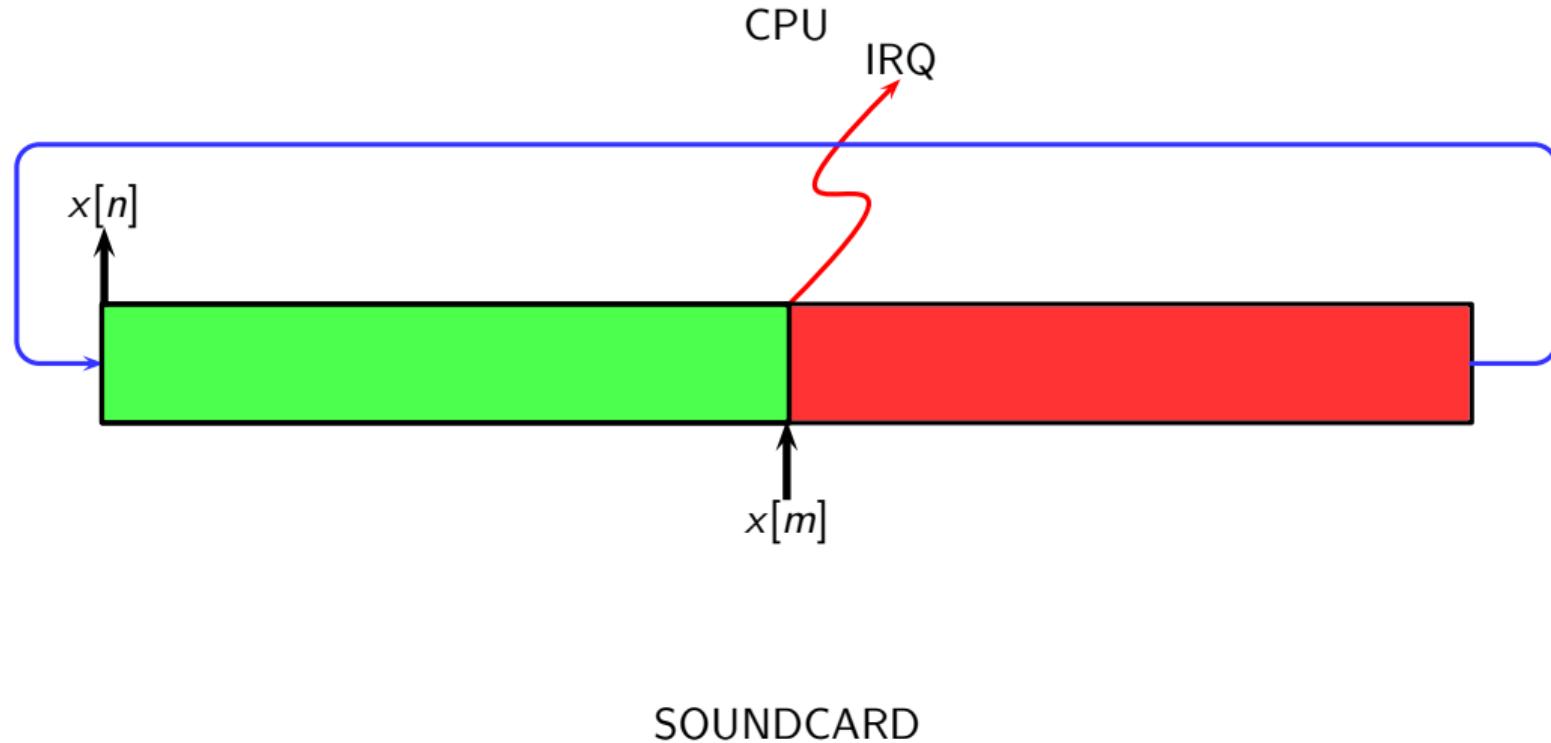
Example: double buffering (input)



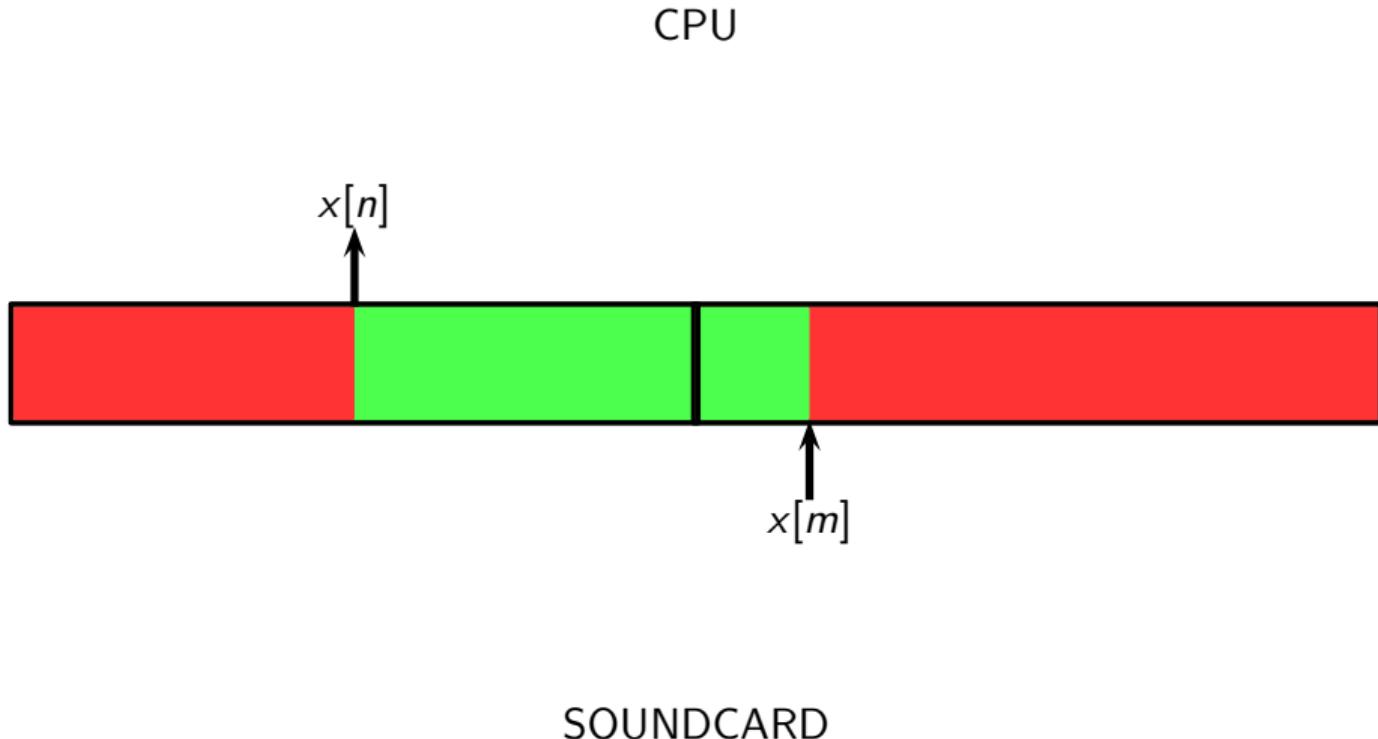
Example: double buffering (input)



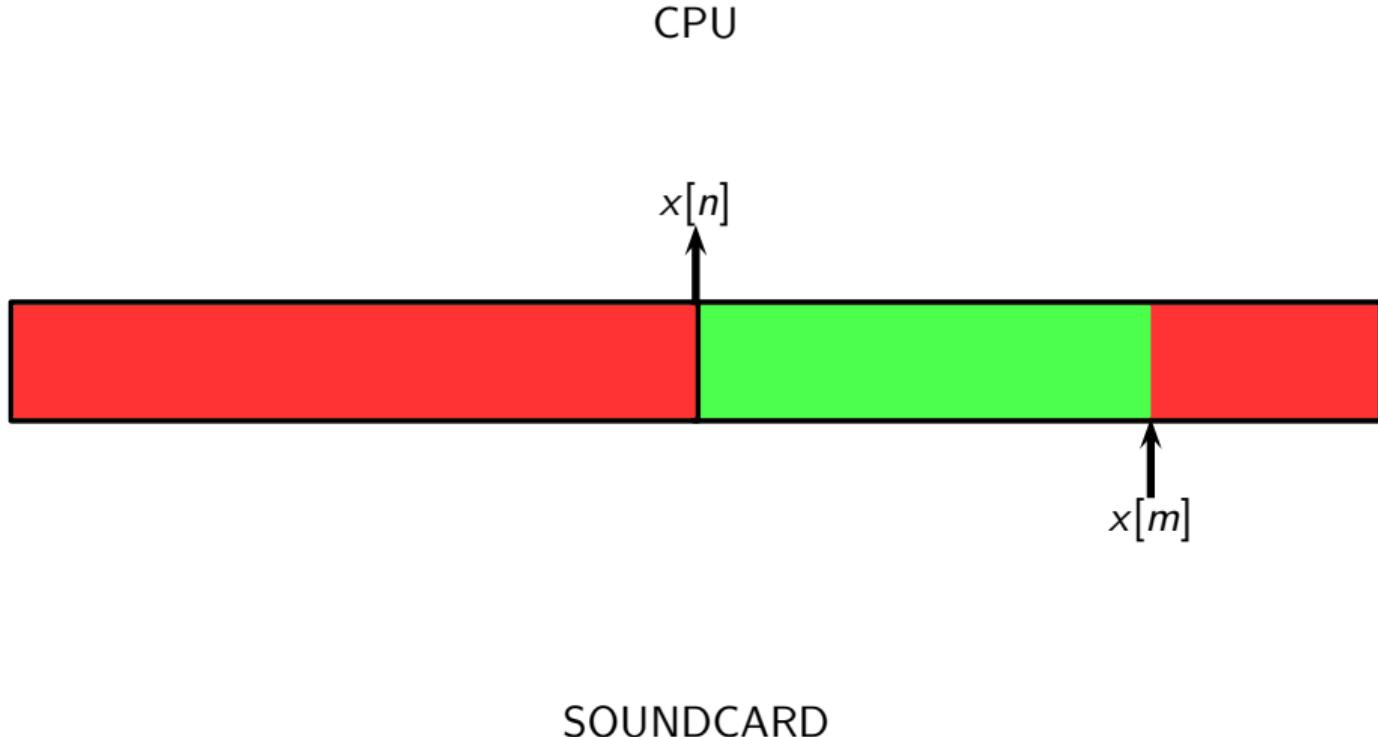
Example: double buffering (input)



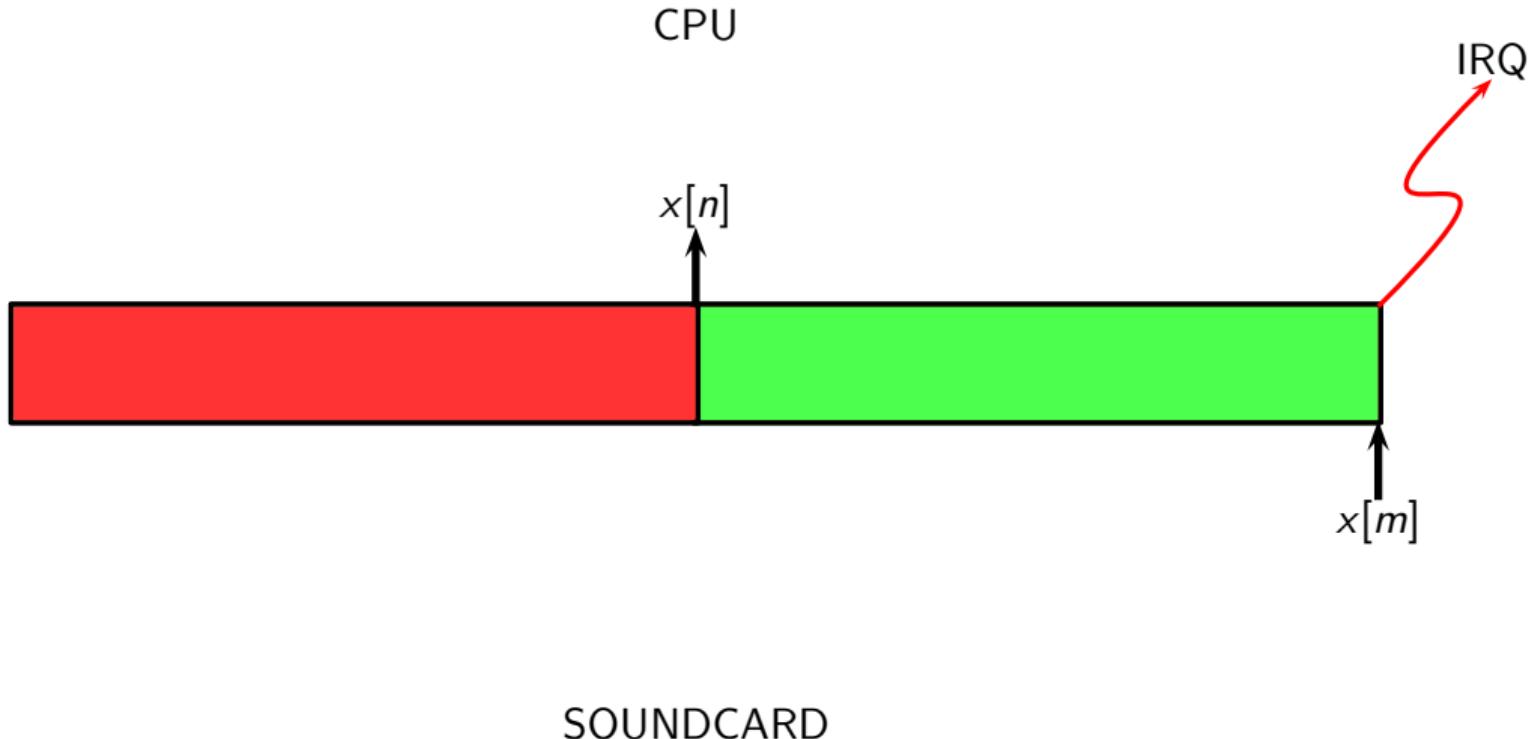
Example: double buffering (input)



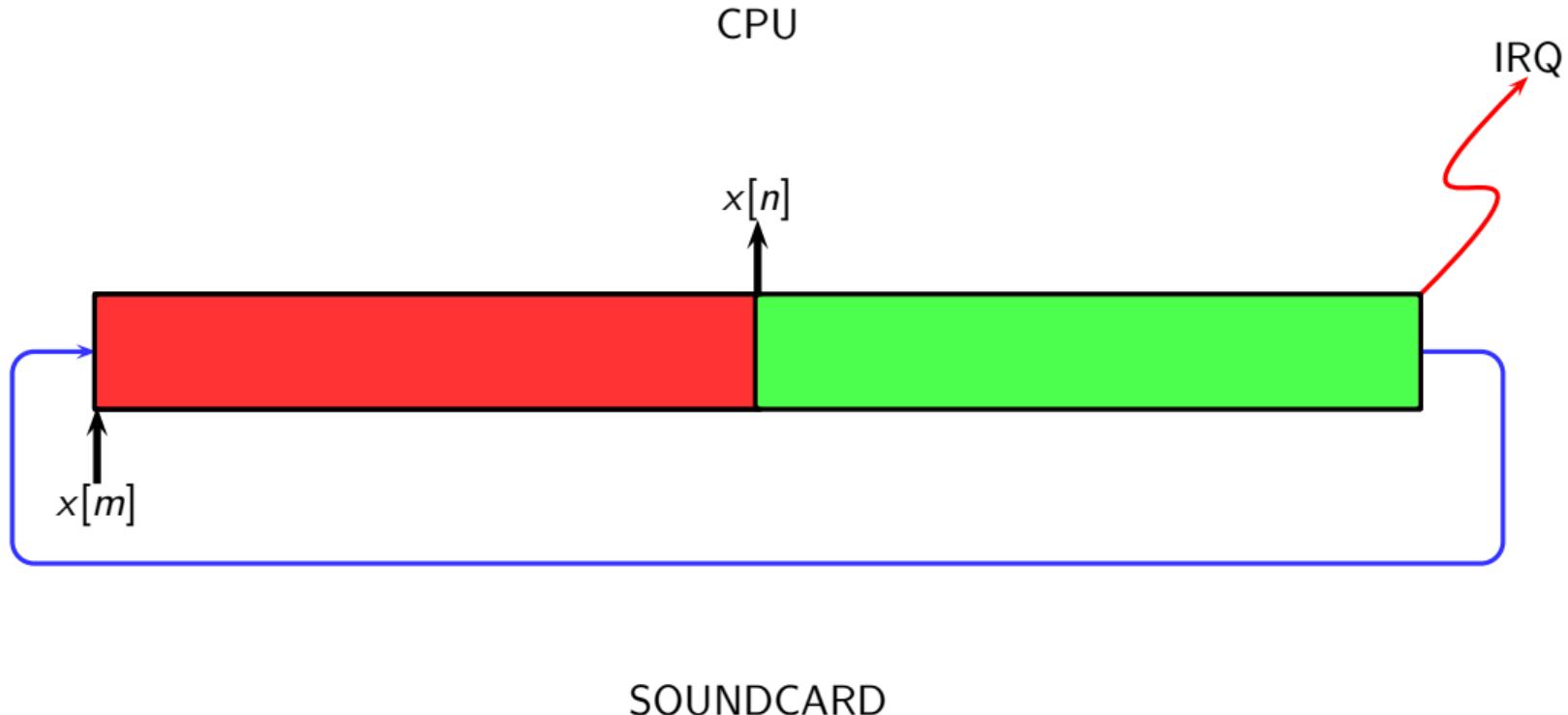
Example: double buffering (input)



Example: double buffering (input)



Example: double buffering (input)



Putting it all together

- ▶ multiple input buffers and output buffers
- ▶ equal chunk sizes
- ▶ input IRQ drives processing

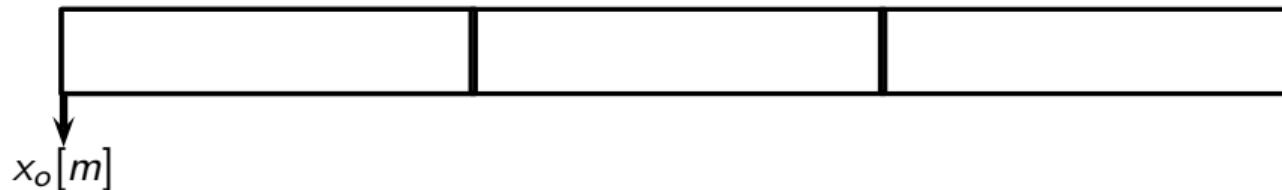
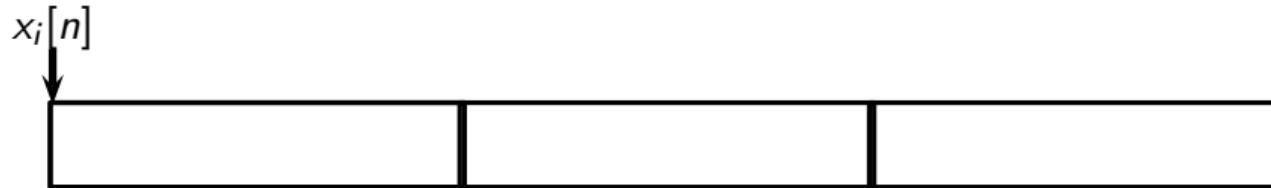
Putting it all together

- ▶ multiple input buffers and output buffers
- ▶ equal chunk sizes
- ▶ input IRQ drives processing

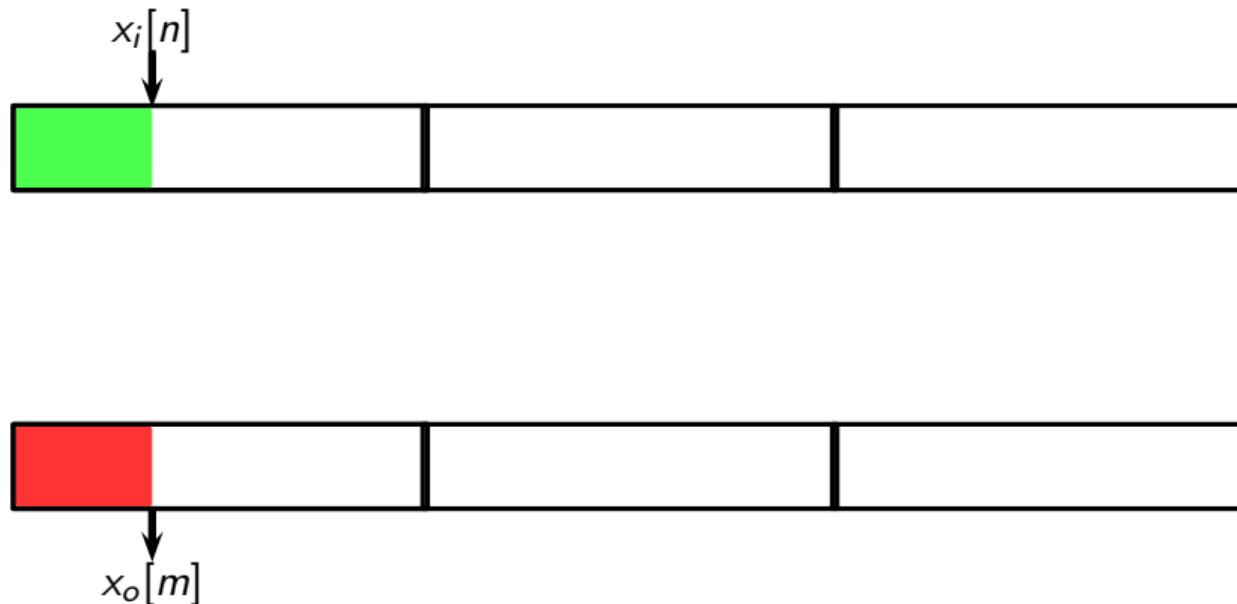
Putting it all together

- ▶ multiple input buffers and output buffers
- ▶ equal chunk sizes
- ▶ input IRQ drives processing

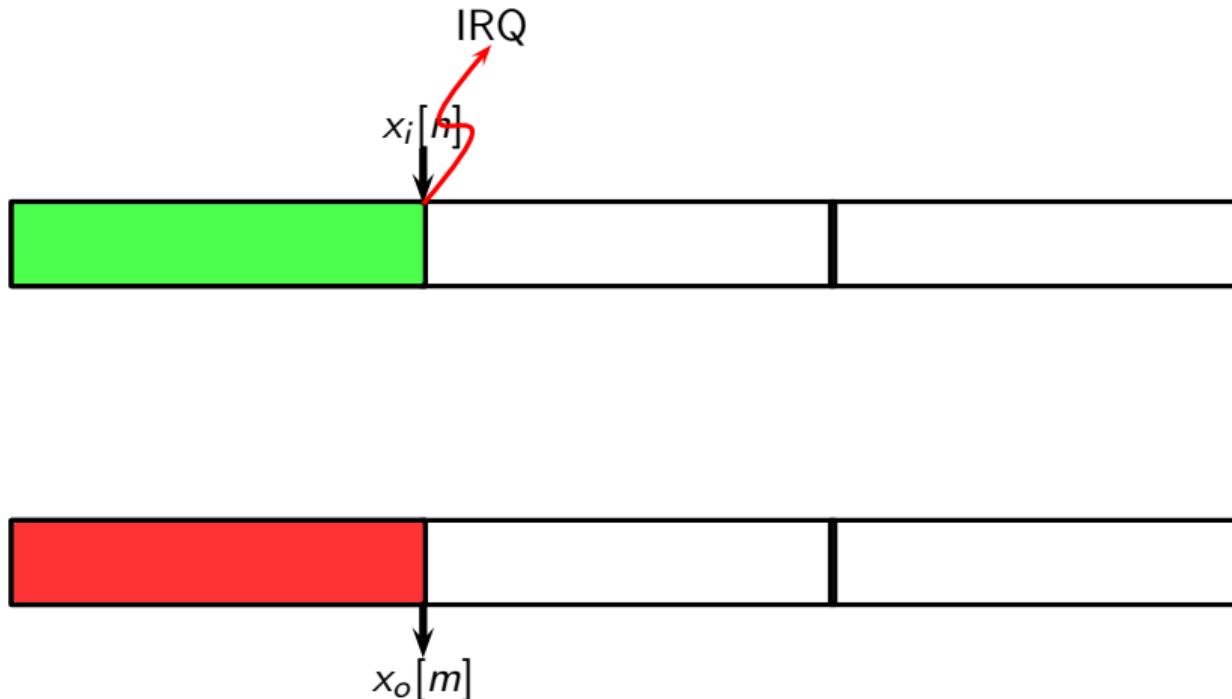
Real-time I/O processing with multiple buffering



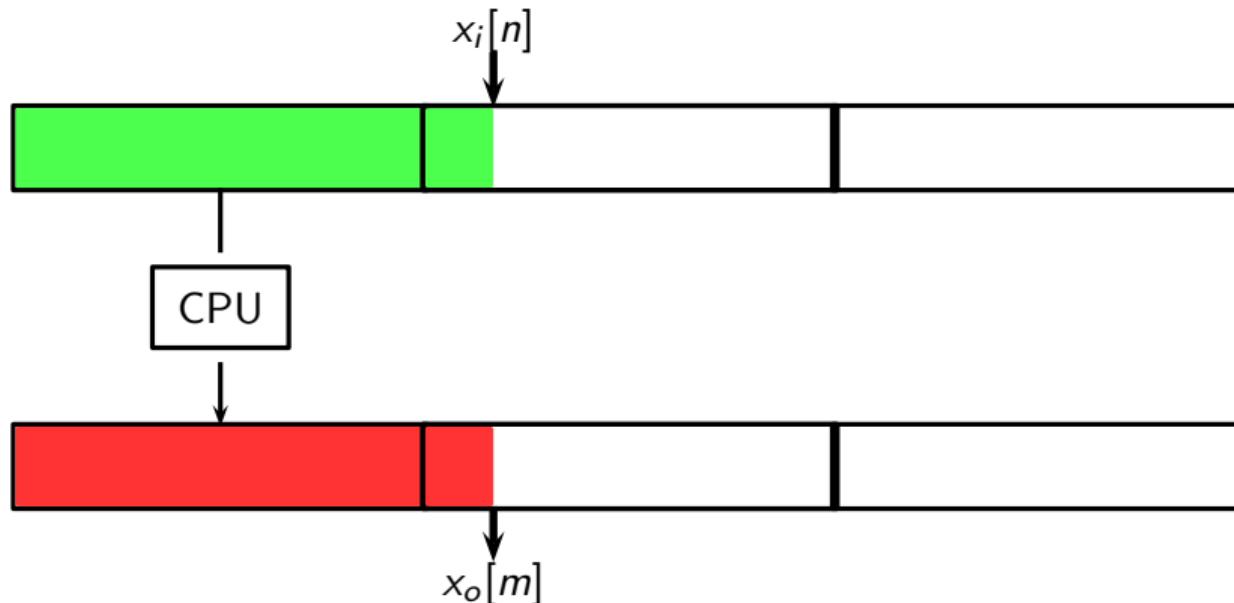
Real-time I/O processing with multiple buffering



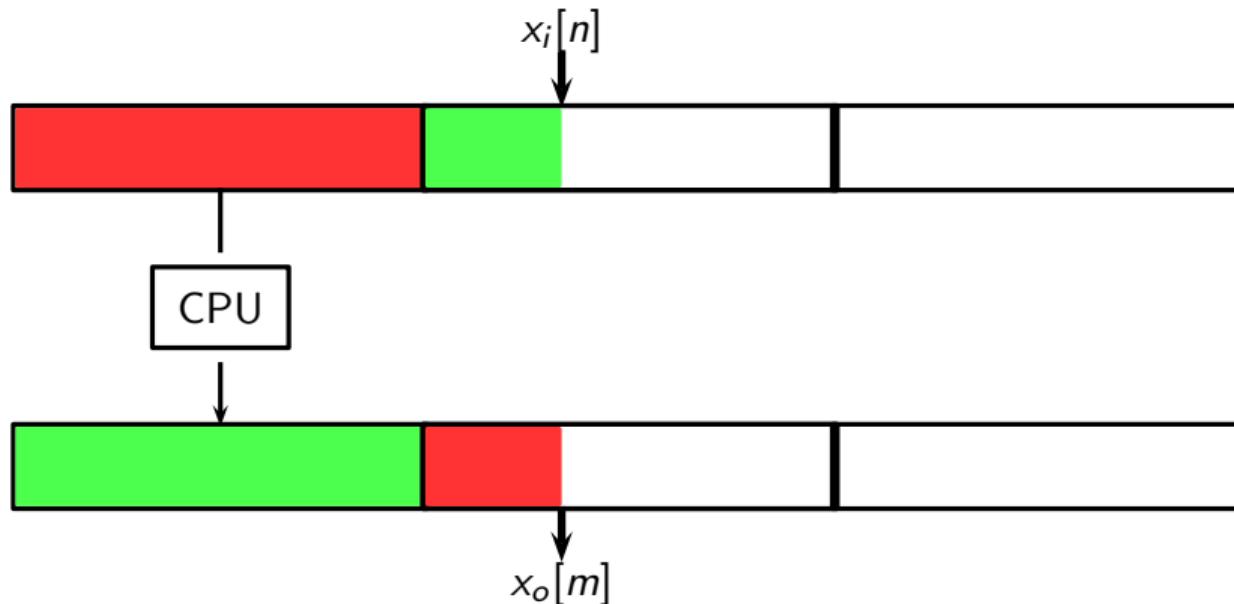
Real-time I/O processing with multiple buffering



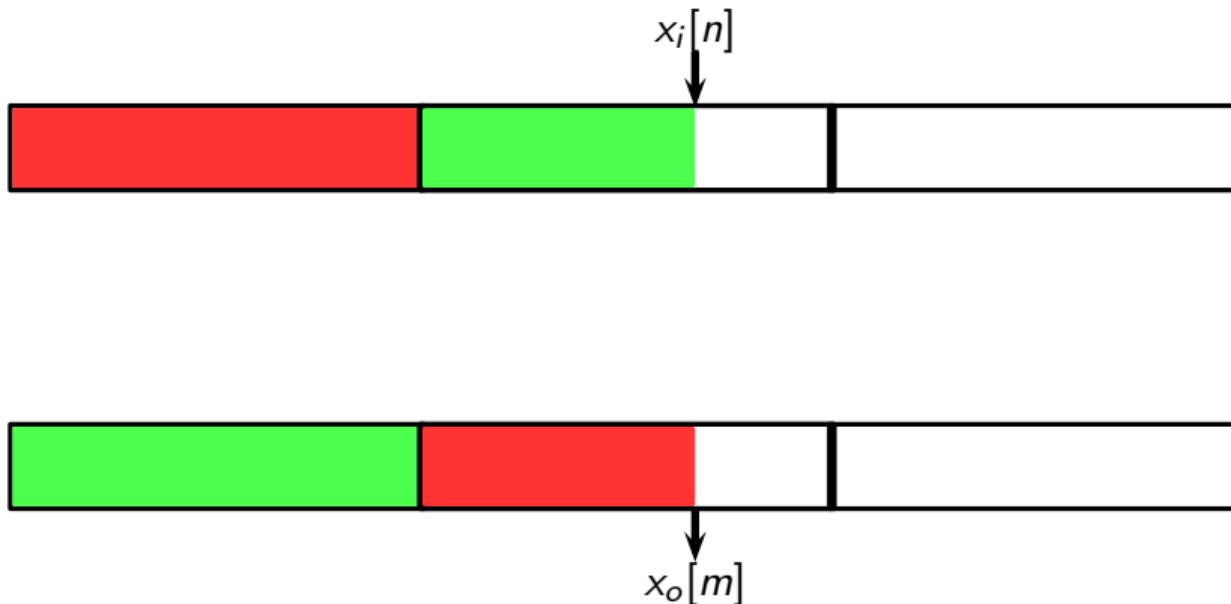
Real-time I/O processing with multiple buffering



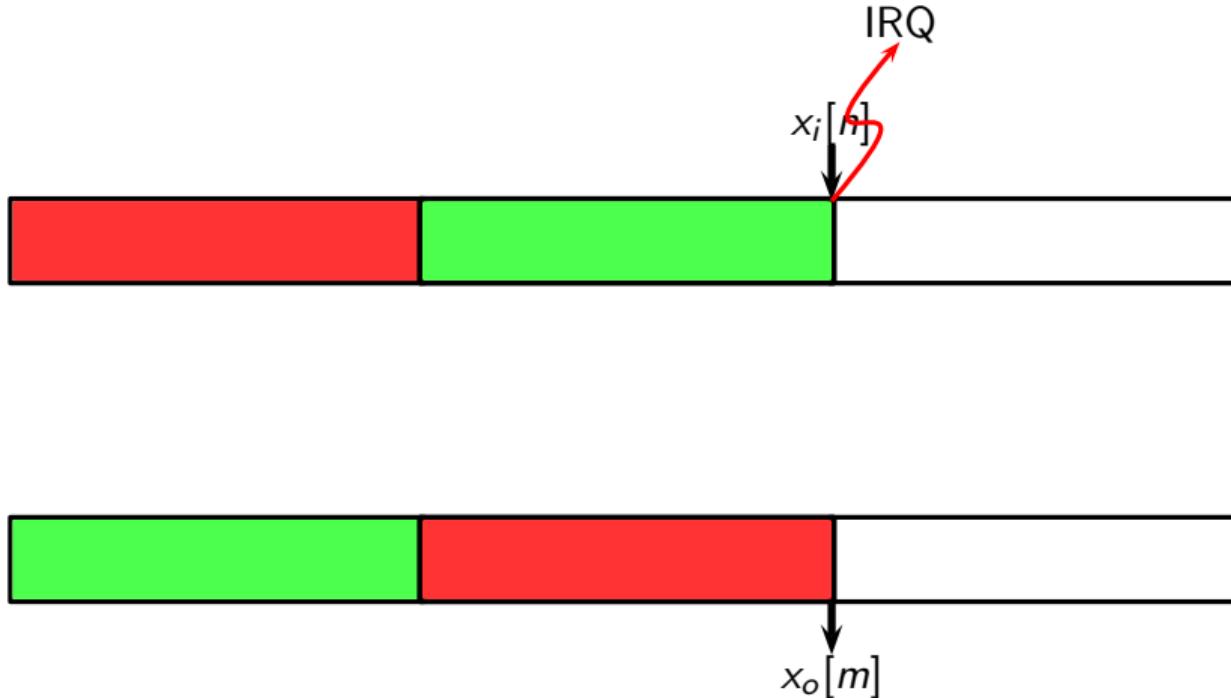
Real-time I/O processing with multiple buffering



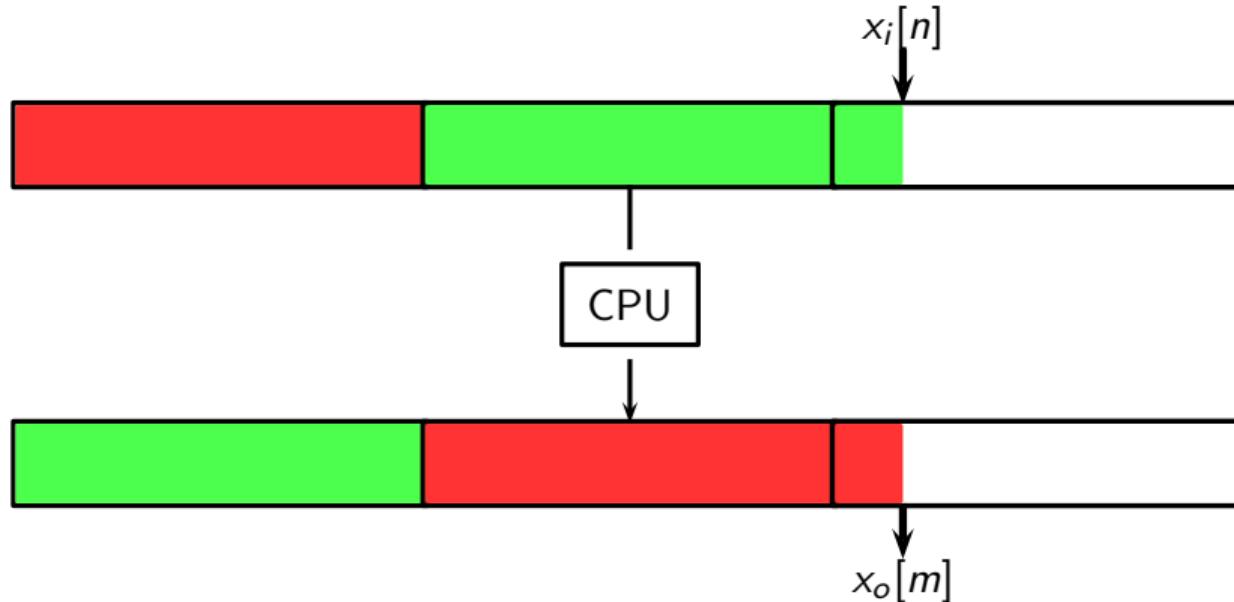
Real-time I/O processing with multiple buffering



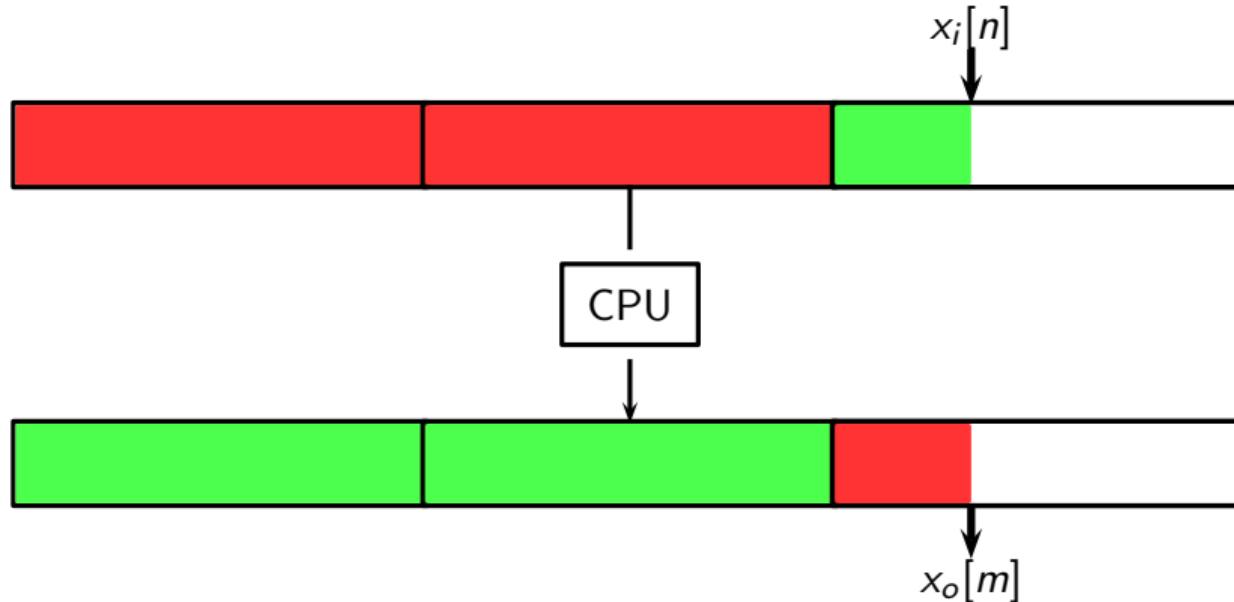
Real-time I/O processing with multiple buffering



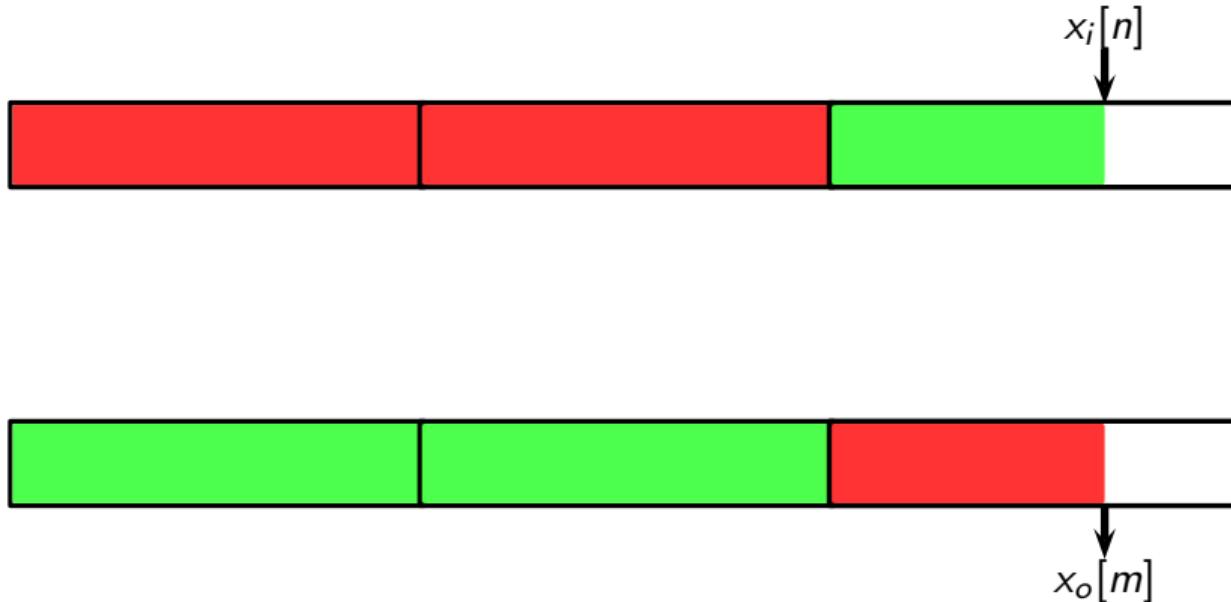
Real-time I/O processing with multiple buffering



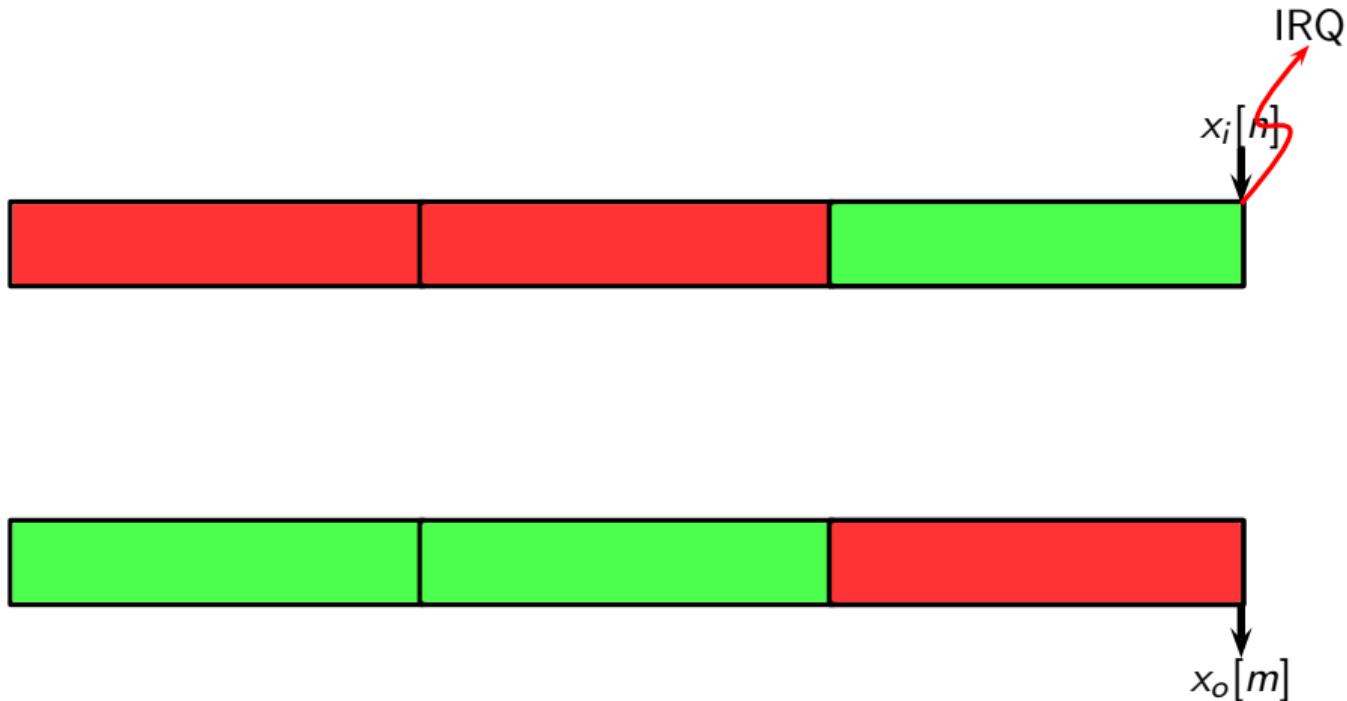
Real-time I/O processing with multiple buffering



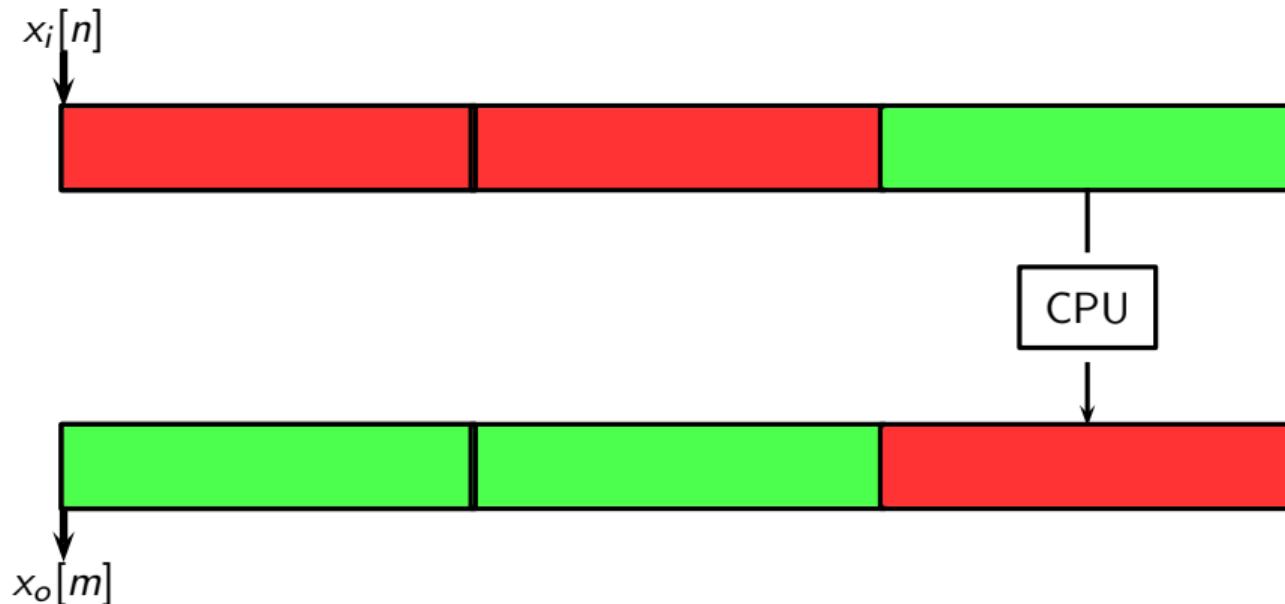
Real-time I/O processing with multiple buffering



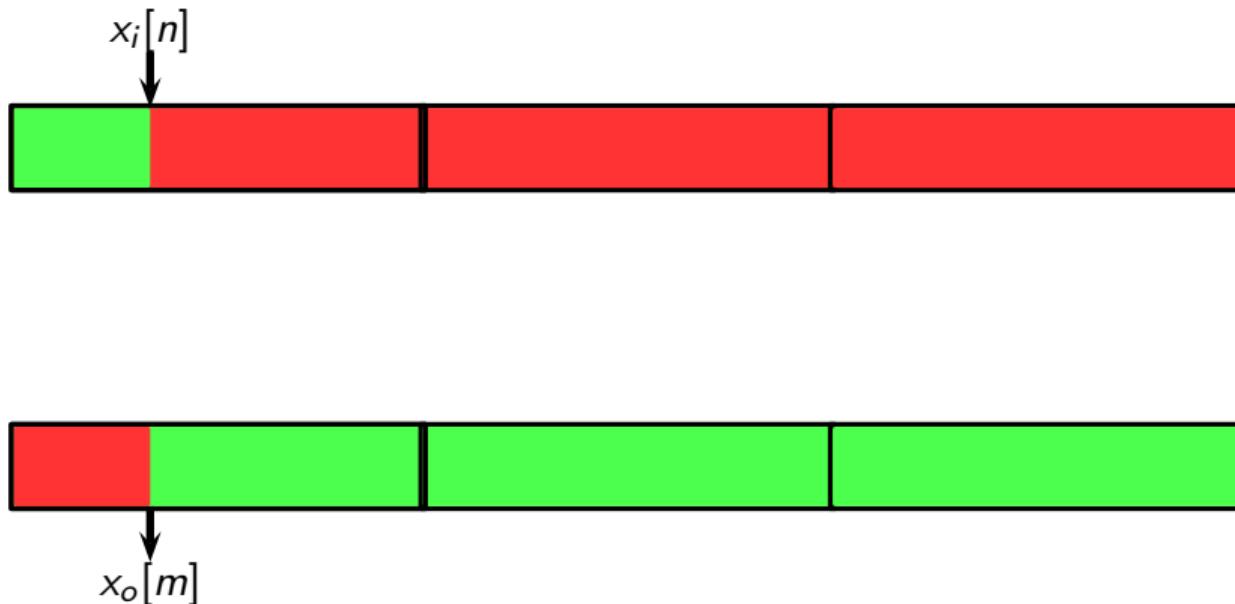
Real-time I/O processing with multiple buffering



Real-time I/O processing with multiple buffering



Real-time I/O processing with multiple buffering



Real-time I/O processing with multiple buffering



Real-time I/O processing with multiple buffering

- ▶ total delay $d = T_s \times L$ seconds
- ▶ usually start output process first
- ▶ buffers can be collapsed

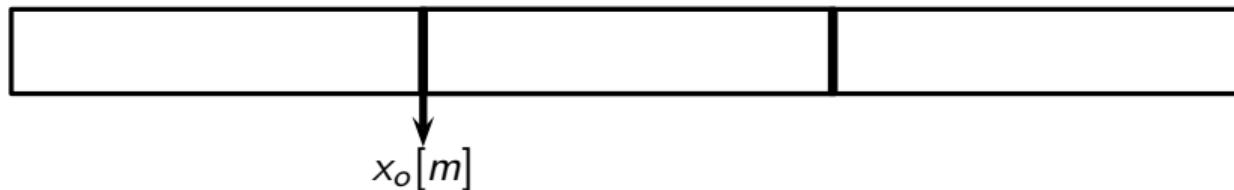
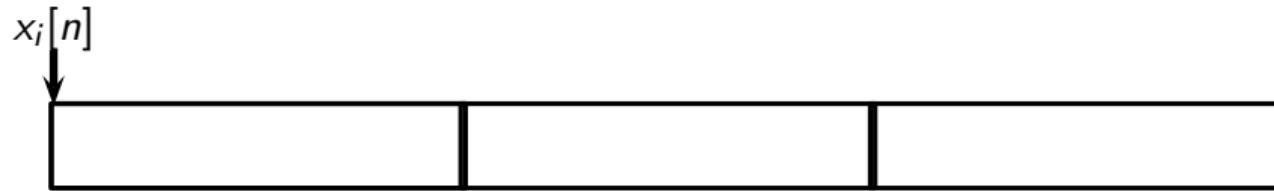
Real-time I/O processing with multiple buffering

- ▶ total delay $d = T_s \times L$ seconds
- ▶ usually start output process first
- ▶ buffers can be collapsed

Real-time I/O processing with multiple buffering

- ▶ total delay $d = T_s \times L$ seconds
- ▶ usually start output process first
- ▶ buffers can be collapsed

Less delay, more risk



Implementation

► low level:

- study soundcard data sheet (each one is different)
- write code to program soundcard via writes to IO ports
- write an interrupt handler
- write the code to handle the data

► high level:

- choose a good API (eg. PortAudio)
- write a callback function to handle the data

Implementation

► low level:

- study soundcard data sheet (each one is different)
- write code to program soundcard via writes to IO ports
- write an interrupt handler
- write the code to handle the data

► high level:

- choose a good API (eg. PortAudio)
- write a callback function to handle the data

Implementation

► low level:

- study soundcard data sheet (each one is different)
- write code to program soundcard via writes to IO ports
 - write an interrupt handler
 - write the code to handle the data

► high level:

- choose a good API (eg. PortAudio)
- write a callback function to handle the data

Implementation

► low level:

- study soundcard data sheet (each one is different)
- write code to program soundcard via writes to IO ports
- write an interrupt handler
- write the code to handle the data

► high level:

- choose a good API (eg. PortAudio)
- write a callback function to handle the data

Implementation

► low level:

- study soundcard data sheet (each one is different)
- write code to program soundcard via writes to IO ports
- write an interrupt handler
- write the code to handle the data

► high level:

- choose a good API (eg. PortAudio)
- write a callback function to handle the data

Implementation

► low level:

- study soundcard data sheet (each one is different)
- write code to program soundcard via writes to IO ports
- write an interrupt handler
- write the code to handle the data

► high level:

- choose a good API (eg. PortAudio)
- write a callback function to handle the data

Implementation

► low level:

- study soundcard data sheet (each one is different)
- write code to program soundcard via writes to IO ports
- write an interrupt handler
- write the code to handle the data

► high level:

- choose a good API (eg. PortAudio)
- write a callback function to handle the data

Implementation

► low level:

- study soundcard data sheet (each one is different)
- write code to program soundcard via writes to IO ports
- write an interrupt handler
- write the code to handle the data

► high level:

- choose a good API (eg. PortAudio)
- write a callback function to handle the data

Callback prototype for PortAudio

```
def callback(in_data, ...):
    audio_data = np.fromstring(in_data, dtype=np.int32)
    for n in range(0, len(audio_data)):
        audio_data[n] = np.int32(processor.process(audio_data[n]))
    return audio_data
```

Processing gateway

```
class RTProcessor(object):
    def __init__(self, rate, channels=1, max_delay=1):
        self.SF = rate
        self.x = CircularBuffer(max_delay)
        self.y = CircularBuffer(max_delay)

    def process(self, sample):
        self.x.push(sample)
        y = self._process()
        self.y.push(y)
        return y
```

Circular Buffer

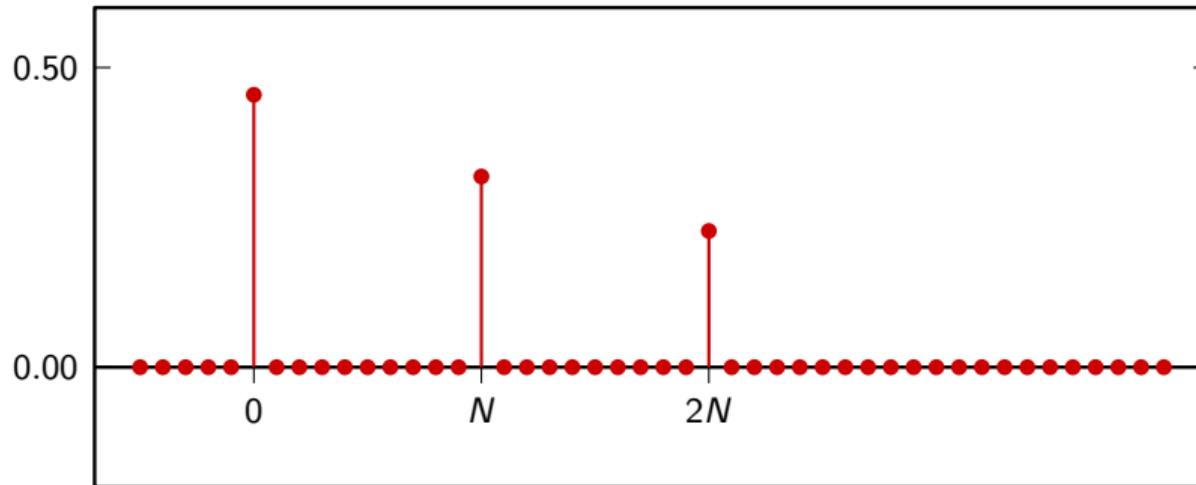
```
class CircularBuffer(object):
    def __init__(self, length):
        self.length = length + 1
        self.buf = np.zeros(self.length)
        self.ix = self.length - 1

    def push(self, x):
        self.ix = np.mod(self.ix + 1, self.length)
        self.buf[self.ix] = x

    def get(self, n):
        return self.buf[np.mod(self.ix + self.length - n, self.length)]
```

Simple Echo

$$y[n] = \frac{a x[n] + b x[n - N] + c x[n - 2N]}{a + b + c}$$



Simple Echo

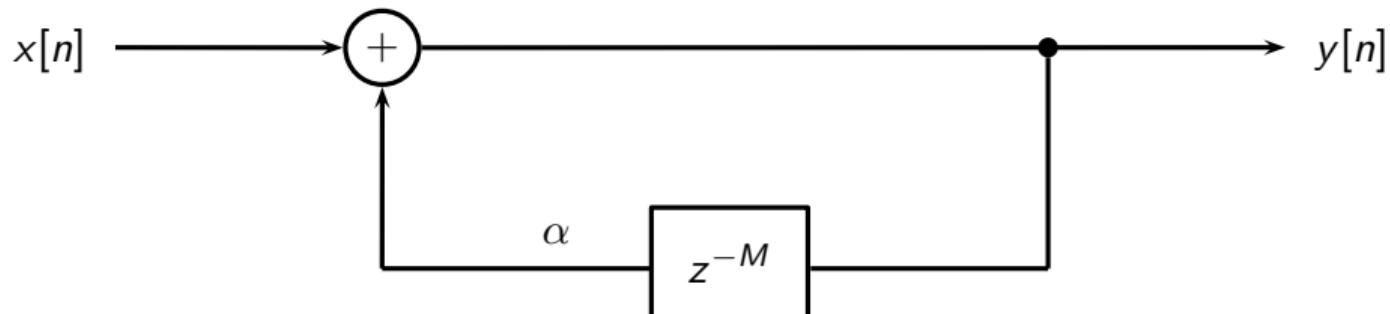
```
class Echo(RTProcessor):
    def __init__(self, rate, channels):
        # 2 replicas, 1/3 of a sec apart -> 1 sec buffering
        super(Echo, self).__init__(rate, channels, max_delay=rate)

        self.a = 1
        self.b = 0.7
        self.c = 0.5
        self.norm = 1.0 / (self.a + self.b + self.c)
        self.N = int(0.3 * self.SF)

    def _process(self):
        return self.norm * (
            self.a * self.x.get(0) +
            self.b * self.x.get(self.N) +
            self.c * self.x.get(2 * self.N))
```

A better echo

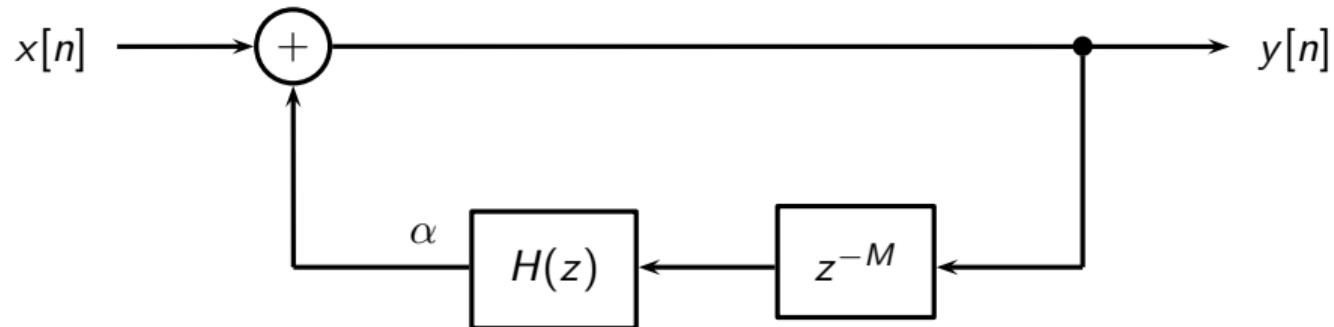
remember the KS algorithm? it's a sort of IIR echo



$$y[n] = \alpha y[n - M] + x[n]$$

A better echo

a natural echo has a lowpass characteristic

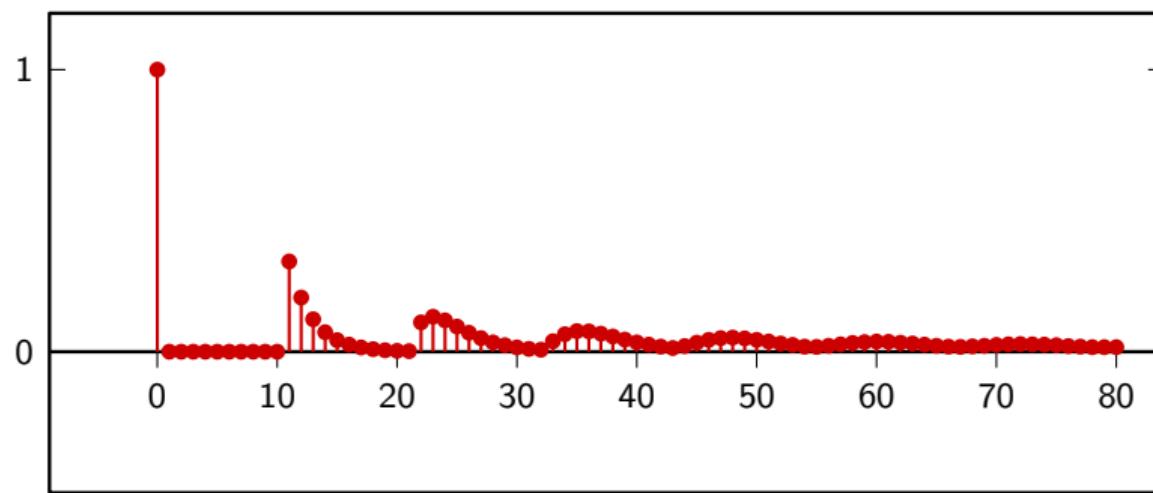


$$y[n] = \alpha(h * y)[n - M] + x[n]$$

A better echo

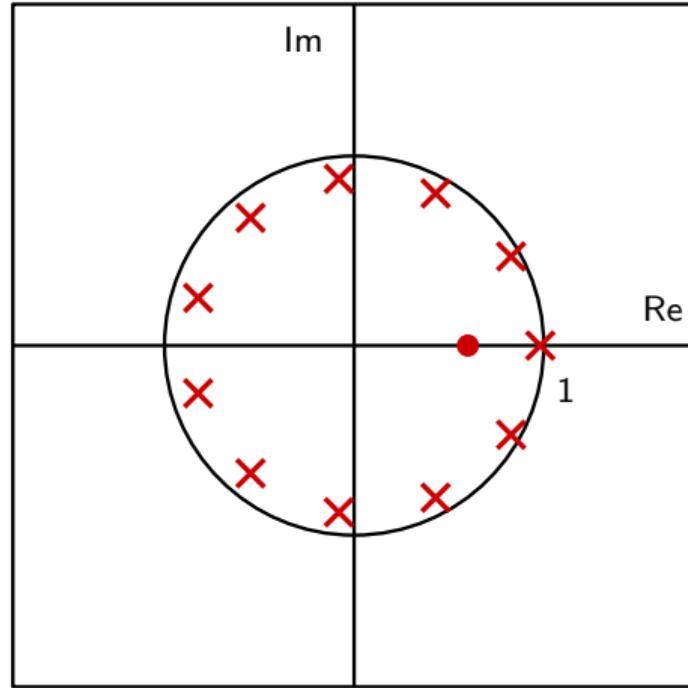
Choose for instance $H(z) = \text{leaky integrator}:$

$$y[n] = x[n] - \lambda x[n-1] + \lambda y[n-1] + \alpha(1-\lambda)y[n-N]$$

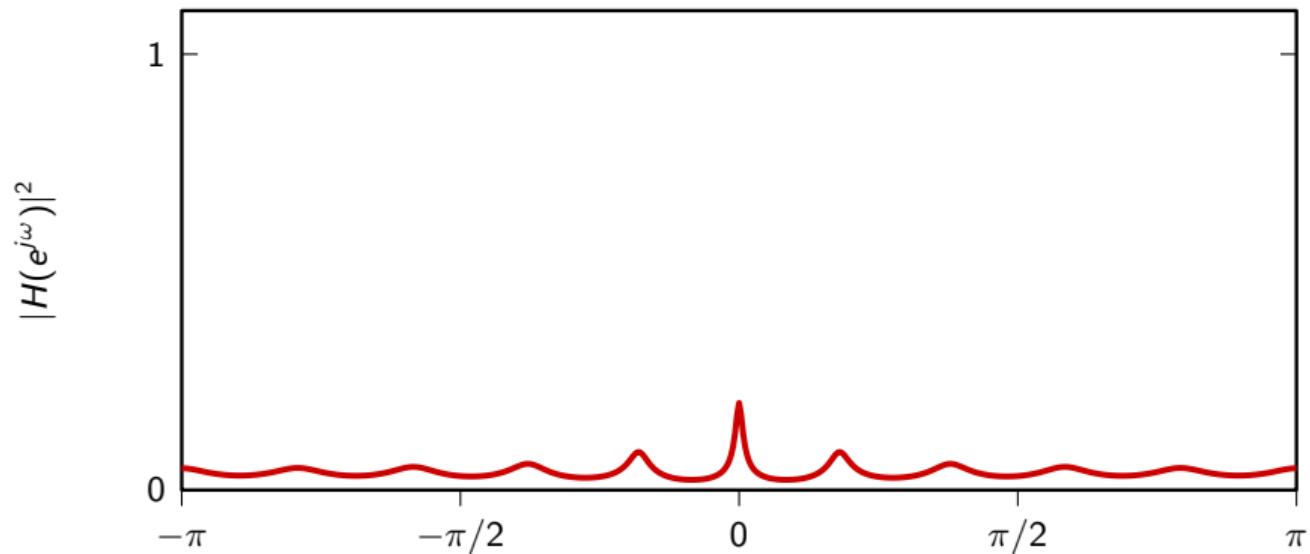


$$N = 10, \lambda = 0.6, \alpha = 0.8$$

A better echo



A better echo



“Natural” Echo

```
class Natural_Echo(RTProcessor):
    def __init__(self, rate, channels):
        super(Natural_Echo, self).__init__(rate, channels, max_delay=rate)

        self.a = 0.9
        self.l = 0.8
        self.N = int(0.3 * self.SF)

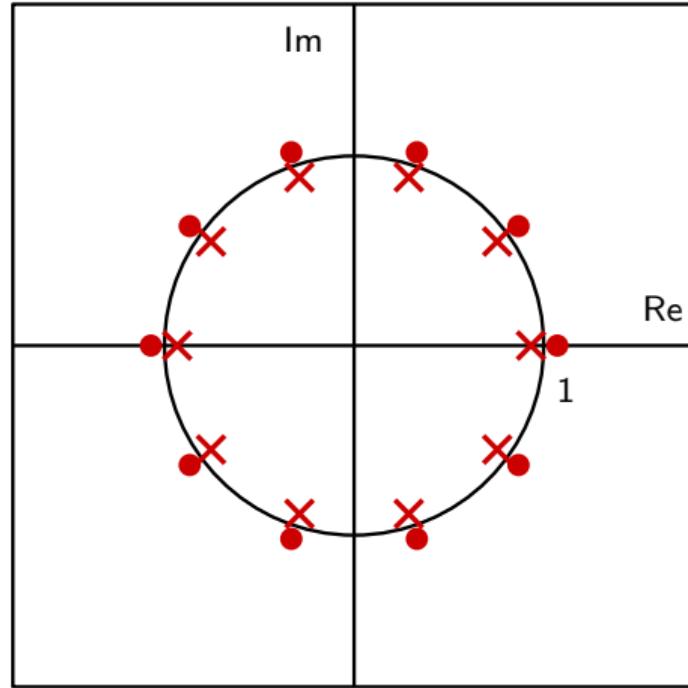
    def _process(self):
        return self.x.get(0) - self.l * self.x.get(1) + \
               self.l * self.y.get(1) + self.a * (1-self.l) * self.y.get(self.N)
```

Reverb

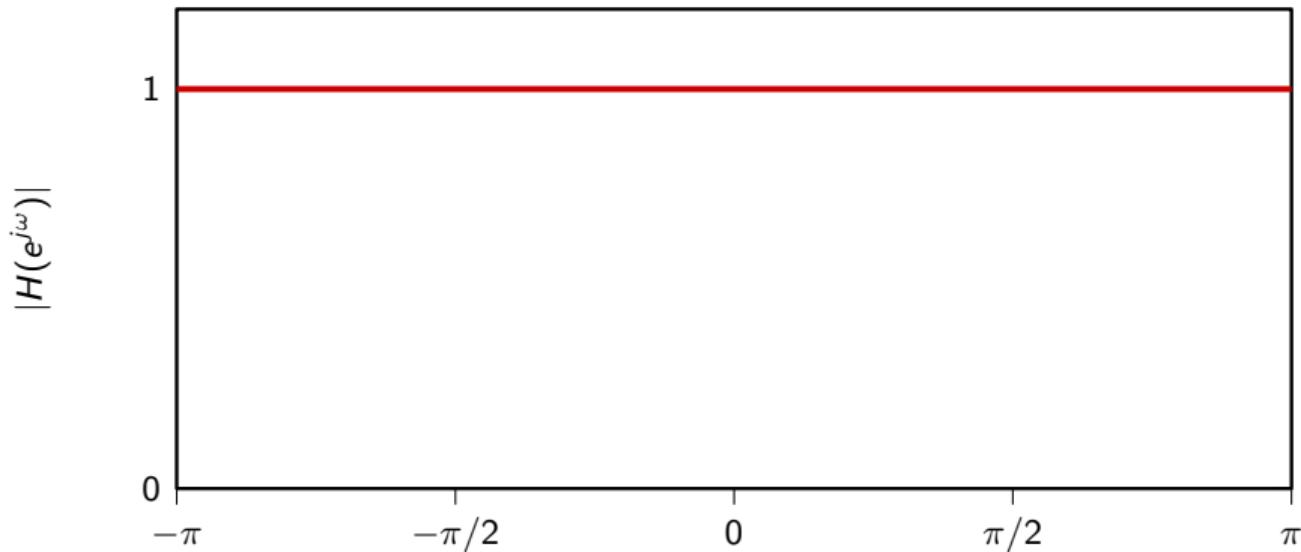
- ▶ reverb is given by the superposition of many many echos with different delays and magnitudes
- ▶ many ways to simulate, always rather costly
- ▶ a cheap alternative is to use an allpass filter

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

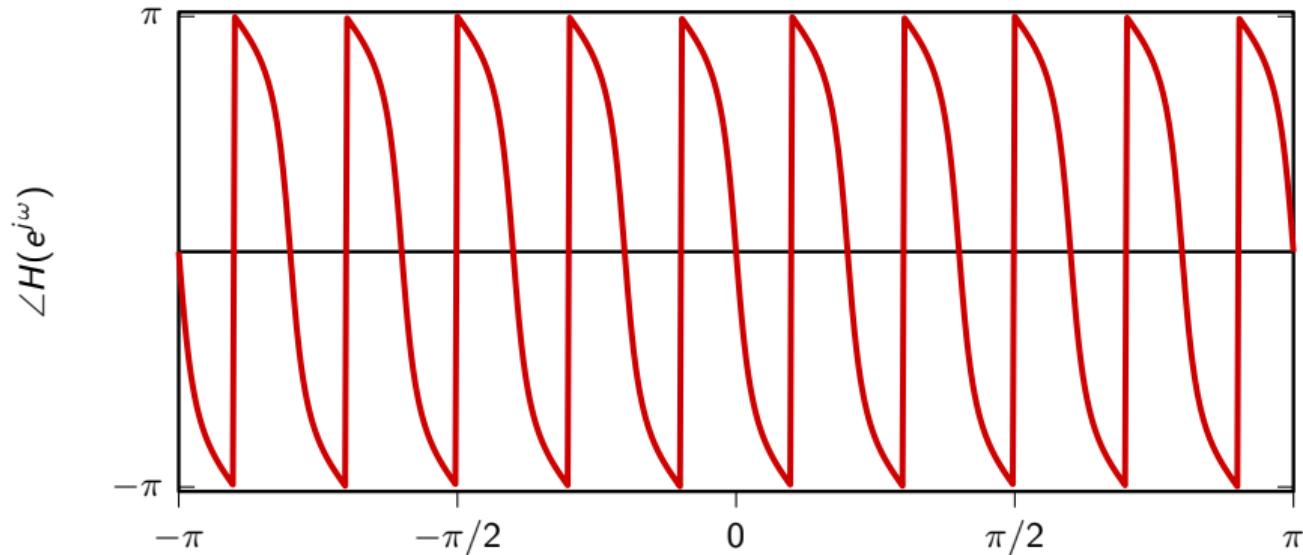
Reverb, poles and zeros ($\alpha = 0.5$, $N = 10$)



Reverb, magnitude response



Reverb, phase response



Reverb

```
class Reverb(RTProcessor):
    def __init__(self, rate, channels):
        super(Reverb, self).__init__(rate, channels, max_delay=rate)

        self.a = 0.8
        self.norm = 0.5
        self.N = int(0.02 * self.SF)

    def _process(self):
        return self.norm *
               (-self.x.get(0) + self.x.get(self.N) + self.a * self.y.get(self.N))
```

Some non-LTI effects

- distortion (fuzz): clip the signal

$$y[n] = \text{trunc}(ax[n])/a$$

- tremolo: sinusoidal amplitude modulation

$$y[n] = (1 + \cos(\omega_0 n)/G)x[n]$$

- flanger: sinusoidal delay

$$y[n] = (x[n] + x[n - d(n)])/2$$

$$d(n) = \text{round}(M(1 - \cos(\omega_0 n)))$$

- wah-wah: time-varying bandpass filter

$$H(z, n) = \frac{(1 - z(n)z^{-1})(1 - z^*(n)z^{-1})}{(1 - p(n)z^{-1})(1 - p^*(n)z^{-1})}$$

$$p(n) = \rho(1 + (\cos \omega_0 n)) e^{j\theta(1 + \cos \omega_1 n)}$$

Some non-LTI effects

- distortion (fuzz): clip the signal

$$y[n] = \text{trunc}(ax[n])/a$$

- tremolo: sinusoidal amplitude modulation

$$y[n] = (1 + \cos(\omega_0 n)/G)x[n]$$

- flanger: sinusoidal delay

$$y[n] = (x[n] + x[n - d(n)])/2$$

$$d(n) = \text{round}(M(1 - \cos(\omega_0 n)))$$

- wah-wah: time-varying bandpass filter

$$H(z, n) = \frac{(1 - z(n)z^{-1})(1 - z^*(n)z^{-1})}{(1 - p(n)z^{-1})(1 - p^*(n)z^{-1})}$$

$$p(n) = \rho(1 + (\cos \omega_0 n)) e^{j\theta(1+\cos \omega_1 n)}$$

Some non-LTI effects

- distortion (fuzz): clip the signal

$$y[n] = \text{trunc}(ax[n])/a$$

- tremolo: sinusoidal amplitude modulation

$$y[n] = (1 + \cos(\omega_0 n)/G)x[n]$$

- flanger: sinusoidal delay

$$y[n] = (x[n] + x[n - d(n)])/2$$

$$d(n) = \text{round}(M(1 - \cos(\omega_0 n)))$$

- wah-wah: time-varying bandpass filter

$$H(z, n) = \frac{(1 - z(n)z^{-1})(1 - z^*(n)z^{-1})}{(1 - p(n)z^{-1})(1 - p^*(n)z^{-1})}$$

$$p(n) = \rho(1 + (\cos \omega_0 n)) e^{j\theta(1+\cos \omega_1 n)}$$

Some non-LTI effects

- distortion (fuzz): clip the signal

$$y[n] = \text{trunc}(ax[n])/a$$

- tremolo: sinusoidal amplitude modulation

$$y[n] = (1 + \cos(\omega_0 n)/G)x[n]$$

- flanger: sinusoidal delay

$$y[n] = (x[n] + x[n - d(n)])/2$$

$$d(n) = \text{round}(M(1 - \cos(\omega_0 n)))$$

- wah-wah: time-varying bandpass filter

$$H(z, n) = \frac{(1 - z(n)z^{-1})(1 - z^*(n)z^{-1})}{(1 - p(n)z^{-1})(1 - p^*(n)z^{-1})}$$

$$p(n) = \rho(1 + (\cos \omega_0 n)) e^{j\theta(1+\cos \omega_1 n)}$$

Fuzz

```
class Fuzz(RTProcessor):
    def __init__(self, rate, channels):
        # memoryless
        super(Fuzz, self).__init__(rate, channels)

        self.T = 0.005
        self.G = 5
        self.limit = 0x7FFFFFFF * self.T

    def _process(self):
        y = self.x.get(0)
        if (y > self.limit):
            y = self.limit
        if (y < -self.limit):
            y = -self.limit
        return self.G * y
```

Tremolo

```
class Tremolo(RTProcessor):
    def __init__(self, rate, channels):
        super(Tremolo, self).__init__(rate, channels, max_delay=1)

        self.depth = 0.9
        self.phi = 5 * 2*np.pi / self.SF
        self.omega = 0

    def _process(self):
        self.omega += self.phi;
        return ((1.0 - self.depth) +
                self.depth * 0.5 * (1 + np.cos(self.omega))) * self.x.get(0)
```

Flanger

```
class Flanger(RTProcessor):
    def __init__(self, rate, channels):
        super(Flanger, self).__init__(rate, channels, max_delay=rate)

        self.maxd = 0.008 * self.SF
        self.phi = 0.2 * 2*np.pi / self.SF
        self.omega = 0
        self.a = 0.6

    def _process(self):
        self.omega += self.phi;
        d = int(self.maxd * (1.0 - np.cos(self.omega)))
        return self.a * self.x.get(0) + (1.0 - self.a) * self.x.get(d)
```

Wah

```
def _process(self):
    """ Wah-wah autopedal. A slow oscillator moves the positions of
    the poles in a second-order filter around their nominal value
    The result is a time-varying bandpass filter
    """
    # current angle of the pole
    d = self.pole_delta * (1.0 + np.cos(self.omega)) / 2.0
    self.omega += self.phi

    # recompute the filter's coefficients
    self.b1 = -2.0 * self.zero_mag * np.cos(self.zero_phase + d)
    self.a1 = -2.0 * self.pole_mag * np.cos(self.pole_phase + d)

    return 0.3 *
           (self.x.get(0) + self.b1 * self.x.get(1) + self.b2 * self.x.get(2) - \
            self.a1 * self.y.get(1) - self.a2 * self.y.get(2))
```

COM303: Digital Signal Processing

Lecture 15: Stochastic and adaptive signal processing

Update the book!

Please download the new and improved Chapter 8 from the website!

overview

- ▶ random variables, random processes and stationarity
- ▶ spectral representation of random processes
- ▶ adaptive signal processing

from random variables to stationary random processes

Deterministic vs. stochastic

- ▶ deterministic signals are known in advance: $x[n] = \sin(0.2 n)$
- ▶ interesting signals are *not* known in advance: $s[n] = \text{what I'm going to say next}$
- ▶ we usually know something, though: $s[n]$ is a speech signal
- ▶ stochastic signals can be described probabilistically
- ▶ can we do signal processing with random signals? Yes!

Deterministic vs. stochastic

- ▶ deterministic signals are known in advance: $x[n] = \sin(0.2 n)$
- ▶ interesting signals are *not* known in advance: $s[n] = \text{what I'm going to say next}$
- ▶ we usually know something, though: $s[n]$ is a speech signal
- ▶ stochastic signals can be described probabilistically
- ▶ can we do signal processing with random signals? Yes!

Deterministic vs. stochastic

- ▶ deterministic signals are known in advance: $x[n] = \sin(0.2 n)$
- ▶ interesting signals are *not* known in advance: $s[n] = \text{what I'm going to say next}$
- ▶ we usually know something, though: $s[n]$ is a speech signal
- ▶ stochastic signals can be described probabilistically
- ▶ can we do signal processing with random signals? Yes!

Deterministic vs. stochastic

- ▶ deterministic signals are known in advance: $x[n] = \sin(0.2 n)$
- ▶ interesting signals are *not* known in advance: $s[n] = \text{what I'm going to say next}$
- ▶ we usually know something, though: $s[n]$ is a speech signal
- ▶ stochastic signals can be described probabilistically
- ▶ can we do signal processing with random signals? Yes!

Deterministic vs. stochastic

- ▶ deterministic signals are known in advance: $x[n] = \sin(0.2 n)$
- ▶ interesting signals are *not* known in advance: $s[n] = \text{what I'm going to say next}$
- ▶ we usually know something, though: $s[n]$ is a speech signal
- ▶ stochastic signals can be described probabilistically
- ▶ can we do signal processing with random signals? Yes!

Random variable

a *mapping* from a random event to a value $x \in \mathbb{R}$

Examples:

- ▶ tossing a coin: map heads to 0, tails to 1
- ▶ tossing a die: *discrete r.v.* is face value
- ▶ electric circuit: *continuous r.v.* is output voltage

Random variable

a *mapping* from a random event to a value $x \in \mathbb{R}$

Examples:

- ▶ tossing a coin: map heads to 0, tails to 1
- ▶ tossing a die: *discrete r.v.* is face value
- ▶ electric circuit: *continuous r.v.* is output voltage

Random variable

a *mapping* from a random event to a value $x \in \mathbb{R}$

Examples:

- ▶ tossing a coin: map heads to 0, tails to 1
- ▶ tossing a die: *discrete* r.v. is face value
- ▶ electric circuit: *continuous* r.v. is output voltage

Random variable

a *mapping* from a random event to a value $x \in \mathbb{R}$

Examples:

- ▶ tossing a coin: map heads to 0, tails to 1
- ▶ tossing a die: *discrete* r.v. is face value
- ▶ electric circuit: *continuous* r.v. is output voltage

Measuring probability

- ▶ cumulative distribution function (cdf):

$$F_x(\alpha) = P[x \leq \alpha], \quad \alpha \in \mathbb{R}$$

$$\lim_{\alpha \rightarrow \infty} F_x(\alpha) = 1$$

- ▶ probability density function (pdf):

$$f_x(\alpha) = \frac{dF_x(\alpha)}{d\alpha}, \quad \alpha \in \mathbb{R}$$

$$F_x(\alpha) = \int_{-\infty}^{\alpha} f_x(x) dx$$

Measuring probability

- ▶ cumulative distribution function (cdf):

$$F_x(\alpha) = P[x \leq \alpha], \quad \alpha \in \mathbb{R}$$

$$\lim_{\alpha \rightarrow \infty} F_x(\alpha) = 1$$

- ▶ probability density function (pdf):

$$f_x(\alpha) = \frac{dF_x(\alpha)}{d\alpha}, \quad \alpha \in \mathbb{R}$$

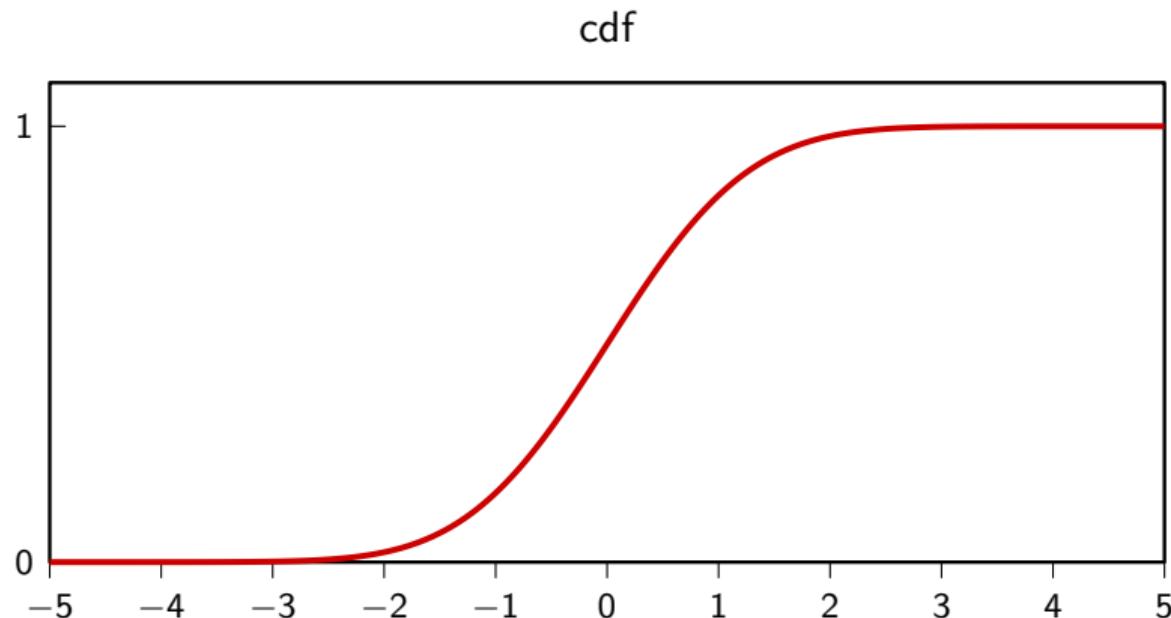
$$F_x(\alpha) = \int_{-\infty}^{\alpha} f_x(x) dx$$

Example

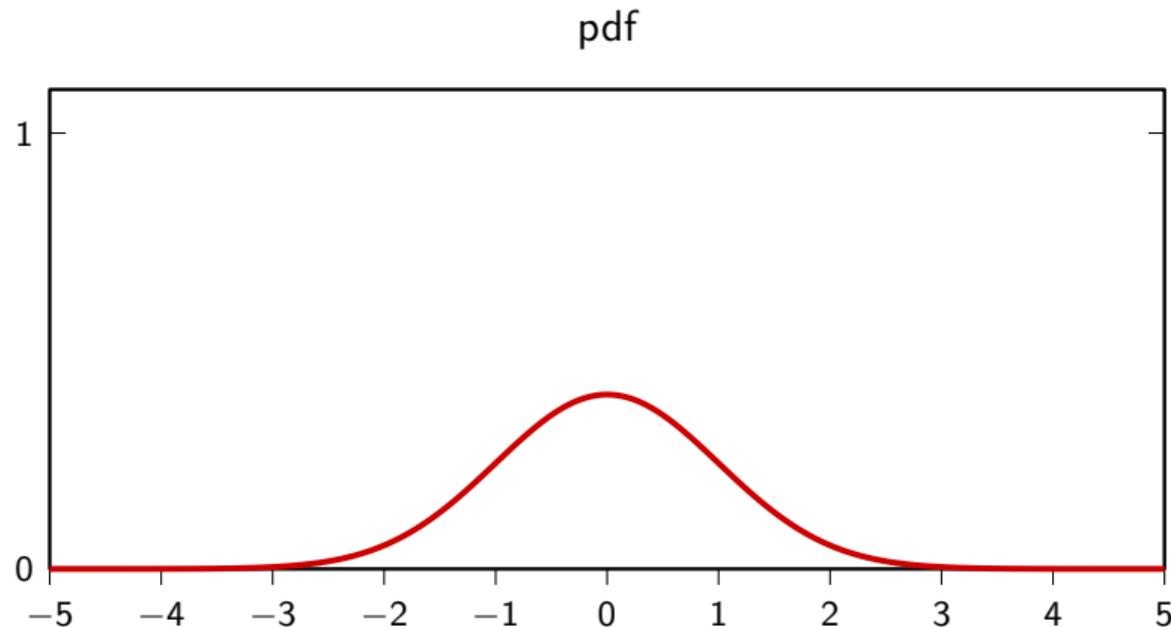
Measure repeatedly the temperature of melting ice:

- ▶ continuous random variable is the measured temperature
- ▶ should be zero Celsius
- ▶ changes in barometric pressure
- ▶ different mineral content in water
- ▶ inaccurate thermometer...

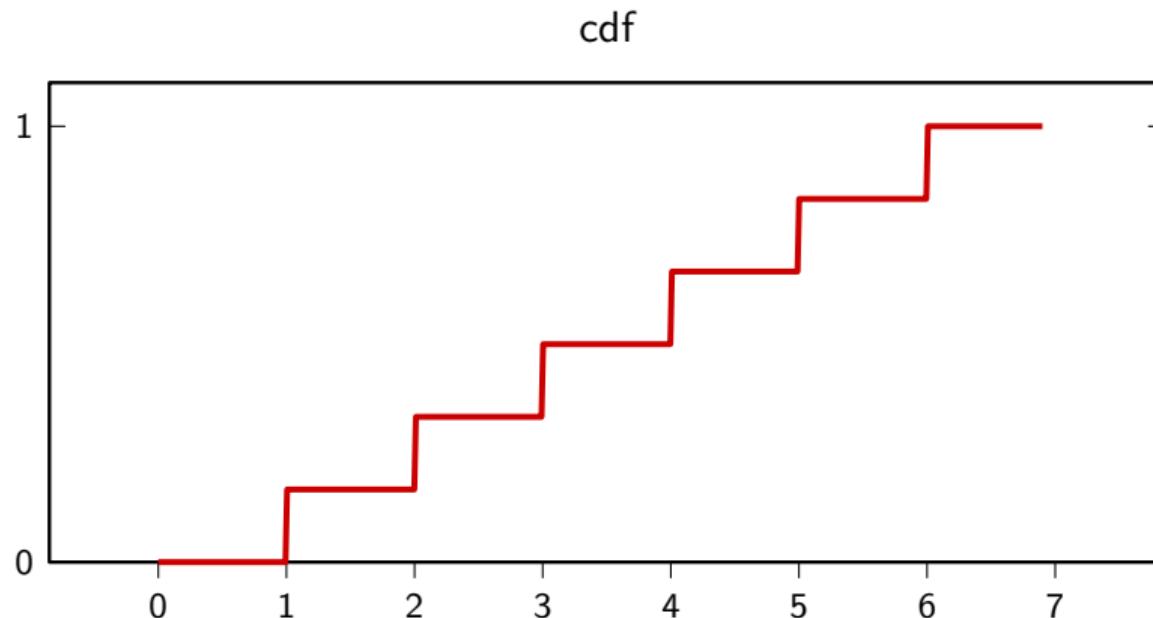
Continuous random variable: Gaussian $\mathcal{N}(0, 1)$



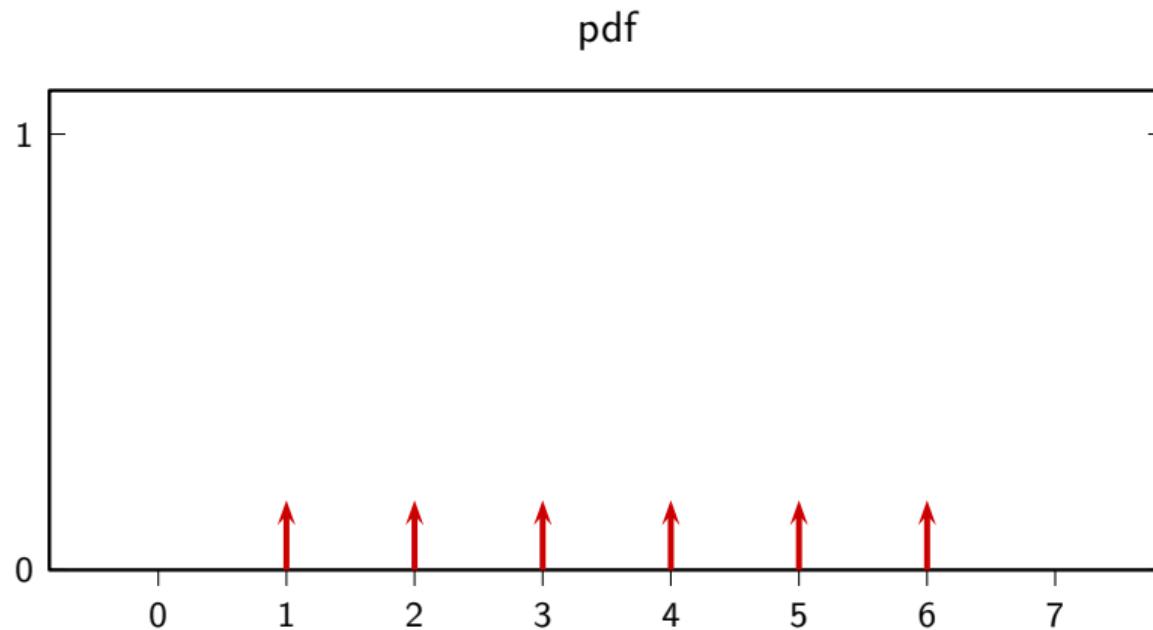
Continuous random variable: Gaussian $\mathcal{N}(0, 1)$



Discrete random variable: die toss



Discrete random variable: die toss



Expectation

$$\mathbb{E}[x] = \int_{-\infty}^{\infty} x f_x(x) \, dx$$

$$\mathbb{E}[g(x)] = \int_{-\infty}^{\infty} g(x) f_x(x) \, dx$$

Expectation

$$\mathbb{E}[x] = \int_{-\infty}^{\infty} x f_x(x) \, dx$$

$$\mathbb{E}[g(x)] = \int_{-\infty}^{\infty} g(x) f_x(x) \, dx$$

Moments

- ▶ raw moments: $E[x^n] = \int_{-\infty}^{\infty} x^n f_x(x) dx$
- ▶ special case: mean: $m_x = E[x] = \int_{-\infty}^{\infty} x f_x(x) dx$
- ▶ central moments: $E[(x - m_x)^n] = \int_{-\infty}^{\infty} (x - m_x)^n f_x(x) dx$
- ▶ special case: variance: $\sigma_x^2 = E[(x - m_x)^2]$

Gaussian Random Variable

$$f(x) = \mathcal{N}(m, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-m)^2}{2\sigma^2}}$$

- ▶ m is the mean
- ▶ σ^2 is the variance

Uniform Random Variable

$$f(x) = \mathcal{U}(A, B) = \frac{1}{B - A}$$

- ▶ $m = \frac{A + B}{2}$
- ▶ $\sigma^2 = \frac{(B - A)^2}{12}$

Discrete Uniform Random Variable

$$f(x) = \mathcal{U}\{A, B\} = \frac{1}{B - A + 1} \sum_{k=A}^B \delta(x - k)$$

- ▶ $m = \frac{A + B}{2}$
- ▶ $\sigma^2 = \frac{(B - A + 1)^2 - 1}{12}$

Relations between random variables

- ▶ cross-correlation: $R_{xy} = E[x y]$.
- ▶ covariance: $C_{xy} = E[(x - m_x)(y - m_y)]$.
- ▶ if zero-mean: $C_{xy} = R_{xy}$

to compute the covariance we need to know the *joint* pdf $f_{xy}(x, y)$:

$$E[g(x, y)] = \int \int_{-\infty}^{\infty} g(x, y) \boxed{f_{xy}(x, y)} dx dy$$

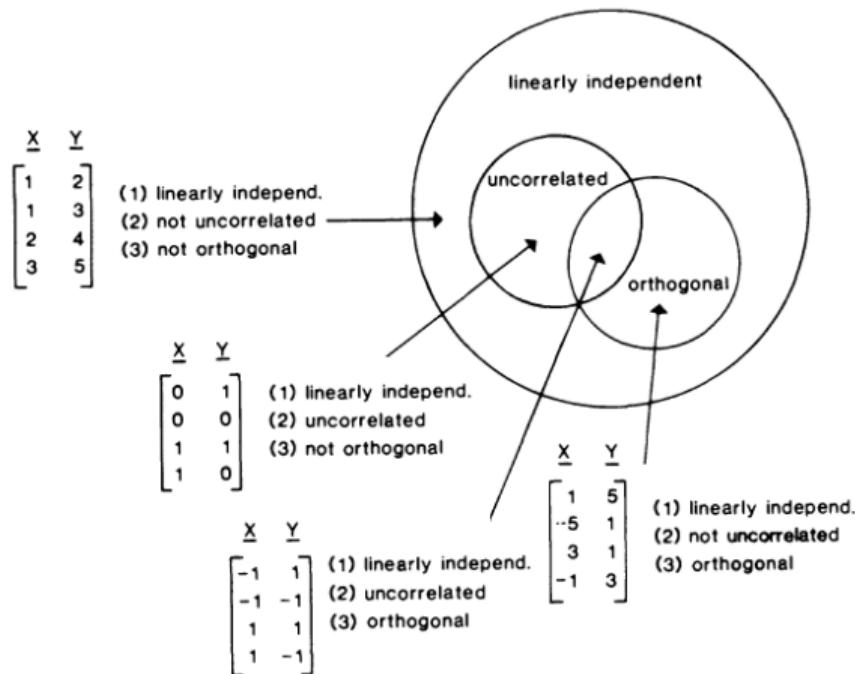
Special relations between random variables

- ▶ uncorrelated elements: $E[xy] = E[x]E[y] = m_x m_y$
(no linear relationship)
- ▶ independent elements: $f_{xy}(x, y) = f_X(x)f_Y(y)$
(no relationship)

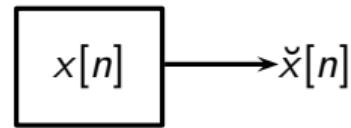
Special relations between random variables

- ▶ uncorrelated elements: $E[xy] = E[x]E[y] = m_x m_y$
(no linear relationship)
- ▶ independent elements: $f_{xy}(x, y) = f_X(x)f_Y(y)$
(no relationship)

A handy map...



Discrete-Time Random Processes



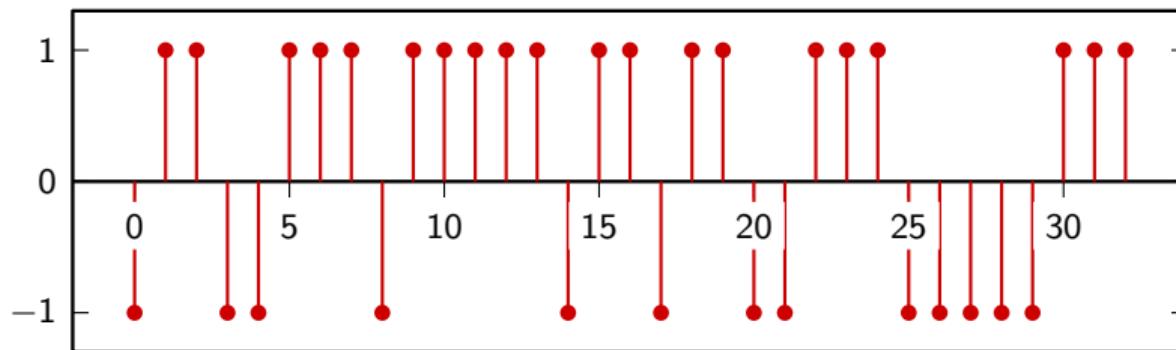
A simple discrete-time random signal generator

For each new sample, toss a fair coin:

$$\check{x}[n] = \begin{cases} +1 & \text{if the outcome of the } n\text{-th toss is head} \\ -1 & \text{if the outcome of the } n\text{-th toss is tail} \end{cases}$$

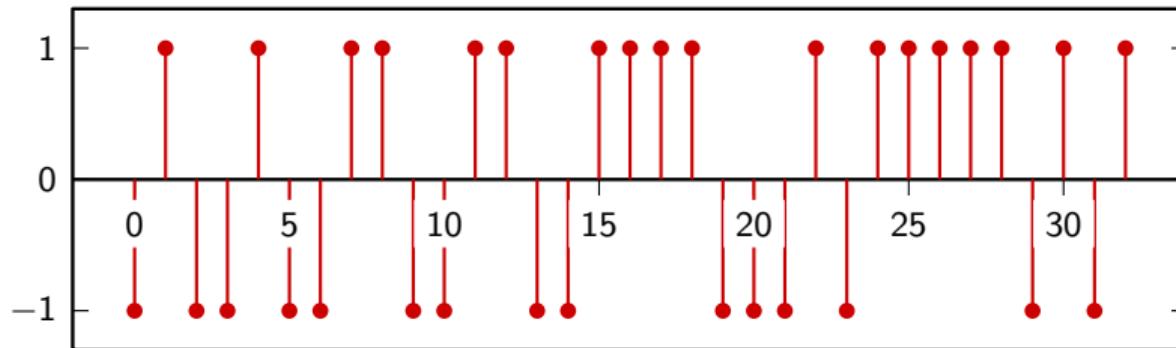
A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



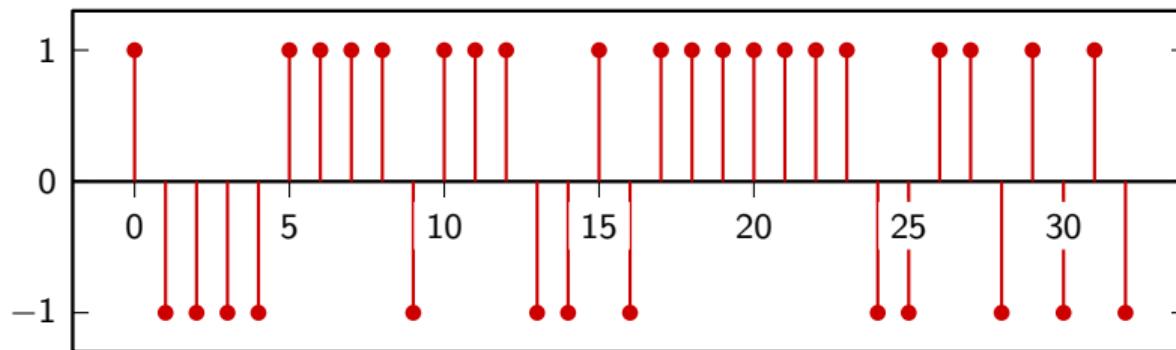
A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



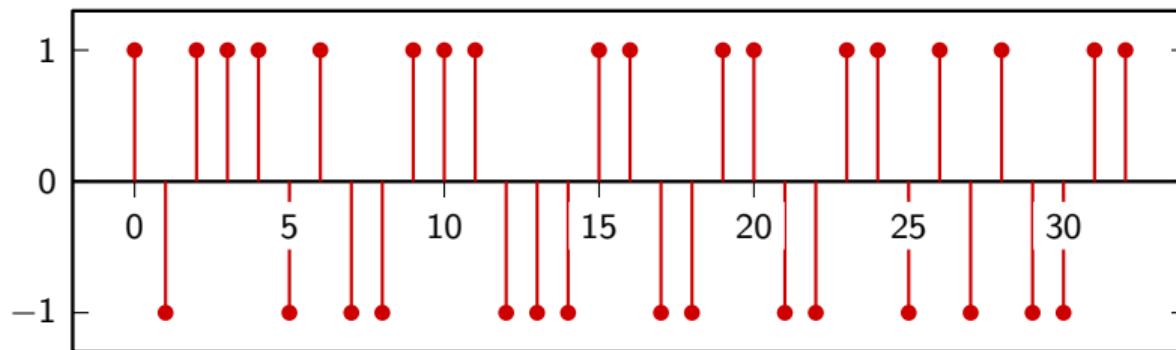
A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



Discrete-Time Random Processes

DT random processes generate an infinite-length sequence of random values

- ▶ what is the distribution of each value ?
- ▶ what are the statistical relations between values?

Discrete-Time Random Processes

- ▶ infinite-length sequence of *interdependent* random variables
- ▶ a full characterization requires knowing

$$f_{x[n_0]x[n_1]\dots x[n_{k-1}]}(x_0, x_1, \dots, x_{k-1})$$

for *all* possible sets of k indices $\{n_0, n_1, \dots, n_{k-1}\}$ and for *all* $k \in \mathbb{N}$

- ▶ clearly too much to handle

k-th order descriptions

► first-order description:

- $f_{X[n]}(x[n]) \rightarrow$ time-varying mean $\bar{x}[n] = E[x[n]]$

► second-order description:

- $f_{X[n]}(x[n]) \rightarrow$ time-varying mean $\bar{x}[n] = E[x[n]]$
- $f_{X[n]X[m]}(x[n], x[m]) \rightarrow$ time-varying auto-correlation $r_x[n, m] = E[x[n]x[m]]$

► third-order description:

- time-varying mean
- time-varying auto-correlation
- $f_{X[n]X[m]X[p]}(x[n], x[m], x[p]) \rightarrow$ time-varying third moment

► ...

Manageable random processes: 1 – Stationarity

for a stationary process, all partial-order descriptions are **time-invariant**:

$$f_{x[n_0]x[n_1]\dots x[n_{k-1}]}(\dots) = f_{x[n_0+M]x[n_1+M]\dots x[n_{k-1}+M]}(\dots)$$

Manageable random processes: 1 – Stationarity

For stationary random processes:

- ▶ mean is time-invariant: $E[x[n]] = m_x$
- ▶ autocorrelation depends only on time lag: $E[x[n]x[m]] = r_x[n - m]$
- ▶ (higher-order moments depend only on relative time differences, etc...)

Manageable random processes: 2 – Wide-Sense Stationarity

For WSS random processes we only care about the first two moments:

- ▶ $E[x[n]] = m_x$
- ▶ $E[x[n]x[m]] = r[n - m]$

Manageable random processes: 2 – Wide-Sense Stationarity

Why WSS?

- ▶ most stochastic SP techniques use quadratic “cost” functions
- ▶ algorithms require only the first and second moments
- ▶ quadratic optimization (Mean Square Error) mathematically well-behaved

White Processes (White Noise)

White noise process:

- ▶ zero-mean: $E[x[n]] = 0$
- ▶ uncorrelated: $E[x[n]x[m]] = E[x[n]]E[x[m]]$ for $m \neq n$
- ▶ autocorrelation $r_x[n] = \sigma_x^2 \delta[n]$

According to underlying distribution:

- ▶ Gaussian white noise
- ▶ uniform white noise
- ▶ ...

White Processes (White Noise)

White noise process:

- ▶ zero-mean: $E[x[n]] = 0$
- ▶ uncorrelated: $E[x[n]x[m]] = E[x[n]]E[x[m]]$ for $m \neq n$
- ▶ autocorrelation $r_x[n] = \sigma_x^2 \delta[n]$

According to underlying distribution:

- ▶ Gaussian white noise
- ▶ uniform white noise
- ▶ ...

The coin-toss process

For each new sample, toss a fair coin:

$$x[n] = \begin{cases} +1 & \text{if the outcome of the } n\text{-th toss is head} \\ -1 & \text{if the outcome of the } n\text{-th toss is tail} \end{cases}$$

- ▶ each sample is independent from all others
- ▶ each sample value has a 50% probability: $f_x(x) = \delta(x \pm 1)/2$

white noise process with $r_x[n] = \delta[n]$

Computing the moments: theory

With access to the theoretical univariate and bivariate pdfs of the WSS process:

$$m_x = E[x[n]] = \int_{-\infty}^{\infty} af_{x[0]}(a)da$$

$$r_x[k] = E[x[0]x[k]] = \iint_{-\infty}^{\infty} ab f_{x[0]x[k]}(a, b) da db$$

Computing the moments: ensemble averages

With access to M realizations of the WSS process:

$$m_x \approx \frac{1}{M} \sum_{i=0}^{M-1} \check{x}_i[n]$$

$$r_x[k] \approx \frac{1}{M} \sum_{i=0}^{M-1} \check{x}_i[n] \check{x}_i[n+k]$$

Computing the moments: time averages

With access to M samples of a single realization of the WSS process:

$$m_x \approx \frac{1}{M} \sum_{n=0}^{M-1} \check{x}[n]$$

$$r_x[k] \approx \frac{1}{M} \sum_{n=0}^{M-|k|-1} \check{x}[n] \check{x}[n+|k|]$$

- ▶ processes for which this works are called *ergodic*
- ▶ at least $M > 4k_{\max}$

Orthogonality

Correlation (via ensemble average):

$$\mathbb{E}[xy] = \lim_{M \rightarrow \infty} \frac{1}{M} \sum_{i=0}^{M-1} \check{x}_i \check{y}_i.$$

Inner product in $\ell_2(\mathbb{Z})$:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{n=-\infty}^{\infty} x_n y_n.$$

Orthogonality

Correlation (via ensemble average):

$$\mathbb{E}[xy] = \lim_{M \rightarrow \infty} \frac{1}{M} \sum_{i=0}^{M-1} \check{x}_i \check{y}_i.$$

Inner product in $\ell_2(\mathbb{Z})$:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{n=-\infty}^{\infty} x_n y_n.$$

Orthogonality

$$E[xy] = 0 \implies \text{orthogonal random variables}$$

- ▶ if x, y zero mean: orthogonal = uncorrelated
- ▶ no linear relationship between variables
- ▶ variables are maximally different

spectral representation of random processes

A simple discrete-time random signal generator

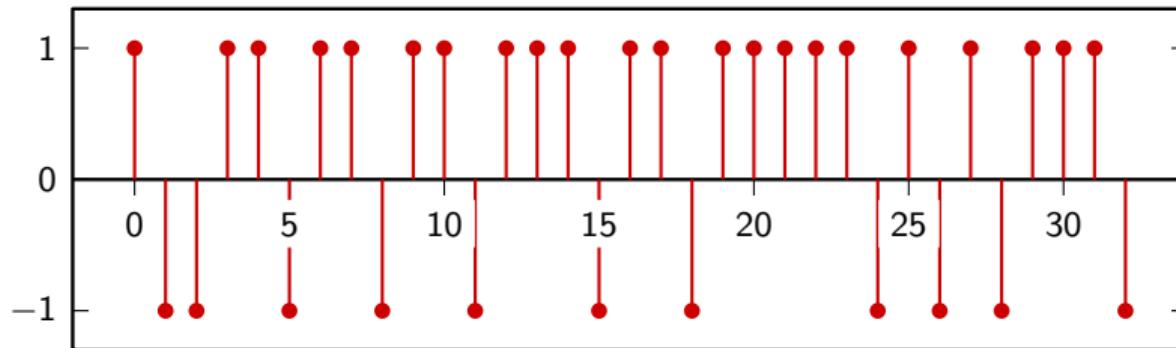
For each new sample, toss a fair coin:

$$x[n] = \begin{cases} +1 & \text{if the outcome of the } n\text{-th toss is head} \\ -1 & \text{if the outcome of the } n\text{-th toss is tail} \end{cases}$$

- ▶ each sample is independent from all others
- ▶ each sample value has a 50% probability

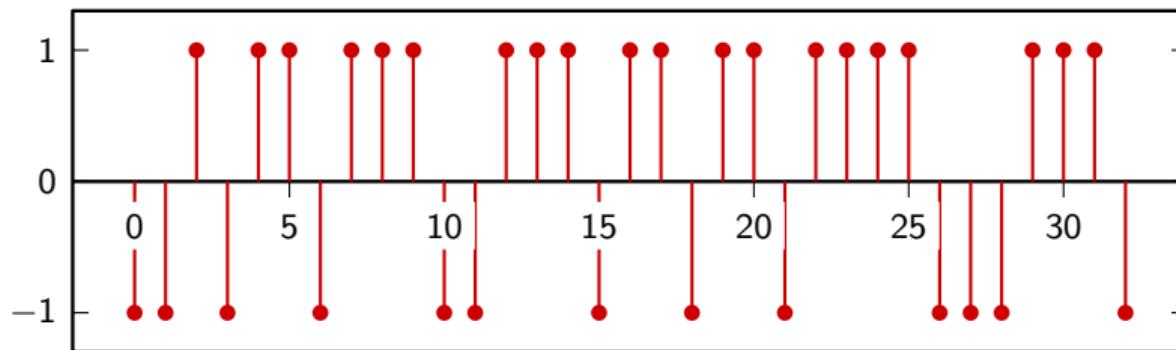
A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



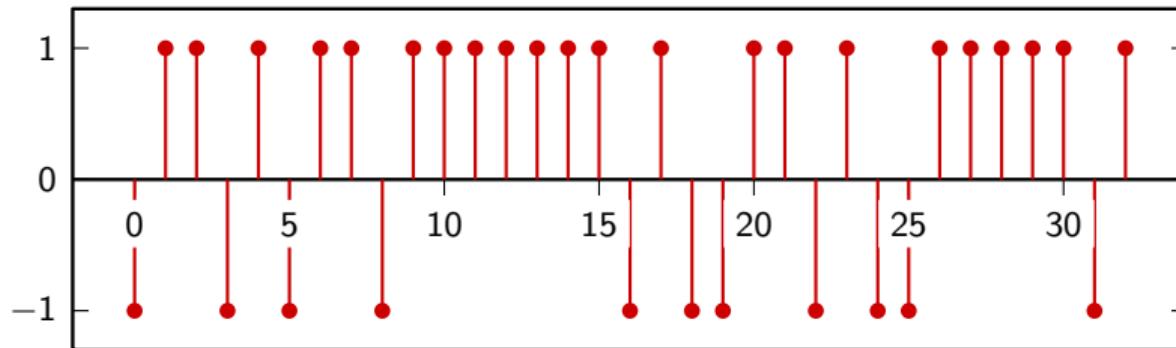
A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



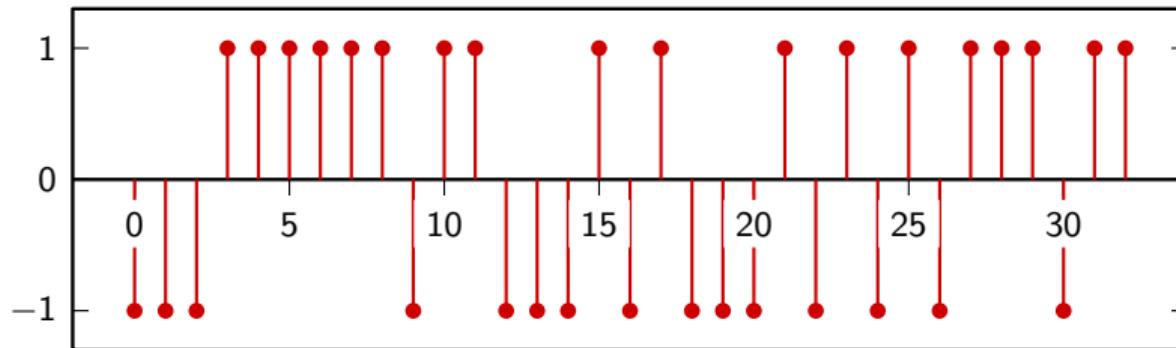
A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal

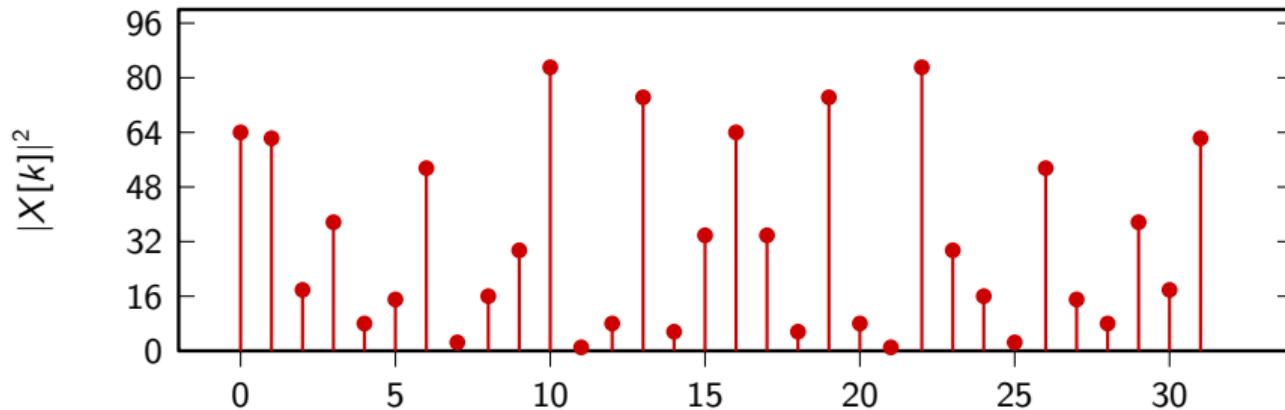


A simple discrete-time random signal generator

- ▶ every time we turn on the generator we obtain a different *realization* of the signal
- ▶ we know the “mechanism” behind each instance
- ▶ but how can we analyze a random signal? What about its frequency content?

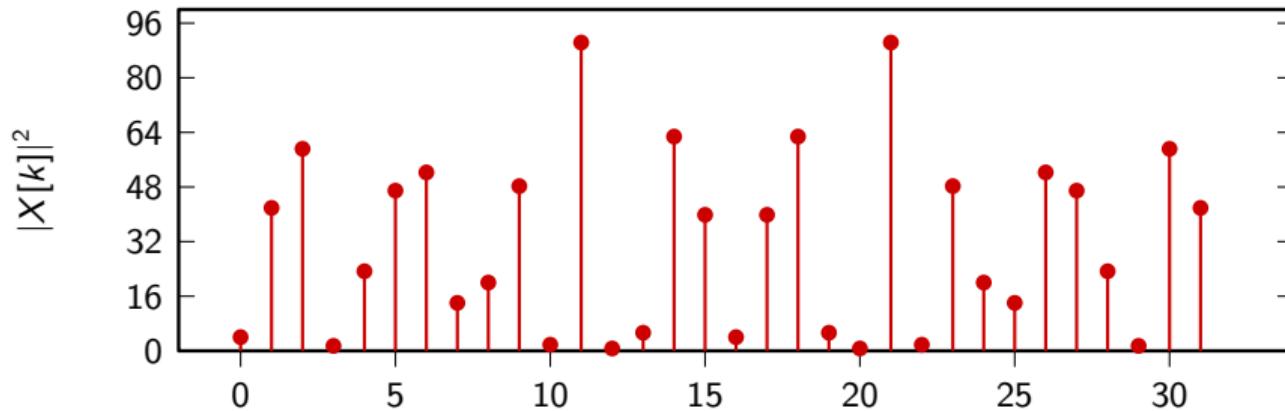
Spectral properties?

let's try with the DFT of a finite set of random samples



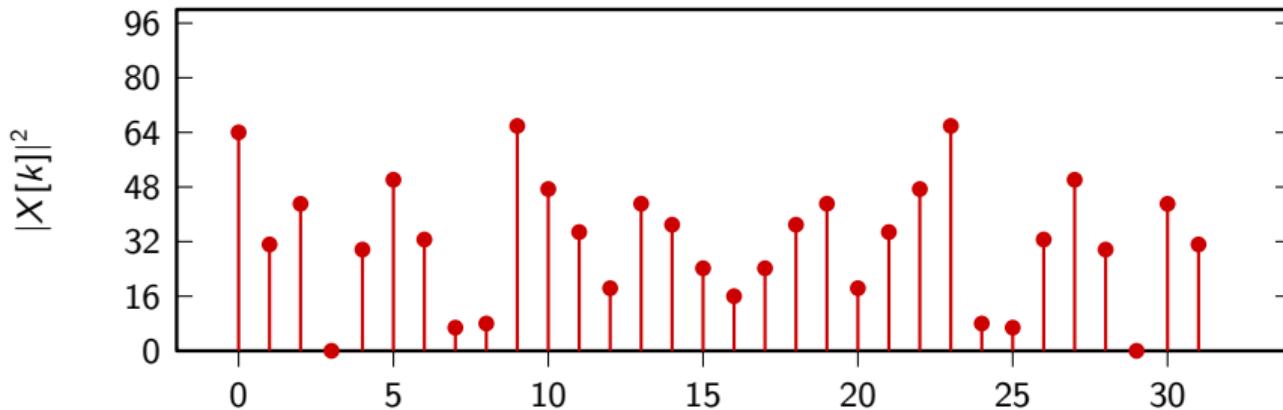
Spectral properties?

let's try with the DFT of a finite set of random samples



Spectral properties?

let's try with the DFT of a finite set of random samples

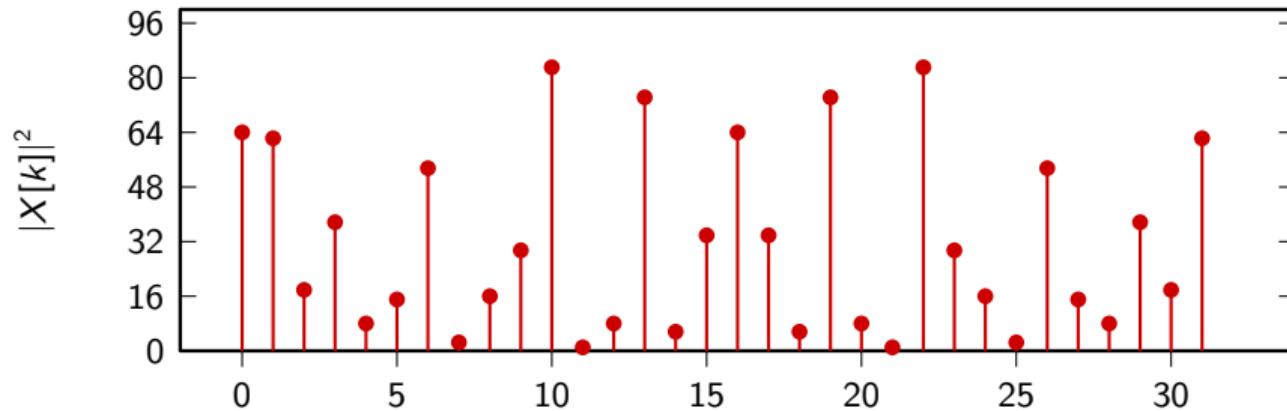


Spectral properties?

every time it's different; maybe with more data?

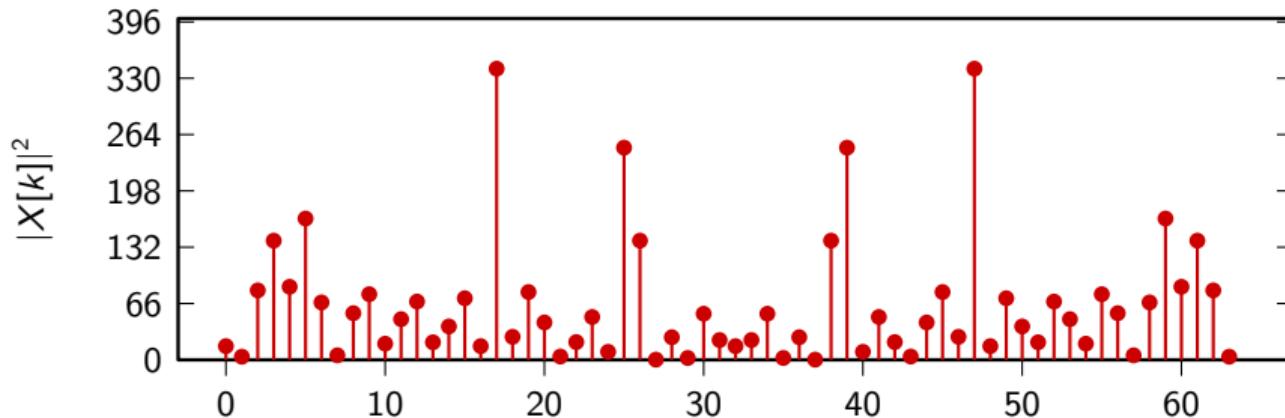
Spectral properties?

DFT of an increasing number of samples



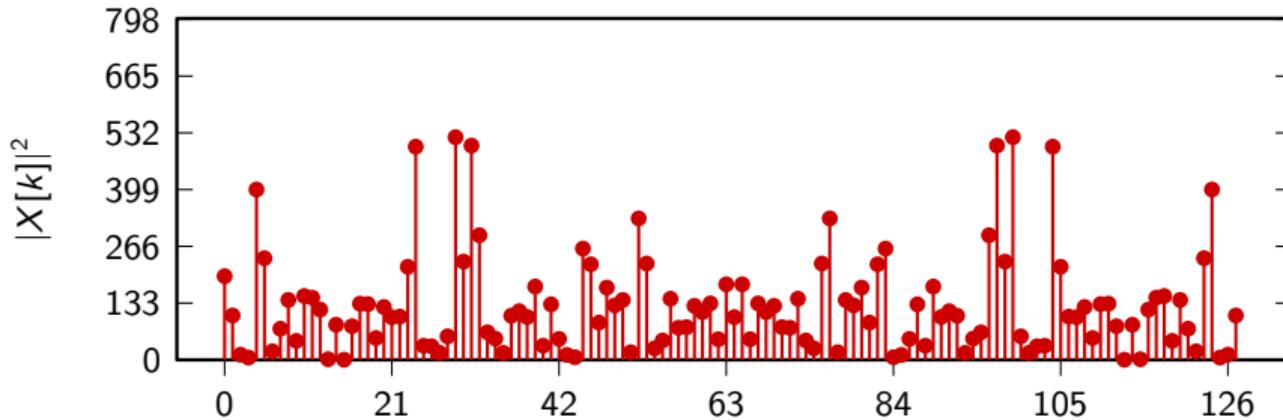
Spectral properties?

DFT of an increasing number of samples



Spectral properties?

DFT of an increasing number of samples



Averaging

- ▶ DFTs of realizations show no clear pattern... we need a new strategy
- ▶ when faced with random data an intuitive response is to take “averages” (i.e. expectation)
- ▶ for the coin-toss signal:

$$E[x[n]] = -1 \cdot P[\text{n-th toss is tail}] + 1 \cdot P[\text{n-th toss is head}] = 0$$

- ▶ so the average value for each sample is zero...

Averaging

- ▶ DFTs of realizations show no clear pattern... we need a new strategy
- ▶ when faced with random data an intuitive response is to take “averages” (i.e. expectation)
- ▶ for the coin-toss signal:

$$E[x[n]] = -1 \cdot P[\text{n-th toss is tail}] + 1 \cdot P[\text{n-th toss is head}] = 0$$

- ▶ so the average value for each sample is zero...

Averaging

- ▶ DFTs of realizations show no clear pattern... we need a new strategy
- ▶ when faced with random data an intuitive response is to take “averages” (i.e. expectation)
- ▶ for the coin-toss signal:

$$E[x[n]] = -1 \cdot P[\text{n-th toss is tail}] + 1 \cdot P[\text{n-th toss is head}] = 0$$

- ▶ so the average value for each sample is zero...

Averaging

- ▶ DFTs of realizations show no clear pattern... we need a new strategy
- ▶ when faced with random data an intuitive response is to take “averages” (i.e. expectation)
- ▶ for the coin-toss signal:

$$E[x[n]] = -1 \cdot P[\text{n-th toss is tail}] + 1 \cdot P[\text{n-th toss is head}] = 0$$

- ▶ so the average value for each sample is zero...

Averaging the DFT

- ▶ but the DFT is linear so $E[\text{DFT}\{x[n]\}] = \text{DFT}\{E[x[n]]\} = 0$
- ▶ however the signal “moves”, so its energy or power must be nonzero

Averaging the DFT

- ▶ but the DFT is linear so $E[\text{DFT}\{x[n]\}] = \text{DFT}\{E[x[n]]\} = 0$
- ▶ however the signal “moves”, so its energy or power must be nonzero

Energy and power

- ▶ the coin-toss process produces realizations with infinite energy:

$$E_x = \lim_{N \rightarrow \infty} \sum_{n=-N}^N |\check{x}[n]|^2 = \lim_{N \rightarrow \infty} (2N + 1) = \infty$$

- ▶ which, however, have has finite *power*:

$$P_x = \lim_{N \rightarrow \infty} \frac{1}{2N + 1} \sum_{n=-N}^N |\check{x}[n]|^2 = 1$$

Averaging

let's try to average the DFT's square magnitude, normalized:

- ▶ pick an interval length N
- ▶ pick a number of iterations M
- ▶ run the signal generator M times and obtain M N -point realizations
- ▶ compute the DFT of each realization
- ▶ average their square magnitude divided by N

Averaging

let's try to average the DFT's square magnitude, normalized:

- ▶ pick an interval length N
- ▶ pick a number of iterations M
- ▶ run the signal generator M times and obtain M N -point realizations
- ▶ compute the DFT of each realization
- ▶ average their square magnitude divided by N

Averaging

let's try to average the DFT's square magnitude, normalized:

- ▶ pick an interval length N
- ▶ pick a number of iterations M
- ▶ run the signal generator M times and obtain M N -point realizations
- ▶ compute the DFT of each realization
- ▶ average their square magnitude divided by N

Averaging

let's try to average the DFT's square magnitude, normalized:

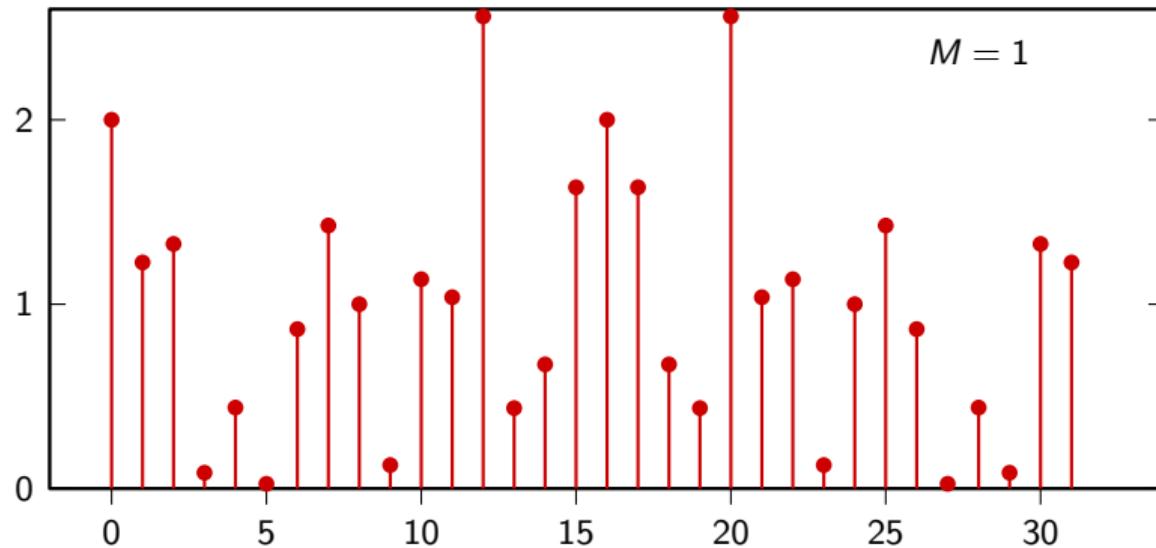
- ▶ pick an interval length N
- ▶ pick a number of iterations M
- ▶ run the signal generator M times and obtain M N -point realizations
- ▶ compute the DFT of each realization
- ▶ average their square magnitude divided by N

Averaging

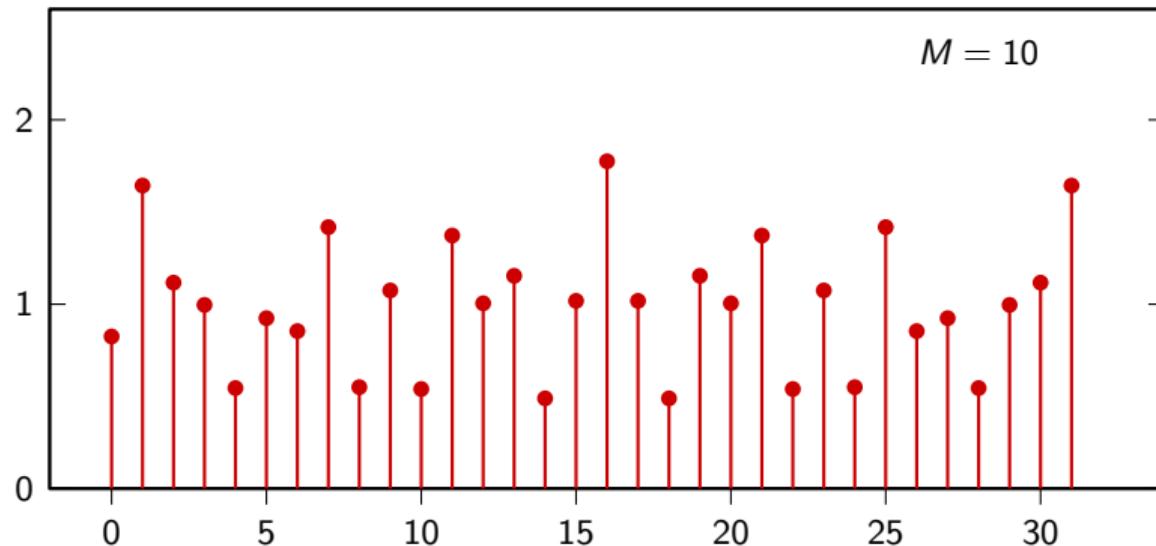
let's try to average the DFT's square magnitude, normalized:

- ▶ pick an interval length N
- ▶ pick a number of iterations M
- ▶ run the signal generator M times and obtain M N -point realizations
- ▶ compute the DFT of each realization
- ▶ average their square magnitude divided by N

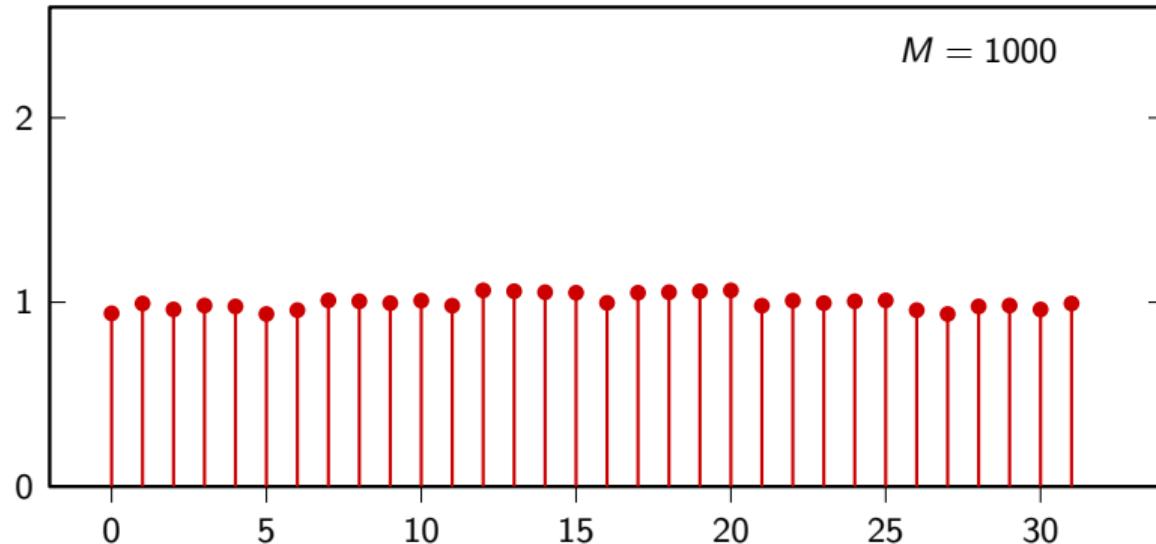
Averaged DFT square magnitude



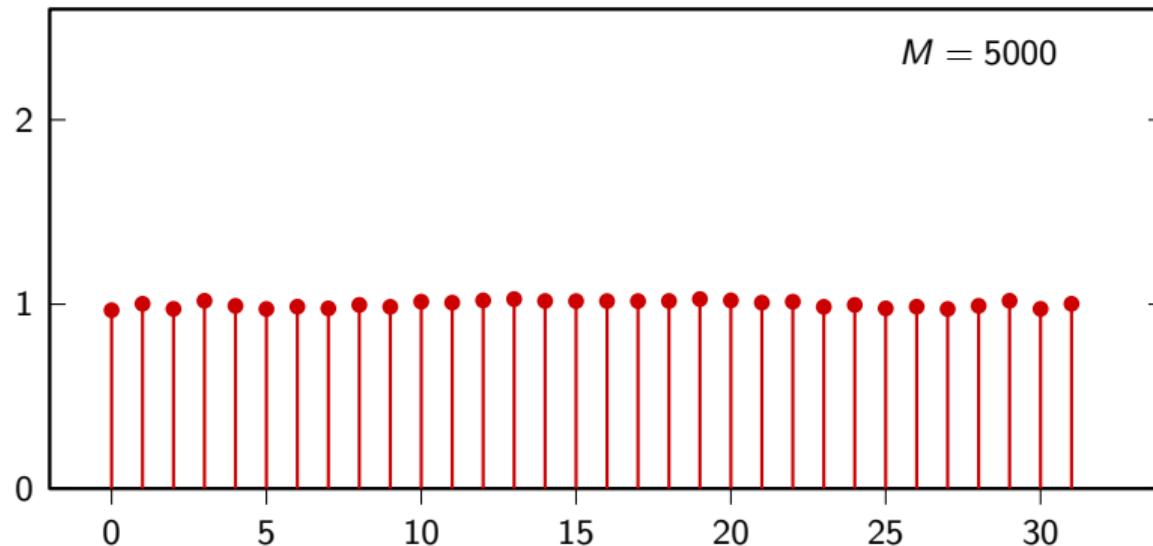
Averaged DFT square magnitude



Averaged DFT square magnitude



Averaged DFT square magnitude



Power spectral density

$$P[k] = \mathbb{E} [|X_N[k]|^2 / N]$$

- ▶ it looks very much as if $P[k] = 1$
- ▶ if $|X_N[k]|^2$ tends to the *energy* distribution in frequency...
- ▶ ... $|X_N[k]|^2 / N$ tends to the *power* distribution (aka *density*) in frequency
- ▶ the frequency-domain representation for stochastic processes is the power spectral density

Power spectral density

$$P[k] = \mathbb{E} [|X_N[k]|^2 / N]$$

- ▶ it looks very much as if $P[k] = 1$
- ▶ if $|X_N[k]|^2$ tends to the *energy* distribution in frequency...
- ▶ ... $|X_N[k]|^2 / N$ tends to the *power* distribution (aka *density*) in frequency
- ▶ the frequency-domain representation for stochastic processes is the power spectral density

Power spectral density

$$P[k] = \mathbb{E} [|X_N[k]|^2 / N]$$

- ▶ it looks very much as if $P[k] = 1$
- ▶ if $|X_N[k]|^2$ tends to the *energy* distribution in frequency...
- ▶ ... $|X_N[k]|^2 / N$ tends to the *power* distribution (aka *density*) in frequency
- ▶ the frequency-domain representation for stochastic processes is the power spectral density

Power spectral density

$$P[k] = \mathbb{E} [|X_N[k]|^2 / N]$$

- ▶ it looks very much as if $P[k] = 1$
- ▶ if $|X_N[k]|^2$ tends to the *energy* distribution in frequency...
- ▶ ... $|X_N[k]|^2 / N$ tends to the *power* distribution (aka *density*) in frequency
- ▶ the frequency-domain representation for stochastic processes is the power spectral density

Power spectral density: intuition

- ▶ $P[k] = 1$ means that the power is equally distributed over all frequencies
- ▶ i.e., we cannot predict if the signal moves “slowly” or “super-fast”
- ▶ this is because each sample is independent of each other: we could have a realization of all ones or a realization in which the sign changes every other sample or anything in between

Power spectral density: intuition

- ▶ $P[k] = 1$ means that the power is equally distributed over all frequencies
- ▶ i.e., we cannot predict if the signal moves “slowly” or “super-fast”
- ▶ this is because each sample is independent of each other: we could have a realization of all ones or a realization in which the sign changes every other sample or anything in between

Power spectral density: intuition

- ▶ $P[k] = 1$ means that the power is equally distributed over all frequencies
- ▶ i.e., we cannot predict if the signal moves “slowly” or “super-fast”
- ▶ this is because each sample is independent of each other: we could have a realization of all ones or a realization in which the sign changes every other sample or anything in between

Filtering a random process

- ▶ let's filter the random process with a 2-point Moving Average filter
- ▶ $y[n] = (x[n] + x[n - 1])/2$
- ▶ what is the power spectral density?

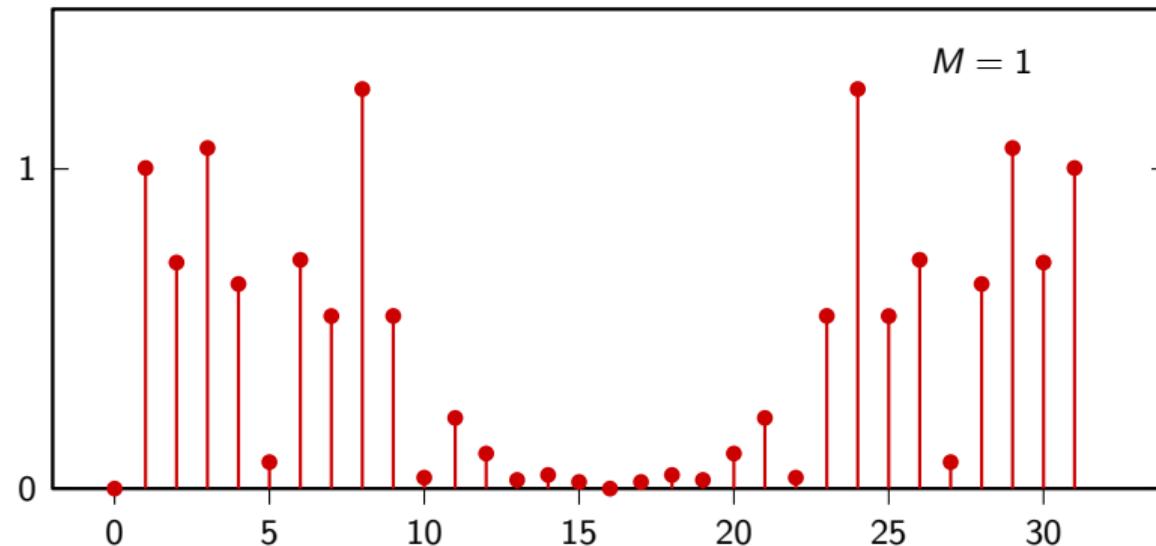
Filtering a random process

- ▶ let's filter the random process with a 2-point Moving Average filter
- ▶ $y[n] = (x[n] + x[n - 1])/2$
- ▶ what is the power spectral density?

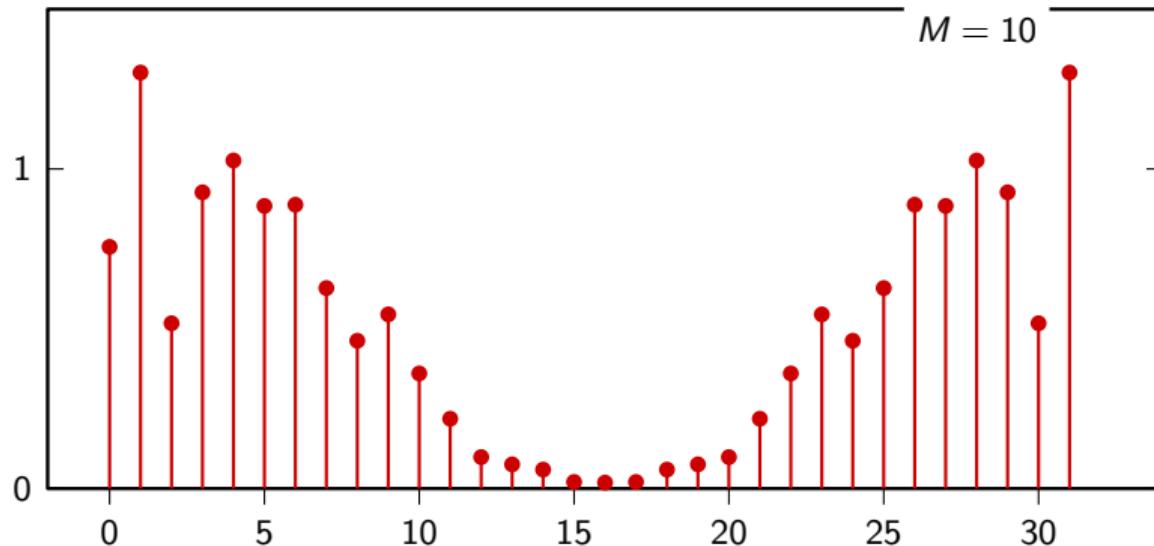
Filtering a random process

- ▶ let's filter the random process with a 2-point Moving Average filter
- ▶ $y[n] = (x[n] + x[n - 1])/2$
- ▶ what is the power spectral density?

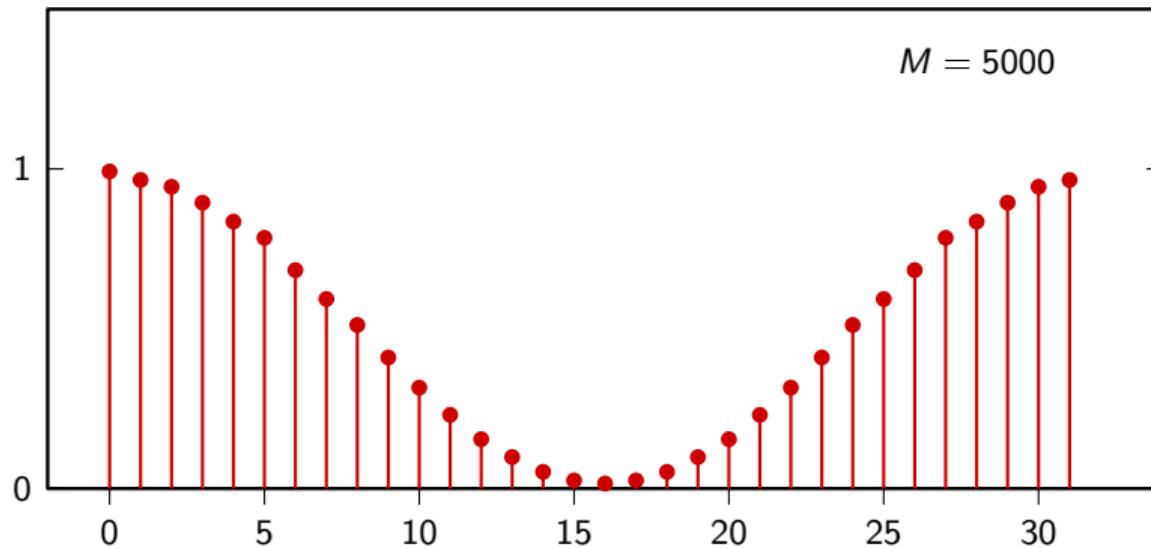
Averaged DFT magnitude of filtered process



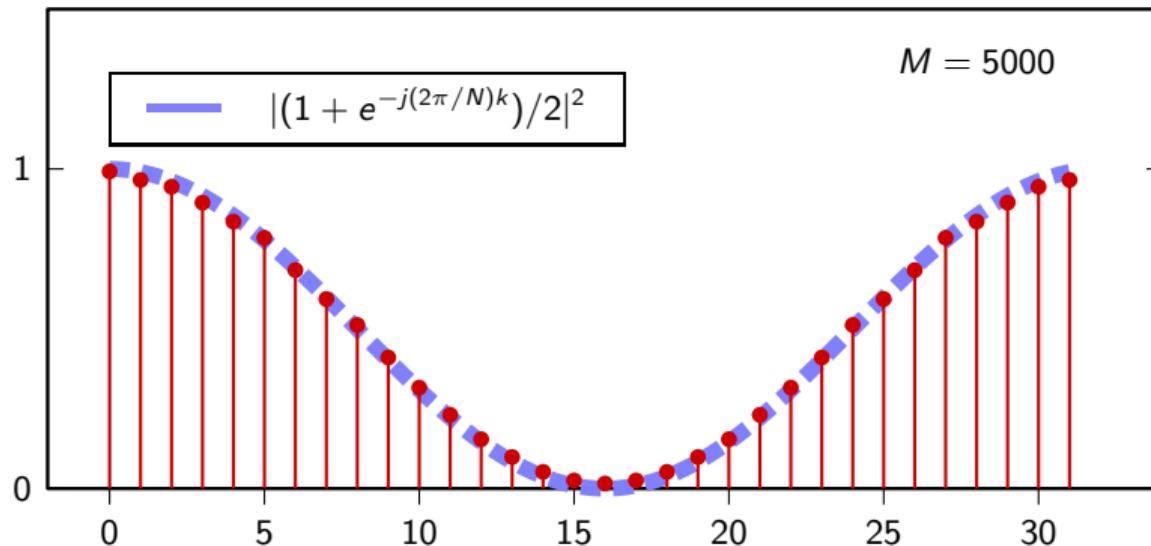
Averaged DFT magnitude of filtered process



Averaged DFT magnitude of filtered process



Averaged DFT magnitude of filtered process



Filtering a random process

- ▶ it looks like $P_y[k] = P_x[k] |H[k]|^2$, where $H[k] = \text{DFT} \{h[n]\}$
- ▶ can we generalize these results beyond a finite set of samples?

Filtering a random process

- ▶ it looks like $P_y[k] = P_x[k] |H[k]|^2$, where $H[k] = \text{DFT} \{h[n]\}$
- ▶ can we generalize these results beyond a finite set of samples?

Energy and Power Signals

- ▶ energy signals: $\sum_{n=-\infty}^{\infty} |x[n]|^2 < \infty$
- ▶ power signals: $\lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N |x[n]|^2 < \infty$

Energy Signals

- ▶ finite support, $\text{sinc}(n)$, $\alpha^n u[n]$ for $|\alpha| < 1$, ...
- ▶ DTFT is well defined
- ▶ DTFT square magnitude is *energy* distribution in frequency

Power Signals

- ▶ $x[n] = 1, u[n], e^{j\omega n}, \sin, \cos, \dots$
- ▶ DTFT uses the Dirac delta formalism
- ▶ “DTFT square magnitude” doesn’t make sense!

Power Spectral Density

Consider a truncated DTFT

$$X_N(e^{j\omega}) = \sum_{n=-N}^N x[n]e^{-j\omega n}$$

define the power spectral density of a signal as:

$$P(e^{j\omega}) = \lim_{N \rightarrow \infty} \frac{1}{2N+1} |X_N(e^{j\omega})|^2$$

Power Spectral Density

Consider a truncated DTFT

$$X_N(e^{j\omega}) = \sum_{n=-N}^N x[n]e^{-j\omega n}$$

define the power spectral density of a signal as:

$$P(e^{j\omega}) = \lim_{N \rightarrow \infty} \frac{1}{2N+1} |X_N(e^{j\omega})|^2$$

Power Spectral Density

Examples:

- ▶ $x[n] = a, P_x(e^{j\omega}) = a^2 \tilde{\delta}(\omega)$
- ▶ $x[n] = ae^{j\sigma n}, P_x(e^{j\omega}) = a^2 \tilde{\delta}(\omega - \sigma)$
- ▶ $x[n] = au[n], P_x(e^{j\omega}) = (a^2/2)\tilde{\delta}(\omega)$

Power Spectral Density

Examples:

- ▶ $x[n] = a, P_x(e^{j\omega}) = a^2 \tilde{\delta}(\omega)$
- ▶ $x[n] = ae^{j\sigma n}, P_x(e^{j\omega}) = a^2 \tilde{\delta}(\omega - \sigma)$
- ▶ $x[n] = au[n], P_x(e^{j\omega}) = (a^2/2) \tilde{\delta}(\omega)$

Power Spectral Density

Examples:

- ▶ $x[n] = a, P_x(e^{j\omega}) = a^2 \tilde{\delta}(\omega)$
- ▶ $x[n] = ae^{j\sigma n}, P_x(e^{j\omega}) = a^2 \tilde{\delta}(\omega - \sigma)$
- ▶ $x[n] = au[n], P_x(e^{j\omega}) = (a^2/2)\tilde{\delta}(\omega)$

Power Spectral Density for WSS Processes

For a random process

$$P_x(e^{j\omega}) = \lim_{N \rightarrow \infty} \frac{1}{2N+1} E [|X_N(e^{j\omega})|^2]$$

Power Spectral Density for WSS Processes

$$E \left[\left| \sum_{n=-N}^N x[n] e^{-j\omega n} \right|^2 \right] = E \left[\sum_{n=-N}^N x[n] e^{j\omega n} \sum_{m=-N}^N x[m] e^{-j\omega m} \right]$$

Power Spectral Density for WSS Processes

$$\begin{aligned} \mathbb{E} \left[\left| \sum_{n=-N}^N x[n] e^{-j\omega n} \right|^2 \right] &= \mathbb{E} \left[\sum_{n=-N}^N x[n] e^{j\omega n} \sum_{m=-N}^N x[m] e^{-j\omega m} \right] \\ &= \sum_{n=-N}^N \sum_{m=-N}^N \mathbb{E}[x[n]x[m]] e^{-j\omega(m-n)} \end{aligned}$$

Power Spectral Density for WSS Processes

$$\begin{aligned} \mathbb{E} \left[\left| \sum_{n=-N}^N x[n] e^{-j\omega n} \right|^2 \right] &= \mathbb{E} \left[\sum_{n=-N}^N x[n] e^{j\omega n} \sum_{m=-N}^N x[m] e^{-j\omega m} \right] \\ &= \sum_{n=-N}^N \sum_{m=-N}^N \mathbb{E}[x[n]x[m]] e^{-j\omega(m-n)} \\ &= \sum_{n=-N}^N \sum_{m=-N}^N r_x[m-n] e^{-j\omega(m-n)} \end{aligned}$$

WSS

A clever manipulation

$$S = \sum_{m=-N}^N \sum_{n=-N}^N f(m-n)$$

$$-2N \leq (m - n) \leq 2N$$

$$S = \sum_{k=-2N}^{2N} c(k)f(k)$$

A clever manipulation

$$S = \sum_{m=-N}^N \sum_{n=-N}^N f(m-n)$$

$$-2N \leq (m - n) \leq 2N$$

$$S = \sum_{k=-2N}^{2N} c(k)f(k)$$

A clever manipulation

$$S = \sum_{m=-N}^N \sum_{n=-N}^N f(m-n)$$

$$-2N \leq (m - n) \leq 2N$$

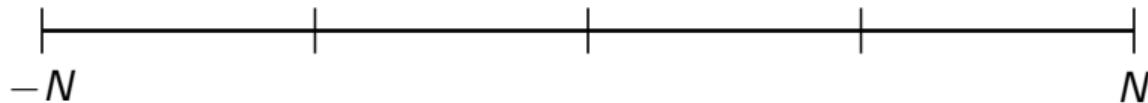
$$S = \sum_{k=-2N}^{2N} c(k)f(k)$$

A clever manipulation

$c(k)$: number of ways we can pick n, m in $[-N, N]$ so that $(m - n) = k$

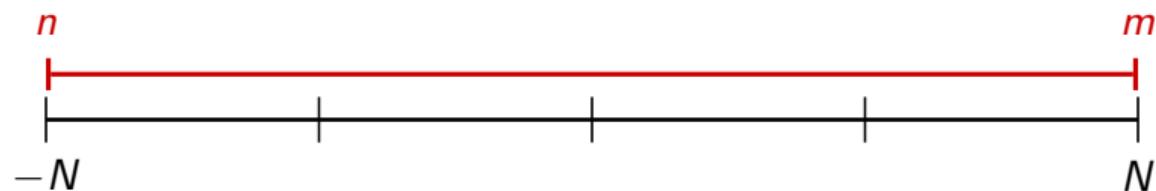
A clever manipulation

geometrically: $c(k) = \text{number of ways we can fit a segment of length } k \text{ over } [-N, N]:$



A clever manipulation

geometrically: $c(k) = \text{number of ways we can fit a segment of length } k \text{ over } [-N, N]:$



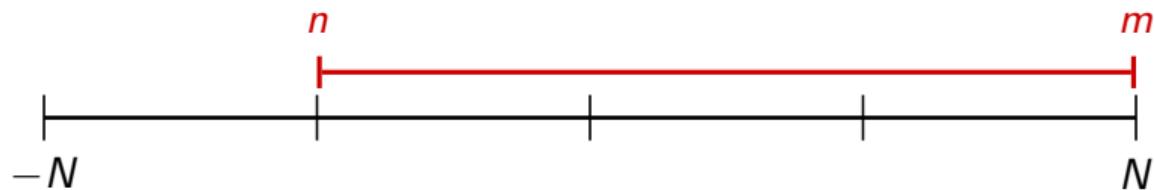
A clever manipulation

geometrically: $c(k) = \text{number of ways we can fit a segment of length } k \text{ over } [-N, N]:$



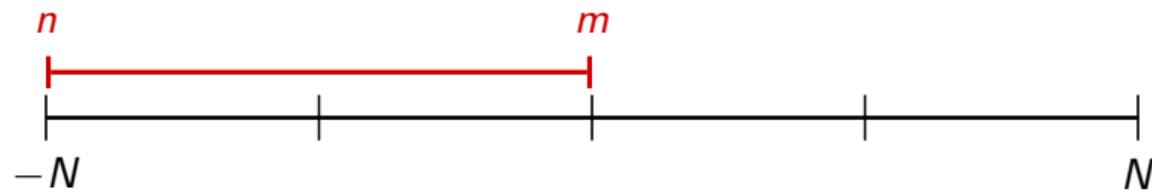
A clever manipulation

geometrically: $c(k) = \text{number of ways we can fit a segment of length } k \text{ over } [-N, N]:$



A clever manipulation

geometrically: $c(k) = \text{number of ways we can fit a segment of length } k \text{ over } [-N, N]:$



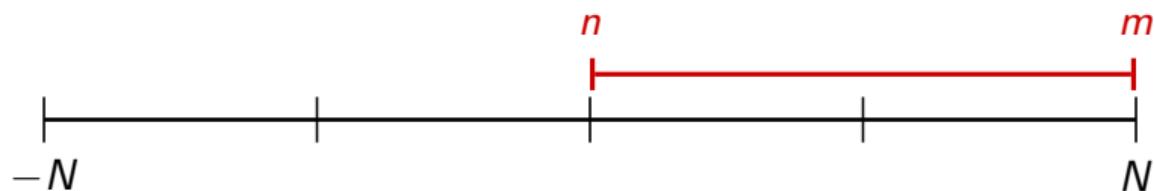
A clever manipulation

geometrically: $c(k) = \text{number of ways we can fit a segment of length } k \text{ over } [-N, N]:$



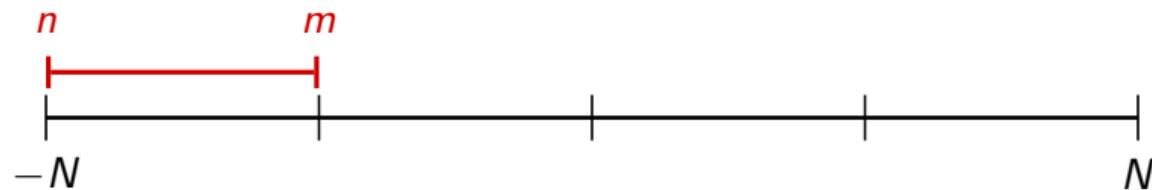
A clever manipulation

geometrically: $c(k) = \text{number of ways we can fit a segment of length } k \text{ over } [-N, N]:$



A clever manipulation

geometrically: $c(k) = \text{number of ways we can fit a segment of length } k \text{ over } [-N, N]:$



A clever manipulation

$c(k)$: number of ways we can pick n, m in $[-N, N]$ so that $(m - n) = k$

$$c(k) = 2N + 1 - |k|$$

Power Spectral Density for WSS Processes

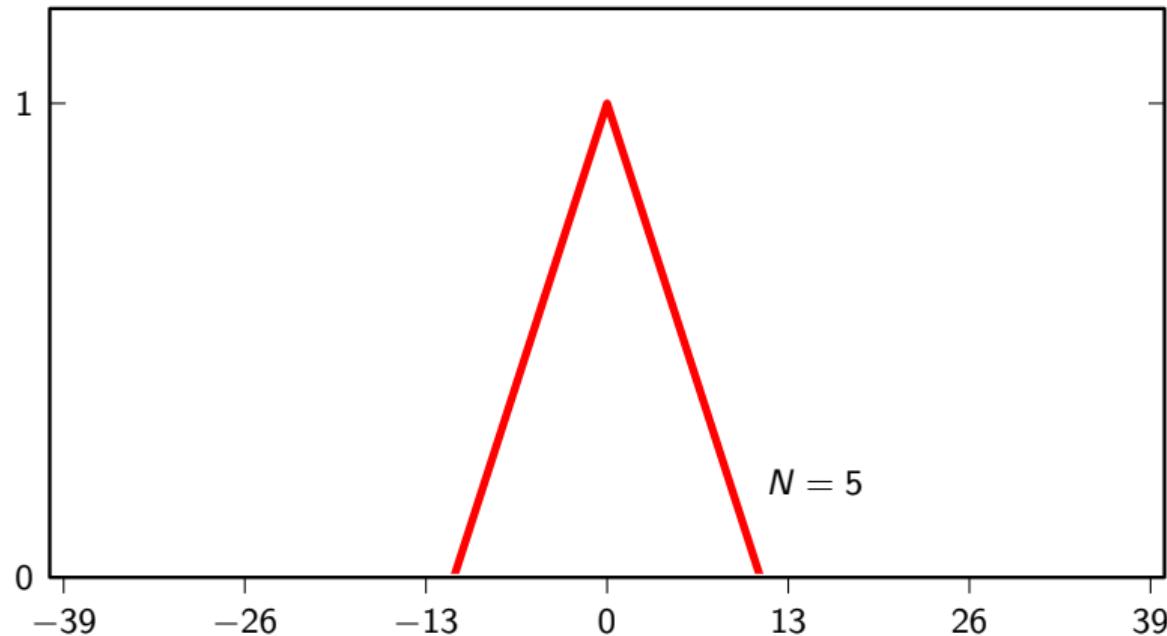
$$E \left[\left| \sum_{n=-N}^N x[n] e^{-j\omega n} \right|^2 \right] = \sum_{k=-2N}^{2N} (2N+1 - |k|) r_x[k] e^{-j\omega k}$$

Power Spectral Density for WSS Processes

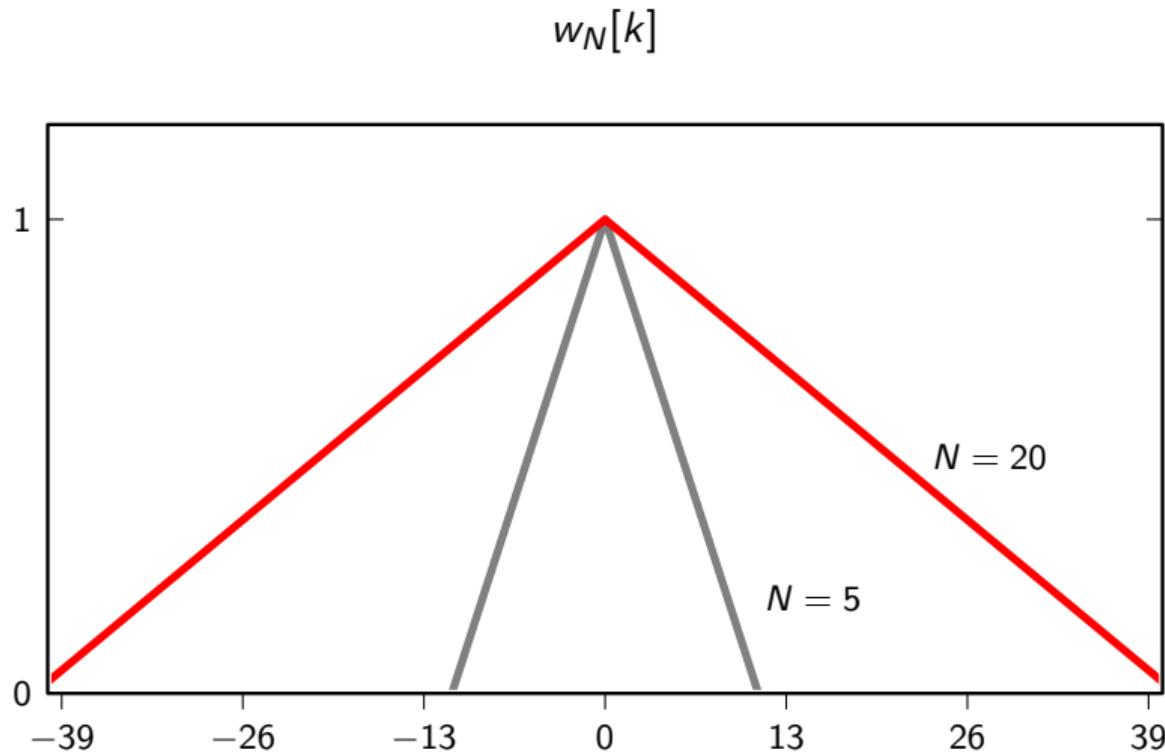
$$\begin{aligned} P_x(e^{j\omega}) &= \lim_{N \rightarrow \infty} \sum_{k=-2N}^{2N} \left(\frac{2N+1-|k|}{2N+1} \right) (r_X[k]e^{-j\omega k}) \\ &= \lim_{N \rightarrow \infty} \sum_{k=-2N}^{2N} w_N[k]r_X[k]e^{-j\omega k} \end{aligned}$$

Power Spectral Density for WSS Processes

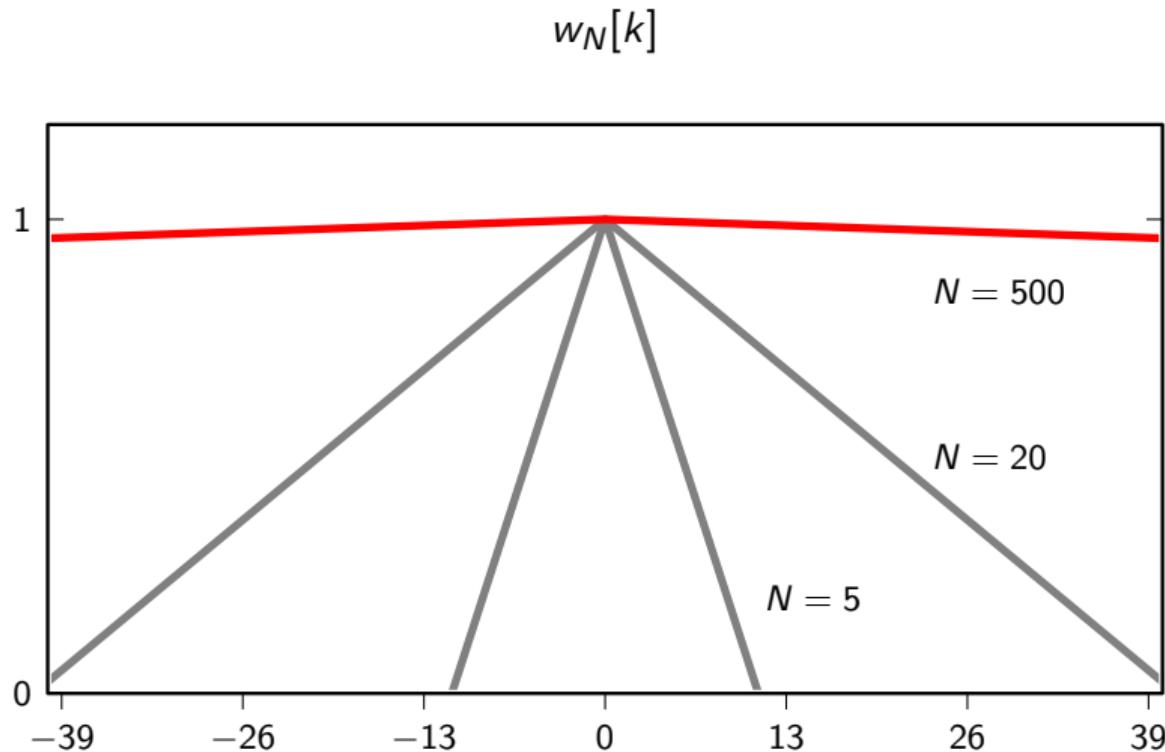
$w_N[k]$



Power Spectral Density for WSS Processes



Power Spectral Density for WSS Processes



Power Spectral Density for WSS Processes

$$\lim_{N \rightarrow \infty} w_N[k] = 1$$

Power Spectral Density for WSS Processes

$$\begin{aligned} P_x(e^{j\omega}) &= \lim_{N \rightarrow \infty} \sum_{k=-2N}^{2N} w_N[k] r_x[k] e^{-j\omega k} \\ &= \sum_{k=-\infty}^{\infty} r_x[k] e^{-j\omega k} \\ &= \text{DTFT}\{r_x[k]\} \end{aligned}$$

Noise

- ▶ noise is everywhere:
 - thermal noise
 - sum of extraneous interferences
 - quantization and numerical errors
 - ...
- ▶ we can model noise as a stochastic signal
- ▶ the most important noise is white noise

Noise

- ▶ noise is everywhere:
 - thermal noise
 - sum of extraneous interferences
 - quantization and numerical errors
 - ...
- ▶ we can model noise as a stochastic signal
- ▶ the most important noise is white noise

Noise

- ▶ noise is everywhere:
 - thermal noise
 - sum of extraneous interferences
 - quantization and numerical errors
 - ...
- ▶ we can model noise as a stochastic signal
- ▶ the most important noise is white noise

Noise

- ▶ noise is everywhere:
 - thermal noise
 - sum of extraneous interferences
 - quantization and numerical errors
 - ...
- ▶ we can model noise as a stochastic signal
- ▶ the most important noise is white noise

Noise

- ▶ noise is everywhere:
 - thermal noise
 - sum of extraneous interferences
 - quantization and numerical errors
 - ...
- ▶ we can model noise as a stochastic signal
- ▶ the most important noise is white noise

Noise

- ▶ noise is everywhere:
 - thermal noise
 - sum of extraneous interferences
 - quantization and numerical errors
 - ...
- ▶ we can model noise as a stochastic signal
- ▶ the most important noise is white noise

Noise

- ▶ noise is everywhere:
 - thermal noise
 - sum of extraneous interferences
 - quantization and numerical errors
 - ...
- ▶ we can model noise as a stochastic signal
- ▶ the most important noise is white noise

PSD of white noise

White noise:

- ▶ $m = 0$
- ▶ $r[k] = \sigma^2 \delta[k]$

$$P(e^{j\omega}) = \sigma^2$$

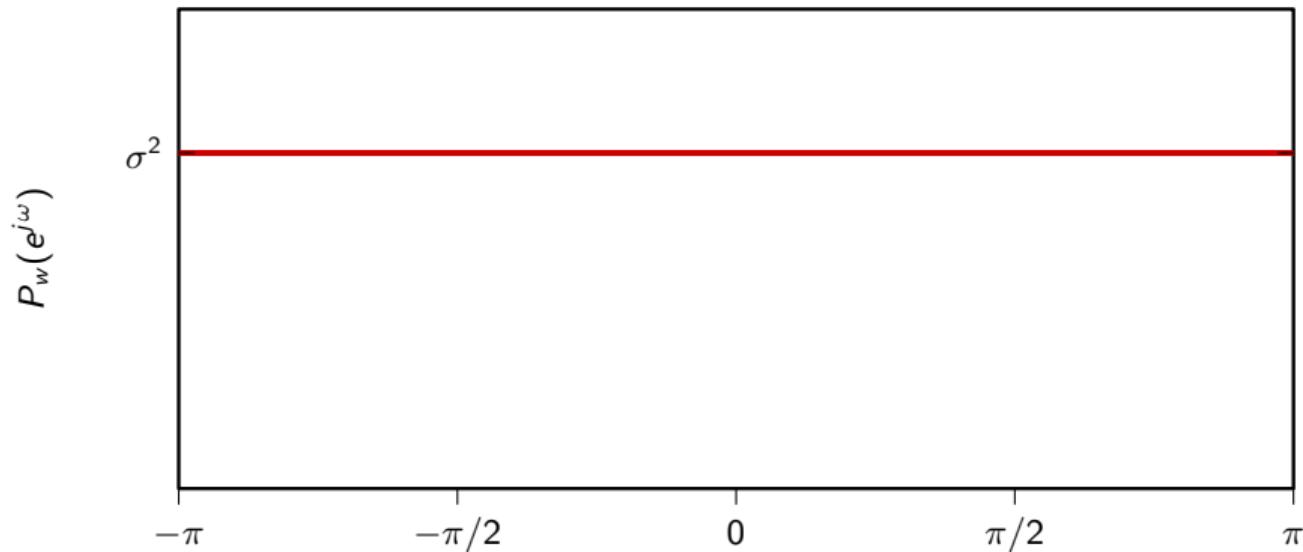
PSD of white noise

White noise:

- ▶ $m = 0$
- ▶ $r[k] = \sigma^2 \delta[k]$

$$P(e^{j\omega}) = \sigma^2$$

PSD of white noise



White noise

- ▶ the PSD is independent of the probability distribution of the single samples (depends only on the variance)
- ▶ distribution is important to estimate bounds for the signal
- ▶ very often a Gaussian distribution models the experimental data the best
- ▶ AWGN: additive white Gaussian noise

White noise

- ▶ the PSD is independent of the probability distribution of the single samples (depends only on the variance)
- ▶ distribution is important to estimate bounds for the signal
 - ▶ very often a Gaussian distribution models the experimental data the best
 - ▶ AWGN: additive white Gaussian noise

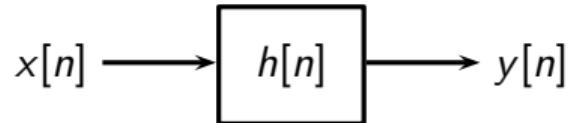
White noise

- ▶ the PSD is independent of the probability distribution of the single samples (depends only on the variance)
- ▶ distribution is important to estimate bounds for the signal
- ▶ very often a Gaussian distribution models the experimental data the best
- ▶ AWGN: additive white Gaussian noise

White noise

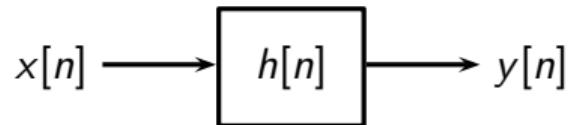
- ▶ the PSD is independent of the probability distribution of the single samples (depends only on the variance)
- ▶ distribution is important to estimate bounds for the signal
- ▶ very often a Gaussian distribution models the experimental data the best
- ▶ AWGN: additive white Gaussian noise

Filtering a Random Process



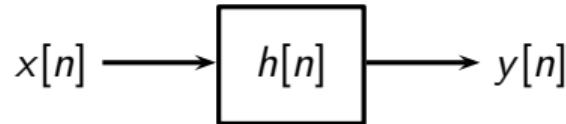
- ▶ is $y[n]$ a random process?
- ▶ if $x[n]$ WSS, is $y[n]$ WSS?
- ▶ what are m_y and $r_y[n]$?

Filtering a Random Process



- ▶ is $y[n]$ a random process?
- ▶ if $x[n]$ WSS, is $y[n]$ WSS?
- ▶ what are m_y and $r_y[n]$?

Filtering a Random Process



- ▶ is $y[n]$ a random process?
- ▶ if $x[n]$ WSS, is $y[n]$ WSS?
- ▶ what are m_y and $r_y[n]$?

Mean of the Filtered Process

$$\begin{aligned}m_{y[n]} &= \mathbb{E}[y[n]] = \mathbb{E}\left[\sum_{k=-\infty}^{\infty} h[k]x[n-k]\right] \\&= \sum_{k=-\infty}^{\infty} h[k]\mathbb{E}[x[n-k]] \\&= \sum_{k=-\infty}^{\infty} h[k]m_x \quad (x[n] \text{ is WSS}) \\&= m_x \sum_{k=-\infty}^{\infty} h[k] \\&= m_x H(e^{j0})\end{aligned}$$

Autocorrelation of the Filtered Process

$$\begin{aligned} \mathbb{E}[y[n]y[m]] &= \mathbb{E}\left[\sum_{k=-\infty}^{\infty} h[k]x[n-k] \sum_{i=-\infty}^{\infty} h[i]x[m-i]\right] \\ &= \sum_{k=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} h[k]h[i]\mathbb{E}[x[n-k]x[m-i]] \\ &= \sum_{k=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} h[k]h[i]r_x[(n-m) - (k+i)] \end{aligned}$$

output depends only on lag $(n - m) \rightarrow y[n]$ is WSS

Fundamental Result

with a change of variable in the double sum:

$$r_y[n] = h[n] * h[-n] * r_x[n]$$

so that:

$$P_y(e^{j\omega}) = |H(e^{j\omega})|^2 P_x(e^{j\omega})$$

Deterministic filters can be used to shape the power distribution of WSS random processes

Stochastic signal processing

key points:

- ▶ filters designed for deterministic signals still work (in magnitude) in the stochastic case
- ▶ we lose the concept of phase since we don't know the shape of a realization in advance

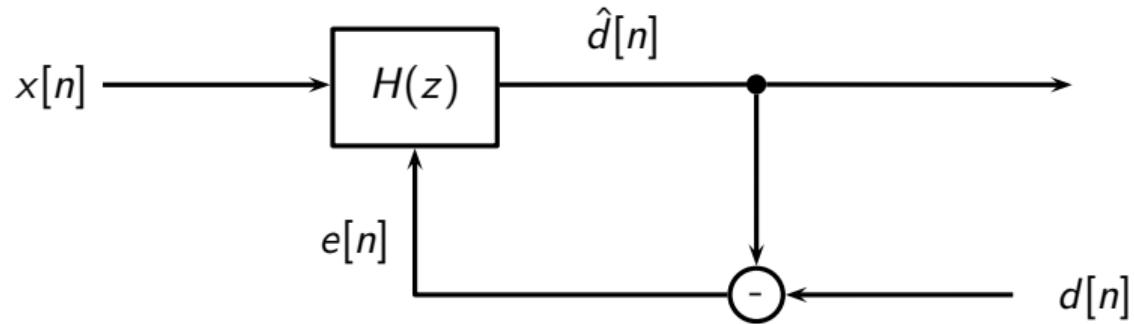
Stochastic signal processing

key points:

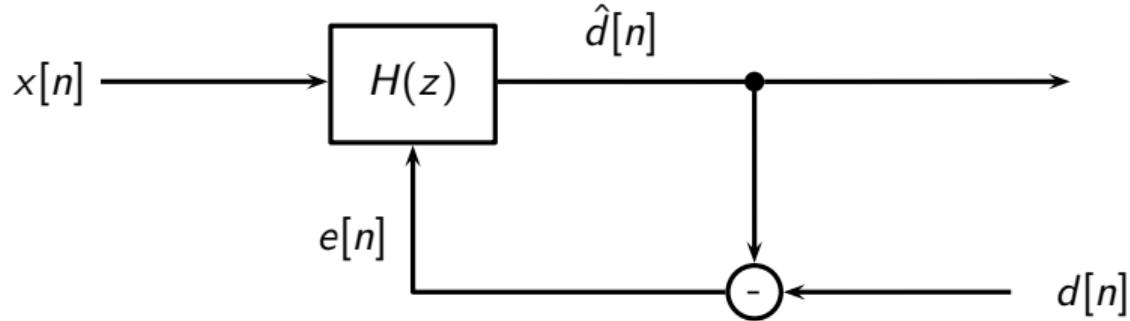
- ▶ filters designed for deterministic signals still work (in magnitude) in the stochastic case
- ▶ we lose the concept of phase since we don't know the shape of a realization in advance

adaptive signal processing

Adaptive signal processing

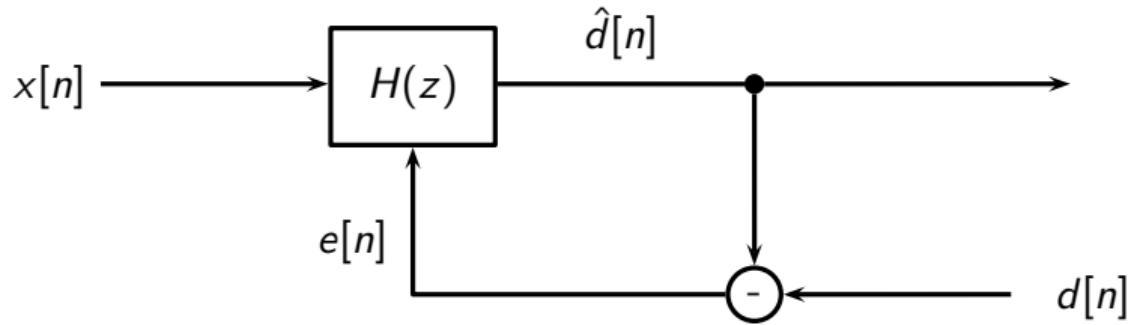


Adaptive signal processing



- ▶ $d[n]$: desired signal
- ▶ $\hat{d}[n]$: adaptive approximation
- ▶ $e[n]$: error signal

Adaptive signal processing



how do we find the filter's coefficients?

Optimal adaptive filter

optimal filter $H(z)$ minimizes the Mean Square Error

$$H(z) = \arg \min_{H(z)} \{E[|e[n]|^2]\}$$

Optimal adaptive filter

$$H(z) = \arg \min_{H(z)} \{E [|e[n]|^2]\}$$

Advantages of a squared error measure:

- ▶ minimum always exist
- ▶ error easily differentiable
- ▶ output will be orthogonal to error
- ▶ only need second moments!

Just FIR adaptive filters for us

Will only consider FIR adaptive filters:

$$\hat{d}[n] = \sum_{k=0}^{N-1} h[k]x[n - k]$$

Finding the minimum squared error

Two cases:

- ▶ for WSS signals, one-shot solution: Optimal Least Squares
- ▶ for “almost” WSS signals, iterative solutions: stochastic gradient descent or LMS

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

Minimum is found by setting all partial derivatives to zero

$$\begin{aligned}\frac{\partial \mathbb{E}[e^2[n]]}{\partial h[i]} &= 2\mathbb{E}\left[e[n]\frac{\partial e[n]}{\partial h[i]}\right] \\ &= -2\mathbb{E}[e[n]x[n-i]] = 0\end{aligned}$$

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

Minimum is found by setting all partial derivatives to zero

$$\begin{aligned}\frac{\partial E[e^2[n]]}{\partial h[i]} &= 2E \left[e[n] \frac{\partial e[n]}{\partial h[i]} \right] \\ &= -2E [e[n] x[n-i]] = 0\end{aligned}$$

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

Minimum is found by setting all partial derivatives to zero

$$\begin{aligned}\frac{\partial E[e^2[n]]}{\partial h[i]} &= 2E \left[e[n] \frac{\partial e[n]}{\partial h[i]} \right] \\ &= -2E [e[n] x[n-i]] = 0\end{aligned}$$

Orthogonality principle

$$E[e[n] x[n - i]] = 0$$

error is orthogonal to all input values we used:
all useful information has been extracted!

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\begin{aligned}\frac{1}{2} \frac{\partial \mathbb{E}[e^2[n]]}{\partial h[i]} &= -\mathbb{E}[e[n]x[n-i]] \\ &= \mathbb{E}\left[\sum_{k=0}^{N-1} h[k]x[n-k]x[n-i]\right] - \mathbb{E}[d[n]x[n-i]] \\ &= \sum_{k=0}^{N-1} h[k]r_{dx}[i-k] - r_{dx}[i] \quad (\text{WSS signals})\end{aligned}$$

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\begin{aligned}\frac{1}{2} \frac{\partial \mathbb{E}[e^2[n]]}{\partial h[i]} &= -\mathbb{E}[e[n]x[n-i]] \\ &= \mathbb{E}\left[\sum_{k=0}^{N-1} h[k]x[n-k]x[n-i]\right] - \mathbb{E}[d[n]x[n-i]] \\ &= \sum_{k=0}^{N-1} h[k]r_{dx}[i-k] - r_{dx}[i] \quad (\text{WSS signals})\end{aligned}$$

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\begin{aligned}\frac{1}{2} \frac{\partial \mathbb{E}[e^2[n]]}{\partial h[i]} &= -\mathbb{E}[e[n]x[n-i]] \\ &= \mathbb{E}\left[\sum_{k=0}^{N-1} h[k]x[n-k]x[n-i]\right] - \mathbb{E}[d[n]x[n-i]] \\ &= \sum_{k=0}^{N-1} h[k]r_{dx}[i-k] - r_{dx}[i] \quad (\text{WSS signals})\end{aligned}$$

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\begin{aligned}\frac{1}{2} \frac{\partial \mathbb{E}[e^2[n]]}{\partial h[i]} &= -\mathbb{E}[e[n]x[n-i]] \\ &= \mathbb{E}\left[\sum_{k=0}^{N-1} h[k]x[n-k]x[n-i]\right] - \mathbb{E}[d[n]x[n-i]] \\ &= \sum_{k=0}^{N-1} h[k]r_{dx}[i-k] - r_{dx}[i] \quad (\text{WSS signals})\end{aligned}$$

Optimal Least Squares

setting all partial derivatives to zero:

$$\sum_{k=0}^{N-1} h[k]r_x[i - k] = r_{dx}[i]$$

in matrix form:

$$\mathbf{R}\mathbf{h} = \mathbf{g}$$

Optimal Least Squares

$$\mathbf{h} = [h[0] \quad h[1] \quad h[2] \quad \dots \quad h[N-1]]^T$$

$$\mathbf{R} = \begin{bmatrix} r_x[0] & r_x[1] & r_x[2] & \dots & r_x[N-1] \\ r_x[1] & r_x[0] & r_x[1] & \dots & r_x[N-2] \\ r_x[2] & r_x[1] & r_x[0] & \dots & r_x[N-3] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_x[N-1] & r_x[N-2] & \dots & \dots & r_x[0] \end{bmatrix}$$

$$\mathbf{g} = [r_{dx}[0] \quad r_{dx}[1] \quad r_{dx}[2] \quad \dots \quad r_{dx}[N-1]]^T$$

Error surface ($N = 2$)

$$\begin{aligned} J &= \mathbb{E} [e^2[n]] \\ &= \mathbb{E} \left[(d[n] - \hat{d}[n])^2 \right] \\ &= \mathbb{E} \left[(d[n] - (h_0x[n] + h_1x[n-1]))^2 \right] \\ &= \sigma_d^2 + r_x[0]h_0^2 + r_x[0]h_1^2 + 2r_x[1]h_0h_1 - 2r_{dx}[0]h_0 - 2r_{dx}[1]h_1 \\ &= \sigma_d^2 + \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_x[0] & r_x[1] \\ r_x[1] & r_x[0] \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} - 2 \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_{dx}[0] \\ r_{dx}[1] \end{bmatrix} \end{aligned}$$

Error surface ($N = 2$)

$$\begin{aligned} J &= \mathbb{E} [e^2[n]] \\ &= \mathbb{E} \left[(d[n] - \hat{d}[n])^2 \right] \\ &= \mathbb{E} \left[(d[n] - (h_0x[n] + h_1x[n-1]))^2 \right] \\ &= \sigma_d^2 + r_x[0]h_0^2 + r_x[0]h_1^2 + 2r_x[1]h_0h_1 - 2r_{dx}[0]h_0 - 2r_{dx}[1]h_1 \\ &= \sigma_d^2 + \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_x[0] & r_x[1] \\ r_x[1] & r_x[0] \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} - 2 \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_{dx}[0] \\ r_{dx}[1] \end{bmatrix} \end{aligned}$$

Error surface ($N = 2$)

$$\begin{aligned} J &= \mathbb{E} [e^2[n]] \\ &= \mathbb{E} \left[(d[n] - \hat{d}[n])^2 \right] \\ &= \mathbb{E} \left[(d[n] - (h_0x[n] + h_1x[n-1]))^2 \right] \\ &= \sigma_d^2 + r_x[0]h_0^2 + r_x[0]h_1^2 + 2r_x[1]h_0h_1 - 2r_{dx}[0]h_0 - 2r_{dx}[1]h_1 \\ &= \sigma_d^2 + \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_x[0] & r_x[1] \\ r_x[1] & r_x[0] \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} - 2 \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_{dx}[0] \\ r_{dx}[1] \end{bmatrix} \end{aligned}$$

Error surface ($N = 2$)

$$\begin{aligned} J &= E[e^2[n]] \\ &= E\left[\left(d[n] - \hat{d}[n]\right)^2\right] \\ &= E\left[\left(d[n] - (h_0x[n] + h_1x[n-1]))^2\right]\right] \\ &= \sigma_d^2 + r_x[0]h_0^2 + r_x[0]h_1^2 + 2r_x[1]h_0h_1 - 2r_{dx}[0]h_0 - 2r_{dx}[1]h_1 \\ &= \sigma_d^2 + \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_x[0] & r_x[1] \\ r_x[1] & r_x[0] \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} - 2 \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_{dx}[0] \\ r_{dx}[1] \end{bmatrix} \end{aligned}$$

Error surface ($N = 2$)

$$\begin{aligned} J &= E[e^2[n]] \\ &= E\left[\left(d[n] - \hat{d}[n]\right)^2\right] \\ &= E\left[\left(d[n] - (h_0x[n] + h_1x[n-1])\right)^2\right] \\ &= \sigma_d^2 + r_x[0]h_0^2 + r_x[0]h_1^2 + 2r_x[1]h_0h_1 - 2r_{dx}[0]h_0 - 2r_{dx}[1]h_1 \\ &= \sigma_d^2 + \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_x[0] & r_x[1] \\ r_x[1] & r_x[0] \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} - 2 \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_{dx}[0] \\ r_{dx}[1] \end{bmatrix} \end{aligned}$$

Error surface ($N = 2$)

$$J = \sigma_d^2 + \mathbf{h}^T \mathbf{R} \mathbf{h} - 2\mathbf{h}^T \mathbf{g}$$

Error surface ($N = 2$)

minimum achievable MSE

$$J = \boxed{\sigma_d^2} + \mathbf{h}^T \mathbf{R} \mathbf{h} - 2\mathbf{h}^T \mathbf{g}$$

Error surface ($N = 2$)

translation term (minimum is not in origin)

$$J = \sigma_d^2 + \mathbf{h}^T \mathbf{R} \mathbf{h} - 2\mathbf{h}^T \mathbf{g}$$

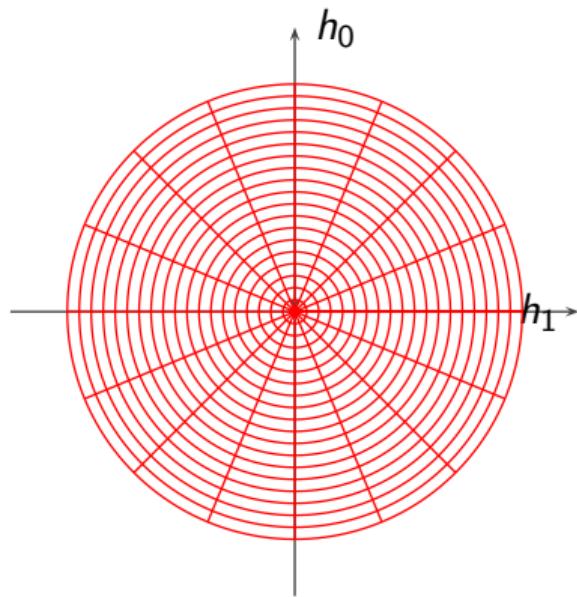
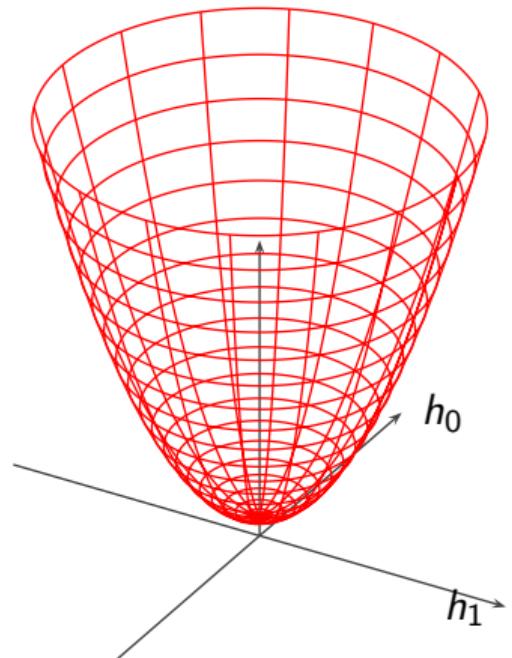
Error surface ($N = 2$)

$$J = \sigma_d^2 + \mathbf{h}^T \mathbf{R} \mathbf{h}$$

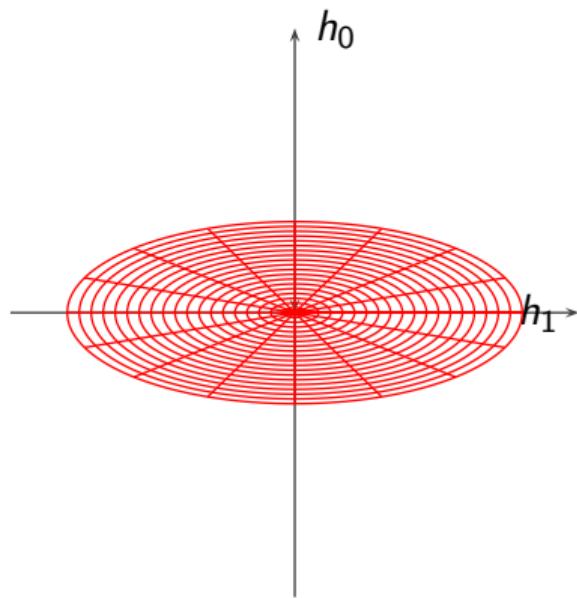
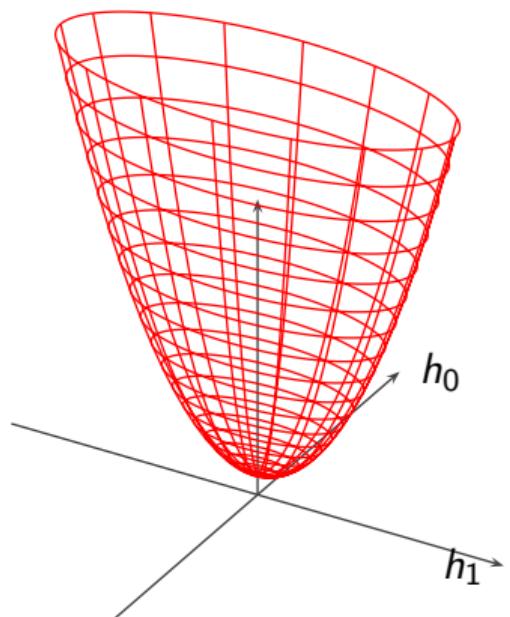
error surface is an elliptic paraboloid:

- ▶ major and minor axes are proportional to $1/\sqrt{\lambda_{0,1}}$
- ▶ signal's autocorrelation determines the shape of the error surface

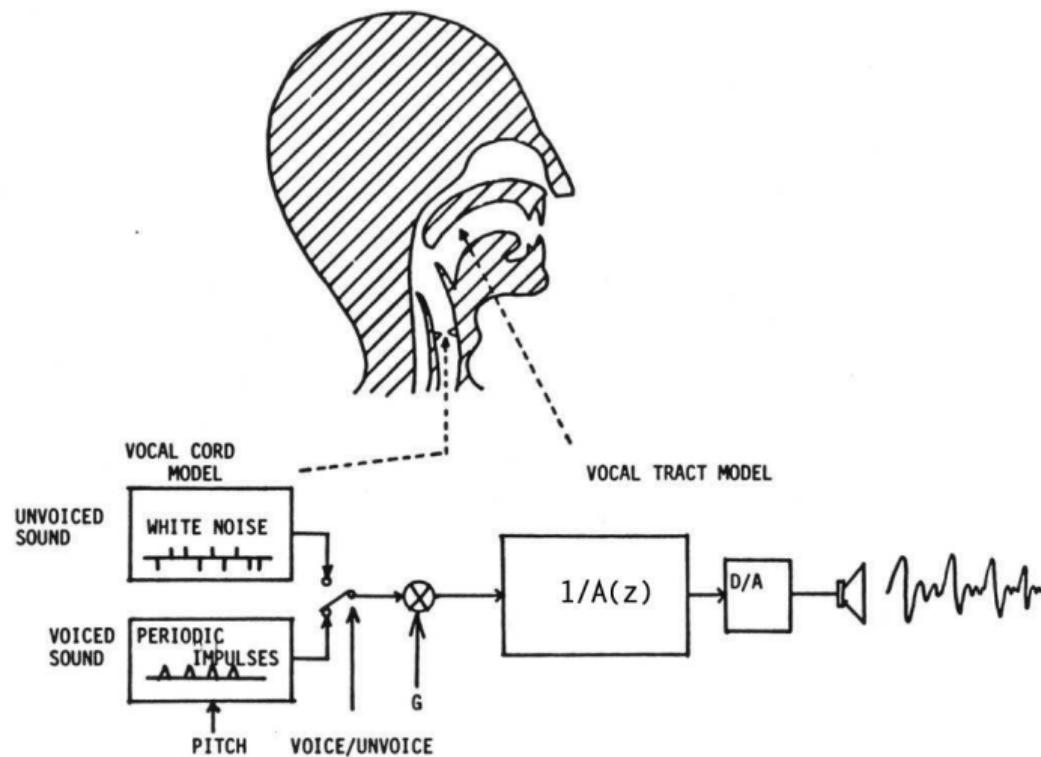
Error surface for white input



Error surface for correlated input



Example: linear prediction coding of speech

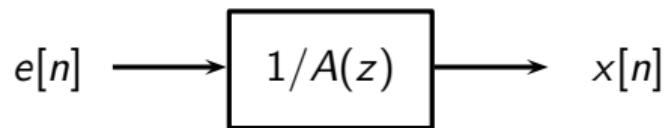


All-pole models

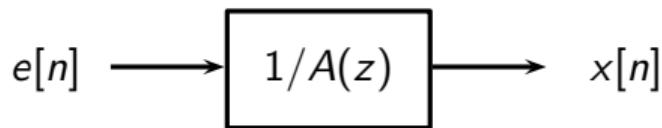
$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2} - \dots - a_N z^{-N}};$$

- ▶ poles model natural resonances of physical systems
- ▶ model is also called autoregressive (output is purely recursive)

Estimating an all-pole model



Estimating an all-pole model



- ▶ $e[n]$: unknown excitation
- ▶ $x[n]$: observable signal
- ▶ can we determine $A(z)$?

Linear Prediction

$$X(z) = E(z)/A(z)$$

$$E(z) = X(z)A(z)$$

$$e[n] = x[n] - \sum_{k=1}^N a_k x[n-k]$$

Linear Prediction

$$X(z) = E(z)/A(z)$$

$$E(z) = X(z)A(z)$$

$$e[n] = x[n] - \sum_{k=1}^N a_k x[n-k]$$

Linear Prediction

$$X(z) = E(z)/A(z)$$

$$E(z) = X(z)A(z)$$

$$e[n] = x[n] - \sum_{k=1}^N a_k x[n-k]$$

Remember the optimal Least Squares solution...

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

Linear Prediction

$$e[n] = x[n] - \sum_{k=1}^N a_k x[n-k]$$

- ▶ we shouldn't be able to predict excitation $e[n]$
- ▶ excitation and prediction should be orthogonal
- ▶ Least Squares solution is *the* solution

Linear Prediction

by setting $\partial E [e^2[n]] / \partial a_i$ to zero...

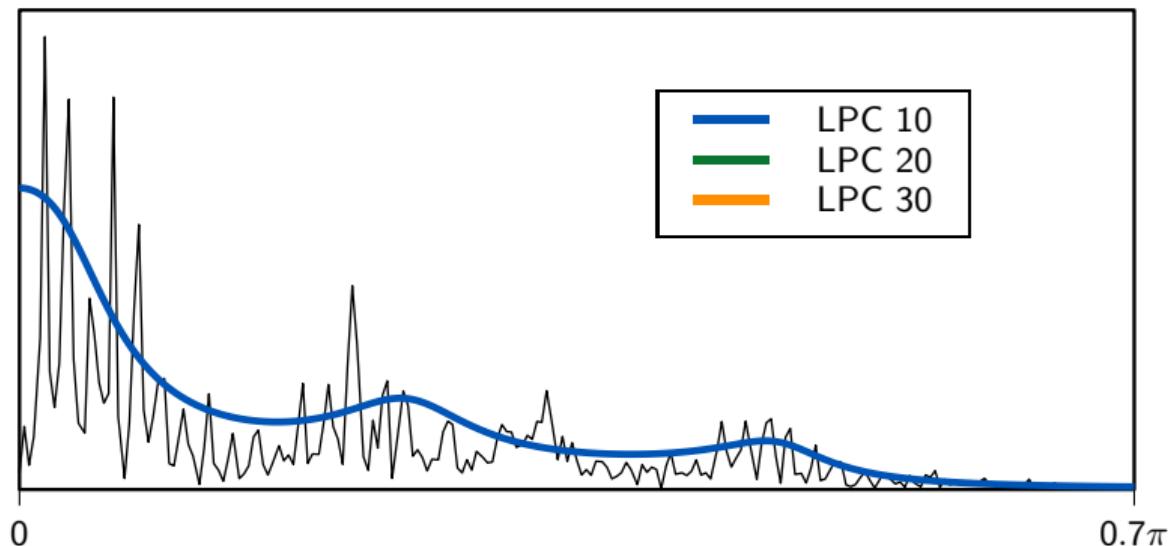
$$\mathbf{R}\hat{\mathbf{a}} = \mathbf{r}$$

$$\begin{bmatrix} r_x[0] & r_x[1] & \dots & r_x[N-1] \\ r_x[1] & r_x[0] & \dots & r_x[N-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_x[N-1] & r_x[N-2] & \dots & r_x[0] \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \vdots \\ \hat{a}_N \end{bmatrix} = \begin{bmatrix} r_x[1] \\ r_x[2] \\ \vdots \\ r_x[N] \end{bmatrix}.$$

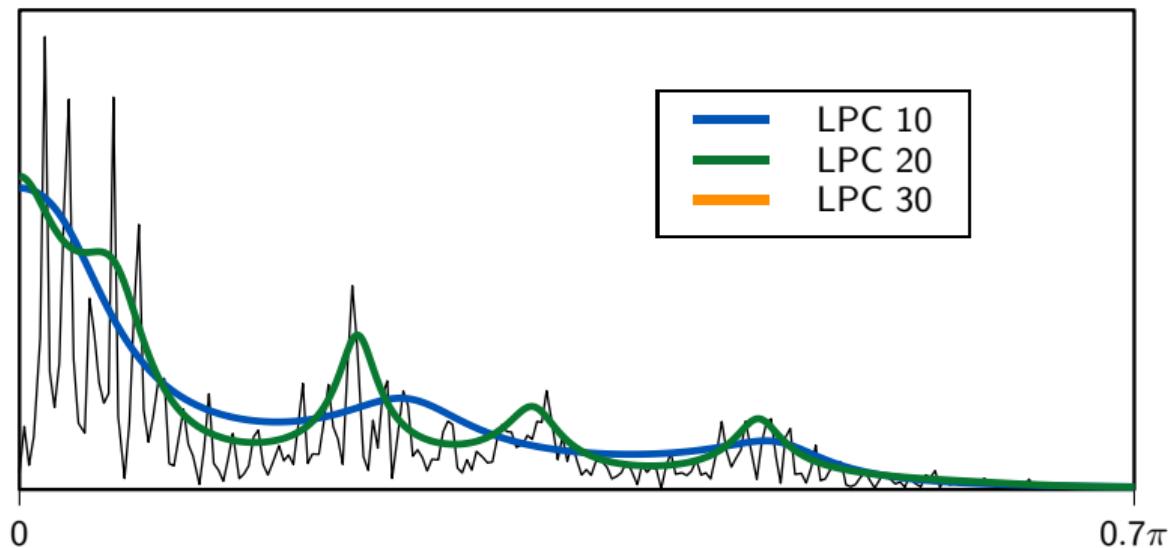
LPC speech coding

- ▶ segment speech in 20ms chunks (approx. stationary)
- ▶ find the coefficients for an all-pole model
- ▶ inverse filter and find the residual
- ▶ classify the residual excitation as voiced/unvoiced

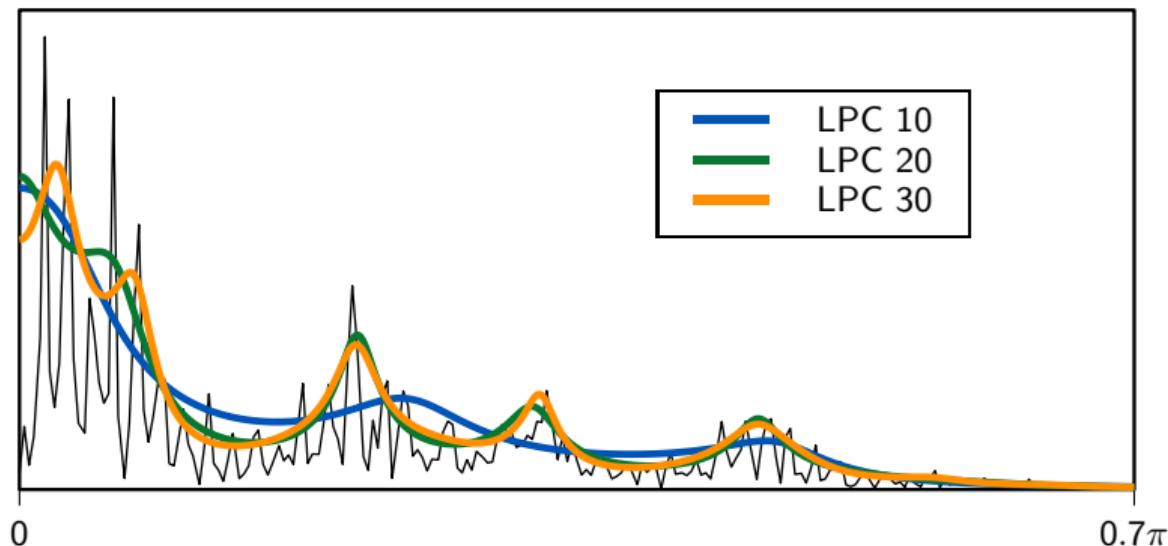
LPC order selection



LPC order selection



LPC order selection



LPC speech coding

- ▶ normally $N = 20$
- ▶ average bitrate 4Kbit/sec (raw data: 48Kbit/sec)
- ▶ many improvements exist: CELP & Co

original

LPC-coded

Finding the minimum squared error

Two cases:

- ▶ for WSS signals, one-shot solution: Optimal Least Squares
- ▶ for “almost” WSS signals, iterative solutions: stochastic gradient descent or LMS

Iterative minimization

Steepest descent:

start with a guess \mathbf{x}_0 and then, iteratively,

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha_n \nabla f(\mathbf{x}_n)$$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_0} & \frac{\partial f(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f(\mathbf{x})}{\partial x_{N-1}} \end{bmatrix}^T$$

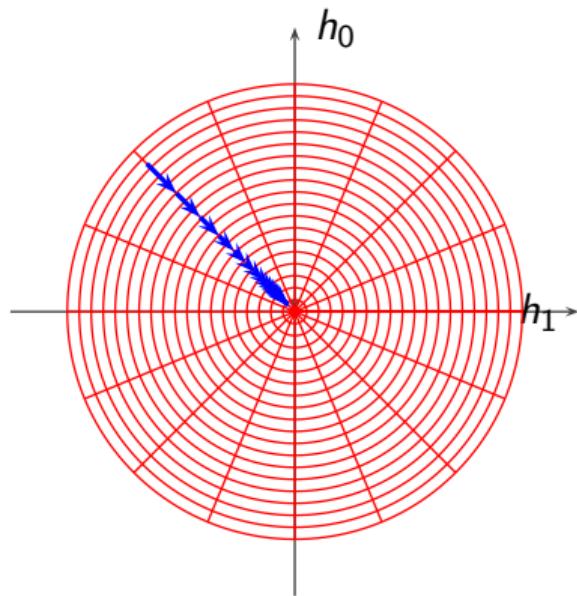
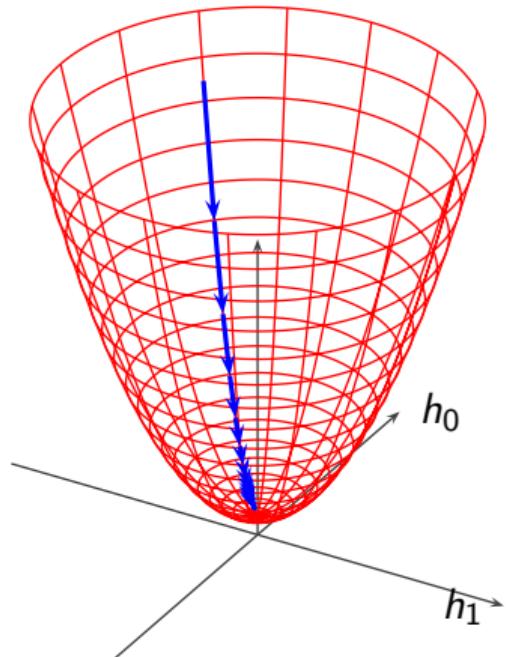
α_n : learning factor

Iterative minimization

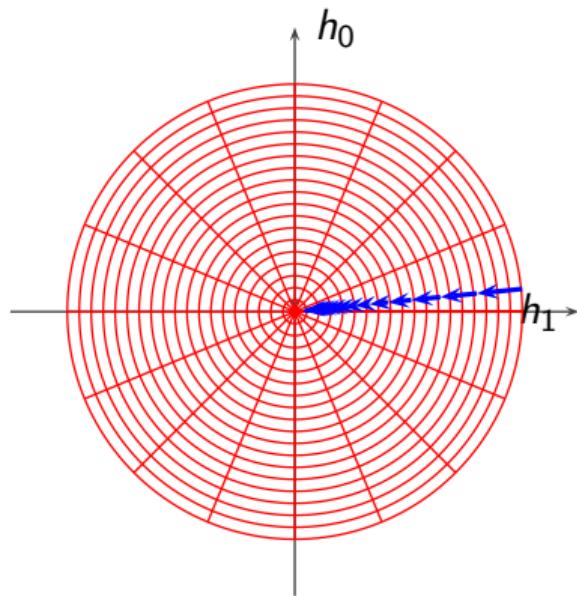
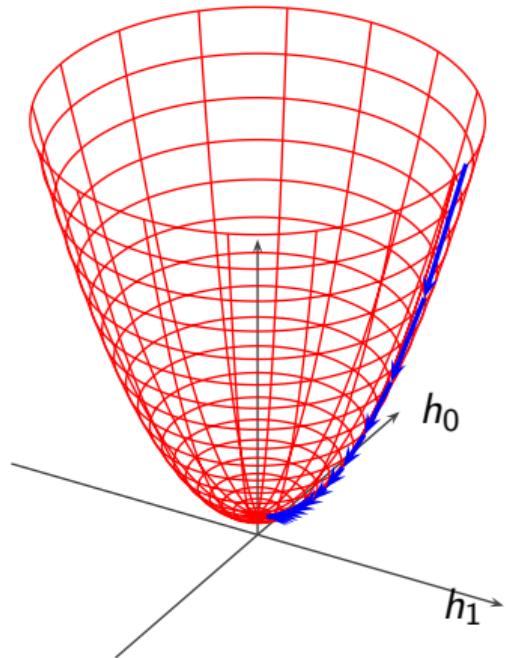
- ▶ for a quadratic error surface, minimum is always global
- ▶ gradient is easy to compute:

$$\begin{aligned}\nabla J(\mathbf{h}) &= \left[\frac{\partial E[e^2[n]]}{\partial h[0]} \quad \frac{\partial E[e^2[n]]}{\partial h[1]} \quad \cdots \quad \frac{\partial E[e^2[n]]}{\partial h[N-1]} \right]^T \\ &= 2(\mathbf{R}\mathbf{h} - \mathbf{g})\end{aligned}$$

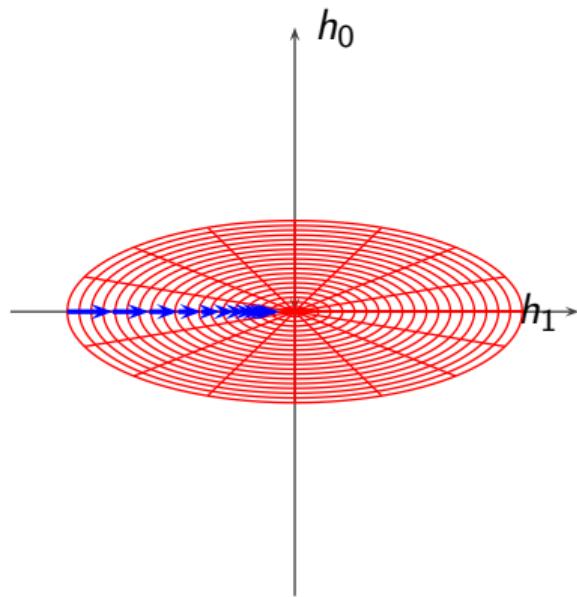
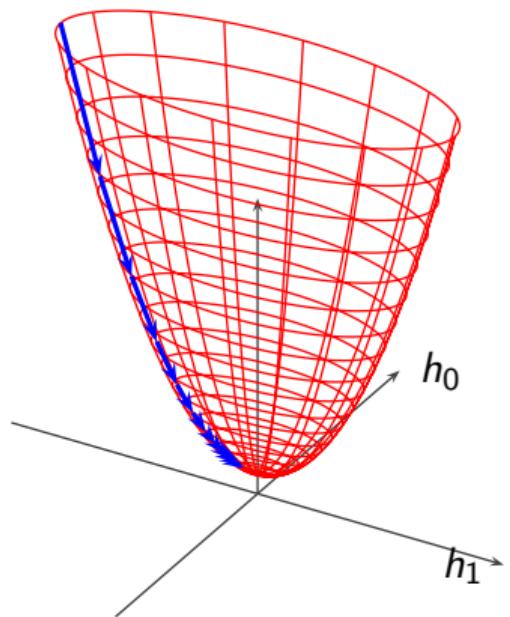
Steepest descent for white input



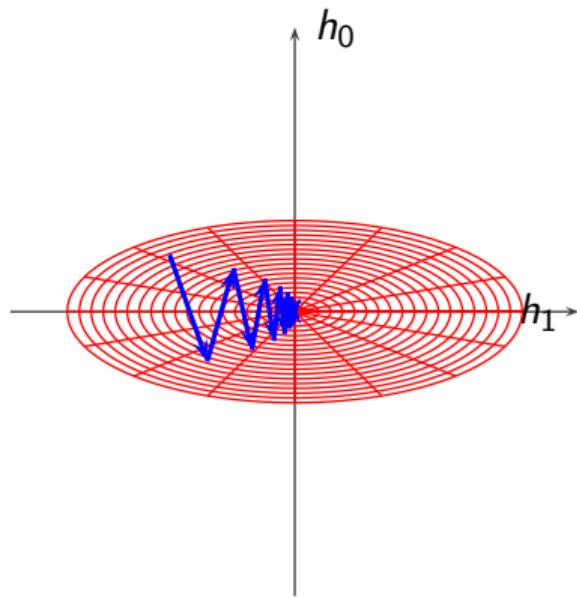
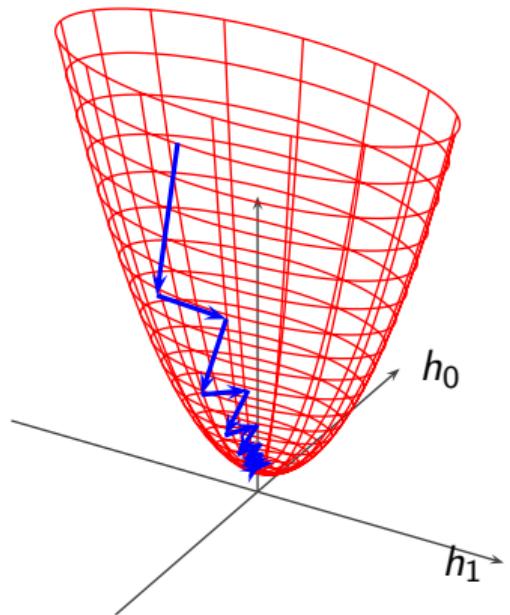
Steepest descent for white input



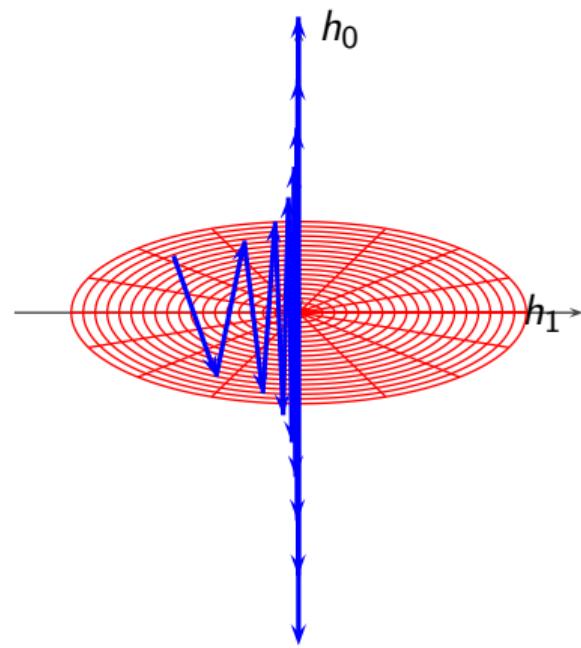
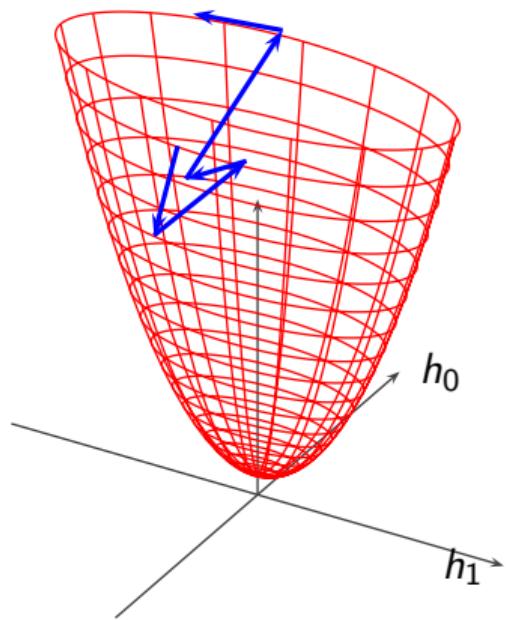
Error surface for correlated input: good guess



Error surface for correlated input: less good guess



Error surface for correlated input: learning factor too large!



Iterative minimization

- ▶ for WSS signals, one-shot and iterative are the same
- ▶ for time-varying signals, we need to follow the changes: iterative solution
- ▶ computation of time-varying correlations is costly
- ▶ *stochastic* gradient descent:

$$\mathbb{E} [e^2[n]] \leftarrow e^2[n]$$

Iterative minimization

- ▶ for WSS signals, one-shot and iterative are the same
- ▶ for time-varying signals, we need to follow the changes: iterative solution
- ▶ computation of time-varying correlations is costly
- ▶ *stochastic* gradient descent:

$$\mathbb{E} [e^2[n]] \leftarrow e^2[n]$$

Iterative minimization

- ▶ for WSS signals, one-shot and iterative are the same
- ▶ for time-varying signals, we need to follow the changes: iterative solution
- ▶ computation of time-varying correlations is costly
- ▶ *stochastic gradient descent:*

$$\mathbb{E}[e^2[n]] \leftarrow e^2[n]$$

Iterative minimization

- ▶ for WSS signals, one-shot and iterative are the same
- ▶ for time-varying signals, we need to follow the changes: iterative solution
- ▶ computation of time-varying correlations is costly
- ▶ *stochastic* gradient descent:

$$E[e^2[n]] \leftarrow e^2[n]$$

Stochastic gradient descent

$$\nabla J = \left[\frac{\partial E [e^2[n]]}{\partial h[0]} \quad \frac{\partial E [e^2[n]]}{\partial h[1]} \quad \cdots \quad \frac{\partial E [e^2[n]]}{\partial h[N-1]} \right]^T$$

Stochastic gradient descent

$$\nabla J_n = \begin{bmatrix} \frac{\partial e^2[n]}{\partial h[0]} & \frac{\partial e^2[n]}{\partial h[1]} & \cdots & \frac{\partial e^2[n]}{\partial h[N-1]} \end{bmatrix}^T$$

Stochastic gradient descent

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\frac{\partial e^2[n]}{\partial h[i]} = -2e[n]x[n-i].$$

$$\nabla J_n = -2e[n] \mathbf{x}_n$$

$$\mathbf{x}_n = [x[n] \ x[n-1] \ x[n-2] \ \dots \ x[n-N+1]]^T.$$

Stochastic gradient descent

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\frac{\partial e^2[n]}{\partial h[i]} = -2e[n]x[n-i].$$

$$\nabla J_n = -2e[n]x_n$$

$$x_n = [x[n] \ x[n-1] \ x[n-2] \ \dots \ x[n-N+1]]^T.$$

Stochastic gradient descent

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\frac{\partial e^2[n]}{\partial h[i]} = -2e[n]x[n-i].$$

$$\nabla J_n = -2e[n] \mathbf{x}_n$$

$$\mathbf{x}_n = [x[n] \ x[n-1] \ x[n-2] \ \dots \ x[n-N+1]]^T.$$

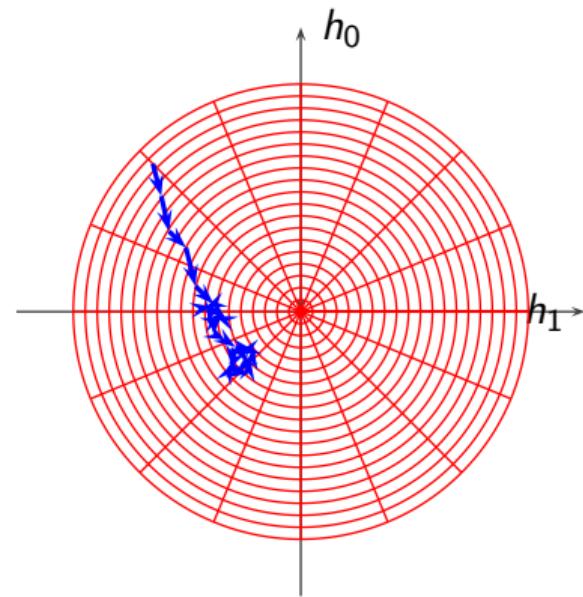
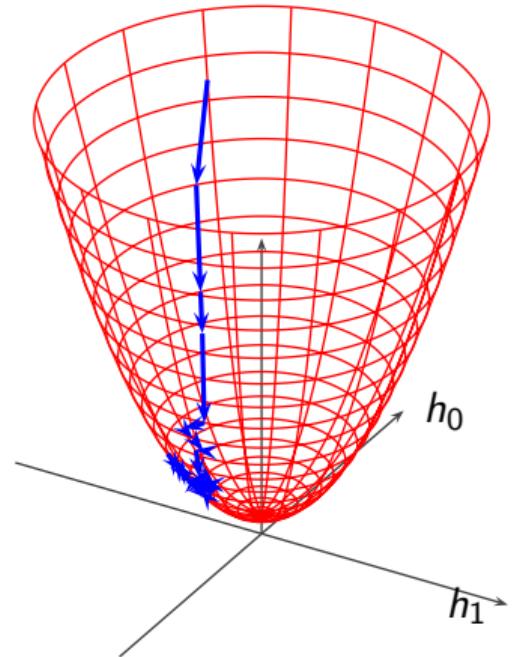
The LMS filter

$$\mathbf{h}_0 = [h_0[0] \quad h_0[1] \quad \dots \quad h_0[N-1]]^T \text{ initial guess}$$

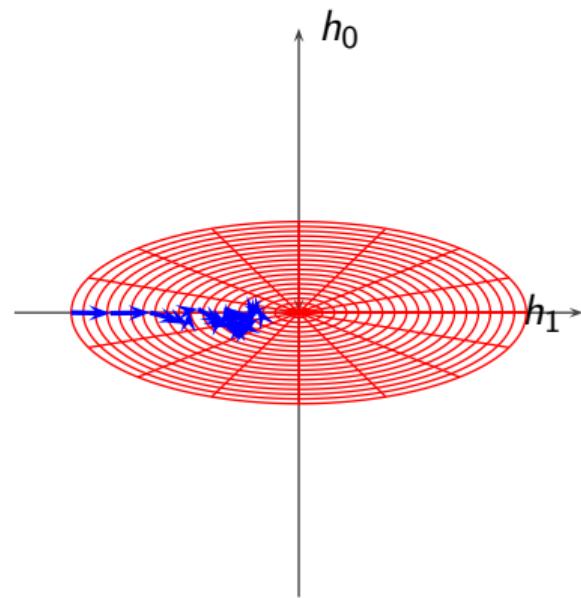
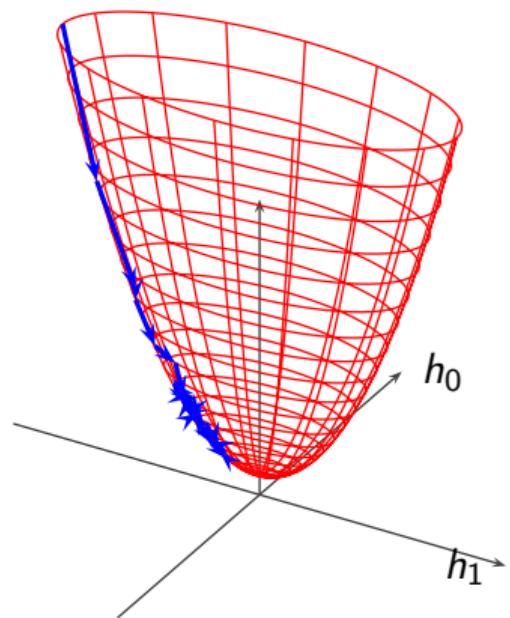
$$e[n] = d[n] - \mathbf{h}_n^T \mathbf{x}_n$$

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \alpha_n e[n] \mathbf{x}_n$$

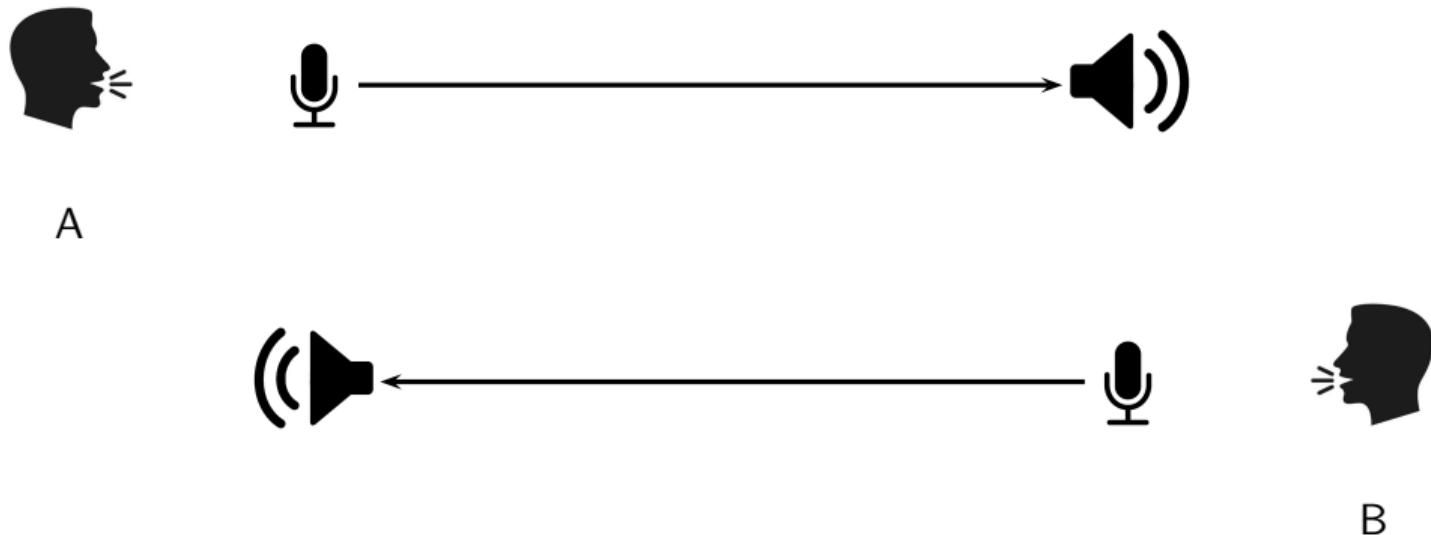
LMS for white input



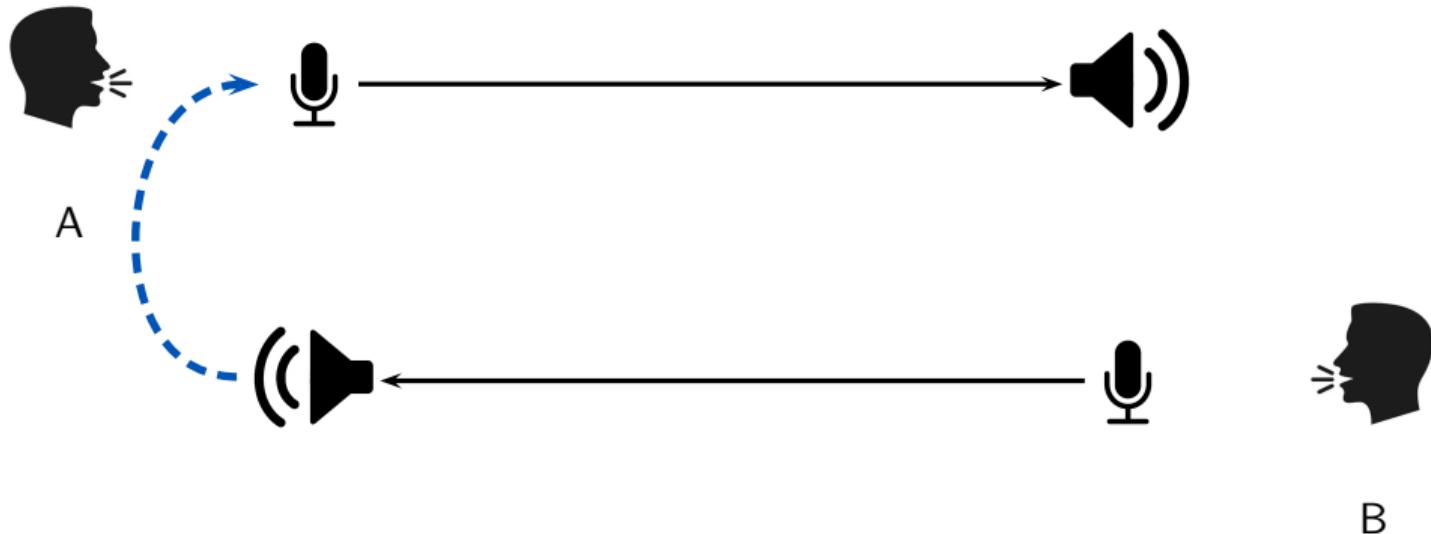
LMS for correlated input



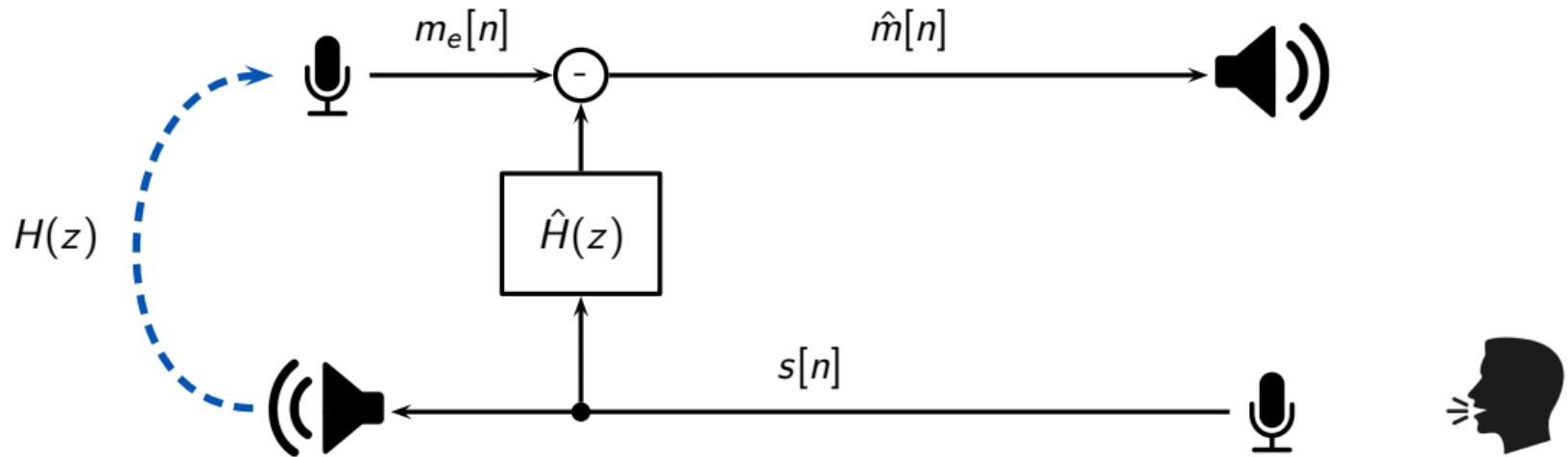
Example: adaptive echo cancellation



Example: adaptive echo cancellation



Example: adaptive echo cancellation



The echo-corrupted signal

Signal captured by the microphone:

$$m_e[n] = m[n] + h[n] * s[n]$$

The echo-corrupted signal

Signal captured by the microphone:

$$m_e[n] = m[n] + h[n] * s[n]$$

speaker A's voice



The echo-corrupted signal

Signal captured by the microphone:

$$m_e[n] = m[n] + h[n] * s[n]$$

echo transfer function

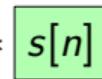


The echo-corrupted signal

Signal captured by the microphone:

$$m_e[n] = m[n] + h[n] * s[n]$$

speaker B's voice



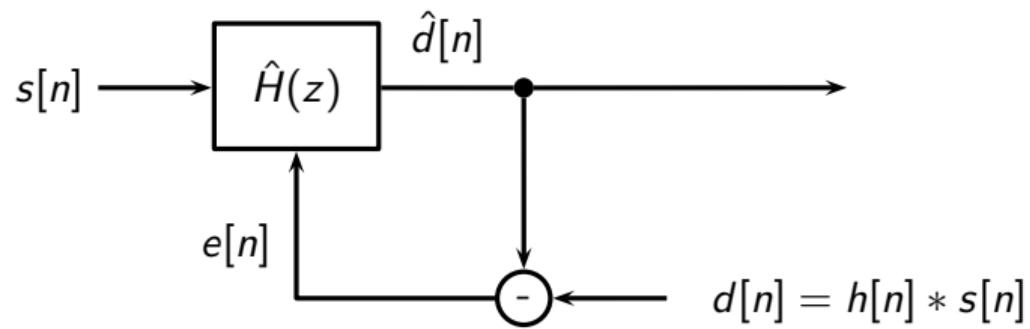
The echo-corrupted signal

Signal captured by the microphone:

$$m_e[n] = m[n] + h[n] * s[n]$$

we need to estimate $h[n]$ in order to *subtract* the unwanted echo

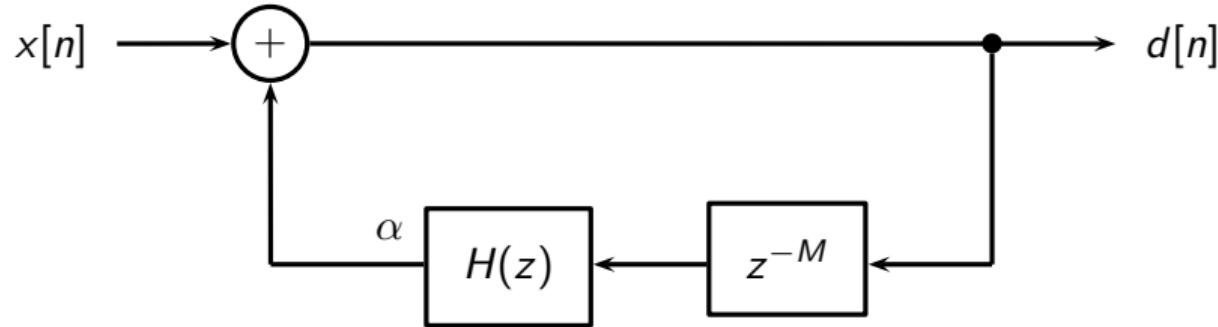
Echo cancellation as adaptive filtering



Training the filter

- ▶ “desired” signal is the echo (so we can subtract it)
- ▶ normally, only one person talks at a time: when B is speaking, $m_e[n] = h[n] * s[n]$
- ▶ people move, volume changes: $H(z)$ is time varying!
- ▶ use the LMS filter

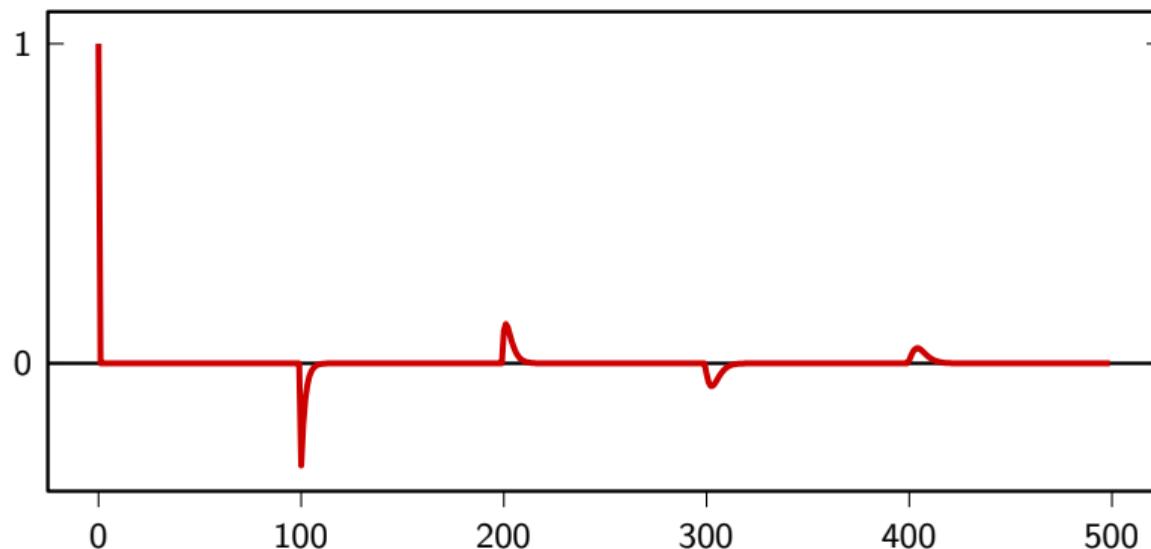
Example: simple echo model



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

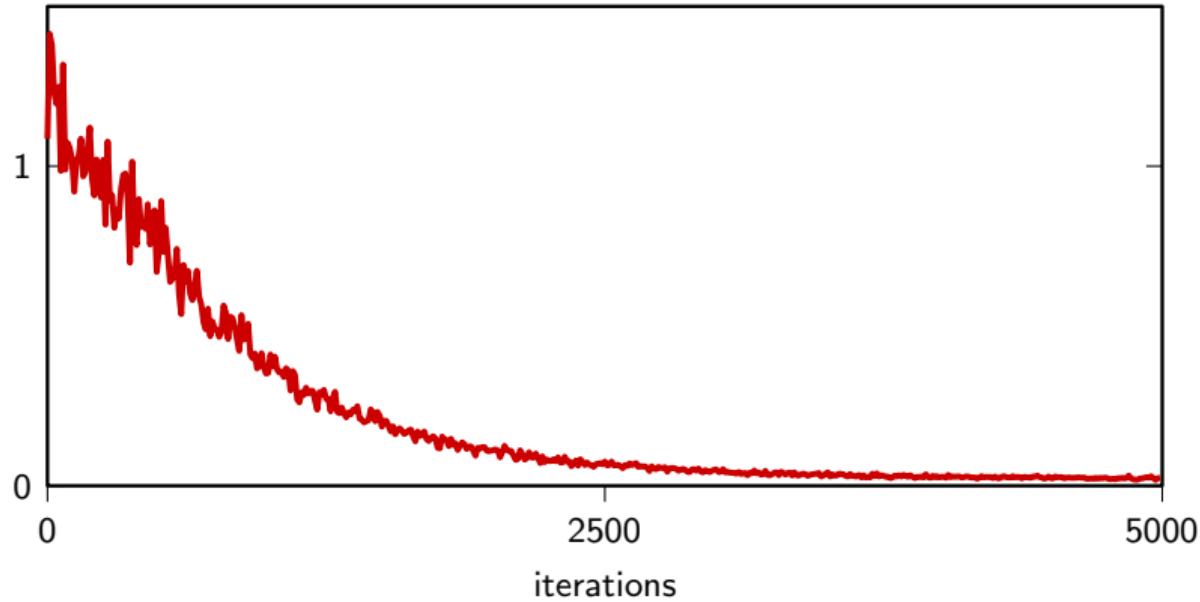
Echo impulse response

$$M = 100, \alpha = -0.8, \lambda = 0.6$$



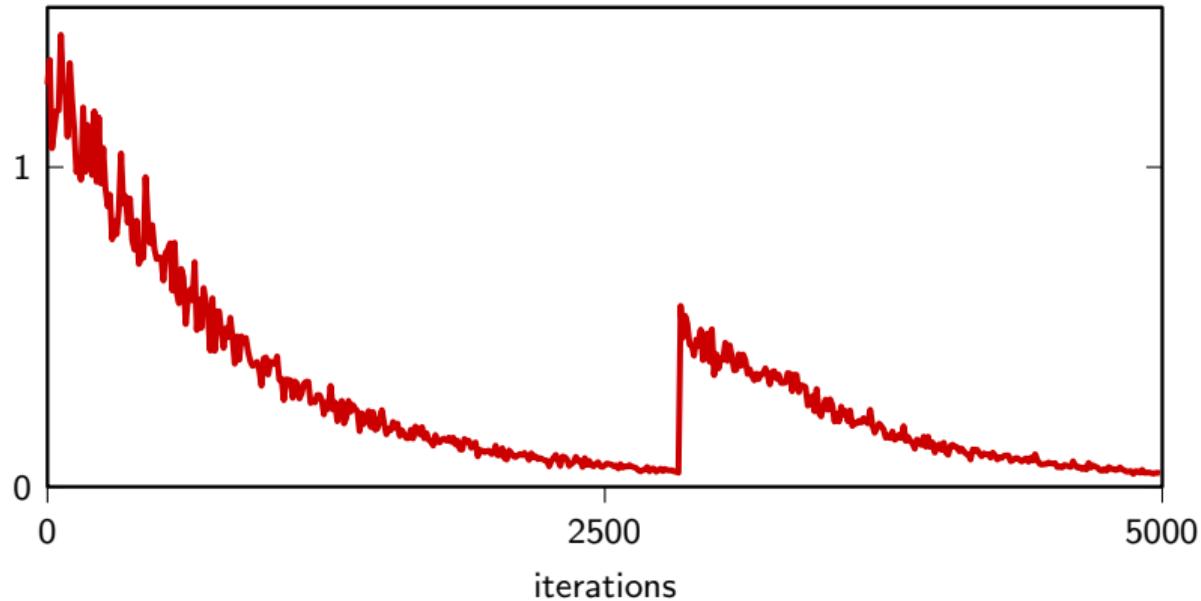
Running the LMS adaptation

white input, averaged MSE over 200 experiments



LMS can catch up with changes

echo delay changes from $M = 100$ to $M = 90$ at $n = 3000$



COM303: Digital Signal Processing

Lecture 16: Interpolation

overview

- ▶ the analog worldview
- ▶ interpolation of discrete-time signals
- ▶ bandlimited functions
- ▶ the sinc basis and sinc sampling

Two views of the world



Analog/continuous versus discrete/digital

Two views of the world

analog worldview:

- ▶ calculus
- ▶ distributions
- ▶ system theory
- ▶ electronics

Two views of the world

analog worldview:

- ▶ calculus
- ▶ distributions
- ▶ system theory
- ▶ electronics

digital worldview:

- ▶ arithmetic
- ▶ combinatorics
- ▶ computer science
- ▶ DSP

Two views of the world

digital worldview:

- ▶ countable integer index n
- ▶ sequences $x[n] \in \ell_2(\mathbb{Z})$
- ▶ frequency $\omega \in [-\pi, \pi]$
- ▶ DTFT: $\ell_2(\mathbb{Z}) \mapsto L_2([-\pi, \pi])$

Two views of the world

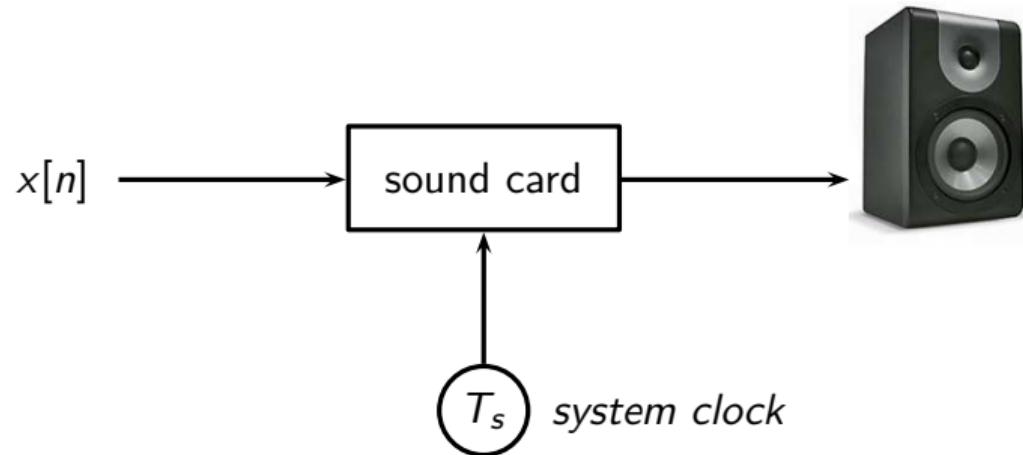
digital worldview:

- ▶ countable integer index n
- ▶ sequences $x[n] \in \ell_2(\mathbb{Z})$
- ▶ frequency $\omega \in [-\pi, \pi]$
- ▶ DTFT: $\ell_2(\mathbb{Z}) \mapsto L_2([-\pi, \pi])$

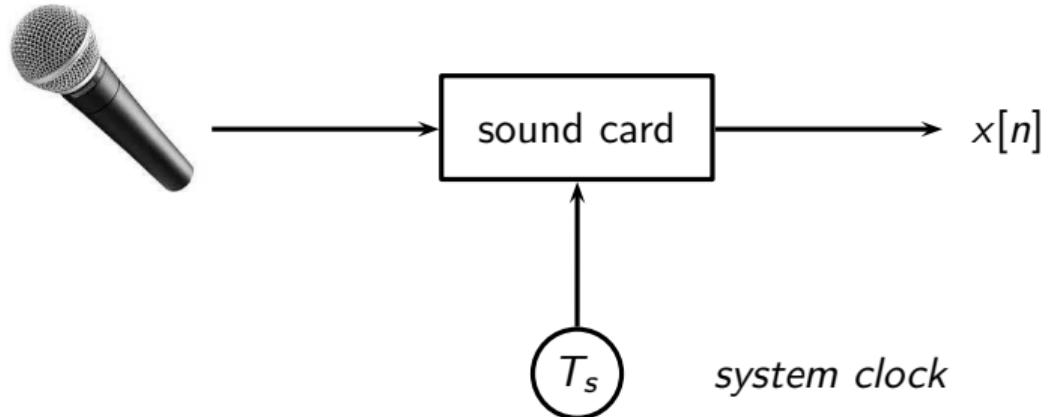
analog worldview:

- ▶ real-valued time t (sec)
- ▶ functions $x(t) \in L_2(\mathbb{R})$
- ▶ frequency $f \in \mathbb{R}$ (Hz)
- ▶ FT: $L_2(\mathbb{R}) \mapsto L_2(\mathbb{R})$

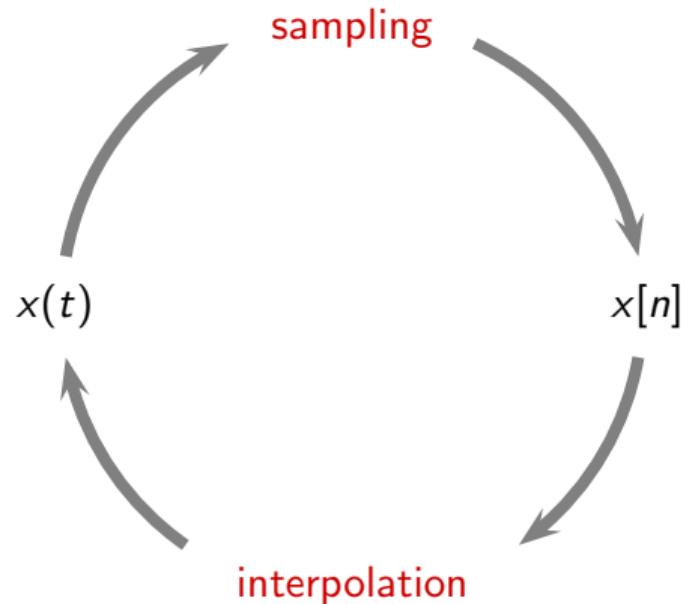
Bridging the gap: interpolation



Bridging the gap: sampling



Bridging the gap

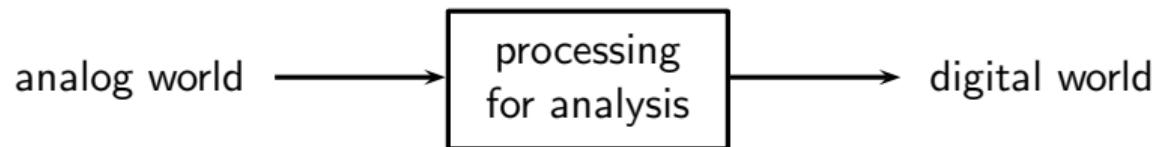


Today, processing is as digital as possible

- ▶ analog to digital
- ▶ digital to analog
- ▶ analog to digital to analog

Digital processing of signals from the analog world

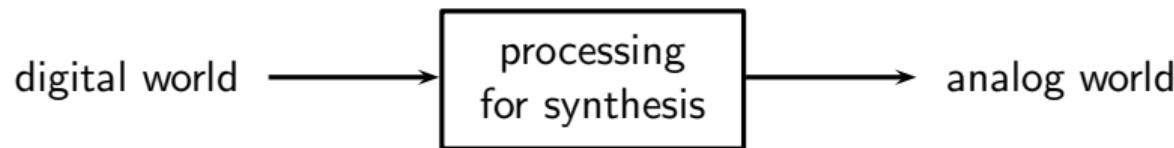
- ▶ input is continuous-time: $x(t)$
- ▶ output is discrete-time: $y[n]$
- ▶ processing is on sequences: $x[n], y[n]$



examples: storage and compression (MP3, JPG), control systems, monitoring

Digital processing of signals to the analog world

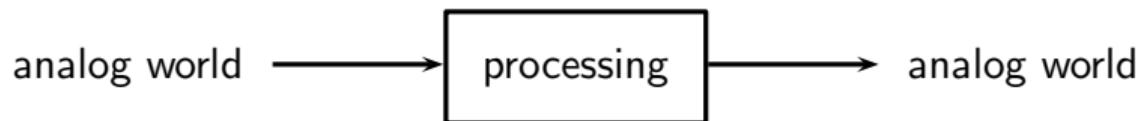
- ▶ input is discrete-time: $x[n]$
- ▶ output is continuous-time: $y(t)$
- ▶ processing is on sequences: $x[n], y[n]$



examples: music synthesizers, computer graphics, video games

Digital processing of signals from/to the analog world

- ▶ input is continuous-time: $x(t)$
- ▶ output is continuous-time: $y(t)$
- ▶ processing is on sequences: $x[n], y[n]$



examples: telephony, VOIP, sound effects, digital photography

continuous-time signal processing

About continuous time

- ▶ time: real variable t
- ▶ signal $x(t)$: complex functions of a real variable
- ▶ finite energy: $x(t) \in L_2(\mathbb{R})$ (square integrable functions)
- ▶ inner product in $L_2(\mathbb{R})$

$$\langle x(t), y(t) \rangle = \int_{-\infty}^{\infty} x^*(t)y(t)dt$$

- ▶ energy: $\|x(t)\|^2 = \langle x(t), x(t) \rangle$

About continuous time

- ▶ time: real variable t
- ▶ signal $x(t)$: complex functions of a real variable
- ▶ finite energy: $x(t) \in L_2(\mathbb{R})$ (square integrable functions)
- ▶ inner product in $L_2(\mathbb{R})$

$$\langle x(t), y(t) \rangle = \int_{-\infty}^{\infty} x^*(t)y(t)dt$$

- ▶ energy: $\|x(t)\|^2 = \langle x(t), x(t) \rangle$

About continuous time

- ▶ time: real variable t
- ▶ signal $x(t)$: complex functions of a real variable
- ▶ finite energy: $x(t) \in L_2(\mathbb{R})$ (square integrable functions)
- ▶ inner product in $L_2(\mathbb{R})$

$$\langle x(t), y(t) \rangle = \int_{-\infty}^{\infty} x^*(t)y(t)dt$$

- ▶ energy: $\|x(t)\|^2 = \langle x(t), x(t) \rangle$

About continuous time

- ▶ time: real variable t
- ▶ signal $x(t)$: complex functions of a real variable
- ▶ finite energy: $x(t) \in L_2(\mathbb{R})$ (square integrable functions)
- ▶ inner product in $L_2(\mathbb{R})$

$$\langle x(t), y(t) \rangle = \int_{-\infty}^{\infty} x^*(t)y(t)dt$$

- ▶ energy: $\|x(t)\|^2 = \langle x(t), x(t) \rangle$

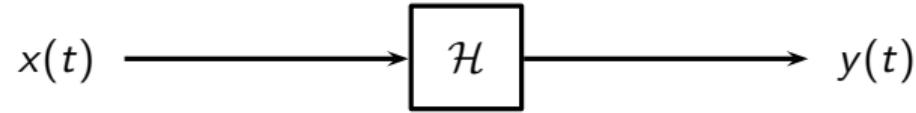
About continuous time

- ▶ time: real variable t
- ▶ signal $x(t)$: complex functions of a real variable
- ▶ finite energy: $x(t) \in L_2(\mathbb{R})$ (square integrable functions)
- ▶ inner product in $L_2(\mathbb{R})$

$$\langle x(t), y(t) \rangle = \int_{-\infty}^{\infty} x^*(t)y(t)dt$$

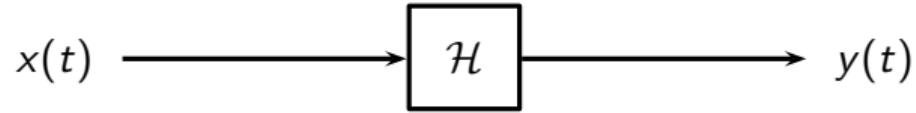
- ▶ energy: $\|x(t)\|^2 = \langle x(t), x(t) \rangle$

Analog LTI filters



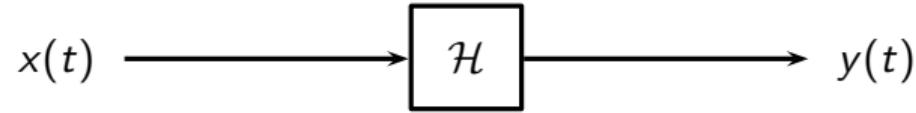
$$\begin{aligned}y(t) &= (x * h)(t) \\&= \int_{-\infty}^{\infty} x(\tau)h(t - \tau)d\tau \\&= \langle h^*(t - \tau), x(\tau) \rangle\end{aligned}$$

Analog LTI filters



$$\begin{aligned}y(t) &= (x * h)(t) \\&= \int_{-\infty}^{\infty} x(\tau)h(t - \tau)d\tau \\&= \langle h^*(t - \tau), x(\tau) \rangle\end{aligned}$$

Analog LTI filters



$$\begin{aligned}y(t) &= (x * h)(t) \\&= \int_{-\infty}^{\infty} x(\tau)h(t - \tau)d\tau \\&= \langle h^*(t - \tau), x(\tau) \rangle\end{aligned}$$

Real-world frequency

frequency: number of repetitions per *second*

- ▶ f expressed in Hz (1/sec)
- ▶ alternatively, angular frequency in rad/s: $\Omega = 2\pi f$
- ▶ period for periodic signals is $T = \frac{1}{f} = \frac{2\pi}{\Omega}$

Fourier analysis

- ▶ in discrete time max angular frequency is $\pm\pi$
- ▶ in continuous time no max frequency: $f \in \mathbb{R}$
- ▶ concept is the same: similarity to sinusoidal components

$$\begin{aligned} X(f) &= \langle e^{j2\pi ft}, x(t) \rangle \\ &= \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt \quad \leftarrow \text{not periodic!} \end{aligned}$$

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{j2\pi ft} df$$

Fourier analysis

- ▶ in discrete time max angular frequency is $\pm\pi$
- ▶ in continuous time no max frequency: $f \in \mathbb{R}$
- ▶ concept is the same: similarity to sinusoidal components

$$\begin{aligned} X(f) &= \langle e^{j2\pi ft}, x(t) \rangle \\ &= \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt \quad \leftarrow \text{not periodic!} \end{aligned}$$

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{j2\pi ft} df$$

Fourier analysis

- ▶ in discrete time max angular frequency is $\pm\pi$
- ▶ in continuous time no max frequency: $f \in \mathbb{R}$
- ▶ concept is the same: similarity to sinusoidal components

$$\begin{aligned} X(f) &= \langle e^{j2\pi ft}, x(t) \rangle \\ &= \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt \quad \leftarrow \text{not periodic!} \end{aligned}$$

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{j2\pi ft} df$$

Fourier analysis (in rad/s)

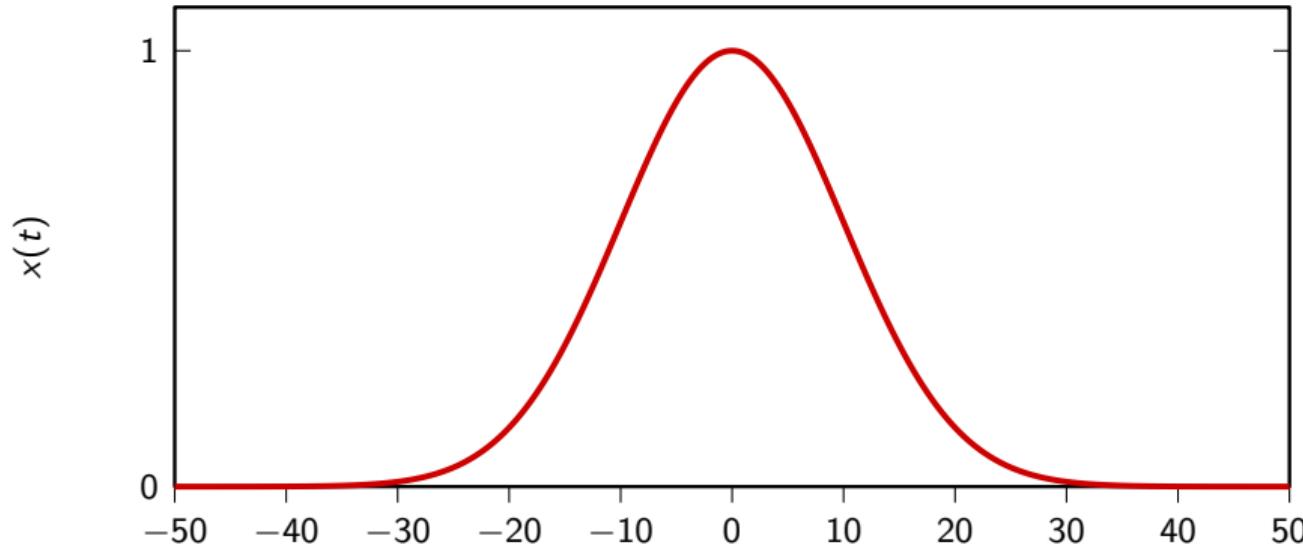
$$\begin{aligned} X(j\Omega) &= \langle e^{j\Omega t}, x(t) \rangle \\ &= \int_{-\infty}^{\infty} x(t) e^{-j\Omega t} dt \quad \leftarrow \text{not periodic!} \end{aligned}$$

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\Omega) e^{j\Omega t} d\Omega$$

- ▶ Laplace transform computed on the imaginary axis

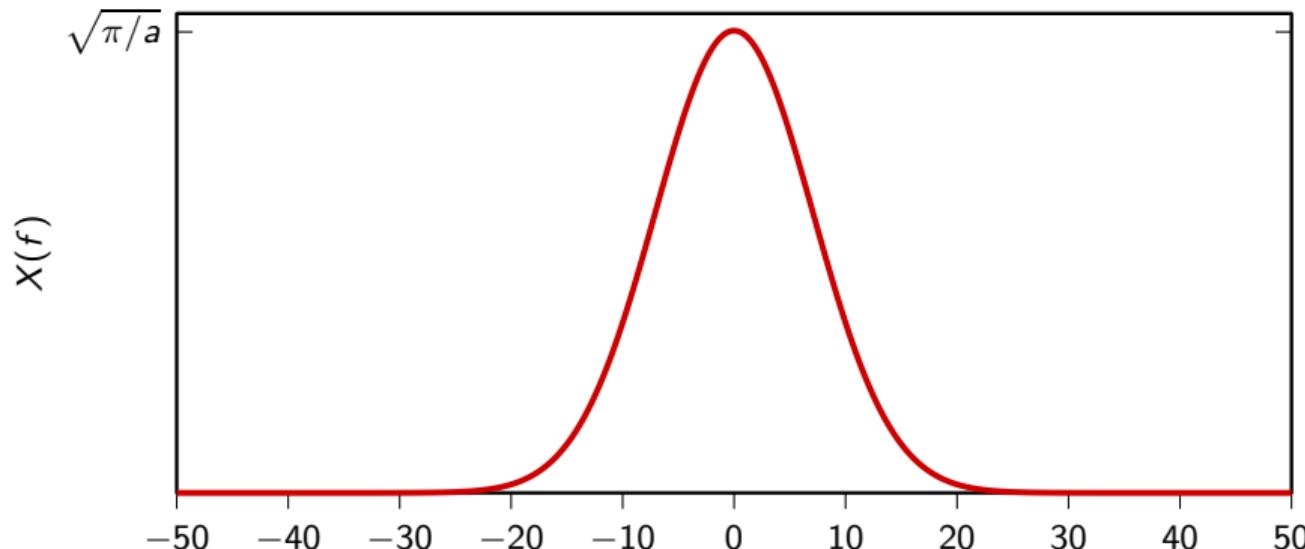
Example

$$x(t) = e^{-at^2}$$



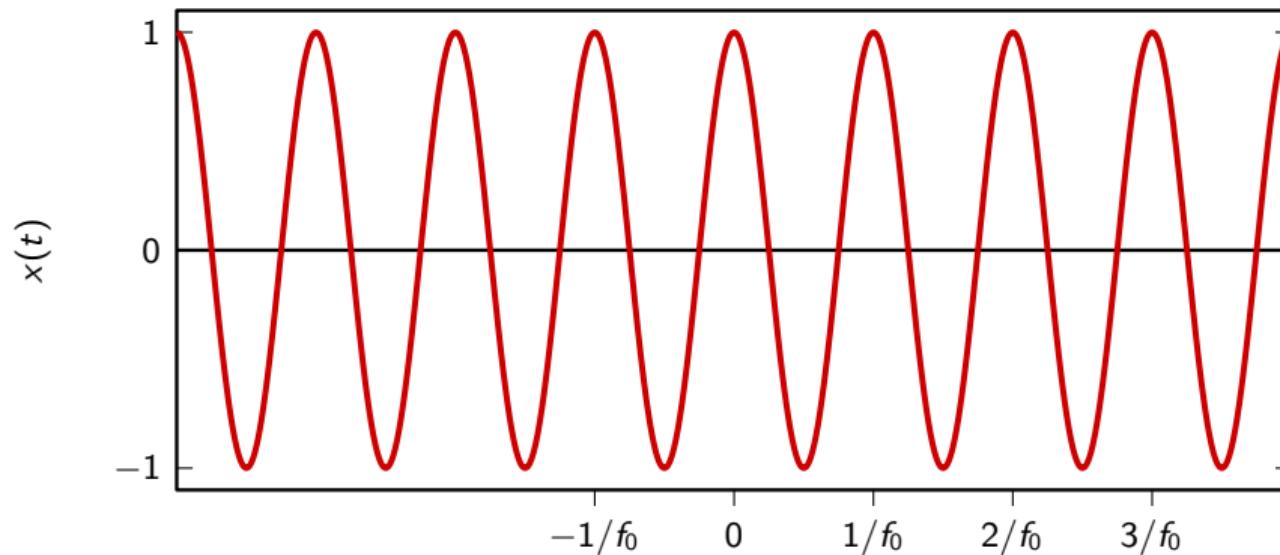
Example

$$X(f) = \sqrt{\pi/a} e^{-\frac{\pi^2}{a}f^2}$$



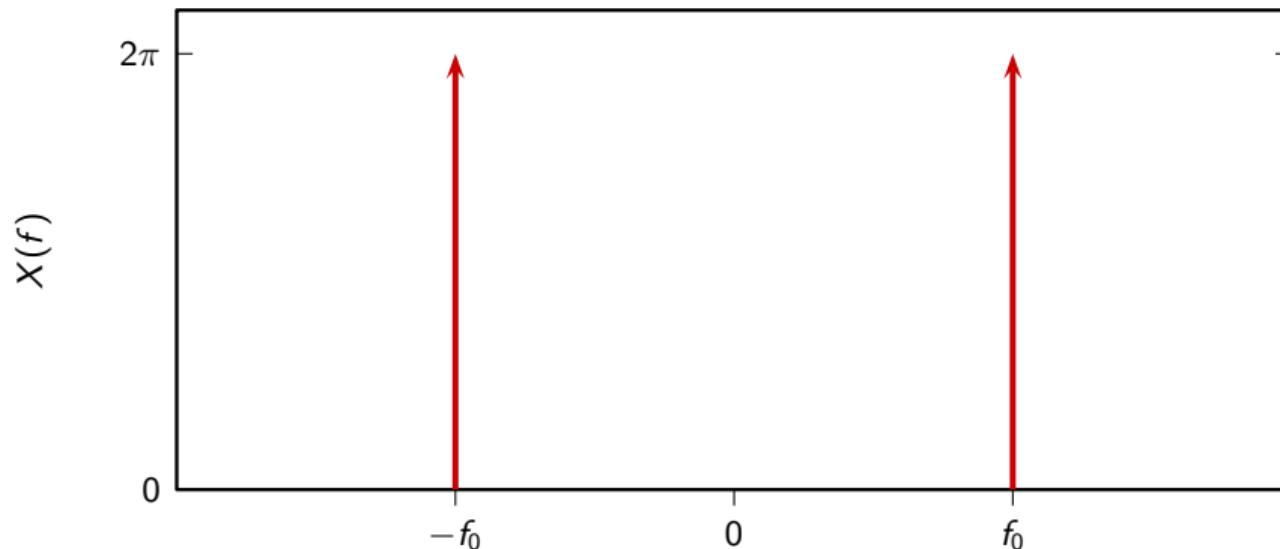
Example

$$x(t) = \cos(2\pi f_0 t)$$

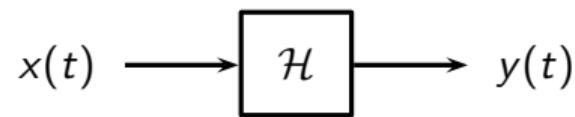


Example

$$X(f) = (1/2)\delta(f \pm f_0)$$



Convolution theorem



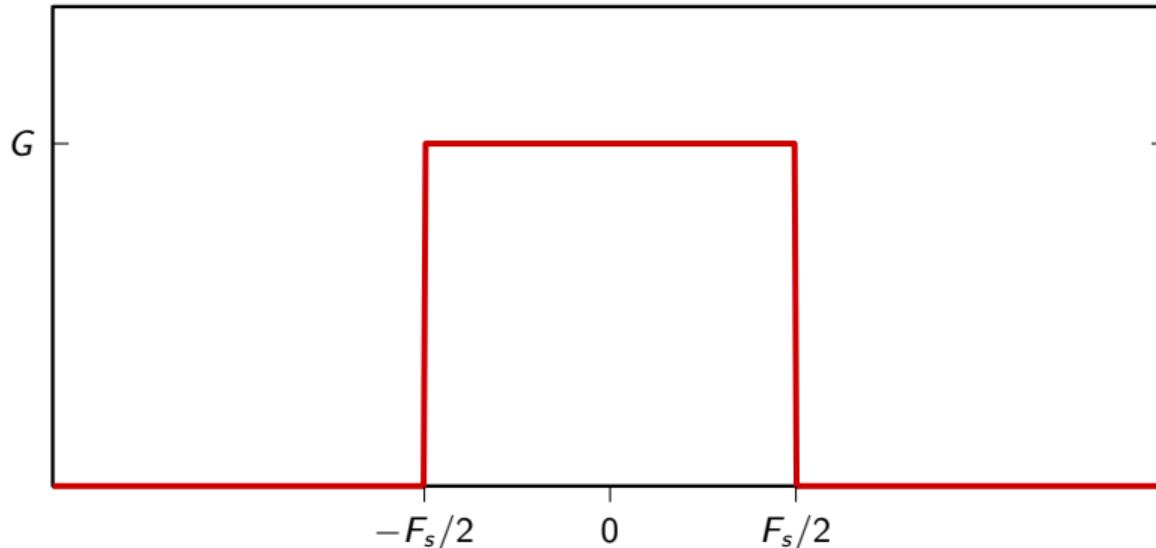
$$Y(f) = X(f) H(f)$$

A new concept: bandlimited functions

a continuous-time signal is bandlimited if there exists a frequency F_s such that:

$$X(f) = 0 \quad \text{for } |f| > F_s/2$$

Prototypical bandlimited function



The prototypical bandlimited function

$$\Phi(f) = G \operatorname{rect}\left(\frac{f}{F_s}\right)$$

$$\varphi(t) = \int_{-\infty}^{\infty} \Phi(f) e^{j2\pi ft} df$$

= ...

$$= GF_s \operatorname{sinc}(tF_s)$$

The prototypical bandlimited function

$$\Phi(f) = G \operatorname{rect}\left(\frac{f}{F_s}\right)$$

$$\varphi(t) = \int_{-\infty}^{\infty} \Phi(f) e^{j2\pi ft} df$$

= ...

$$= GF_s \operatorname{sinc}(tF_s)$$

The prototypical bandlimited function

- ▶ total bandwidth: F_s
- ▶ define $T_s = 1/F_s$
- ▶ normalization: $G = 1/F_s = T_s$

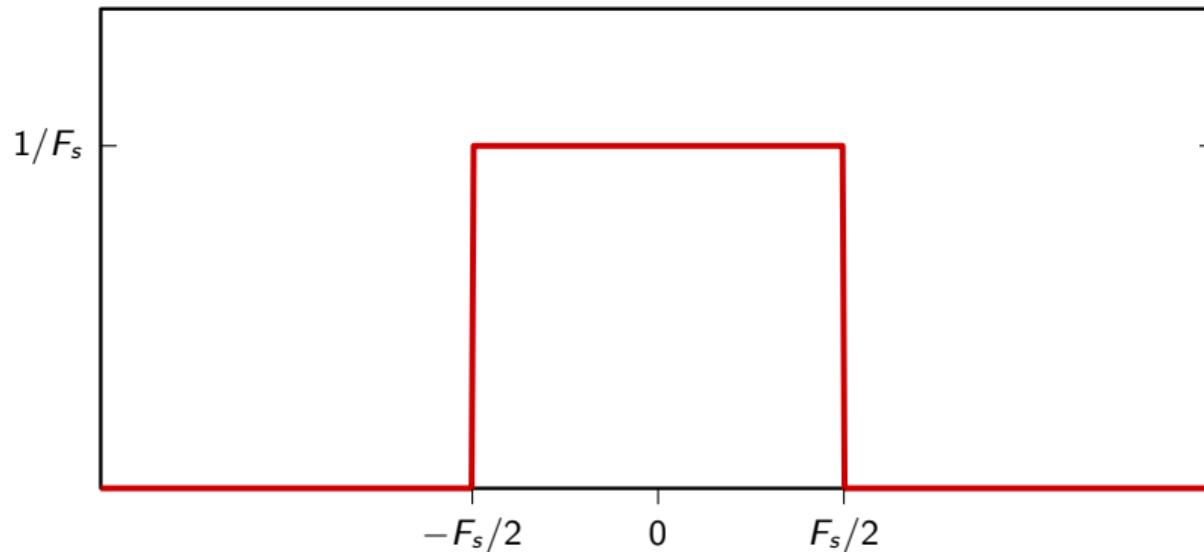
The prototypical bandlimited function

$$\Phi(f) = \frac{1}{F_s} \text{rect}\left(\frac{f}{F_s}\right)$$

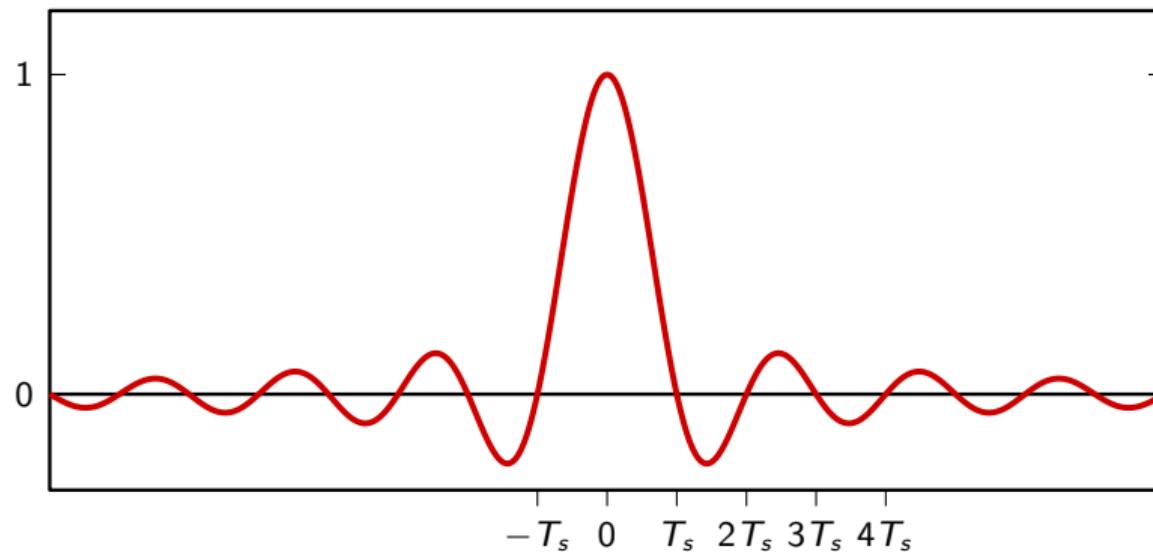
$$\varphi(t) = \text{sinc}\left(\frac{t}{T_s}\right)$$

The prototypical bandlimited function

$$F_s = 1/T_s$$



The prototypical bandlimited function

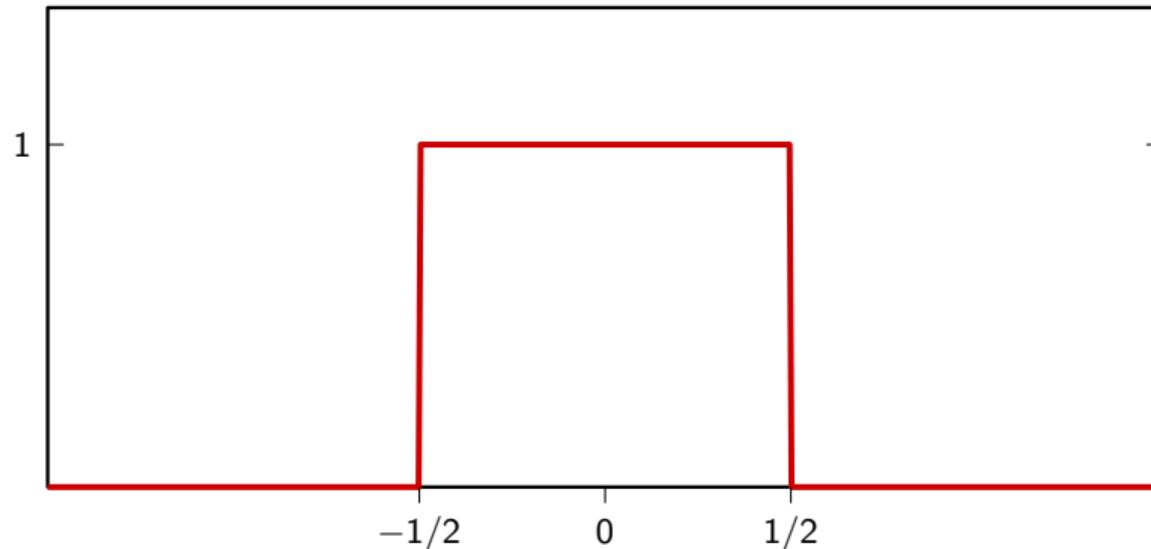


When $T_s = 1$

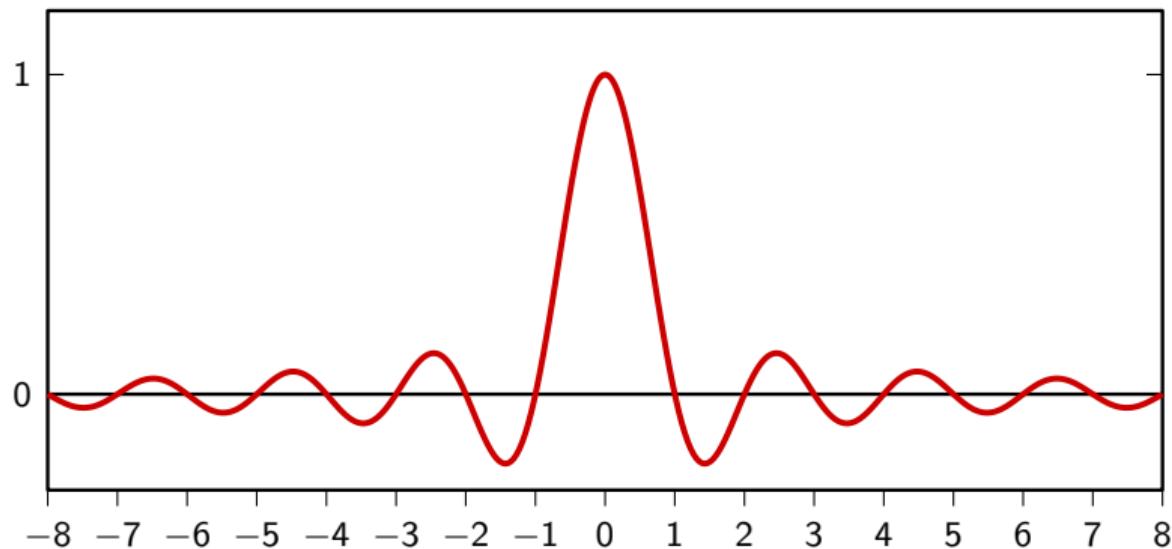
$$\Phi(f) = \text{rect}(f)$$

$$\varphi(t) = \text{sinc}(t)$$

The prototypical bandlimited function ($T_s = 1$)



The prototypical bandlimited function ($T_s = 1$)



interpolation

Overview:

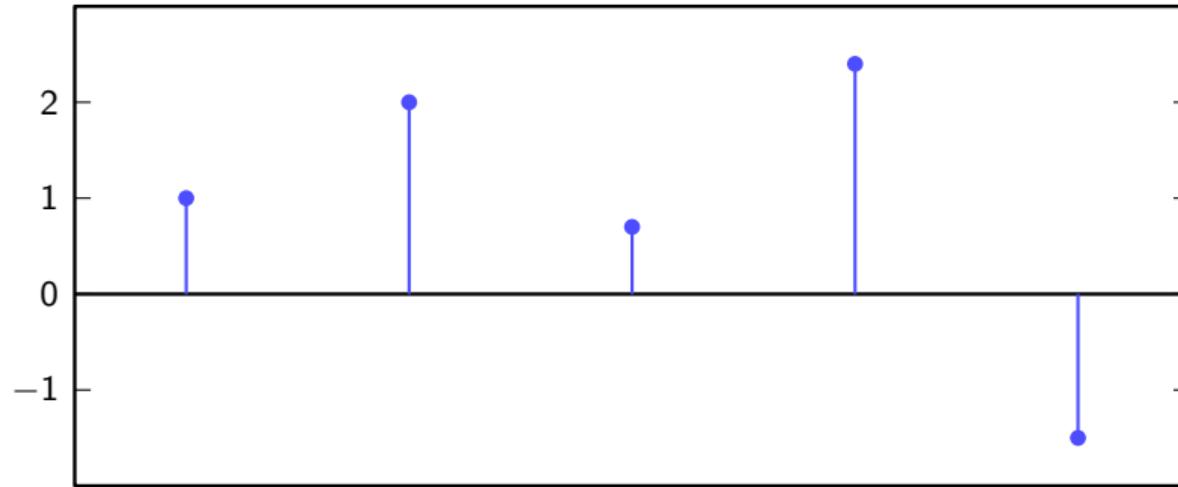
- ▶ Polynomial interpolation
- ▶ Local interpolation
- ▶ Sinc interpolation

Interpolation

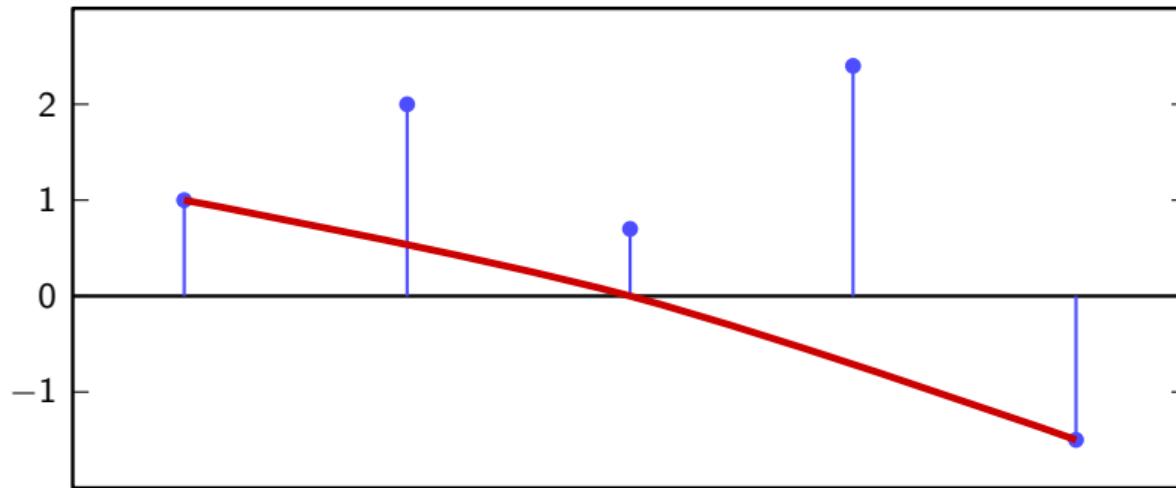
$$x[n] \longrightarrow x(t)$$

“fill the gaps” between samples

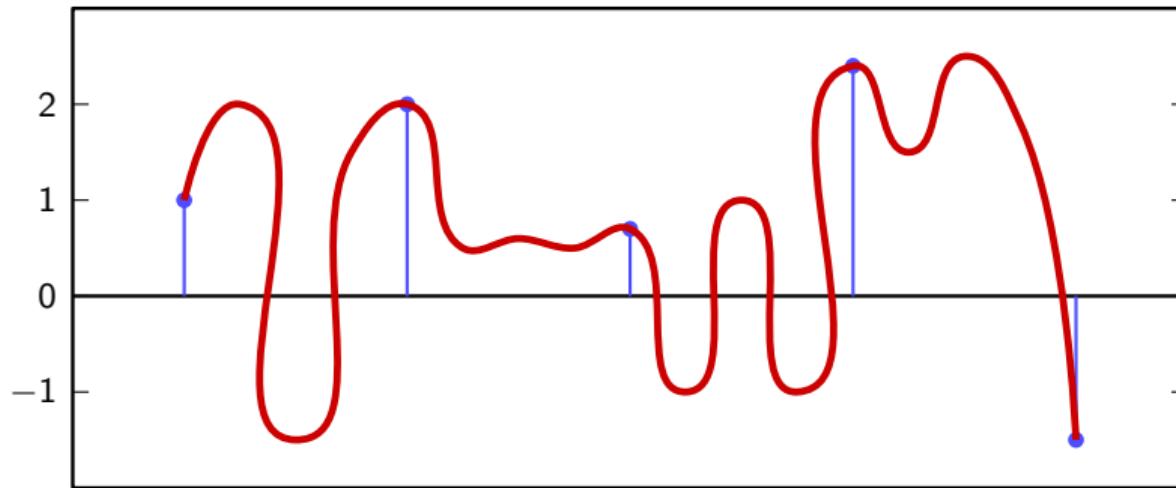
Example



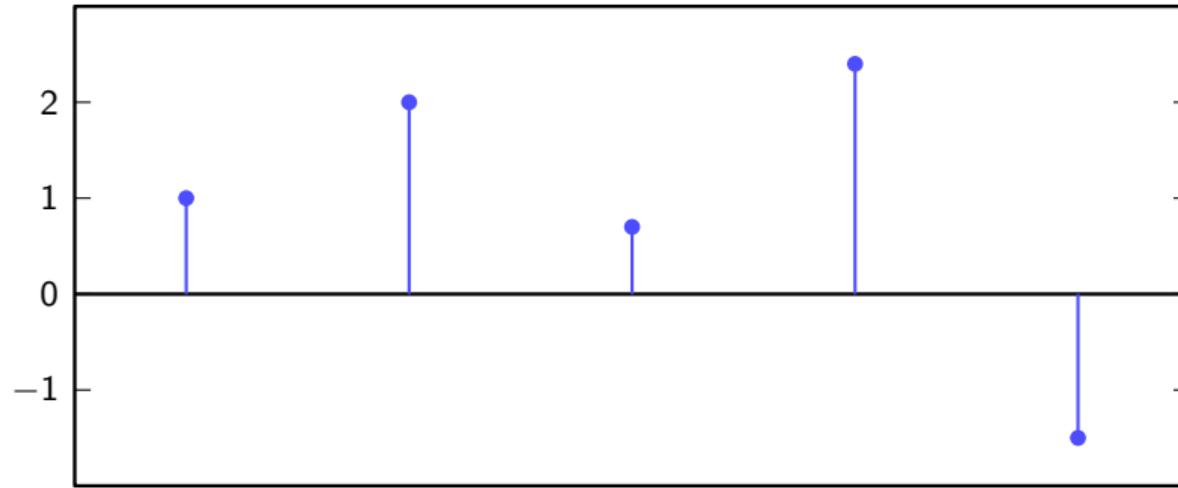
Example



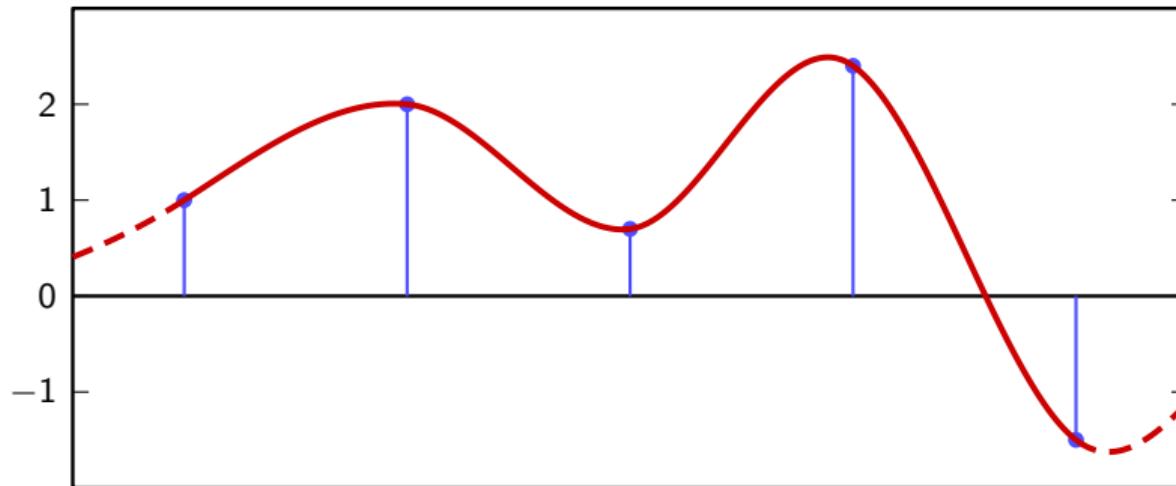
Example



Example



Example



Interpolation requirements

- ▶ decide on T_s
- ▶ make sure $x(nT_s) = x[n]$
- ▶ make sure $x(t)$ is *smooth*

Interpolation requirements

- ▶ decide on T_s
- ▶ make sure $x(nT_s) = x[n]$
- ▶ make sure $x(t)$ is *smooth*

Interpolation requirements

- ▶ decide on T_s
- ▶ make sure $x(nT_s) = x[n]$
- ▶ make sure $x(t)$ is *smooth*

Why smoothness?

- ▶ jumps (1st order discontinuities) would require the signal to move “faster than light” ...
- ▶ 2nd order discontinuities would require infinite acceleration
- ▶ ...
- ▶ the interpolation should be infinitely differentiable
- ▶ “natural” solution: **polynomial interpolation**

Why smoothness?

- ▶ jumps (1st order discontinuities) would require the signal to move “faster than light” ...
- ▶ 2nd order discontinuities would require infinite acceleration
- ▶ ...
- ▶ the interpolation should be infinitely differentiable
- ▶ “natural” solution: **polynomial interpolation**

Why smoothness?

- ▶ jumps (1st order discontinuities) would require the signal to move “faster than light” ...
- ▶ 2nd order discontinuities would require infinite acceleration
- ▶ ...
- ▶ the interpolation should be infinitely differentiable
- ▶ “natural” solution: **polynomial interpolation**

Why smoothness?

- ▶ jumps (1st order discontinuities) would require the signal to move “faster than light” ...
- ▶ 2nd order discontinuities would require infinite acceleration
- ▶ ...
- ▶ the interpolation should be infinitely differentiable
- ▶ “natural” solution: **polynomial interpolation**

Why smoothness?

- ▶ jumps (1st order discontinuities) would require the signal to move “faster than light” ...
- ▶ 2nd order discontinuities would require infinite acceleration
- ▶ ...
- ▶ the interpolation should be infinitely differentiable
- ▶ “natural” solution: **polynomial interpolation**

Polynomial interpolation

- ▶ N points → polynomial of degree $(N - 1)$

- ▶ $p(t) = a_0 + a_1 t + a_2 t^2 + \dots + a_{N-1} t^{(N-1)}$

- ▶ straightforward approach:

$$\left\{ \begin{array}{l} p(0) = x[0] \\ p(T_s) = x[1] \\ p(2T_s) = x[2] \\ \dots \\ p((N-1)T_s) = x[N-1] \end{array} \right.$$

Polynomial interpolation

- ▶ N points → polynomial of degree $(N - 1)$
- ▶ $p(t) = a_0 + a_1 t + a_2 t^2 + \dots + a_{N-1} t^{(N-1)}$
- ▶ straightforward approach:

$$\left\{ \begin{array}{l} p(0) = x[0] \\ p(T_s) = x[1] \\ p(2T_s) = x[2] \\ \dots \\ p((N-1)T_s) = x[N-1] \end{array} \right.$$

Polynomial interpolation

- ▶ N points \rightarrow polynomial of degree $(N - 1)$
- ▶ $p(t) = a_0 + a_1 t + a_2 t^2 + \dots + a_{N-1} t^{(N-1)}$
- ▶ straightforward approach:

$$\left\{ \begin{array}{l} p(0) = x[0] \\ p(T_s) = x[1] \\ p(2T_s) = x[2] \\ \dots \\ p((N-1)T_s) = x[N-1] \end{array} \right.$$

Polynomial interpolation

Without loss of generality:

- ▶ consider a symmetric interval $I_N = [-N, \dots, N]$

- ▶ set $T_s = 1$

$$\left\{ \begin{array}{l} p(-N) = x[-N] \\ p(-N+1) = x[-N+1] \\ \quad \cdots \\ p(0) = x[0] \\ \quad \cdots \\ p(N-1) = x[N-1] \\ p(N) = x[N] \end{array} \right.$$

Polynomial interpolation

Without loss of generality:

- ▶ consider a symmetric interval $I_N = [-N, \dots, N]$
- ▶ set $T_s = 1$

$$\left\{ \begin{array}{l} p(-N) = x[-N] \\ p(-N + 1) = x[-N + 1] \\ \quad \cdots \\ p(0) = x[0] \\ \quad \cdots \\ p(N - 1) = x[N - 1] \\ p(N) = x[N] \end{array} \right.$$

Lagrange interpolation

Let's use the power of vector spaces:

- ▶ P_N : space of degree- $2N$ polynomials over I_N
- ▶ interpolation will be a linear combination of basis vectors for P_N
- ▶ what is a good basis for *interpolation*?

Aside: N -degree polynomial bases on the interval

- ▶ naive basis: $1, t, t^2, \dots, t^N$
- ▶ Legendre basis: orthonormal, increasing degree, good for MSE approximation
- ▶ Chebyshev basis: orthonormal, increasing degree, good for minimax approximation
- ▶ Lagrange polynomials: equal degree, interpolation property

Lagrange interpolation

- ▶ P_N : space of degree- $2N$ polynomials over I_N
- ▶ a basis for P_N is the family of $2N + 1$ Lagrange polynomials

$$L_n^{(N)}(t) = \prod_{\substack{k=-N \\ k \neq n}}^N \frac{t - k}{n - k} \quad n = -N, \dots, N$$

- ▶ interpolation property:

$$L_n^{(N)}(m \in \mathbb{N}) = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{if } n \neq m \end{cases} \quad -N \leq n, m \leq N$$

Lagrange polynomials for I_2

$$L_{-2}^{(2)}(t) = \left(\frac{t+1}{-2+1}\right) \left(\frac{t}{-2}\right) \left(\frac{t-1}{-2-1}\right) \left(\frac{t-2}{-2-2}\right)$$

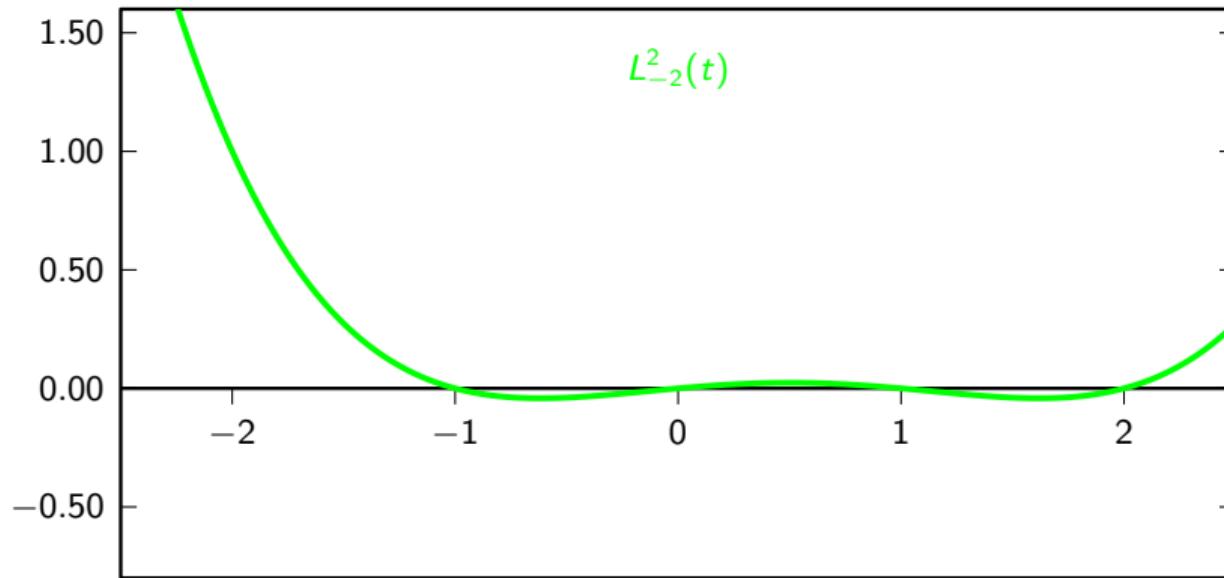
$$L_{-1}^{(2)}(t) = \left(\frac{t+2}{-1+2}\right) \left(\frac{t}{-1}\right) \left(\frac{t-1}{-1-1}\right) \left(\frac{t-2}{-1-2}\right)$$

$$L_0^{(2)}(t) = \left(\frac{t+2}{2}\right) \left(\frac{t+1}{1}\right) \left(\frac{t-1}{-1}\right) \left(\frac{t-2}{-2}\right)$$

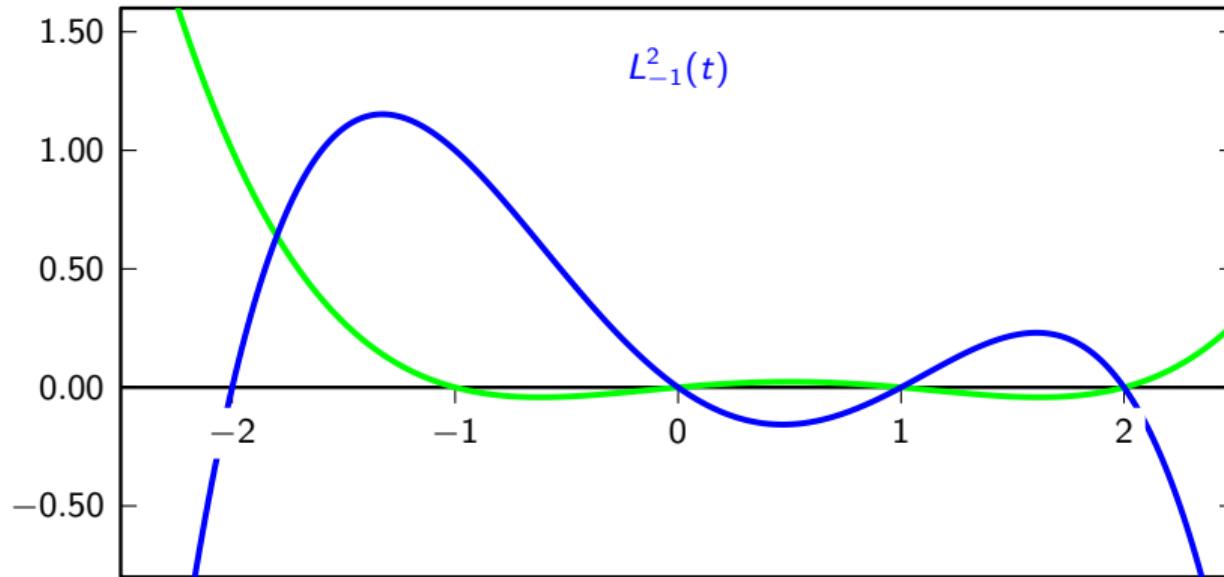
$$L_1^{(2)}(t) = L_{-1}^{(2)}(-t)$$

$$L_2^{(2)}(t) = L_{-2}^{(2)}(-t)$$

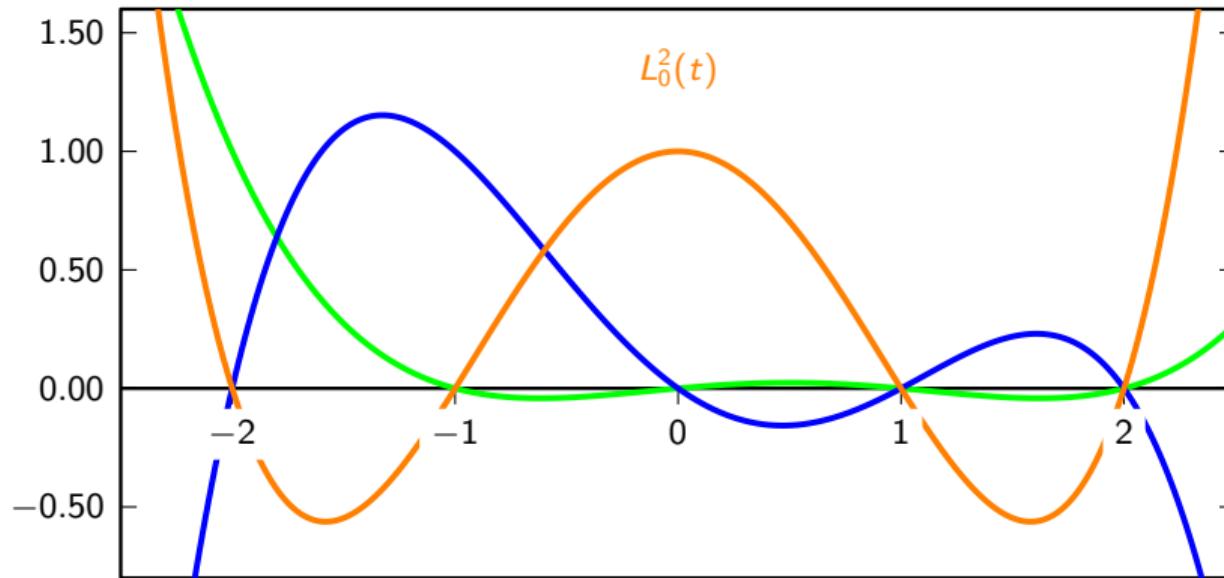
Lagrange interpolation polynomials



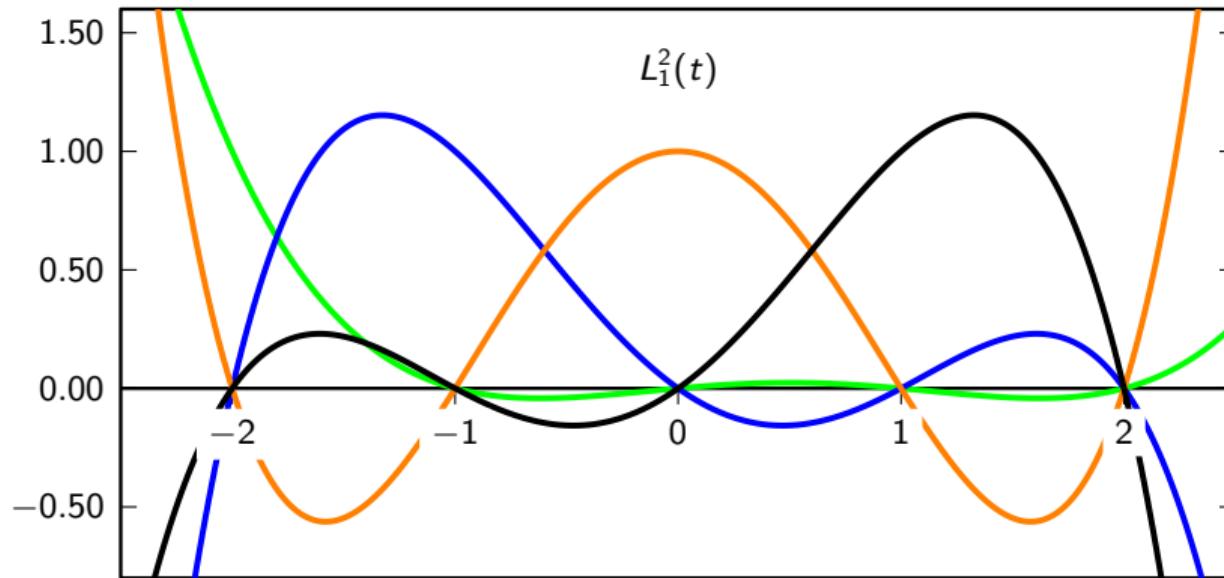
Lagrange interpolation polynomials



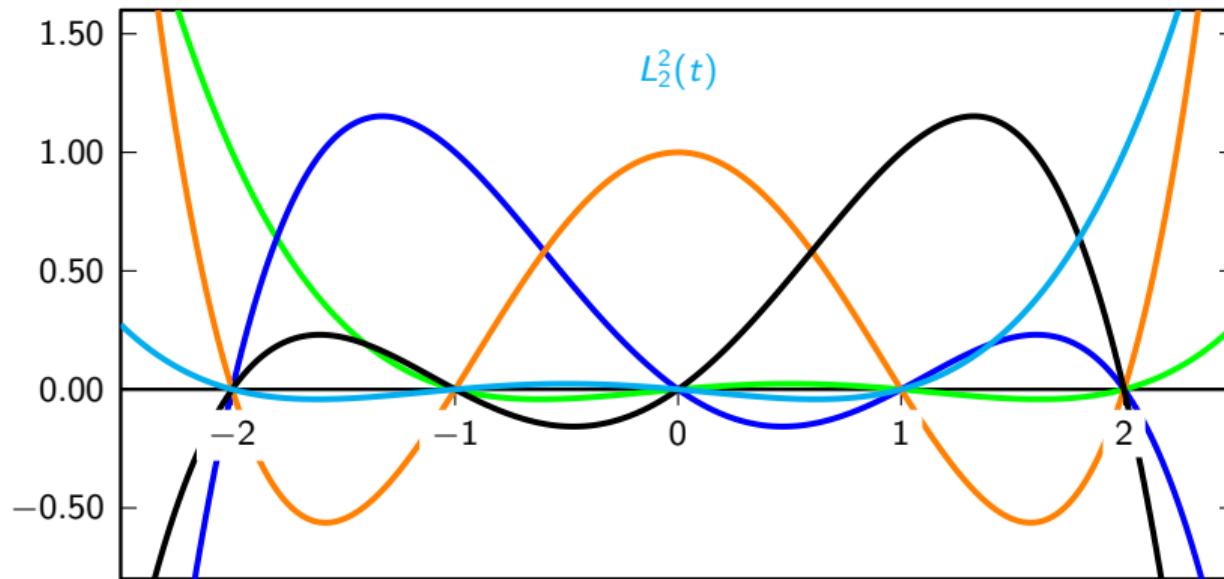
Lagrange interpolation polynomials



Lagrange interpolation polynomials



Lagrange interpolation polynomials



Lagrange interpolation

$$p(t) = \sum_{n=-N}^N x[n] L_n^{(N)}(t)$$

Lagrange interpolation

The Lagrange interpolation *is* the unique polynomial interpolation:

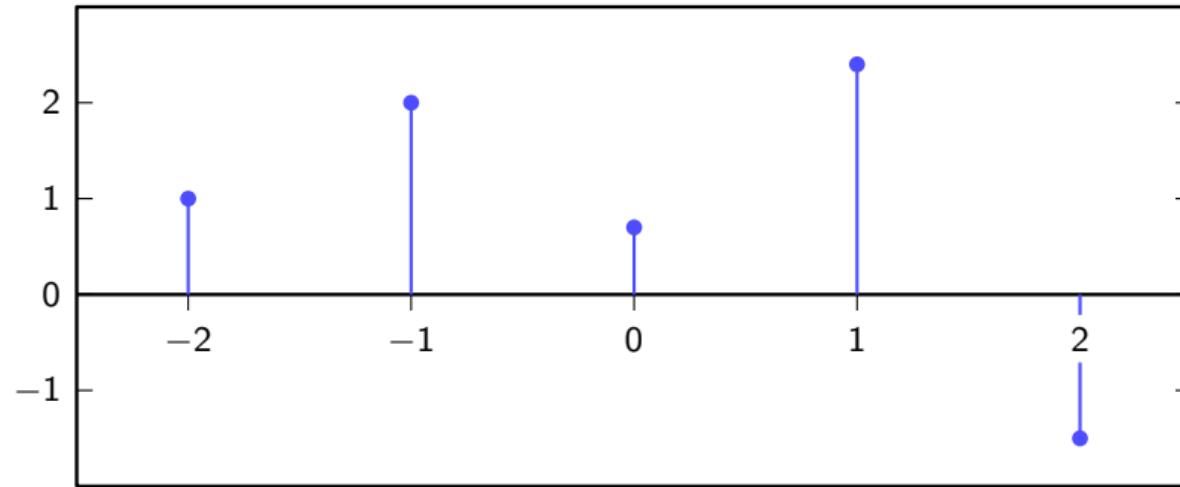
- ▶ polynomial of degree $2N$ through $2N + 1$ points is unique
- ▶ the Lagrangian interpolator satisfies

$$p(n) = x[n] \quad \text{for } -N \leq n \leq N$$

since

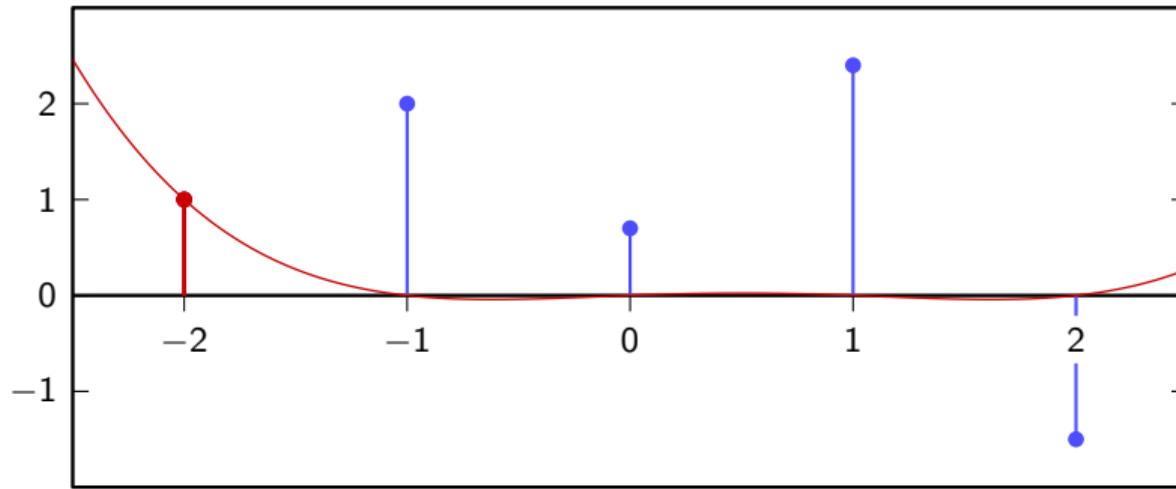
$$L_n^{(N)}(m) = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{if } n \neq m \end{cases} \quad -N \leq n, m \leq N$$

Lagrange interpolation



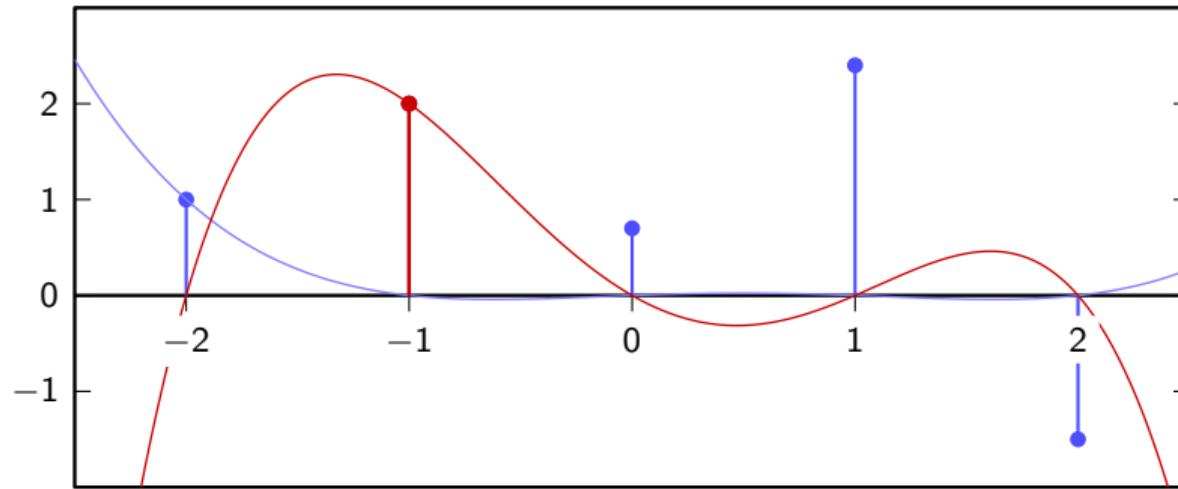
Lagrange interpolation

$$x[-2]L_{-2}^{(2)}(t)$$



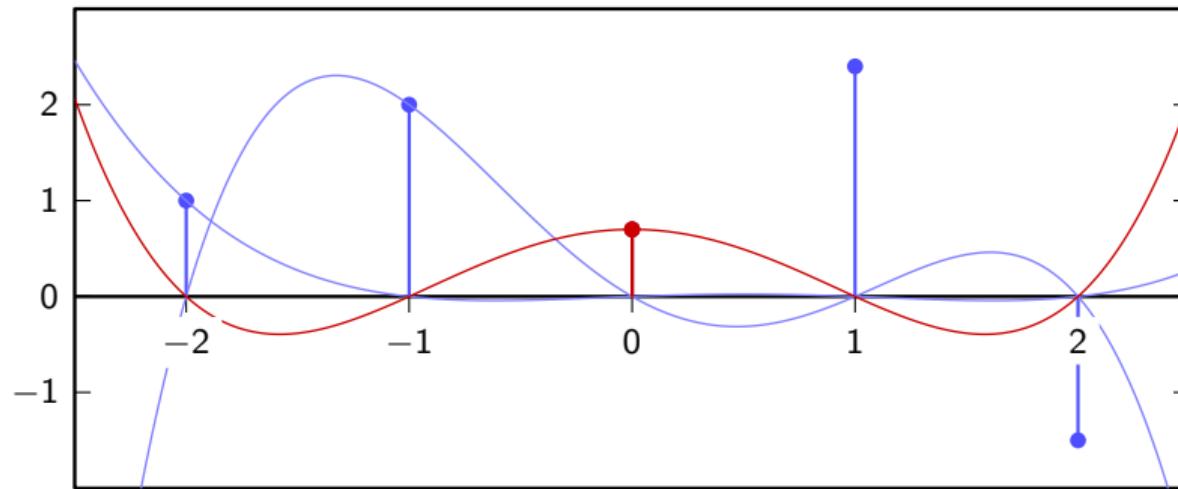
Lagrange interpolation

$$x[-1]L_{-1}^{(2)}(t)$$



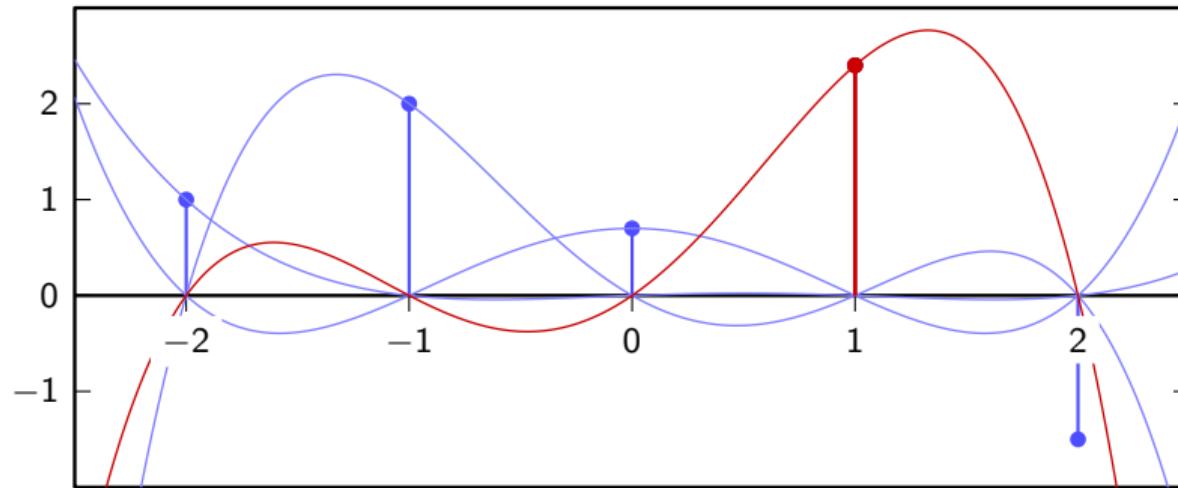
Lagrange interpolation

$$x[0]L_0^{(2)}(t)$$



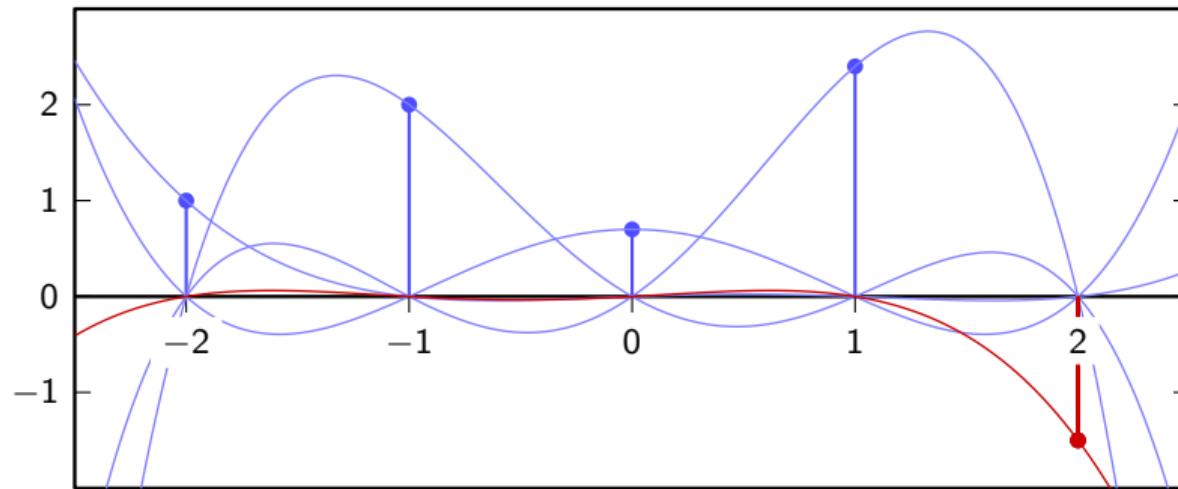
Lagrange interpolation

$$x[1]L_1^{(2)}(t)$$

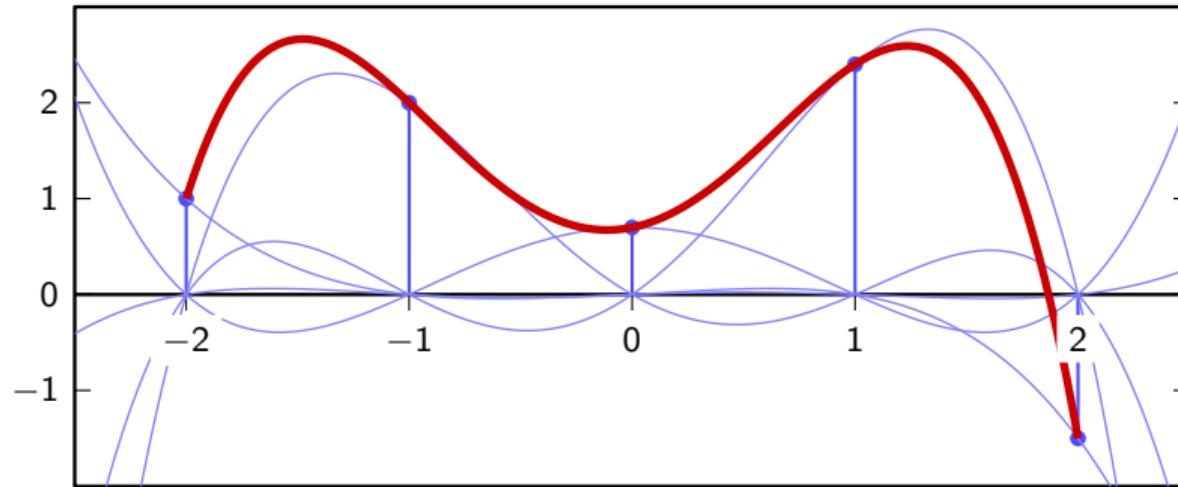


Lagrange interpolation

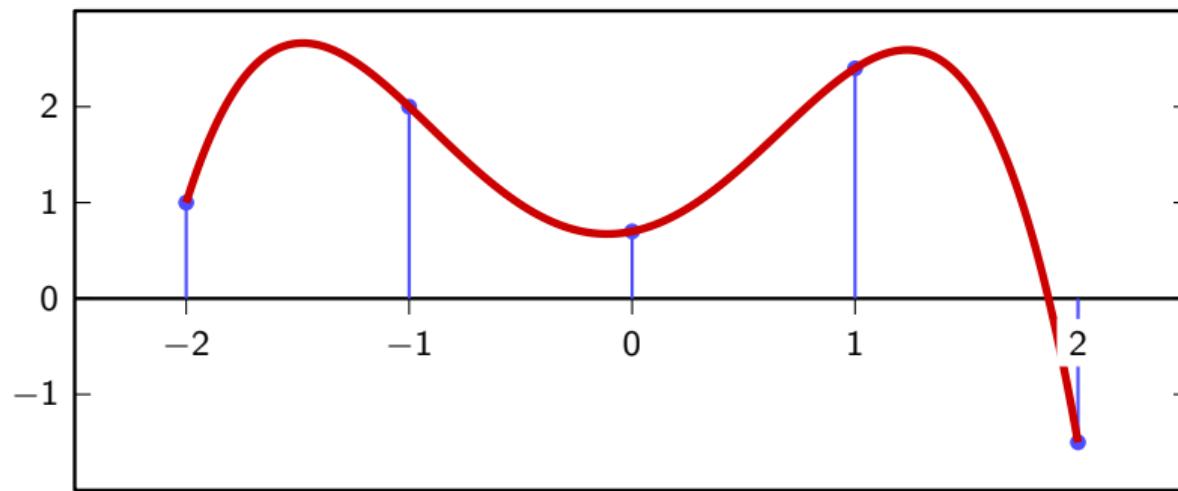
$$x[2]L_2^{(2)}(t)$$



Lagrange interpolation



Lagrange interpolation



Polynomial interpolation

key property:

- ▶ maximally smooth (infinitely many continuous derivatives)

drawback:

- ▶ interpolation “machine” depend on N : we need to use a different set of polynomials if the length of the dataset changes

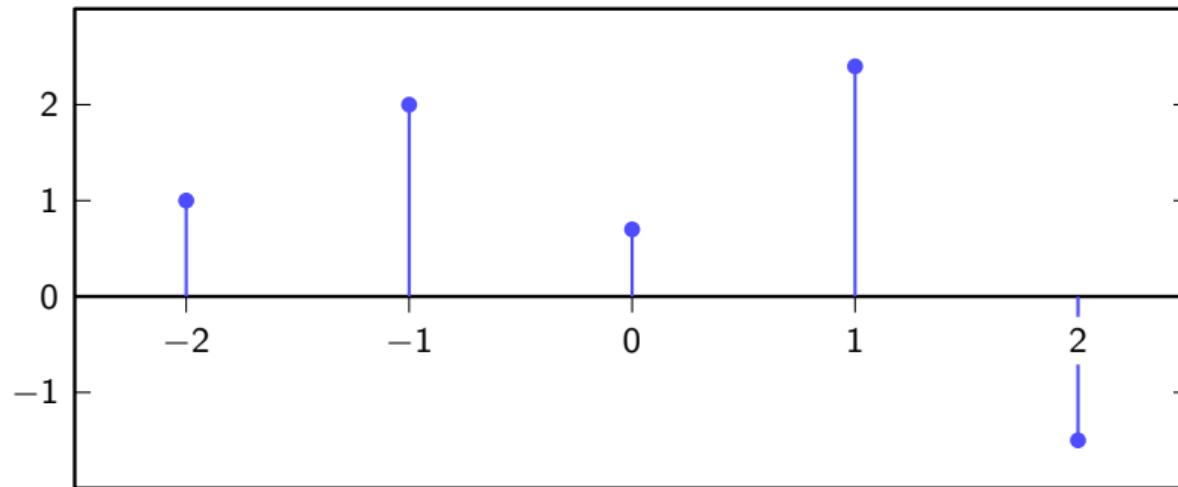
Relaxing the interpolation requirements

- ▶ decide on T_s
- ▶ make sure $x(nT_s) = x[n]$
- ▶ make sure $x(t)$ is *smooth*

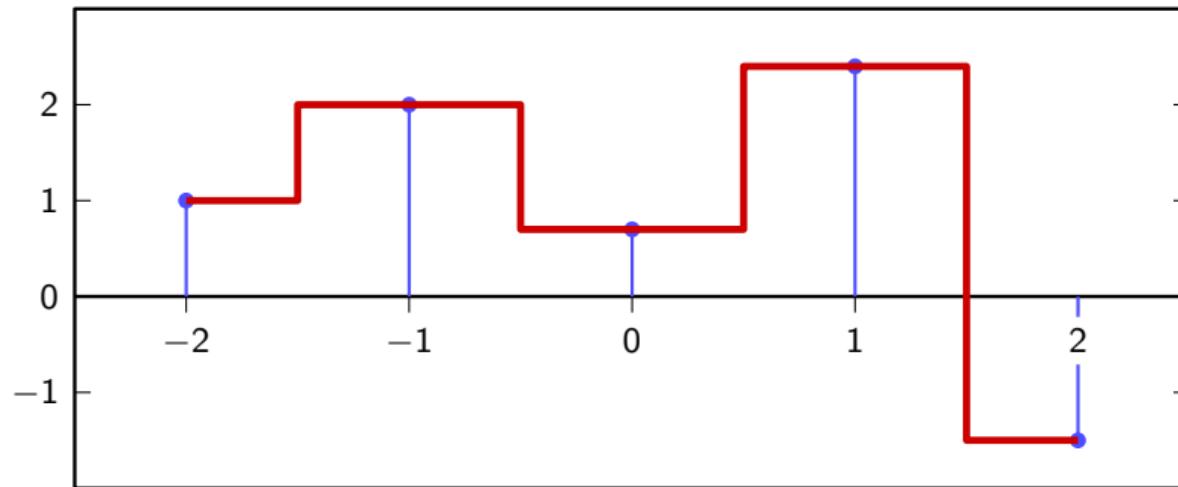
Relaxing the interpolation requirements

- ▶ decide on T_s
- ▶ make sure $x(nT_s) = x[n]$
- ▶ make sure $x(t)$ is *smooth*

Zero-order interpolation



Zero-order interpolation



Zero-order interpolation

► $x(t) = x[\lfloor t + 0.5 \rfloor], \quad -N \leq t \leq N$

► $x(t) = \sum_{n=-N}^N x[n] \text{rect}(t - n)$

► interpolation kernel: $i_0(t) = \text{rect}(t)$

► $i_0(t)$: “zero-order hold”

► interpolator's support is 1

► interpolation is not even continuous

Zero-order interpolation

$$\blacktriangleright x(t) = x[\lfloor t + 0.5 \rfloor], \quad -N \leq t \leq N$$

$$\blacktriangleright x(t) = \sum_{n=-N}^N x[n] \operatorname{rect}(t - n)$$

- ▶ interpolation kernel: $i_0(t) = \operatorname{rect}(t)$
- ▶ $i_0(t)$: “zero-order hold”
- ▶ interpolator’s support is 1
- ▶ interpolation is not even continuous

Zero-order interpolation

► $x(t) = x[\lfloor t + 0.5 \rfloor], \quad -N \leq t \leq N$

► $x(t) = \sum_{n=-N}^N x[n] \operatorname{rect}(t - n)$

► interpolation kernel: $i_0(t) = \operatorname{rect}(t)$

► $i_0(t)$: “zero-order hold”

► interpolator's support is 1

► interpolation is not even continuous

Zero-order interpolation

► $x(t) = x[\lfloor t + 0.5 \rfloor], \quad -N \leq t \leq N$

► $x(t) = \sum_{n=-N}^N x[n] \operatorname{rect}(t - n)$

► interpolation kernel: $i_0(t) = \operatorname{rect}(t)$

► $i_0(t)$: “zero-order hold”

► interpolator's support is 1

► interpolation is not even continuous

Zero-order interpolation

► $x(t) = x[\lfloor t + 0.5 \rfloor], \quad -N \leq t \leq N$

► $x(t) = \sum_{n=-N}^N x[n] \operatorname{rect}(t - n)$

► interpolation kernel: $i_0(t) = \operatorname{rect}(t)$

► $i_0(t)$: “zero-order hold”

► interpolator's support is 1

► interpolation is not even continuous

Zero-order interpolation

► $x(t) = x[\lfloor t + 0.5 \rfloor], \quad -N \leq t \leq N$

► $x(t) = \sum_{n=-N}^N x[n] \operatorname{rect}(t - n)$

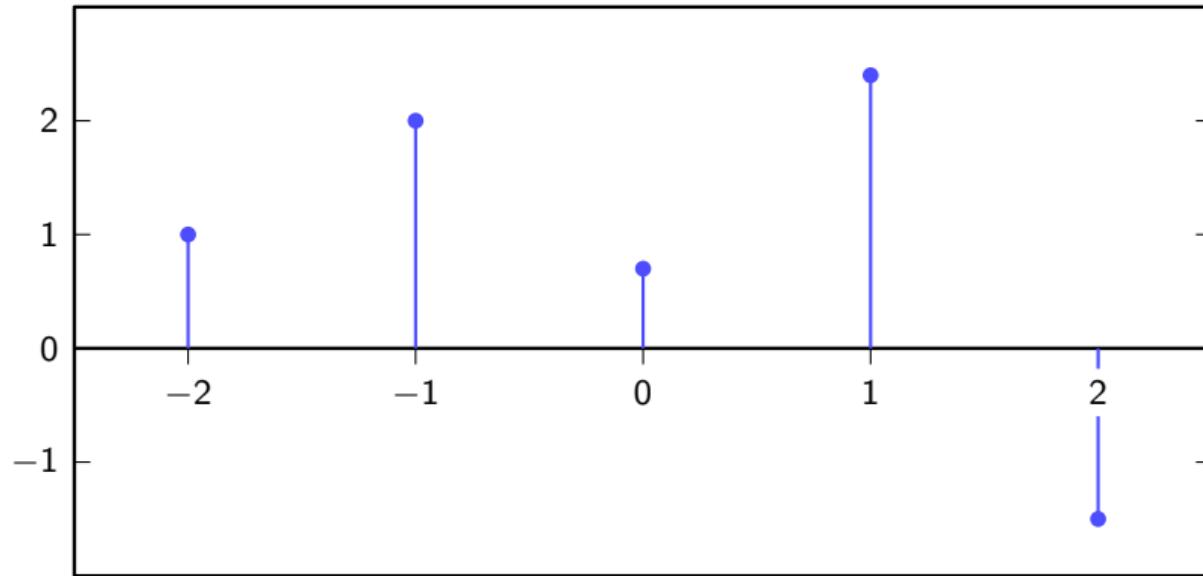
► interpolation kernel: $i_0(t) = \operatorname{rect}(t)$

► $i_0(t)$: “zero-order hold”

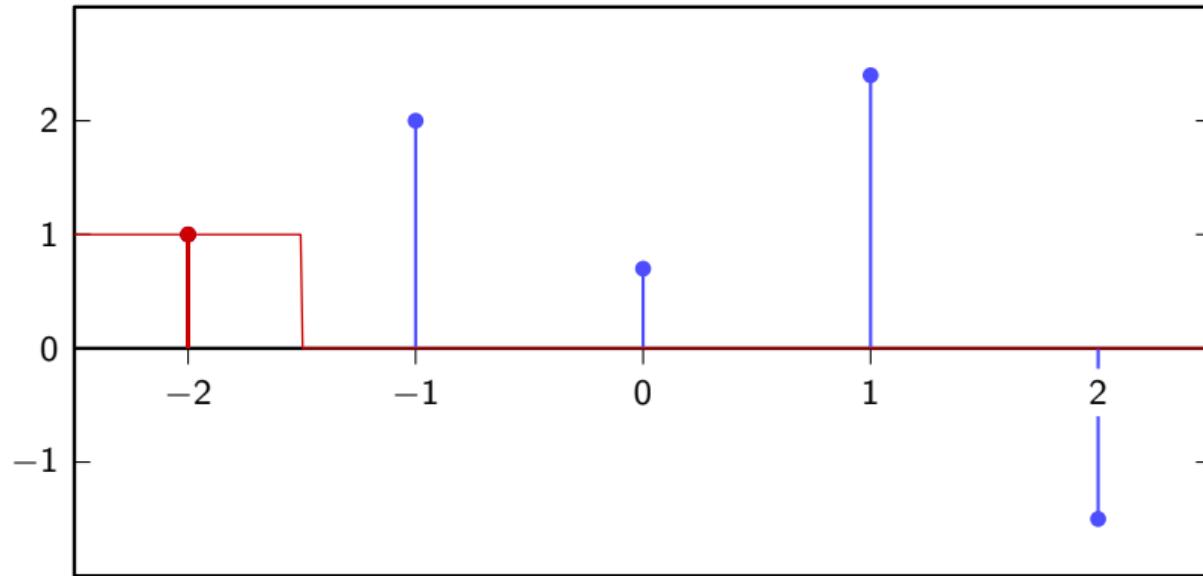
► interpolator's support is 1

► interpolation is not even continuous

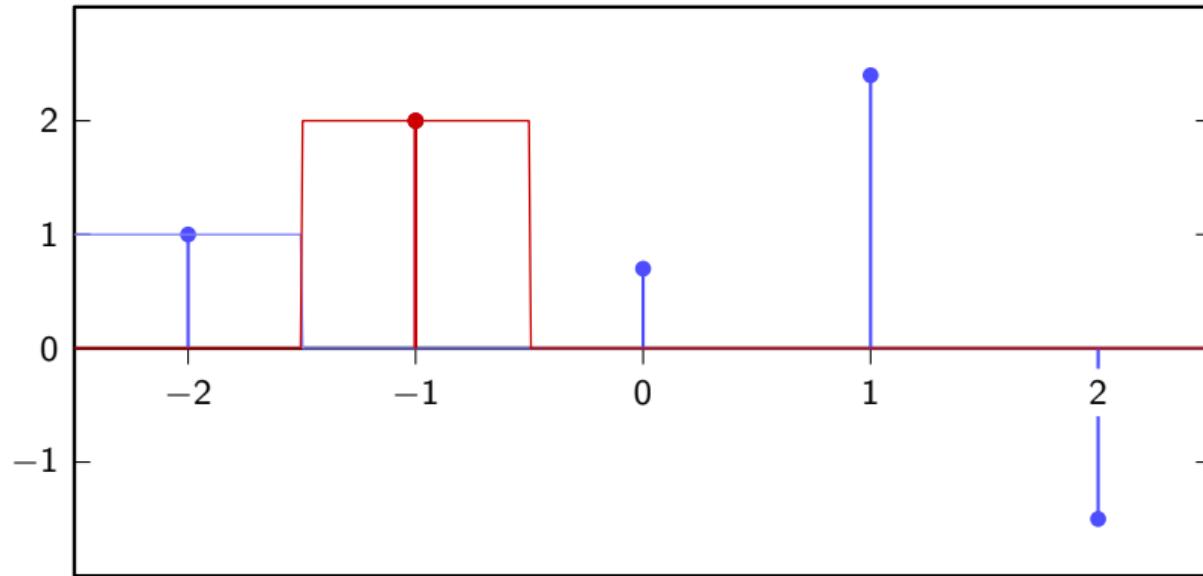
Zero-order interpolation



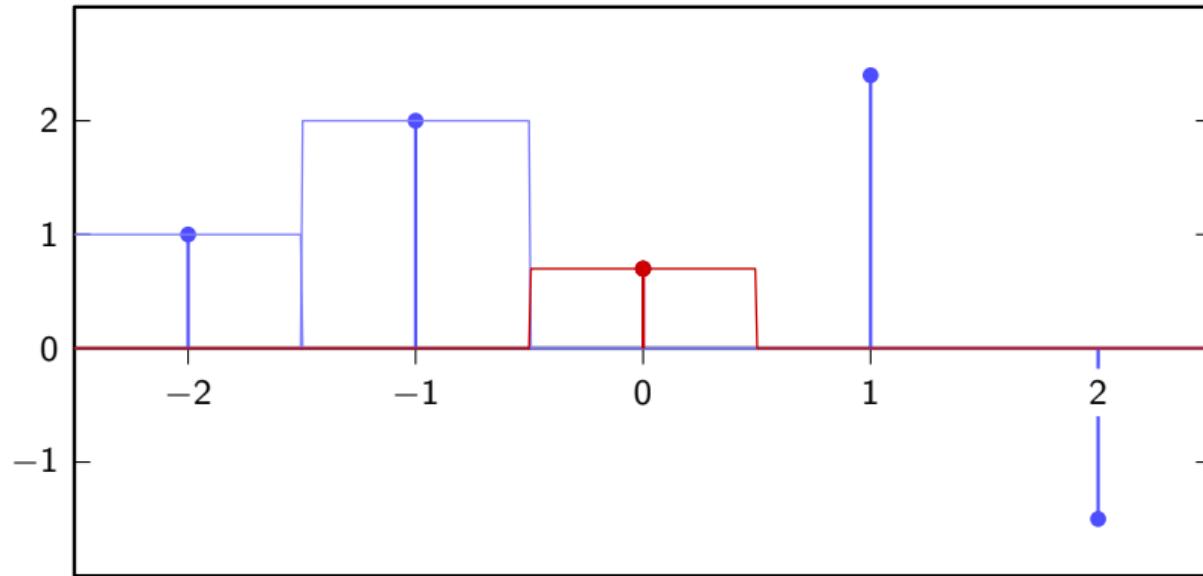
Zero-order interpolation



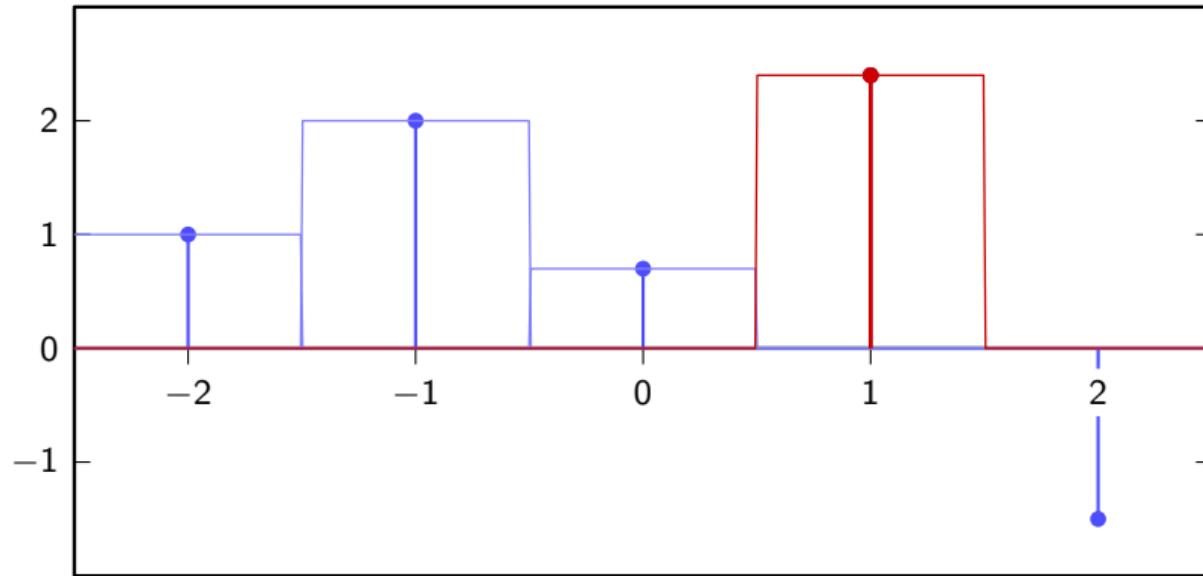
Zero-order interpolation



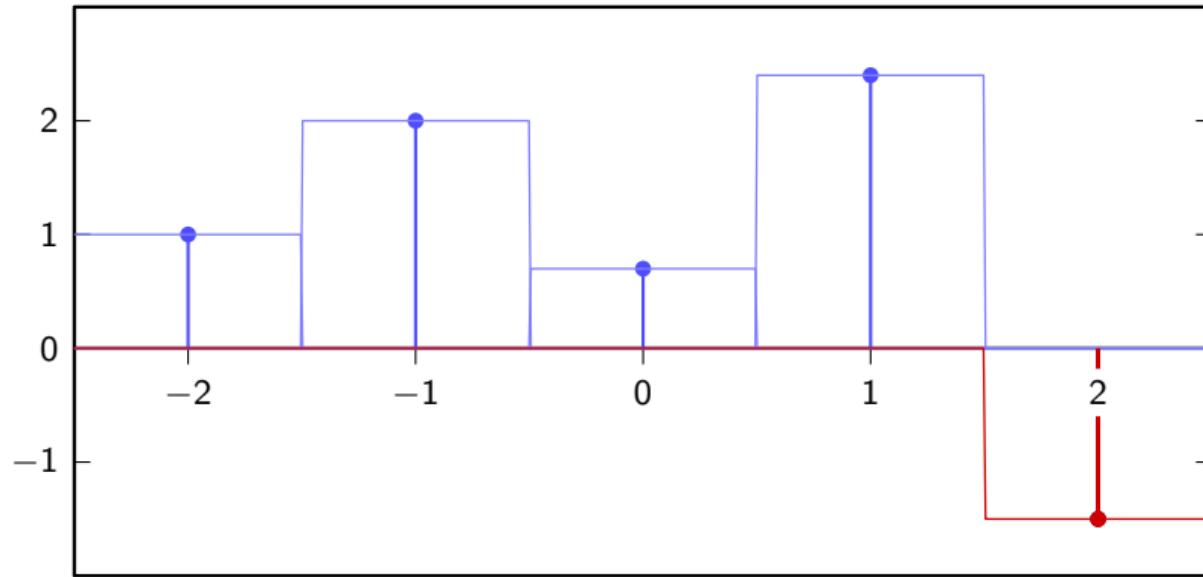
Zero-order interpolation



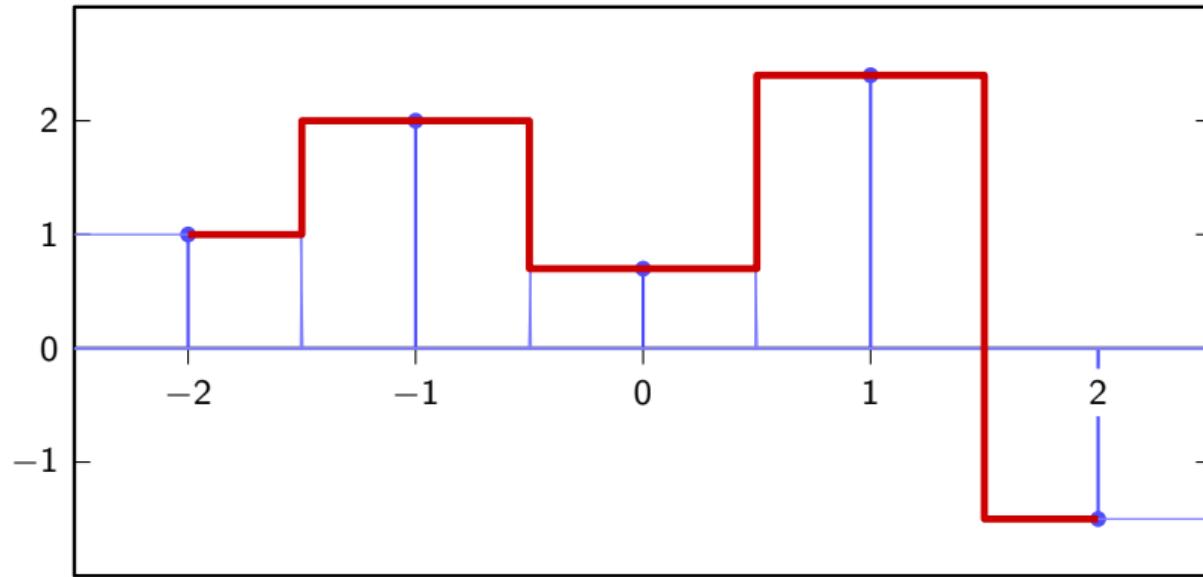
Zero-order interpolation



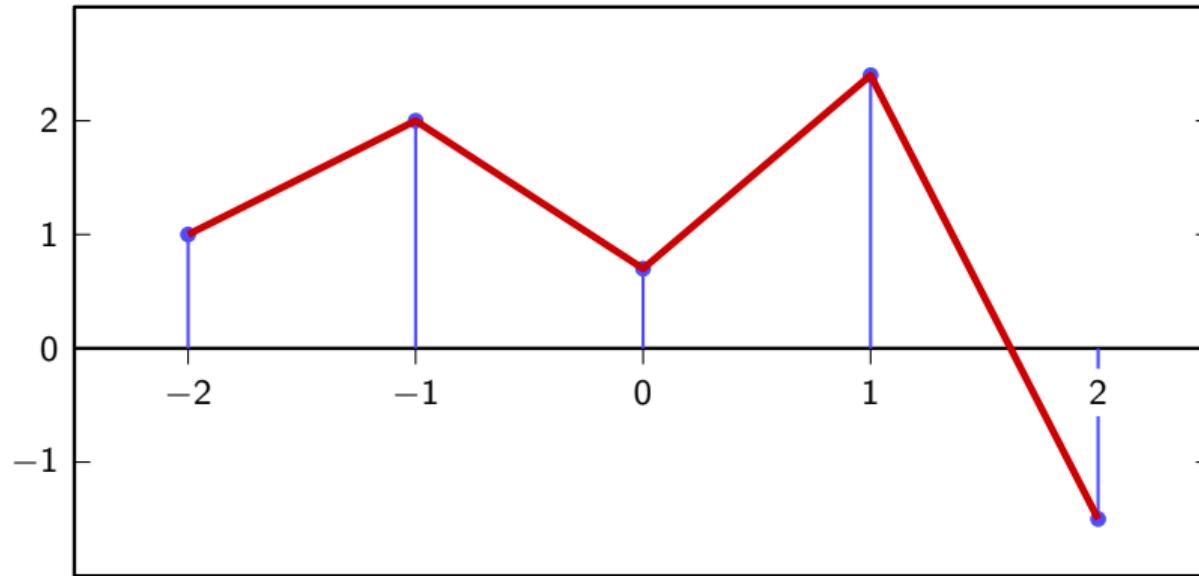
Zero-order interpolation



Zero-order interpolation



First-order interpolation



First-order interpolation

- ▶ “connect the dots” strategy

- ▶
$$x(t) = \sum_{n=-N}^N x[n] i_1(t - n)$$

- ▶ interpolation kernel:

$$i_1(t) = \begin{cases} 1 - |t| & |t| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

- ▶ interpolator's support is 2
- ▶ interpolation is continuous but derivative is not

First-order interpolation

- ▶ “connect the dots” strategy

- ▶
$$x(t) = \sum_{n=-N}^N x[n] i_1(t - n)$$

- ▶ interpolation kernel:

$$i_1(t) = \begin{cases} 1 - |t| & |t| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

- ▶ interpolator's support is 2
- ▶ interpolation is continuous but derivative is not

First-order interpolation

- ▶ “connect the dots” strategy

- ▶
$$x(t) = \sum_{n=-N}^N x[n] i_1(t - n)$$

- ▶ interpolation kernel:

$$i_1(t) = \begin{cases} 1 - |t| & |t| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

- ▶ interpolator's support is 2
- ▶ interpolation is continuous but derivative is not

First-order interpolation

- ▶ “connect the dots” strategy

- ▶
$$x(t) = \sum_{n=-N}^N x[n] i_1(t - n)$$

- ▶ interpolation kernel:

$$i_1(t) = \begin{cases} 1 - |t| & |t| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

- ▶ interpolator's support is 2
- ▶ interpolation is continuous but derivative is not

First-order interpolation

- ▶ “connect the dots” strategy

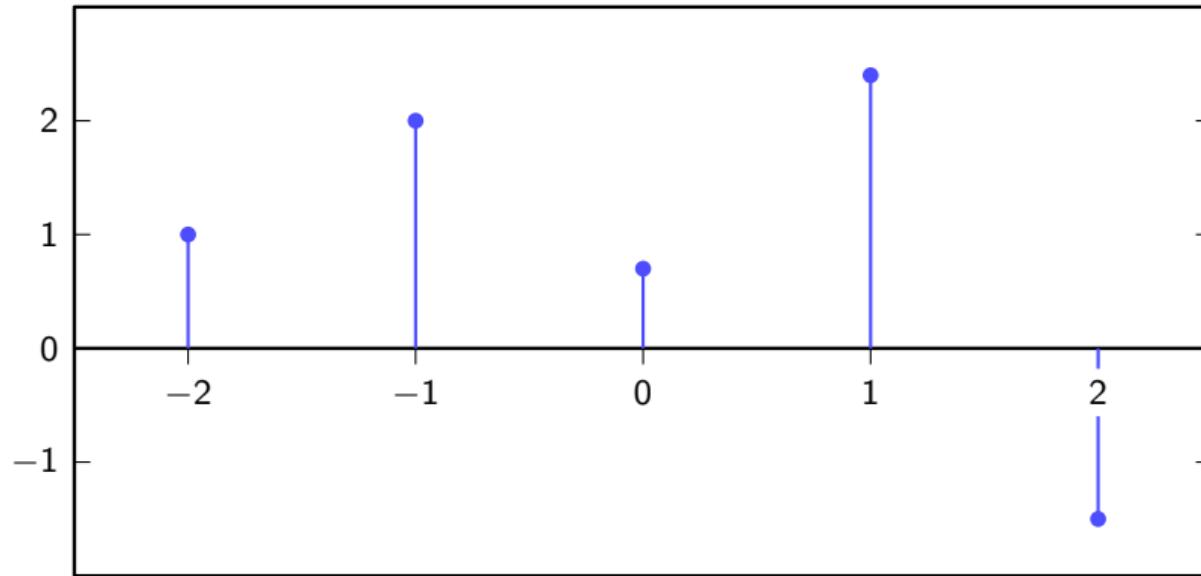
- ▶
$$x(t) = \sum_{n=-N}^N x[n] i_1(t - n)$$

- ▶ interpolation kernel:

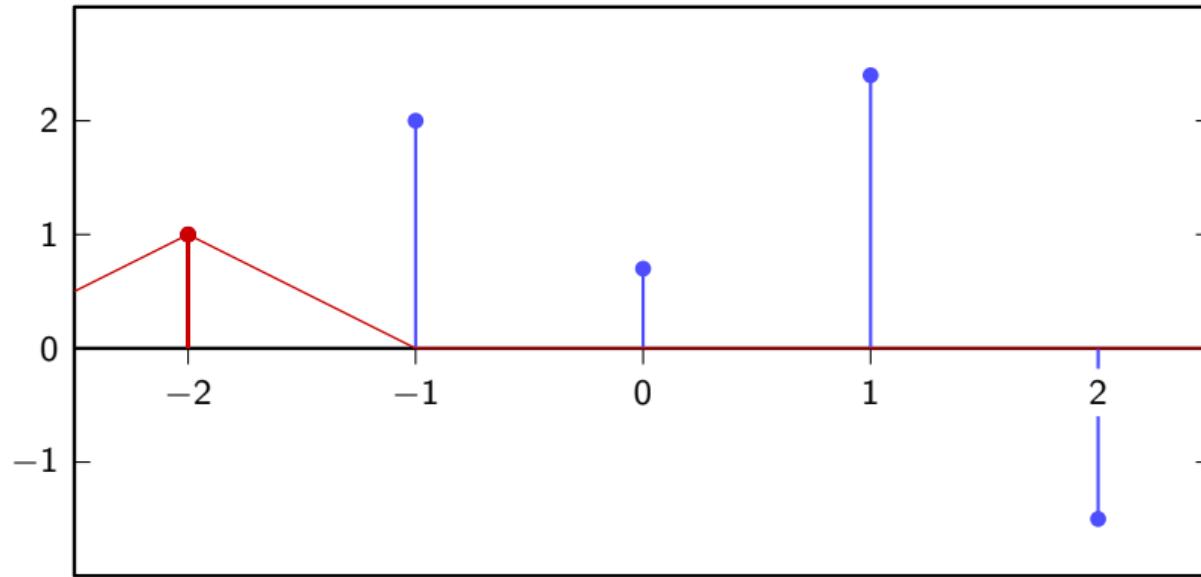
$$i_1(t) = \begin{cases} 1 - |t| & |t| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

- ▶ interpolator's support is 2
- ▶ interpolation is continuous but derivative is not

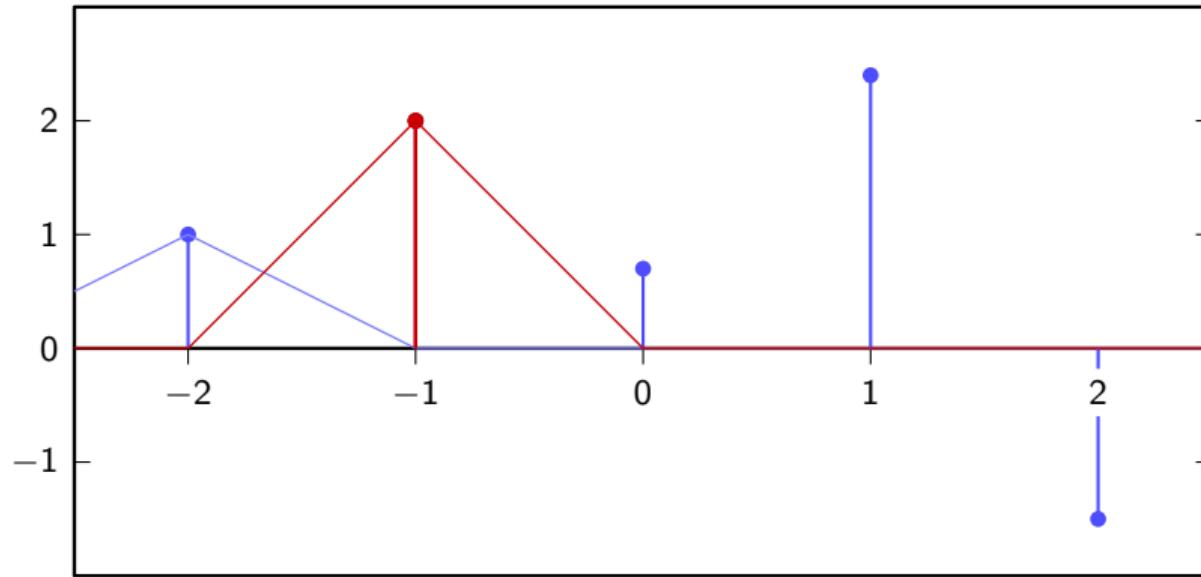
First-order interpolation



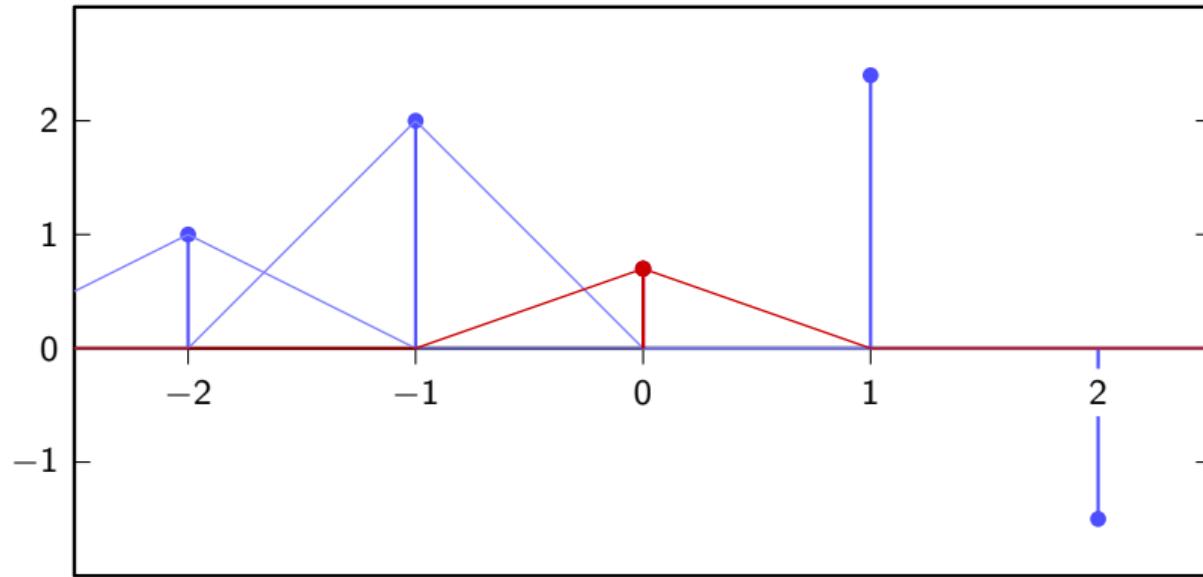
First-order interpolation



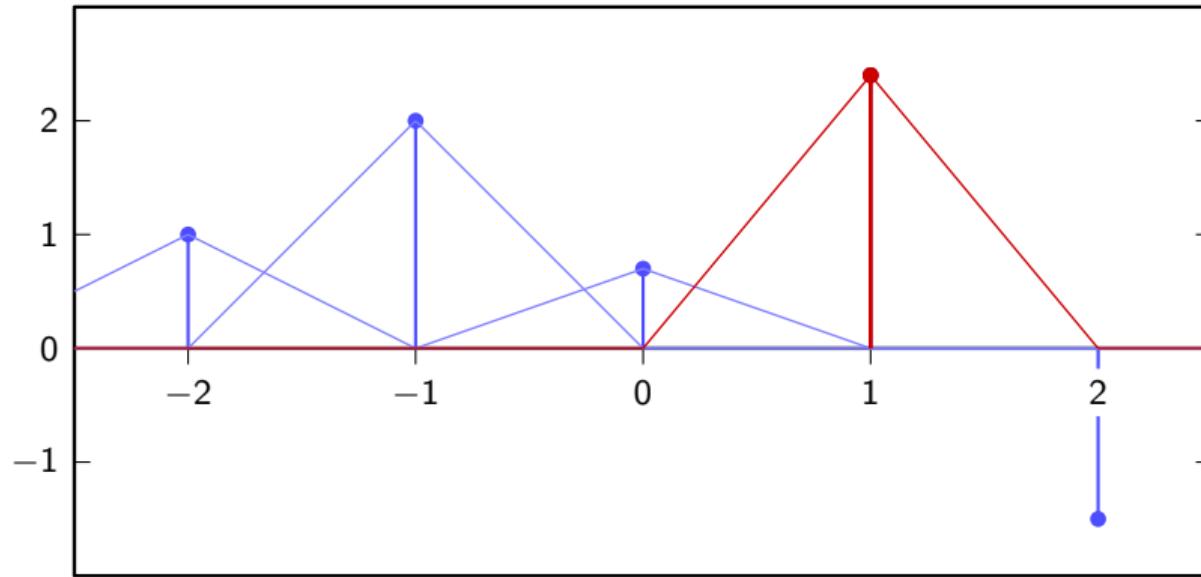
First-order interpolation



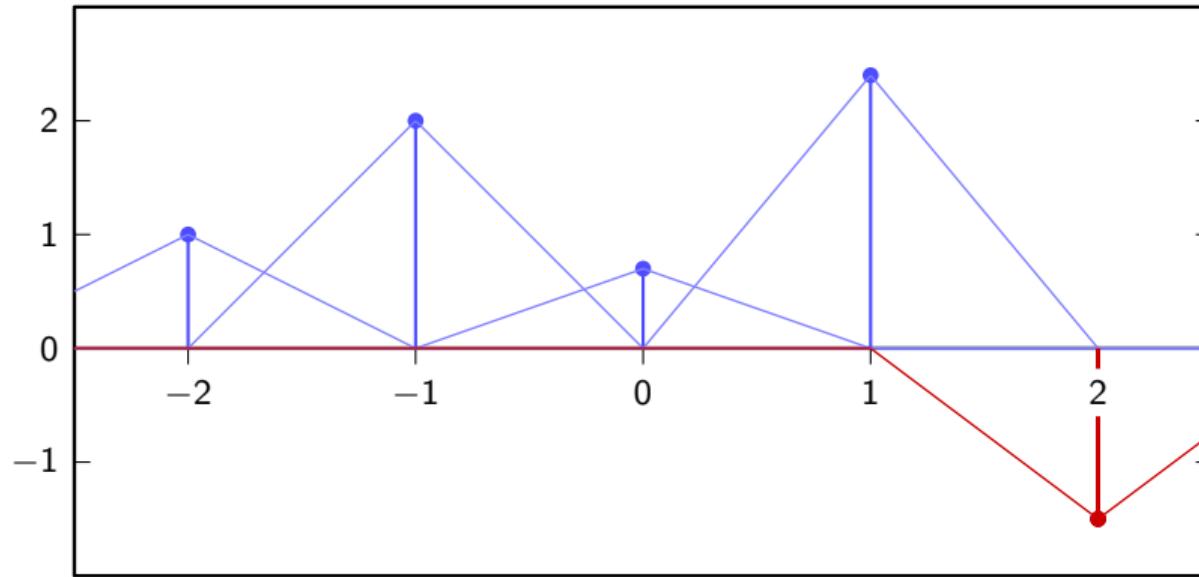
First-order interpolation



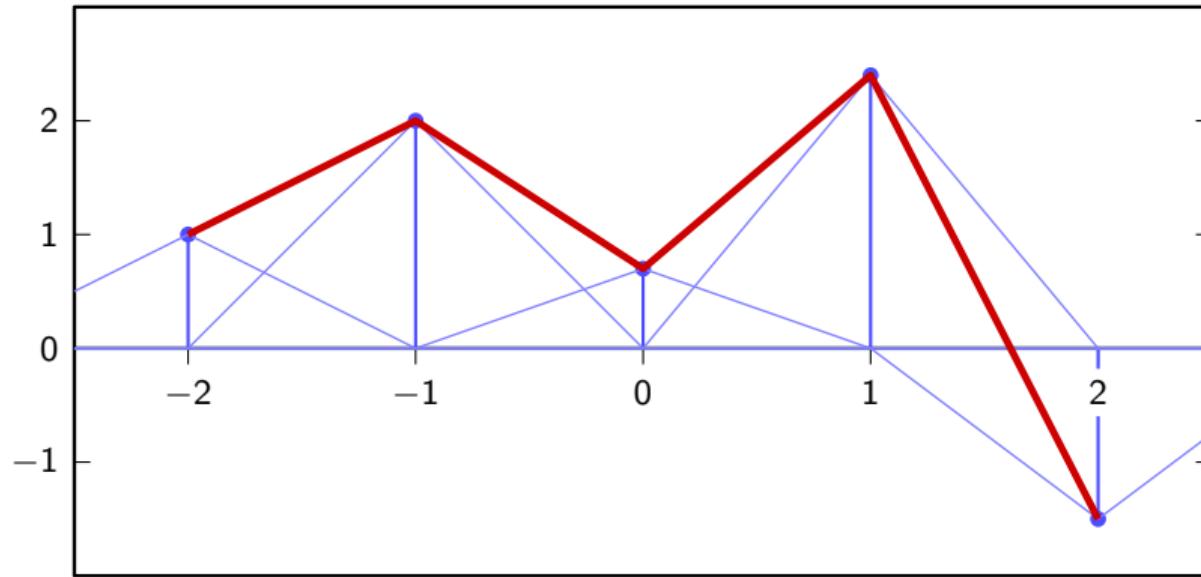
First-order interpolation



First-order interpolation



First-order interpolation



Third-order interpolation

$$\blacktriangleright x(t) = \sum_{n=-N}^N x[n] i_3(t - n)$$

- ▶ interpolation kernel obtained by splicing two cubic polynomials
- ▶ interpolator's support is 4
- ▶ interpolation is continuous up to second derivative

Third-order interpolation

- ▶
$$x(t) = \sum_{n=-N}^N x[n] i_3(t - n)$$
- ▶ interpolation kernel obtained by splicing two cubic polynomials
- ▶ interpolator's support is 4
- ▶ interpolation is continuous up to second derivative

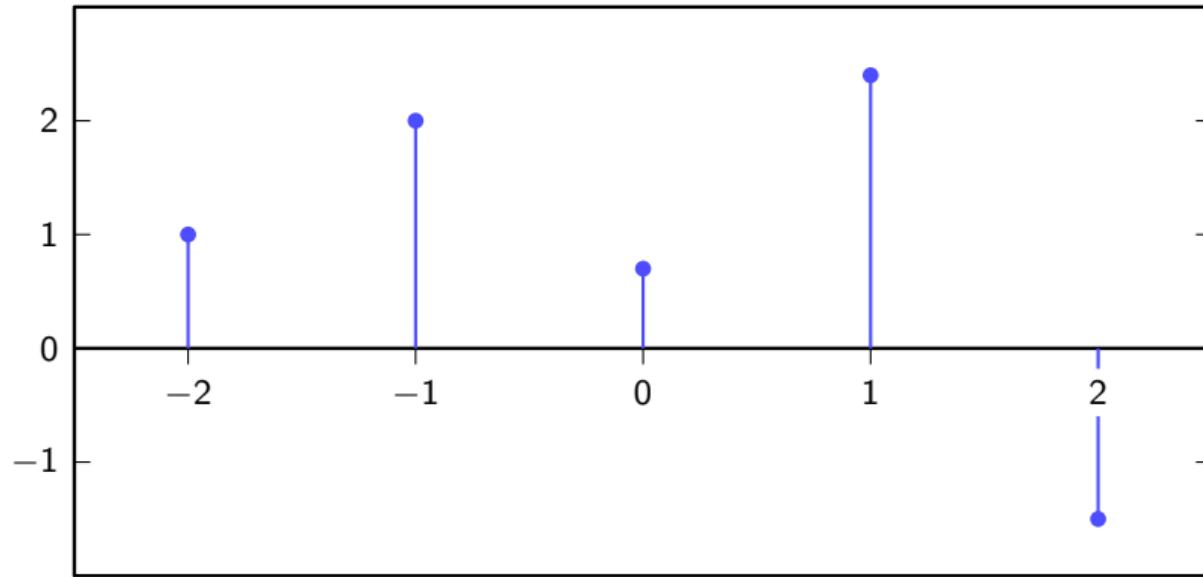
Third-order interpolation

- ▶
$$x(t) = \sum_{n=-N}^N x[n] i_3(t - n)$$
- ▶ interpolation kernel obtained by splicing two cubic polynomials
- ▶ interpolator's support is 4
- ▶ interpolation is continuous up to second derivative

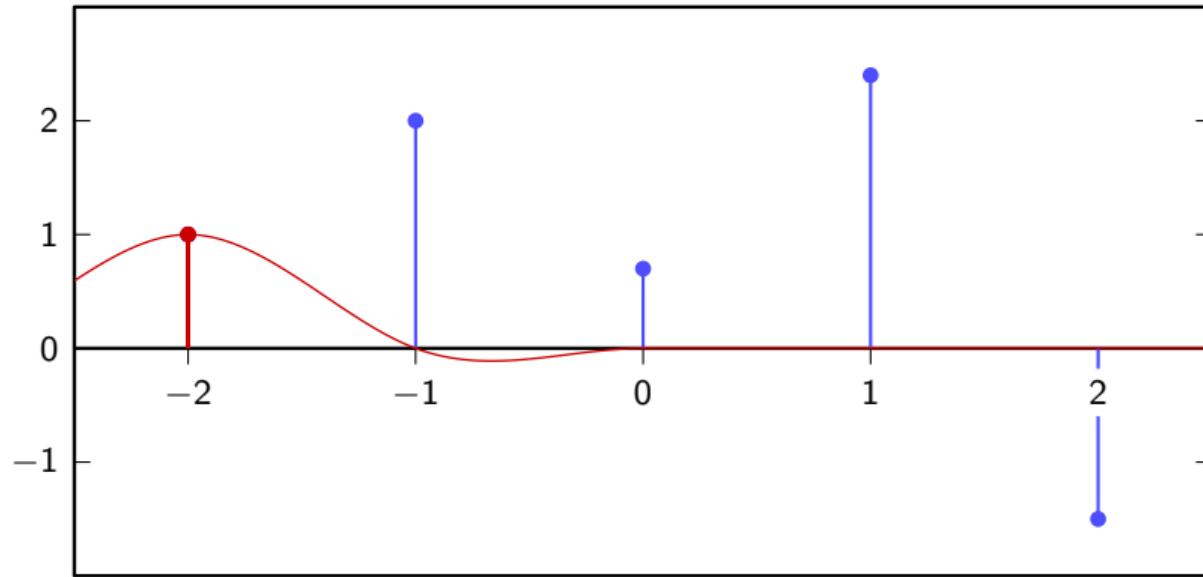
Third-order interpolation

- ▶
$$x(t) = \sum_{n=-N}^N x[n] i_3(t - n)$$
- ▶ interpolation kernel obtained by splicing two cubic polynomials
- ▶ interpolator's support is 4
- ▶ interpolation is continuous up to second derivative

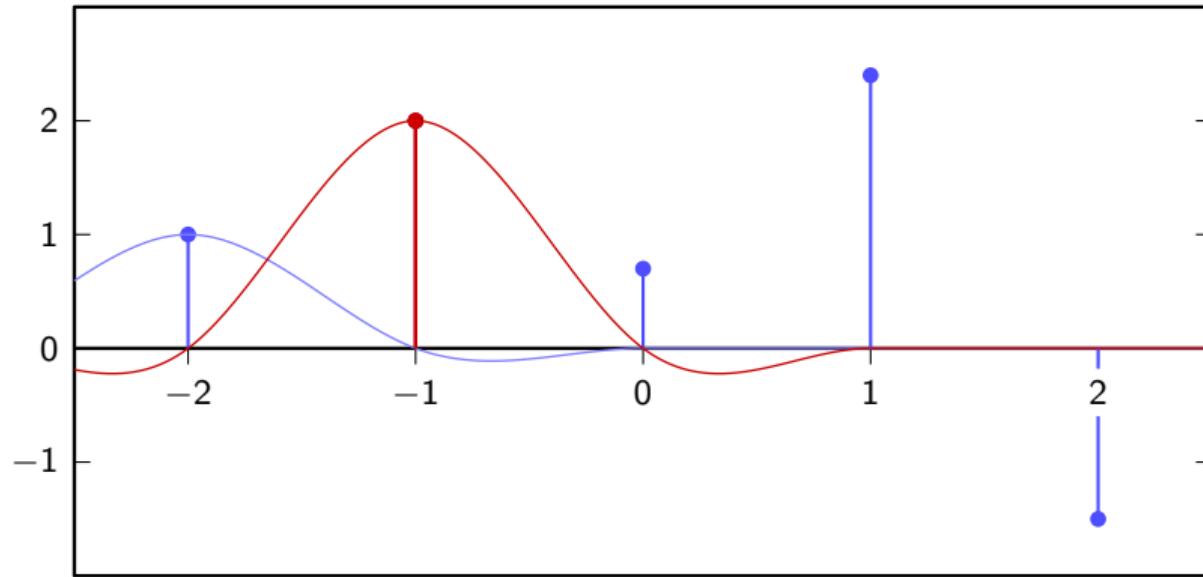
Third-order interpolation



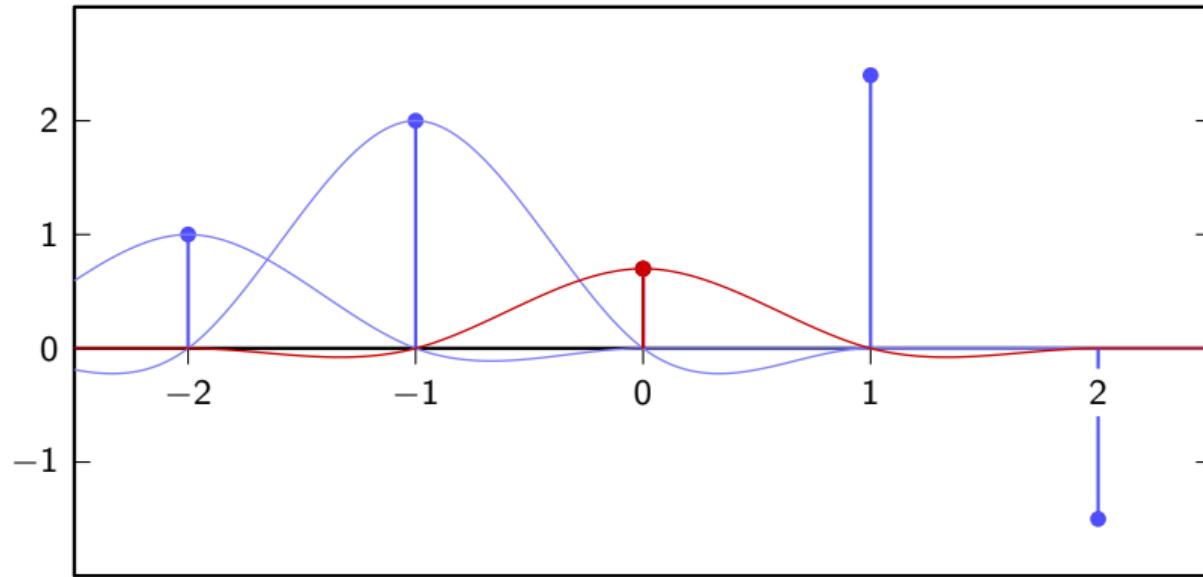
Third-order interpolation



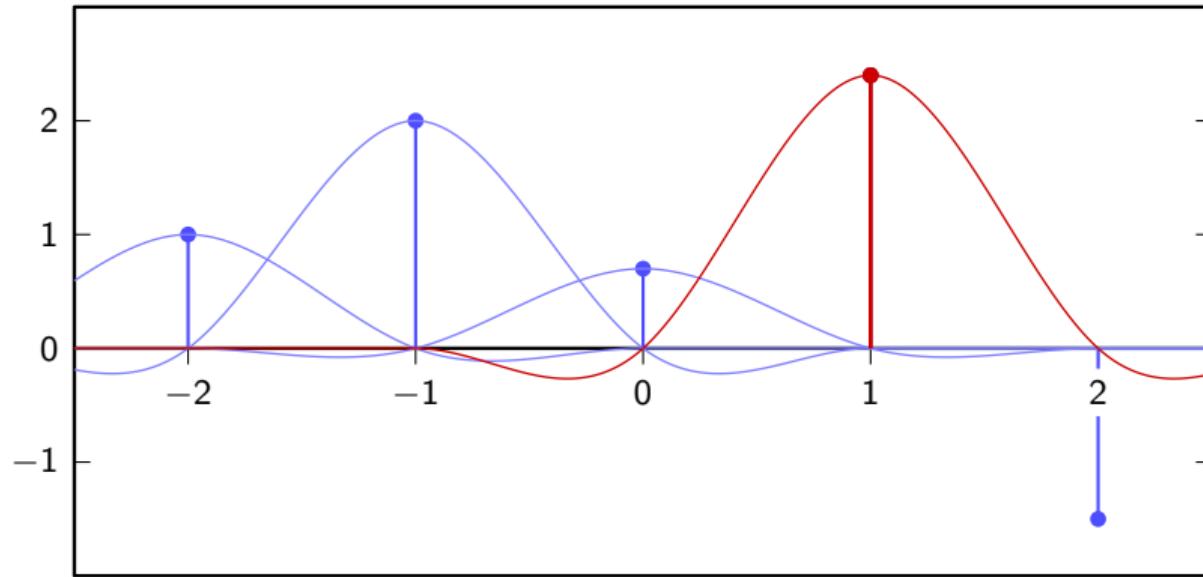
Third-order interpolation



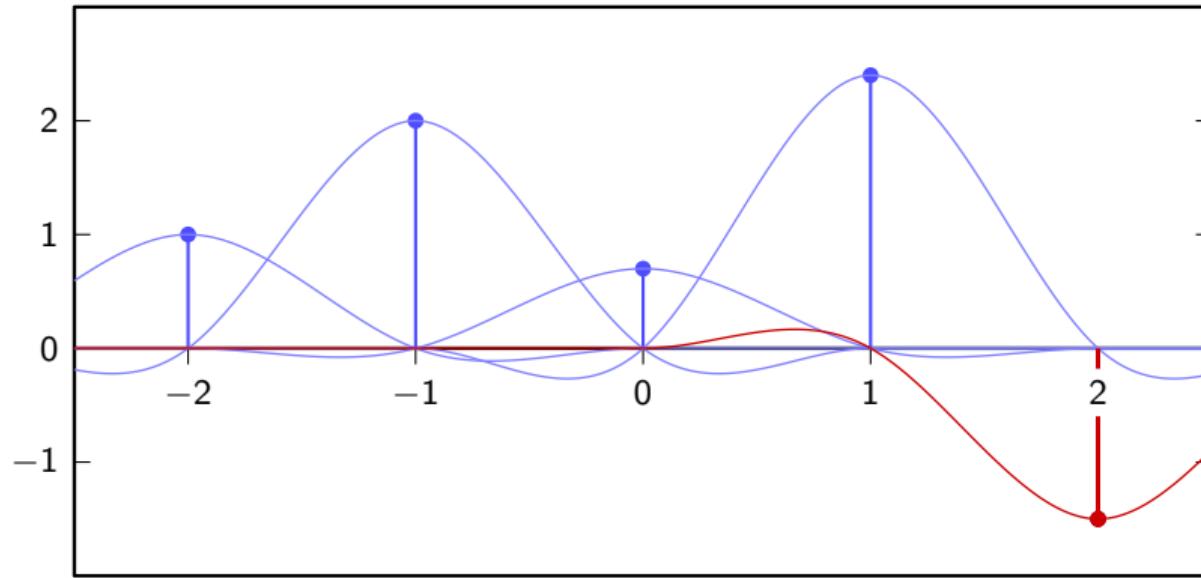
Third-order interpolation



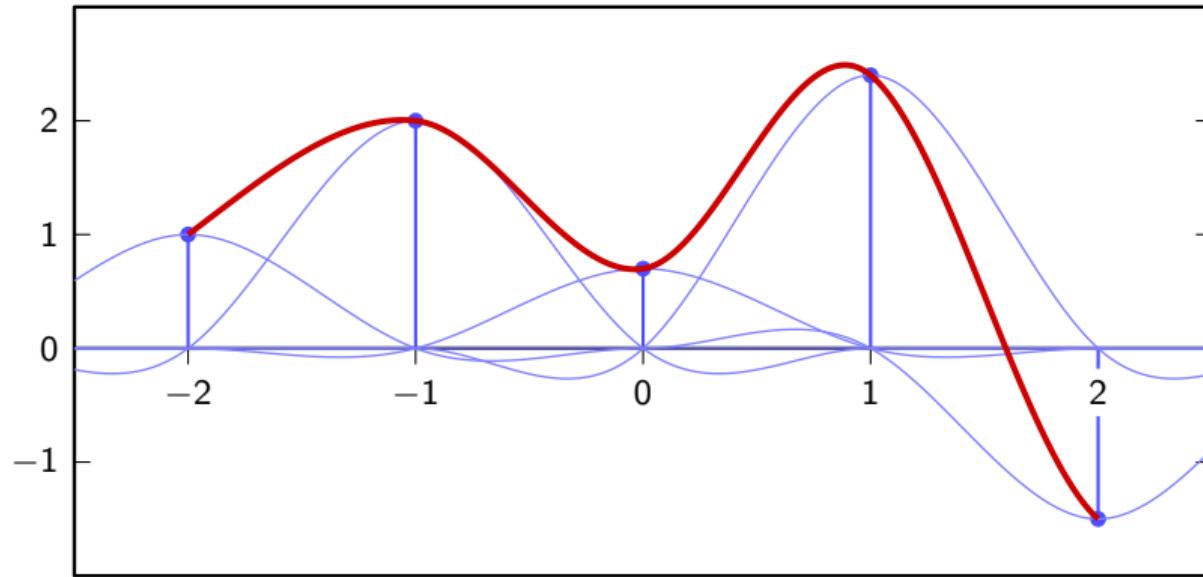
Third-order interpolation



Third-order interpolation



Third-order interpolation



Local interpolation schemes

$$x(t) = \sum_{n=-N}^N x[n] i_c(t - n)$$

Kernel must satisfy the interpolation property:

- ▶ $i_c(0) = 1$
- ▶ $i_c(m) = 0$ for m a nonzero integer.

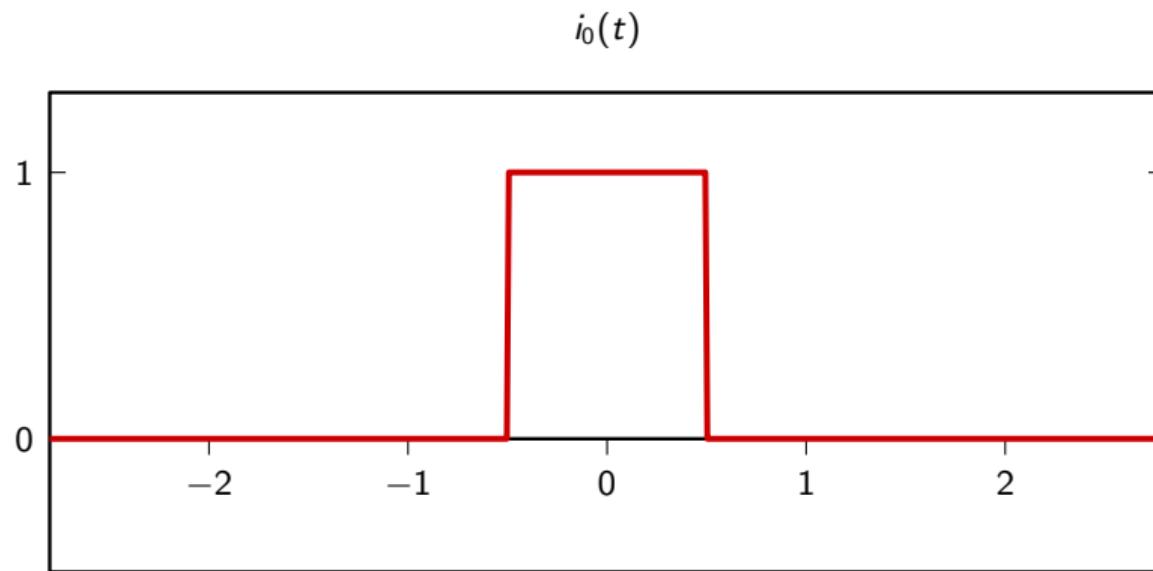
Local interpolation schemes

$$x(t) = \sum_{n=-N}^N x[n] i_c(t - n)$$

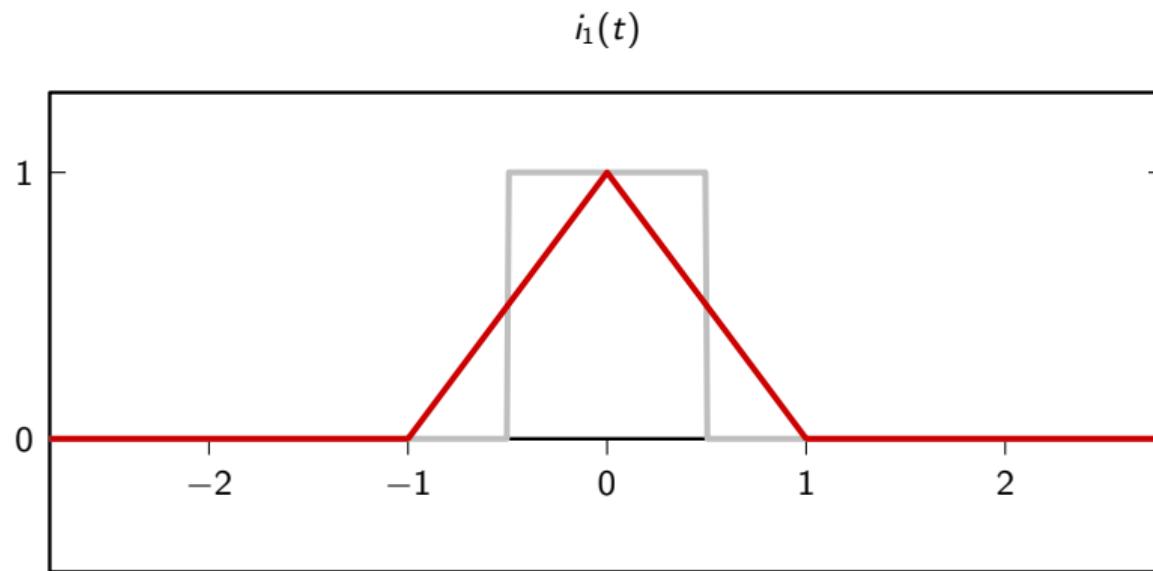
Kernel must satisfy the interpolation property:

- ▶ $i_c(0) = 1$
- ▶ $i_c(m) = 0$ for m a nonzero integer.

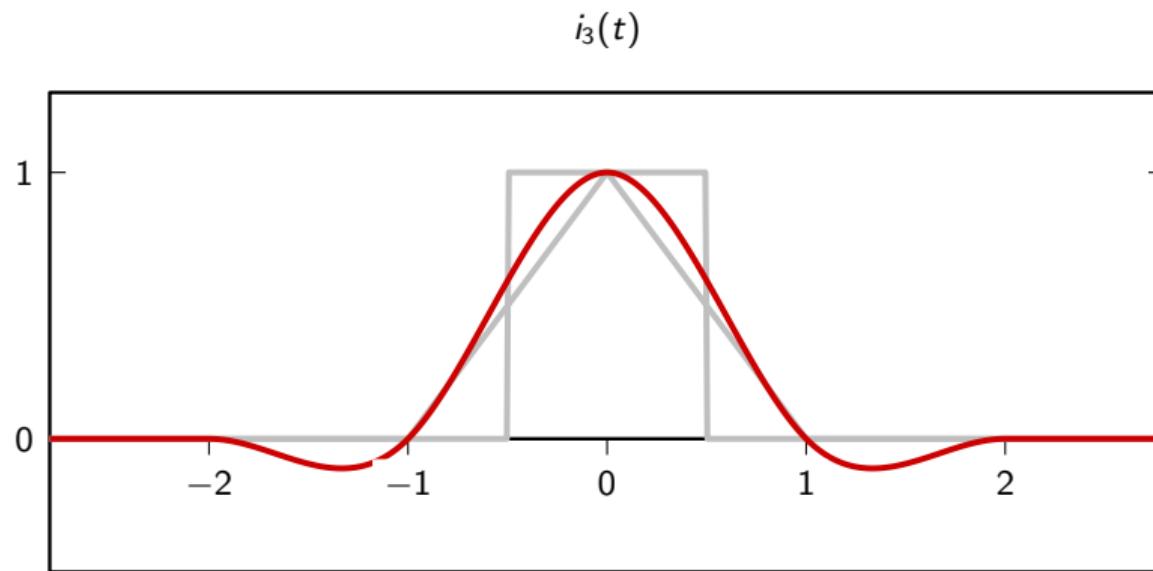
Local interpolators



Local interpolators



Local interpolators



Local interpolation

key property:

- ▶ same interpolating function independently of N

drawback:

- ▶ lack of smoothness

Polynomial interpolation

key property:

- ▶ maximally smooth (infinitely many continuous derivatives)

drawback:

- ▶ interpolation kernels depend on N

A remarkable result:

$$\lim_{N \rightarrow \infty} L_n^{(N)}(t) = f(t - n)$$

in the limit, local and global interpolation are the same!

A remarkable result:

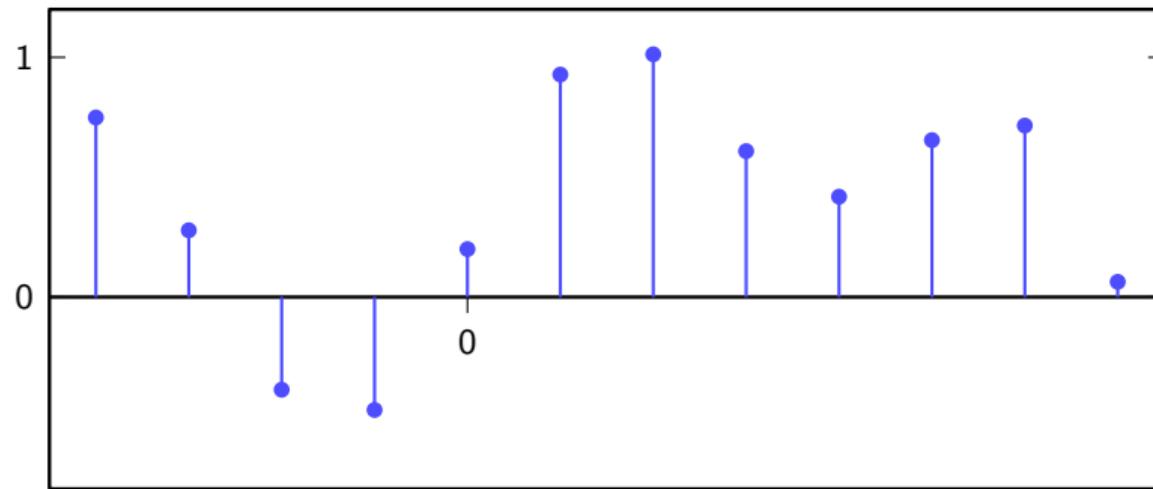
$$\lim_{N \rightarrow \infty} L_n^{(N)}(t) = \text{sinc}(t - n)$$

in the limit, local and global interpolation are the same!

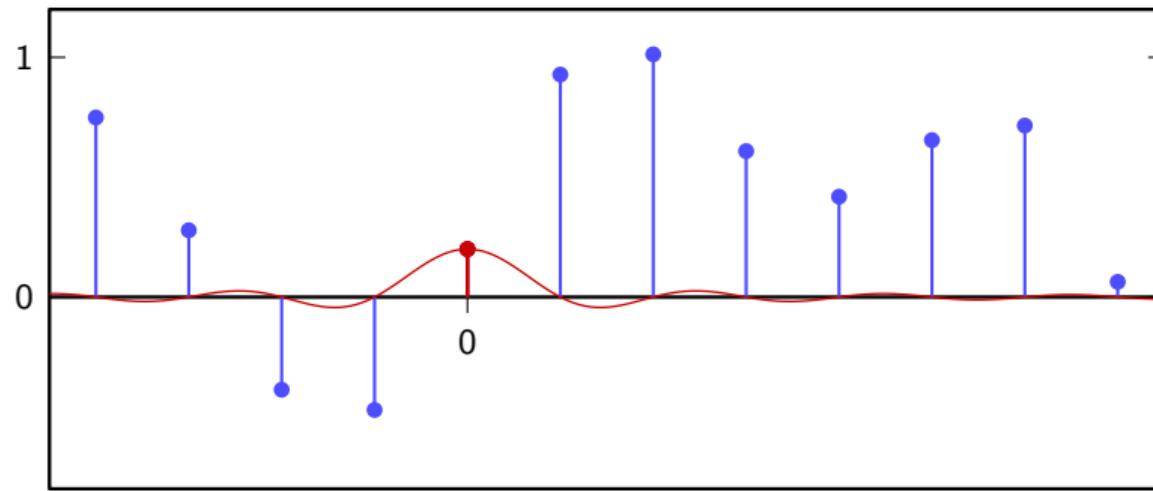
Sinc interpolation formula

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right)$$

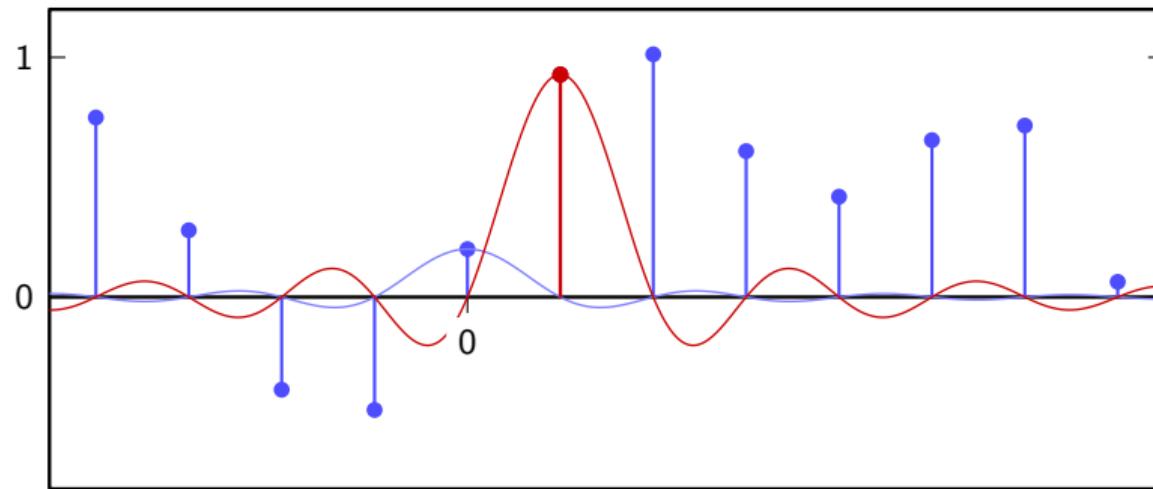
Sinc interpolation



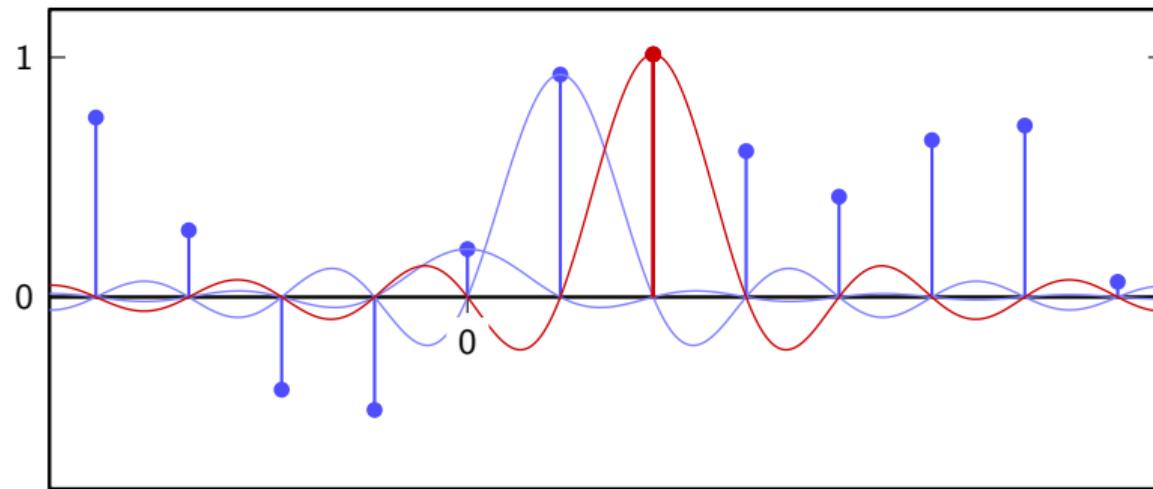
Sinc interpolation



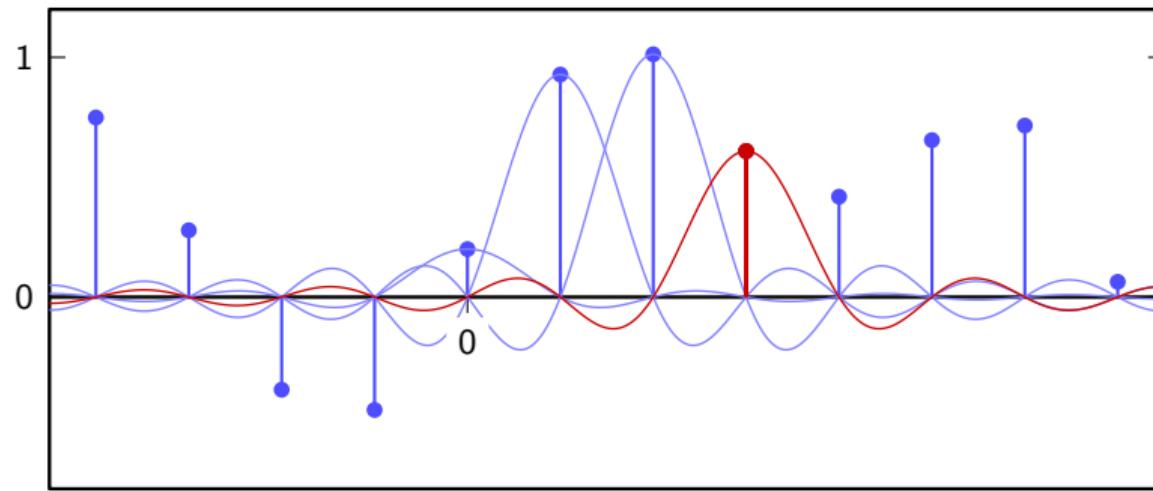
Sinc interpolation



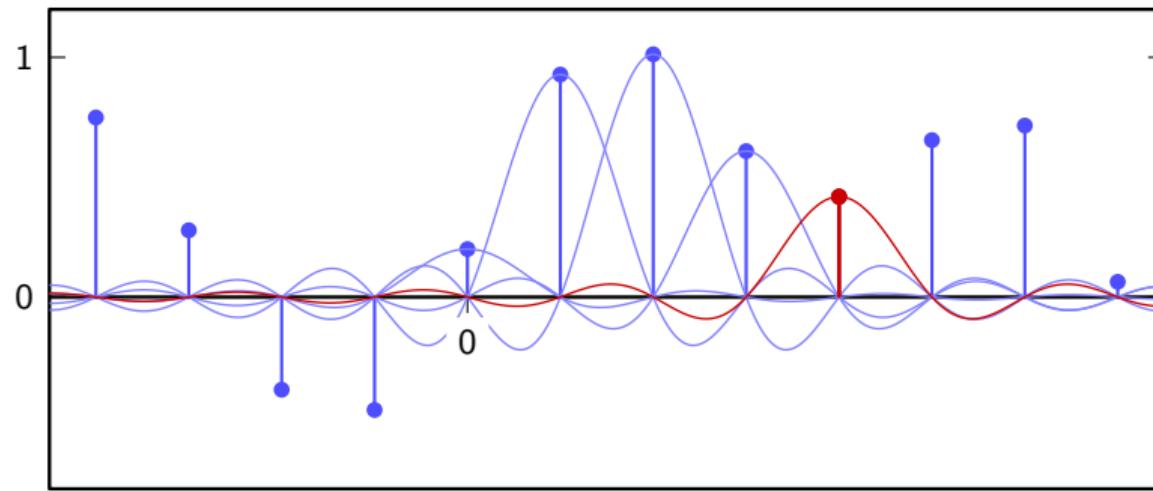
Sinc interpolation



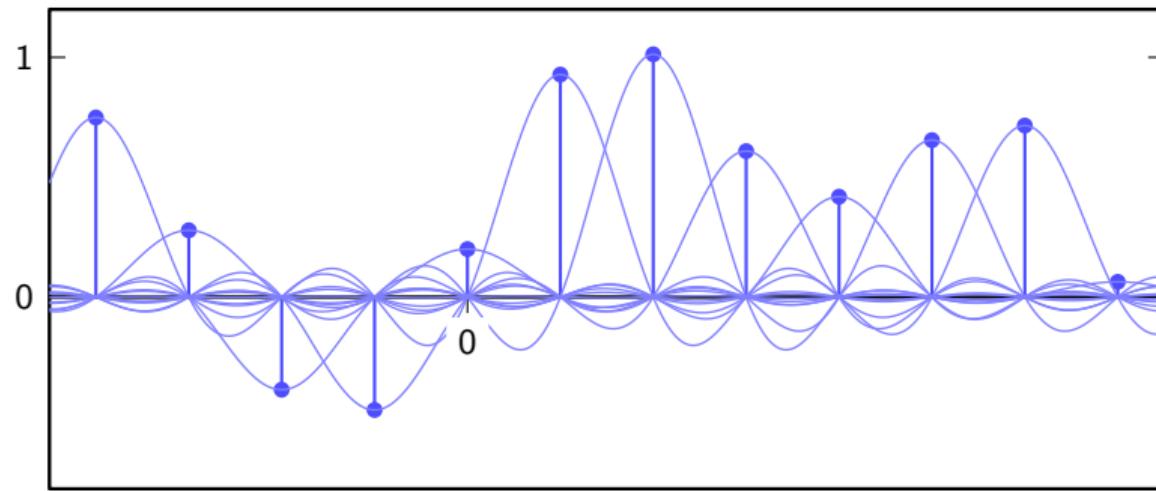
Sinc interpolation



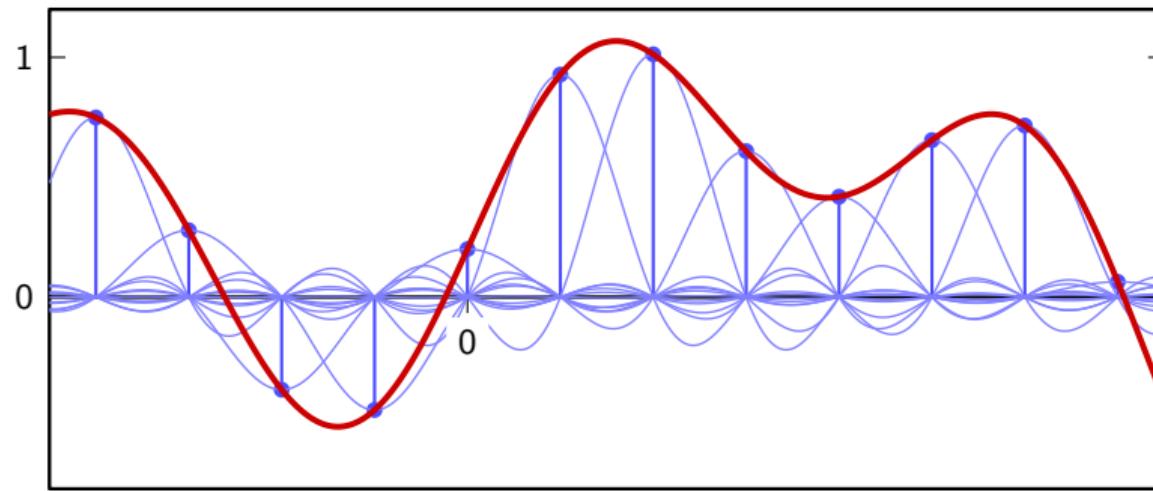
Sinc interpolation



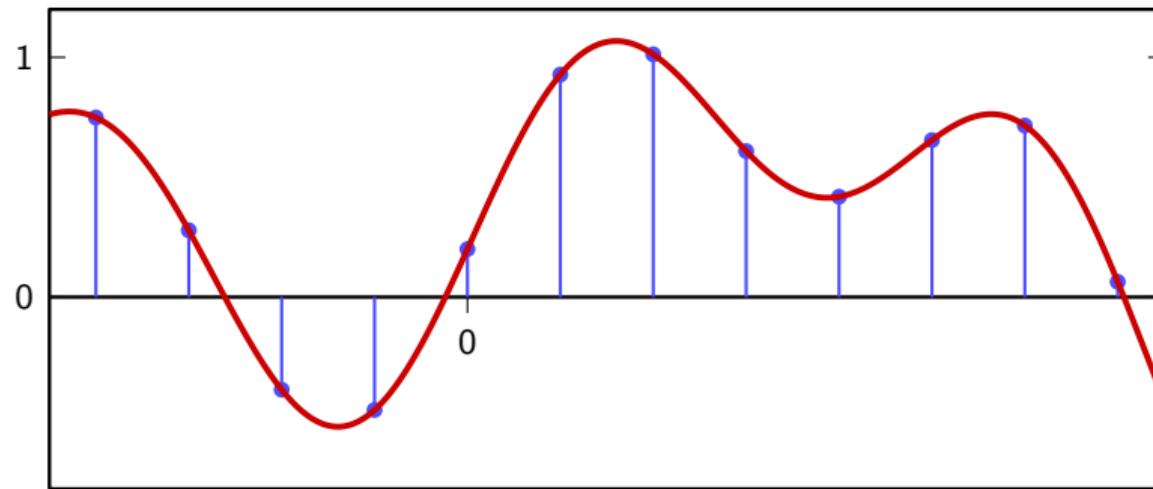
Sinc interpolation



Sinc interpolation



Sinc interpolation



Convergence: graphical “proof”

$$L_n^{(N)}(t) = \prod_{\substack{k=-N \\ k \neq n}}^N \frac{t-k}{n-k}$$

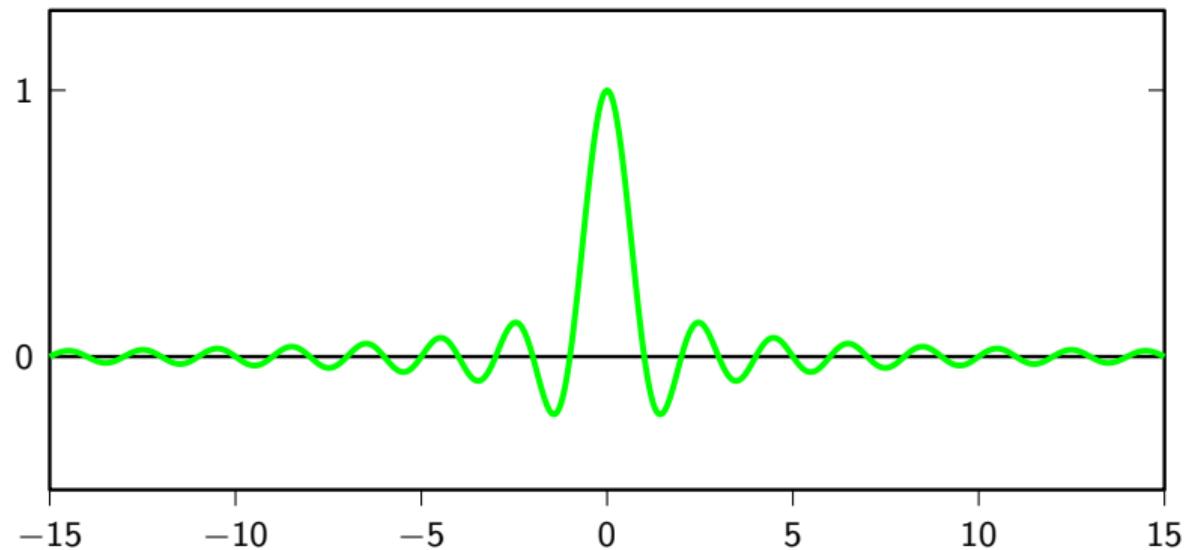
$$\begin{aligned} L_0^N(t) &= \prod_{\substack{k=-N \\ k \neq 0}}^N \frac{t-k}{-k} = \prod_{k=-N}^{-1} \frac{t-k}{-k} \prod_{k=1}^N \frac{t-k}{-k} \\ &= \prod_{k=1}^N \frac{t+k}{k} \prod_{k=1}^N \frac{t-k}{-k} \\ &= \prod_{k=1}^N \left(1 - \frac{t^2}{k^2}\right) \end{aligned}$$

Convergence: graphical “proof”

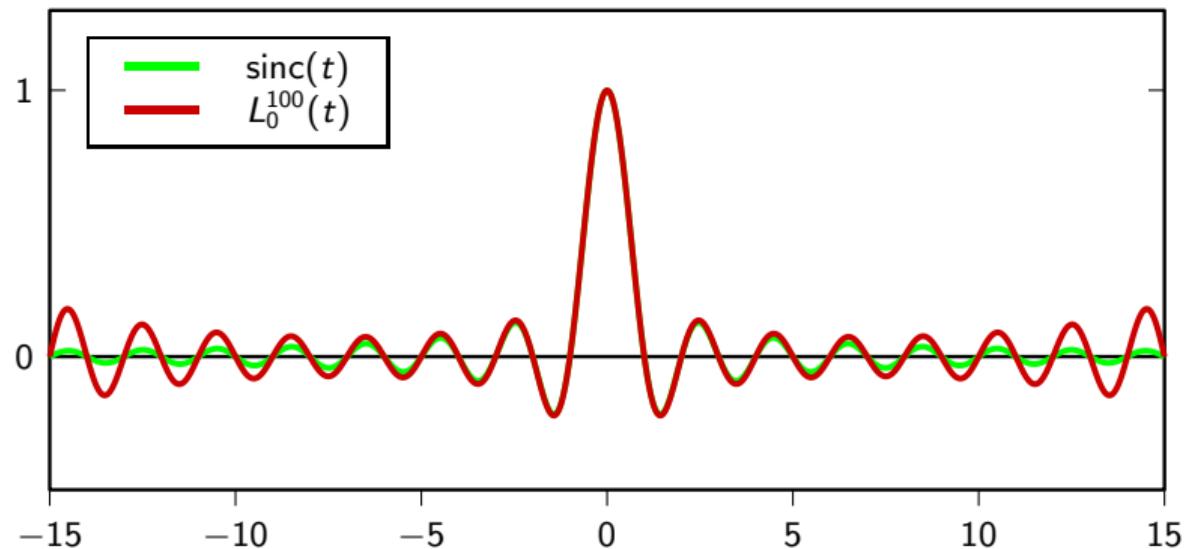
$$L_n^{(N)}(t) = \prod_{\substack{k=-N \\ k \neq n}}^N \frac{t-k}{n-k}$$

$$\begin{aligned} L_0^N(t) &= \prod_{\substack{k=-N \\ k \neq 0}}^N \frac{t-k}{-k} = \prod_{k=-N}^{-1} \frac{t-k}{-k} \prod_{k=1}^N \frac{t-k}{-k} \\ &= \prod_{k=1}^N \frac{t+k}{k} \prod_{k=1}^N \frac{t-k}{-k} \\ &= \prod_{k=1}^N \left(1 - \frac{t^2}{k^2}\right) \end{aligned}$$

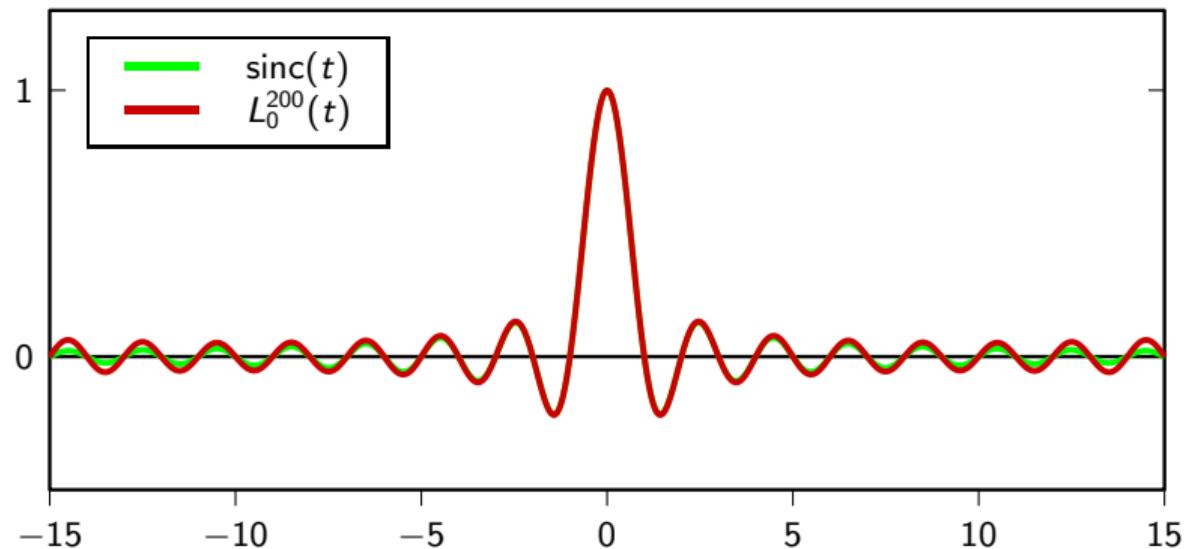
Convergence: graphical “proof”



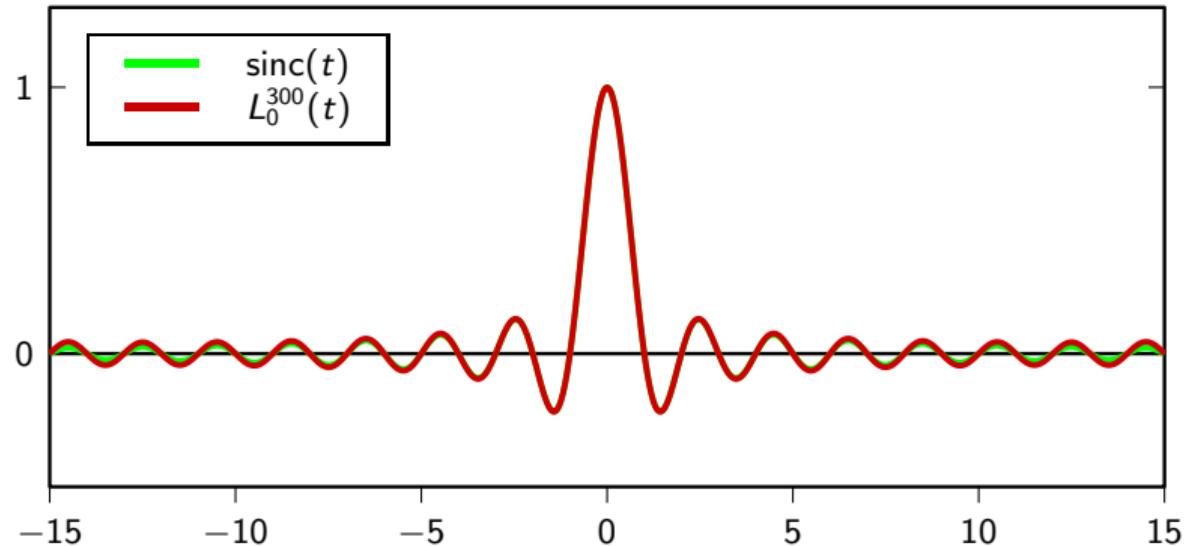
Convergence: graphical “proof”



Convergence: graphical “proof”



Convergence: graphical “proof”



Convergence: mathematical intuition

- ▶ $\text{sinc}(t - n)$ and $L_n^{(\infty)}(t)$ share an infinite number of zeros:

$$\text{sinc}(m - n) = \delta[m - n] \quad m, n \in \mathbb{Z}$$

$$L_n^{(N)}(m) = \delta[m - n] \quad m, n \in \mathbb{Z}, \quad -N \leq n, m \leq N$$

Convergence: Euler's “proof” (1748)

very cute (if non-rigorous) proof – see handout or book for details

Convergence: rigorous proof

uses the properties of Fourier series expansions – see handout or book for details

bandlimited functions and sampling

Overview:

- ▶ Spectrum of interpolated signals
- ▶ Space of bandlimited functions
- ▶ Sinc sampling
- ▶ The sampling theorem

Sinc interpolation

the ingredients:

- ▶ discrete-time signal $x[n]$, $n \in \mathbb{Z}$ (with DTFT $X(e^{j\omega})$)
- ▶ interpolation interval T_s
- ▶ the sinc function

the result:

- ▶ a smooth, continuous-time signal $x(t)$, $t \in \mathbb{R}$

what does the spectrum of $x(t)$ look like?

Sinc interpolation

the ingredients:

- ▶ discrete-time signal $x[n]$, $n \in \mathbb{Z}$ (with DTFT $X(e^{j\omega})$)
- ▶ interpolation interval T_s
- ▶ the sinc function

the result:

- ▶ a smooth, continuous-time signal $x(t)$, $t \in \mathbb{R}$

what does the spectrum of $x(t)$ look like?

Sinc interpolation

the ingredients:

- ▶ discrete-time signal $x[n]$, $n \in \mathbb{Z}$ (with DTFT $X(e^{j\omega})$)
- ▶ interpolation interval T_s
- ▶ the sinc function

the result:

- ▶ a smooth, continuous-time signal $x(t)$, $t \in \mathbb{R}$

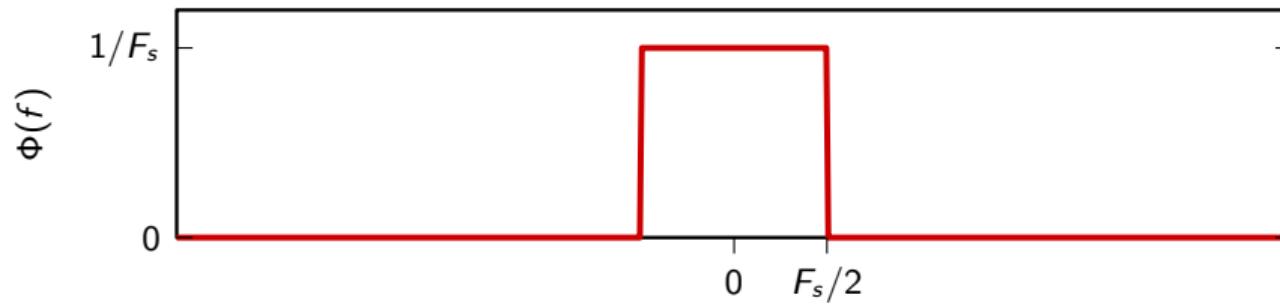
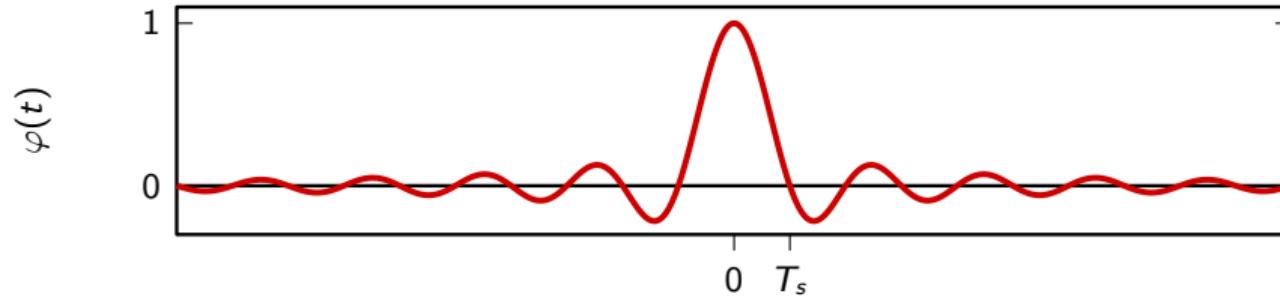
what does the spectrum of $x(t)$ look like?

Key facts about the sinc

$$\varphi(t) = \text{sinc}\left(\frac{t}{T_s}\right) \quad \longleftrightarrow \quad \Phi(f) = \frac{1}{F_s} \text{rect}\left(\frac{f}{F_s}\right)$$

$$T_s = \frac{1}{F_s}$$

Key facts about the sinc



Sinc interpolation

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right)$$

Spectral representation (I)

$$\begin{aligned} X(f) &= \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt \\ &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \\ &= \sum_{n=-\infty}^{\infty} x[n] \int_{-\infty}^{\infty} \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \\ &= \sum_{n=-\infty}^{\infty} x[n] \left(\frac{1}{F_s}\right) \operatorname{rect}\left(\frac{f}{F_s}\right) e^{-j2\pi f nT_s} \end{aligned}$$

Spectral representation (I)

$$\begin{aligned} X(f) &= \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt \\ &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \\ &= \sum_{n=-\infty}^{\infty} x[n] \int_{-\infty}^{\infty} \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \\ &= \sum_{n=-\infty}^{\infty} x[n] \left(\frac{1}{F_s}\right) \operatorname{rect}\left(\frac{f}{F_s}\right) e^{-j2\pi f nT_s} \end{aligned}$$

Spectral representation (I)

$$\begin{aligned} X(f) &= \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt \\ &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \\ &= \sum_{n=-\infty}^{\infty} x[n] \int_{-\infty}^{\infty} \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \\ &= \sum_{n=-\infty}^{\infty} x[n] \left(\frac{1}{F_s}\right) \operatorname{rect}\left(\frac{f}{F_s}\right) e^{-j2\pi f nT_s} \end{aligned}$$

Spectral representation (I)

$$\begin{aligned} X(f) &= \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt \\ &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \\ &= \sum_{n=-\infty}^{\infty} x[n] \int_{-\infty}^{\infty} \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \\ &= \sum_{n=-\infty}^{\infty} x[n] \left(\frac{1}{F_s}\right) \operatorname{rect}\left(\frac{f}{F_s}\right) e^{-j2\pi f n T_s} \end{aligned}$$

Spectral representation (II)

$$\begin{aligned} X(f) &= \sum_{n=-\infty}^{\infty} x[n] \left(\frac{1}{F_s} \right) \operatorname{rect} \left(\frac{f}{F_s} \right) e^{-j2\pi f n T_s} \\ &= T_s \operatorname{rect} \left(\frac{f}{F_s} \right) \sum_{n=-\infty}^{\infty} x[n] e^{-j2\pi(f/F_s)n} \\ &= \begin{cases} T_s X(e^{j2\pi f/F_s}) & \text{for } |f| \leq F_s/2 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Spectral representation (II)

$$\begin{aligned} X(f) &= \sum_{n=-\infty}^{\infty} x[n] \left(\frac{1}{F_s} \right) \operatorname{rect} \left(\frac{f}{F_s} \right) e^{-j2\pi f n T_s} \\ &= T_s \operatorname{rect} \left(\frac{f}{F_s} \right) \sum_{n=-\infty}^{\infty} x[n] e^{-j2\pi (f/F_s)n} \\ &= \begin{cases} T_s X(e^{j2\pi f/F_s}) & \text{for } |f| \leq F_s/2 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Spectral representation (II)

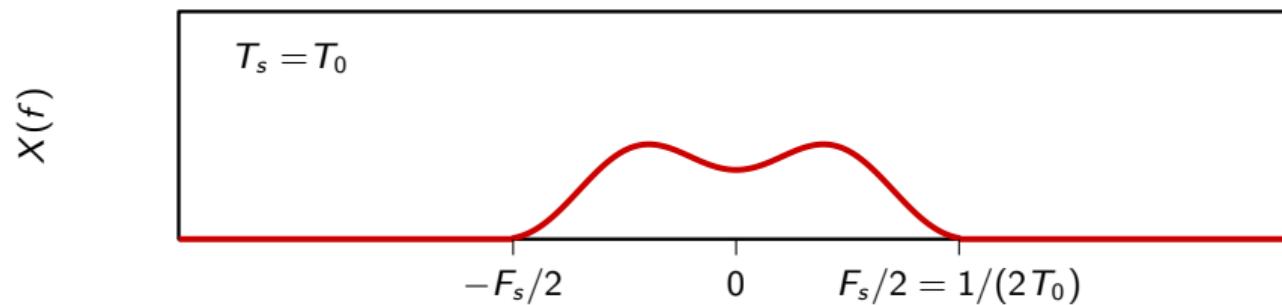
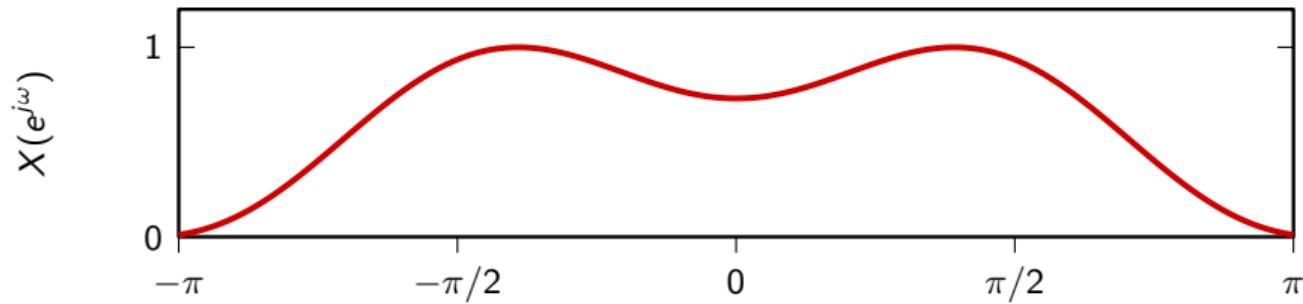
$$\begin{aligned} X(f) &= \sum_{n=-\infty}^{\infty} x[n] \left(\frac{1}{F_s} \right) \operatorname{rect} \left(\frac{f}{F_s} \right) e^{-j2\pi f n T_s} \\ &= T_s \operatorname{rect} \left(\frac{f}{F_s} \right) \sum_{n=-\infty}^{\infty} x[n] e^{-j2\pi (f/F_s)n} \\ &= \begin{cases} T_s X(e^{j2\pi f/F_s}) & \text{for } |f| \leq F_s/2 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Spectral representation (III)

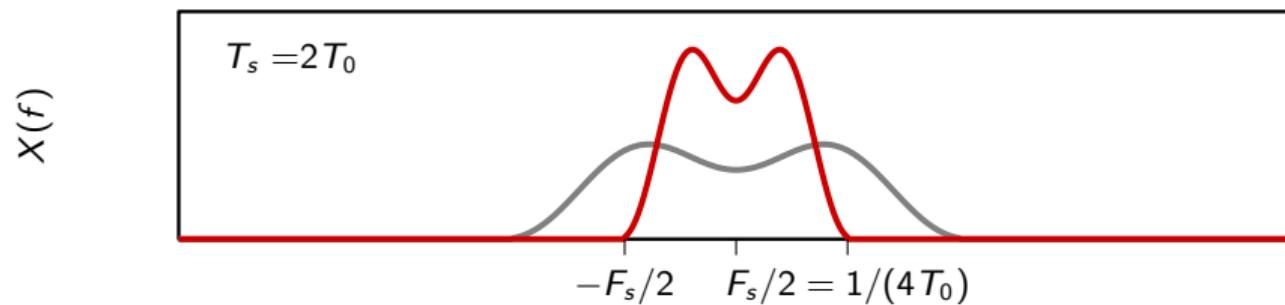
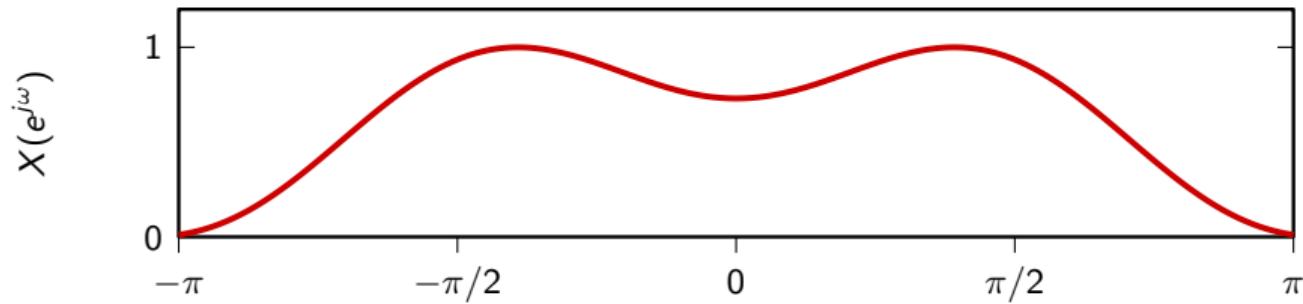
$$X(f) = \begin{cases} T_s X(e^{j2\pi f/F_s}) & \text{for } |f| \leq F_s/2 \\ 0 & \text{otherwise} \end{cases}$$

- ▶ map $\omega = \pi$ to $f = F_s/2$
- ▶ scale spectrum by T_s (total energy constant)
- ▶ rect keeps only the baseband copy of the periodic digital spectrum

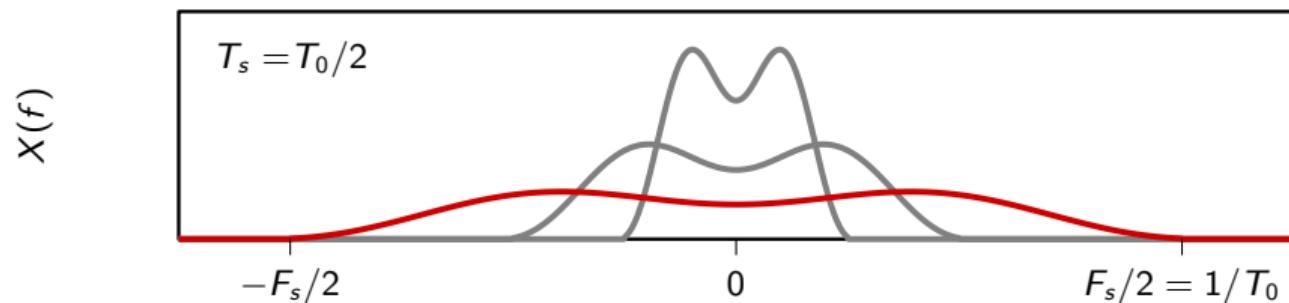
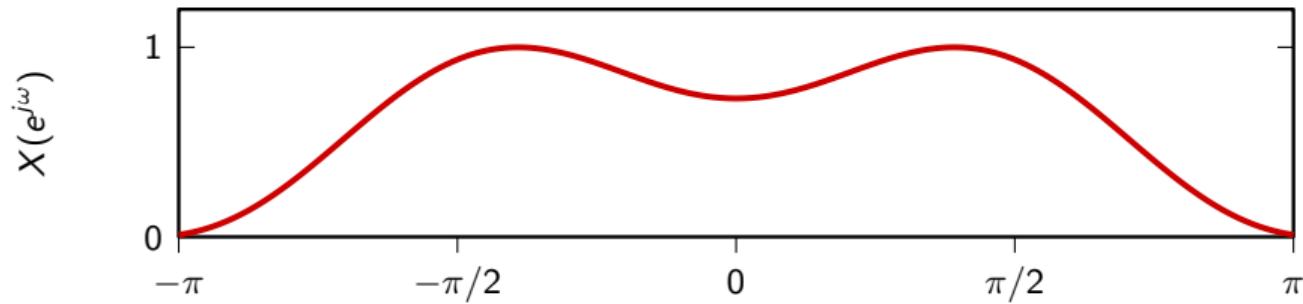
Spectrum of interpolated signals



Spectrum of interpolated signals



Spectrum of interpolated signals



Spectrum of interpolated signals

pick interpolation period T_s :

- ▶ $X(f)$ is F_s -bandlimited, with $F_s = 1/T_s$
- ▶ fast interpolation (T_s small) → wider spectrum
- ▶ slow interpolation (T_s large) → narrower spectrum
- ▶ (for those who remember...) it's like changing the speed of a record player

Spectrum of interpolated signals

pick interpolation period T_s :

- ▶ $X(f)$ is F_s -bandlimited, with $F_s = 1/T_s$
- ▶ fast interpolation (T_s small) → wider spectrum
- ▶ slow interpolation (T_s large) → narrower spectrum
- ▶ (for those who remember...) it's like changing the speed of a record player

Spectrum of interpolated signals

pick interpolation period T_s :

- ▶ $X(f)$ is F_s -bandlimited, with $F_s = 1/T_s$
- ▶ fast interpolation (T_s small) → wider spectrum
- ▶ slow interpolation (T_s large) → narrower spectrum
- ▶ (for those who remember...) it's like changing the speed of a record player

Spectrum of interpolated signals

pick interpolation period T_s :

- ▶ $X(f)$ is F_s -bandlimited, with $F_s = 1/T_s$
- ▶ fast interpolation (T_s small) → wider spectrum
- ▶ slow interpolation (T_s large) → narrower spectrum
- ▶ (for those who remember...) it's like changing the speed of a record player

Space of bandlimited functions

$$x[n] \in \ell_2(\mathbb{Z}) \xrightarrow{T_s} x(t) \in L_2(\mathbb{R})$$

$F_s\text{-BL}$

Space of bandlimited functions

$$x[n] \in \ell_2(\mathbb{Z}) \quad \xleftarrow[T_s]{?} \quad x(t) \in L_2(\mathbb{R})$$

$F_s\text{-BL}$

Let's lighten the notation

for a while we will proceed with $T_s = 1$ (so that $F_s = 1$ as well)
(derivations in the general case are in the book)

The road to the sampling theorem

claims:

- ▶ the space of 1-bandlimited functions is a Hilbert space
- ▶ the functions $\varphi^{(n)}(t) = \text{sinc}(t - n)$, with $n \in \mathbb{Z}$, form a basis for the space
- ▶ if $x(t)$ is 1-BL, the sequence $x[n] = x(n)$, with $n \in \mathbb{Z}$, is a sufficient representation
(i.e. we can reconstruct $x(t)$ from $x[n]$)

The road to the sampling theorem

claims:

- ▶ the space of 1-bandlimited functions is a Hilbert space
- ▶ the functions $\varphi^{(n)}(t) = \text{sinc}(t - n)$, with $n \in \mathbb{Z}$, form a basis for the space
- ▶ if $x(t)$ is 1-BL, the sequence $x[n] = x(n)$, with $n \in \mathbb{Z}$, is a sufficient representation
(i.e. we can reconstruct $x(t)$ from $x[n]$)

The road to the sampling theorem

claims:

- ▶ the space of 1-bandlimited functions is a Hilbert space
- ▶ the functions $\varphi^{(n)}(t) = \text{sinc}(t - n)$, with $n \in \mathbb{Z}$, form a basis for the space
- ▶ if $x(t)$ is 1-BL, the sequence $x[n] = x(n)$, with $n \in \mathbb{Z}$, is a sufficient representation
(i.e. we can reconstruct $x(t)$ from $x[n]$)

The space 1-BL

- ▶ clearly a vector space because $1\text{-BL} \subset L_2(\mathbb{R})$ (and linear combinations of 1-BL functions are 1-BL functions)
- ▶ inner product is standard inner product in $L_2(\mathbb{R})$
- ▶ completeness... that's more delicate

The space 1-BL

- ▶ clearly a vector space because $1\text{-BL} \subset L_2(\mathbb{R})$ (and linear combinations of 1-BL functions are 1-BL functions)
- ▶ inner product is standard inner product in $L_2(\mathbb{R})$
- ▶ completeness... that's more delicate

The space 1-BL

- ▶ clearly a vector space because $1\text{-BL} \subset L_2(\mathbb{R})$ (and linear combinations of 1-BL functions are 1-BL functions)
- ▶ inner product is standard inner product in $L_2(\mathbb{R})$
- ▶ completeness... that's more delicate

The space of 1-BL functions

recap:

- ▶ inner product:

$$\langle x(t), y(t) \rangle = \int_{-\infty}^{\infty} x^*(t)y(t)dt$$

- ▶ convolution:

$$(x * y)(t) = \langle x^*(\tau), y(t - \tau) \rangle$$

A basis for the 1-BL space

$$\varphi^{(n)}(t) = \text{sinc}(t - n), \quad n \in \mathbb{Z}$$

$$\begin{aligned}\langle \varphi^{(n)}(t), \varphi^{(m)}(t) \rangle &= \langle \varphi^{(0)}(t - n), \varphi^{(0)}(t - m) \rangle \\&= \langle \varphi^{(0)}(t - n), \varphi^{(0)}(m - t) \rangle \\&= \int_{-\infty}^{\infty} \text{sinc}(t - n) \text{sinc}(m - t) dt \\&= \int_{-\infty}^{\infty} \text{sinc}(\tau) \text{sinc}((m - n) - \tau) d\tau \\&= (\text{sinc} * \text{sinc})(m - n)\end{aligned}$$

A basis for the 1-BL space

$$\varphi^{(n)}(t) = \text{sinc}(t - n), \quad n \in \mathbb{Z}$$

$$\begin{aligned}\langle \varphi^{(n)}(t), \varphi^{(m)}(t) \rangle &= \langle \varphi^{(0)}(t - n), \varphi^{(0)}(t - m) \rangle \\&= \langle \varphi^{(0)}(t - n), \varphi^{(0)}(m - t) \rangle \\&= \int_{-\infty}^{\infty} \text{sinc}(t - n) \text{sinc}(m - t) dt \\&= \int_{-\infty}^{\infty} \text{sinc}(\tau) \text{sinc}((m - n) - \tau) d\tau \\&= (\text{sinc} * \text{sinc})(m - n)\end{aligned}$$

A basis for the 1-BL space

$$\varphi^{(n)}(t) = \text{sinc}(t - n), \quad n \in \mathbb{Z}$$

$$\begin{aligned}\langle \varphi^{(n)}(t), \varphi^{(m)}(t) \rangle &= \langle \varphi^{(0)}(t - n), \varphi^{(0)}(t - m) \rangle \\&= \langle \varphi^{(0)}(t - n), \varphi^{(0)}(m - t) \rangle \\&= \int_{-\infty}^{\infty} \text{sinc}(t - n) \text{sinc}(m - t) dt \\&= \int_{-\infty}^{\infty} \text{sinc}(\tau) \text{sinc}((m - n) - \tau) d\tau \\&= (\text{sinc} * \text{sinc})(m - n)\end{aligned}$$

A basis for the 1-BL space

$$\varphi^{(n)}(t) = \text{sinc}(t - n), \quad n \in \mathbb{Z}$$

$$\begin{aligned}\langle \varphi^{(n)}(t), \varphi^{(m)}(t) \rangle &= \langle \varphi^{(0)}(t - n), \varphi^{(0)}(t - m) \rangle \\&= \langle \varphi^{(0)}(t - n), \varphi^{(0)}(m - t) \rangle \\&= \int_{-\infty}^{\infty} \text{sinc}(t - n) \text{sinc}(m - t) dt \\&= \int_{-\infty}^{\infty} \text{sinc}(\tau) \text{sinc}((m - n) - \tau) d\tau \\&= (\text{sinc} * \text{sinc})(m - n)\end{aligned}$$

A basis for the 1-BL space

$$\varphi^{(n)}(t) = \text{sinc}(t - n), \quad n \in \mathbb{Z}$$

$$\begin{aligned}\langle \varphi^{(n)}(t), \varphi^{(m)}(t) \rangle &= \langle \varphi^{(0)}(t - n), \varphi^{(0)}(t - m) \rangle \\&= \langle \varphi^{(0)}(t - n), \varphi^{(0)}(m - t) \rangle \\&= \int_{-\infty}^{\infty} \text{sinc}(t - n) \text{sinc}(m - t) dt \\&= \int_{-\infty}^{\infty} \text{sinc}(\tau) \text{sinc}((m - n) - \tau) d\tau \\&= (\text{sinc} * \text{sinc})(m - n)\end{aligned}$$

A basis for the 1-BL space

now use the convolution theorem knowing that:

$$\text{FT} \{ \text{sinc}(t) \} = \text{rect}(f)$$

$$\begin{aligned} (\text{sinc} * \text{sinc})(m - n) &= \int_{-\infty}^{\infty} \text{rect}^2(f) e^{j2\pi f(m-n)} df \\ &= \int_{-1/2}^{1/2} e^{j2\pi f(m-n)} df \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\Omega(m-n)} d\Omega \\ &= \begin{cases} 1 & \text{for } m = n \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

A basis for the 1-BL space

now use the convolution theorem knowing that:

$$\text{FT} \{ \text{sinc}(t) \} = \text{rect}(f)$$

$$\begin{aligned} (\text{sinc} * \text{sinc})(m - n) &= \int_{-\infty}^{\infty} \text{rect}^2(f) e^{j2\pi f(m-n)} df \\ &= \int_{-1/2}^{1/2} e^{j2\pi f(m-n)} df \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\Omega(m-n)} d\Omega \\ &= \begin{cases} 1 & \text{for } m = n \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

A basis for the 1-BL space

now use the convolution theorem knowing that:

$$\text{FT} \{ \text{sinc}(t) \} = \text{rect}(f)$$

$$\begin{aligned} (\text{sinc} * \text{sinc})(m - n) &= \int_{-\infty}^{\infty} \text{rect}^2(f) e^{j2\pi f(m-n)} df \\ &= \int_{-1/2}^{1/2} e^{j2\pi f(m-n)} df \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\Omega(m-n)} d\Omega \\ &= \begin{cases} 1 & \text{for } m = n \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

A basis for the 1-BL space

now use the convolution theorem knowing that:

$$\text{FT} \{ \text{sinc}(t) \} = \text{rect}(f)$$

$$\begin{aligned} (\text{sinc} * \text{sinc})(m - n) &= \int_{-\infty}^{\infty} \text{rect}^2(f) e^{j2\pi f(m-n)} df \\ &= \int_{-1/2}^{1/2} e^{j2\pi f(m-n)} df \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\Omega(m-n)} d\Omega \\ &= \begin{cases} 1 & \text{for } m = n \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Space of bandlimited functions

$$x[n] \in \ell_2(\mathbb{Z}) \quad \longleftrightarrow \quad x(t) \in L_2(\mathbb{R})$$

1-BL

Sampling as a basis expansion

for any $x(t) \in 1\text{-BL}$:

$$\begin{aligned}\langle \varphi^{(n)}(t), x(t) \rangle &= \langle \text{sinc}(t - n), x(t) \rangle = \langle \text{sinc}(n - t), x(t) \rangle \\&= (\text{sinc} * x)(n) \\&= \int_{-\infty}^{\infty} \text{rect}(f) X(f) e^{j2\pi f n} df \\&= \int_{-\infty}^{\infty} X(f) e^{j2\pi f n} df \\&= x(n)\end{aligned}$$

Sampling as a basis expansion

for any $x(t) \in 1\text{-BL}$:

$$\begin{aligned}\langle \varphi^{(n)}(t), x(t) \rangle &= \langle \text{sinc}(t - n), x(t) \rangle = \langle \text{sinc}(n - t), x(t) \rangle \\ &= (\text{sinc} * x)(n) \\ &= \int_{-\infty}^{\infty} \text{rect}(f) X(f) e^{j2\pi f n} df \\ &= \int_{-\infty}^{\infty} X(f) e^{j2\pi f n} df \\ &= x(n)\end{aligned}$$

Sampling as a basis expansion

for any $x(t) \in 1\text{-BL}$:

$$\begin{aligned}\langle \varphi^{(n)}(t), x(t) \rangle &= \langle \text{sinc}(t - n), x(t) \rangle = \langle \text{sinc}(n - t), x(t) \rangle \\&= (\text{sinc} * x)(n) \\&= \int_{-\infty}^{\infty} \text{rect}(f) X(f) e^{j2\pi f n} df \\&= \int_{-\infty}^{\infty} X(f) e^{j2\pi f n} df \\&= x(n)\end{aligned}$$

Sampling as a basis expansion

for any $x(t) \in 1\text{-BL}$:

$$\begin{aligned}\langle \varphi^{(n)}(t), x(t) \rangle &= \langle \text{sinc}(t - n), x(t) \rangle = \langle \text{sinc}(n - t), x(t) \rangle \\&= (\text{sinc} * x)(n) \\&= \int_{-\infty}^{\infty} \text{rect}(f) X(f) e^{j2\pi f n} df \\&= \int_{-\infty}^{\infty} X(f) e^{j2\pi f n} df \\&= x(n)\end{aligned}$$

Sampling as a basis expansion

for any $x(t) \in 1\text{-BL}$:

$$\begin{aligned}\langle \varphi^{(n)}(t), x(t) \rangle &= \langle \text{sinc}(t - n), x(t) \rangle = \langle \text{sinc}(n - t), x(t) \rangle \\&= (\text{sinc} * x)(n) \\&= \int_{-\infty}^{\infty} \text{rect}(f) X(f) e^{j2\pi f n} df \\&= \int_{-\infty}^{\infty} X(f) e^{j2\pi f n} df \\&= x(n)\end{aligned}$$

Sampling as a basis expansion, 1-BL

Analysis formula:

$$x[n] = \langle \text{sinc}(t - n), x(t) \rangle$$

Synthesis formula:

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] \text{sinc}(t - n)$$

Sampling as a basis expansion, F_s -BL

Analysis formula:

$$x[n] = \langle \text{sinc}\left(\frac{t - nT_s}{T_s}\right), x(t) \rangle = T_s x(nT_s)$$

Synthesis formula:

$$x(t) = \frac{1}{T_s} \sum_{n=-\infty}^{\infty} x[n] \text{sinc}\left(\frac{t - nT_s}{T_s}\right)$$

The sampling theorem

- ▶ the space of F_s -bandlimited functions is a Hilbert space
- ▶ set $T_s = 1/F_s$
- ▶ the functions $\varphi^{(n)}(t) = \text{sinc}\left(\frac{t - nT_s}{T_s}\right)$ form a basis for the space
- ▶ for any $x(t) \in F_s\text{-BL}$ the coefficients in the sinc basis are the (scaled) samples $T_s x(nT_s)$

for any $x(t) \in F_s\text{-BL}$, a sufficient representation is the sequence $x[n] = x(nT_s)$

The sampling theorem

- ▶ the space of F_s -bandlimited functions is a Hilbert space
- ▶ set $T_s = 1/F_s$
- ▶ the functions $\varphi^{(n)}(t) = \text{sinc}\left(\frac{t - nT_s}{T_s}\right)$ form a basis for the space
- ▶ for any $x(t) \in F_s\text{-BL}$ the coefficients in the sinc basis are the (scaled) samples $T_s x(nT_s)$

for any $x(t) \in F_s\text{-BL}$, a sufficient representation is the sequence $x[n] = x(nT_s)$

The sampling theorem, corollary

- ▶ $F_s\text{-BL} \subseteq F\text{-BL}$ for any $F \geq F_s$

for any $x(t) \in F_s\text{-BL}$, a sufficient representation is the sequence

$$x[n] = x(nT_s) \text{ for any } T_s \leq 1/F_s$$

The sampling theorem, corollary

- ▶ F_s -BL $\subseteq F$ -BL for any $F \geq F_s$

for any $x(t) \in F_s$ -BL, a sufficient representation is the sequence
 $x[n] = x(nT_s)$ for any $T_s \leq 1/F_s$

The sampling theorem, again

any signal $x(t)$ whose highest frequency component is F_N Hz
can be sampled with no loss of information
using a sampling frequency $F_s \geq 2F_N$ (i.e. a sampling period $T_s \leq 1/(2F_N)$)

F_N is called the Nyquist frequency of the signal.

COM303: Digital Signal Processing

Lecture 17: Sampling and applications

overview

- ▶ raw sampling and aliasing
- ▶ DT processing of CT signals

Sinc Sampling

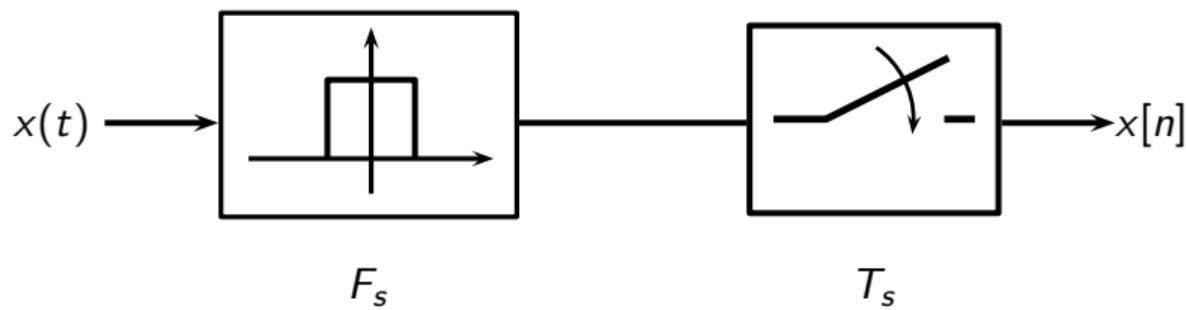
$$x[n] = \langle \text{sinc} \left(\frac{t - nT_s}{T_s} \right), x(t) \rangle$$

Sinc Sampling

$$x[n] = (\text{sinc}_{T_s} * x)(nT_s)$$

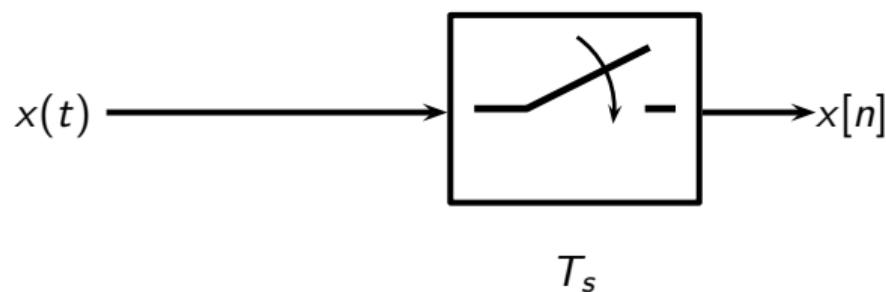
Sinc Sampling

$$x[n] = (\text{sinc}_{T_s} * x)(nT_s)$$



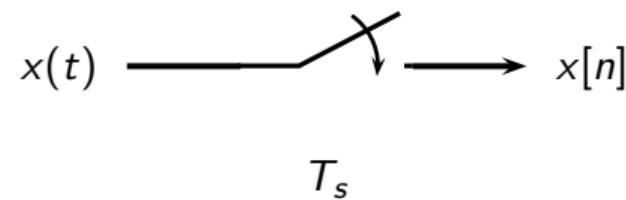
Sinc Sampling for F_s -BL signals

$$x[n] = (\text{sinc}_{T_s} * x)(nT_s) = T_s x(nT_s)$$



“Raw” sampling - can we always do that?

$$x[n] = x(nT_s)$$



Remember the wagonwheel effect?

The continuous-time complex exponential

$$x(t) = e^{j2\pi f_0 t}$$

- ▶ always periodic, period $t_0 = 1/f_0$
- ▶ all angular speeds are allowed
- ▶ FT $\{e^{j2\pi f_0 t}\} = \delta(f - f_0)$
- ▶ highest (and only) frequency is f_0

The continuous-time complex exponential

$$x(t) = e^{j2\pi f_0 t}$$

- ▶ always periodic, period $t_0 = 1/f_0$
- ▶ all angular speeds are allowed
- ▶ $\text{FT} \{ e^{j2\pi f_0 t} \} = \delta(f - f_0)$
- ▶ highest (and only) frequency is f_0

The continuous-time complex exponential

$$x(t) = e^{j2\pi f_0 t}$$

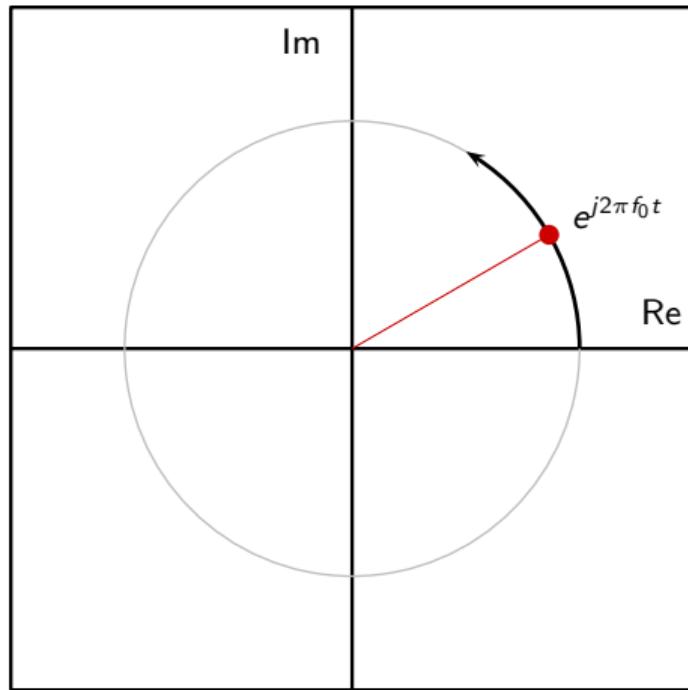
- ▶ always periodic, period $t_0 = 1/f_0$
- ▶ all angular speeds are allowed
- ▶ FT $\{e^{j2\pi f_0 t}\} = \delta(f - f_0)$
- ▶ highest (and only) frequency is f_0

The continuous-time complex exponential

$$x(t) = e^{j2\pi f_0 t}$$

- ▶ always periodic, period $t_0 = 1/f_0$
- ▶ all angular speeds are allowed
- ▶ FT $\{e^{j2\pi f_0 t}\} = \delta(f - f_0)$
- ▶ highest (and only) frequency is f_0

The continuous-time complex exponential



Raw samples of the continuous-time complex exponential

$$x[n] = e^{j2\pi f_0 n T_s}$$

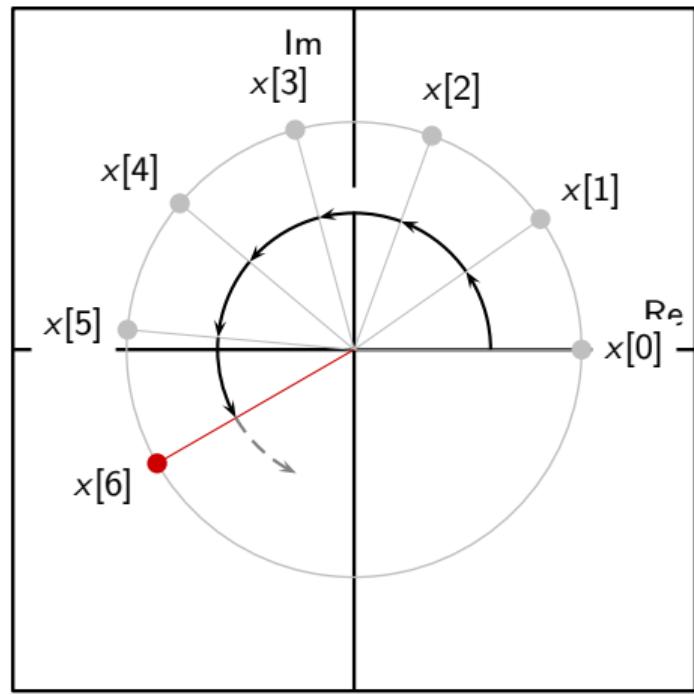
- ▶ raw samples are snapshots at regular intervals of the rotating point
- ▶ resulting digital frequency is $\omega_0 = 2\pi f_0 T_s = 2\pi(f_0/F_s)$

Raw samples of the continuous-time complex exponential

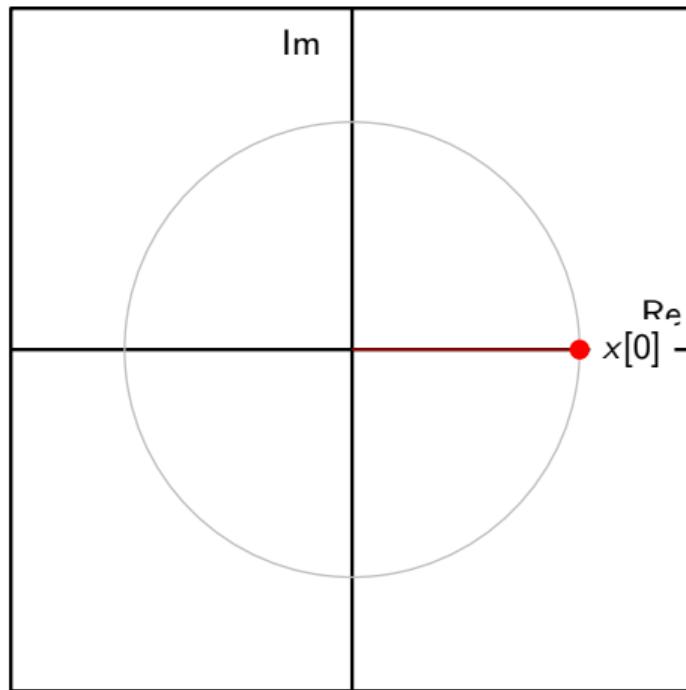
$$x[n] = e^{j2\pi f_0 n T_s}$$

- ▶ raw samples are snapshots at regular intervals of the rotating point
- ▶ resulting digital frequency is $\omega_0 = 2\pi f_0 T_s = 2\pi(f_0/F_s)$

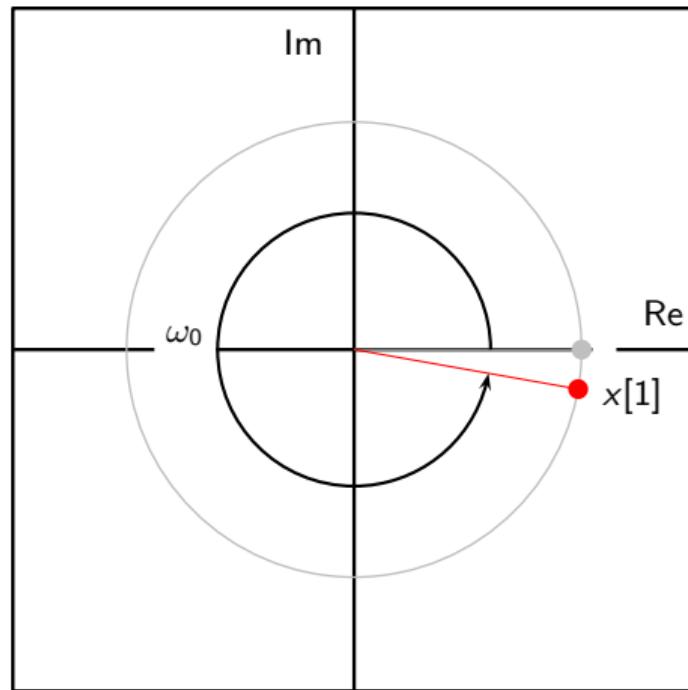
Easy: $f_0 < F_s/2 \Rightarrow \omega_0 < \pi$



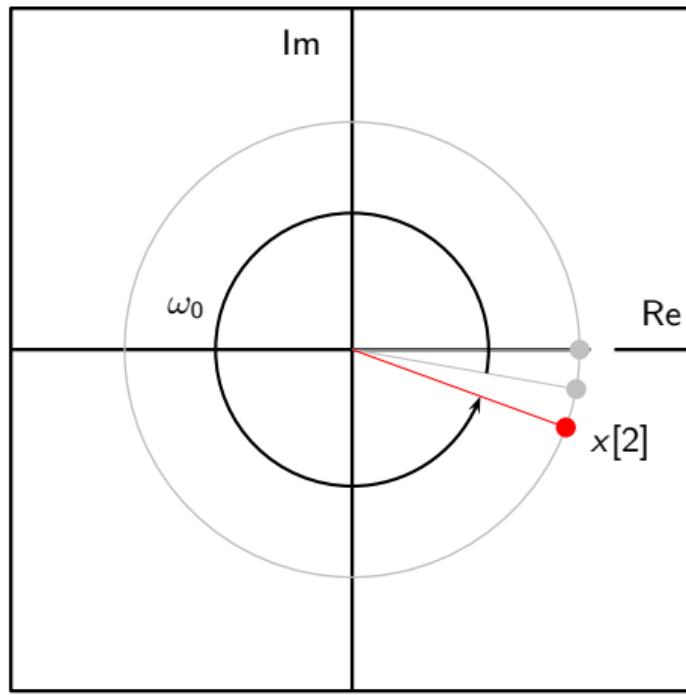
Tricky: $F_s/2 < f_0 < F_s \Rightarrow \pi < \omega_0 < 2\pi$



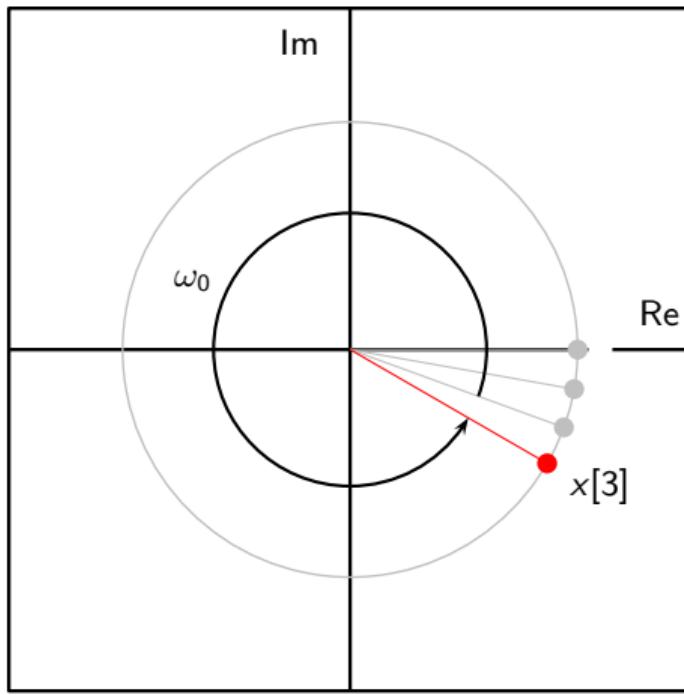
Tricky: $F_s/2 < f_0 < F_s \Rightarrow \pi < \omega_0 < 2\pi$



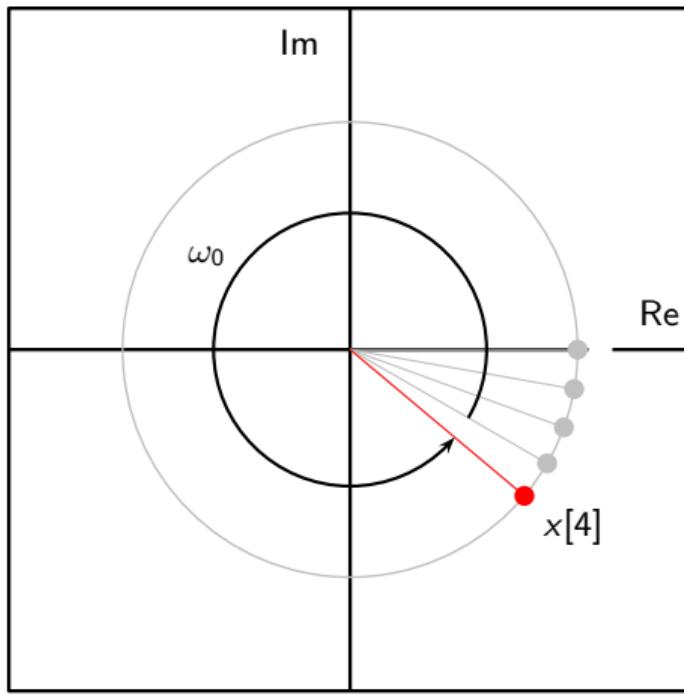
Tricky: $F_s/2 < f_0 < F_s \Rightarrow \pi < \omega_0 < 2\pi$



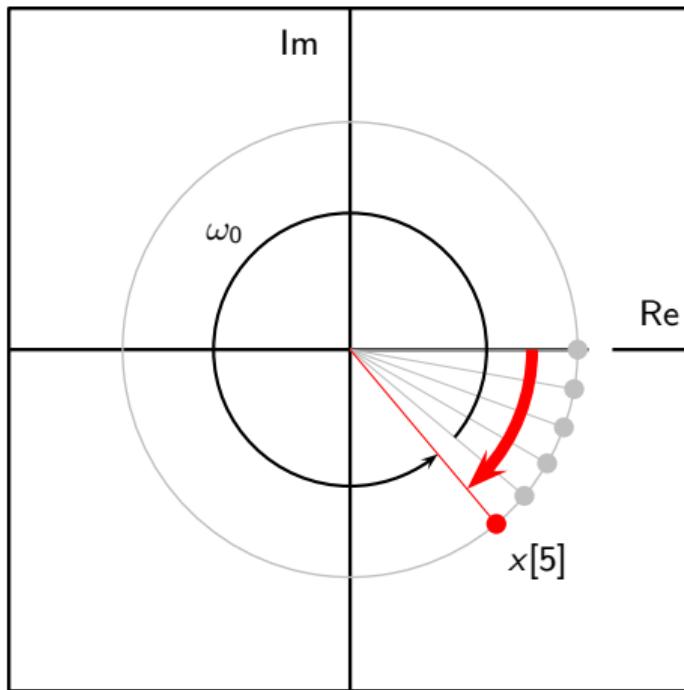
Tricky: $F_s/2 < f_0 < F_s \Rightarrow \pi < \omega_0 < 2\pi$



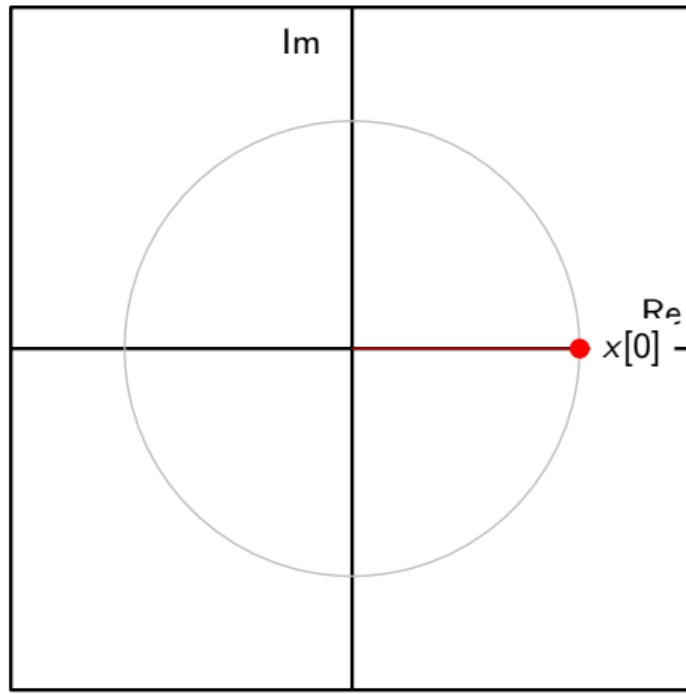
Tricky: $F_s/2 < f_0 < F_s \Rightarrow \pi < \omega_0 < 2\pi$



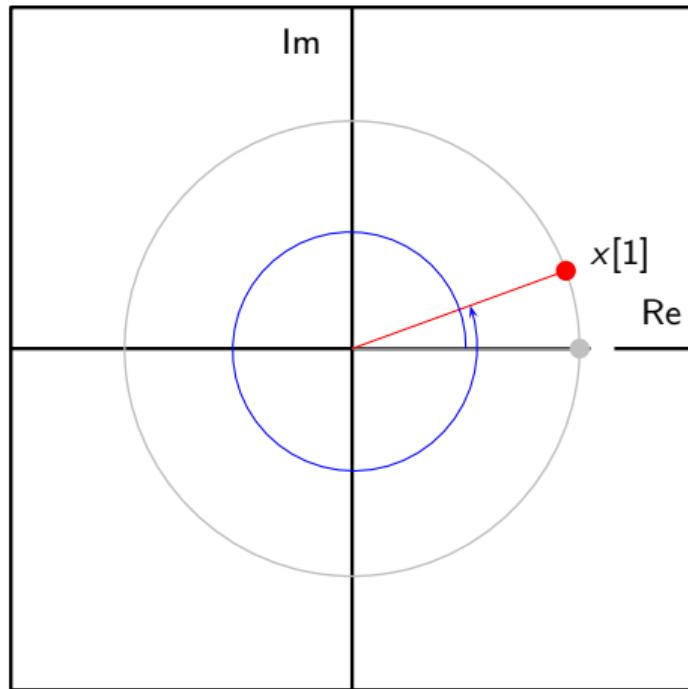
Tricky: $F_s/2 < f_0 < F_s \Rightarrow \pi < \omega_0 < 2\pi$



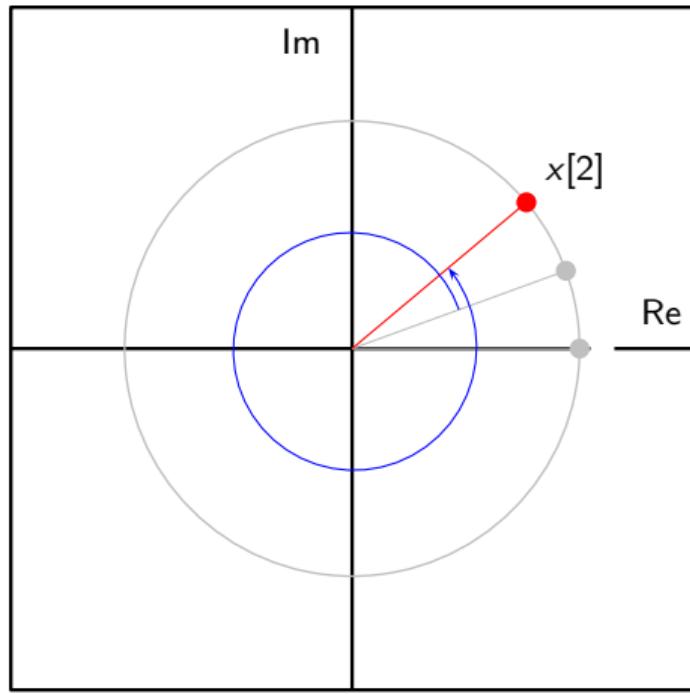
Trouble: $f_0 > F_s \Rightarrow \omega_0 > 2\pi$



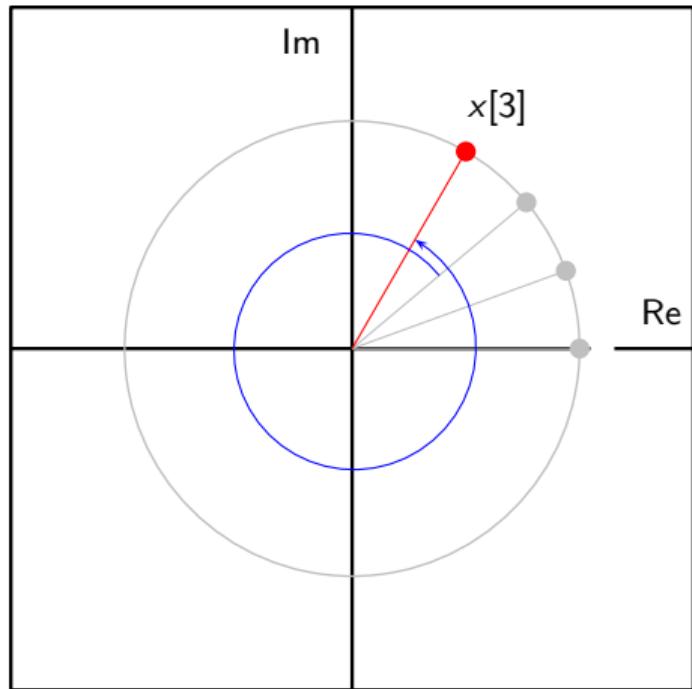
Trouble: $f_0 > F_s \Rightarrow \omega_0 > 2\pi$



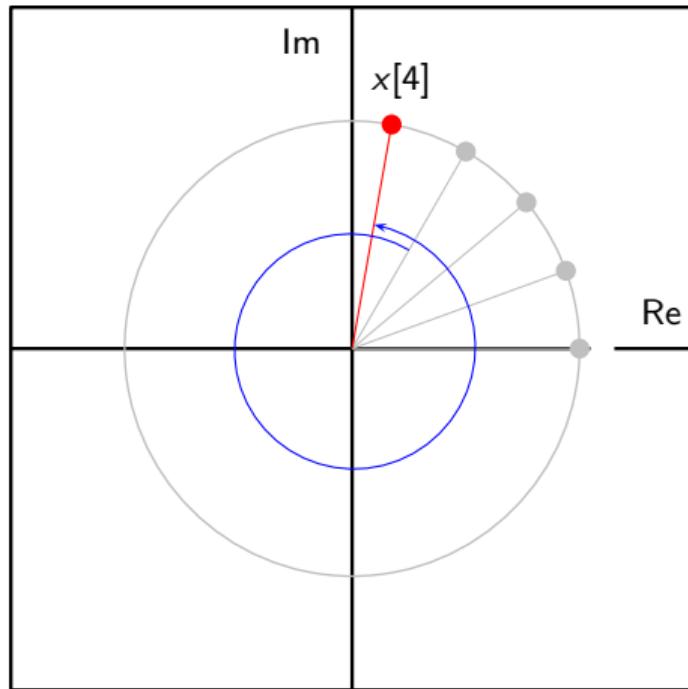
Trouble: $f_0 > F_s \Rightarrow \omega_0 > 2\pi$



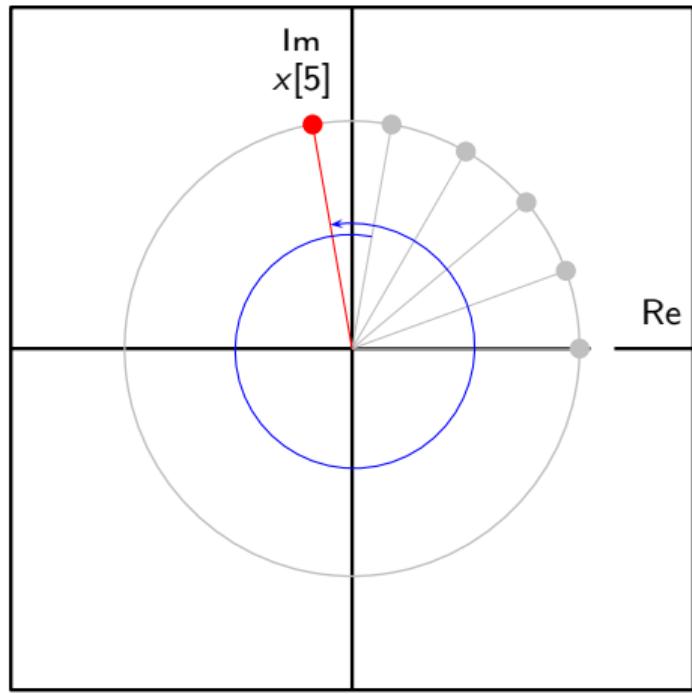
Trouble: $f_0 > F_s \Rightarrow \omega_0 > 2\pi$



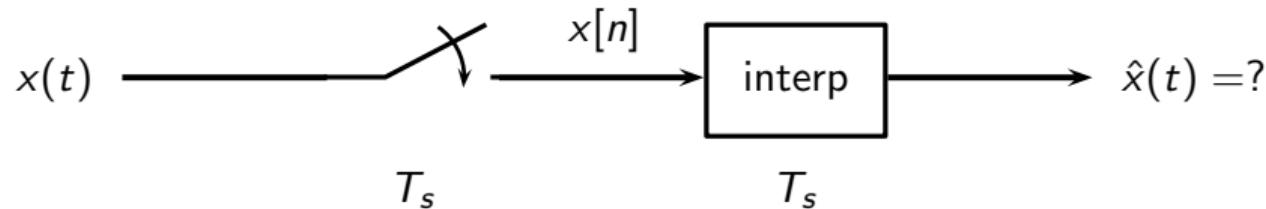
Trouble: $f_0 > F_s \Rightarrow \omega_0 > 2\pi$



Trouble: $f_0 > F_s \Rightarrow \omega_0 > 2\pi$



Aliasing



Aliasing

pick $T_s; F_s = 1/T_s$

input: $x(t) = e^{j2\pi f_0 t}$

digital frequency: $\omega_0 = 2\pi f_0 / F_s$

digital frequency $\hat{x}(t)$

$$f_0 < F_s/2$$

$$0 < \omega_0 < \pi$$

$$e^{j2\pi f_0 t}$$

$$f_0 = F_s/2$$

$$\omega_0 = \pi$$

$$e^{j2\pi f_0 t}$$

$$F_s/2 < f_0 < F_s$$

$$\pi < \omega_0 < 2\pi$$

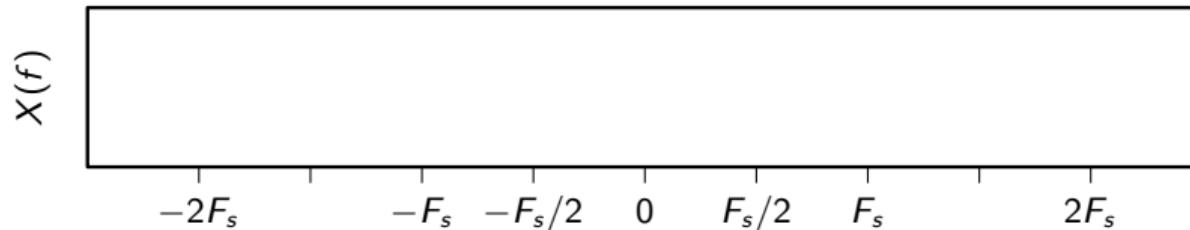
$$e^{j2\pi f_1 t}, \quad f_1 = f_0 - F_s < 0$$

$$f_0 > F_s$$

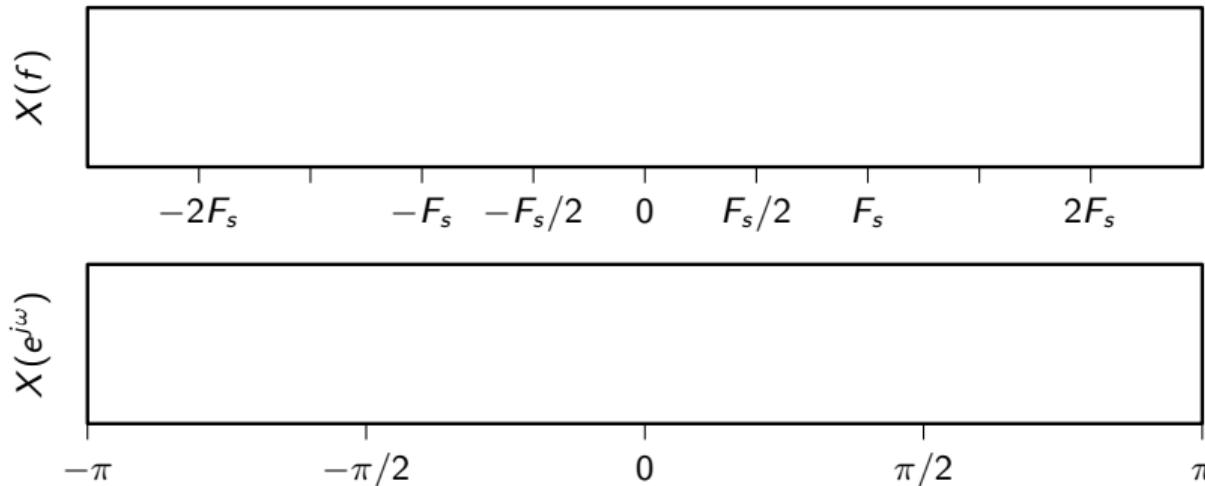
$$\omega_0 > 2\pi$$

$$e^{j2\pi f_2 t}, \quad f_2 = f_0 \bmod F_s$$

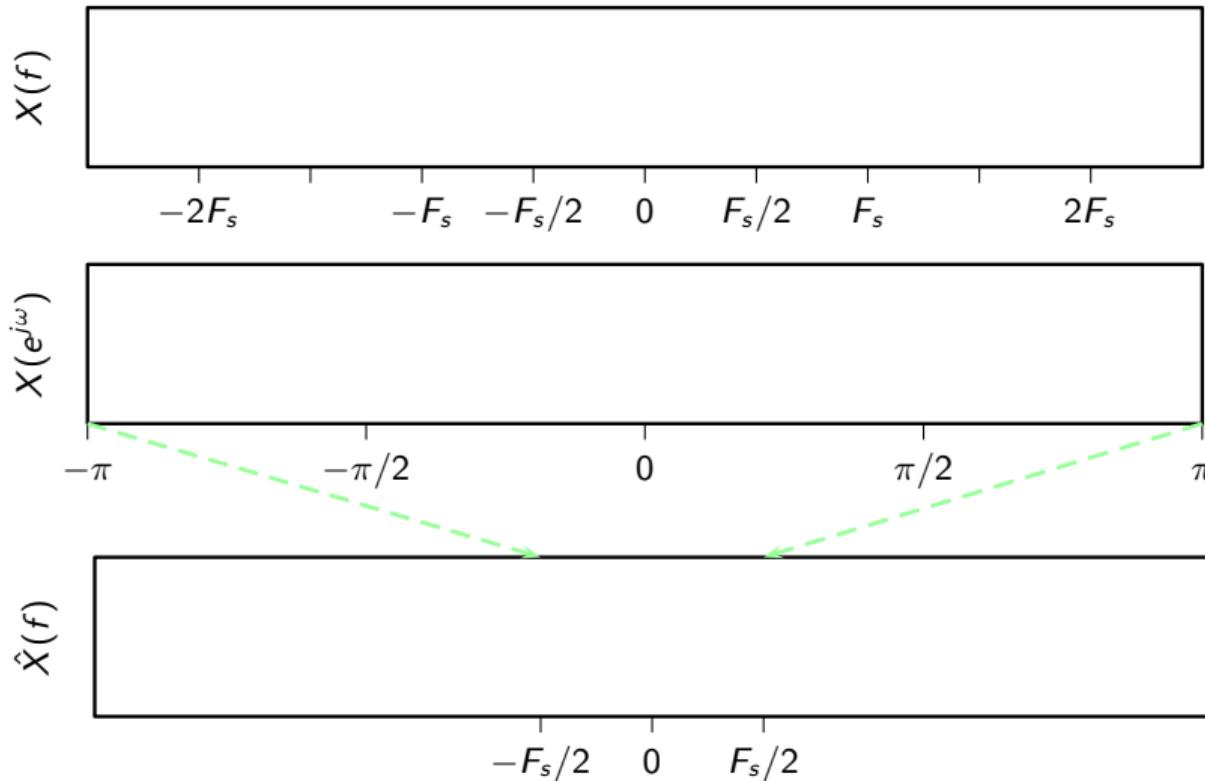
Aliasing of sinusoids: increasing the input frequency



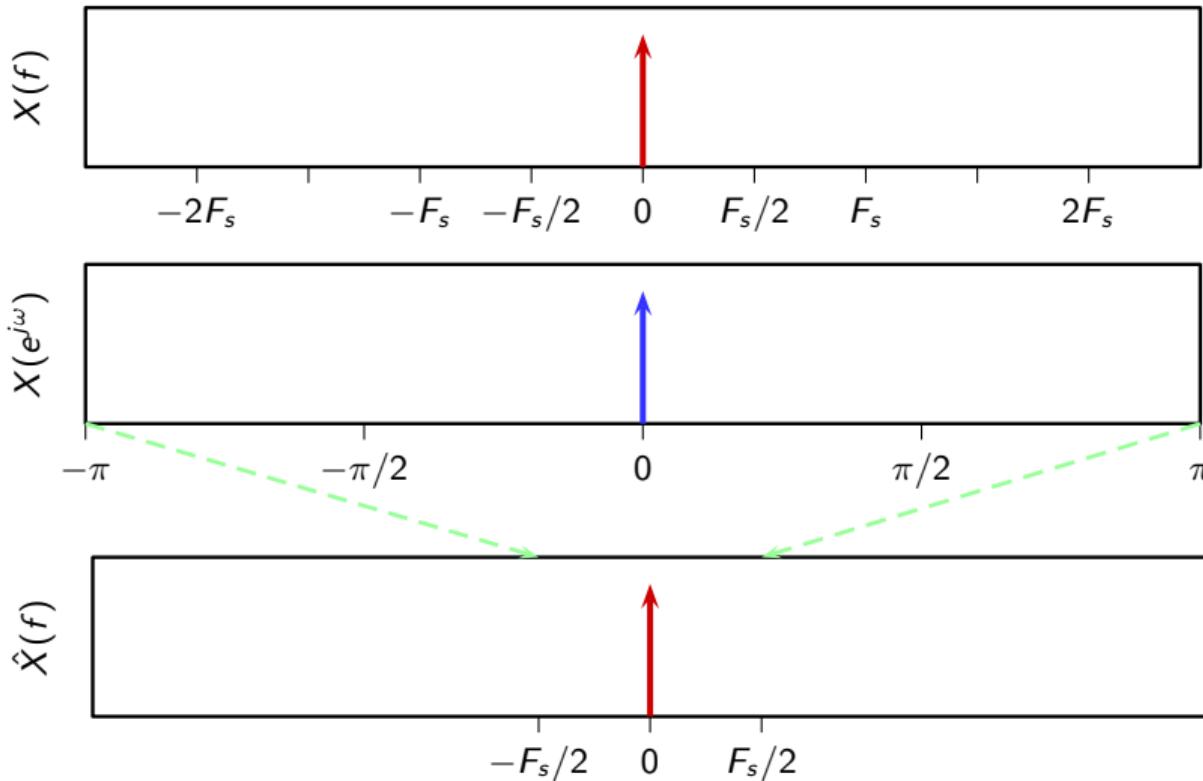
Aliasing of sinusoids: increasing the input frequency



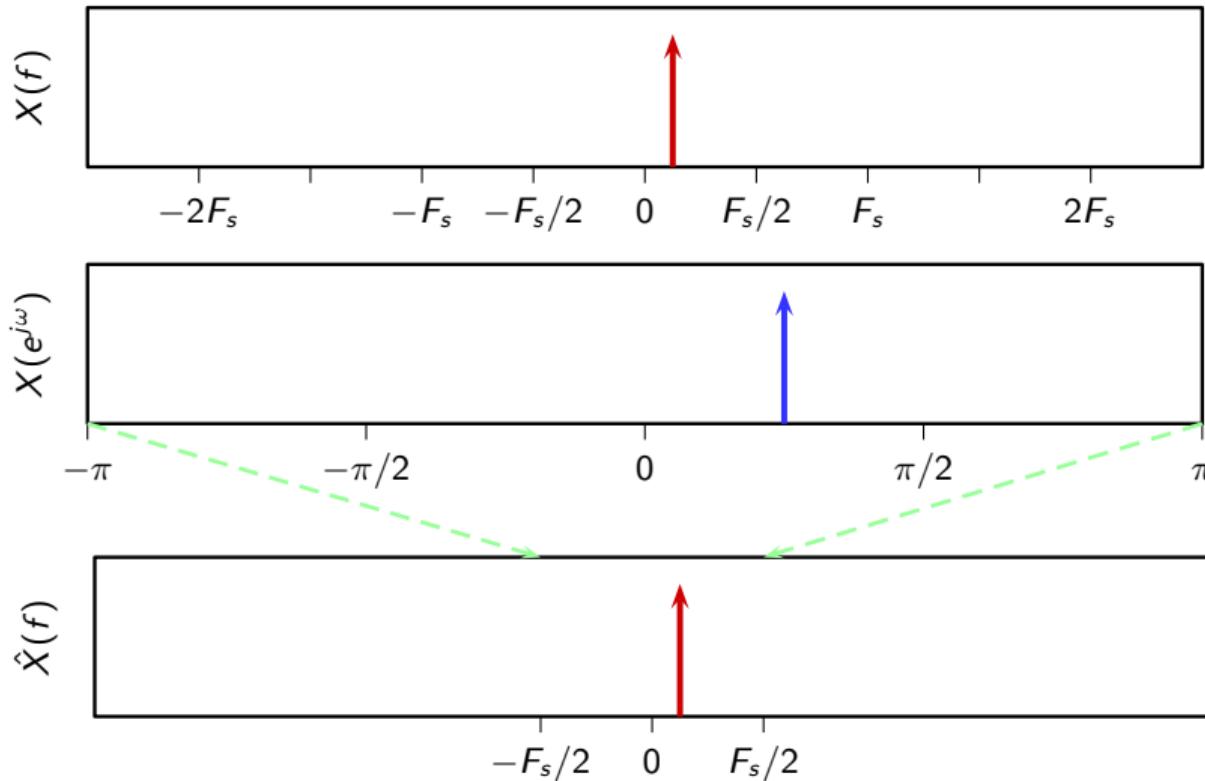
Aliasing of sinusoids: increasing the input frequency



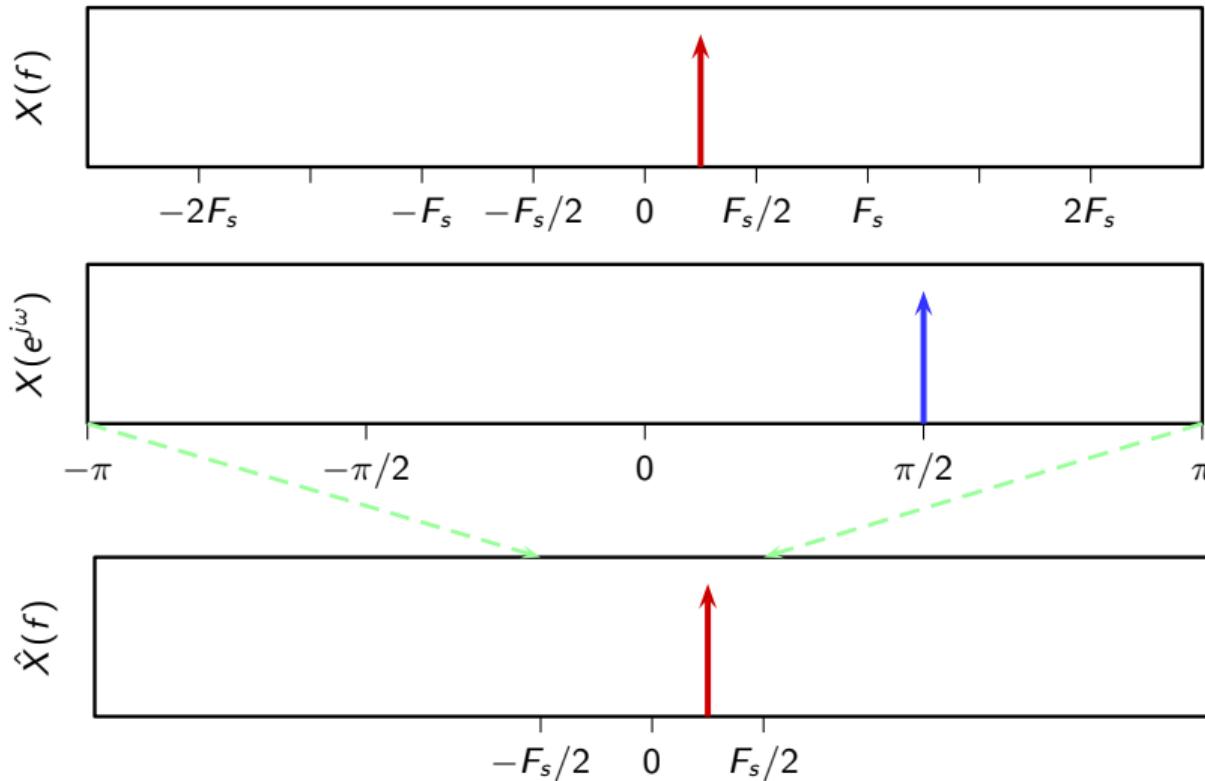
Aliasing of sinusoids: increasing the input frequency



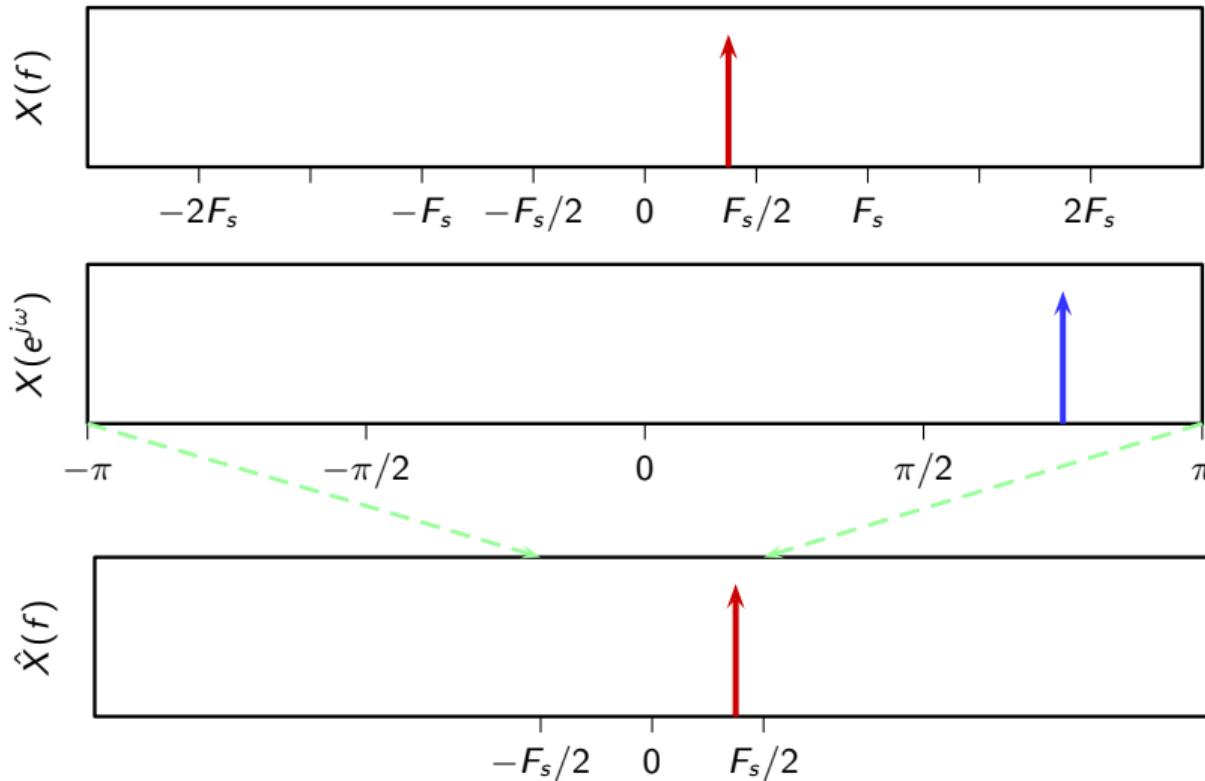
Aliasing of sinusoids: increasing the input frequency



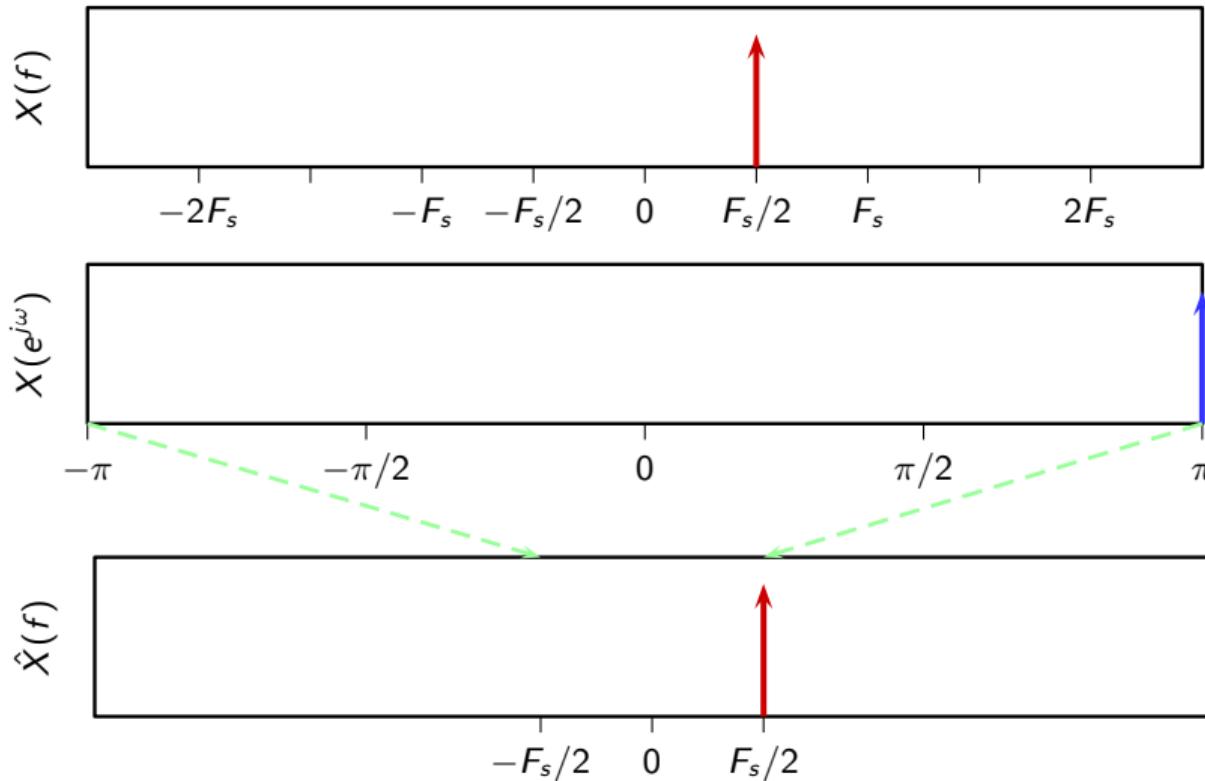
Aliasing of sinusoids: increasing the input frequency



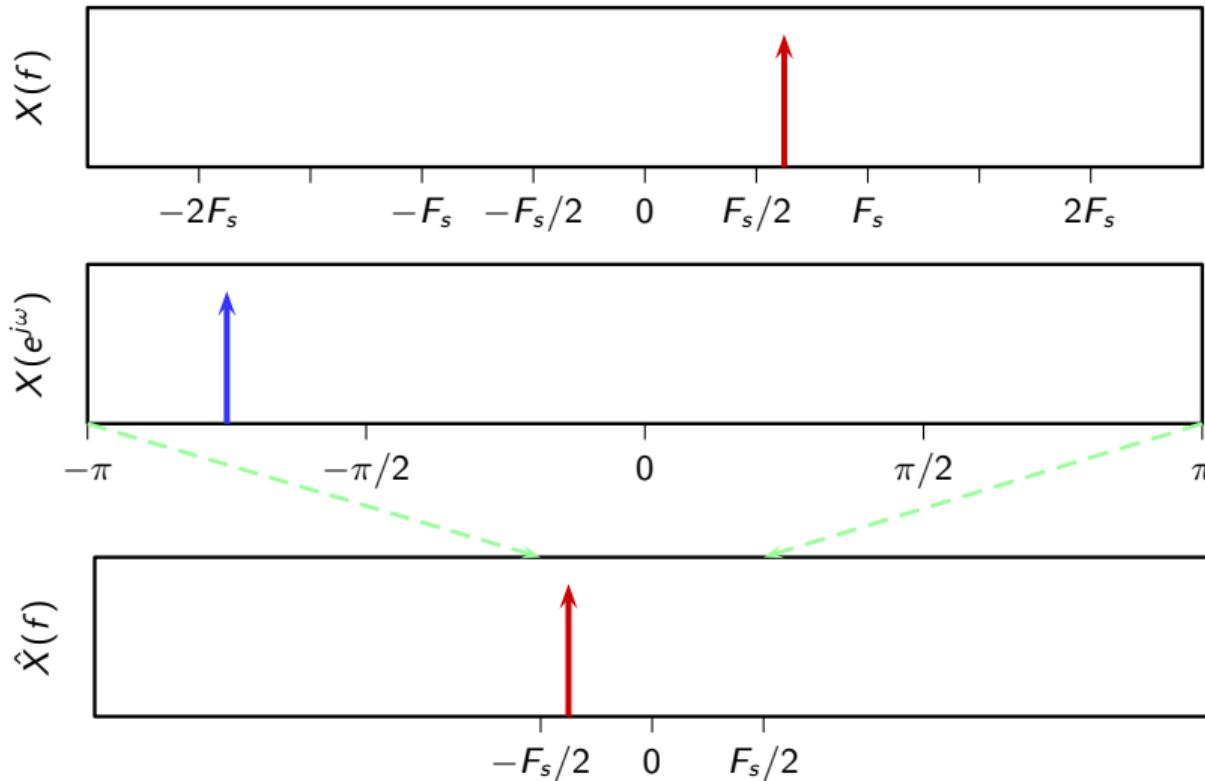
Aliasing of sinusoids: increasing the input frequency



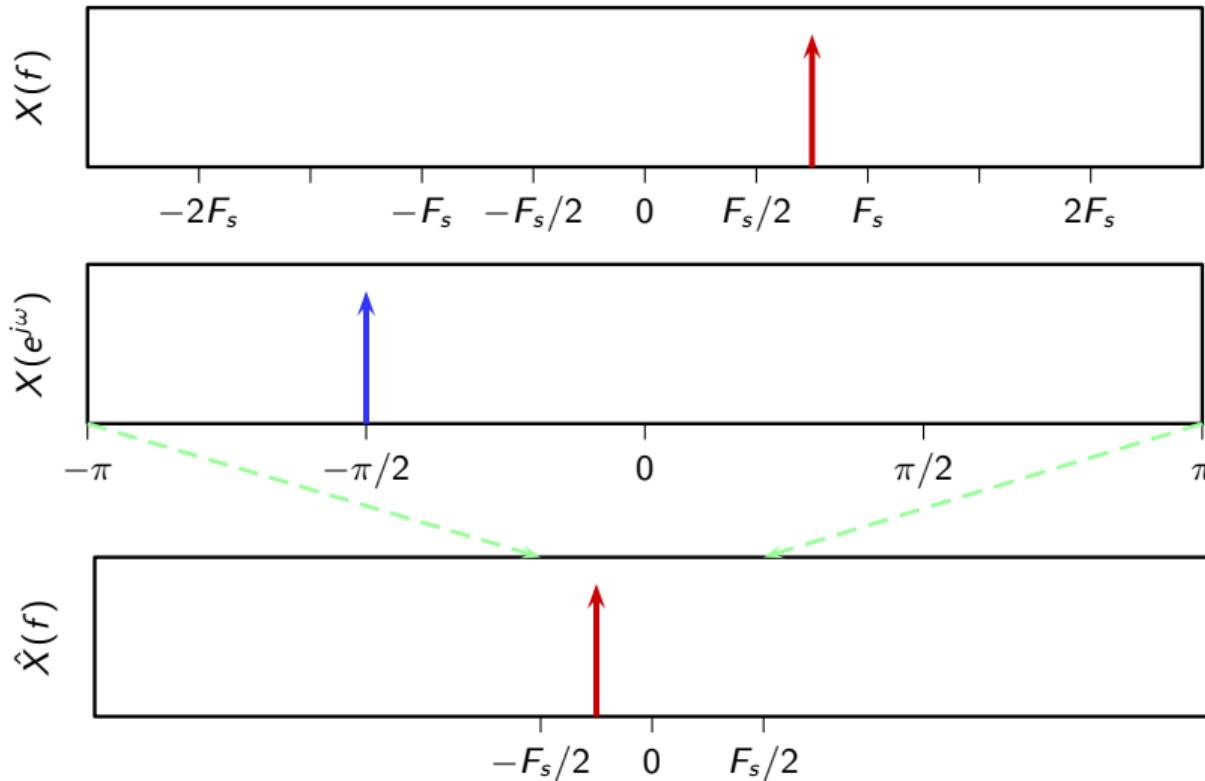
Aliasing of sinusoids: increasing the input frequency



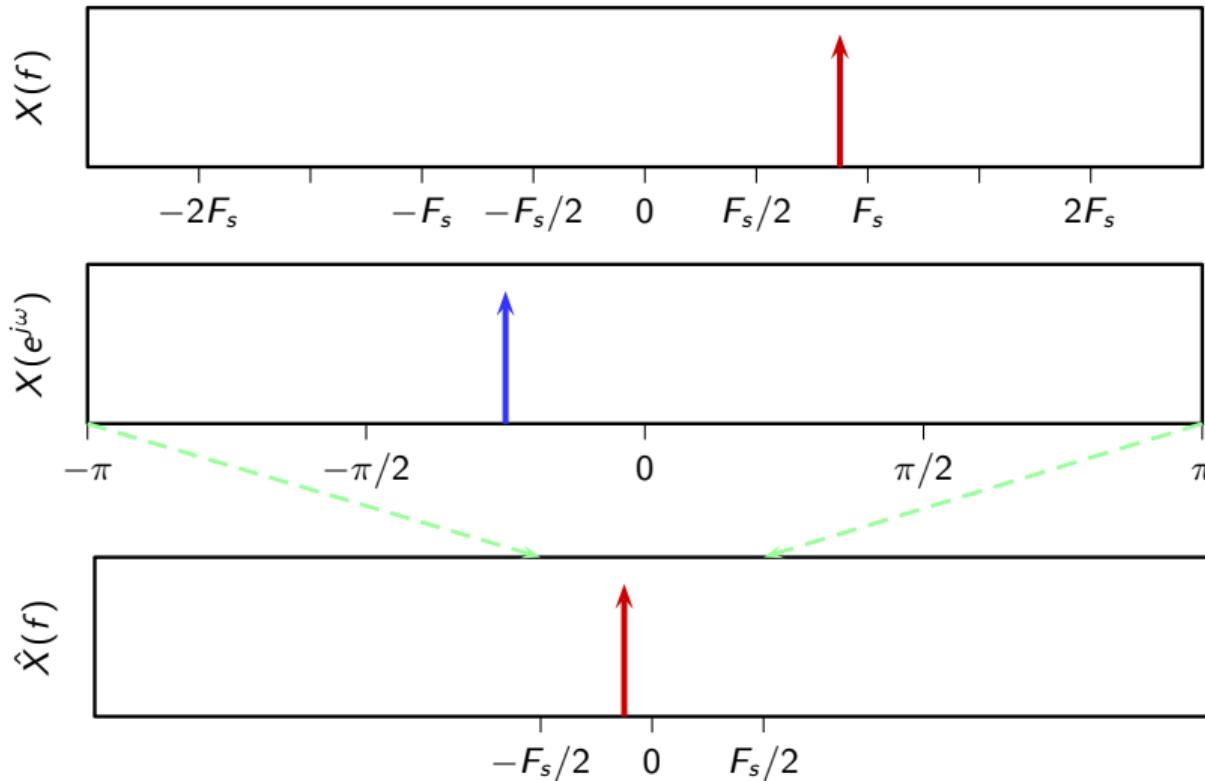
Aliasing of sinusoids: increasing the input frequency



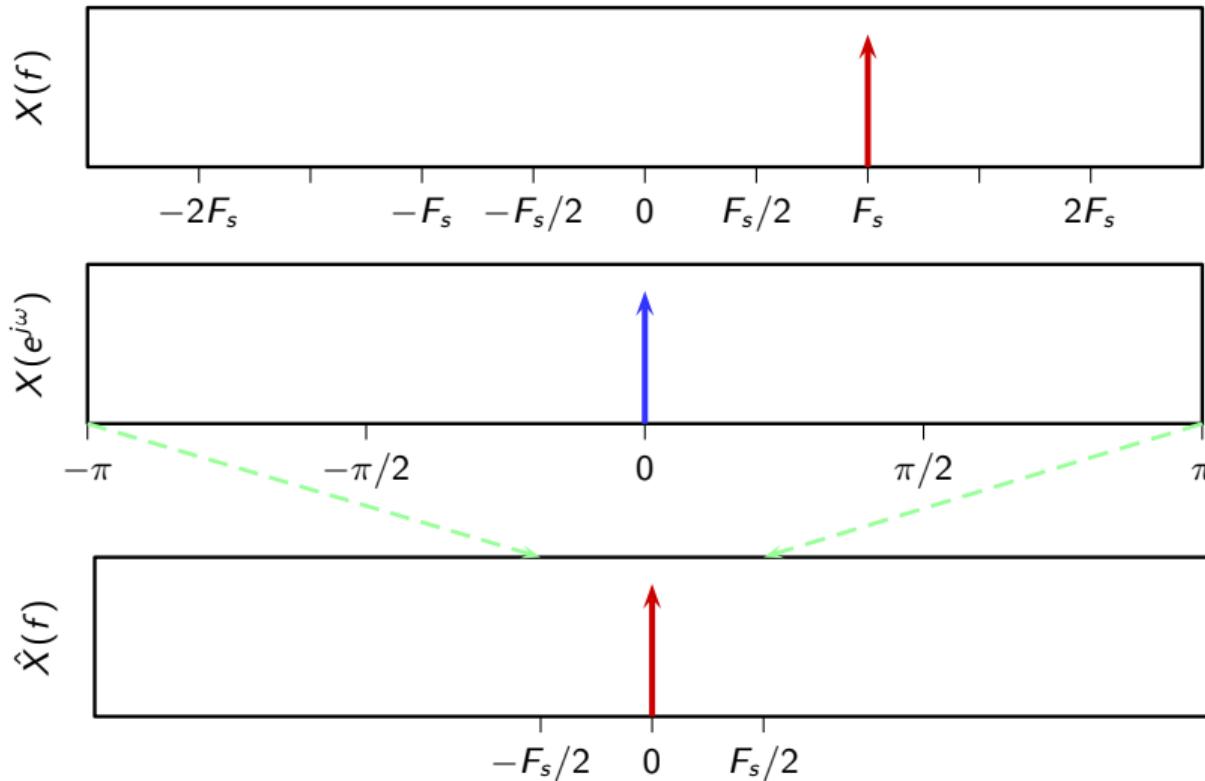
Aliasing of sinusoids: increasing the input frequency



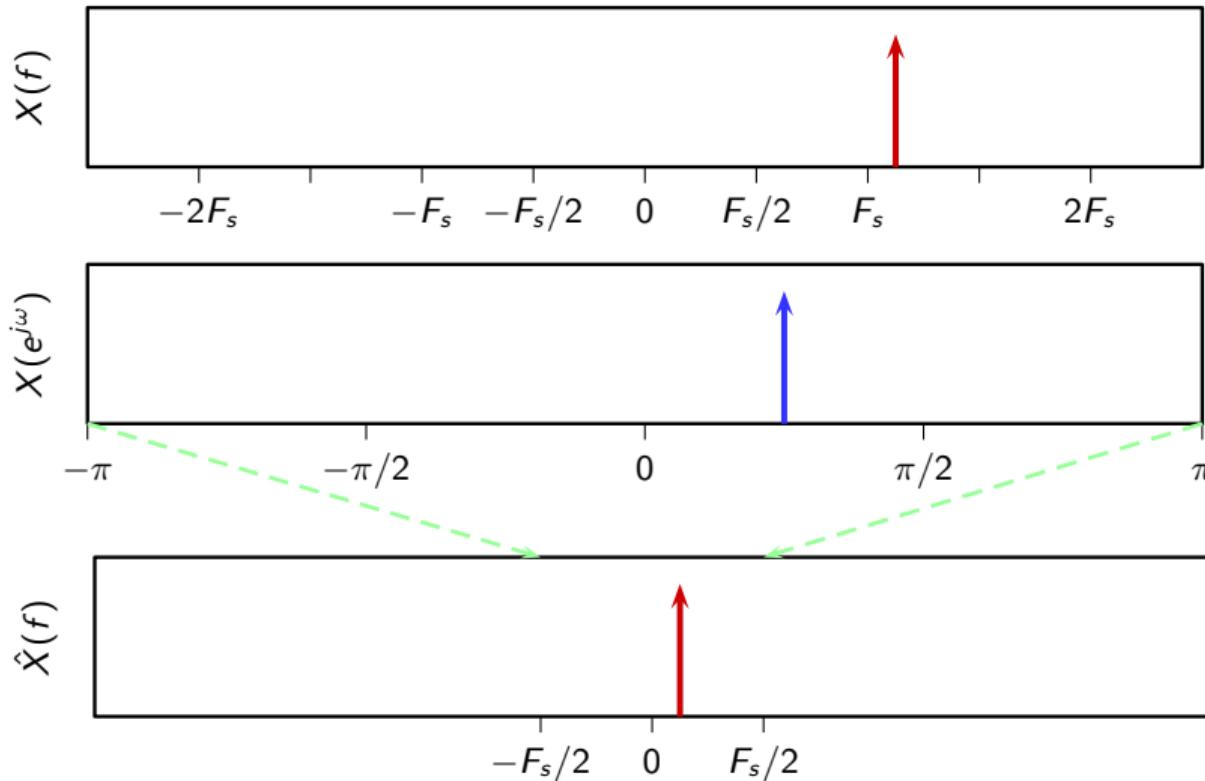
Aliasing of sinusoids: increasing the input frequency



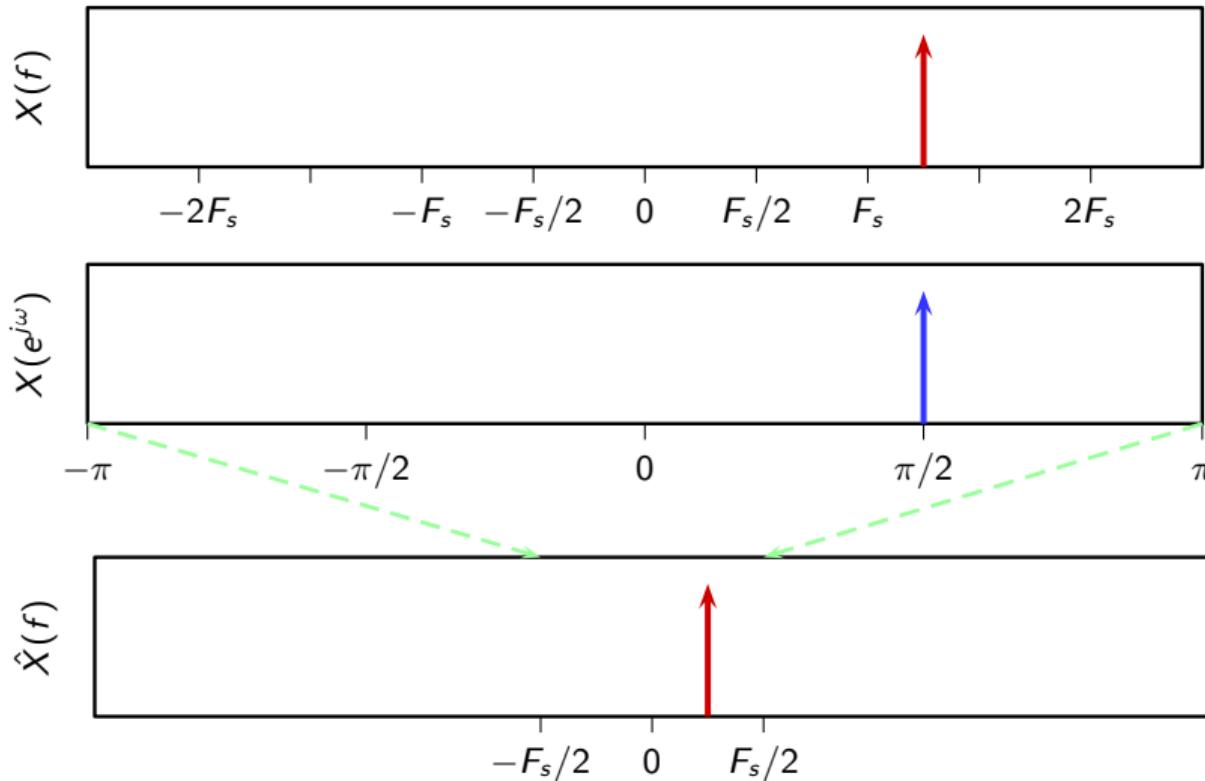
Aliasing of sinusoids: increasing the input frequency



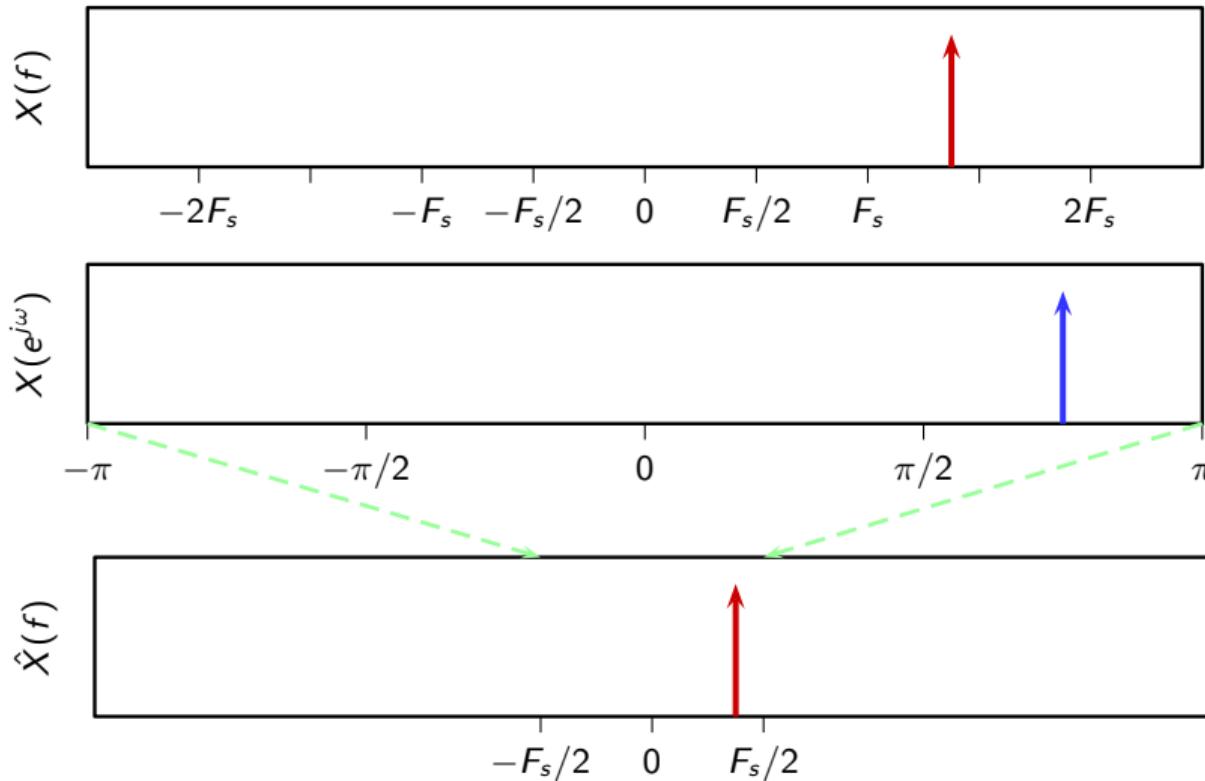
Aliasing of sinusoids: increasing the input frequency



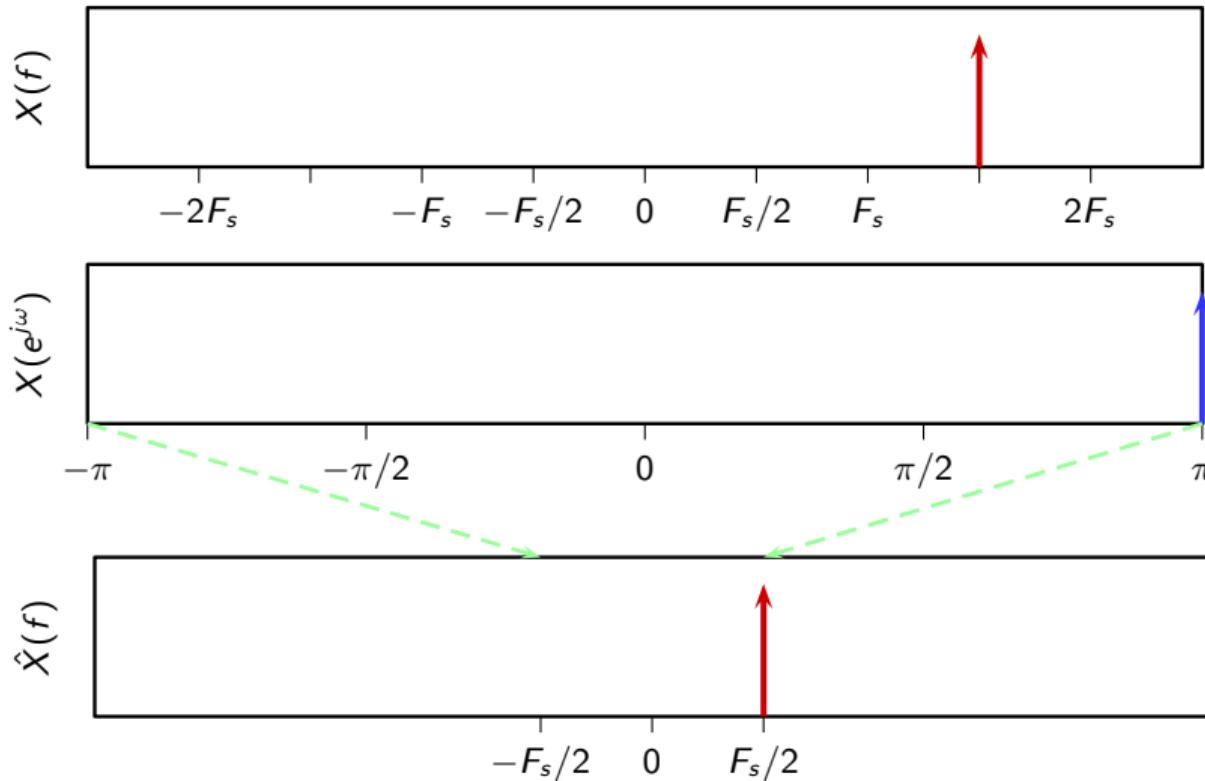
Aliasing of sinusoids: increasing the input frequency



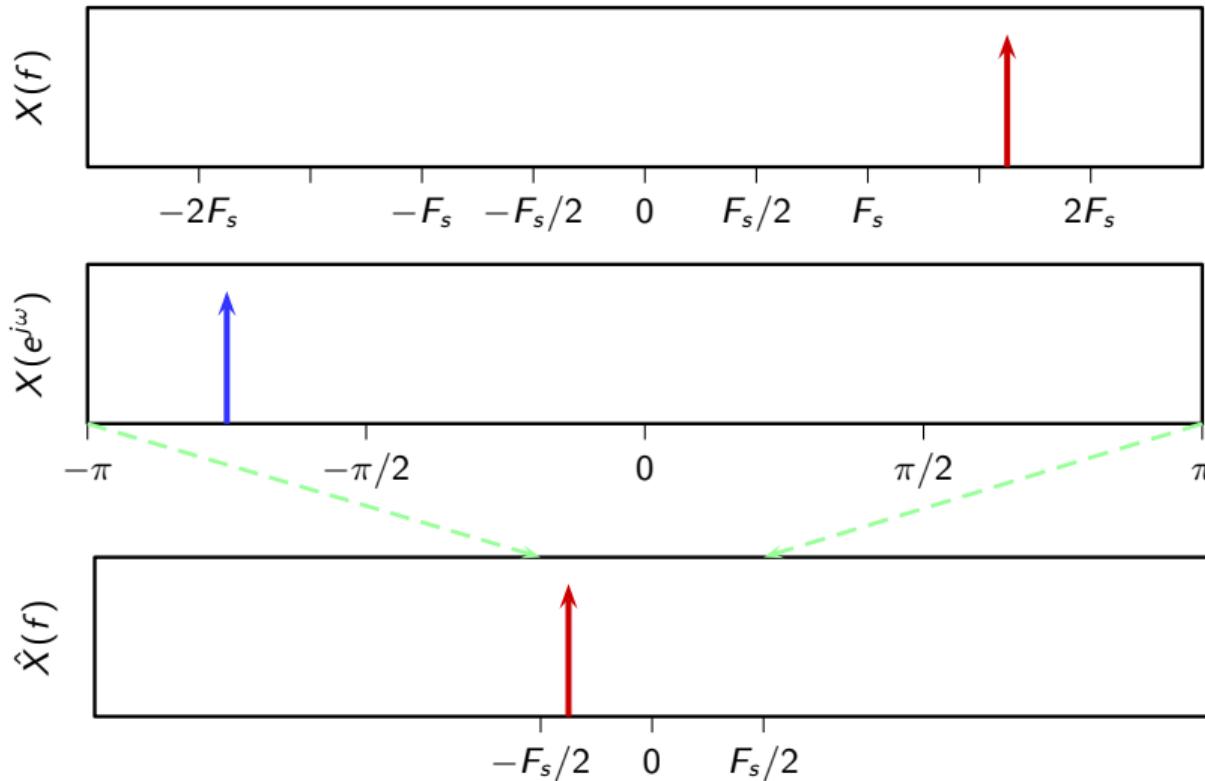
Aliasing of sinusoids: increasing the input frequency



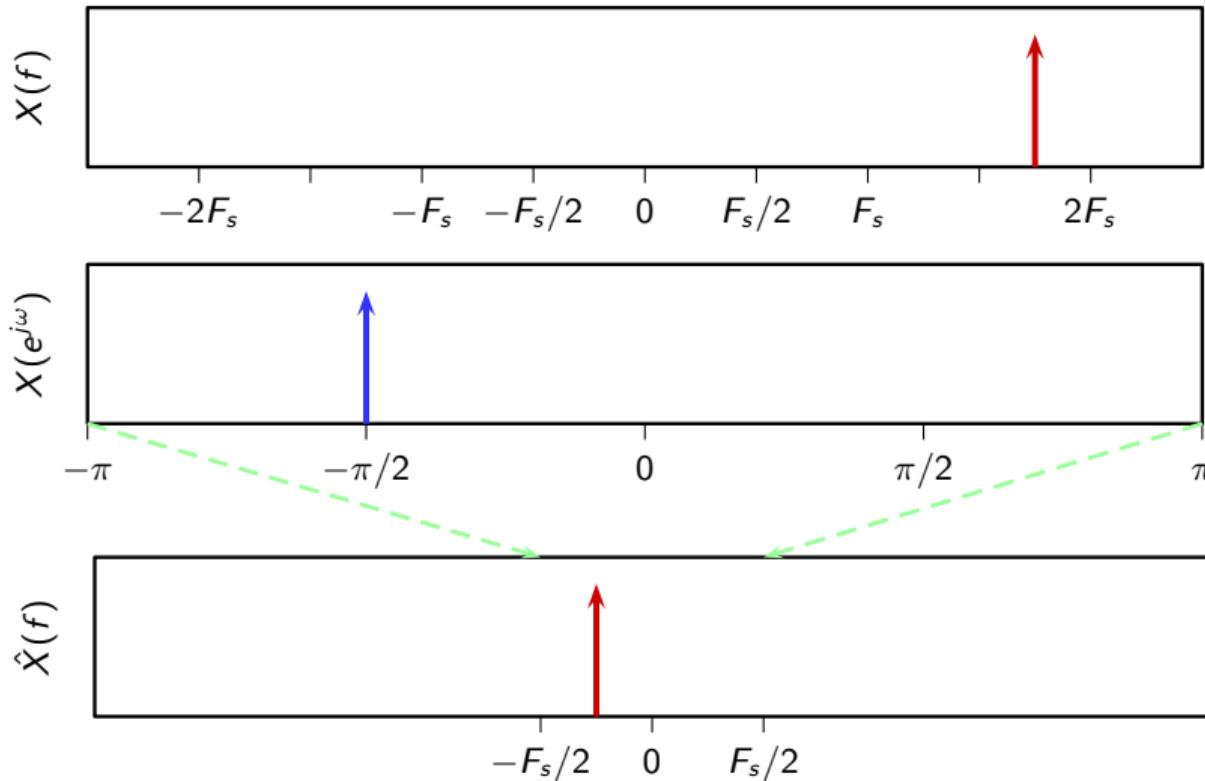
Aliasing of sinusoids: increasing the input frequency



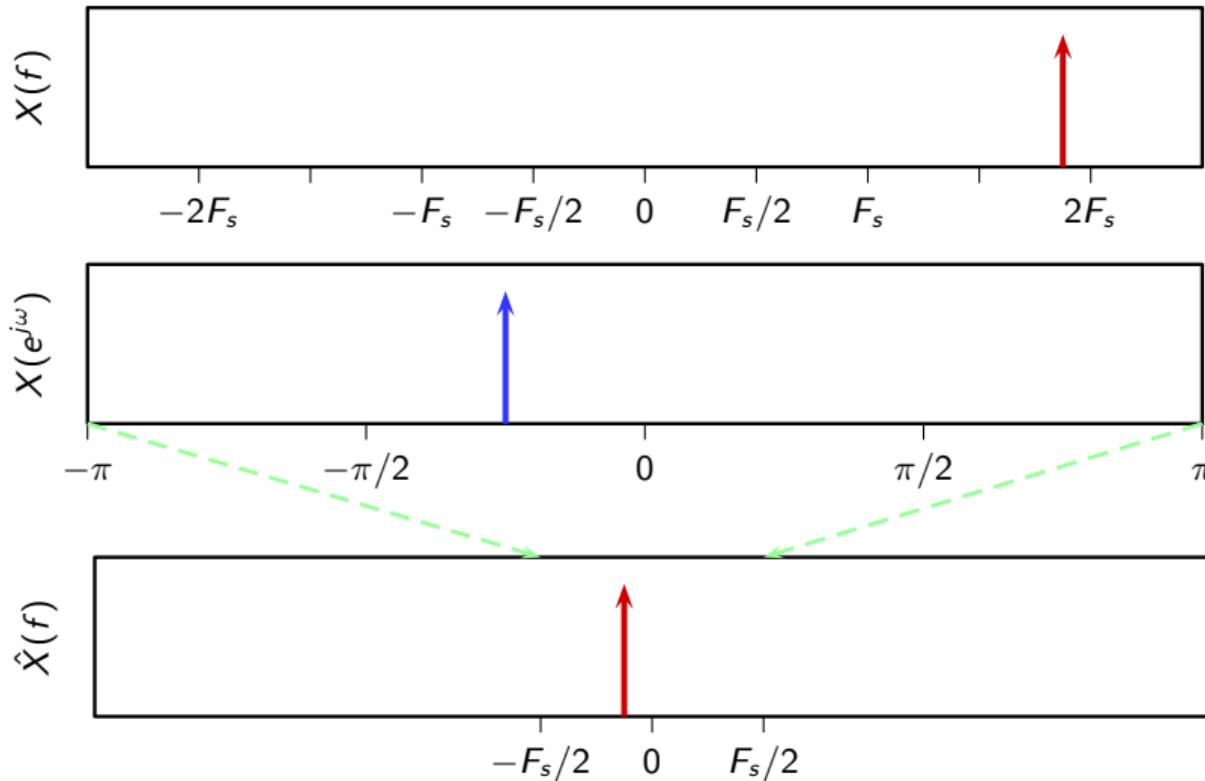
Aliasing of sinusoids: increasing the input frequency



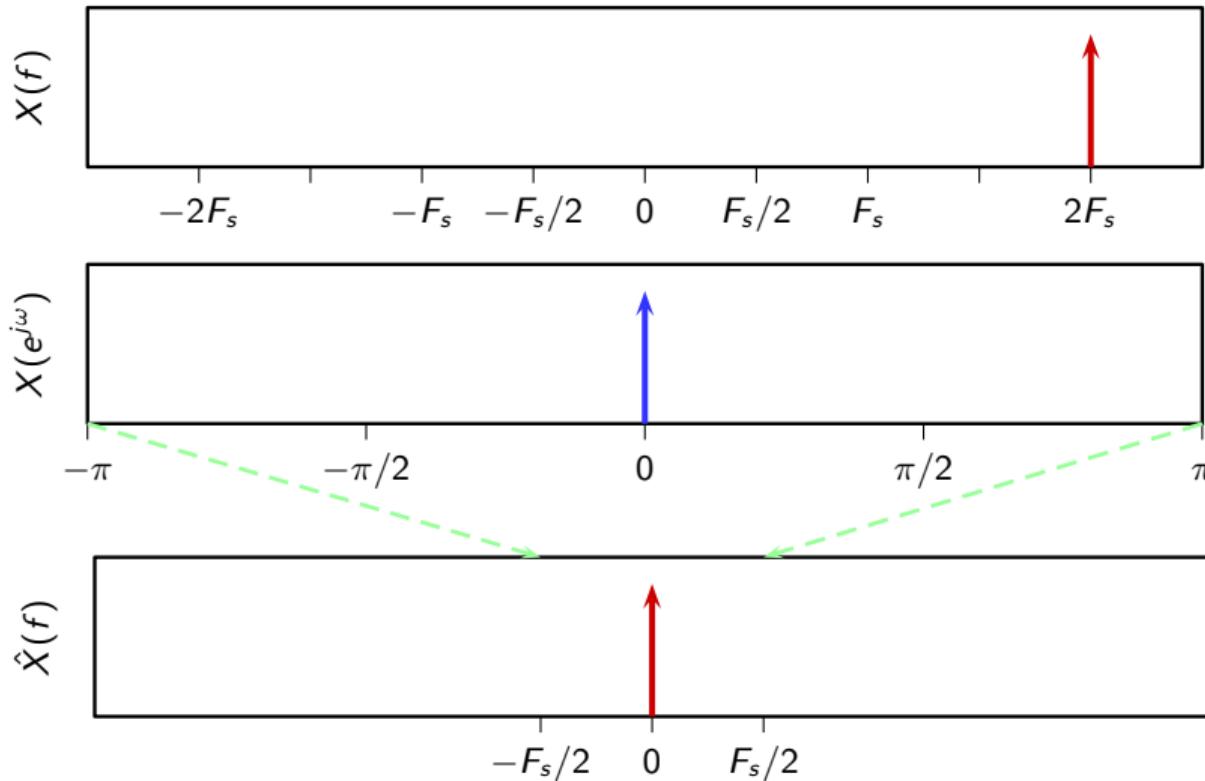
Aliasing of sinusoids: increasing the input frequency



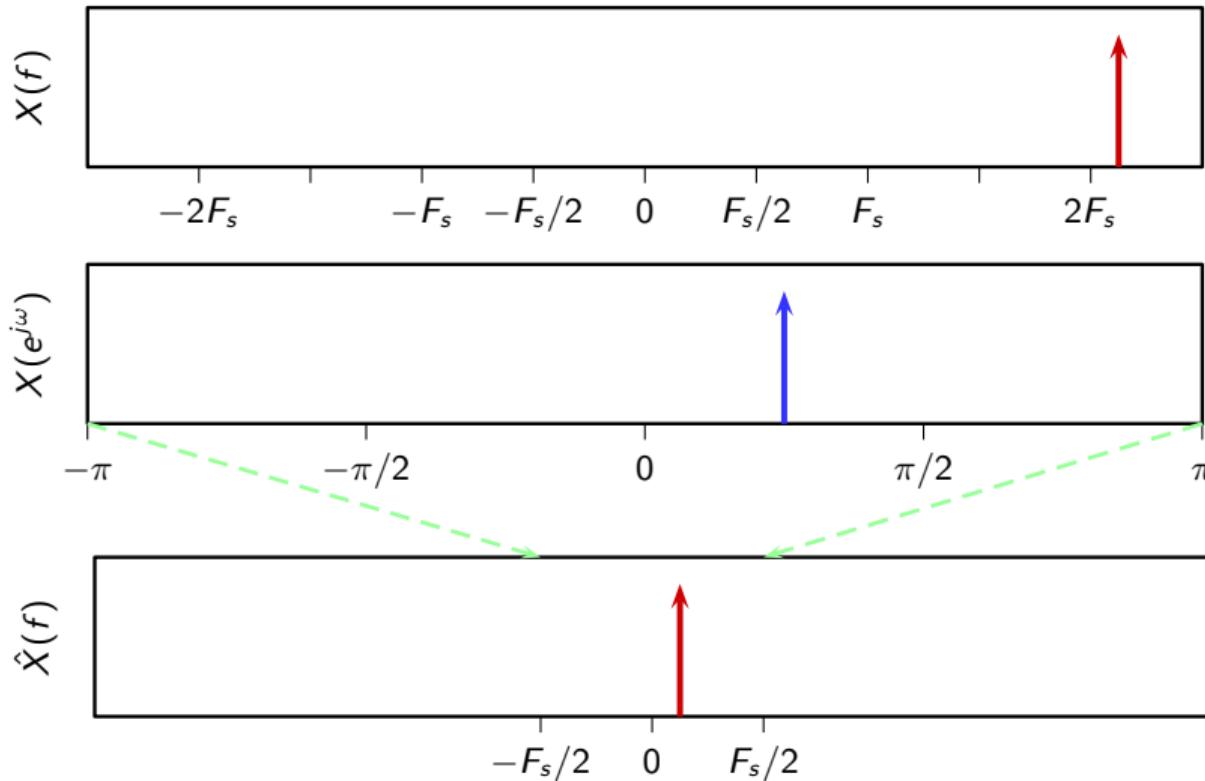
Aliasing of sinusoids: increasing the input frequency



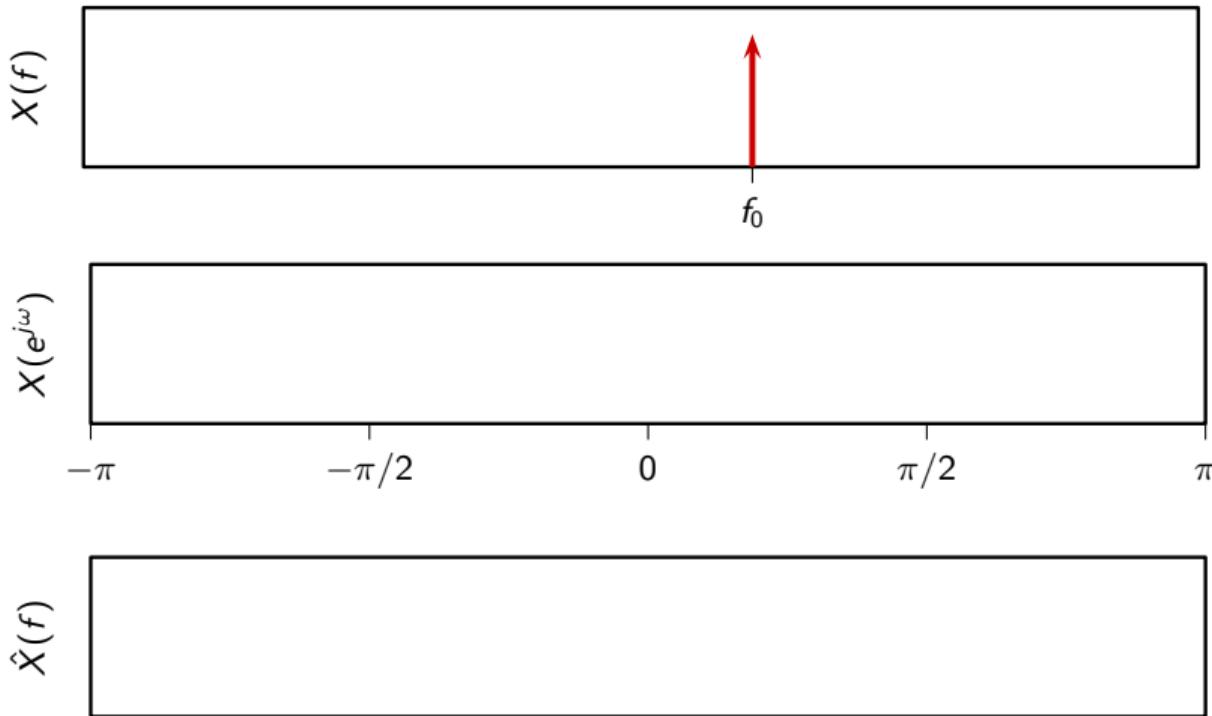
Aliasing of sinusoids: increasing the input frequency



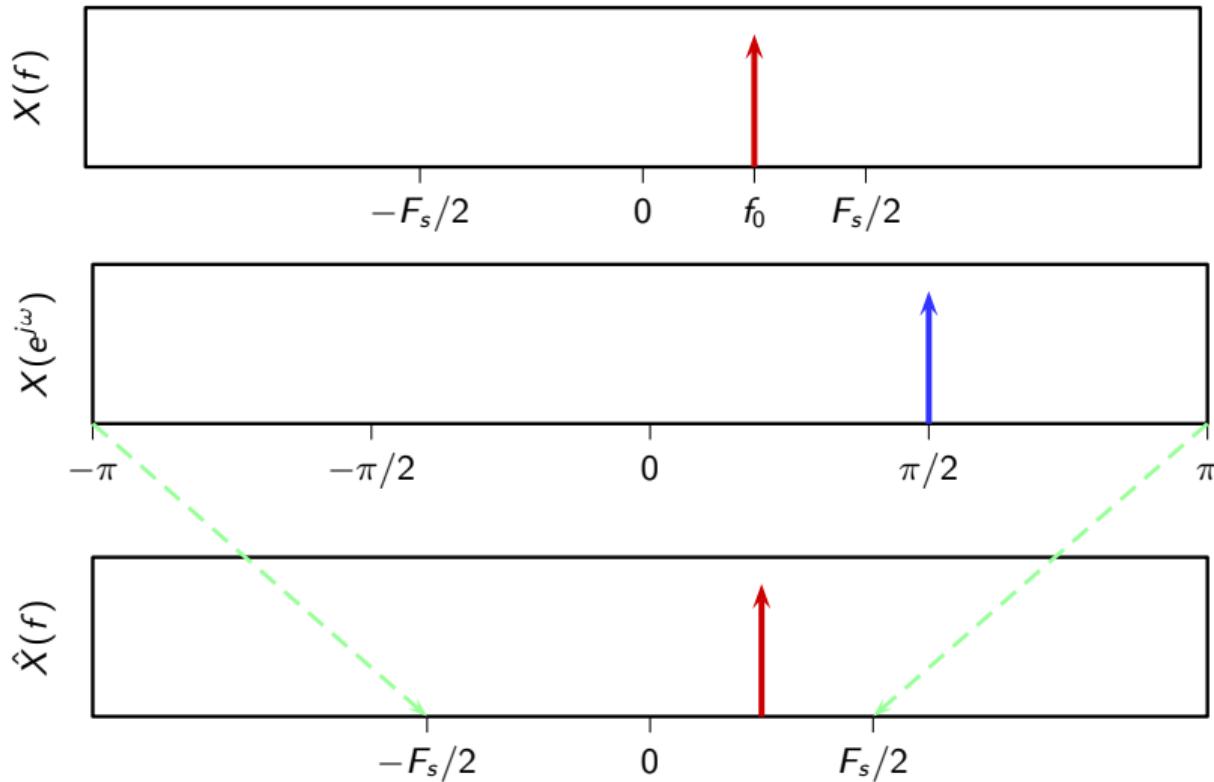
Aliasing of sinusoids: increasing the input frequency



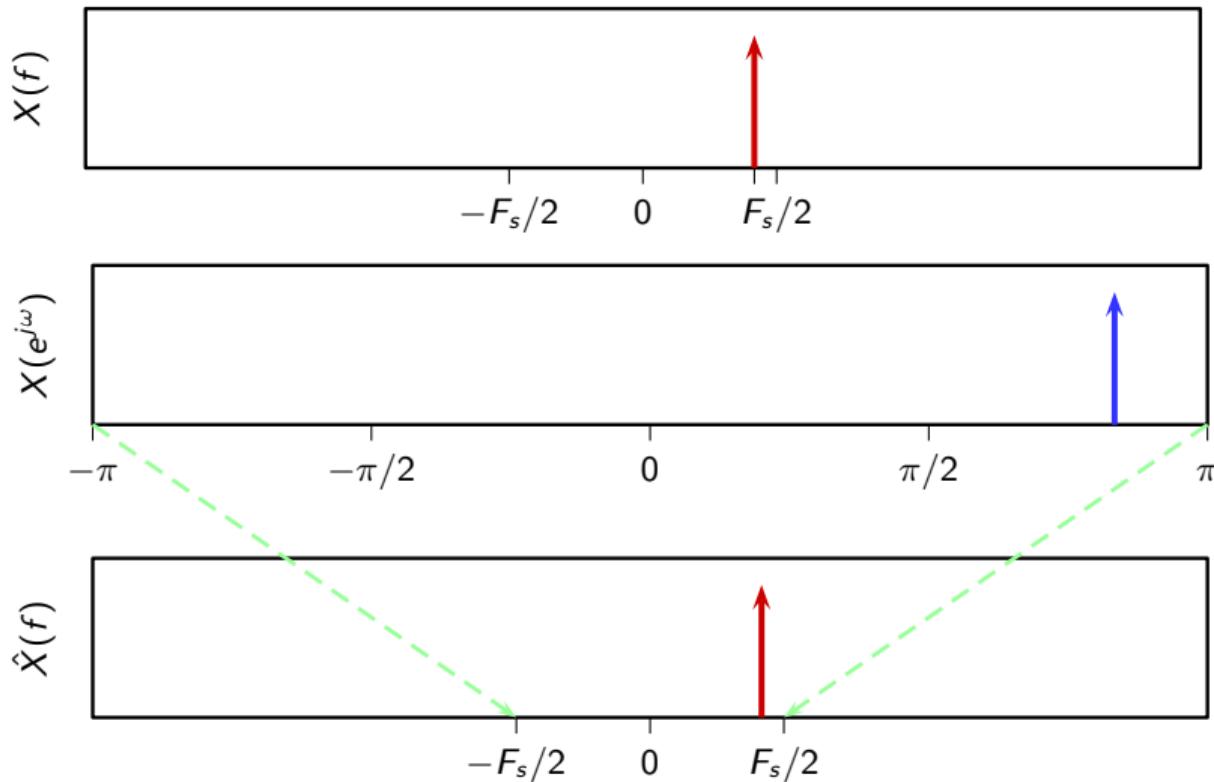
Aliasing of sinusoids: decreasing the sampling frequency



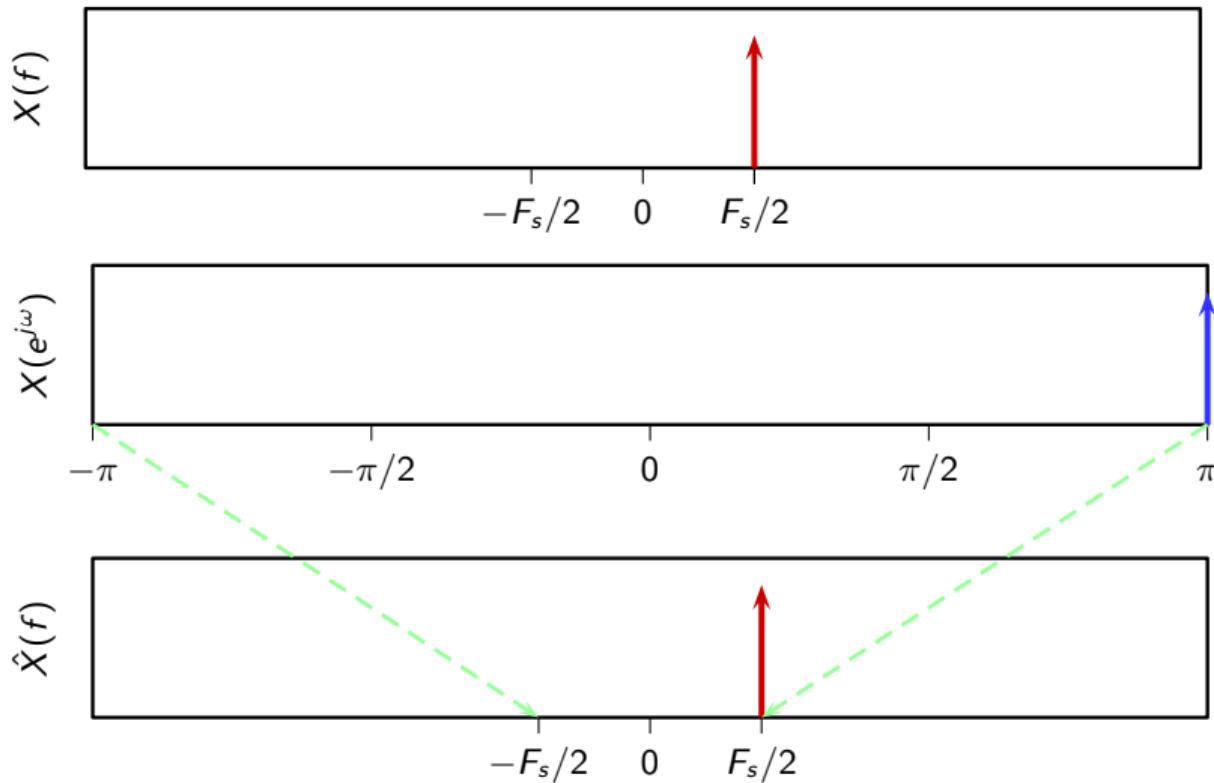
Aliasing of sinusoids: decreasing the sampling frequency



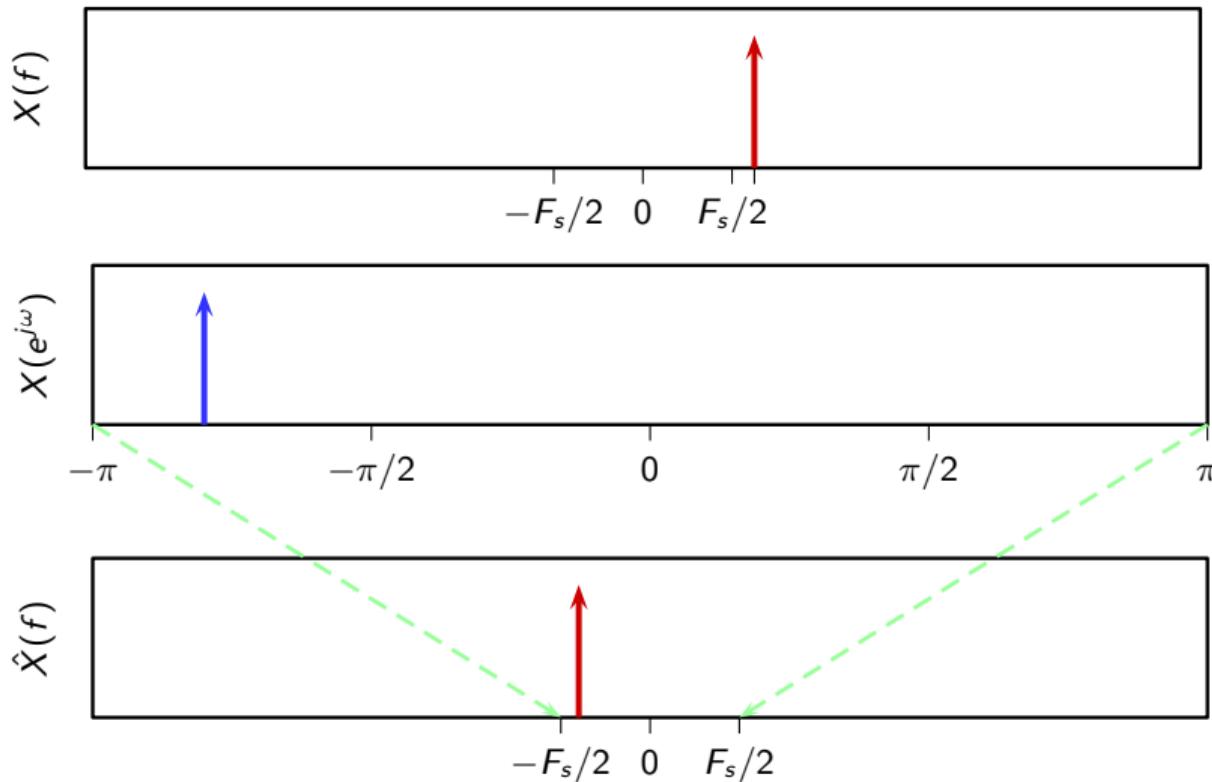
Aliasing of sinusoids: decreasing the sampling frequency



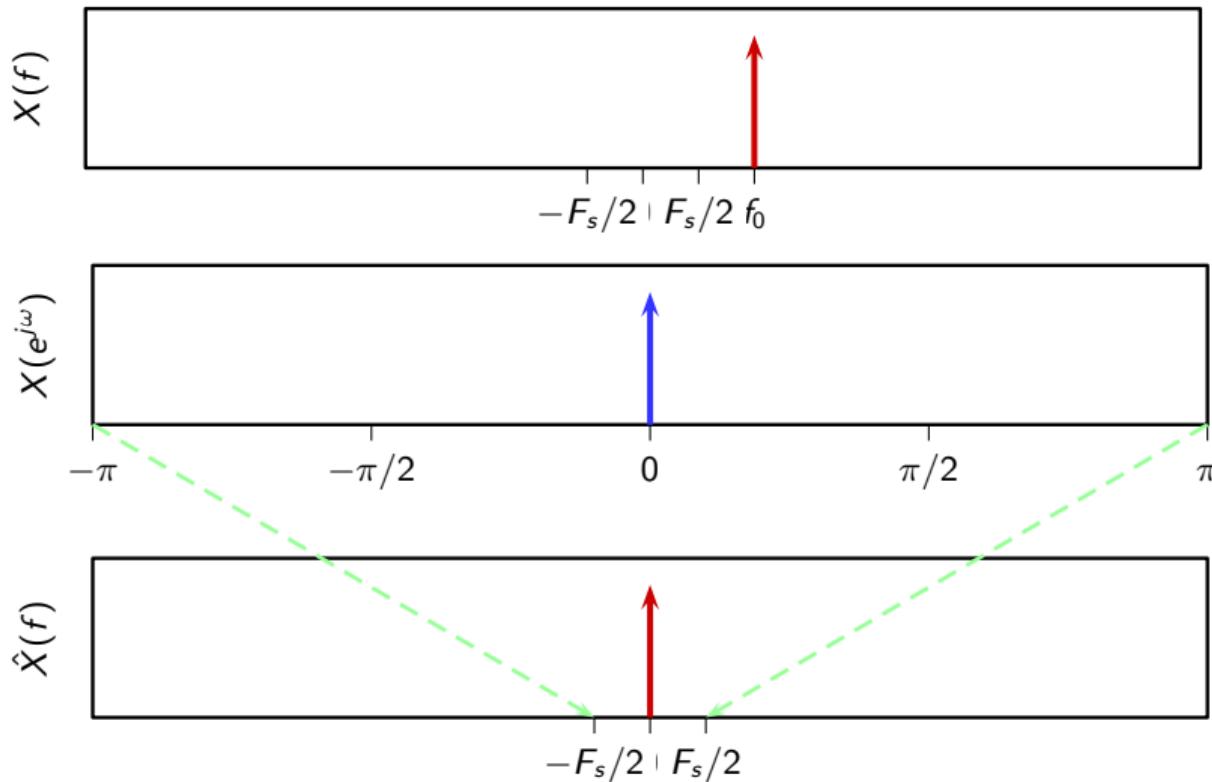
Aliasing of sinusoids: decreasing the sampling frequency



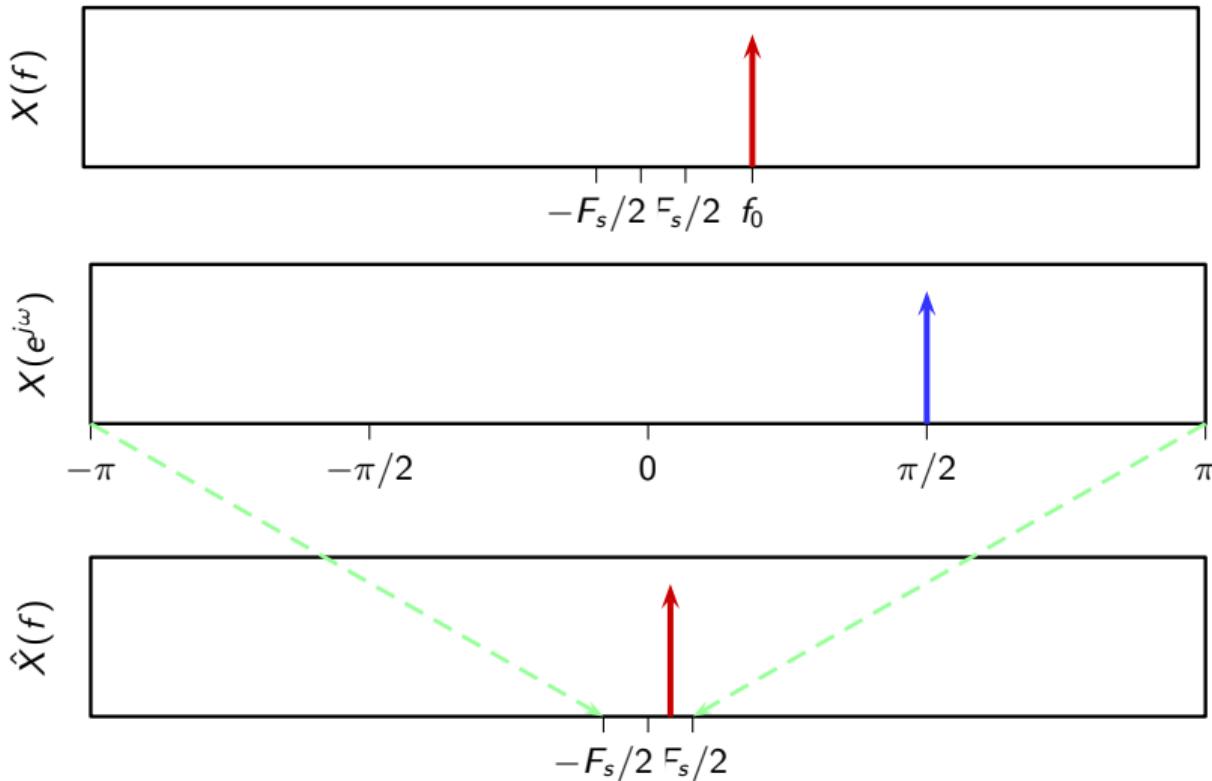
Aliasing of sinusoids: decreasing the sampling frequency



Aliasing of sinusoids: decreasing the sampling frequency



Aliasing of sinusoids: decreasing the sampling frequency

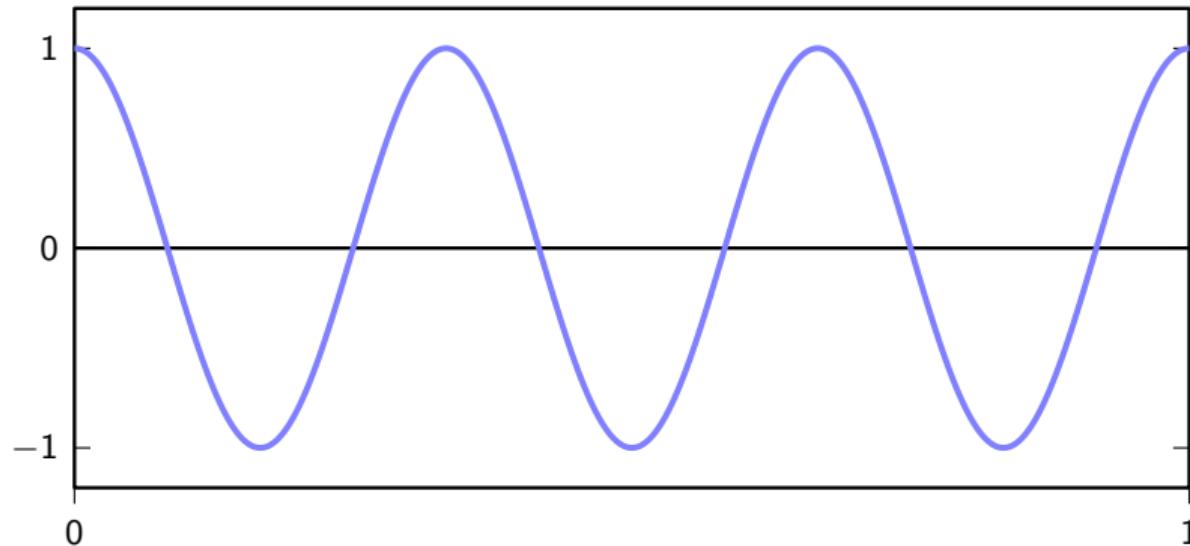


Sampling a Sinusoid

sampling frequency	digital frequency	interpolation
$F_s > 2f_0$	$0 < \omega_0 < \pi$	OK: $\hat{f}_0 = f_0$
$F_s = 2f_0$	$\omega_0 = \pi$	OK (max frequency $\hat{f}_0 = F_s$)
$f_0 < F_s < 2f_0$	$\pi < \omega_0 < 2\pi$	negative frequency: $\hat{f}_0 = f_0 - F_s$
$F_s < f_0$	$\omega_0 > 2\pi$	full aliasing: $\hat{f}_0 = f_0 \bmod F_s$

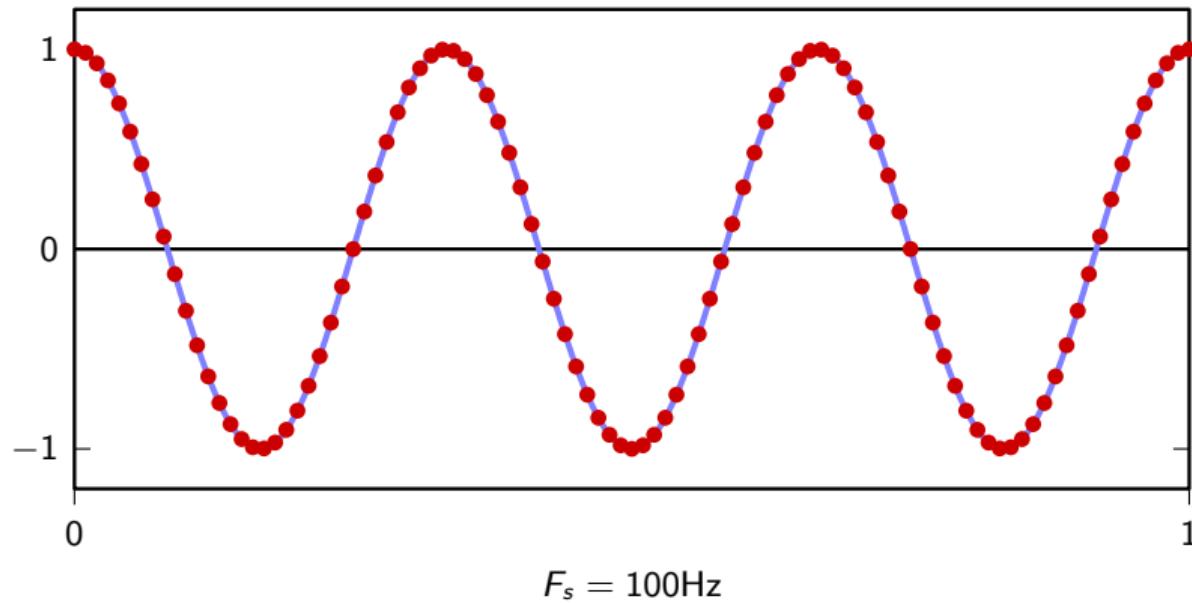
Aliasing: Sampling a Sinusoid

$$x(t) = \cos(6\pi t) \quad (f_0 = 3\text{Hz})$$



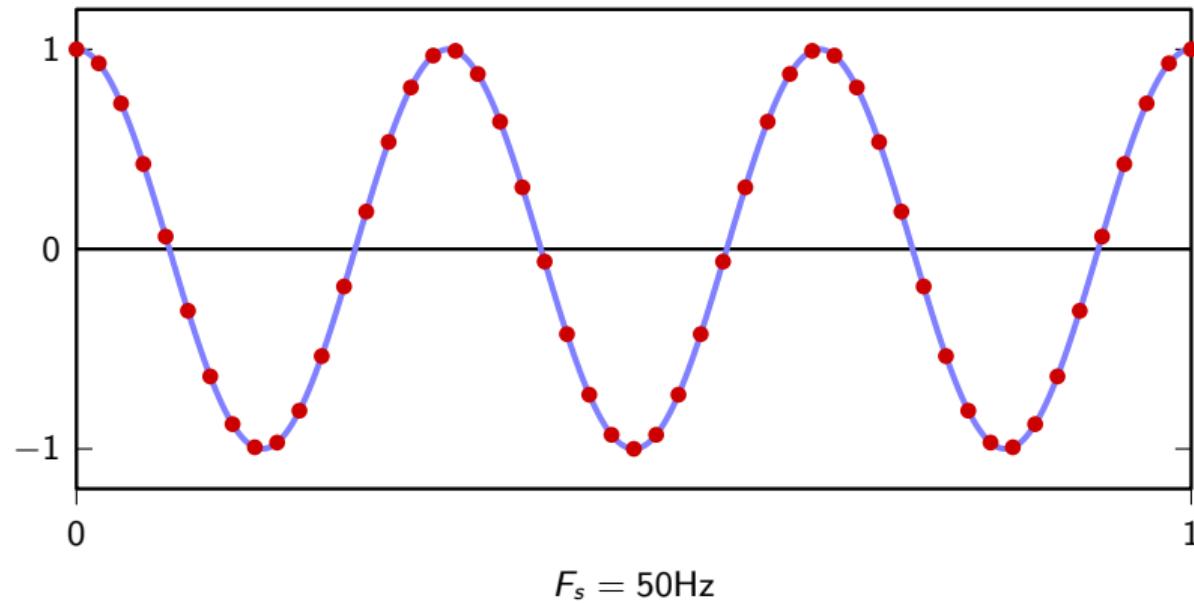
Aliasing: Sampling a Sinusoid

$$x(t) = \cos(6\pi t) \quad (f_0 = 3\text{Hz})$$



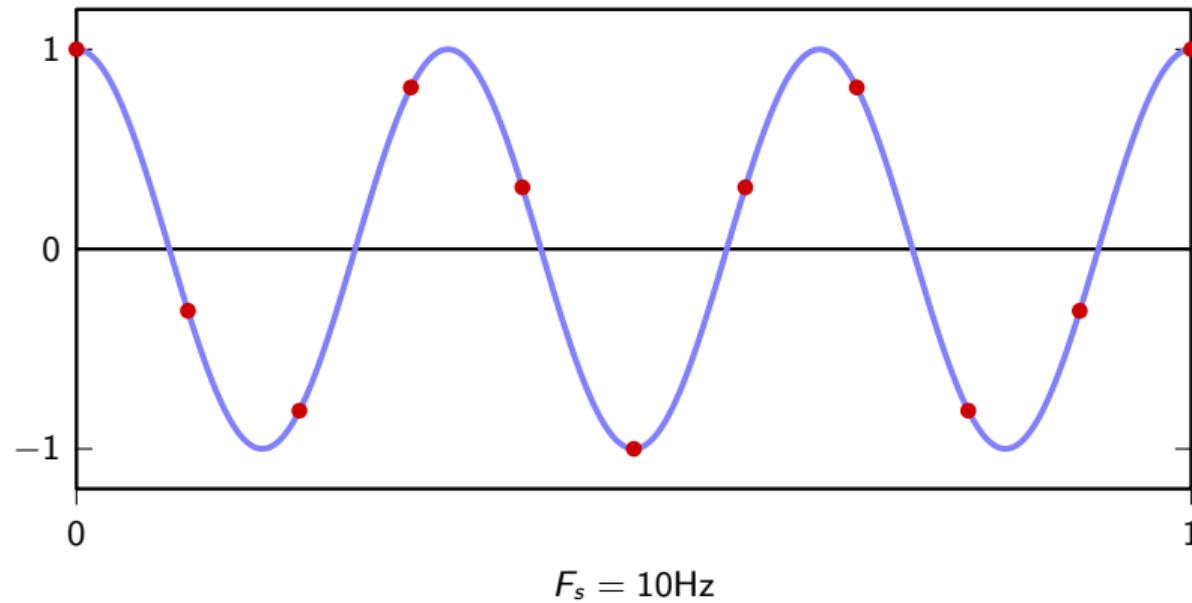
Aliasing: Sampling a Sinusoid

$$x(t) = \cos(6\pi t) \quad (f_0 = 3\text{Hz})$$



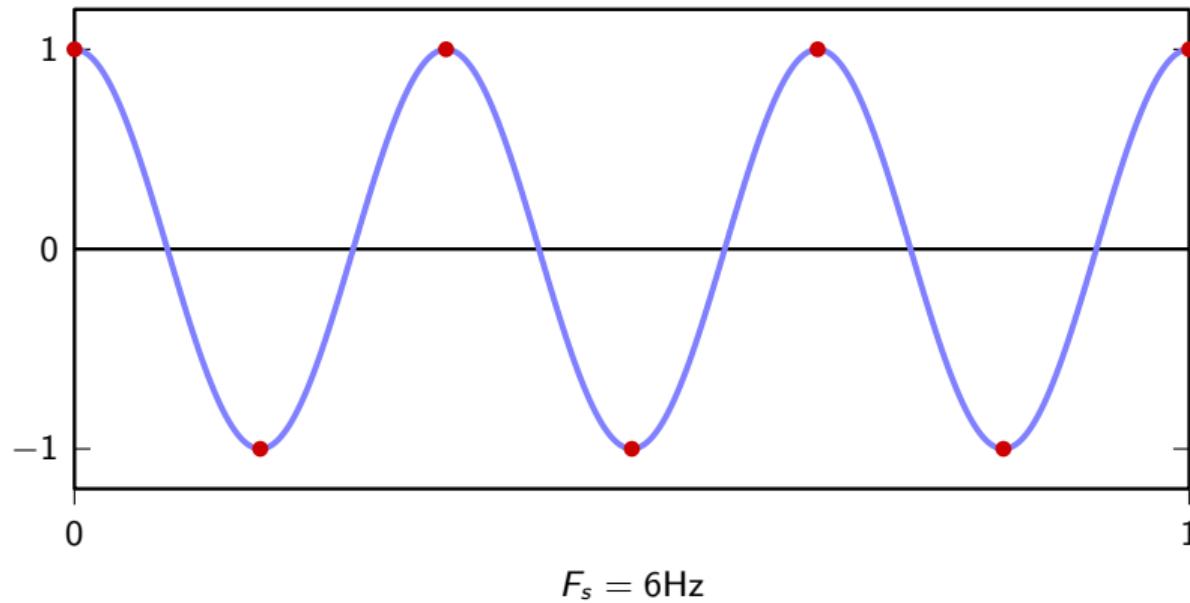
Aliasing: Sampling a Sinusoid

$$x(t) = \cos(6\pi t) \quad (f_0 = 3\text{Hz})$$



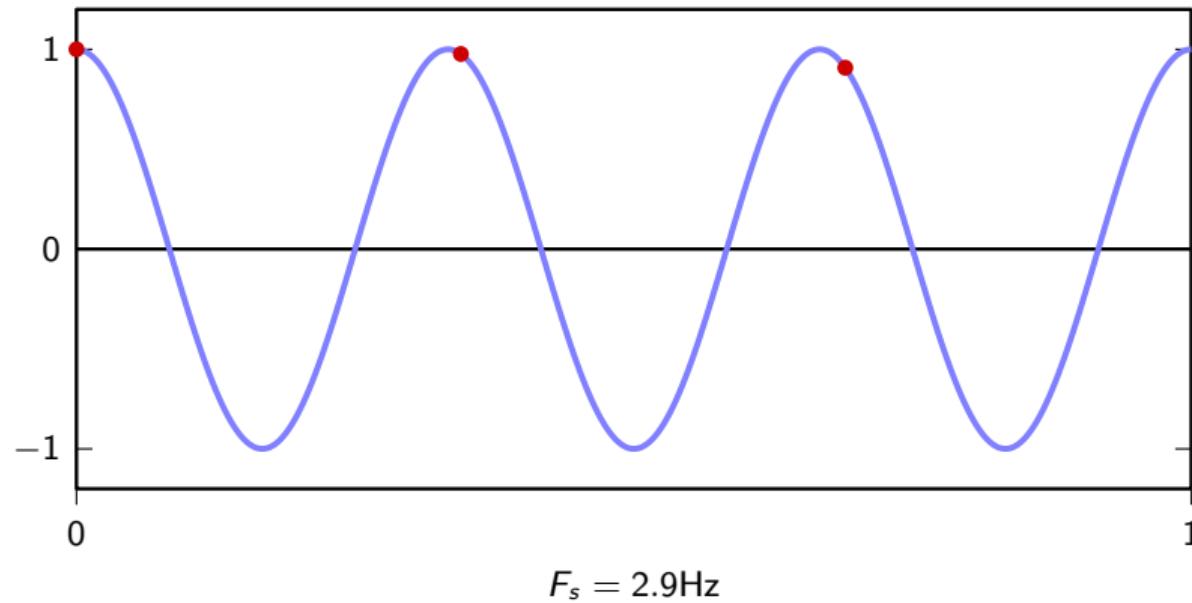
Aliasing: Sampling a Sinusoid

$$x(t) = \cos(6\pi t) \quad (f_0 = 3\text{Hz})$$



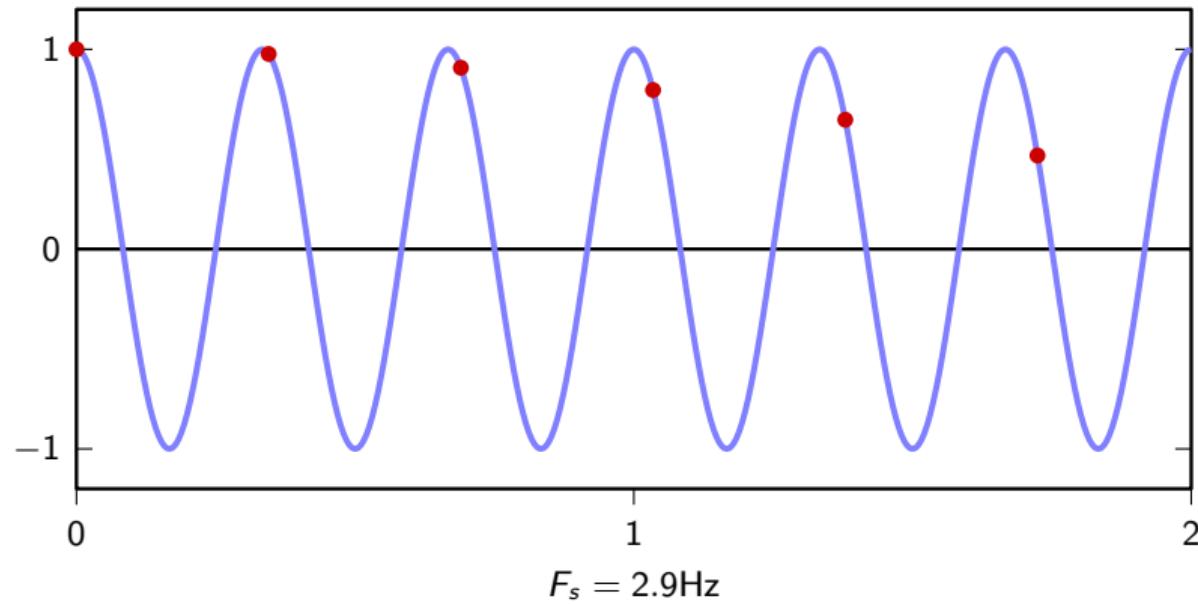
Aliasing: Sampling a Sinusoid

$$x(t) = \cos(6\pi t) \quad (f_0 = 3\text{Hz})$$



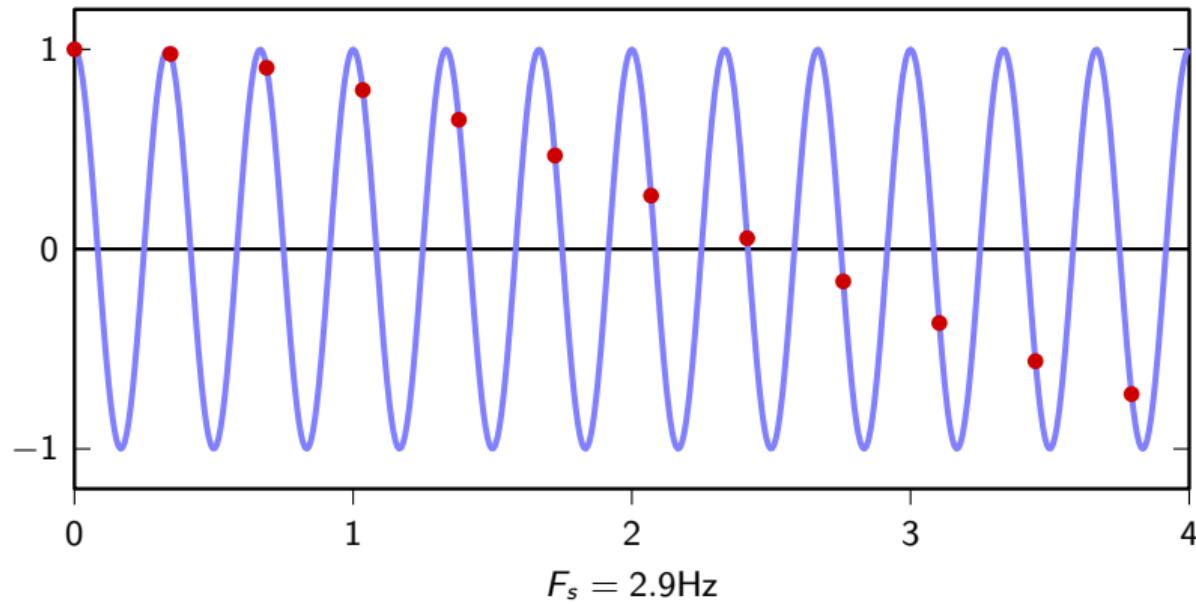
Aliasing: Sampling a Sinusoid

$$x(t) = \cos(6\pi t) \quad (f_0 = 3\text{Hz})$$



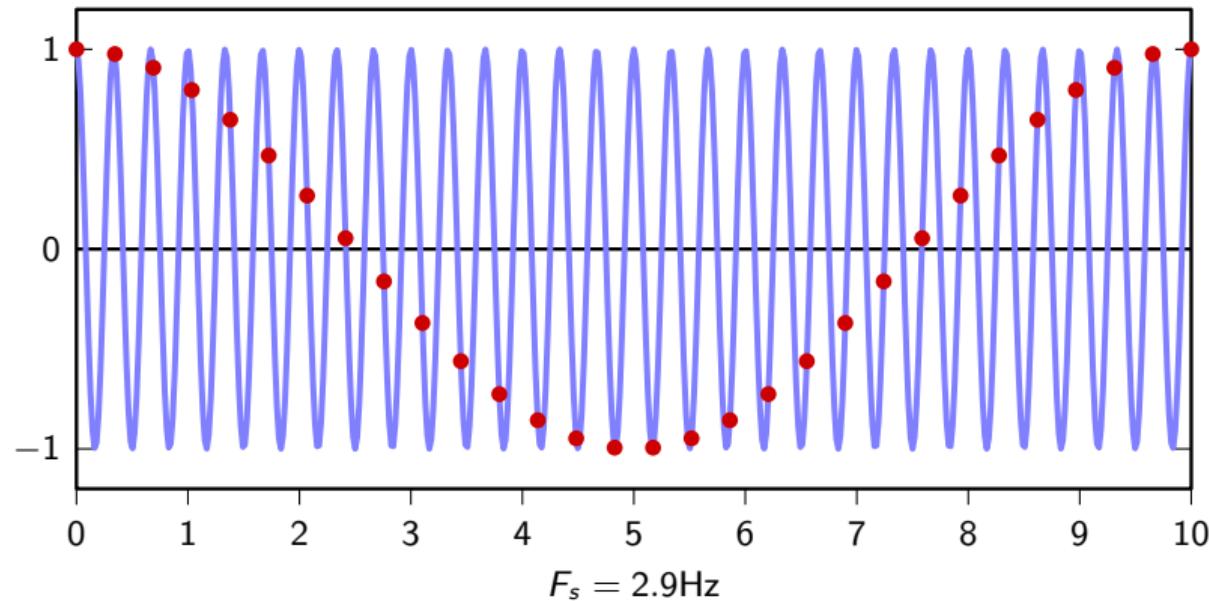
Aliasing: Sampling a Sinusoid

$$x(t) = \cos(6\pi t) \quad (f_0 = 3\text{Hz})$$



Aliasing: Sampling a Sinusoid

$$x(t) = \cos(6\pi t) \quad (f_0 = 3\text{Hz})$$



Raw-sampling an arbitrary signal

$$x_c(t) \xrightarrow{T_s} x[n] = x_c(nT_s)$$

$$T_s$$

Raw-sampling an arbitrary signal

$$X_c(f) \xrightarrow{T_s} X(e^{j\omega}) = ?$$

T_s

Key idea

- ▶ pick T_s (and set $F_s = 1/T_s$)
- ▶ pick $f_0 < F_s/2$

$$e^{j2\pi f_0 t} \xrightarrow{T_s} e^{j2\pi f_0 T_s n}$$

Key idea

- ▶ pick T_s (and set $F_s = 1/T_s$)
- ▶ pick $f_0 < F_s/2$

$$e^{j2\pi(f_0+F_s)t} \xrightarrow{T_s} e^{j2\pi(f_0+F_s)T_s n}$$

Key idea

- ▶ pick T_s (and set $F_s = 1/T_s$)
- ▶ pick $f_0 < F_s/2$

$$e^{j2\pi(f_0+F_s)t} \xrightarrow{T_s} e^{j(2\pi f_0 T_s n + 2\pi F_s T_s n)}$$

Key idea

- ▶ pick T_s (and set $F_s = 1/T_s$)
- ▶ pick $f_0 < F_s/2$

$$e^{j2\pi(f_0+F_s)t} \xrightarrow{T_s} e^{j(2\pi f_0 T_s n + 2\pi n)}$$

T_s

Key idea

- ▶ pick T_s (and set $F_s = 1/T_s$)
- ▶ pick $f_0 < F_s/2$

$$e^{j2\pi(f_0+F_s)t} \xrightarrow{T_s} e^{j2\pi f_0 T_s n}$$

Key idea

- ▶ pick T_s (and set $F_s = 1/T_s$)
- ▶ pick $f_0 < F_s/2$

$$Ae^{j2\pi f_0 t} + Be^{j2\pi(f_0+F_s)t} \xrightarrow{T_s} (A+B)e^{j2\pi f_0 T_s n}$$

T_s

Spectrum of raw-sampled signals

outline: start with the inverse Fourier Transform

$$x[n] = x_c(nT_s) = \int_{-\infty}^{\infty} X_c(f) e^{j2\pi f T_s n} df$$

and manipulate the integral until it looks like

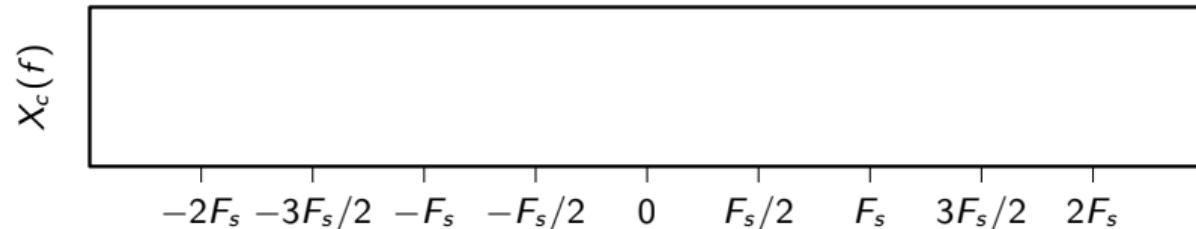
$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(\omega) e^{j\omega n} d\omega$$

Spectrum of raw-sampled signals

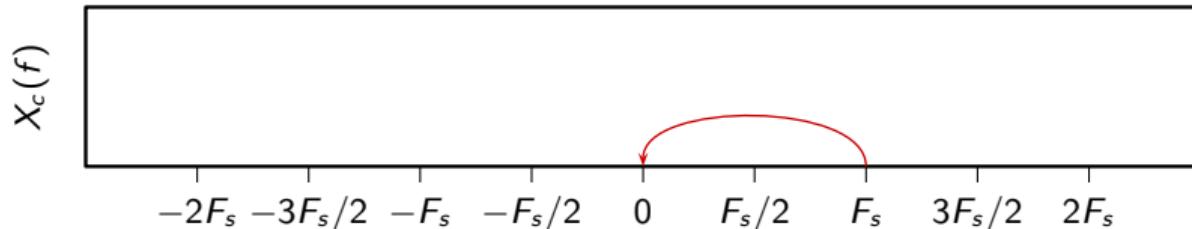
frequencies F_s Hz apart will be aliased, so split the integration interval

$$\begin{aligned}x[n] &= \int_{-\infty}^{\infty} X_c(f) e^{j2\pi f T_s n} df \\&= \sum_{k=-\infty}^{\infty} \int_{kF_s - \frac{F_s}{2}}^{kF_s + \frac{F_s}{2}} X_c(f) e^{j2\pi f T_s n} df\end{aligned}$$

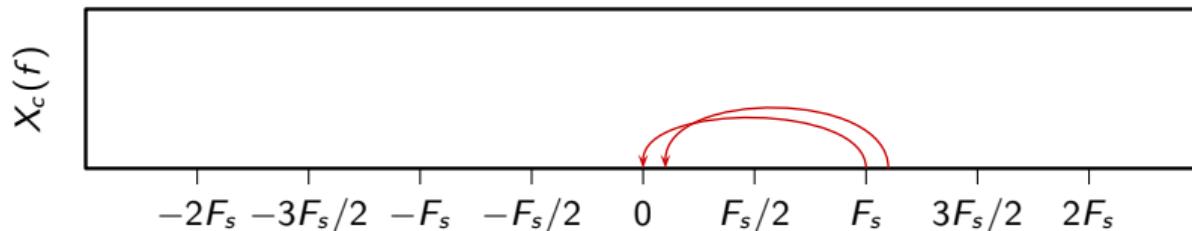
Spectrum of raw-sampled signals



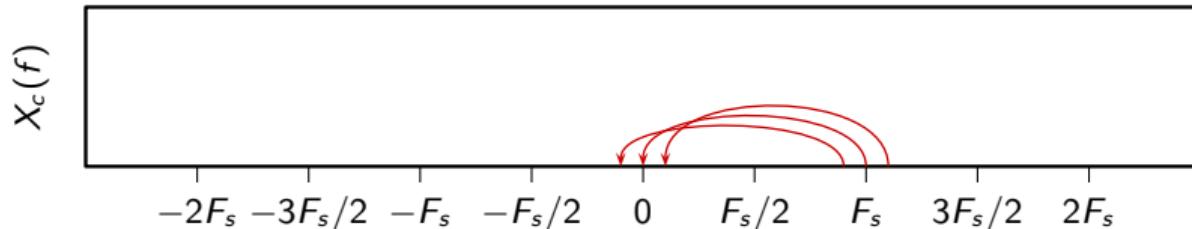
Spectrum of raw-sampled signals



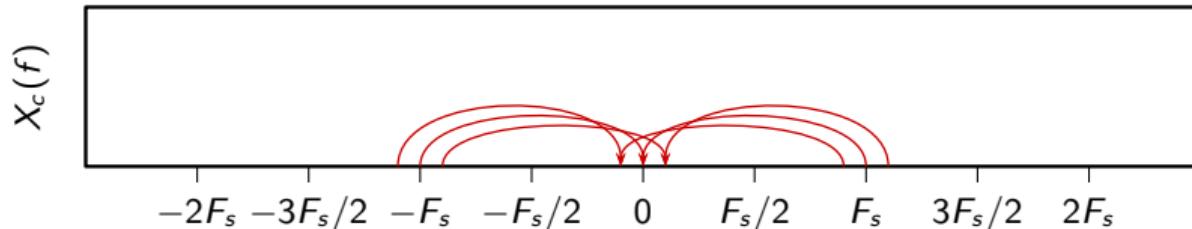
Spectrum of raw-sampled signals



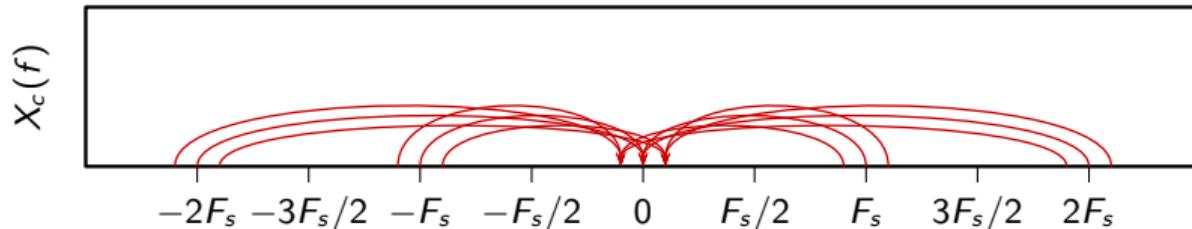
Spectrum of raw-sampled signals



Spectrum of raw-sampled signals



Spectrum of raw-sampled signals



Spectrum of raw-sampled signals

$$x[n] = \sum_{k=-\infty}^{\infty} \int_{kF_s - \frac{F_s}{2}}^{kF_s + \frac{F_s}{2}} X_c(f) e^{j2\pi f T_s n} df$$

operate the change of variable $f \rightarrow f + kF_s$:

- ▶ integration limits become $\pm F_s/2$
- ▶ $e^{j2\pi(f-kF_s)T_s n} = e^{j2\pi f T_s n}$

Spectrum of raw-sampled signals

$$x[n] = \sum_{k=-\infty}^{\infty} \int_{kF_s - \frac{F_s}{2}}^{kF_s + \frac{F_s}{2}} X_c(f) e^{j2\pi f T_s n} df$$

$$= \sum_{k=-\infty}^{\infty} \int_{-F_s/2}^{F_s/2} X_c(f - kF_s) e^{j2\pi f T_s n} df$$

$$= \int_{-F_s/2}^{F_s/2} \left[\sum_{k=-\infty}^{\infty} X_c(f - kF_s) \right] e^{j2\pi f T_s n} df$$

Spectrum of raw-sampled signals

F_s -periodization of the spectrum; define:

$$\tilde{X}_c(f) = \sum_{k=-\infty}^{\infty} X_c(f - kF_s)$$

then:

$$x[n] = \int_{-F_s/2}^{F_s/2} \tilde{X}_c(f) e^{j2\pi f T_s n} df$$

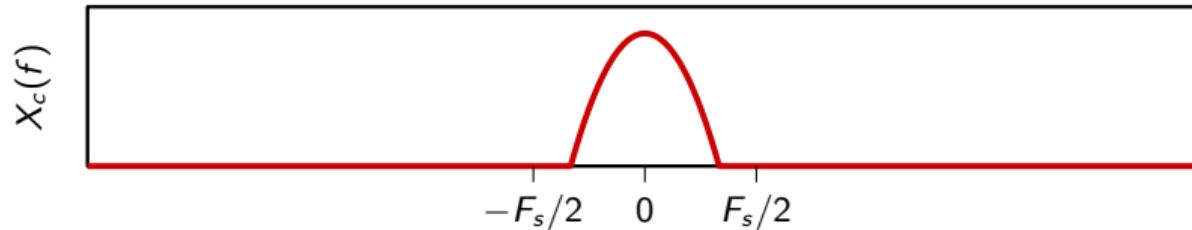
Spectrum of raw-sampled signals

set $\omega = 2\pi f T_s$, so that $f = \frac{\omega}{2\pi} F_s$

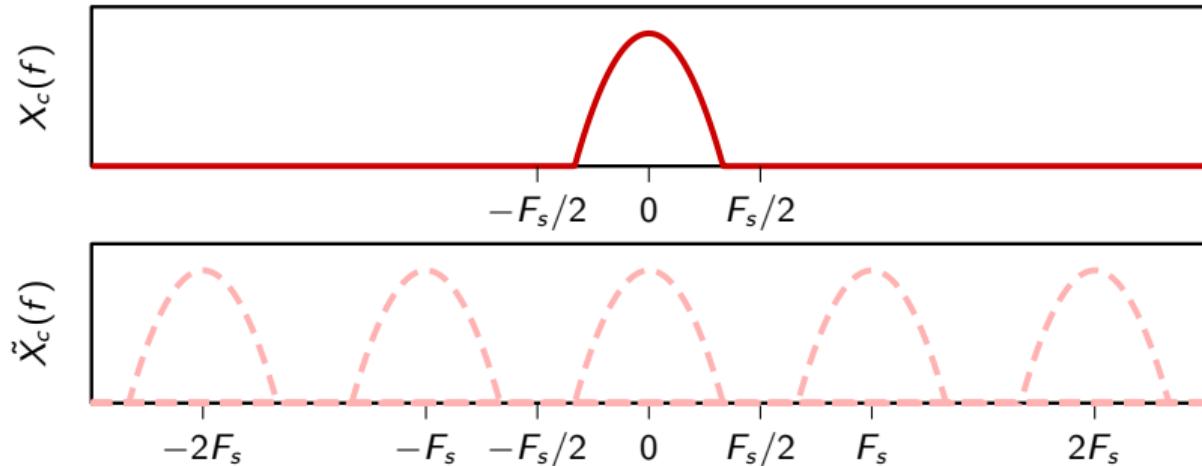
$$\begin{aligned}x[n] &= \frac{1}{2\pi} \int_{-\pi}^{\pi} F_s \tilde{X}_c \left(\frac{\omega}{2\pi} F_s \right) e^{j\omega n} d\omega \\&= \text{IDTFT} \left\{ F_s \tilde{X}_c \left(\frac{\omega}{2\pi} F_s \right) \right\}\end{aligned}$$

$$X(e^{j\omega}) = F_s \tilde{X}_c \left(\frac{\omega}{2\pi} F_s \right) = F_s \sum_{k=-\infty}^{\infty} X_c \left(\frac{\omega}{2\pi} F_s - kF_s \right)$$

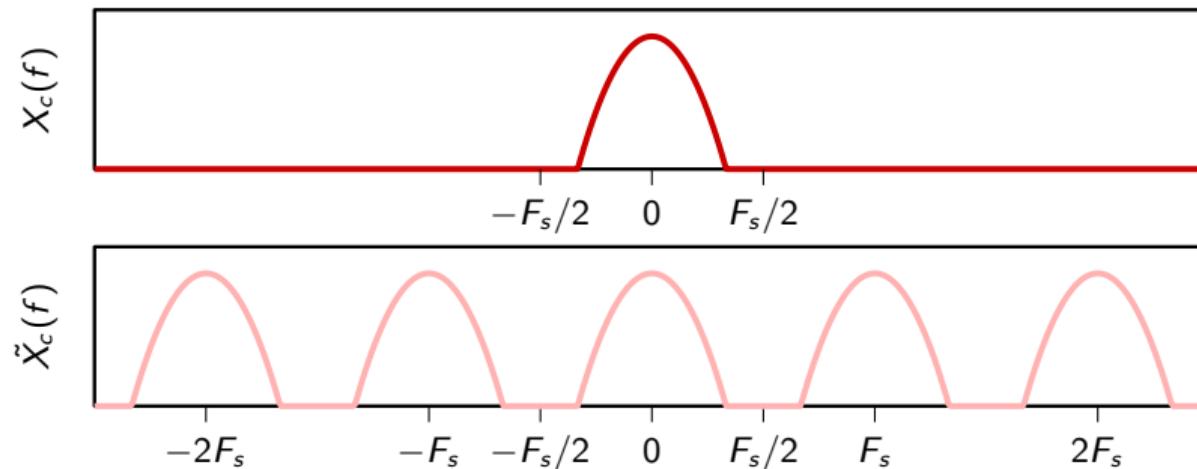
Example: signal bandlimited to f_0 and $F_s > 2f_0$



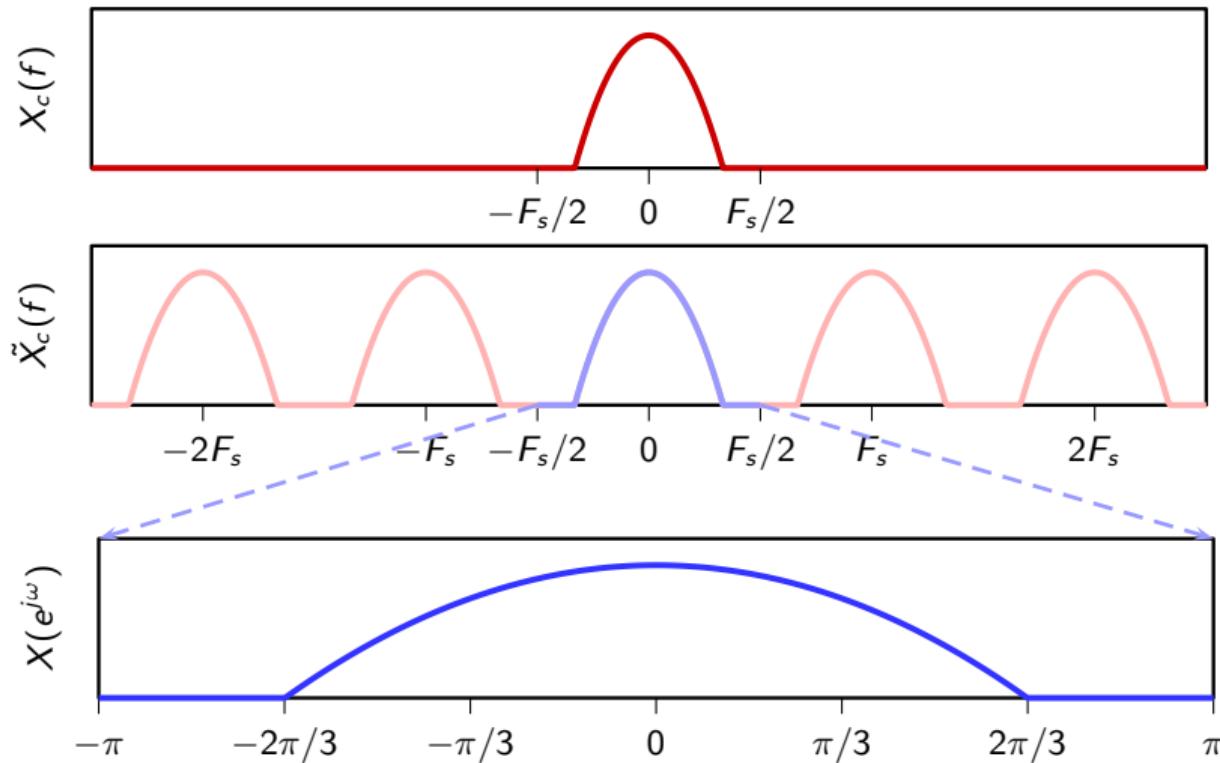
Example: signal bandlimited to f_0 and $F_s > 2f_0$



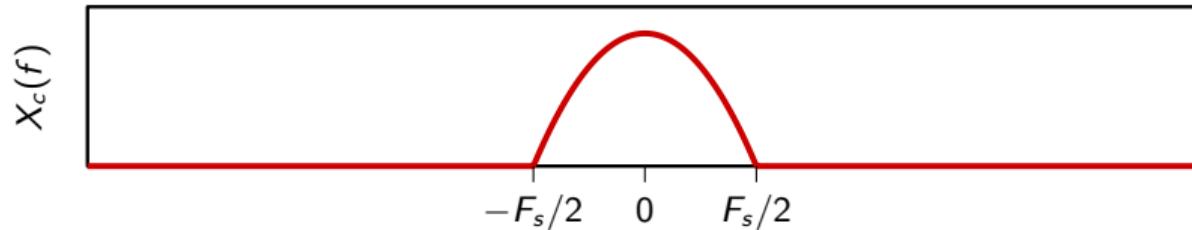
Example: signal bandlimited to f_0 and $F_s > 2f_0$



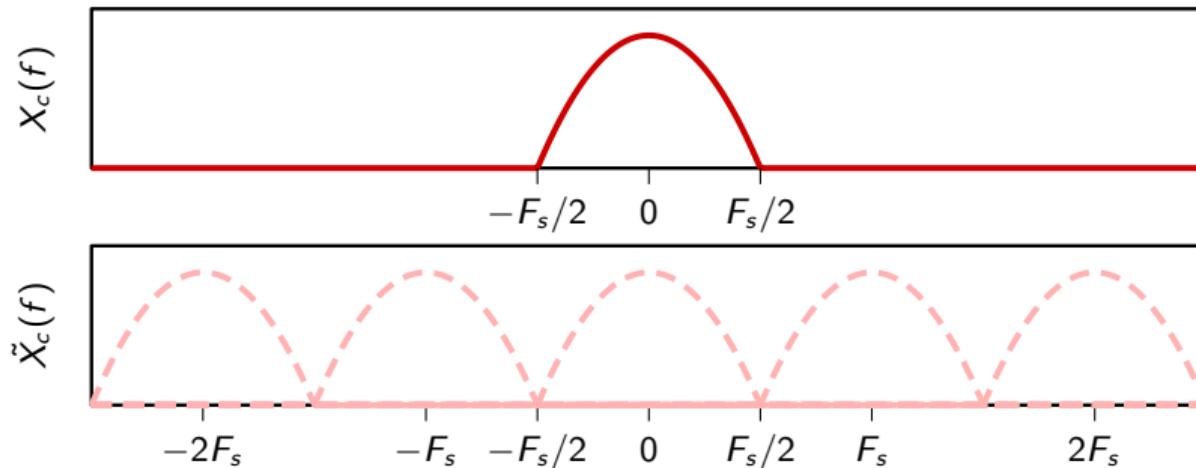
Example: signal bandlimited to f_0 and $F_s > 2f_0$



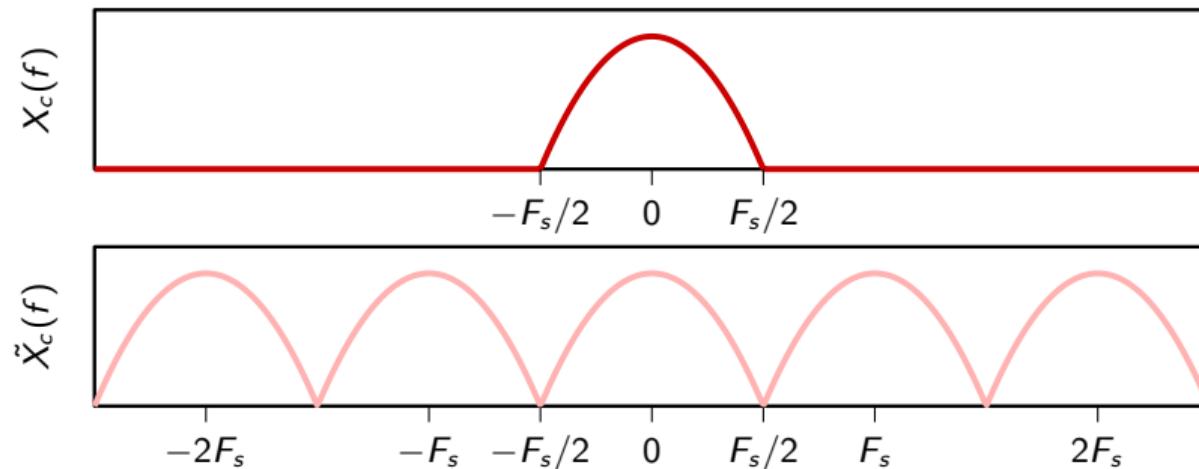
Example: signal bandlimited to f_0 and $F_s = 2f_0$



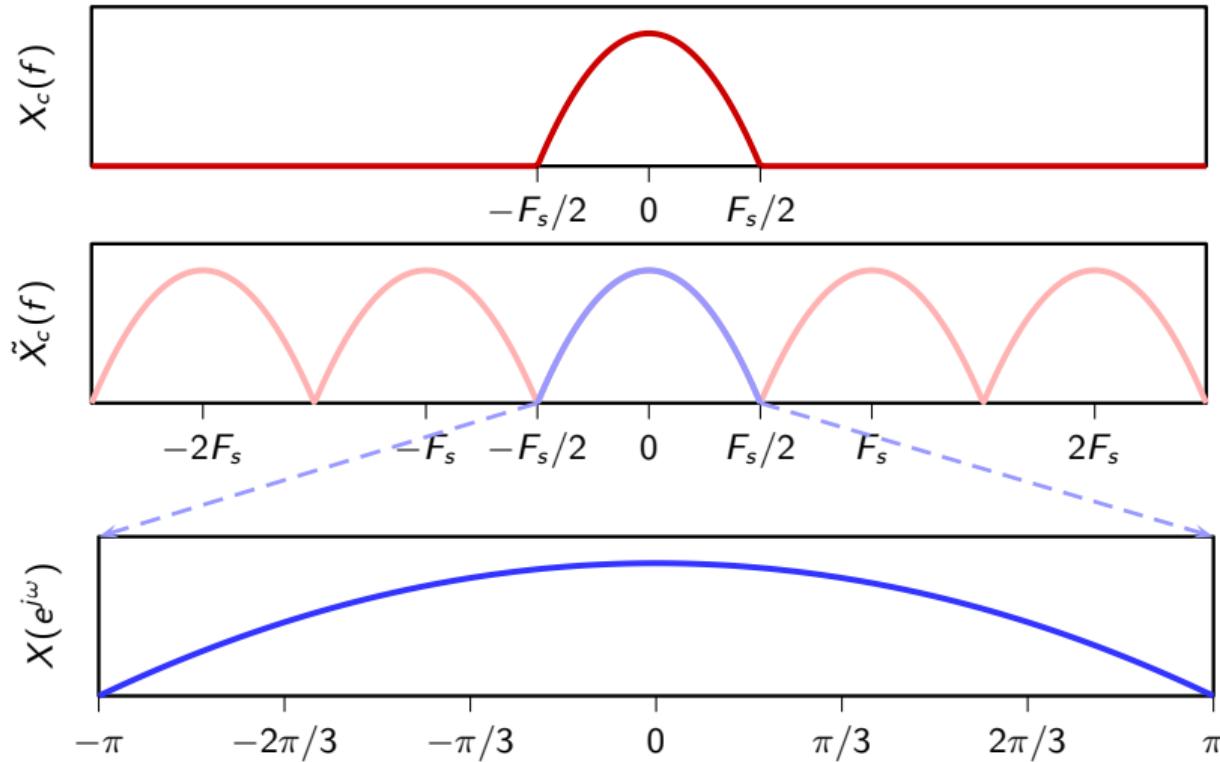
Example: signal bandlimited to f_0 and $F_s = 2f_0$



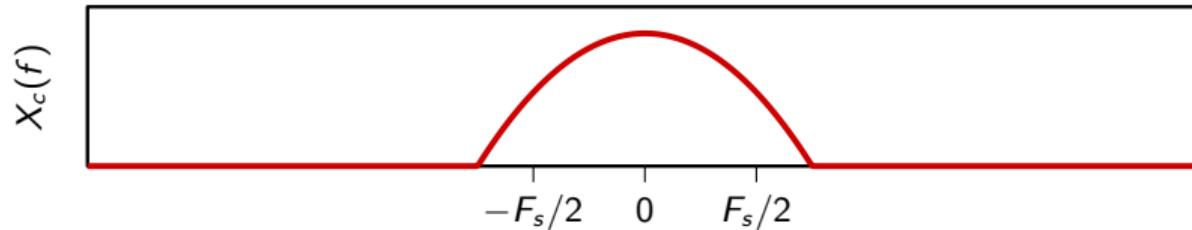
Example: signal bandlimited to f_0 and $F_s = 2f_0$



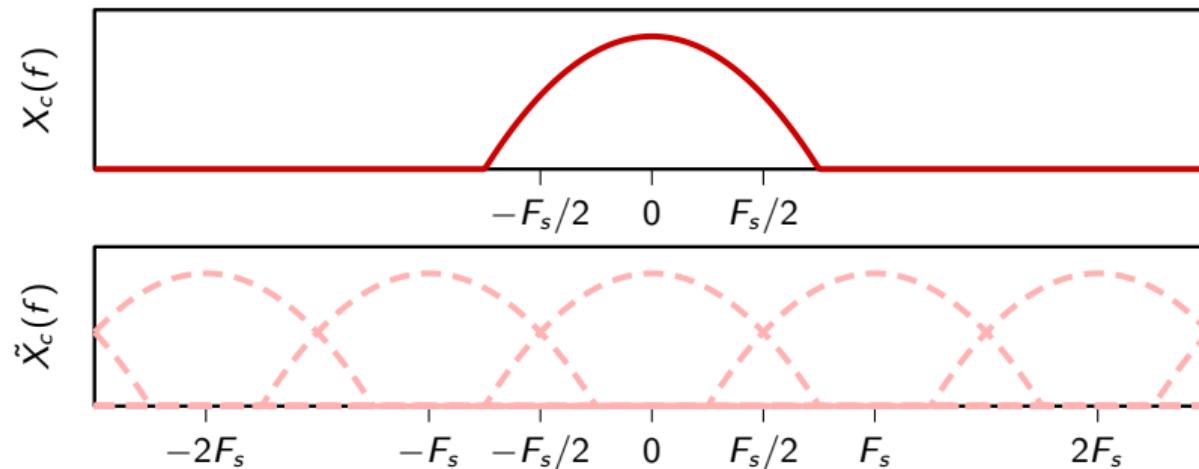
Example: signal bandlimited to f_0 and $F_s = 2f_0$



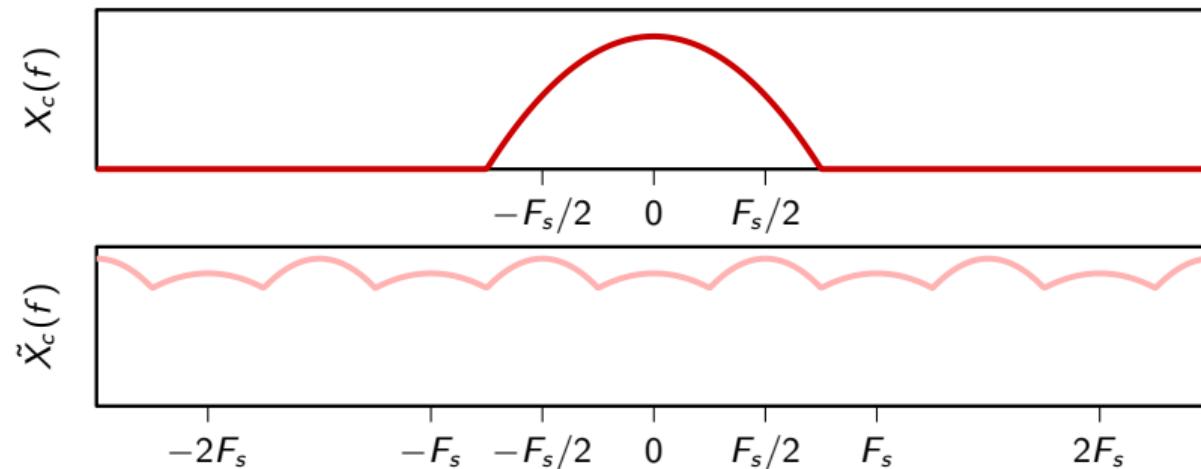
Example: signal bandlimited to f_0 and $F_s < 2f_0$



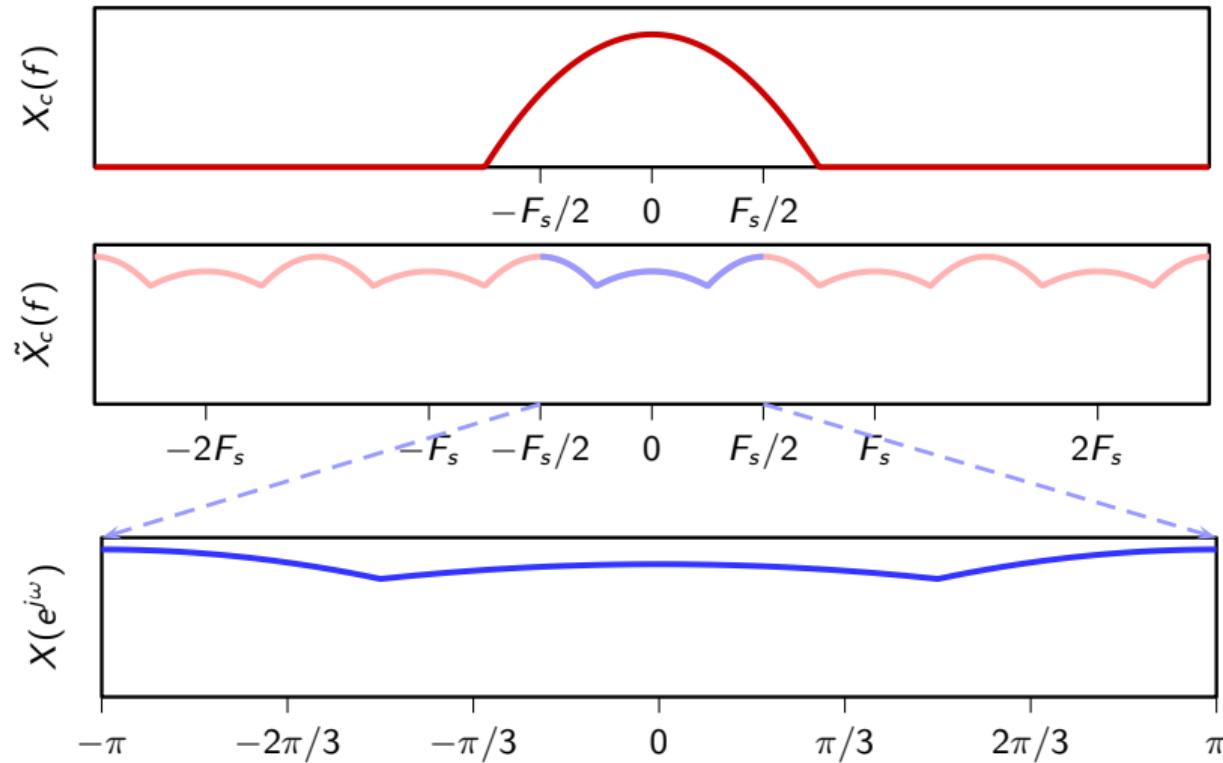
Example: signal bandlimited to f_0 and $F_s < 2f_0$



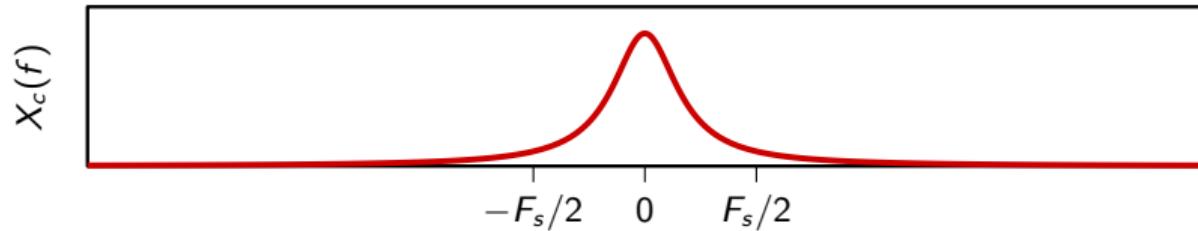
Example: signal bandlimited to f_0 and $F_s < 2f_0$



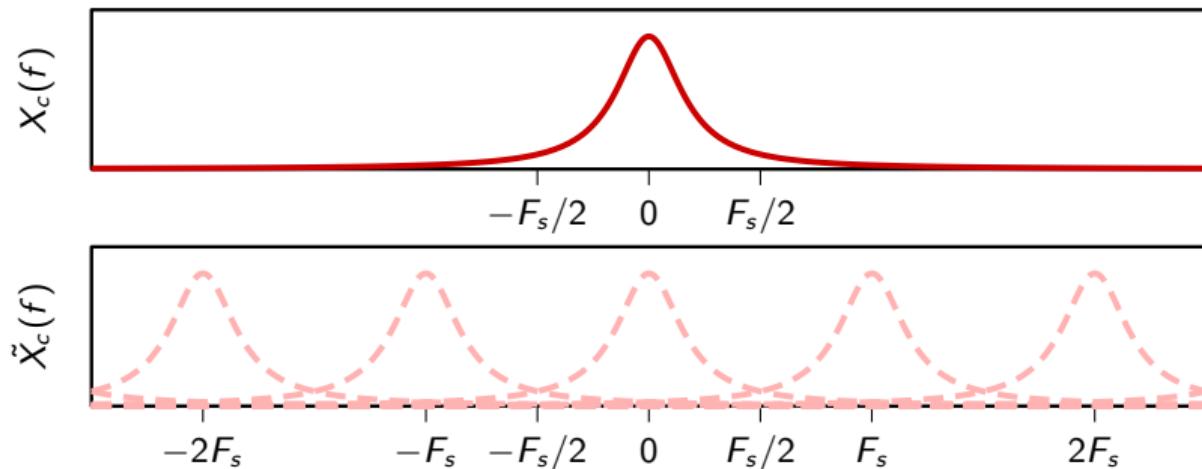
Example: signal bandlimited to f_0 and $F_s < 2f_0$



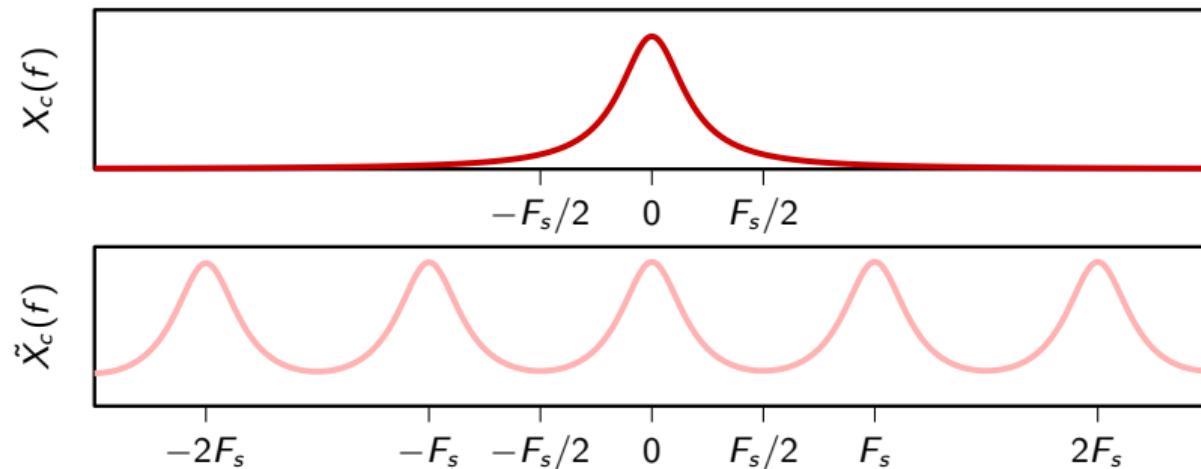
Example: non-bandlimited signal



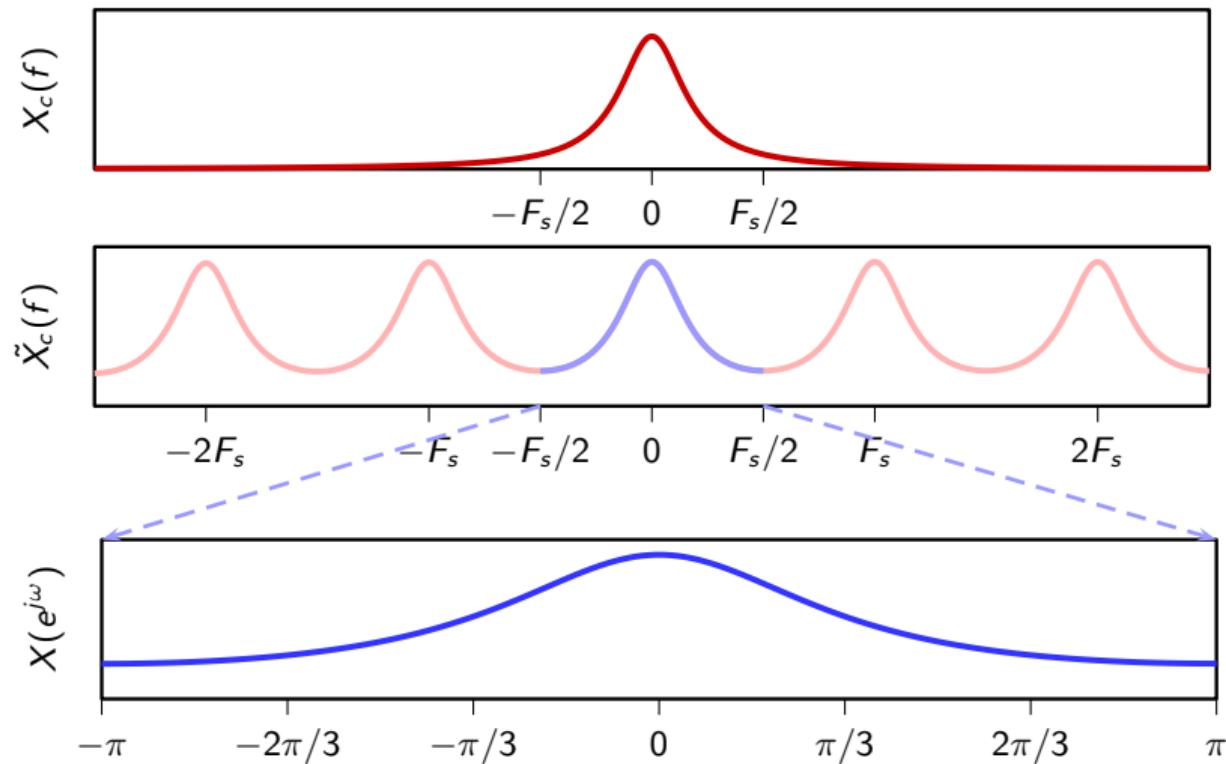
Example: non-bandlimited signal



Example: non-bandlimited signal



Example: non-bandlimited signal



Sampling strategies

given a sampling frequency F_s

- ▶ if the signal is bandlimited to $F_s/2$ or less, raw sampling is fine (i.e. equivalent to sinc sampling up to a scaling factor T_s)
- ▶ if the signal is not bandlimited, two choices:
 - bandlimit via a lowpass filter *in the continuous-time domain* before sampling (i.e. sinc sampling)
 - or, raw sample the signal and incur aliasing
- ▶ aliasing introduces errors we cannot control, so the sensible choice is to bandlimit in continuous time
- ▶ bandlimiting is also optimal wrt least squares approximation!

Sampling strategies

given a sampling frequency F_s

- ▶ if the signal is bandlimited to $F_s/2$ or less, raw sampling is fine (i.e. equivalent to sinc sampling up to a scaling factor T_s)
- ▶ if the signal is not bandlimited, two choices:
 - bandlimit via a lowpass filter *in the continuous-time domain* before sampling (i.e. sinc sampling)
 - or, raw sample the signal and incur aliasing
- ▶ aliasing introduces errors we cannot control, so the sensible choice is to bandlimit in continuous time
- ▶ bandlimiting is also optimal wrt least squares approximation!

Sampling strategies

given a sampling frequency F_s

- ▶ if the signal is bandlimited to $F_s/2$ or less, raw sampling is fine (i.e. equivalent to sinc sampling up to a scaling factor T_s)
- ▶ if the signal is not bandlimited, two choices:
 - bandlimit via a lowpass filter *in the continuous-time domain* before sampling (i.e. sinc sampling)
 - or, raw sample the signal and incur aliasing
- ▶ aliasing introduces errors we cannot control, so the sensible choice is to bandlimit in continuous time
- ▶ bandlimiting is also optimal wrt least squares approximation!

Sampling strategies

given a sampling frequency F_s

- ▶ if the signal is bandlimited to $F_s/2$ or less, raw sampling is fine (i.e. equivalent to sinc sampling up to a scaling factor T_s)
- ▶ if the signal is not bandlimited, two choices:
 - bandlimit via a lowpass filter *in the continuous-time domain* before sampling (i.e. sinc sampling)
 - or, raw sample the signal and incur aliasing
- ▶ aliasing introduces errors we cannot control, so the sensible choice is to bandlimit in continuous time
- ▶ bandlimiting is also optimal wrt least squares approximation!

Sampling strategies

given a sampling frequency F_s

- ▶ if the signal is bandlimited to $F_s/2$ or less, raw sampling is fine (i.e. equivalent to sinc sampling up to a scaling factor T_s)
- ▶ if the signal is not bandlimited, two choices:
 - bandlimit via a lowpass filter *in the continuous-time domain* before sampling (i.e. sinc sampling)
 - or, raw sample the signal and incur aliasing
- ▶ aliasing introduces errors we cannot control, so the sensible choice is to bandlimit in continuous time
- ▶ bandlimiting is also optimal wrt least squares approximation!

Sampling strategies

given a sampling frequency F_s

- ▶ if the signal is bandlimited to $F_s/2$ or less, raw sampling is fine (i.e. equivalent to sinc sampling up to a scaling factor T_s)
- ▶ if the signal is not bandlimited, two choices:
 - bandlimit via a lowpass filter *in the continuous-time domain* before sampling (i.e. sinc sampling)
 - or, raw sample the signal and incur aliasing
- ▶ aliasing introduces errors we cannot control, so the sensible choice is to bandlimit in continuous time
- ▶ bandlimiting is also optimal wrt least squares approximation!

Sinc Sampling and Interpolation

$$\hat{x}[n] = \langle \text{sinc} \left(\frac{t - nT_s}{T_s} \right), x_c(t) \rangle = (\text{sinc}_{T_s} * x_c)(nT_s)$$

Sinc Sampling and Interpolation

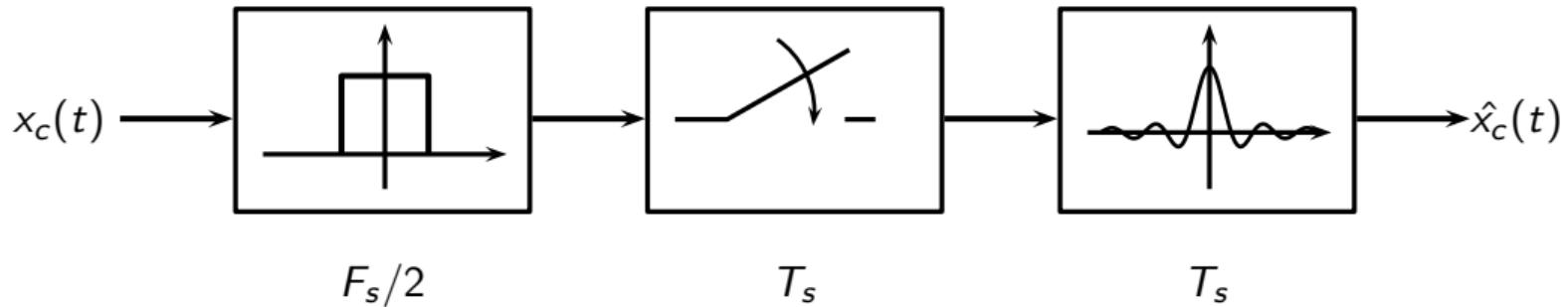
$$\hat{x}[n] = \langle \text{sinc} \left(\frac{t - nT_s}{T_s} \right), x_c(t) \rangle = (\text{sinc}_{T_s} * x_c)(nT_s)$$

$$\hat{x}_c(t) = \sum_n x[n] \text{sinc} \left(\frac{t - nT_s}{T_s} \right)$$

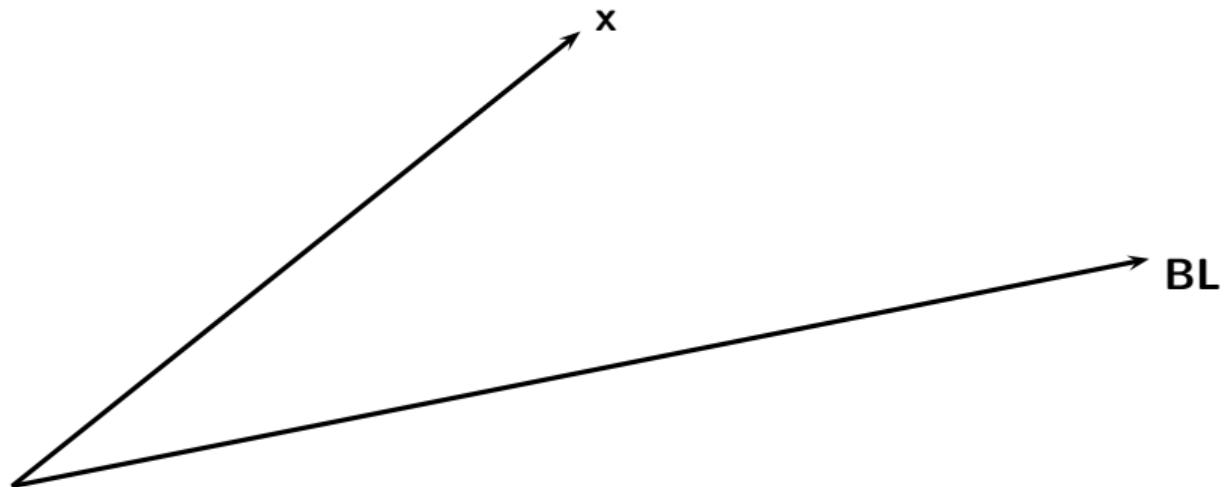
Sinc Sampling and Interpolation

$$\hat{x}[n] = \langle \text{sinc} \left(\frac{t - nT_s}{T_s} \right), x_c(t) \rangle = (\text{sinc}_{T_s} * x_c)(nT_s)$$

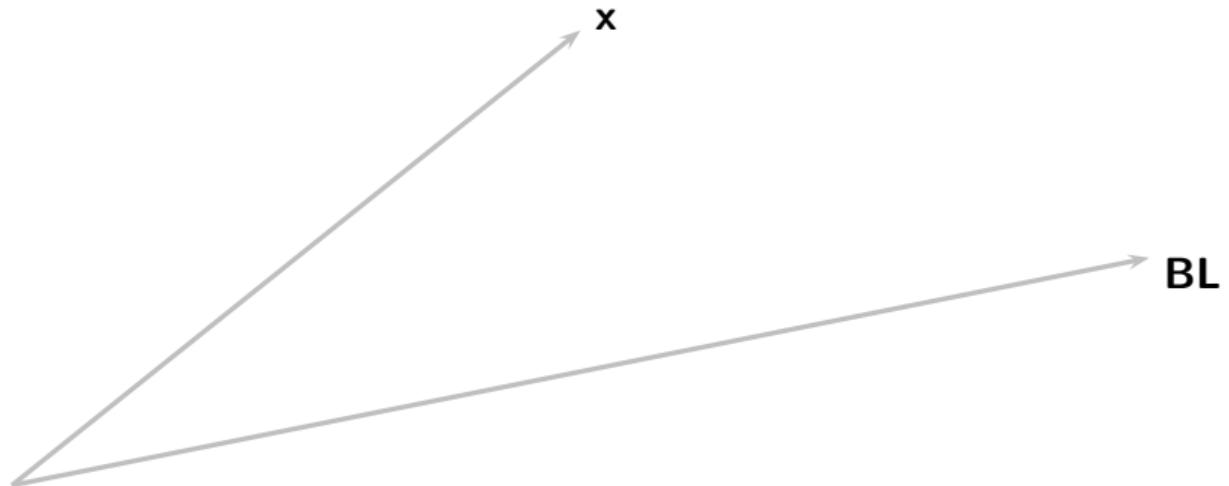
$$\hat{x}_c(t) = \sum_n x[n] \text{sinc} \left(\frac{t - nT_s}{T_s} \right)$$



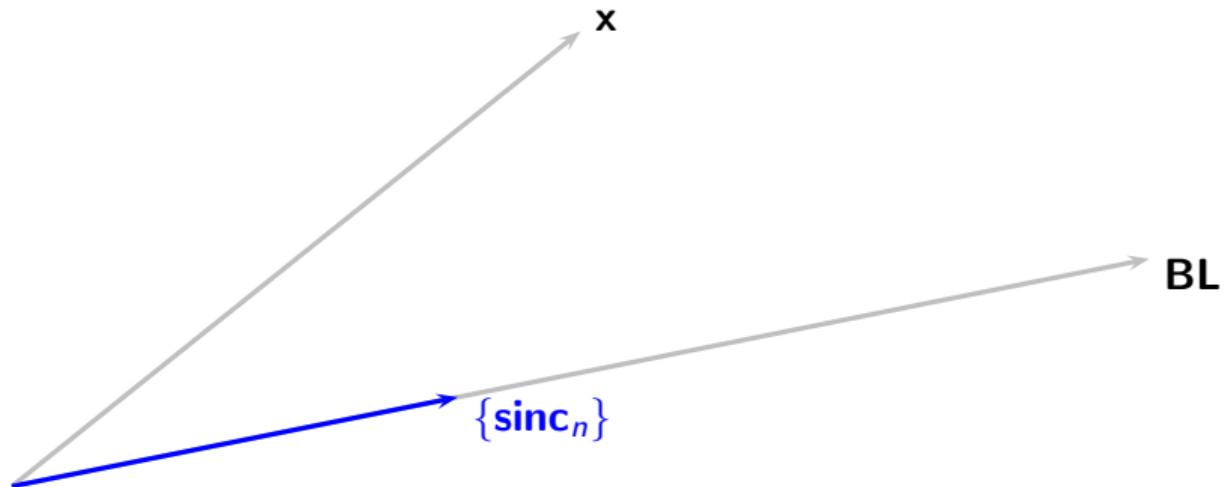
Least squares approximation with sinc sampling and interpolation



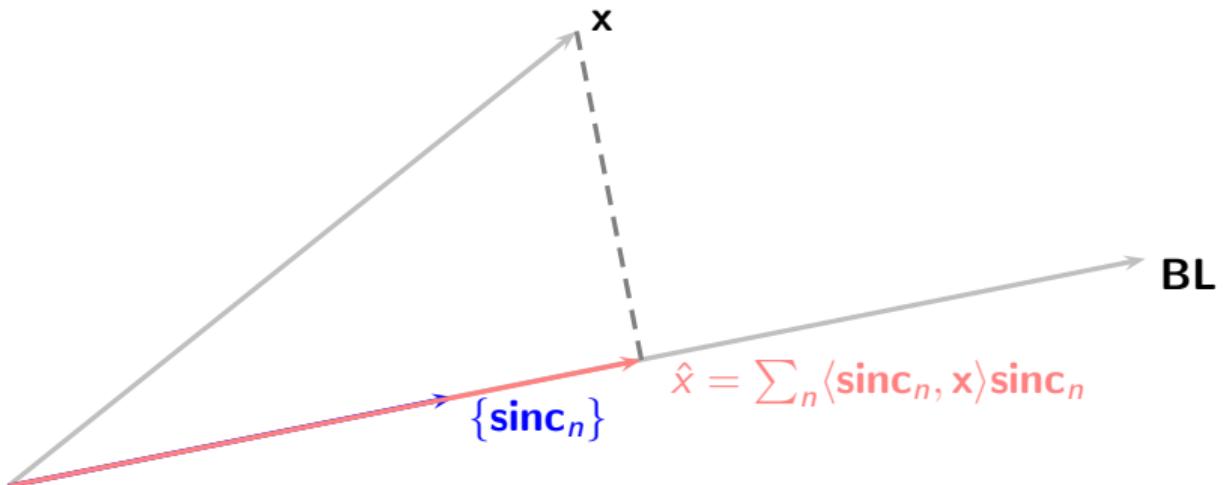
Least squares approximation with sinc sampling and interpolation



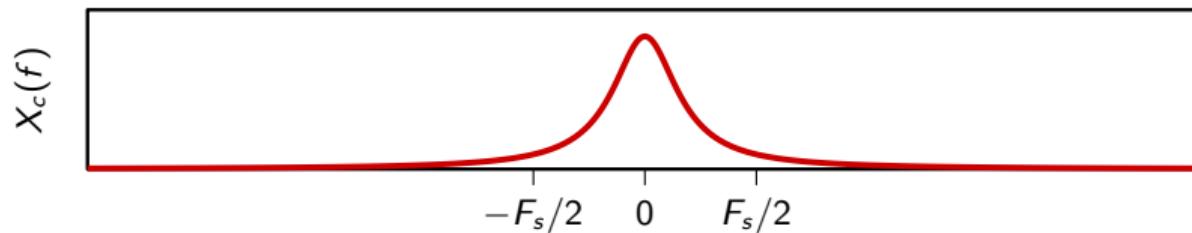
Least squares approximation with sinc sampling and interpolation



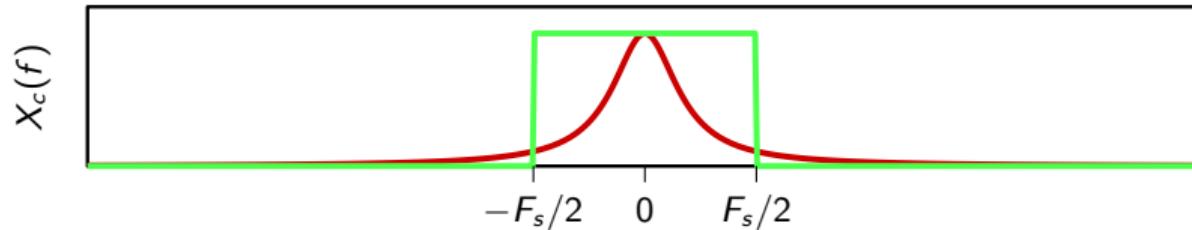
Least squares approximation with sinc sampling and interpolation



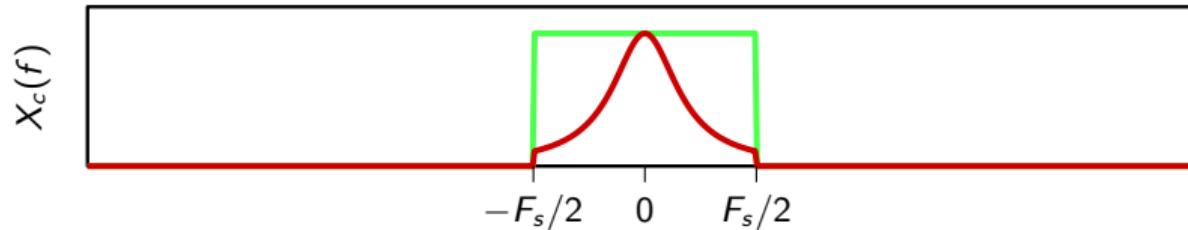
Least squares approximation with sinc sampling and interpolation



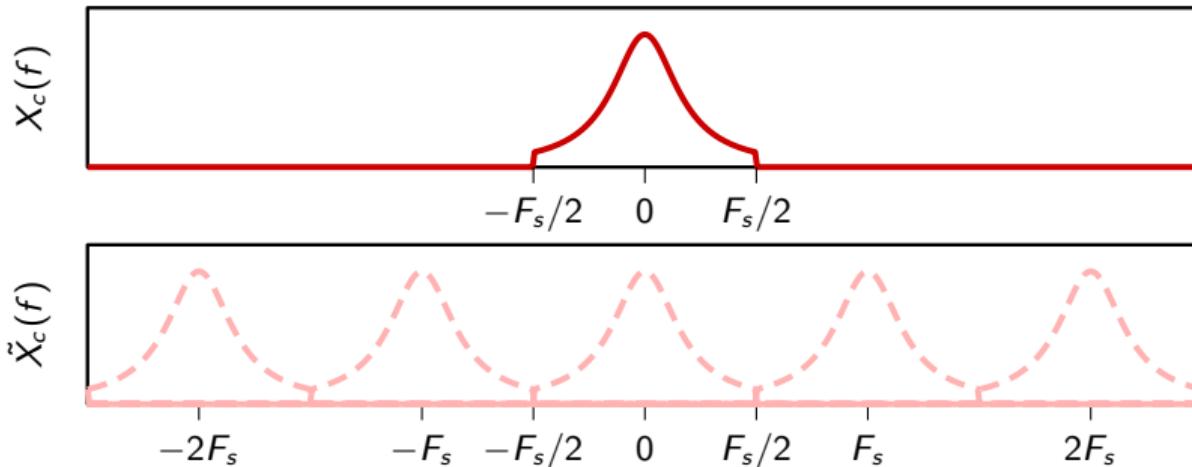
Least squares approximation with sinc sampling and interpolation



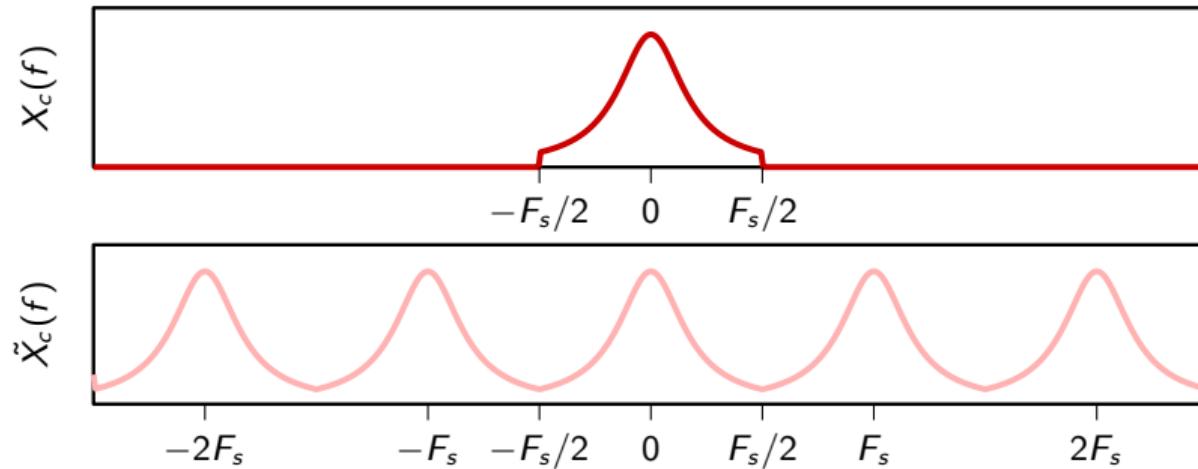
Least squares approximation with sinc sampling and interpolation



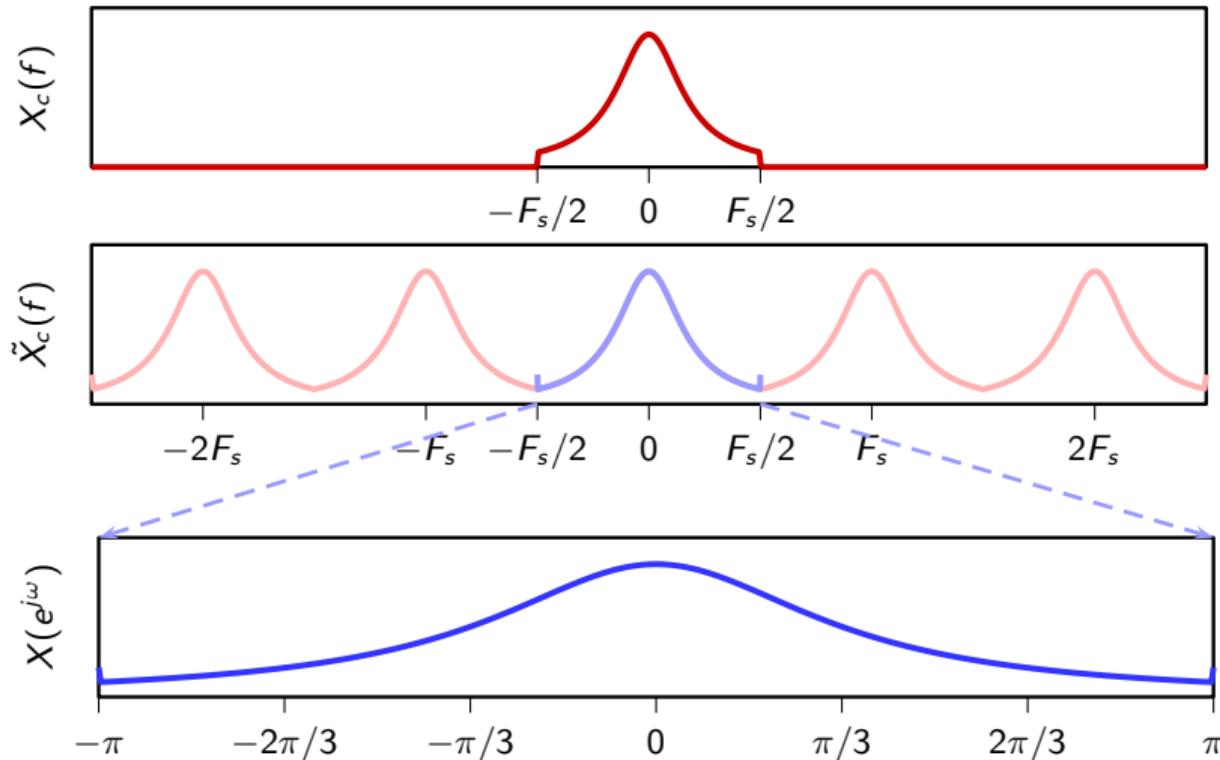
Least squares approximation with sinc sampling and interpolation



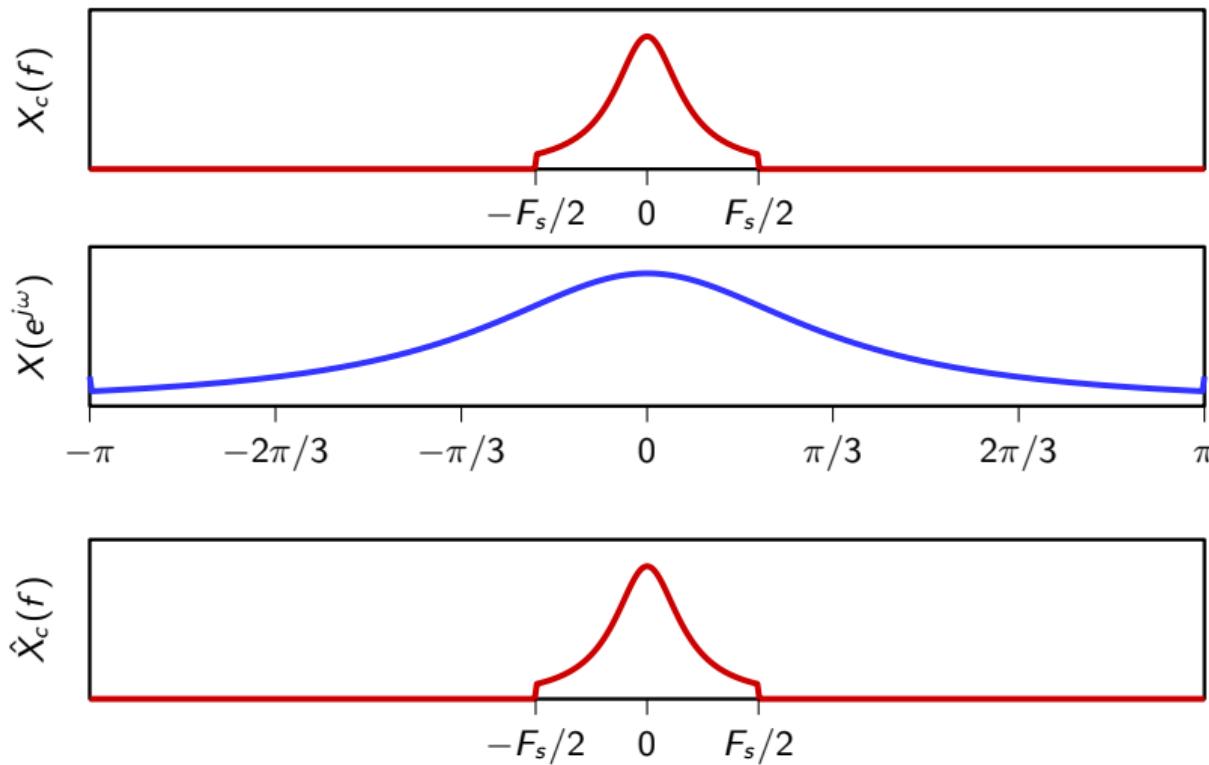
Least squares approximation with sinc sampling and interpolation



Least squares approximation with sinc sampling and interpolation



Least squares approximation with sinc sampling and interpolation



Sinc Sampling and Interpolation

$$\hat{x}[n] = \langle \text{sinc} \left(\frac{t - nT_s}{T_s} \right), x_c(t) \rangle = (\text{sinc}_{T_s} * x_c)(nT_s)$$

$$X(e^{j\omega}) = X_c \left(\frac{\omega}{2\pi} F_s \right) \quad \text{prolonged by periodicity outside of } [-\pi, \pi]$$

Alternatively:

$$X(e^{j\omega}) = X_c \left(\left(\frac{\omega}{\pi} - 2 \left\lfloor \frac{\omega - \pi}{2\pi} \right\rfloor - 2 \right) F_s \right)$$

processing of analog signals

Overview:

- ▶ Impulse invariance
- ▶ Duality
- ▶ Examples

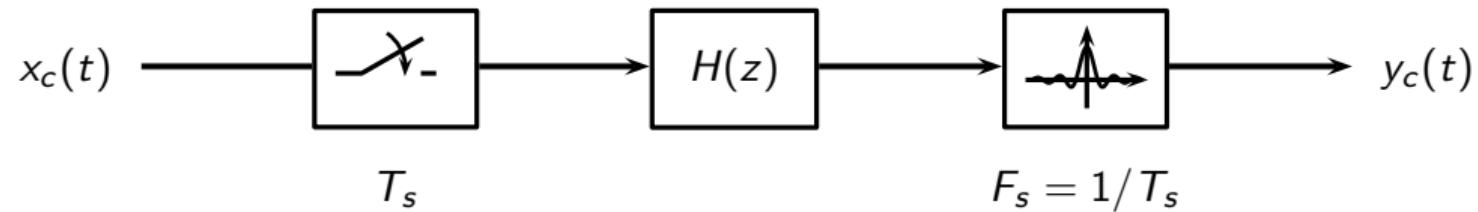
Overview:

- ▶ Impulse invariance
- ▶ Duality
- ▶ Examples

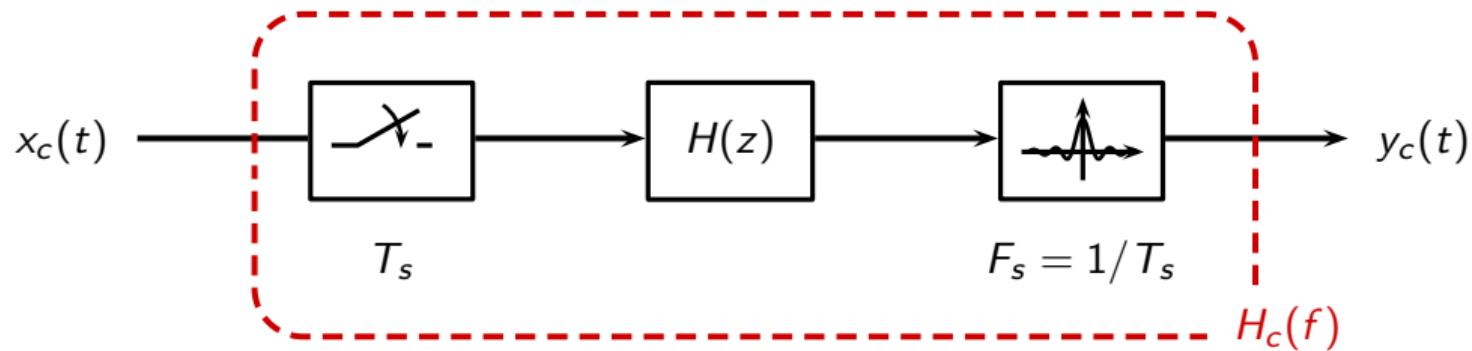
Overview:

- ▶ Impulse invariance
- ▶ Duality
- ▶ Examples

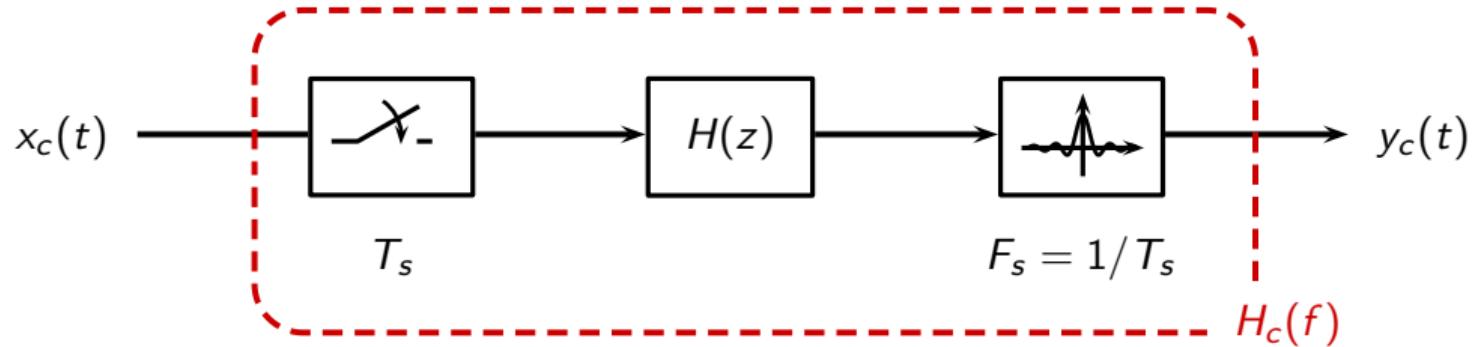
Basic setup



Basic setup

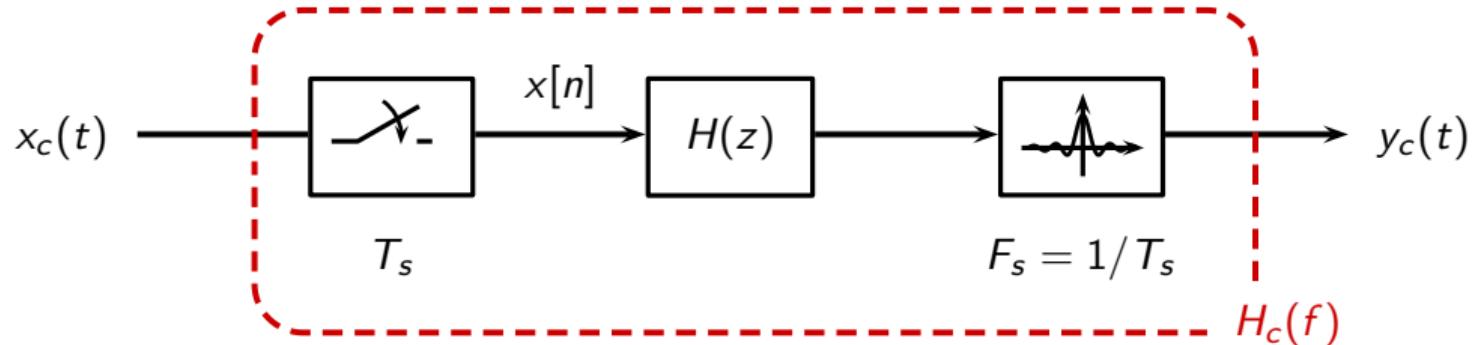


Basic setup



assume $x_c(t)$ is F_s -bandlimited and $T_s = 1/F_s$:

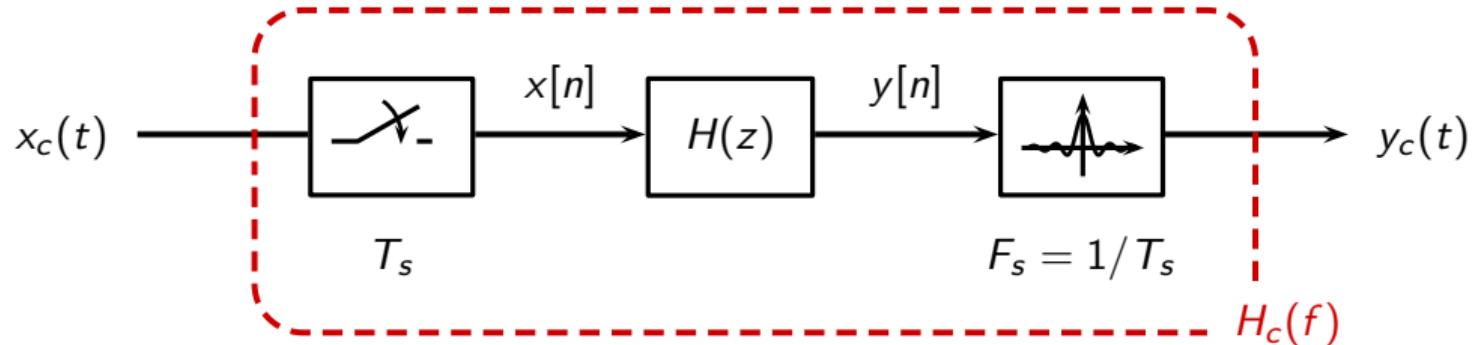
Basic setup



assume $x_c(t)$ is F_s -bandlimited and $T_s = 1/F_s$:

► $X(e^{j\omega}) = F_s X_c \left(\frac{\omega}{2\pi} F_s \right)$

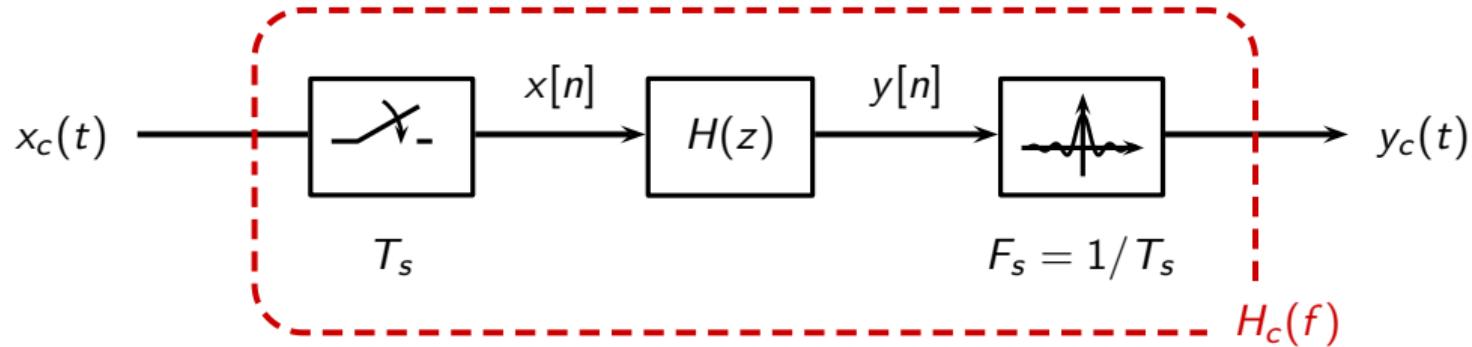
Basic setup



assume $x_c(t)$ is F_s -bandlimited and $T_s = 1/F_s$:

- ▶ $X(e^{j\omega}) = F_s X_c \left(\frac{\omega}{2\pi} F_s \right)$
- ▶ $Y(e^{j\omega}) = X(e^{j\omega}) H(e^{j\omega})$

Basic setup



assume $x_c(t)$ is F_s -bandlimited and $T_s = 1/F_s$:

- ▶ $X(e^{j\omega}) = F_s X_c \left(\frac{\omega}{2\pi} F_s \right)$
- ▶ $Y(e^{j\omega}) = X(e^{j\omega}) H(e^{j\omega})$
- ▶ $Y_c(f) = (1/F_s) Y(e^{j2\pi f/F_s})$

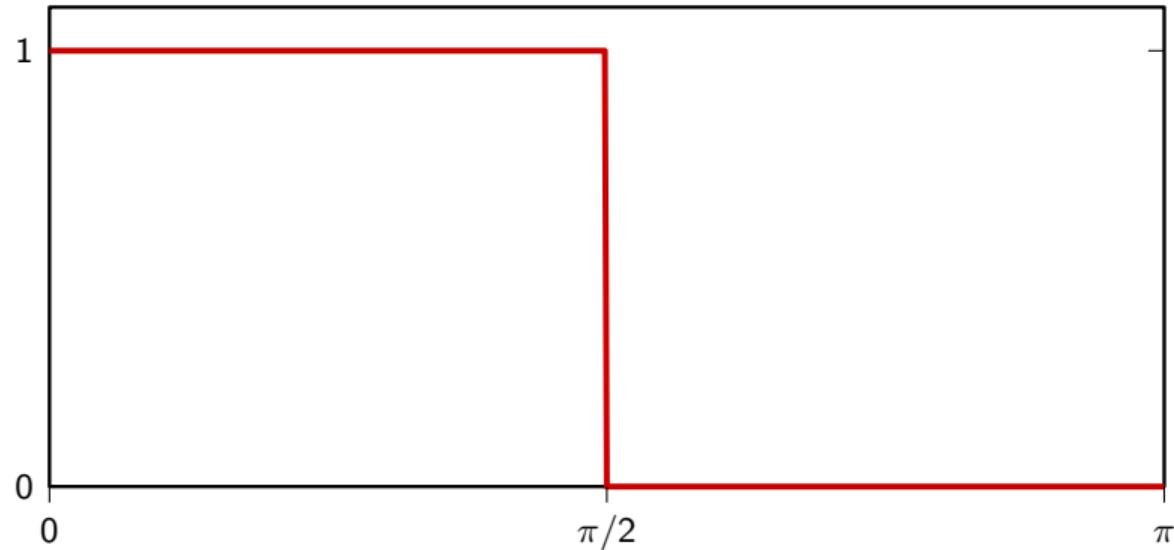
Impulse invariance

$$Y_c(f) = X_c(f) H(e^{j2\pi f/F_s})$$

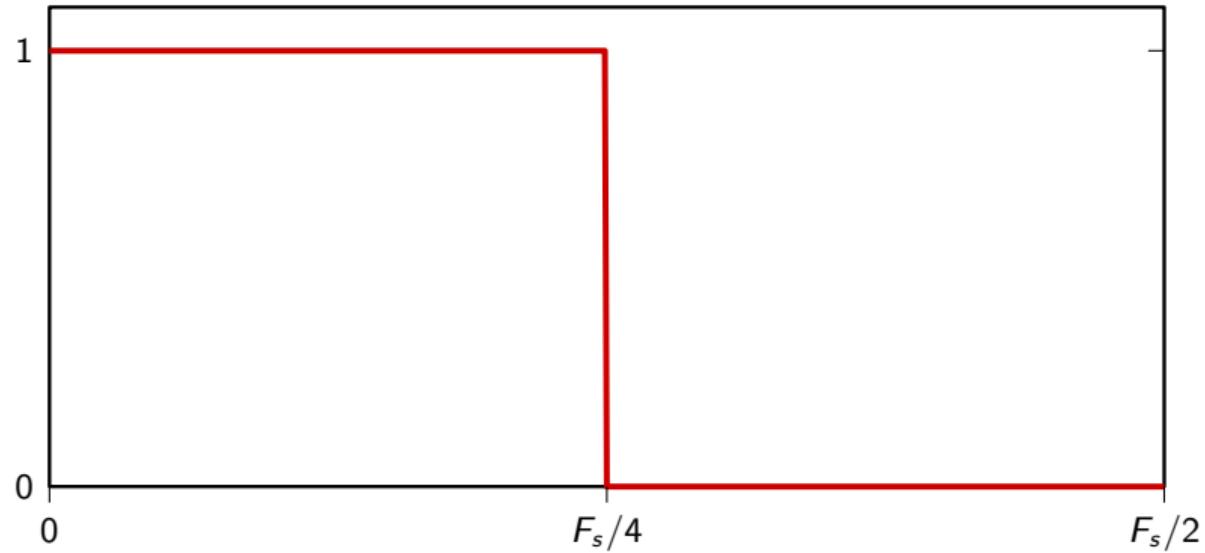
Impulse invariance

$$H_c(f) = H(e^{j2\pi f/F_s})$$

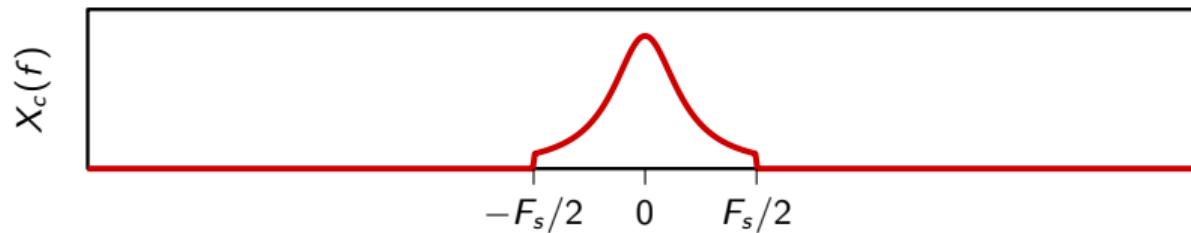
Impulse invariance



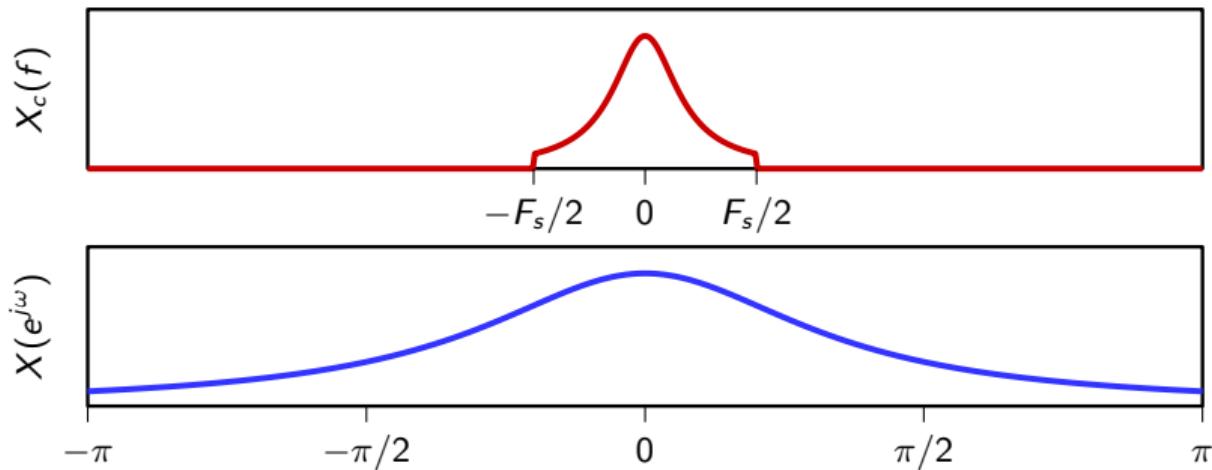
Impulse invariance



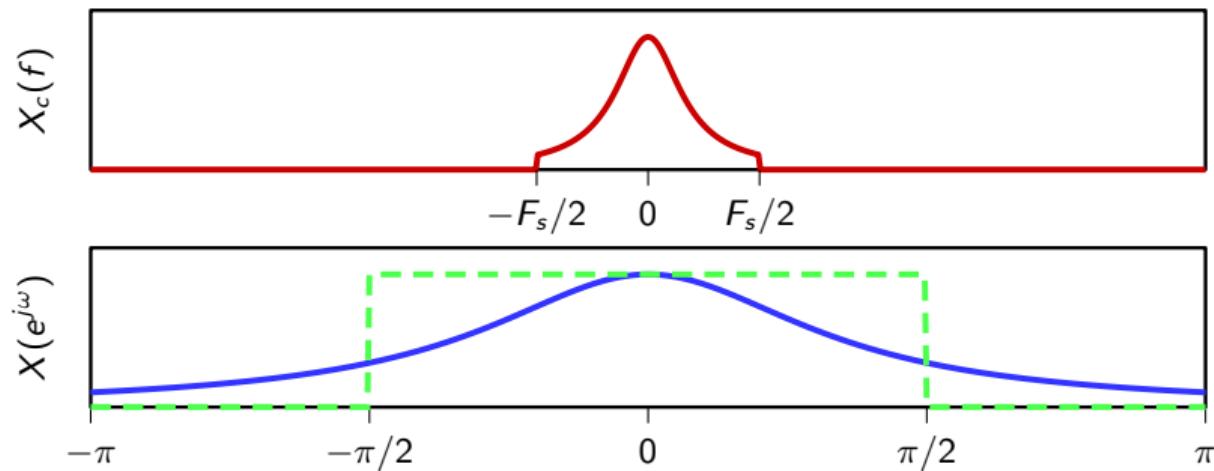
DT processing of CT signals



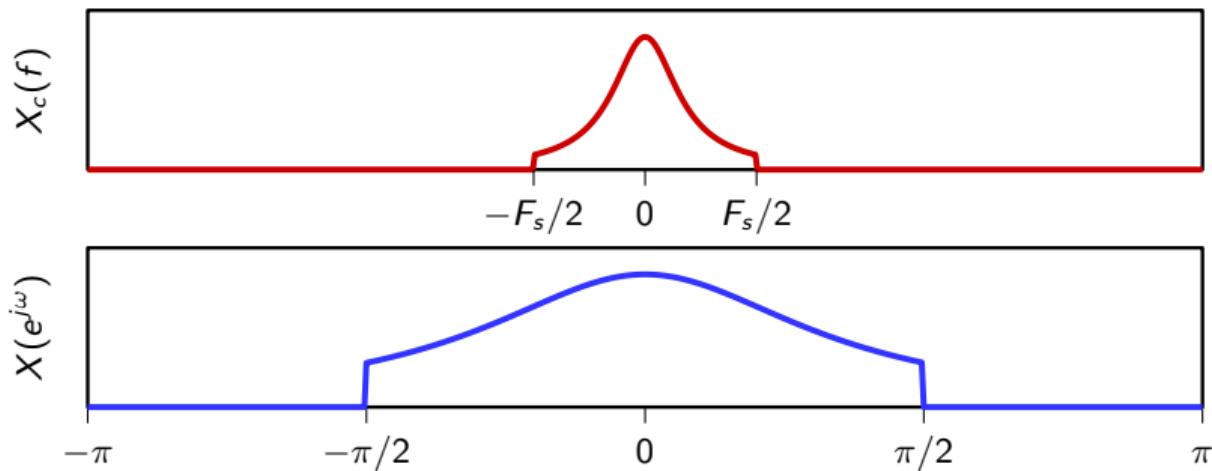
DT processing of CT signals



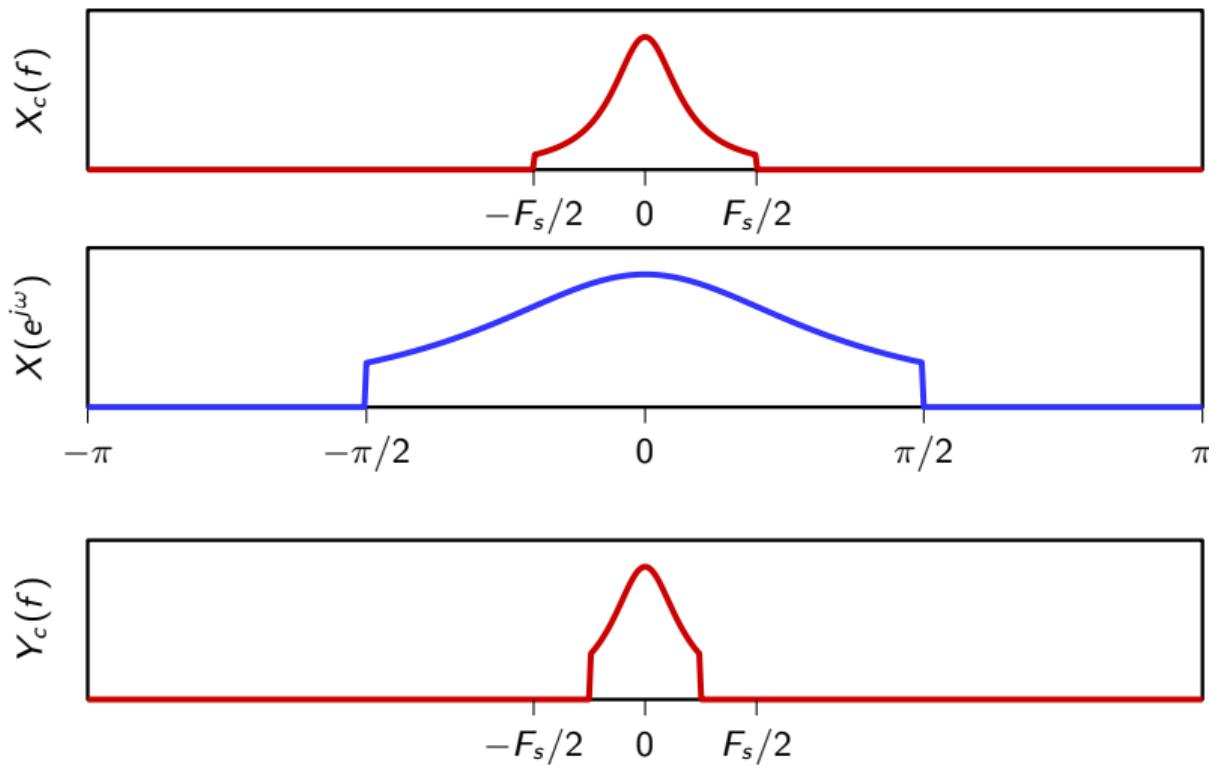
DT processing of CT signals



DT processing of CT signals



DT processing of CT signals



Example

quickly design a discrete-time filter to isolate a band of frequencies between 4000 and 5000Hz;
input signals are bandlimited to 7KHz.

Example

- ▶ 7KHz band limit \Rightarrow we can use any sampling frequency above 14KHz
- ▶ pick $F_s = 16\text{KHz}$
- ▶ we need a bandpass with a 1000Hz bandwidth
- ▶ start with some lowpass with cutoff $f_c = 500\text{Hz}$
- ▶ modulate it to center it on $f_0 = 4500\text{Hz}$

Example

- ▶ 7KHz band limit \Rightarrow we can use any sampling frequency above 14KHz
- ▶ pick $F_s = 16\text{KHz}$
- ▶ we need a bandpass with a 1000Hz bandwidth
- ▶ start with some lowpass with cutoff $f_c = 500\text{Hz}$
- ▶ modulate it to center it on $f_0 = 4500\text{Hz}$

Example

- ▶ 7KHz band limit \Rightarrow we can use any sampling frequency above 14KHz
- ▶ pick $F_s = 16\text{KHz}$
- ▶ we need a bandpass with a 1000Hz bandwidth
- ▶ start with some lowpass with cutoff $f_c = 500\text{Hz}$
- ▶ modulate it to center it on $f_0 = 4500\text{Hz}$

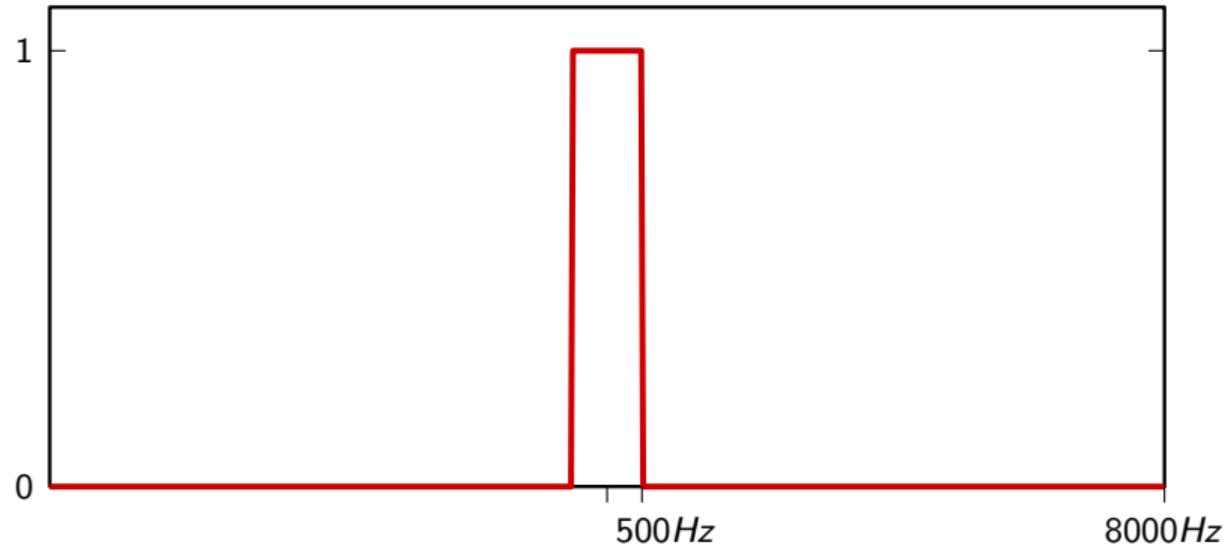
Example

- ▶ 7KHz band limit \Rightarrow we can use any sampling frequency above 14KHz
- ▶ pick $F_s = 16\text{KHz}$
- ▶ we need a bandpass with a 1000Hz bandwidth
- ▶ start with some lowpass with cutoff $f_c = 500\text{Hz}$
- ▶ modulate it to center it on $f_0 = 4500\text{Hz}$

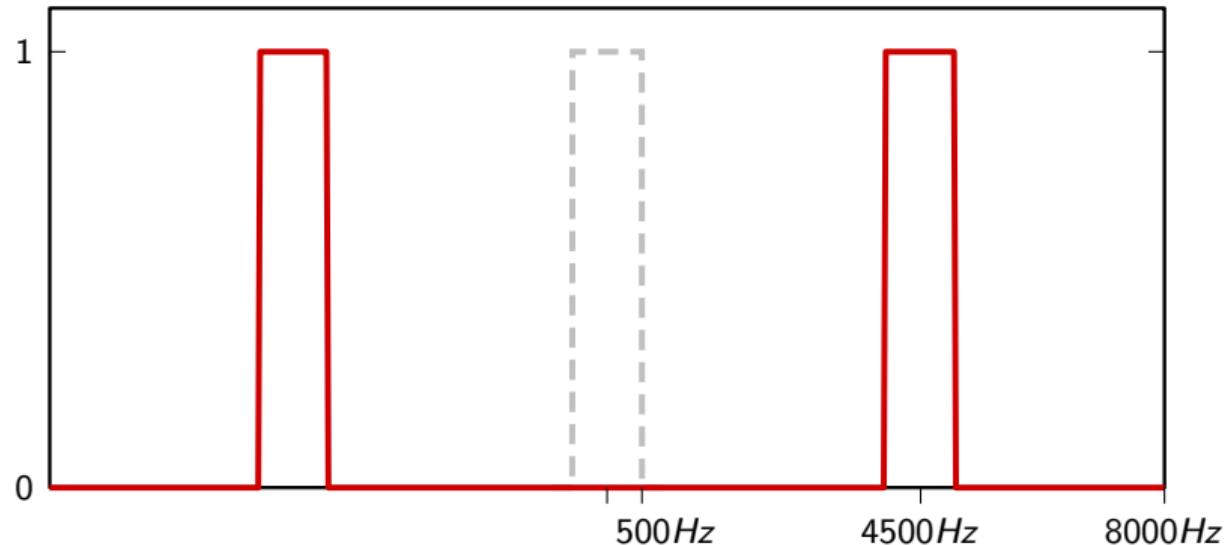
Example

- ▶ 7KHz band limit \Rightarrow we can use any sampling frequency above 14KHz
- ▶ pick $F_s = 16\text{KHz}$
- ▶ we need a bandpass with a 1000Hz bandwidth
- ▶ start with some lowpass with cutoff $f_c = 500\text{Hz}$
- ▶ modulate it to center it on $f_0 = 4500\text{Hz}$

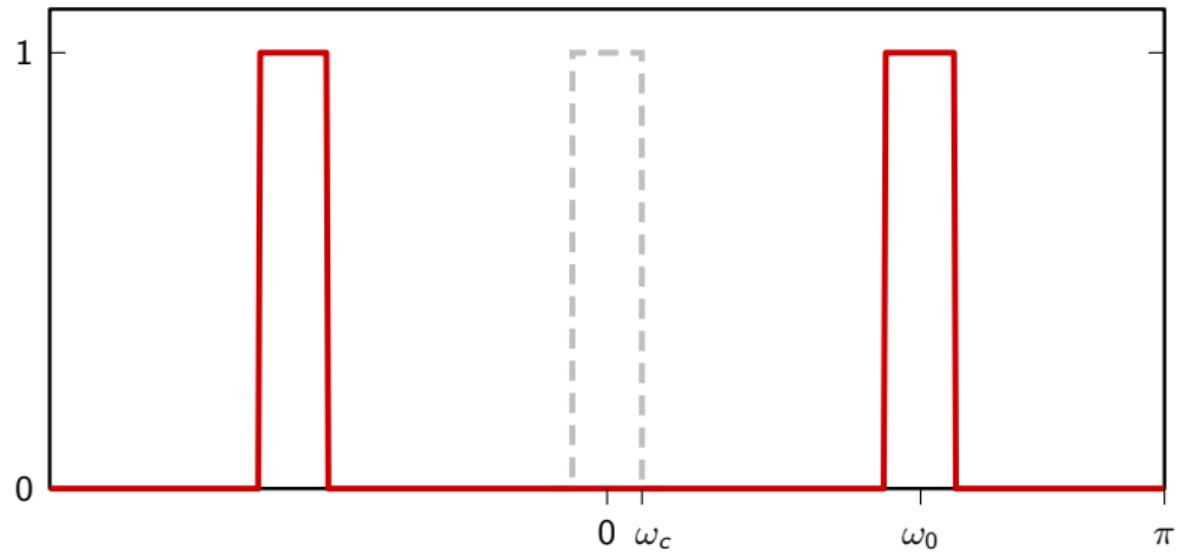
Impulse invariance



Impulse invariance



Impulse invariance



Example

- ▶ $\omega_c = 2\pi \frac{f_c}{F_s} = 2\pi \frac{500}{16000} = 0.0625\pi$
- ▶ $\omega_0 = 2\pi \frac{4500}{16000} = 0.5625\pi$
- ▶ modulate the impulse response of an ideal lowpass with cutoss ω_c by $2 \cos \omega_0 n$
- ▶ truncate the impulse response with an appropriate window

Example

- ▶ $\omega_c = 2\pi \frac{f_c}{F_s} = 2\pi \frac{500}{16000} = 0.0625\pi$
- ▶ $\omega_0 = 2\pi \frac{4500}{16000} = 0.5625\pi$
- ▶ modulate the impulse response of an ideal lowpass with cutoss ω_c by $2 \cos \omega_0 n$
- ▶ truncate the impulse response with an appropriate window

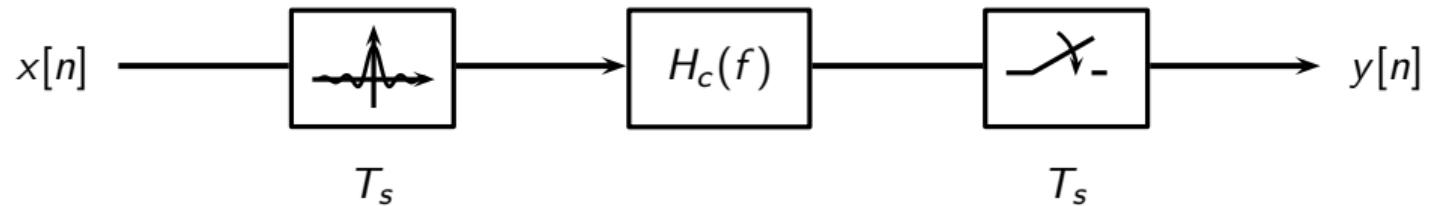
Example

- ▶ $\omega_c = 2\pi \frac{f_c}{F_s} = 2\pi \frac{500}{16000} = 0.0625\pi$
- ▶ $\omega_0 = 2\pi \frac{4500}{16000} = 0.5625\pi$
- ▶ modulate the impulse response of an ideal lowpass with cutoff ω_c by $2 \cos \omega_0 n$
- ▶ truncate the impulse response with an appropriate window

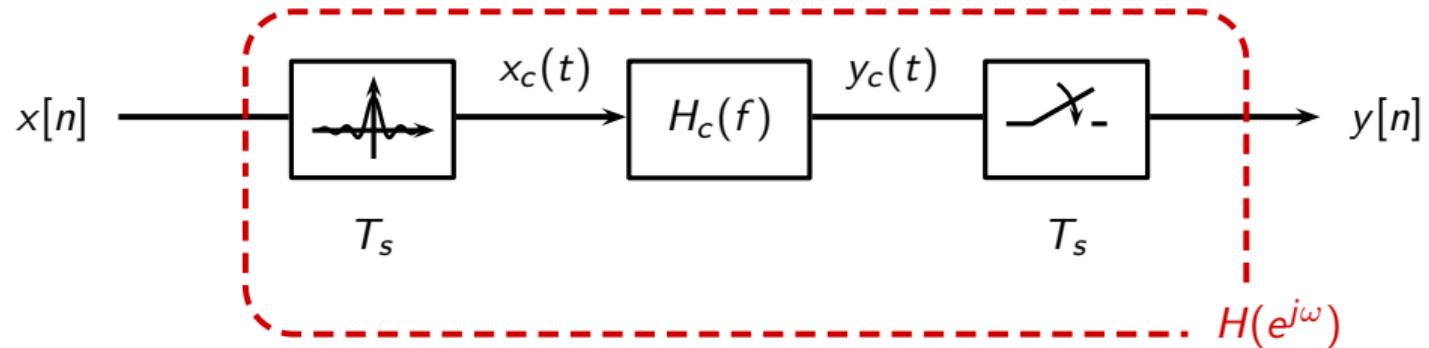
Example

- ▶ $\omega_c = 2\pi \frac{f_c}{F_s} = 2\pi \frac{500}{16000} = 0.0625\pi$
- ▶ $\omega_0 = 2\pi \frac{4500}{16000} = 0.5625\pi$
- ▶ modulate the impulse response of an ideal lowpass with cutoff ω_c by $2 \cos \omega_0 n$
- ▶ truncate the impulse response with an appropriate window

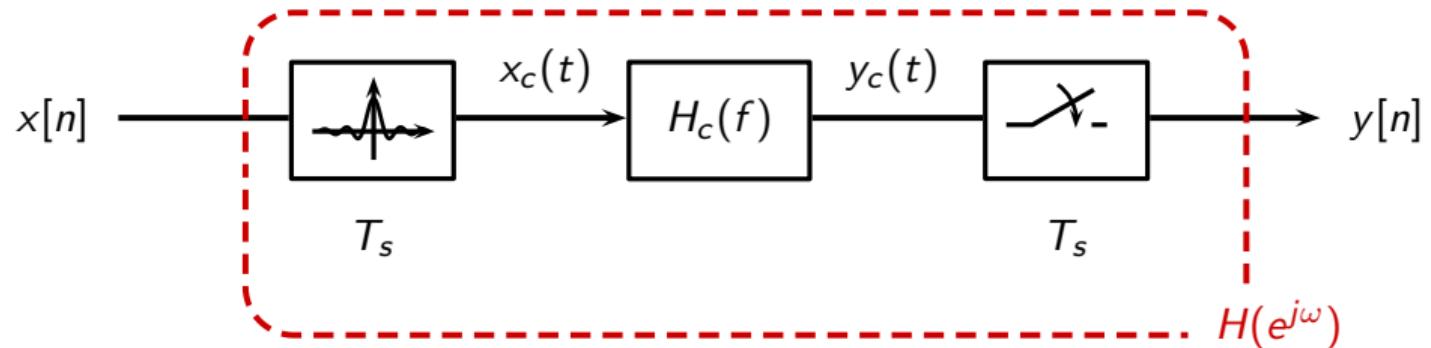
Duality



Duality

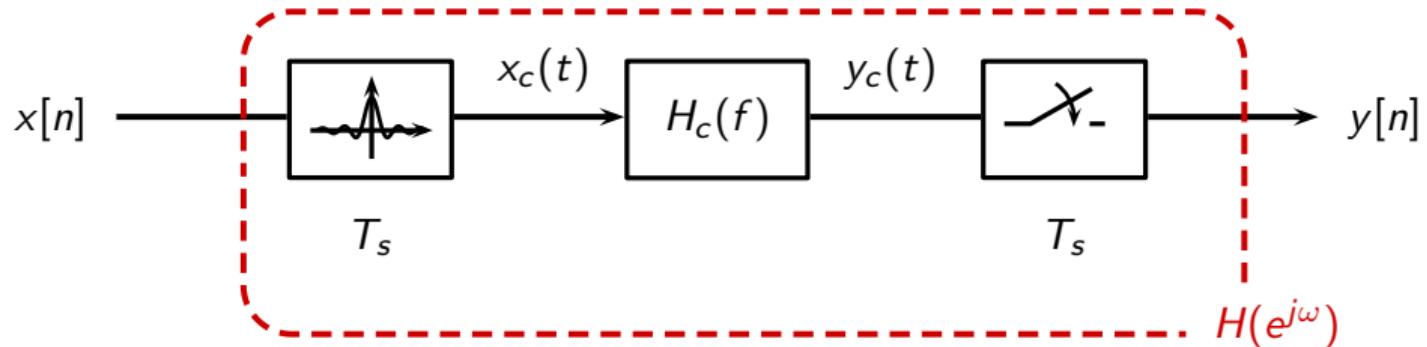


Duality



we can pick any T_s so pick $T_s = 1$:

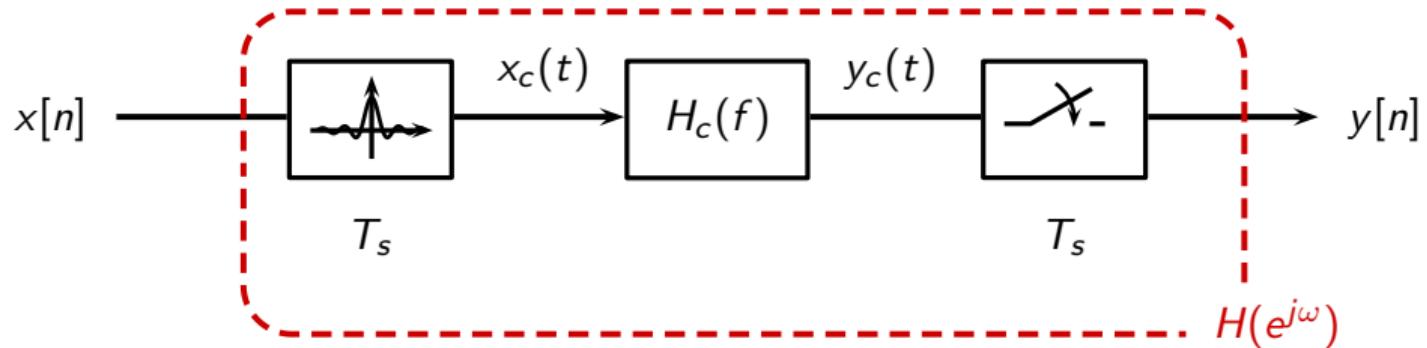
Duality



we can pick any T_s so pick $T_s = 1$:

► $X_c(f) = X(e^{j2\pi f})$

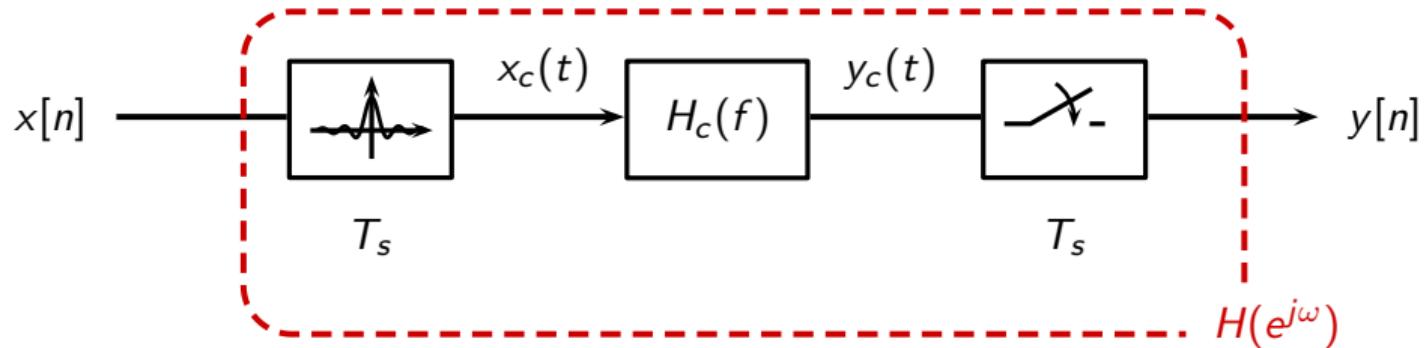
Duality



we can pick any T_s so pick $T_s = 1$:

- ▶ $X_c(f) = X(e^{j2\pi f})$
- ▶ $Y_c(f) = X_c(f)H_c(f)$

Duality



we can pick any T_s so pick $T_s = 1$:

- ▶ $X_c(f) = X(e^{j2\pi f})$
- ▶ $Y_c(f) = X_c(f)H_c(f)$
- ▶ LTI systems cannot change the bandwidth $\Rightarrow Y(e^{j\omega}) = Y_c(\frac{\omega}{2\pi})$

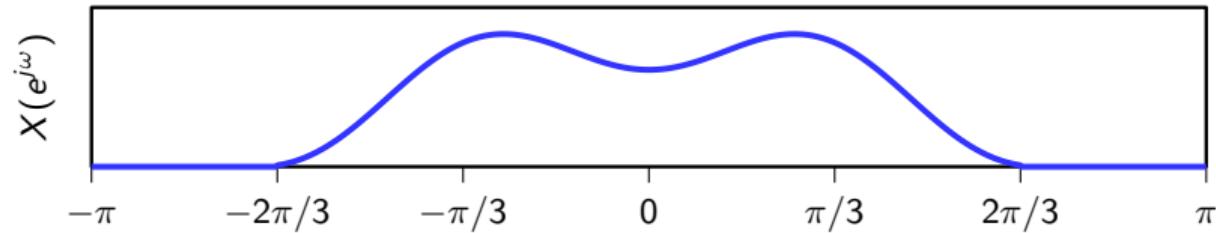
Duality

$$Y(e^{j\omega}) = X(e^{j\omega}) H_c \left(\frac{\omega}{2\pi} \right)$$

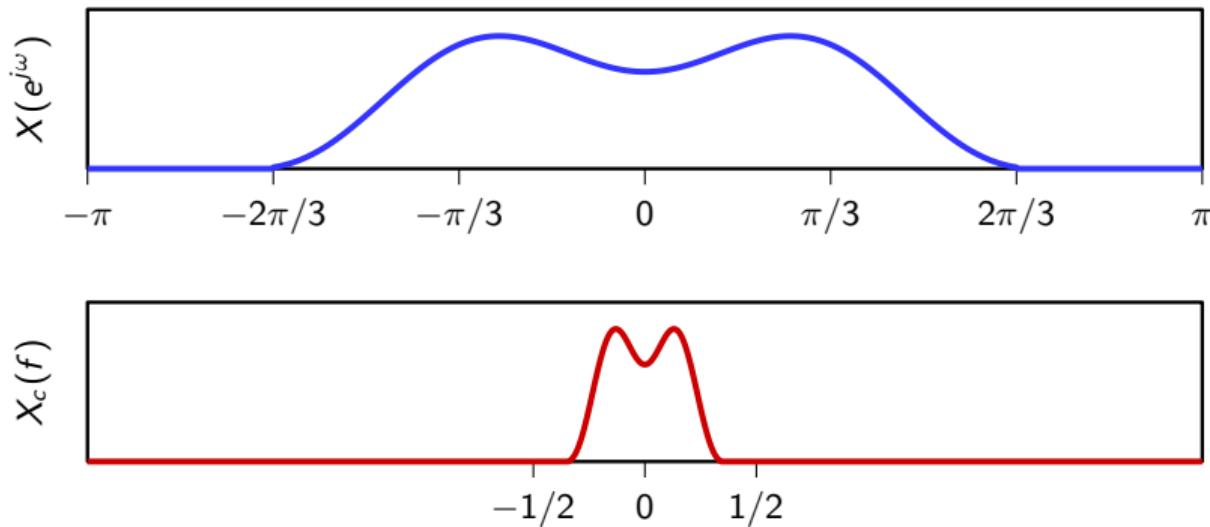
Duality

$$H(e^{j\omega}) = H_c\left(\frac{\omega}{2\pi}\right)$$

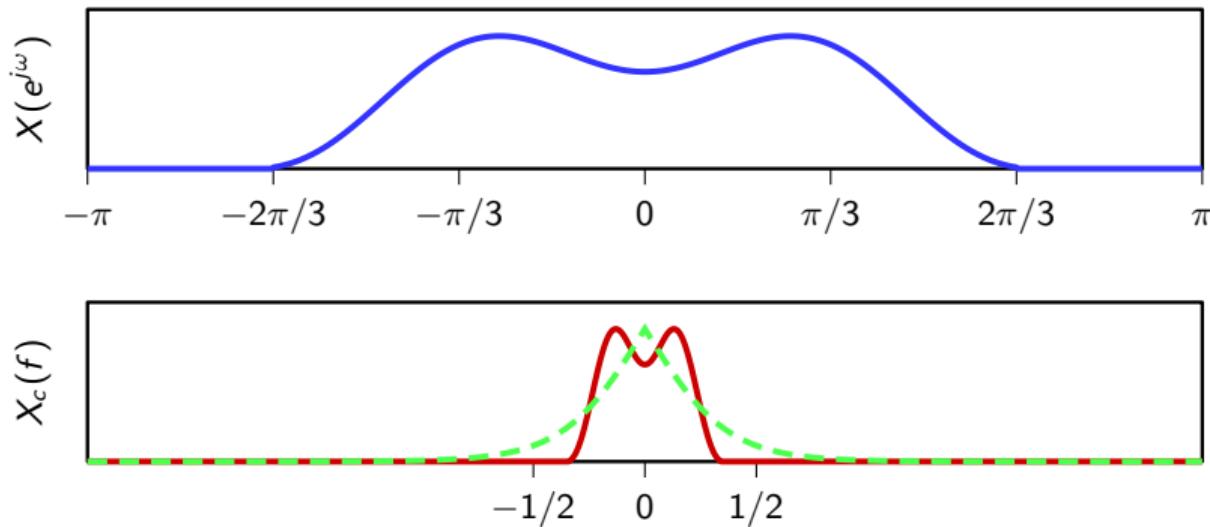
CT processing of DT signals



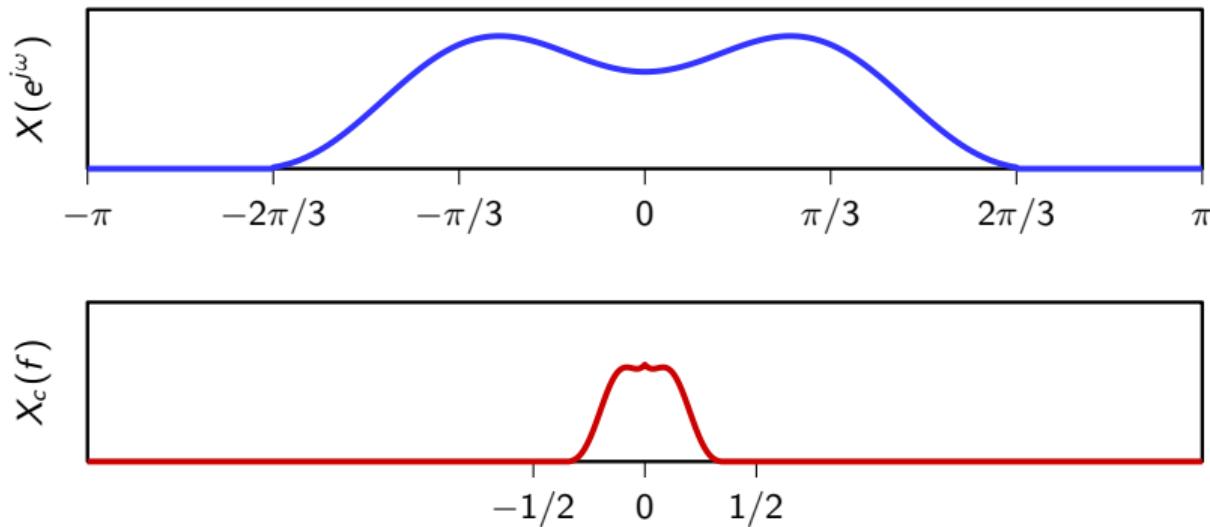
CT processing of DT signals



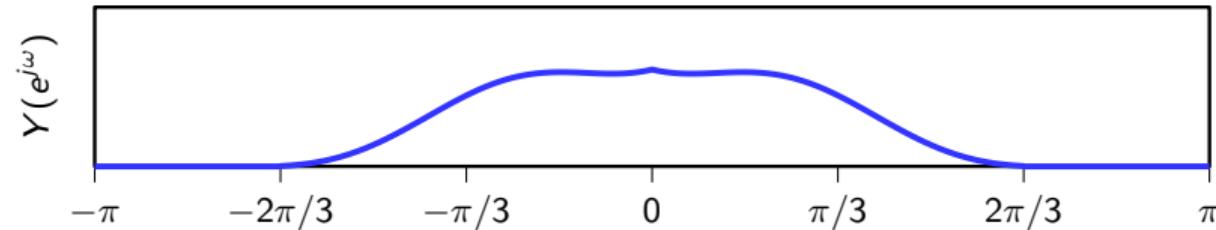
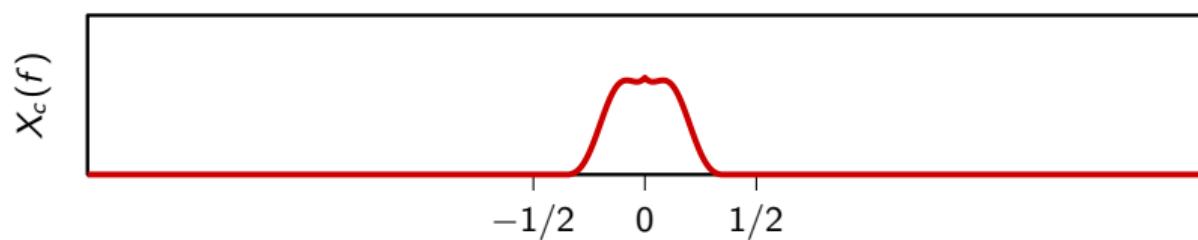
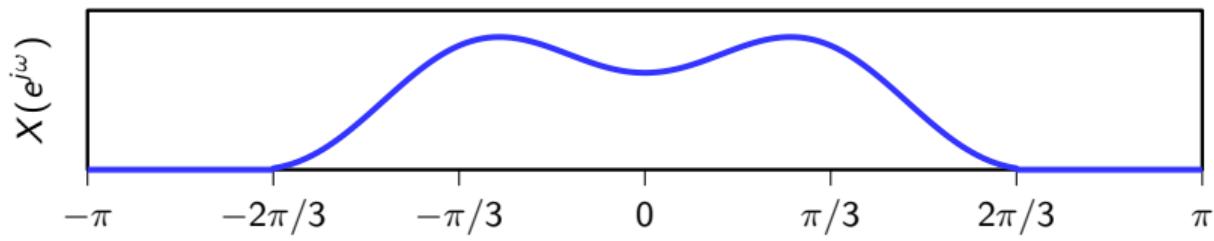
CT processing of DT signals



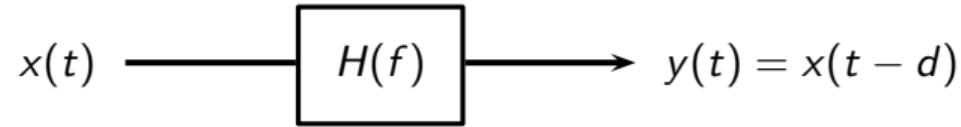
CT processing of DT signals



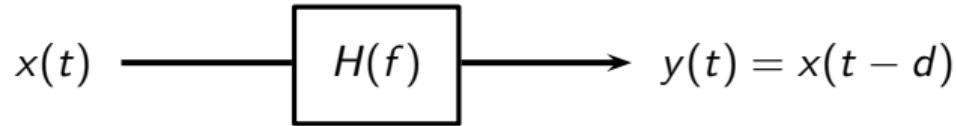
CT processing of DT signals



Delays in continuous time

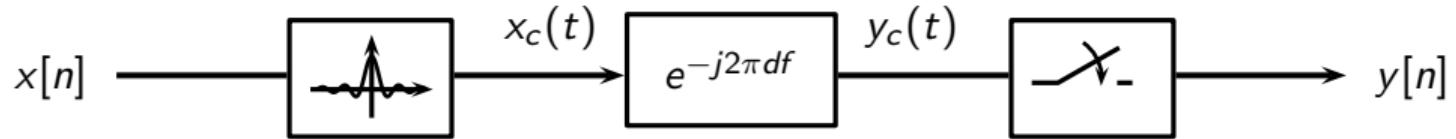


Delays in continuous time

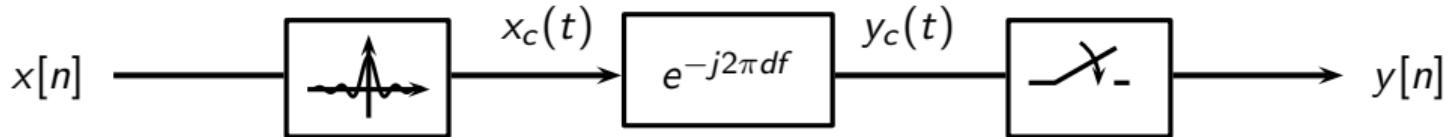


- ▶ in continuous time, delays are well defined for all $d \in \mathbb{R}$
- ▶ $H(f) = e^{-j2\pi df}$

Interpretation of fractional delay by duality

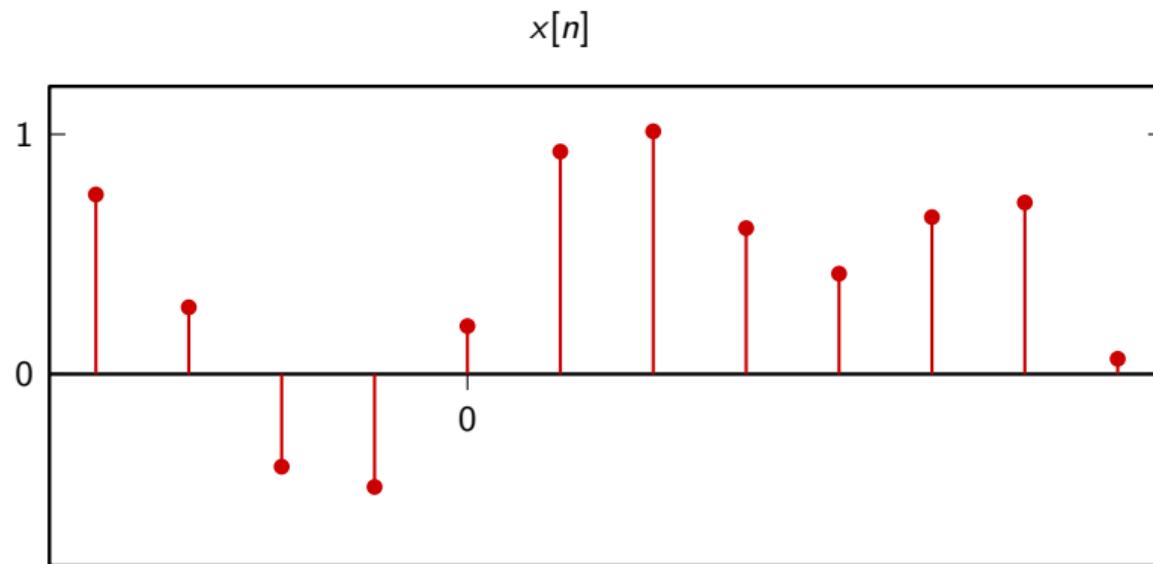


Interpretation of fractional delay by duality

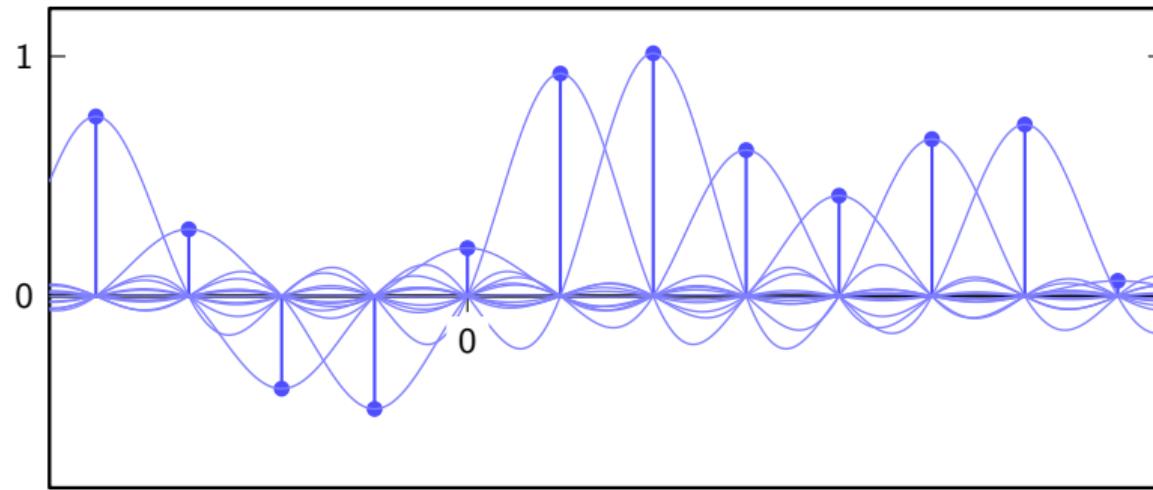


- ▶ chain interpolates $x[n]$, delays the result by d and resamples
- ▶ equivalent filter $H(e^{j\omega}) = H_c(\omega/(2\pi)) = e^{-j\omega d}$
- ▶ that's how a discrete-time fractional delay works internally

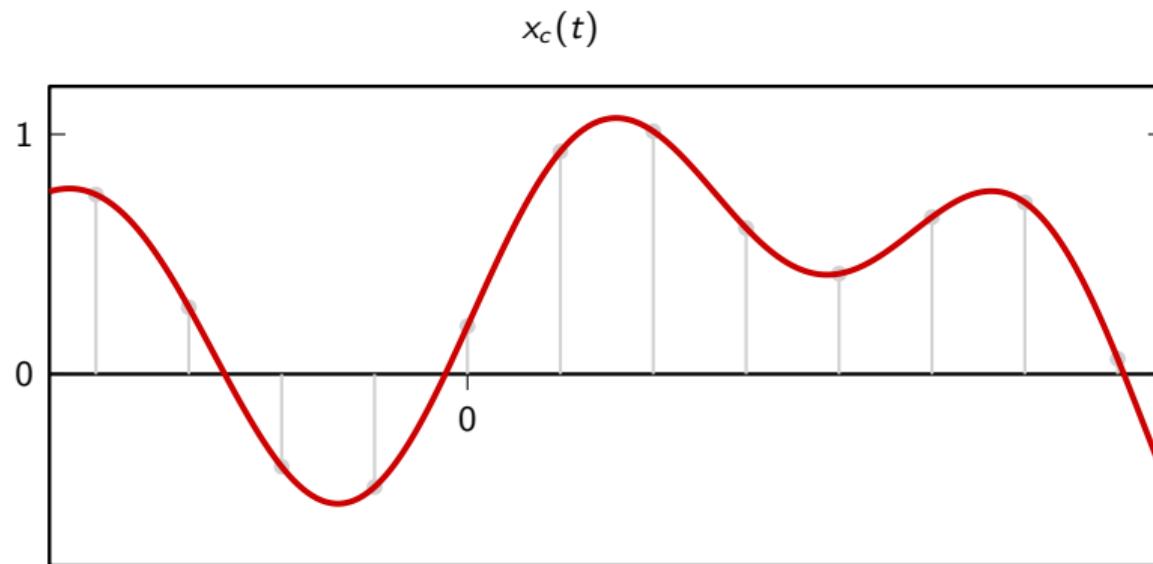
Example: fractional delay



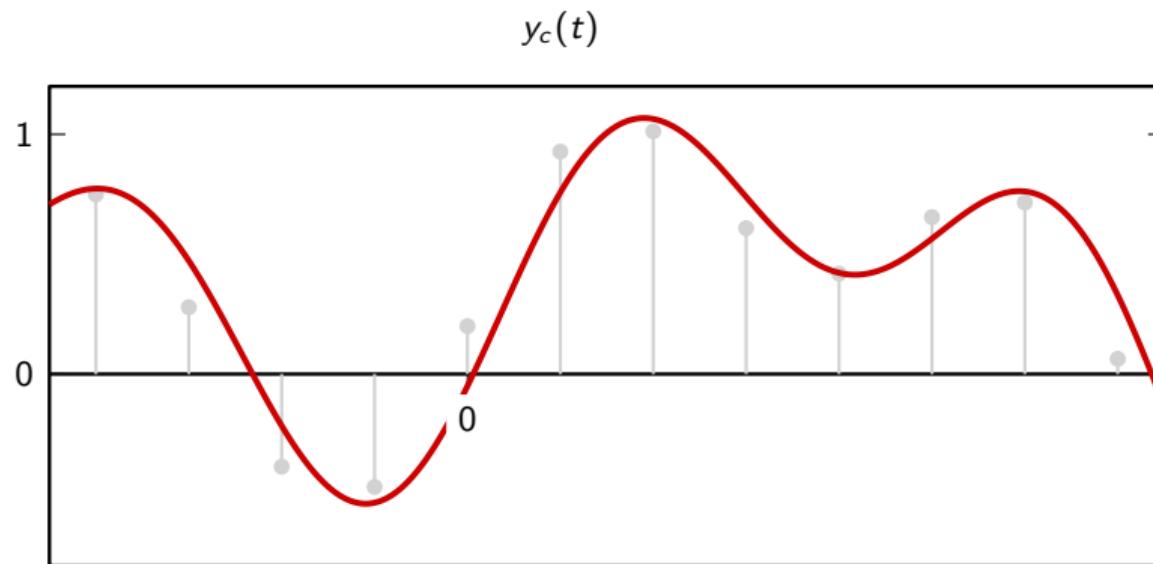
Example: fractional delay



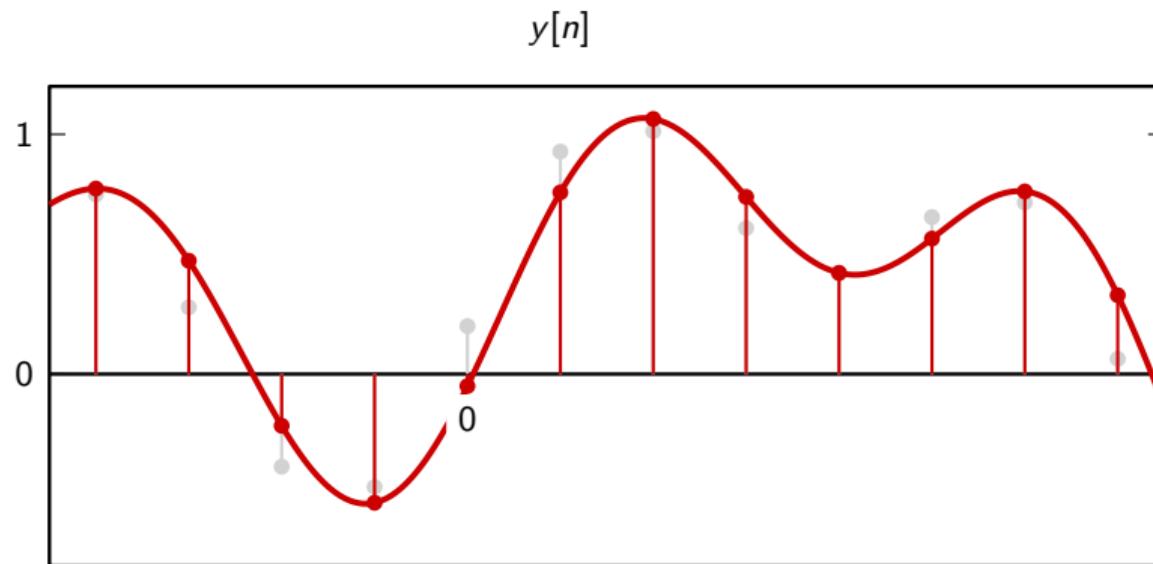
Example: fractional delay



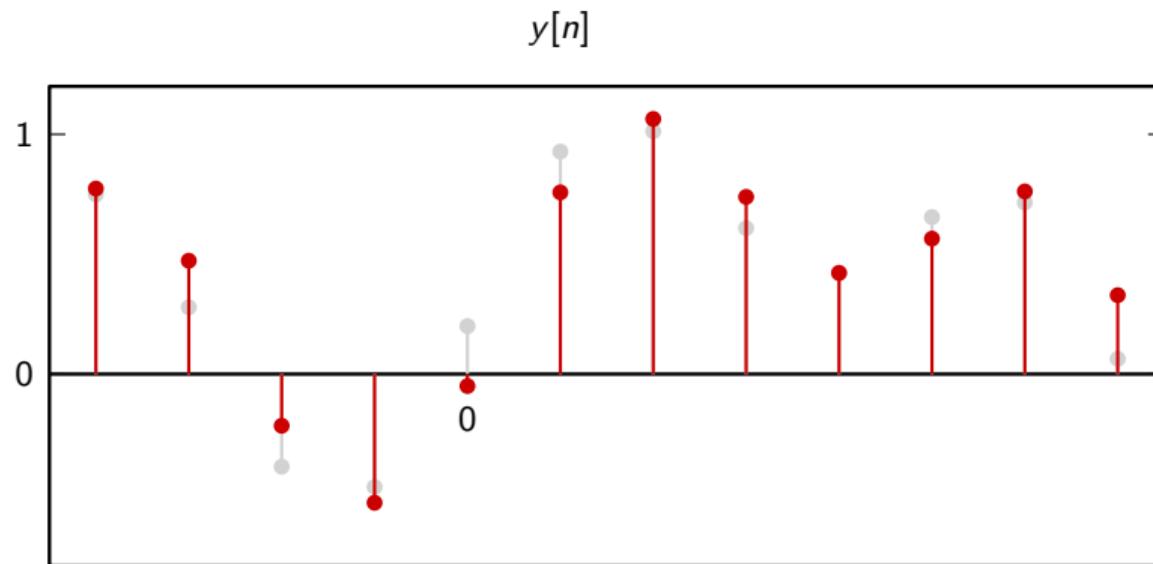
Example: fractional delay



Example: fractional delay



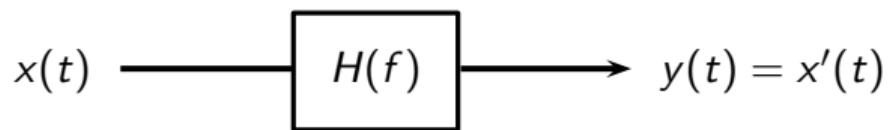
Example: fractional delay



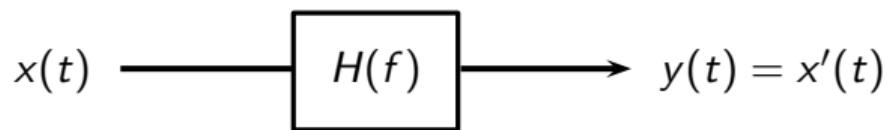
Example: fractional delay

- ▶ $h[n] = \text{sinc}(n - d)$
- ▶ to delay a discrete-time signal by a fraction of a sample we need an ideal filter!
- ▶ efficient approximations exist (e.g. cubic local interpolation)

Differentiation in continuous time

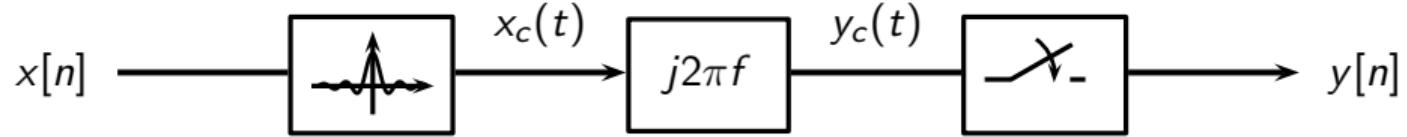


Differentiation in continuous time

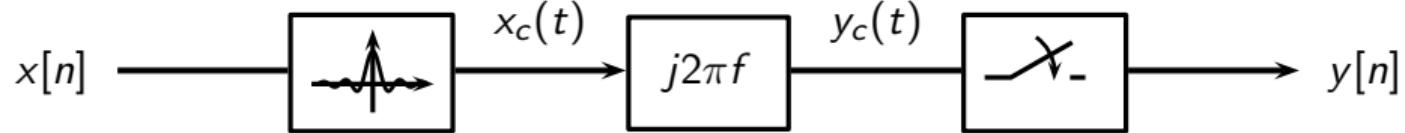


- ▶ easy to show that $\text{FT} \{x'(t)\} = j2\pi f X(f)$
- ▶ $H(f) = j2\pi f$

By duality



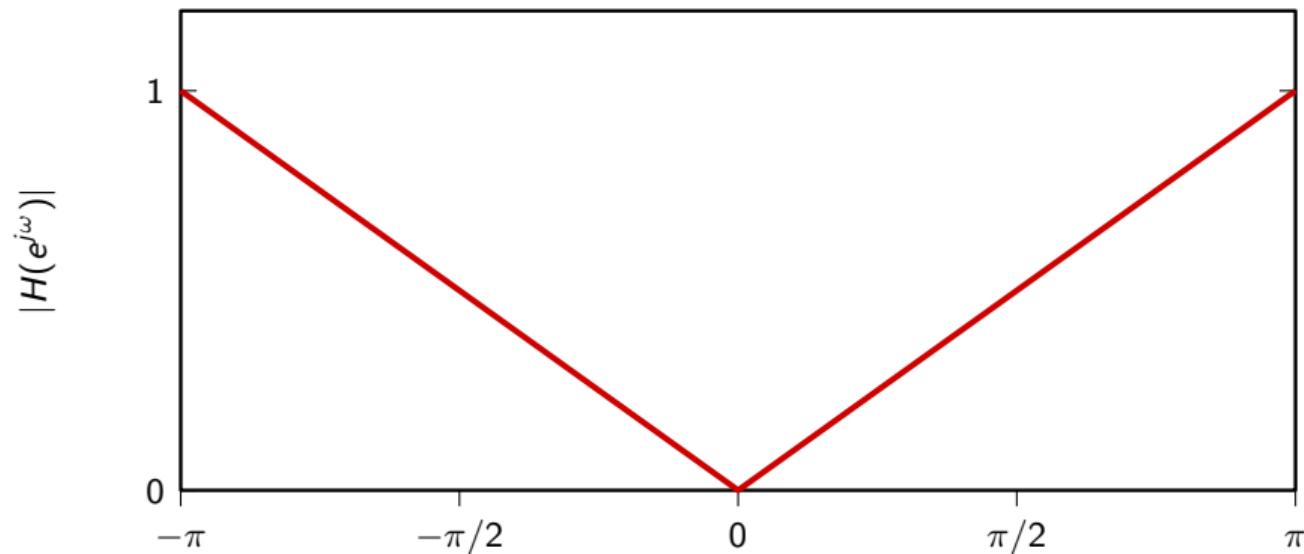
By duality



- ▶ chain interpolates $x[n]$, differentiates the result by d and resamples
- ▶ equivalent filter $H(e^{j\omega}) = H_c(\omega/(2\pi)) = j\omega$
- ▶ equivalent filter defines a digital differentiator

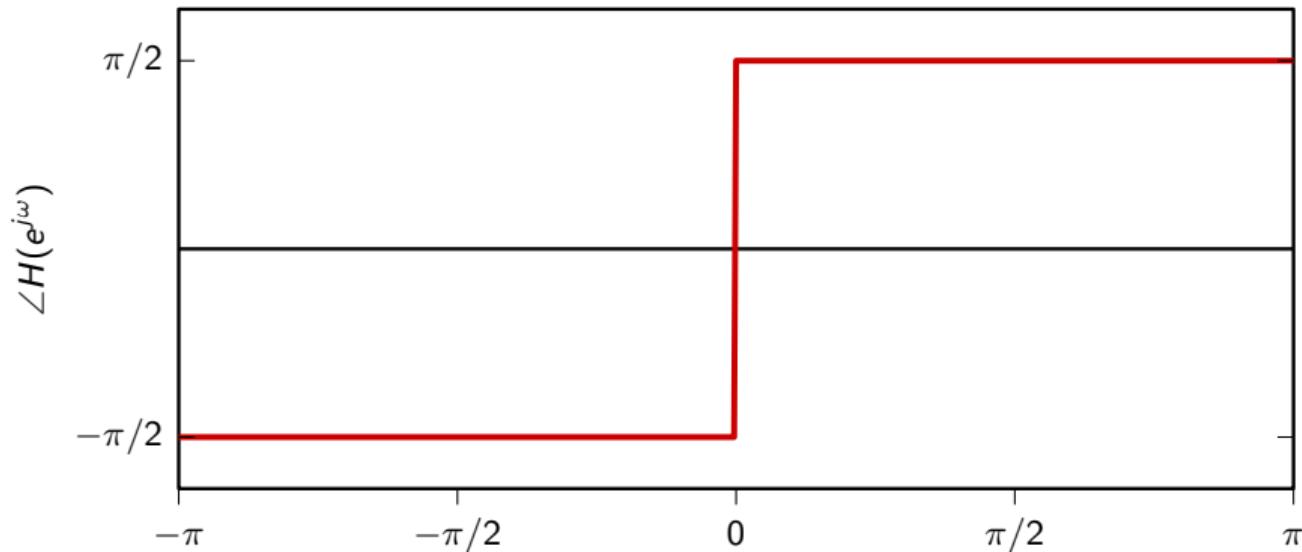
Digital differentiator, magnitude response

$$H(e^{j\omega}) = j\omega$$



Digital differentiator, phase response

$$H(e^{j\omega}) = j\omega$$

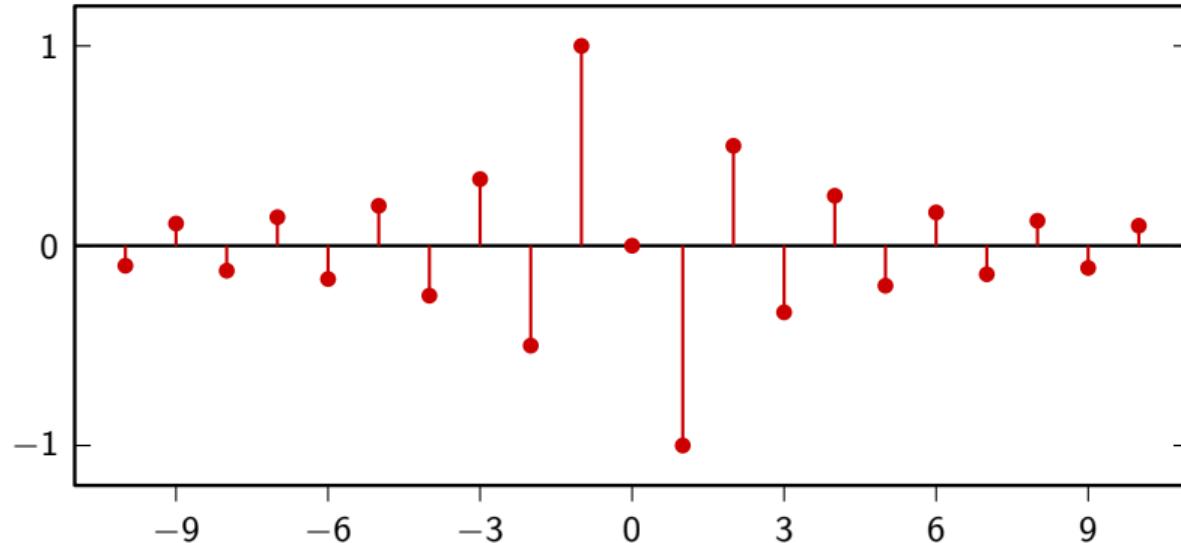


Digital differentiator, impulse response

$$\begin{aligned} h[n] &= \frac{1}{2\pi} \int_{-\pi}^{\pi} j\omega e^{j\omega n} d\omega \\ &= \dots (\textit{integration by parts}) \dots \end{aligned}$$

$$= \begin{cases} 0 & n = 0 \\ \frac{(-1)^n}{n} & n \neq 0 \end{cases}$$

Digital differentiator, impulse response



Digital differentiator

- ▶ the digital differentiator is again an ideal filter!
- ▶ many approximations exist, with different properties

Wrap up

- ▶ Continuous-time processing of discrete-time sequences
- ▶ Discrete-time processing of continuous-time signals
- ▶ Jumping back and forth using sampling and interpolation
- ▶ In practice: Many applications of processing continuous-time signals in discrete time!

COM303: Digital Signal Processing

Lecture 18: Multirate signal processing

overview

- ▶ ideal and practical sampling and interpolation
- ▶ bandpass sampling
- ▶ multirate signal processing

advanced topics in sampling

From Continuous to Discrete Time

$$x_c(t) \longrightarrow x[n]$$

From Continuous to Discrete Time

$$x_c(t) \longrightarrow x[n]$$

ideally

in practice

From Continuous to Discrete Time

$$x_c(t) \longrightarrow x[n]$$

ideally

in practice

$$x[n] = \langle x_c(t), \text{sinc} \left(\frac{t - nT_s}{T_s} \right) \rangle$$

From Continuous to Discrete Time

$$x_c(t) \longrightarrow x[n]$$

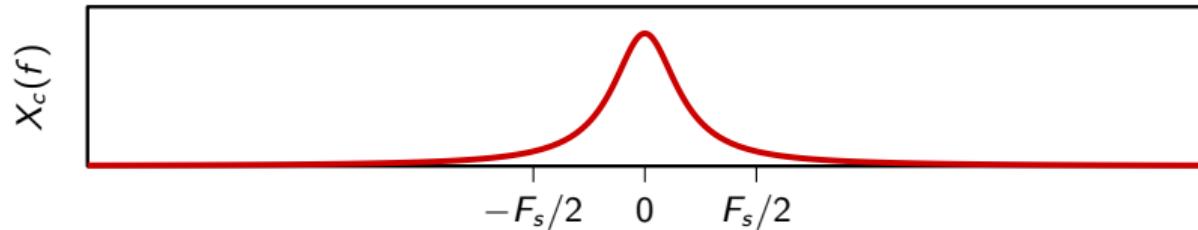
ideally

in practice

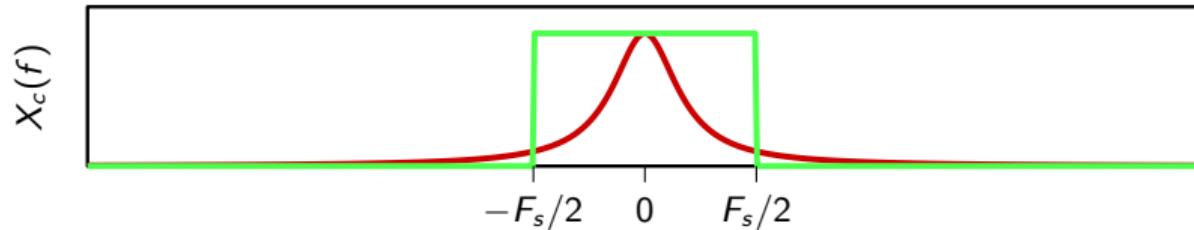
$$x[n] = \langle x_c(t), \text{sinc} \left(\frac{t - nT_s}{T_s} \right) \rangle$$

$$X(e^{j\omega}) = F_s X_c \left(\frac{\omega}{2\pi} F_s \right)$$

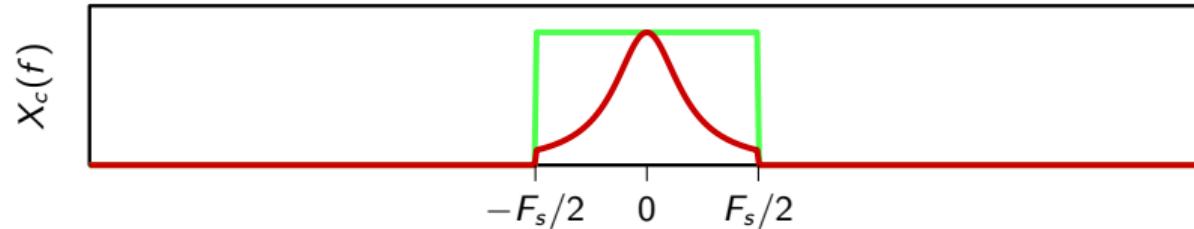
Ideal case



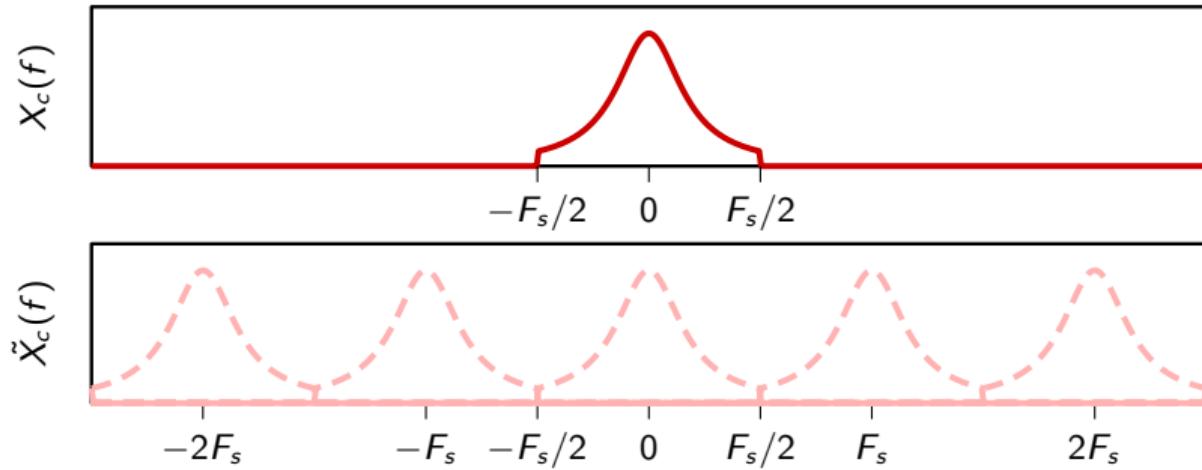
Ideal case



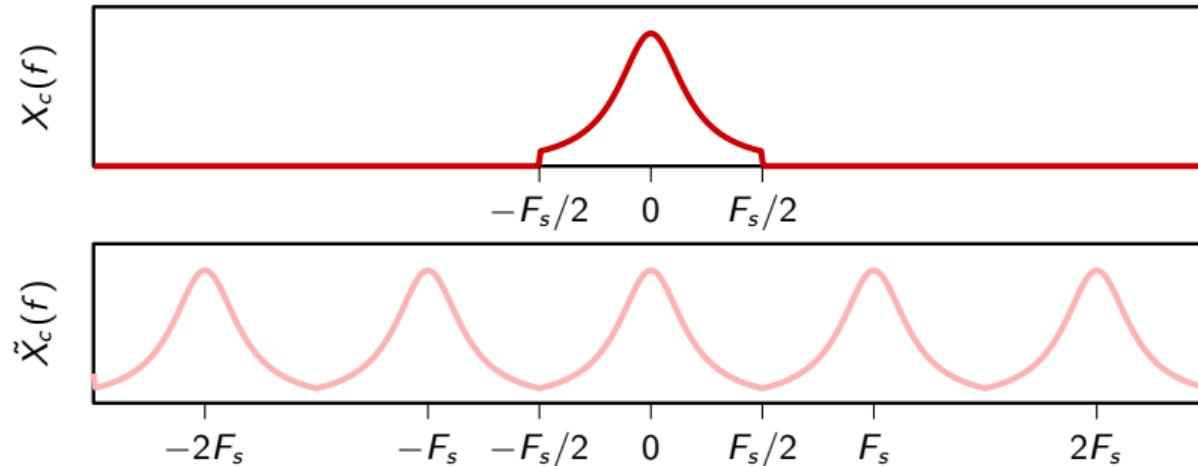
Ideal case



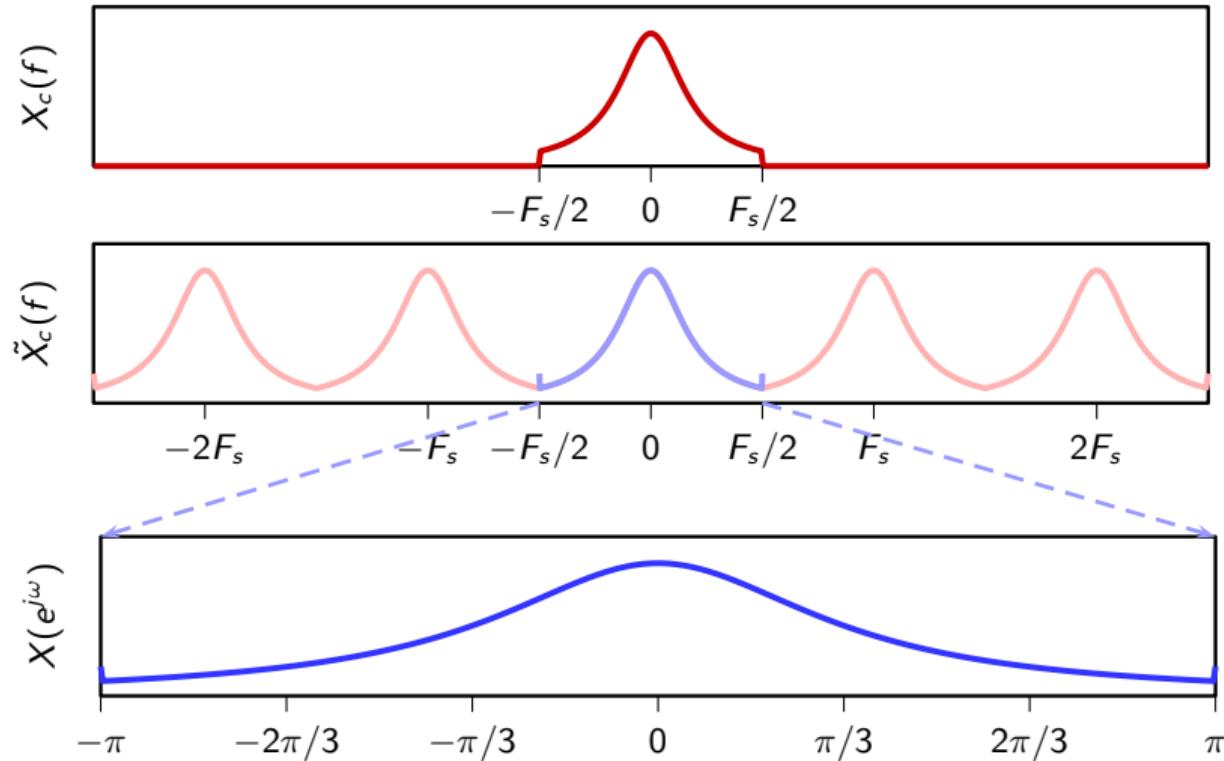
Ideal case



Ideal case



Ideal case



From Continuous to Discrete Time

$$x_c(t) \longrightarrow x[n]$$

ideally

in practice

$$x[n] = \langle x_c(t), \text{sinc}\left(\frac{t - nT_s}{T_s}\right) \rangle$$

$$X(e^{j\omega}) = F_s X_c \left(\frac{\omega}{2\pi} F_s \right)$$

From Continuous to Discrete Time

$$x_c(t) \longrightarrow x[n]$$

ideally

in practice

$$x[n] = \langle x_c(t), \text{sinc}\left(\frac{t - nT_s}{T_s}\right) \rangle \quad x[n] = x_c(nT_s)$$

$$X(e^{j\omega}) = F_s X_c \left(\frac{\omega}{2\pi} F_s \right)$$

From Continuous to Discrete Time

$$x_c(t) \longrightarrow x[n]$$

ideally

$$x[n] = \langle x_c(t), \text{sinc} \left(\frac{t - nT_s}{T_s} \right) \rangle$$

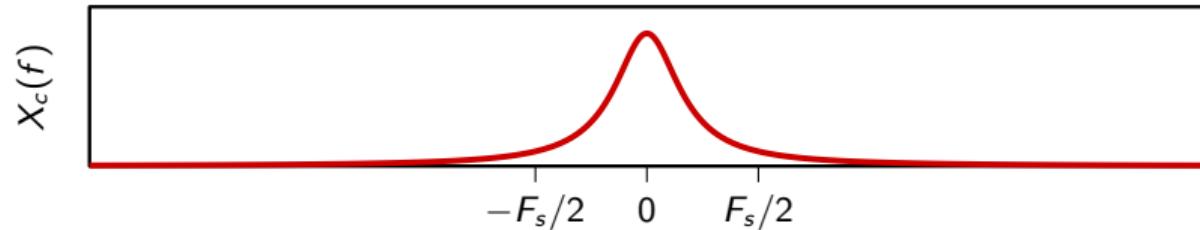
in practice

$$x[n] = x_c(nT_s)$$

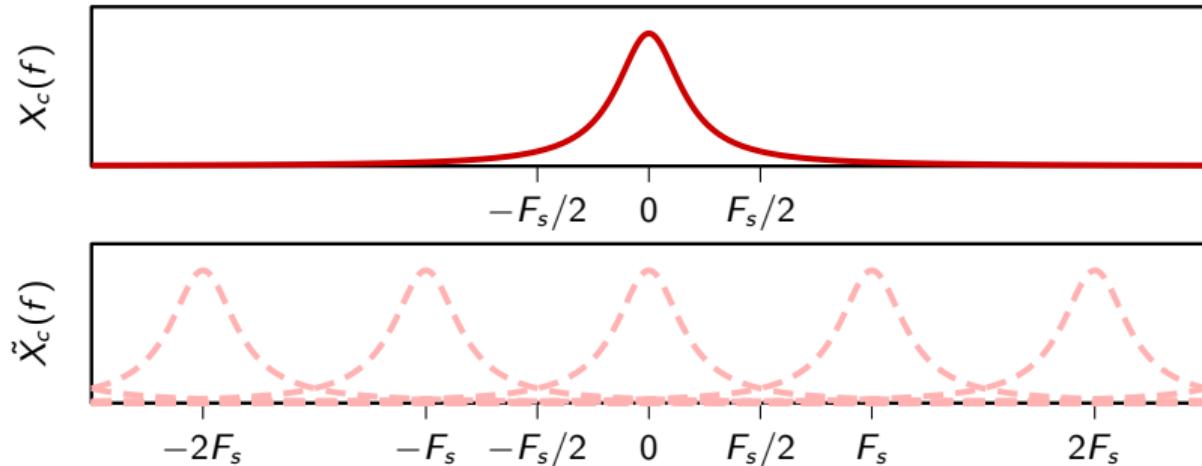
$$X(e^{j\omega}) = F_s X_c \left(\frac{\omega}{2\pi} F_s \right)$$

$$X(e^{j\omega}) = F_s \sum_{k=-\infty}^{\infty} X_c \left(\frac{\omega}{2\pi} F_s - kF_s \right)$$

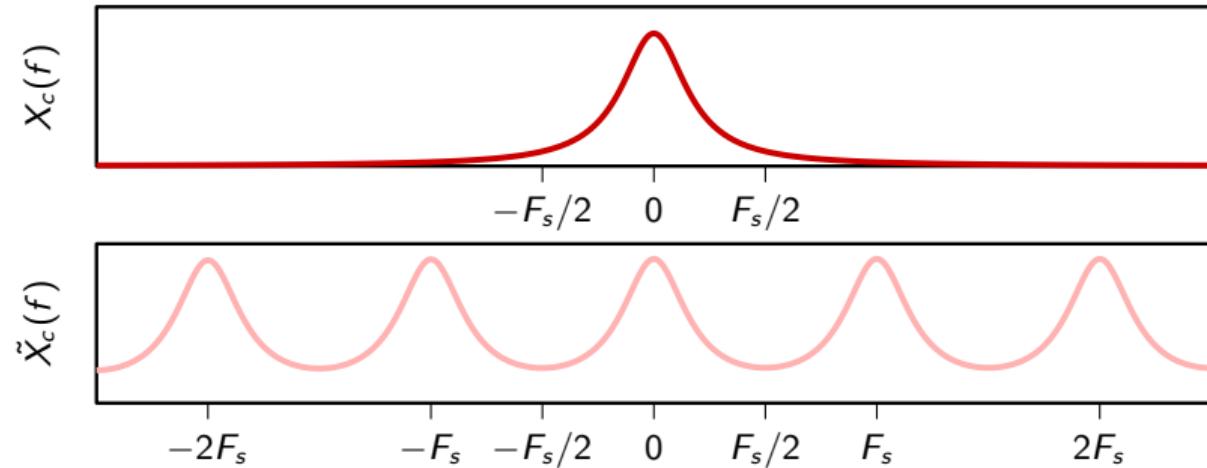
In practice



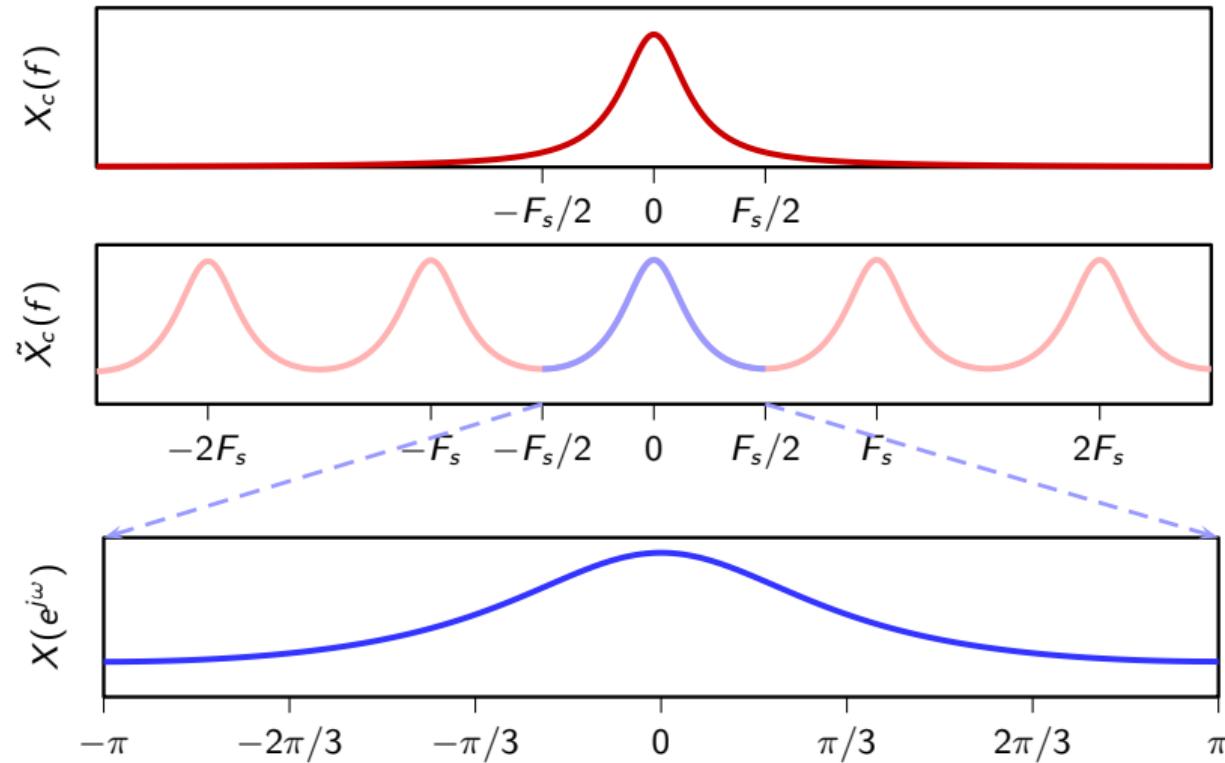
In practice



In practice



In practice



From Discrete to Continuous Time

$$x[n] \longrightarrow x_c(t)$$

From Discrete to Continuous Time

$$x[n] \longrightarrow x_c(t)$$

ideally

$$x_c(t) = \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right)$$

From Discrete to Continuous Time

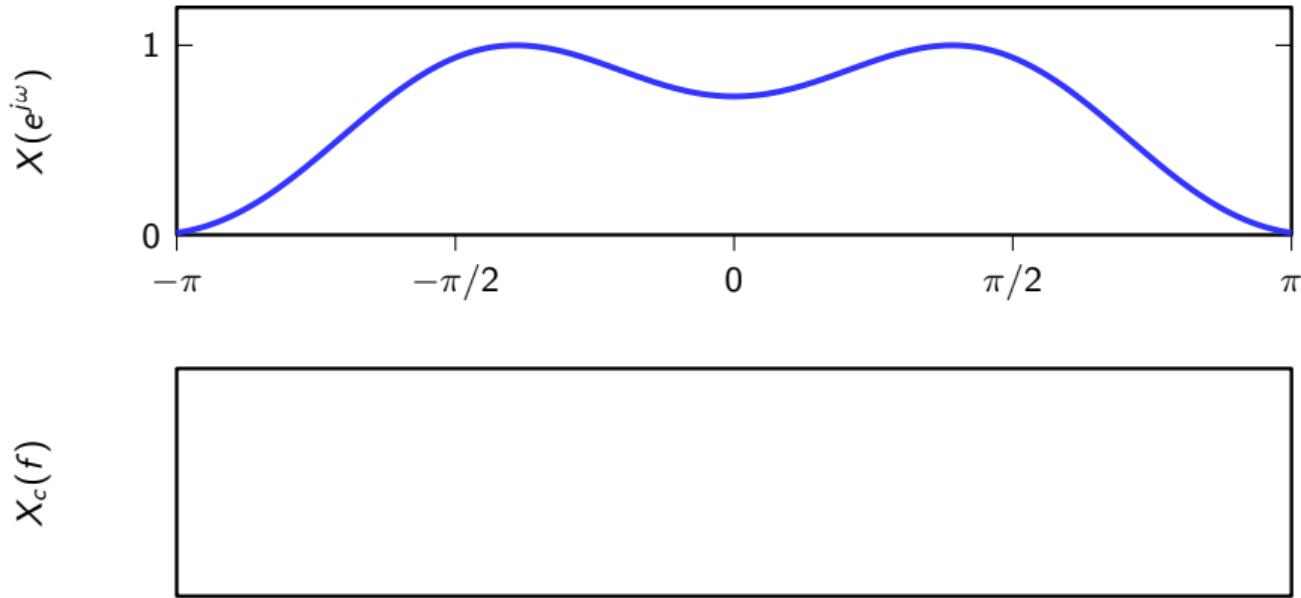
$$x[n] \longrightarrow x_c(t)$$

ideally

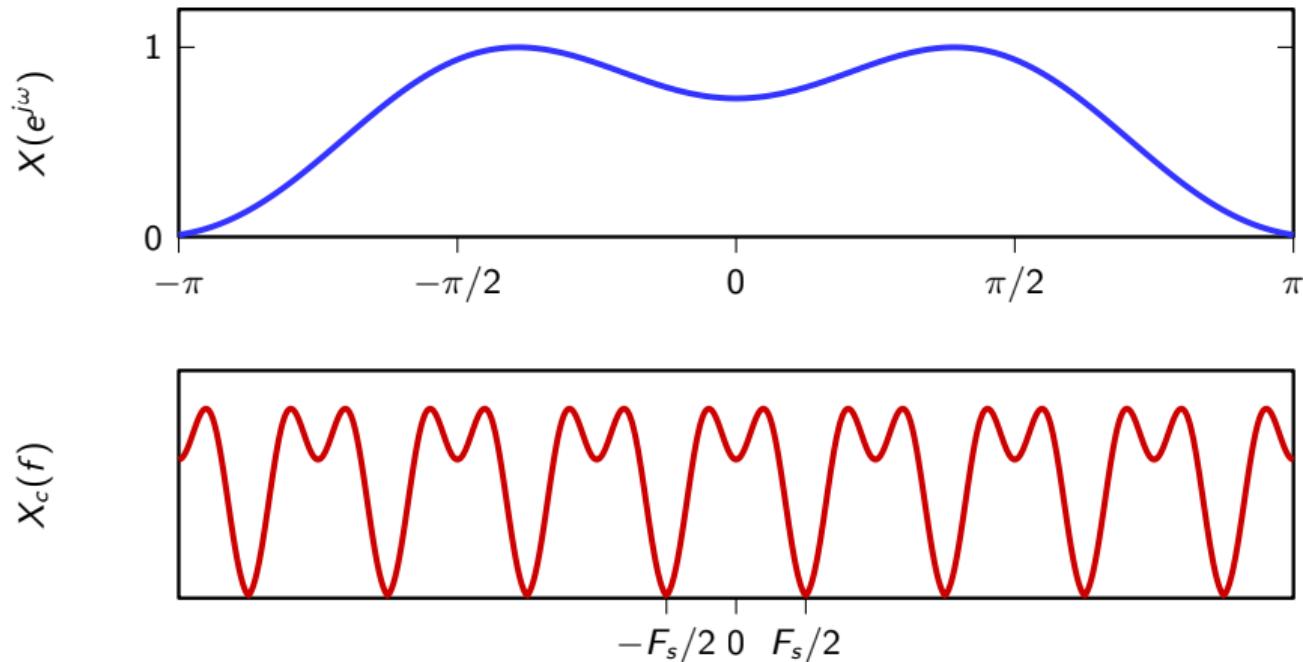
$$x_c(t) = \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right)$$

$$X_c(f) = \frac{1}{F_s} X(e^{j2\pi f/F_s}) \operatorname{rect}\left(\frac{f}{F_s}\right)$$

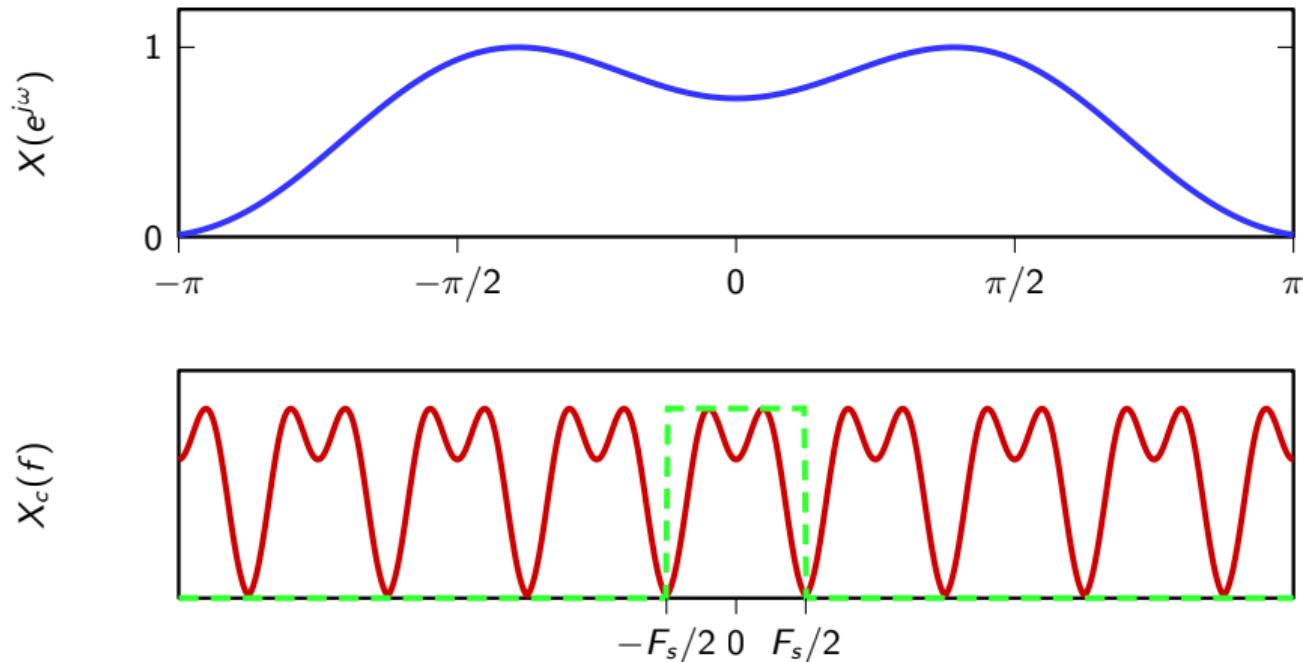
sinc interpolation



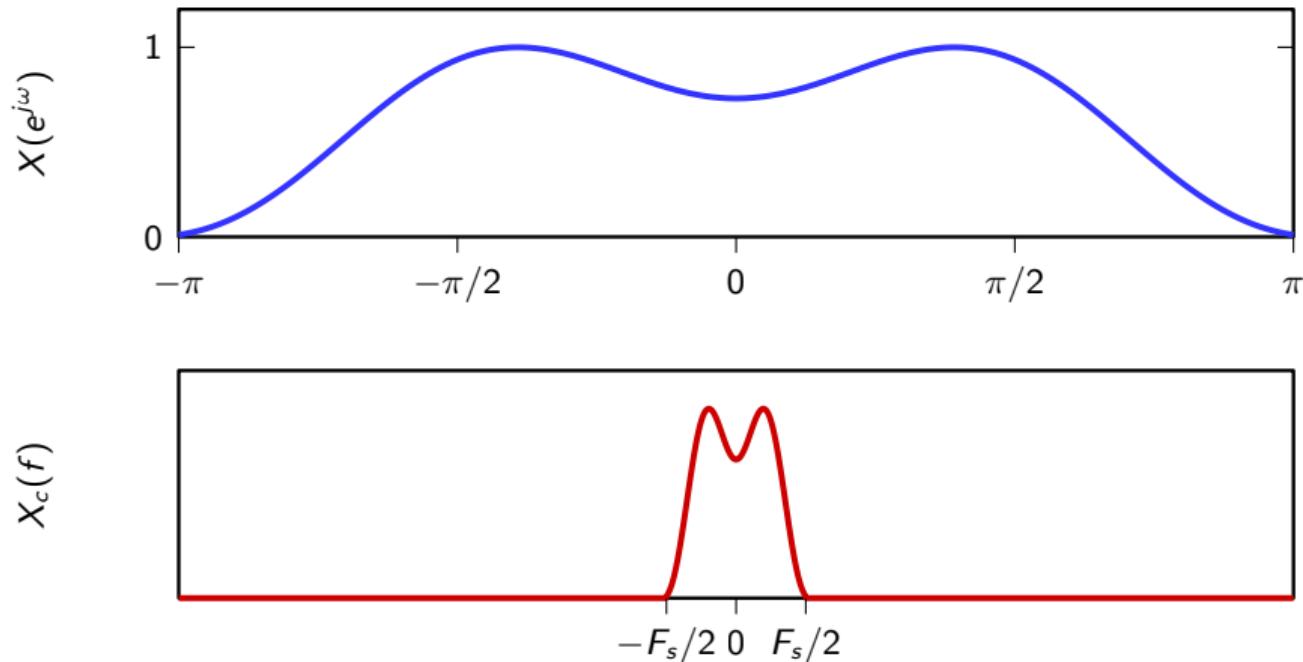
sinc interpolation



sinc interpolation



sinc interpolation



From Discrete to Continuous Time

$$x[n] \longrightarrow x(t)$$

ideally

in practice

$$x_c(t) = \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right)$$

$$X_c(f) = \frac{1}{F_s} X(e^{j2\pi f/F_s}) \operatorname{rect}\left(\frac{f}{F_s}\right)$$

From Discrete to Continuous Time

$$x[n] \longrightarrow x(t)$$

ideally

$$x_c(t) = \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right)$$

in practice

$$x_c(t) = \sum_{n=-\infty}^{\infty} x[n] i\left(\frac{t - nT_s}{T_s}\right)$$

$$X_c(f) = \frac{1}{F_s} X(e^{j2\pi f/F_s}) \operatorname{rect}\left(\frac{f}{F_s}\right)$$

From Discrete to Continuous Time

$$x[n] \longrightarrow x(t)$$

ideally

$$x_c(t) = \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right)$$

in practice

$$x_c(t) = \sum_{n=-\infty}^{\infty} x[n] i\left(\frac{t - nT_s}{T_s}\right)$$

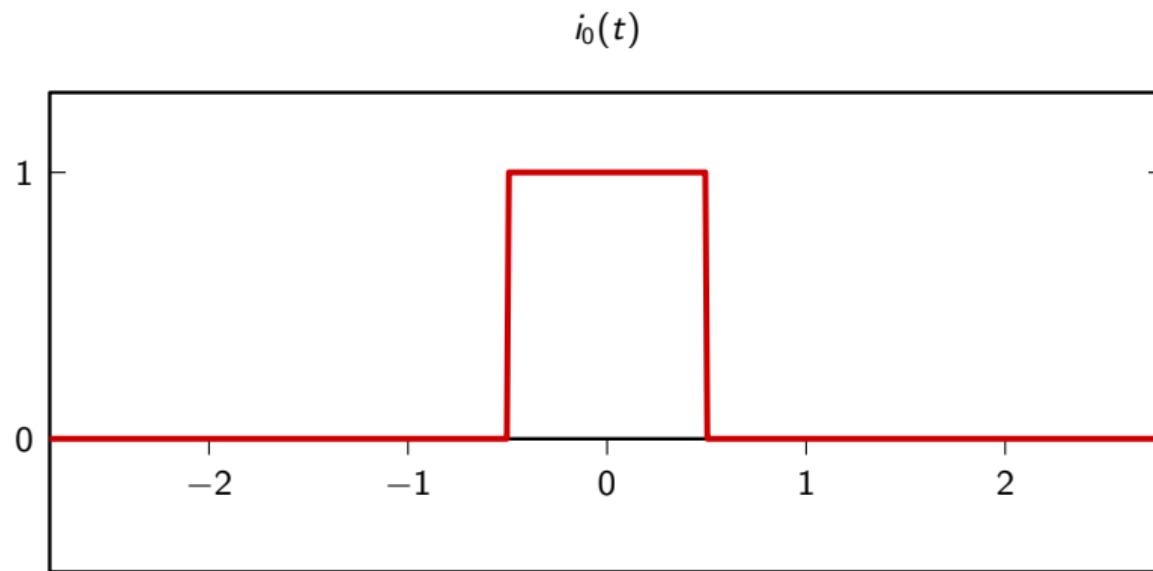
$$X_c(f) = \frac{1}{F_s} X(e^{j2\pi f/F_s}) \operatorname{rect}\left(\frac{f}{F_s}\right)$$

$$X_c(f) = ?$$

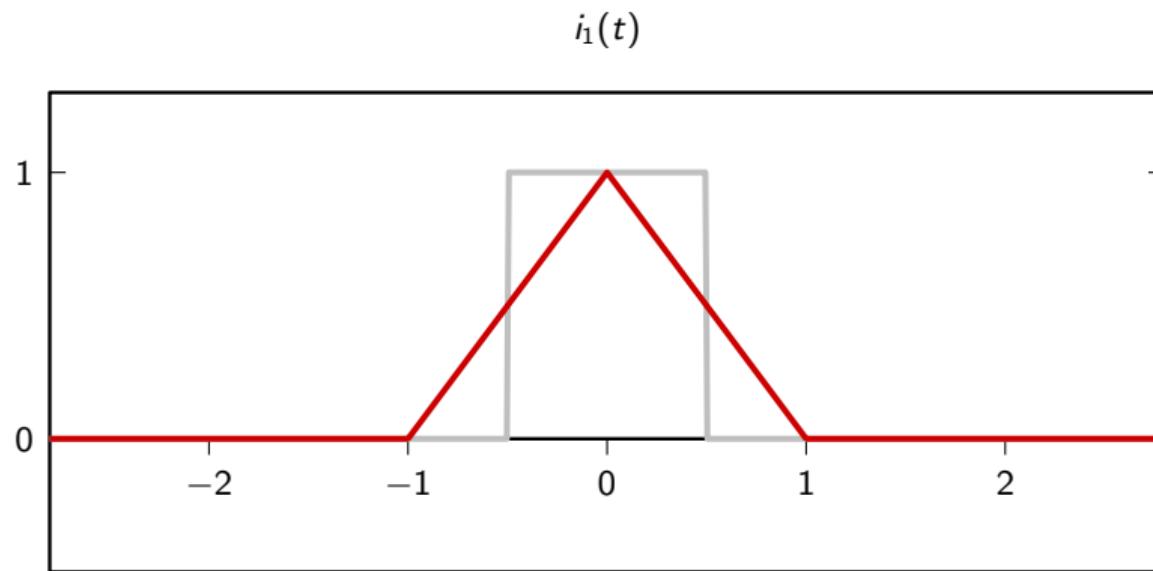
Practical interpolation

$$x_c(t) = \sum_{n=-\infty}^{\infty} x[n] i \left(\frac{t - nT_s}{T_s} \right)$$

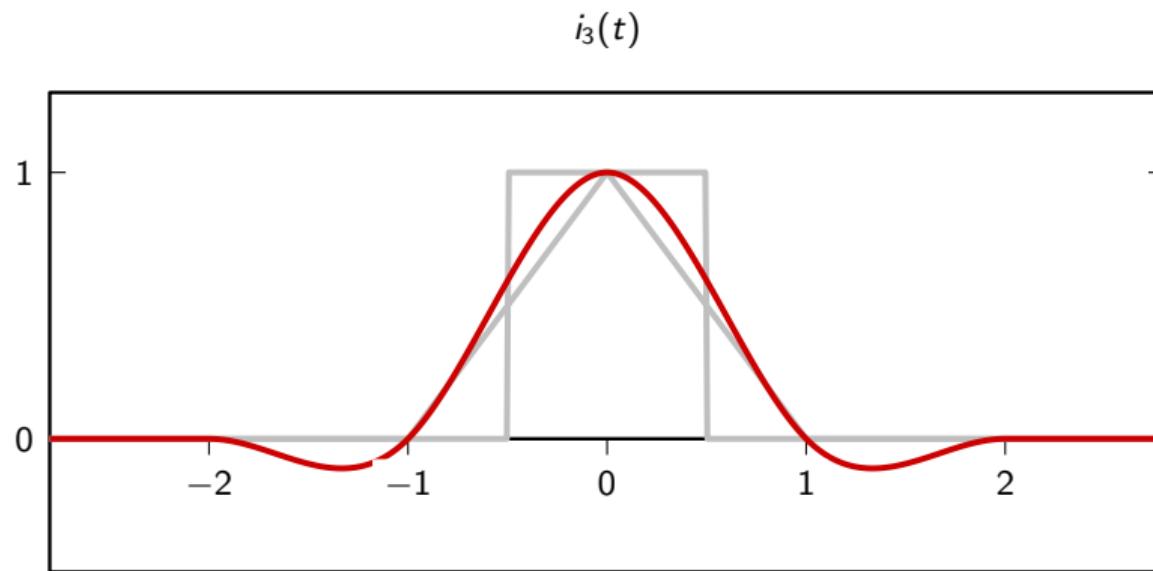
Local interpolators



Local interpolators



Local interpolators



Spectral representation (I)

$$\begin{aligned} X_c(f) &= \int_{-\infty}^{\infty} x_c(t) e^{-j2\pi ft} dt \\ &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x[n] i\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \\ &= \sum_{n=-\infty}^{\infty} x[n] \int_{-\infty}^{\infty} i\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \end{aligned}$$

Spectral representation (I)

$$\begin{aligned} X_c(f) &= \int_{-\infty}^{\infty} x_c(t) e^{-j2\pi ft} dt \\ &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x[n] i\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \\ &= \sum_{n=-\infty}^{\infty} x[n] \int_{-\infty}^{\infty} i\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \end{aligned}$$

Spectral representation (I)

$$\begin{aligned} X_c(f) &= \int_{-\infty}^{\infty} x_c(t) e^{-j2\pi ft} dt \\ &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x[n] i\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \\ &= \sum_{n=-\infty}^{\infty} x[n] \int_{-\infty}^{\infty} i\left(\frac{t - nT_s}{T_s}\right) e^{-j2\pi ft} dt \end{aligned}$$

Fourier Transform properties

$$\text{FT} \{x(t/a)\} = aX(af)$$

$$\text{FT} \{x(t - b)\} = X(f) e^{-j2\pi bf}$$

$$\int_{-\infty}^{\infty} i \left(\frac{t - nT_s}{T_s} \right) e^{-j2\pi ft} dt = T_s I(T_s f) e^{-j2\pi nT_s f}$$

Spectral representation (II)

$$X(f) = T_s \sum_{n=-\infty}^{\infty} x[n] I(T_s f) e^{-j2\pi n T_s f}$$

$$= \frac{1}{F_s} I\left(\frac{f}{F_s}\right) \sum_{n=-\infty}^{\infty} x[n] e^{-j2\pi n f / F_s}$$

$$= \frac{1}{F_s} I\left(\frac{f}{F_s}\right) X(e^{j2\pi f / F_s})$$

Spectral representation (II)

$$\begin{aligned} X(f) &= T_s \sum_{n=-\infty}^{\infty} x[n] I(T_s f) e^{-j2\pi n T_s f} \\ &= \frac{1}{F_s} I\left(\frac{f}{F_s}\right) \sum_{n=-\infty}^{\infty} x[n] e^{-j2\pi n f / F_s} \\ &= \frac{1}{F_s} I\left(\frac{f}{F_s}\right) X(e^{j2\pi f / F_s}) \end{aligned}$$

Spectral representation (II)

$$\begin{aligned} X(f) &= T_s \sum_{n=-\infty}^{\infty} x[n] I(T_s f) e^{-j2\pi n T_s f} \\ &= \frac{1}{F_s} I\left(\frac{f}{F_s}\right) \sum_{n=-\infty}^{\infty} x[n] e^{-j2\pi n f / F_s} \\ &= \frac{1}{F_s} I\left(\frac{f}{F_s}\right) X(e^{j2\pi f / F_s}) \end{aligned}$$

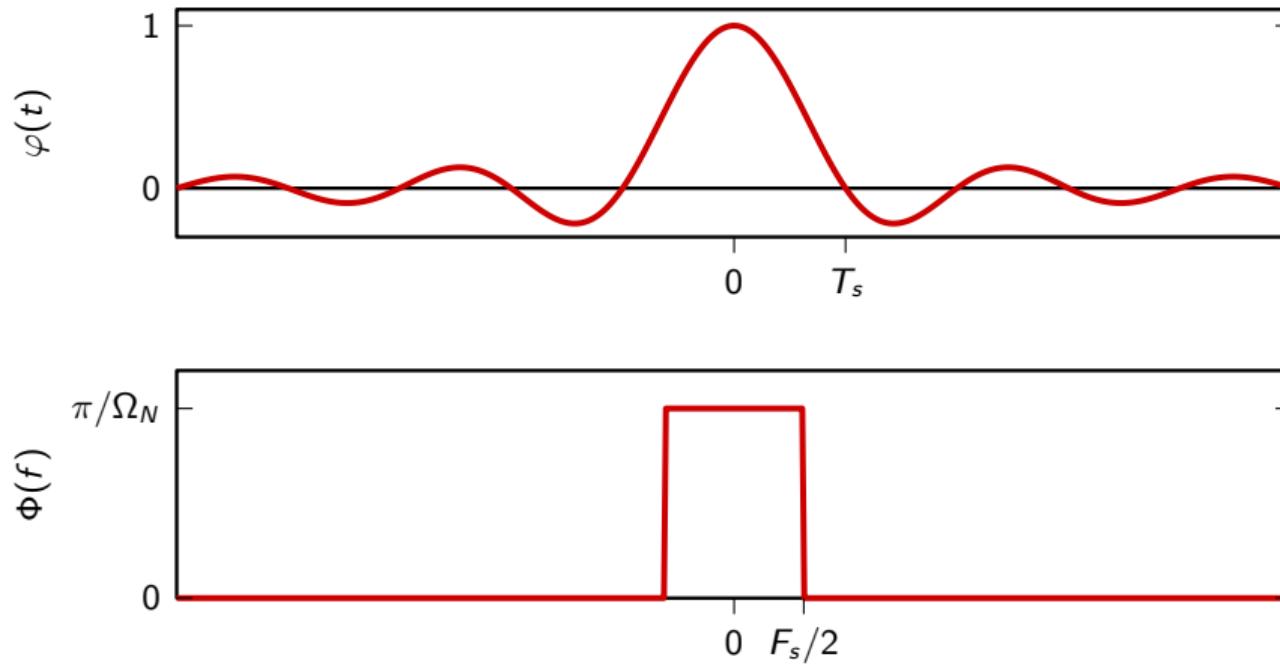
sinc interpolation

$$i(t) = \text{sinc}(t)$$

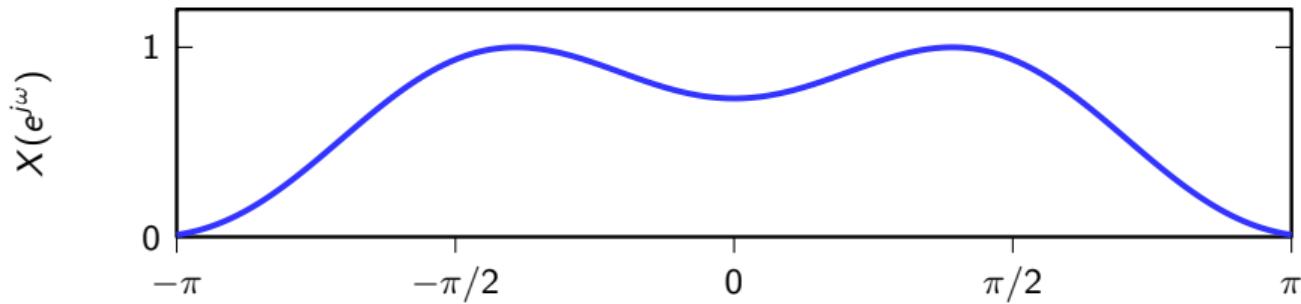
$$I(f) = \text{rect}(f)$$

$$X(f) = \frac{1}{F_s} \text{ rect}\left(\frac{f}{F_s}\right) X(e^{j2\pi f/F_s})$$

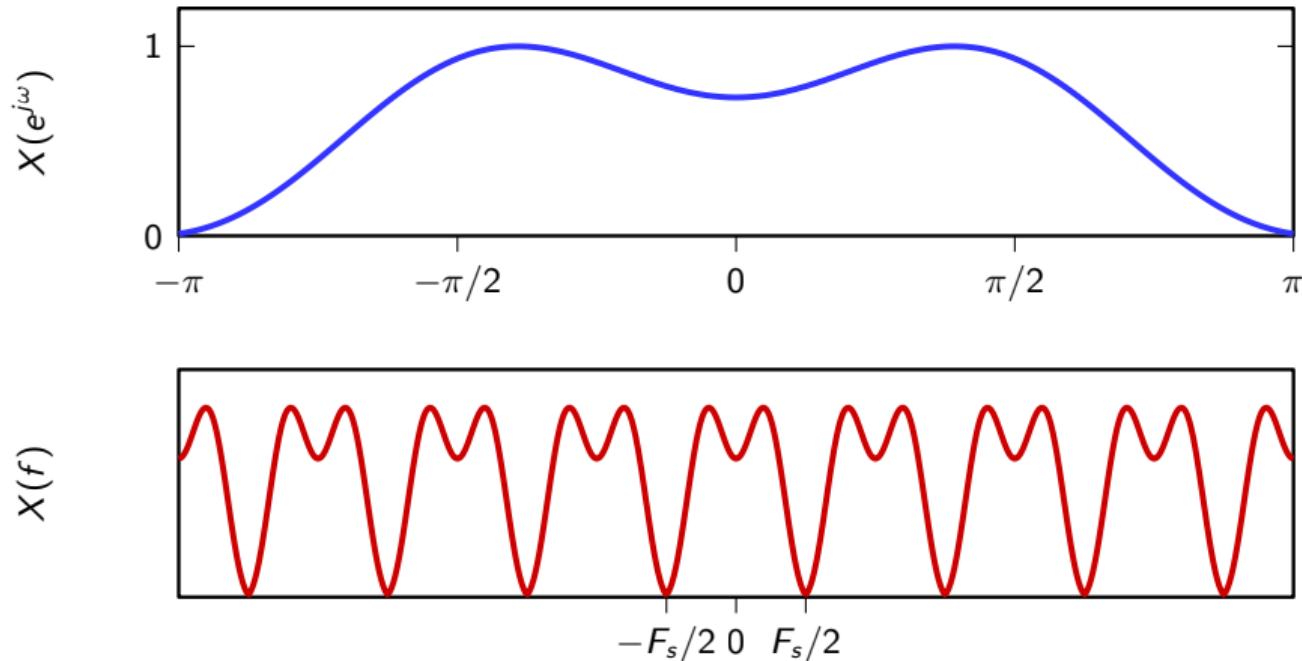
sinc interpolation



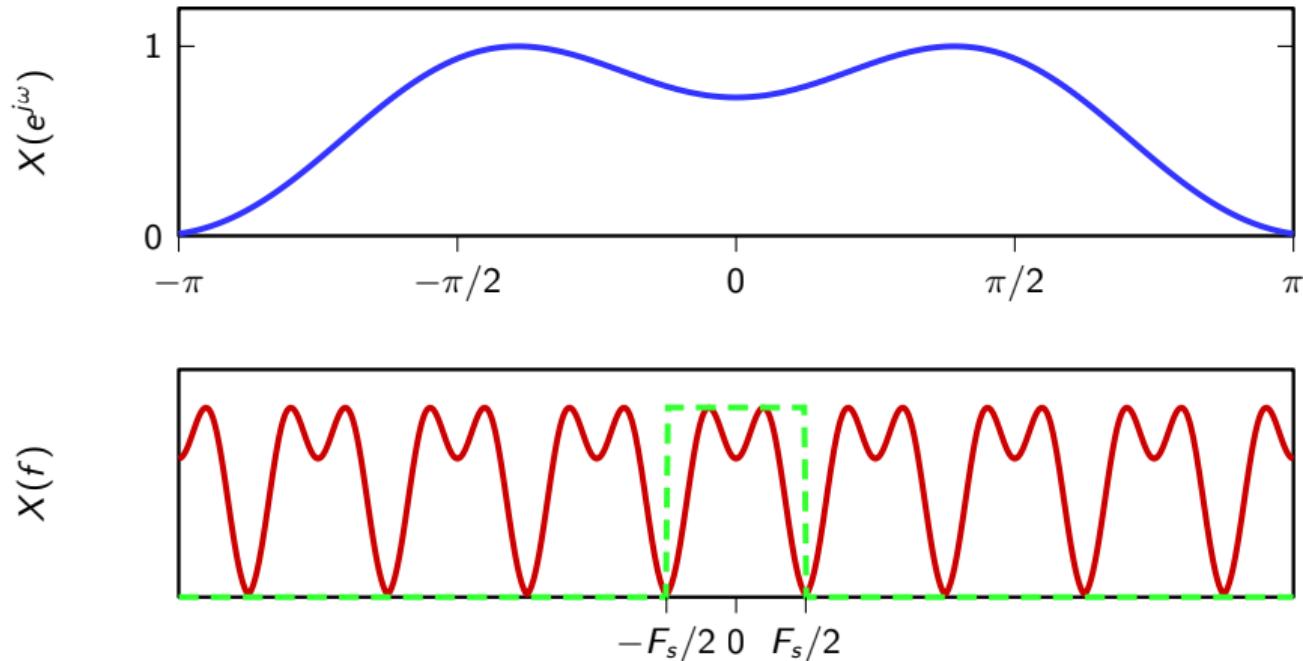
sinc interpolation



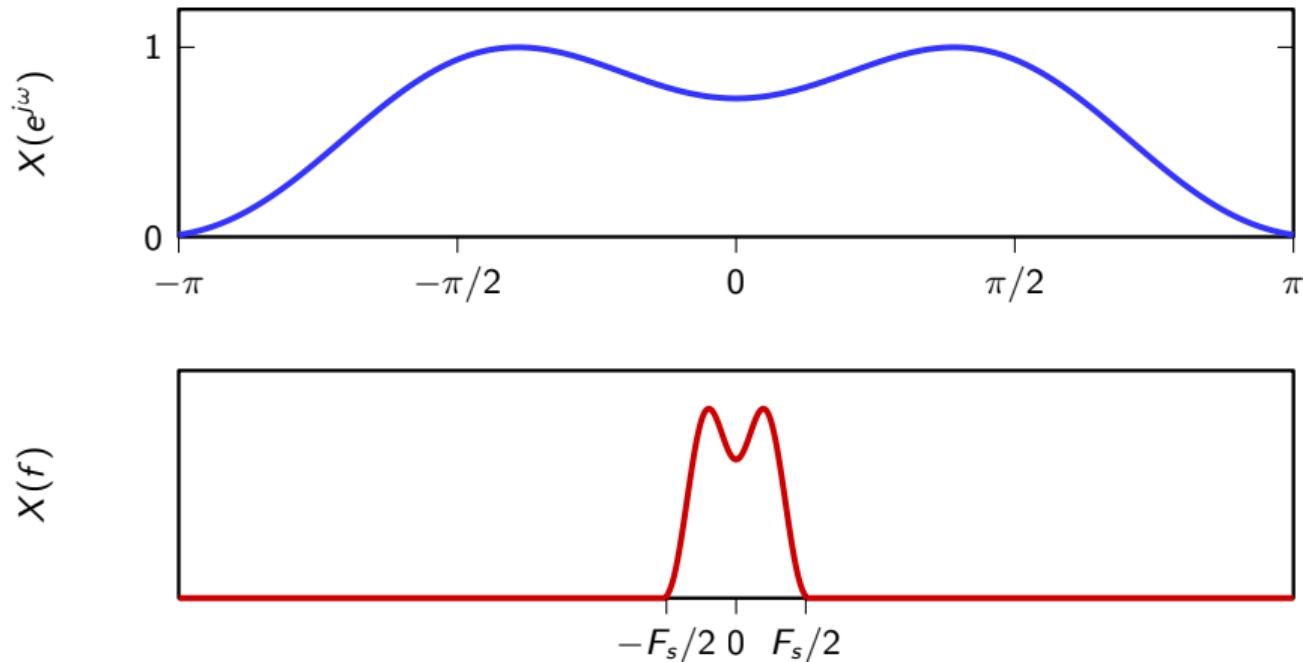
sinc interpolation



sinc interpolation



sinc interpolation



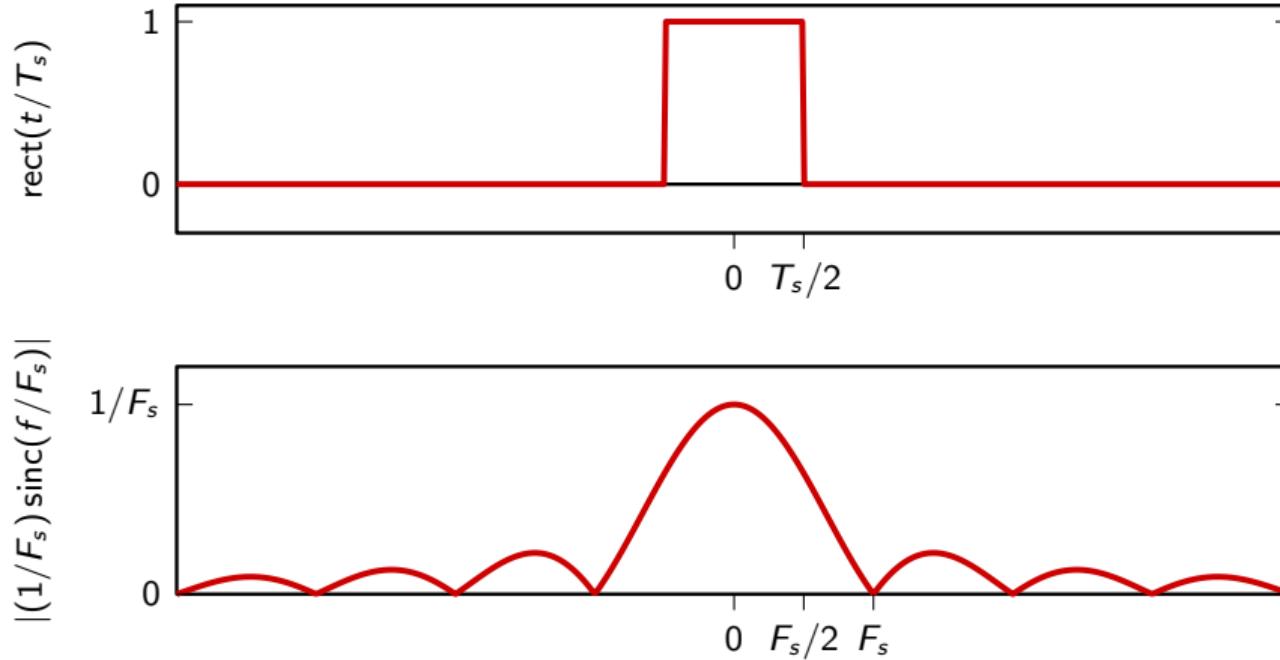
zero-order hold

$$i(t) = \text{rect}(t)$$

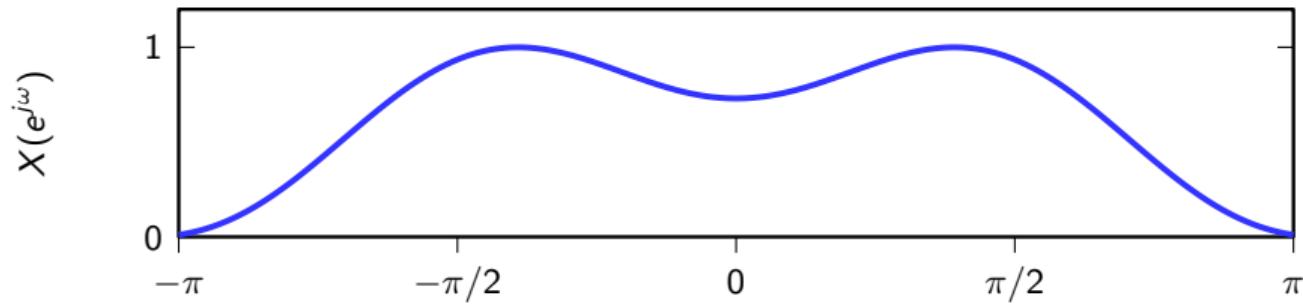
$$I(f) = \text{sinc}(f)$$

$$\begin{aligned} X(f) &= \frac{1}{F_s} I\left(\frac{f}{F_s}\right) X(e^{j2\pi f/F_s}) \\ &= \frac{1}{F_s} \text{sinc}\left(\frac{f}{F_s}\right) X(e^{j2\pi f/F_s}) \end{aligned}$$

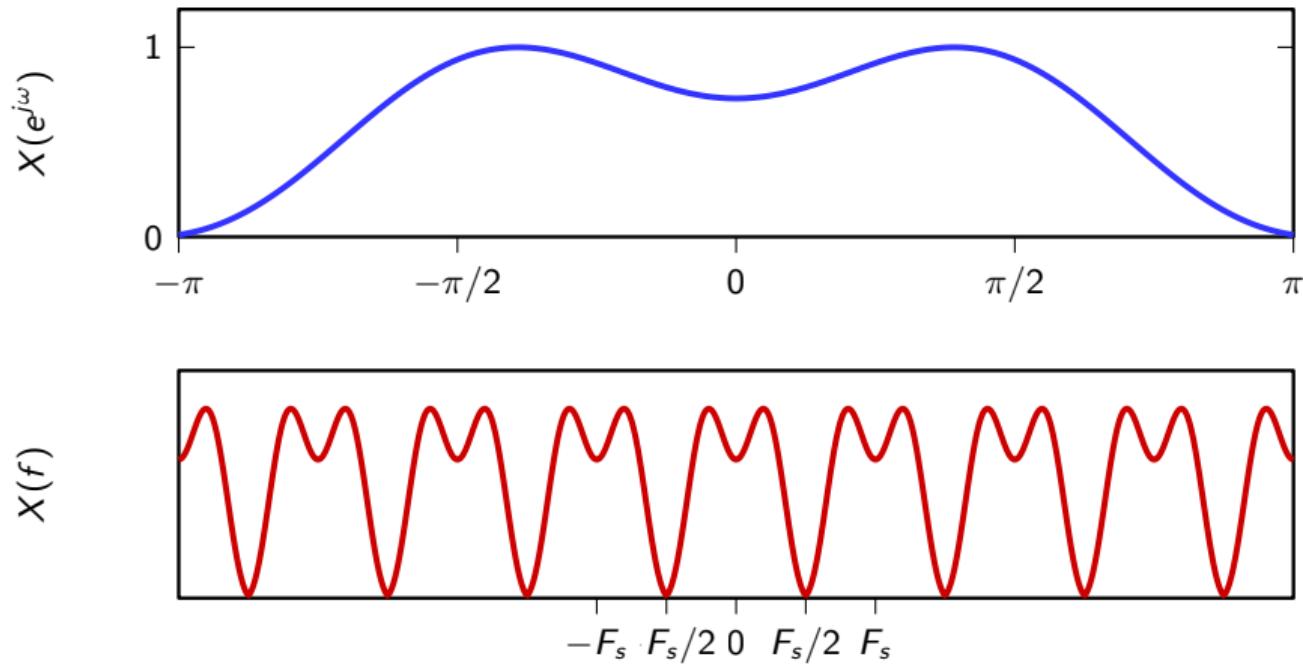
zero-order hold



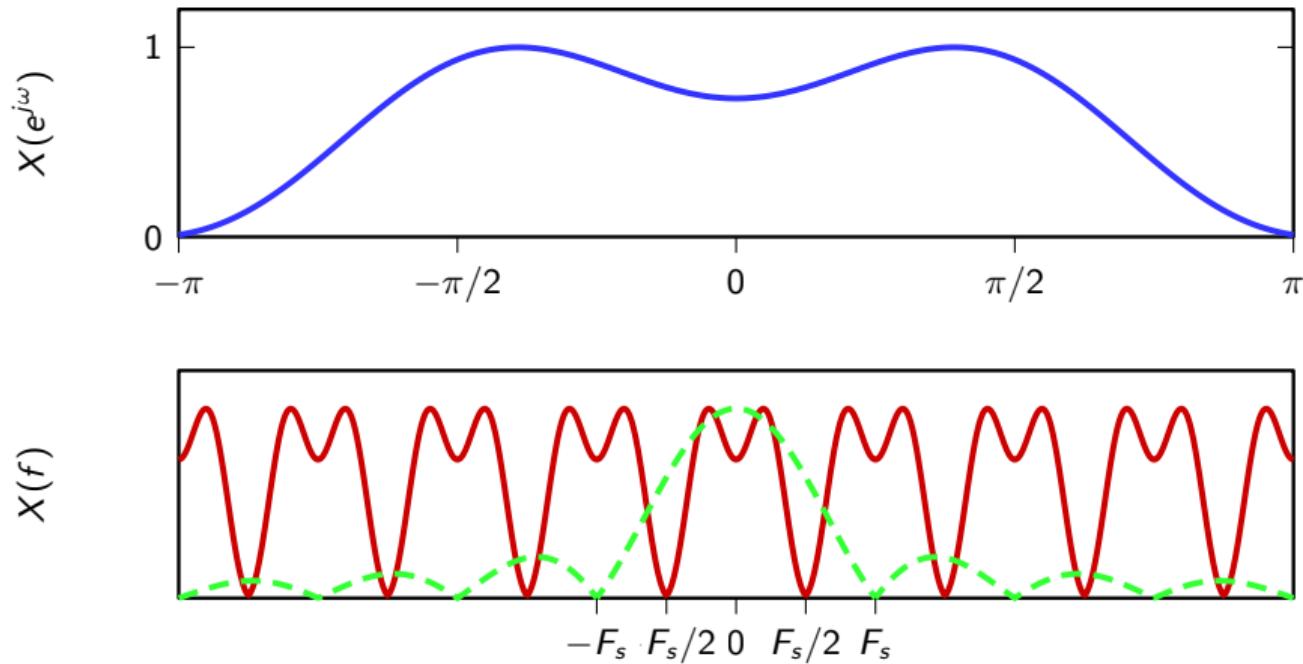
Zero-order hold



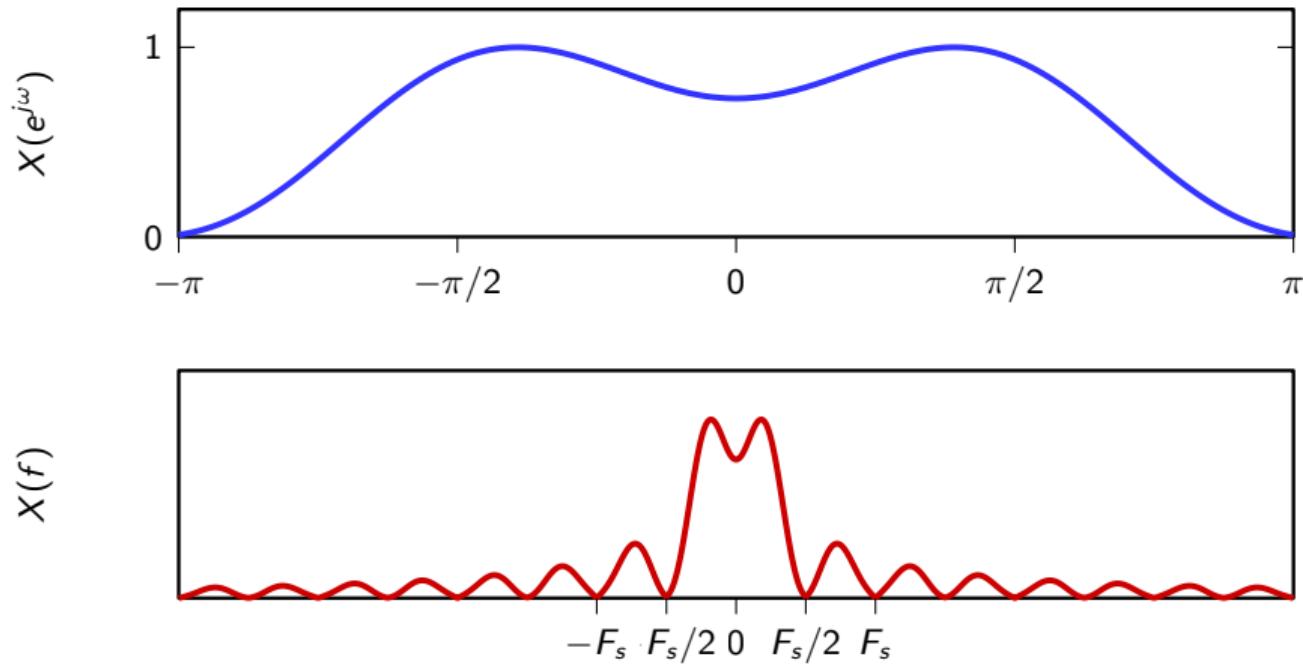
Zero-order hold



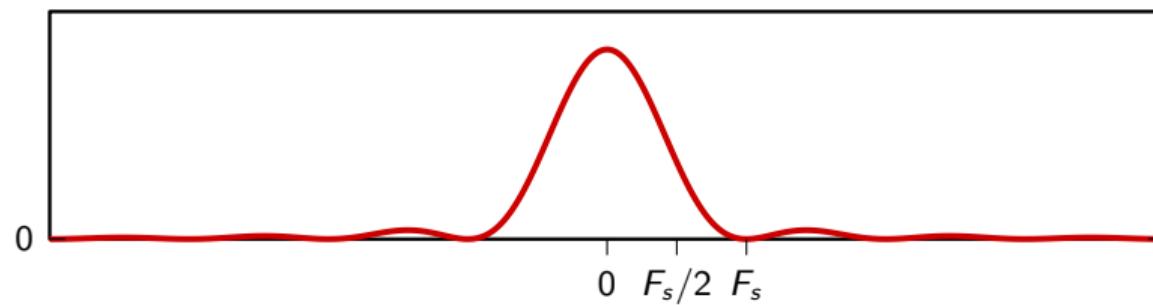
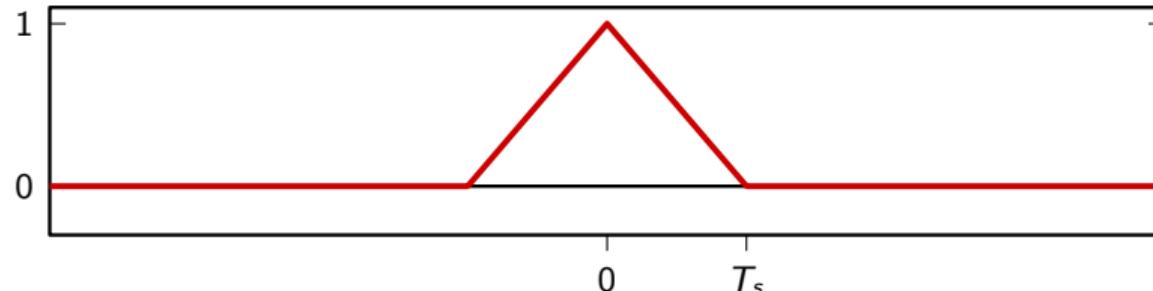
Zero-order hold



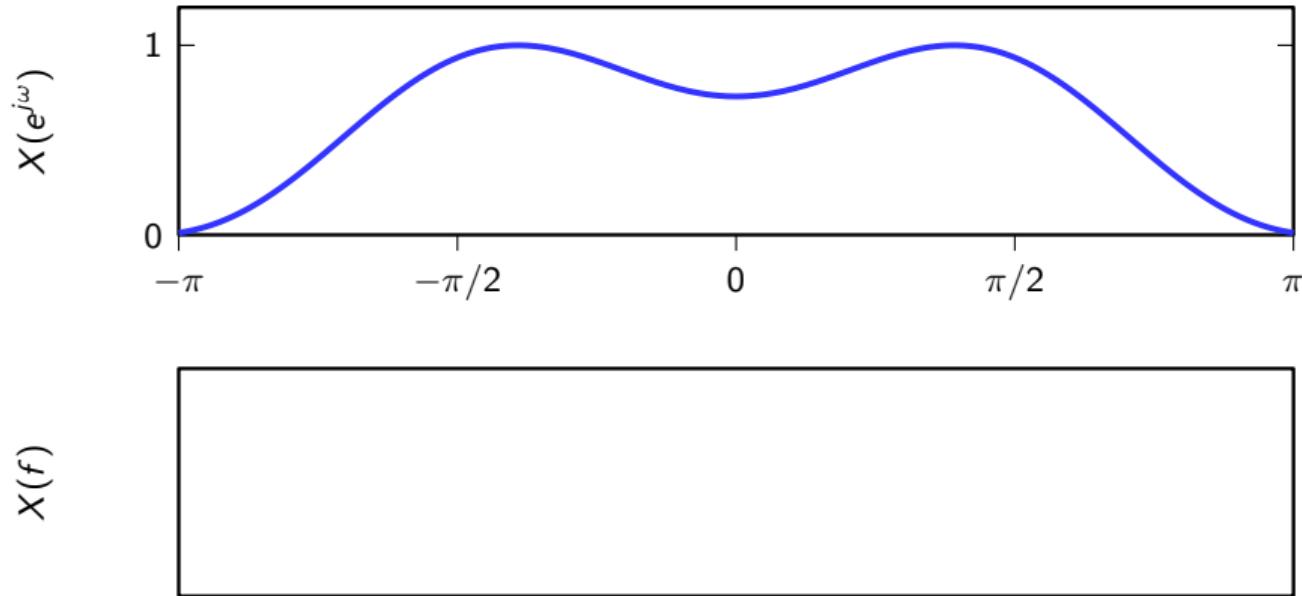
Zero-order hold



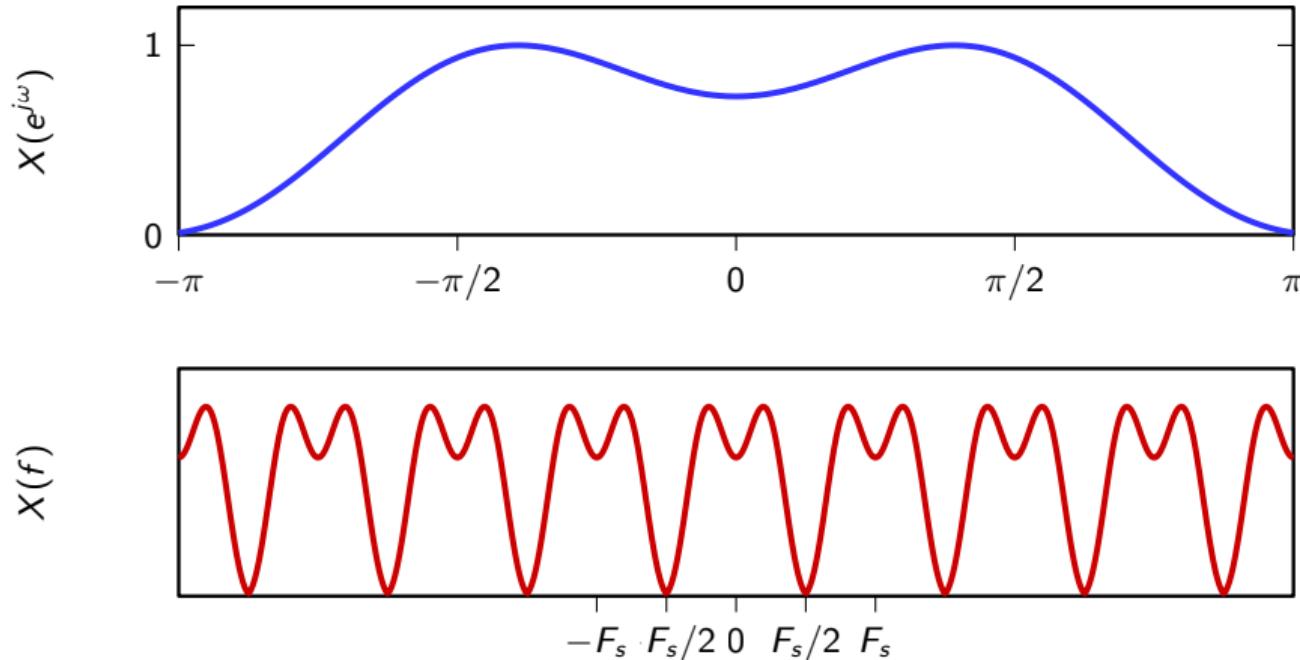
first-order interpolator



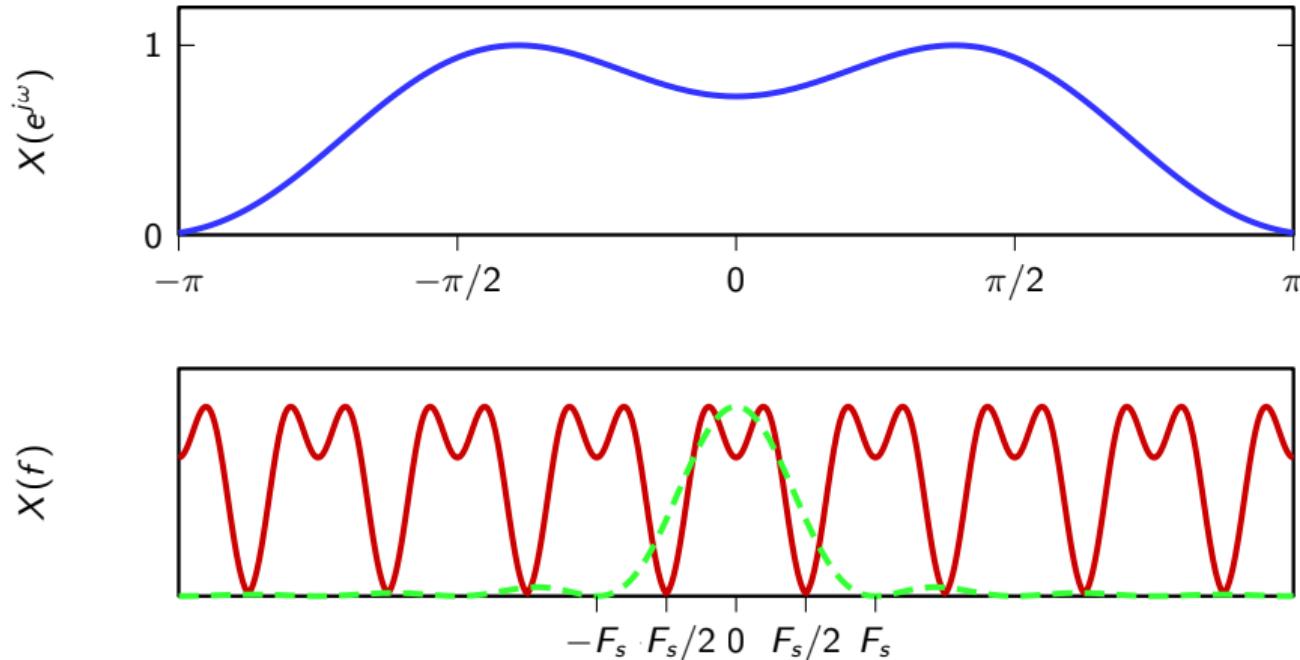
first-order interpolator



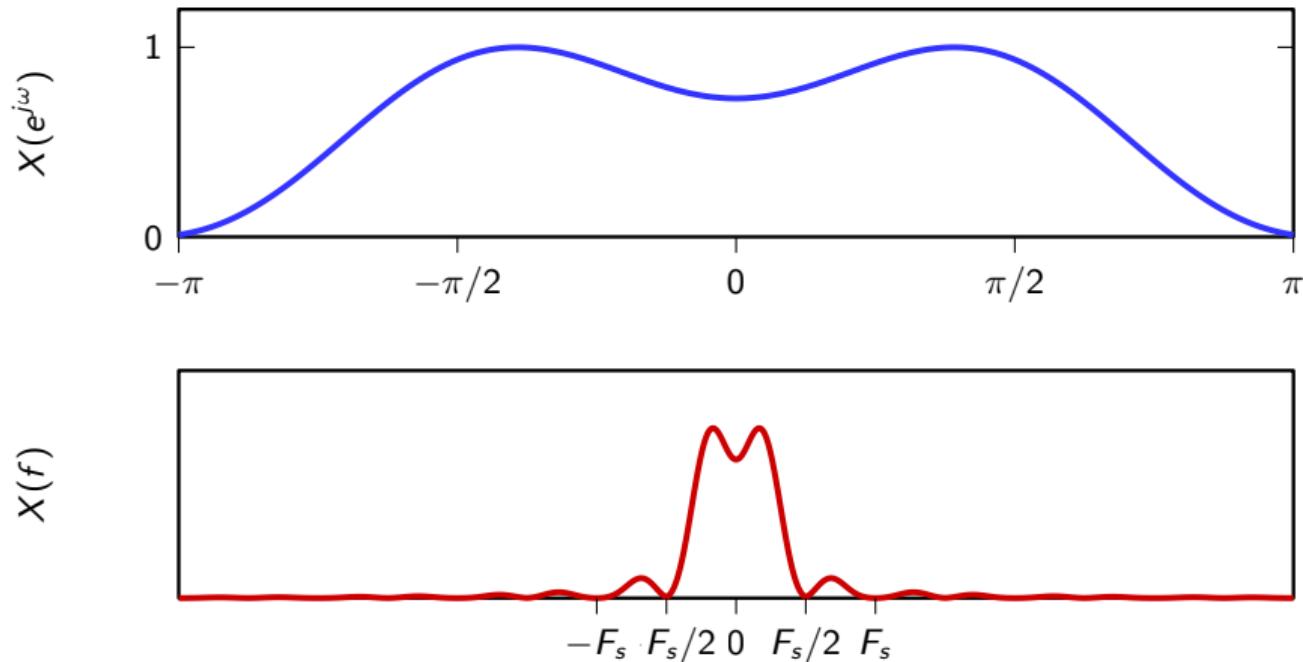
first-order interpolator



first-order interpolator



first-order interpolator



Bandpass Sampling

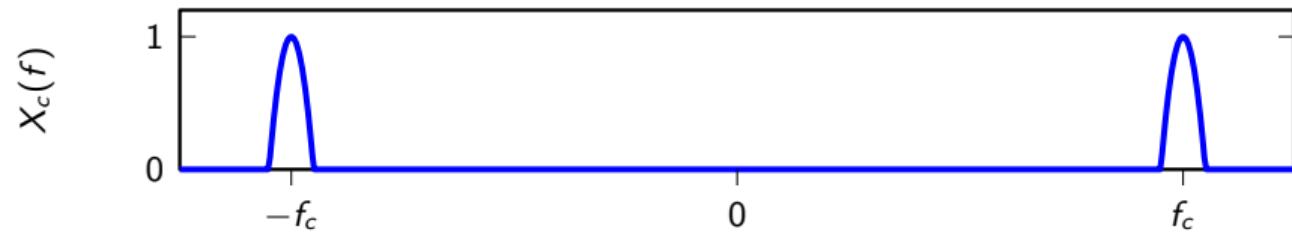
sampling theorem gives a *sufficient* condition

- ▶ in theory, $F_s > 2f_{\max}$
- ▶ what if signal is bandpass?

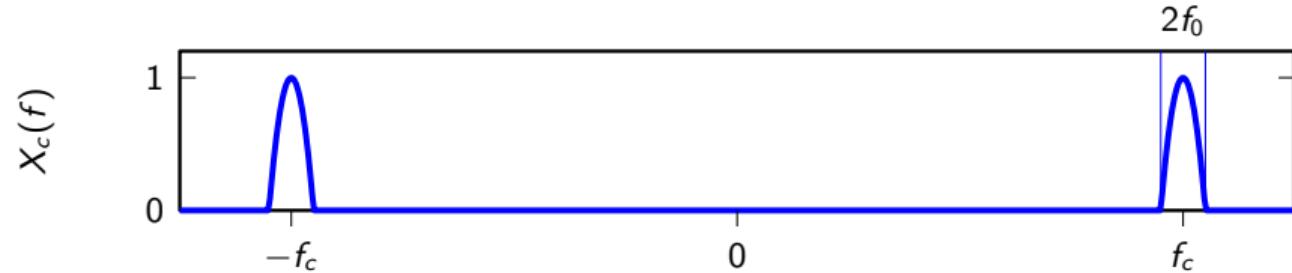
sampling theorem gives a *sufficient* condition

- ▶ in theory, $F_s > 2f_{\max}$
- ▶ what if signal is bandpass?

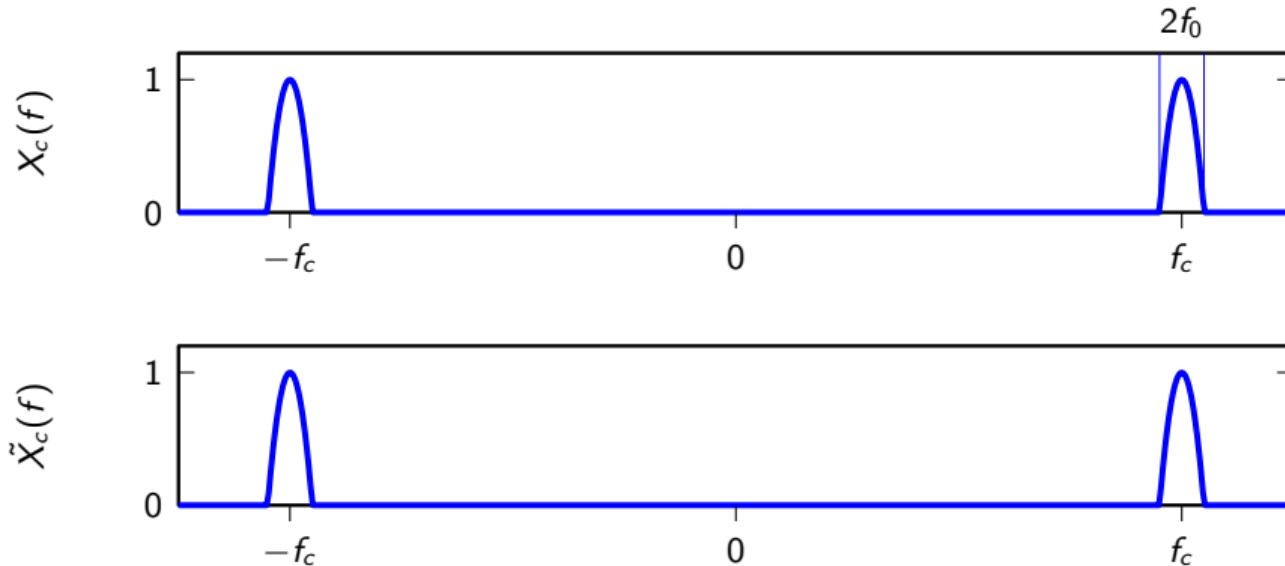
Bandpass sampling



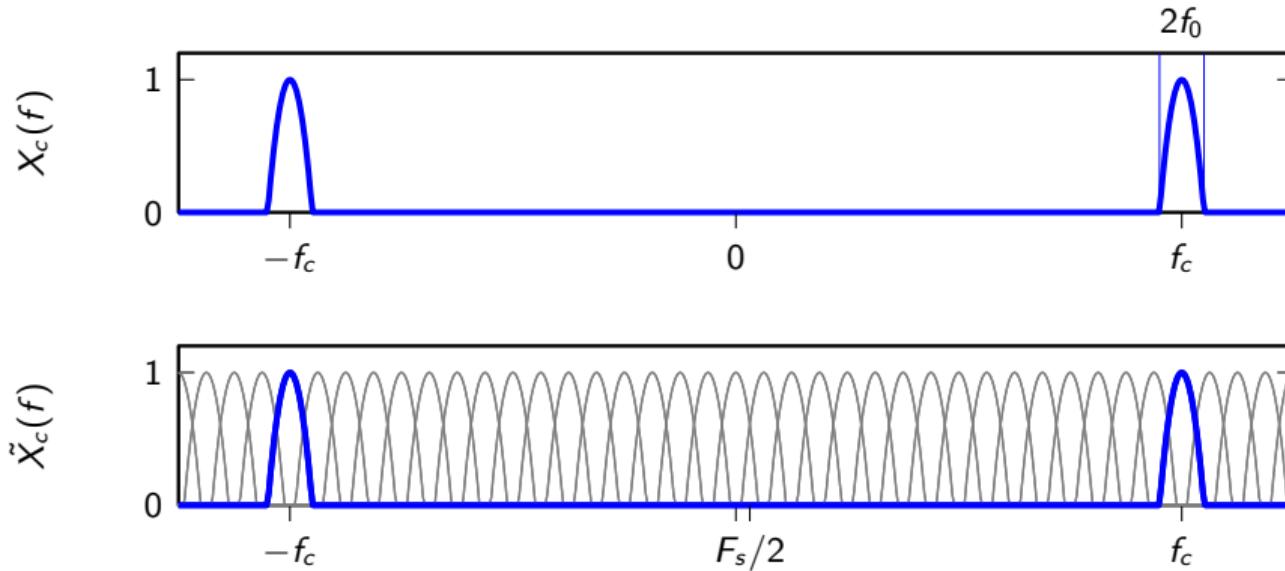
Bandpass sampling



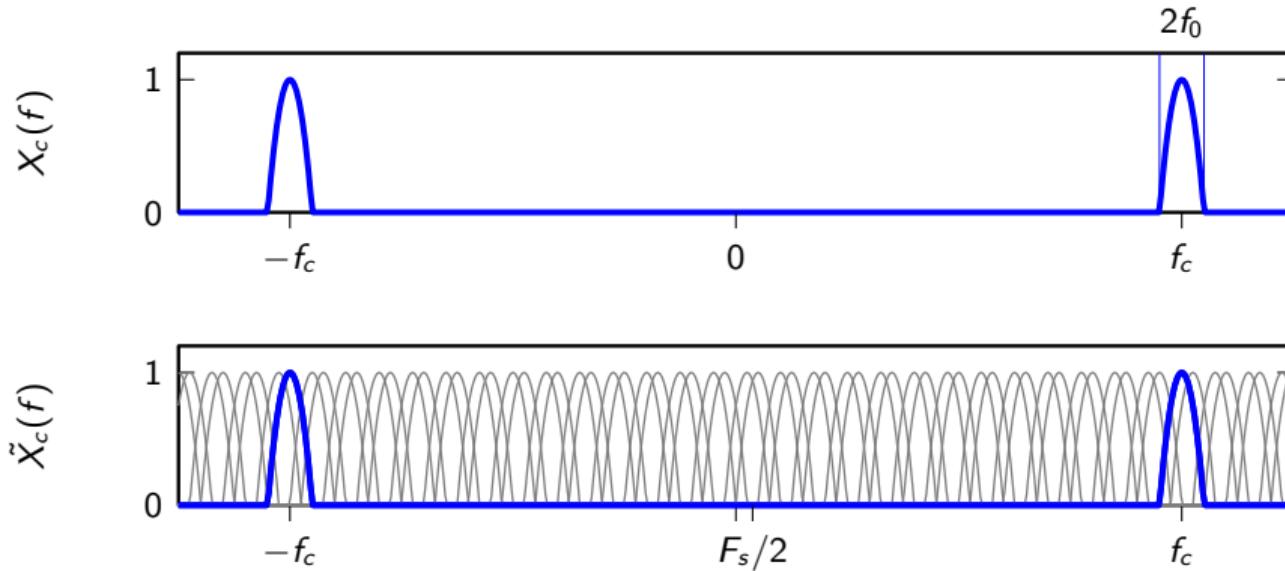
Bandpass sampling



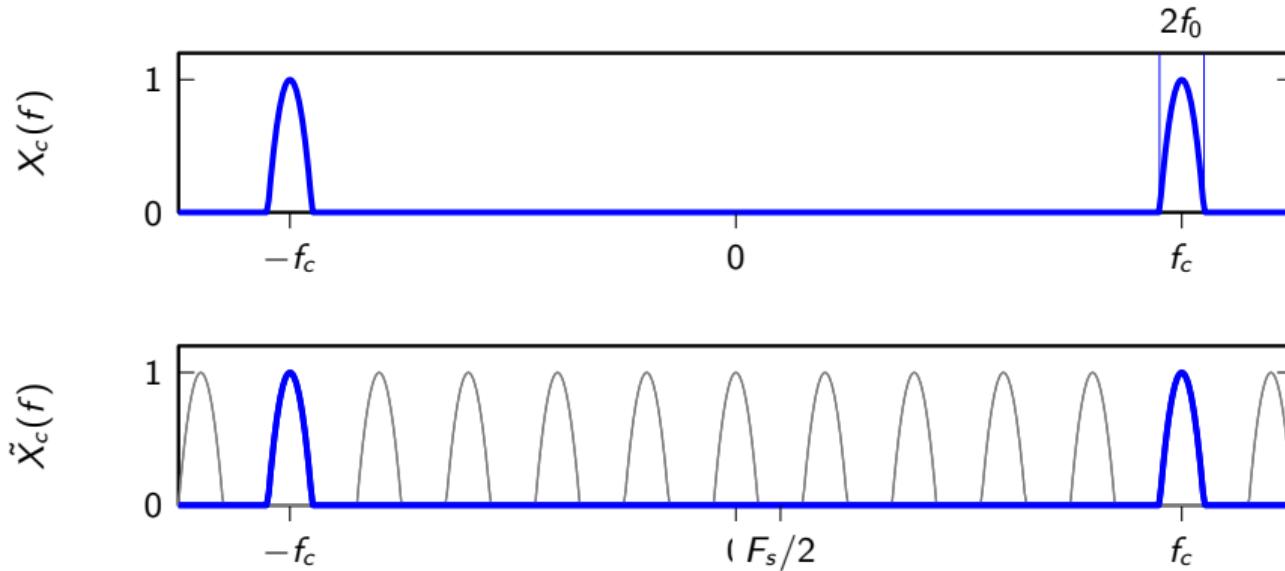
Bandpass sampling



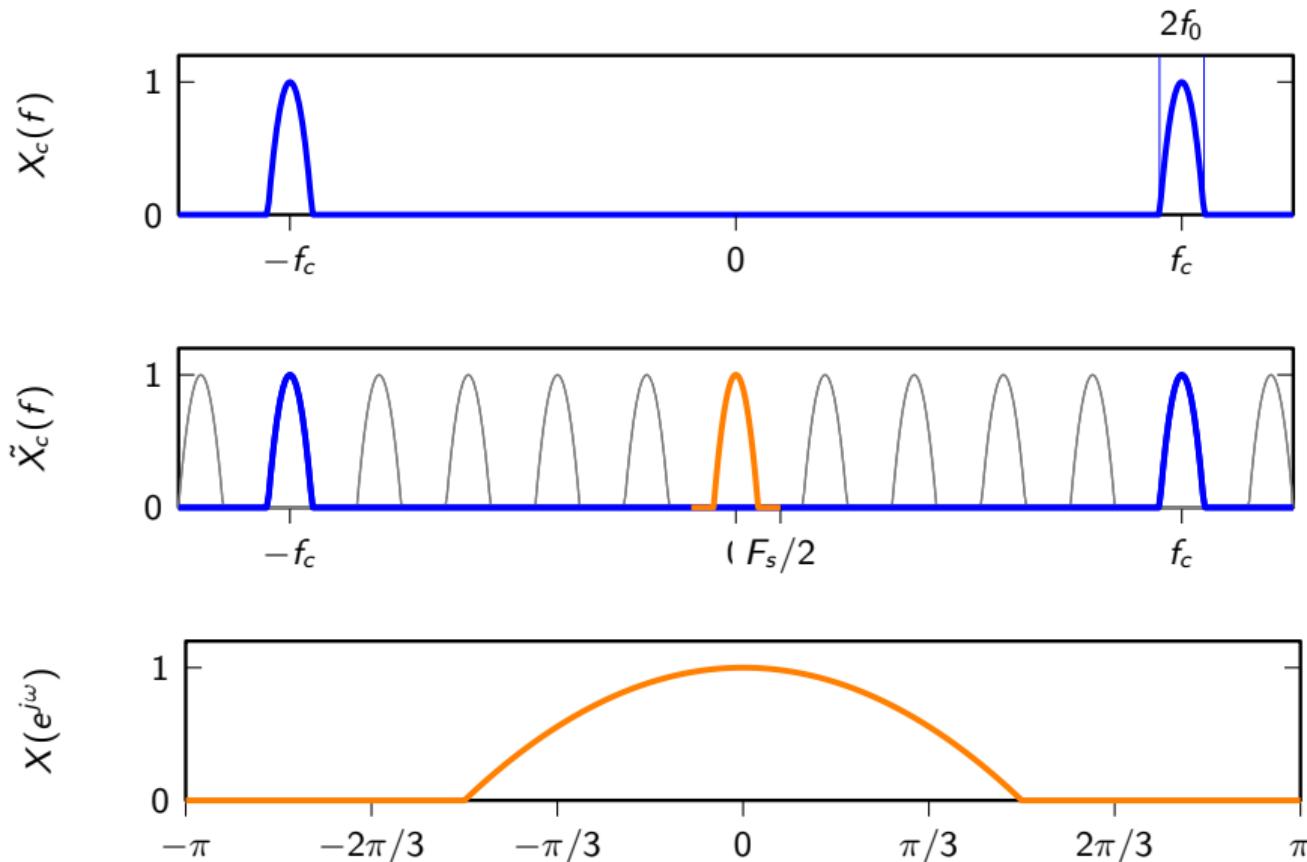
Bandpass sampling



Bandpass sampling



Bandpass sampling



Bandpass sampling conditions

- ▶ bandpass signal: $X(f) = 0$ for $|f - f_c| > f_0$
- ▶ no alias requires at least: $F_s \geq 2f_0$
- ▶ baseband condition: $F_s = 2f_c/k$ for some $k \in \mathbb{N}$
- ▶ for $T_s = 1/F_s$, $x[n] = x(nT_s)$ is an alias-free, baseband DT version of $x(t)$

Bandpass sampling conditions

- ▶ bandpass signal: $X(f) = 0$ for $|f - f_c| > f_0$
- ▶ no alias requires at least: $F_s \geq 2f_0$
- ▶ baseband condition: $F_s = 2f_c/k$ for some $k \in \mathbb{N}$
- ▶ for $T_s = 1/F_s$, $x[n] = x(nT_s)$ is an alias-free, baseband DT version of $x(t)$

Bandpass sampling conditions

- ▶ bandpass signal: $X(f) = 0$ for $|f - f_c| > f_0$
- ▶ no alias requires at least: $F_s \geq 2f_0$
- ▶ baseband condition: $F_s = 2f_c/k$ for some $k \in \mathbb{N}$

- ▶ for $T_s = 1/F_s$, $x[n] = x(nT_s)$ is an alias-free, baseband DT version of $x(t)$

Bandpass sampling conditions

- ▶ bandpass signal: $X(f) = 0$ for $|f - f_c| > f_0$
- ▶ no alias requires at least: $F_s \geq 2f_0$
- ▶ baseband condition: $F_s = 2f_c/k$ for some $k \in \mathbb{N}$

- ▶ for $T_s = 1/F_s$, $x[n] = x(nT_s)$ is an alias-free, baseband DT version of $x(t)$

Example: AM Channel

- ▶ AM radio band: 500kHz to 1.6MHz
- ▶ channel width is 9kHz, i.e. $f_0 = 4.5\text{KHz}$
- ▶ take a channel at $f_c = 1.5\text{MHz}$
- ▶ in theory: $F_s \geq 2 * 1,504,500\text{Hz}$, $T_s < 10^{-6}$ seconds!
- ▶ antialias: $F_s \geq 2f_0 \Rightarrow F_s \geq 9\text{KHz}$
- ▶ baseband: $kF_s/2 = 1500\text{kHz}$
- ▶ pick $k = 300$
- ▶ $F_s = 10\text{KHz}$, $T_s = 10^{-4}$

Example: AM Channel

- ▶ AM radio band: 500kHz to 1.6MHz
- ▶ channel width is 9kHz, i.e. $f_0 = 4.5\text{KHz}$
- ▶ take a channel at $f_c = 1.5\text{MHz}$
- ▶ in theory: $F_s \geq 2 * 1,504,500\text{Hz}$, $T_s < 10^{-6}$ seconds!
- ▶ antialias: $F_s \geq 2f_0 \Rightarrow F_s \geq 9\text{KHz}$
- ▶ baseband: $kF_s/2 = 1500\text{kHz}$
- ▶ pick $k = 300$
- ▶ $F_s = 10\text{KHz}$, $T_s = 10^{-4}$

Example: AM Channel

- ▶ AM radio band: 500kHz to 1.6MHz
- ▶ channel width is 9kHz, i.e. $f_0 = 4.5\text{KHz}$
- ▶ take a channel at $f_c = 1.5\text{MHz}$
- ▶ in theory: $F_s \geq 2 * 1,504,500\text{Hz}$, $T_s < 10^{-6}$ seconds!
- ▶ antialias: $F_s \geq 2f_0 \Rightarrow F_s \geq 9\text{KHz}$
- ▶ baseband: $kF_s/2 = 1500\text{kHz}$
- ▶ pick $k = 300$
- ▶ $F_s = 10\text{KHz}$, $T_s = 10^{-4}$

Example: AM Channel

- ▶ AM radio band: 500kHz to 1.6MHz
- ▶ channel width is 9kHz, i.e. $f_0 = 4.5\text{KHz}$
- ▶ take a channel at $f_c = 1.5\text{MHz}$
- ▶ in theory: $F_s \geq 2 * 1,504,500\text{Hz}$, $T_s < 10^{-6}$ seconds!

- ▶ antialias: $F_s \geq 2f_0 \Rightarrow F_s \geq 9\text{KHz}$
- ▶ baseband: $kF_s/2 = 1500\text{kHz}$

- ▶ pick $k = 300$
- ▶ $F_s = 10\text{KHz}$, $T_s = 10^{-4}$

Example: AM Channel

- ▶ AM radio band: 500kHz to 1.6MHz
- ▶ channel width is 9kHz, i.e. $f_0 = 4.5\text{KHz}$
- ▶ take a channel at $f_c = 1.5\text{MHz}$
- ▶ in theory: $F_s \geq 2 * 1,504,500\text{Hz}$, $T_s < 10^{-6}$ seconds!
- ▶ antialias: $F_s \geq 2f_0 \Rightarrow F_s \geq 9\text{KHz}$
- ▶ baseband: $kF_s/2 = 1500\text{kHz}$
- ▶ pick $k = 300$
- ▶ $F_s = 10\text{KHz}$, $T_s = 10^{-4}$

Example: AM Channel

- ▶ AM radio band: 500kHz to 1.6MHz
- ▶ channel width is 9kHz, i.e. $f_0 = 4.5\text{KHz}$
- ▶ take a channel at $f_c = 1.5\text{MHz}$
- ▶ in theory: $F_s \geq 2 * 1,504,500\text{Hz}$, $T_s < 10^{-6}$ seconds!
- ▶ antialias: $F_s \geq 2f_0 \Rightarrow F_s \geq 9\text{KHz}$
- ▶ baseband: $kF_s/2 = 1500\text{kHz}$
- ▶ pick $k = 300$
- ▶ $F_s = 10\text{KHz}$, $T_s = 10^{-4}$

Example: AM Channel

- ▶ AM radio band: 500kHz to 1.6MHz
- ▶ channel width is 9kHz, i.e. $f_0 = 4.5\text{KHz}$
- ▶ take a channel at $f_c = 1.5\text{MHz}$
- ▶ in theory: $F_s \geq 2 * 1,504,500\text{Hz}$, $T_s < 10^{-6}$ seconds!
- ▶ antialias: $F_s \geq 2f_0 \Rightarrow F_s \geq 9\text{KHz}$
- ▶ baseband: $kF_s/2 = 1500\text{kHz}$
- ▶ pick $k = 300$
- ▶ $F_s = 10\text{KHz}$, $T_s = 10^{-4}$

Example: AM Channel

- ▶ AM radio band: 500kHz to 1.6MHz
- ▶ channel width is 9kHz, i.e. $f_0 = 4.5\text{KHz}$
- ▶ take a channel at $f_c = 1.5\text{MHz}$
- ▶ in theory: $F_s \geq 2 * 1,504,500\text{Hz}$, $T_s < 10^{-6}$ seconds!
- ▶ antialias: $F_s \geq 2f_0 \Rightarrow F_s \geq 9\text{KHz}$
- ▶ baseband: $kF_s/2 = 1500\text{kHz}$
- ▶ pick $k = 300$
- ▶ $F_s = 10\text{KHz}$, $T_s = 10^{-4}$

multirate signal processing

Overview

- ▶ why multirate?
- ▶ upsampling
- ▶ downsampling
- ▶ applications

Overview

- ▶ why multirate?
- ▶ upsampling
- ▶ downsampling
- ▶ applications

Overview

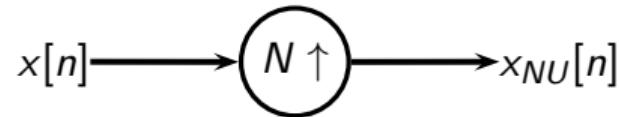
- ▶ why multirate?
- ▶ upsampling
- ▶ downsampling
- ▶ applications

Overview

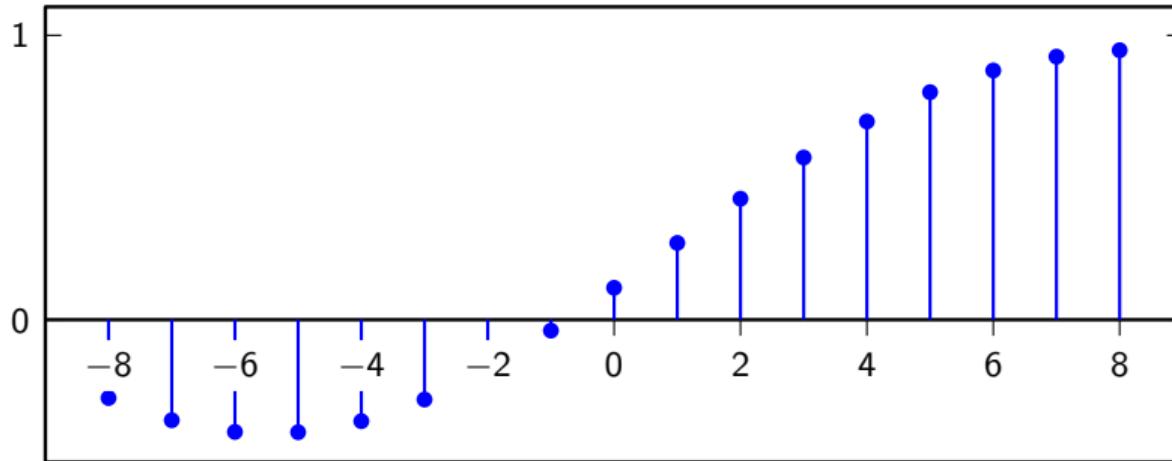
- ▶ why multirate?
- ▶ upsampling
- ▶ downsampling
- ▶ applications

Upsampling

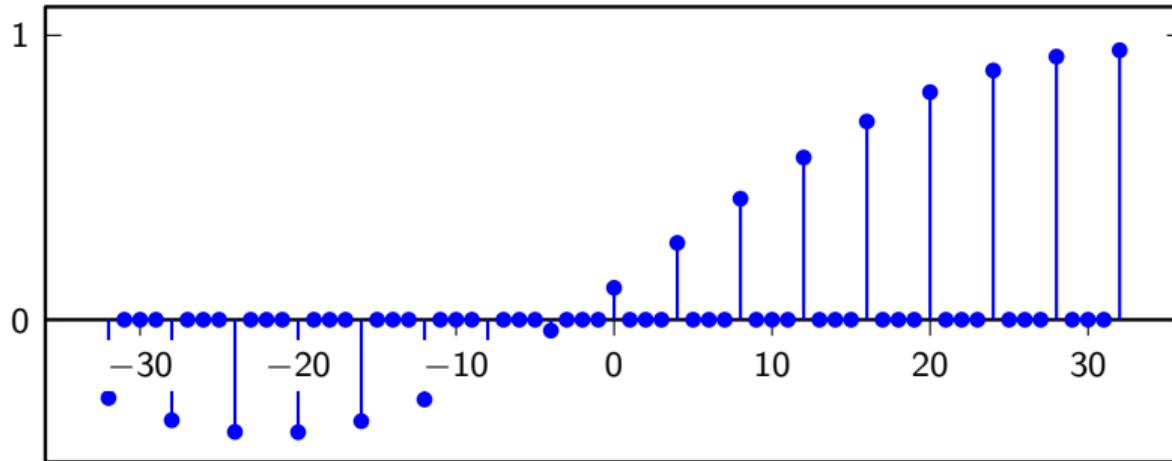
$$x_{NU}[n] = \begin{cases} x[k] & \text{for } n = kN, \quad k \in \mathbb{Z} \\ 0 & \text{otherwise.} \end{cases}$$



Example: upsampling by 4



Example: upsampling by 4



Spectral representation

$$\begin{aligned} X_{NU}(z) &= \sum_{k=-\infty}^{\infty} x_{NU}[k]z^{-k} \\ &= \sum_{k=-\infty}^{\infty} x[k]z^{-Nk} \\ &= X(z^N) \end{aligned}$$

$$X_{NU}(e^{j\omega}) = X(e^{j\omega N})$$

Spectral representation

$$\begin{aligned} X_{NU}(z) &= \sum_{k=-\infty}^{\infty} x_{NU}[k]z^{-k} \\ &= \sum_{k=-\infty}^{\infty} x[k]z^{-Nk} \\ &= X(z^N) \end{aligned}$$

$$X_{NU}(e^{j\omega}) = X(e^{j\omega N})$$

Spectral representation

$$\begin{aligned} X_{NU}(z) &= \sum_{k=-\infty}^{\infty} x_{NU}[k]z^{-k} \\ &= \sum_{k=-\infty}^{\infty} x[k]z^{-Nk} \\ &= X(z^N) \end{aligned}$$

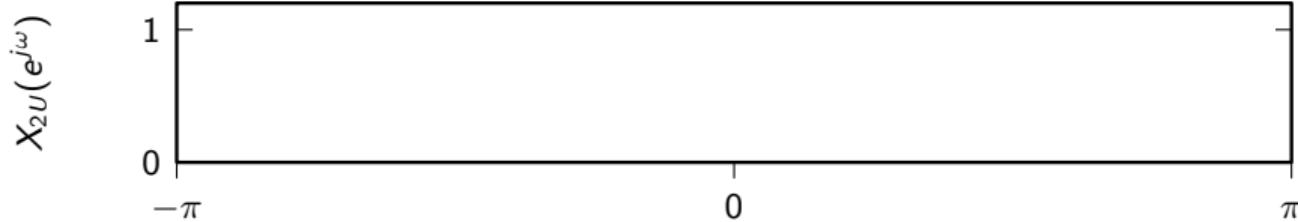
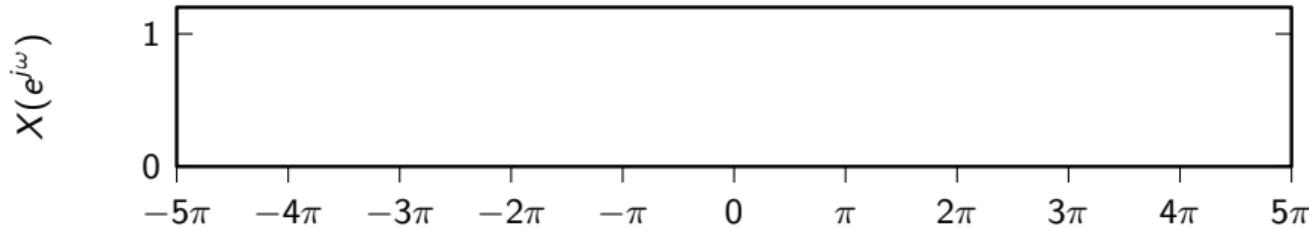
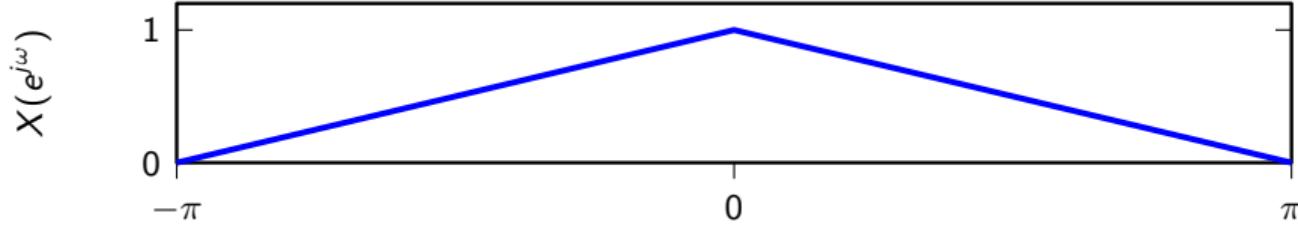
$$X_{NU}(e^{j\omega}) = X(e^{j\omega N})$$

Spectral representation

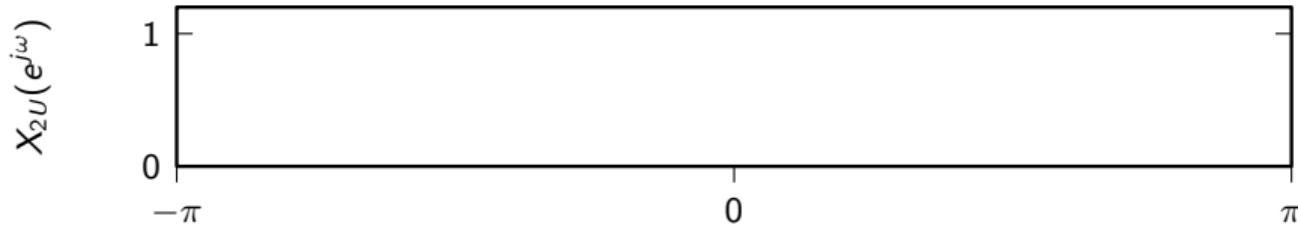
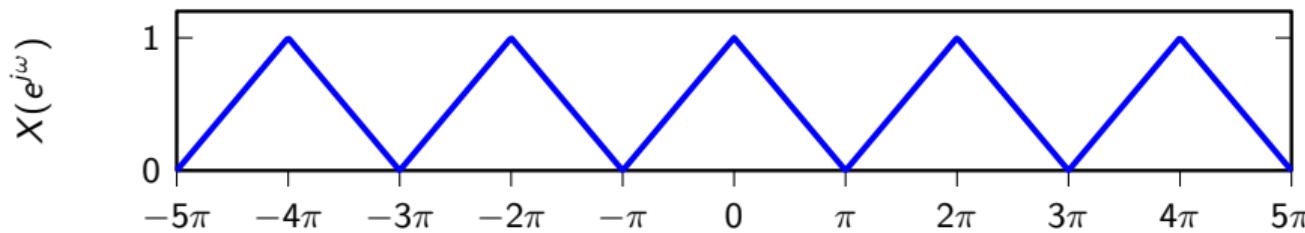
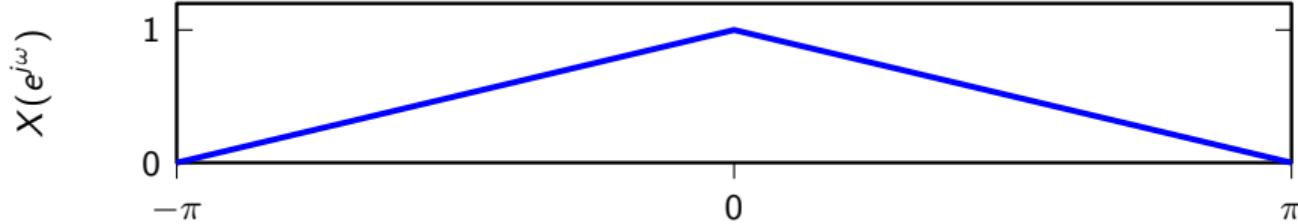
$$\begin{aligned} X_{NU}(z) &= \sum_{k=-\infty}^{\infty} x_{NU}[k]z^{-k} \\ &= \sum_{k=-\infty}^{\infty} x[k]z^{-Nk} \\ &= X(z^N) \end{aligned}$$

$$X_{NU}(e^{j\omega}) = X(e^{j\omega N})$$

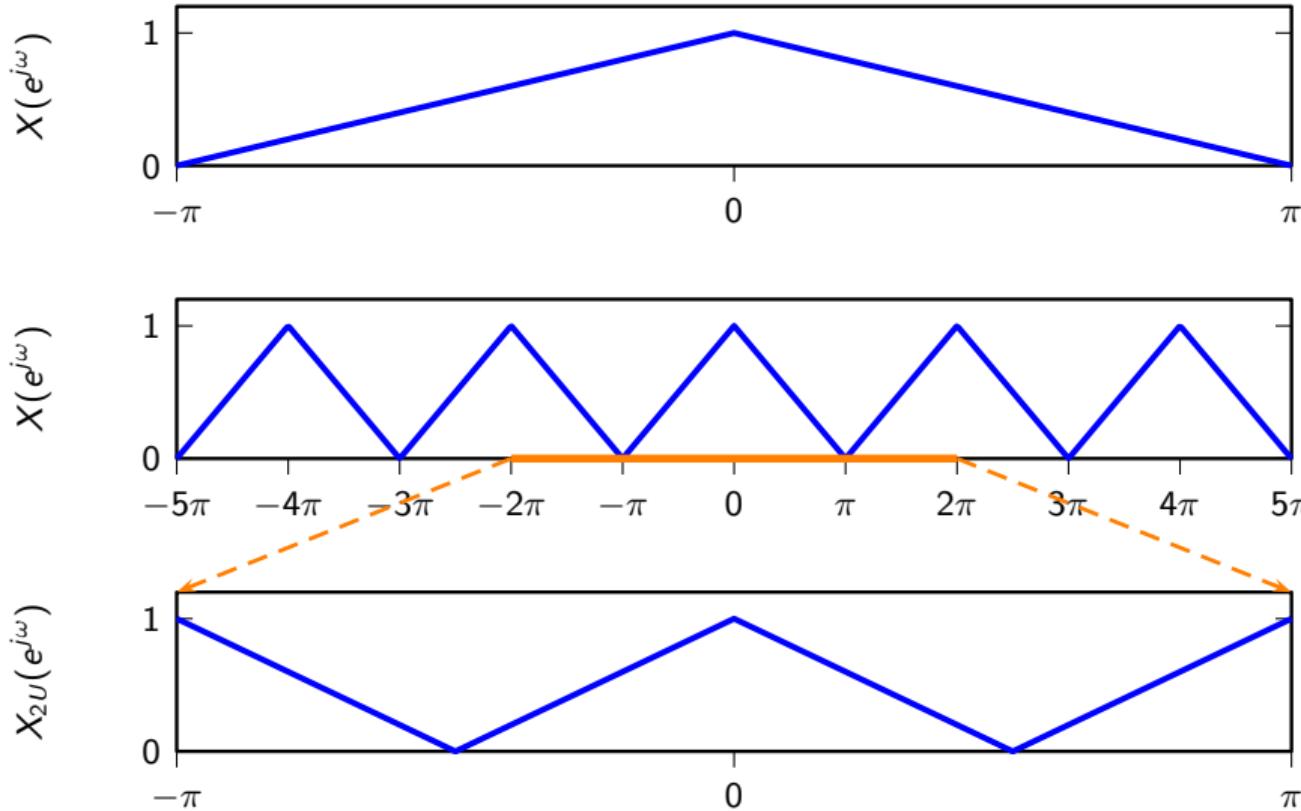
Upsampling by 2



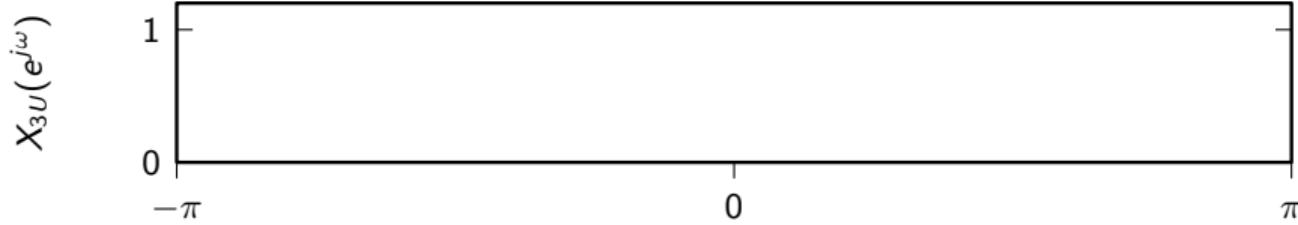
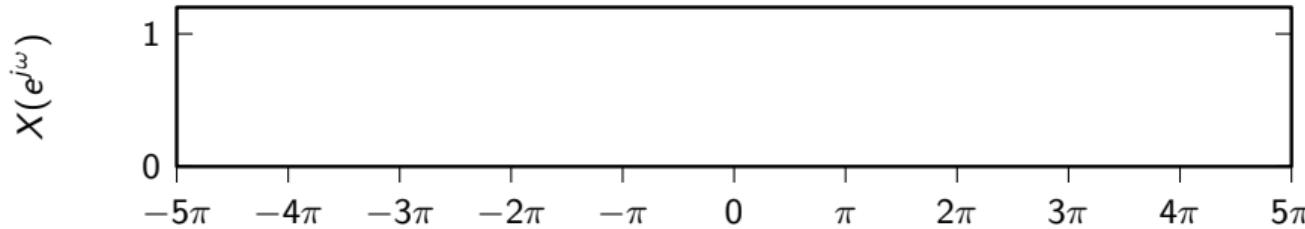
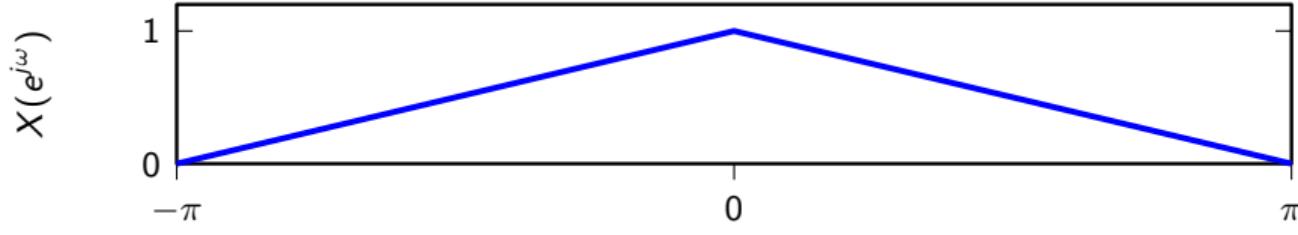
Upsampling by 2



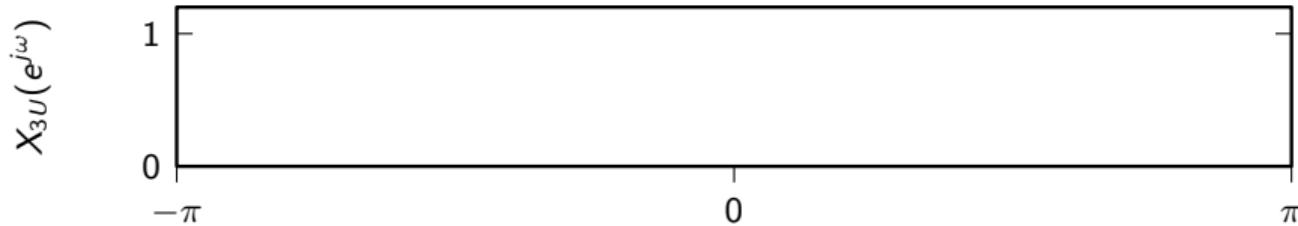
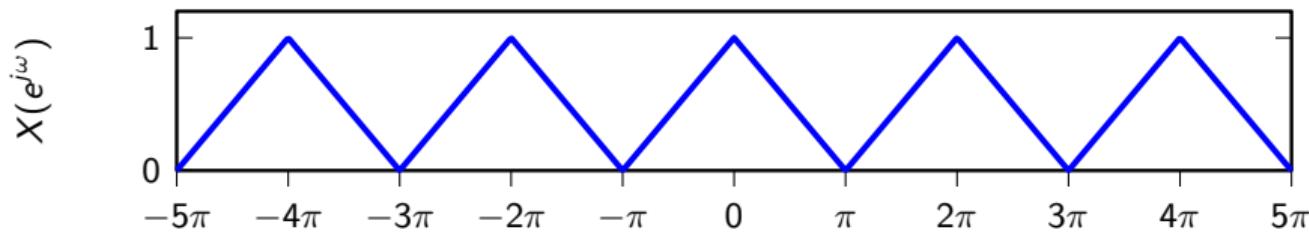
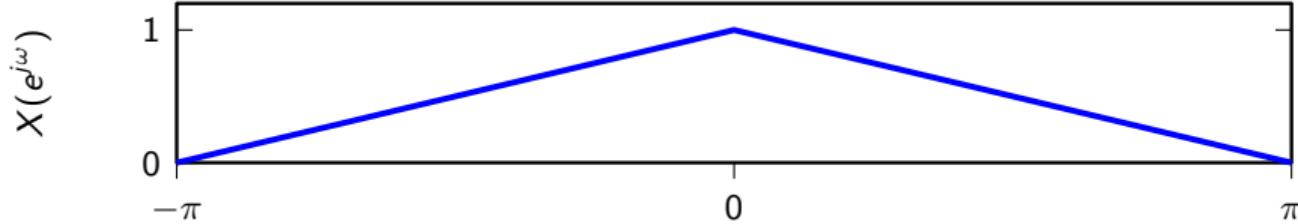
Upsampling by 2



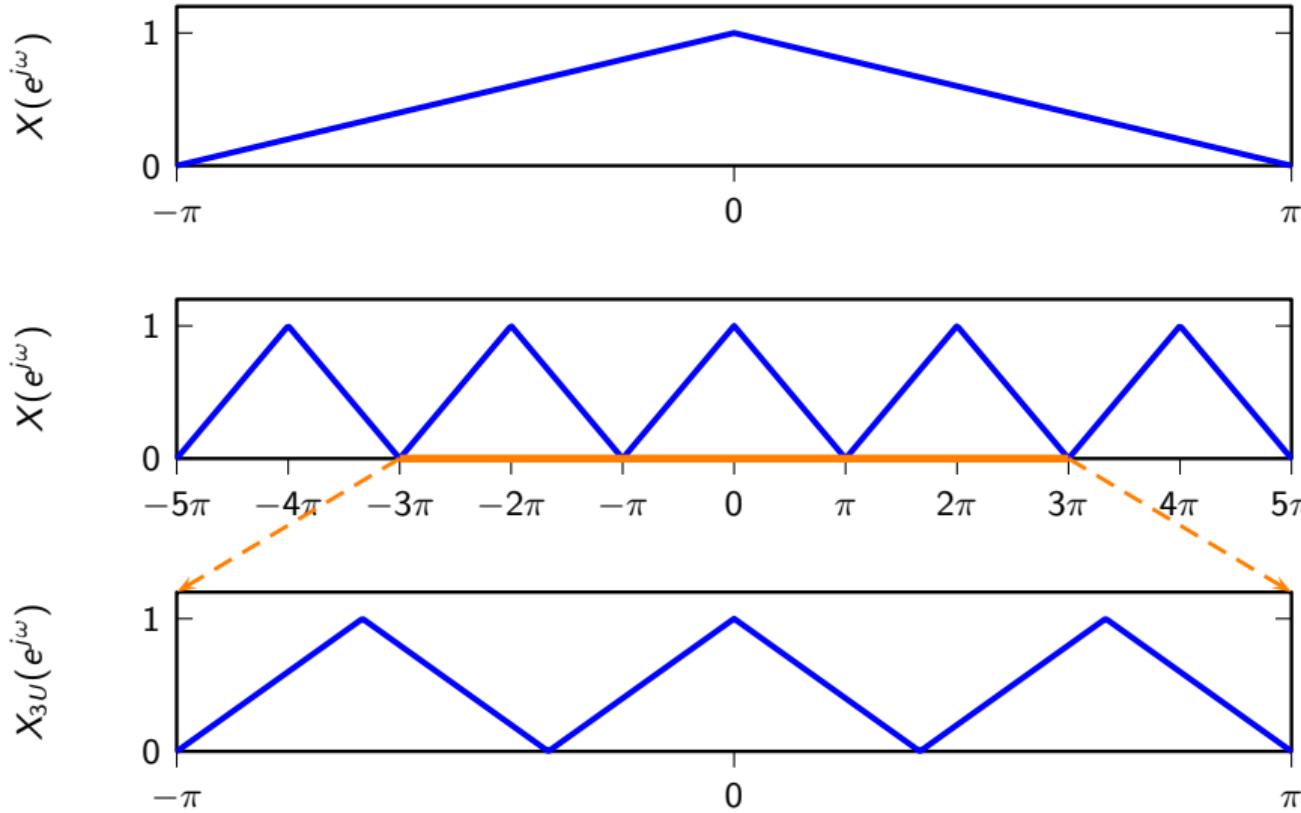
Upsampling by 3



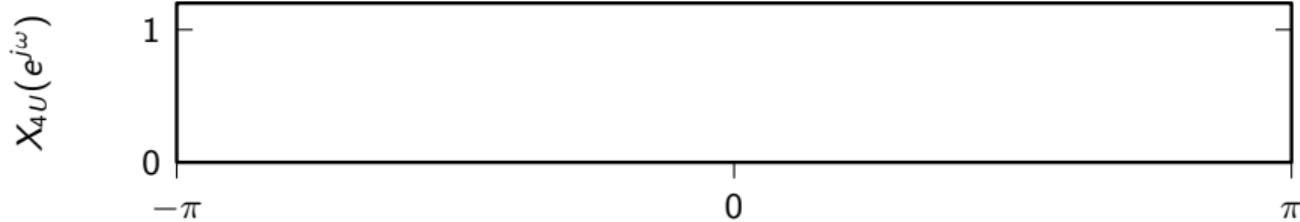
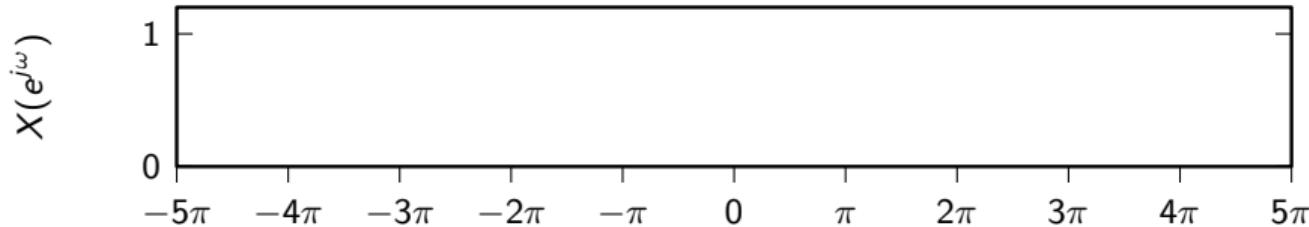
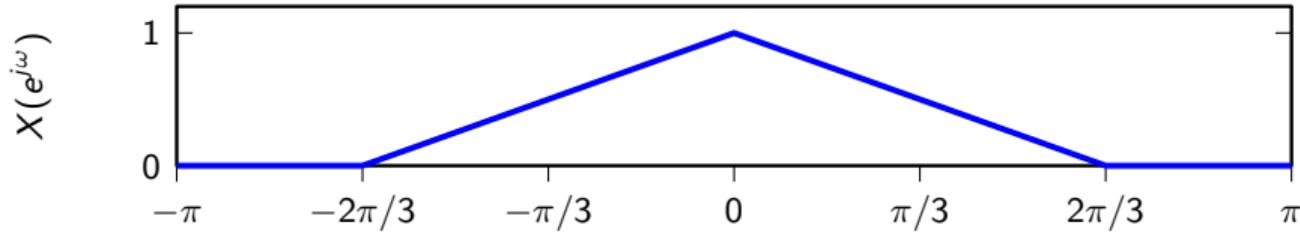
Upsampling by 3



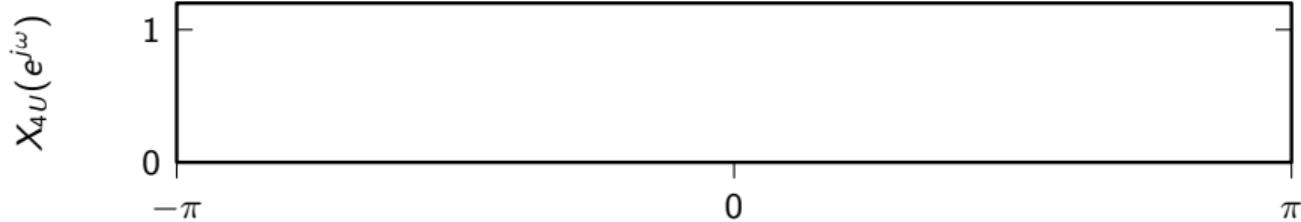
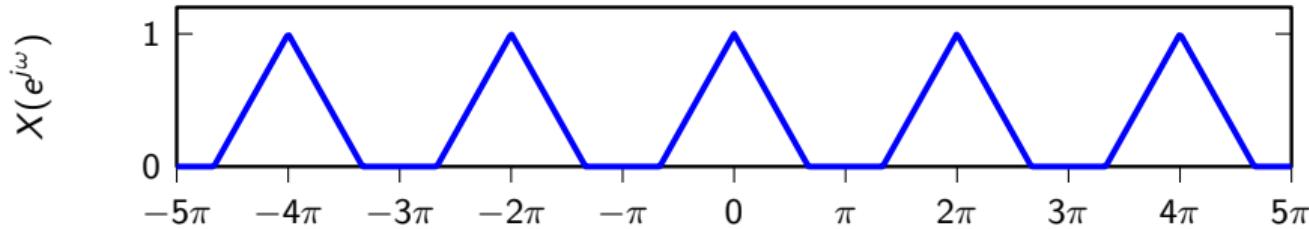
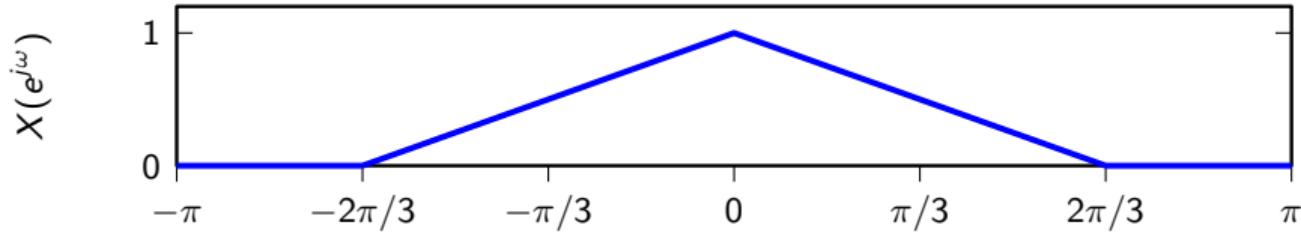
Upsampling by 3



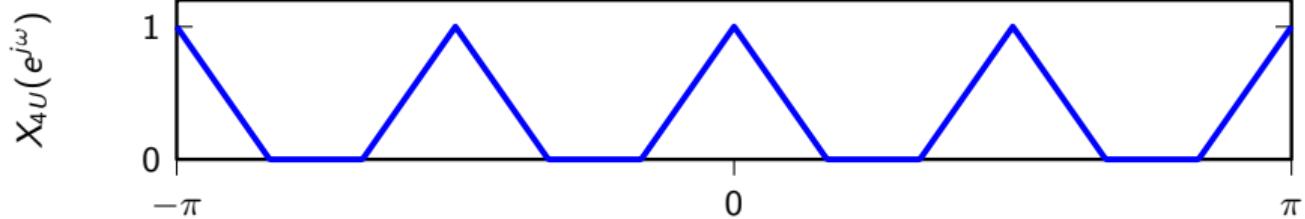
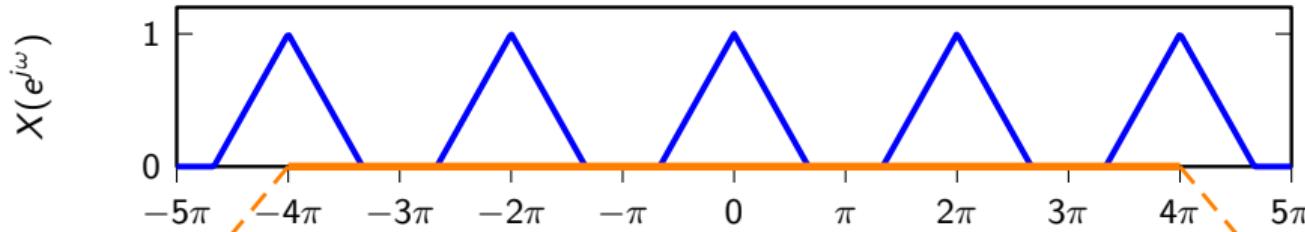
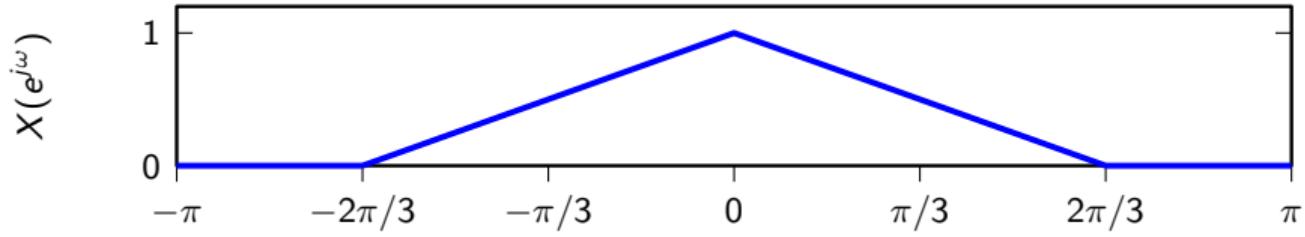
Upsampling by 4



Upsampling by 4



Upsampling by 4



Upsampling: what we don't like

- ▶ in the time domain: zeros between nonzero samples are not “natural”
- ▶ in the frequency domain: extra replicas of the spectrum; can we get rid of them?

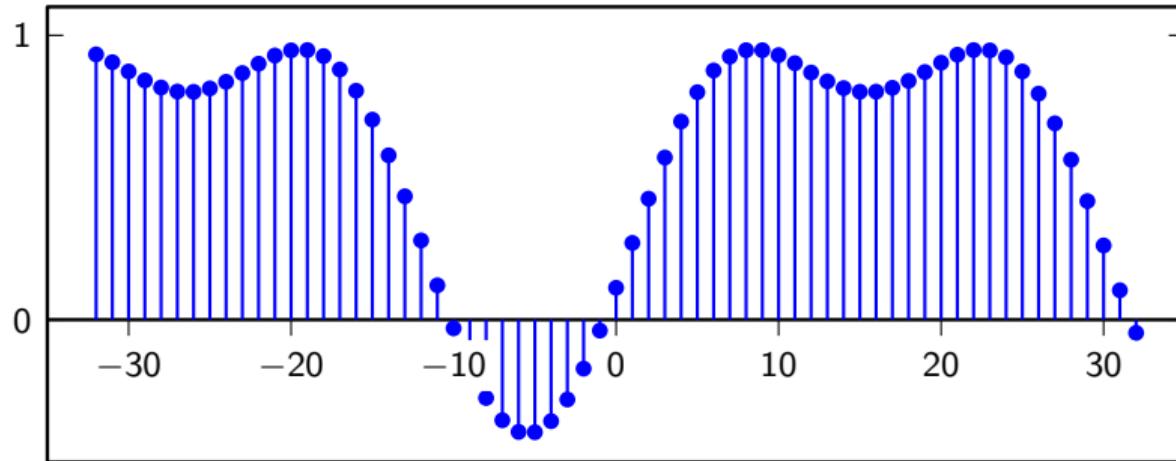
the two problems are the same!

Upsampling: what we don't like

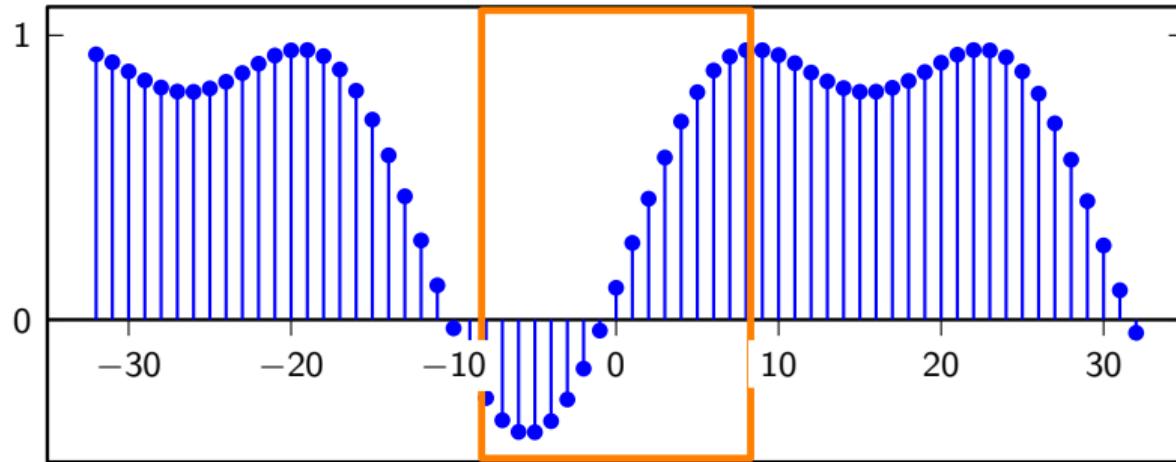
- ▶ in the time domain: zeros between nonzero samples are not “natural”
- ▶ in the frequency domain: extra replicas of the spectrum; can we get rid of them?

the two problems are the same!

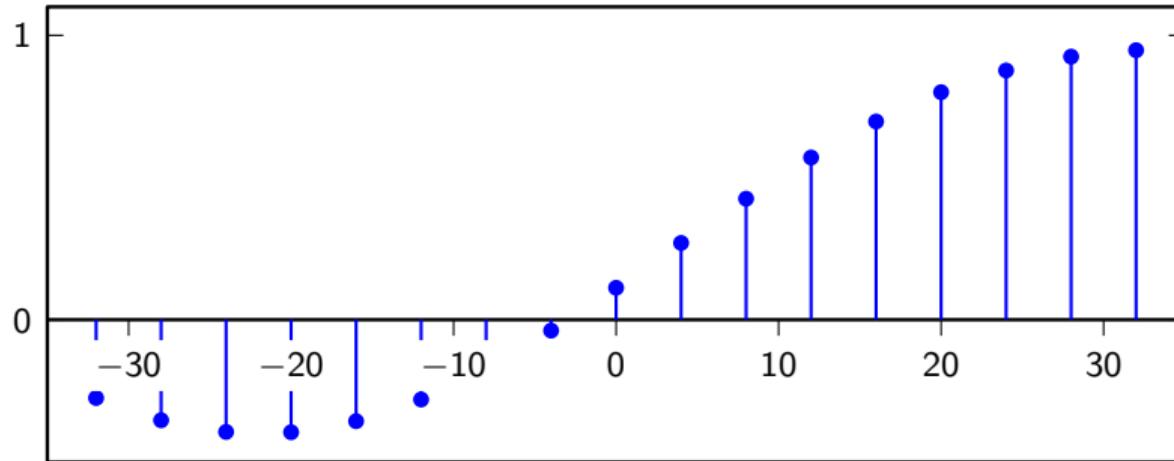
Upsampling in the time domain, revisited



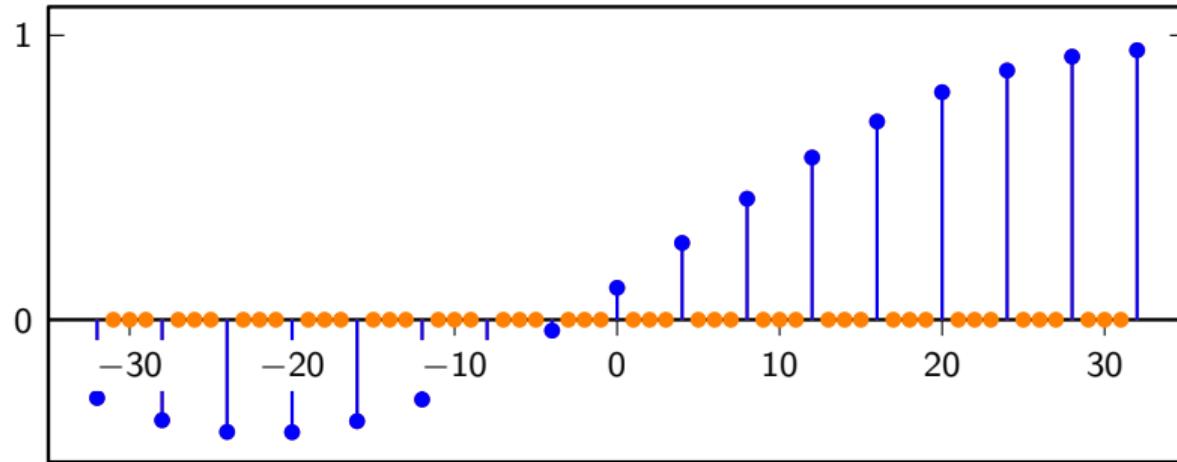
Upsampling in the time domain, revisited



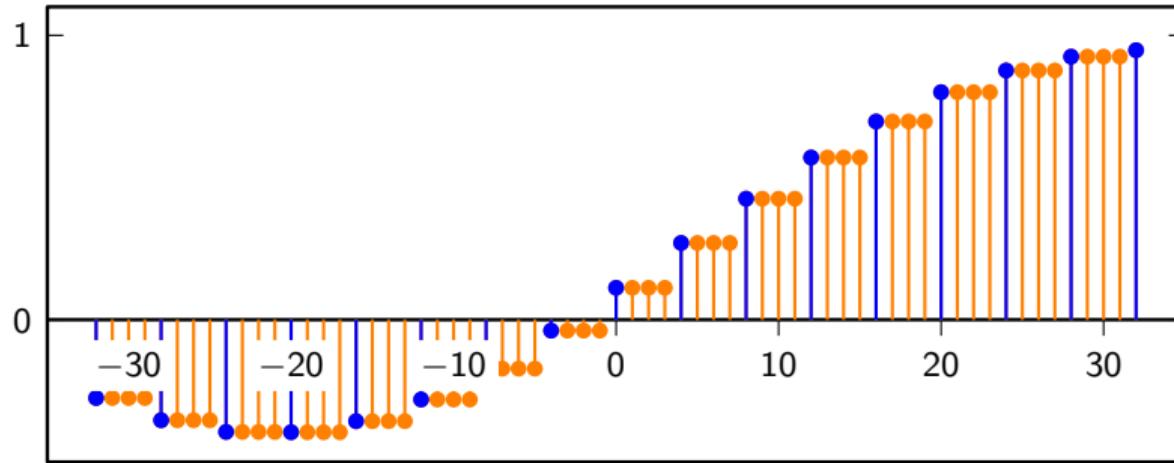
Upsampling in the time domain, revisited



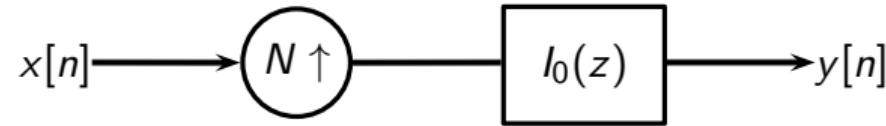
Upsampling in the time domain, revisited



Upsampling in the time domain, revisited

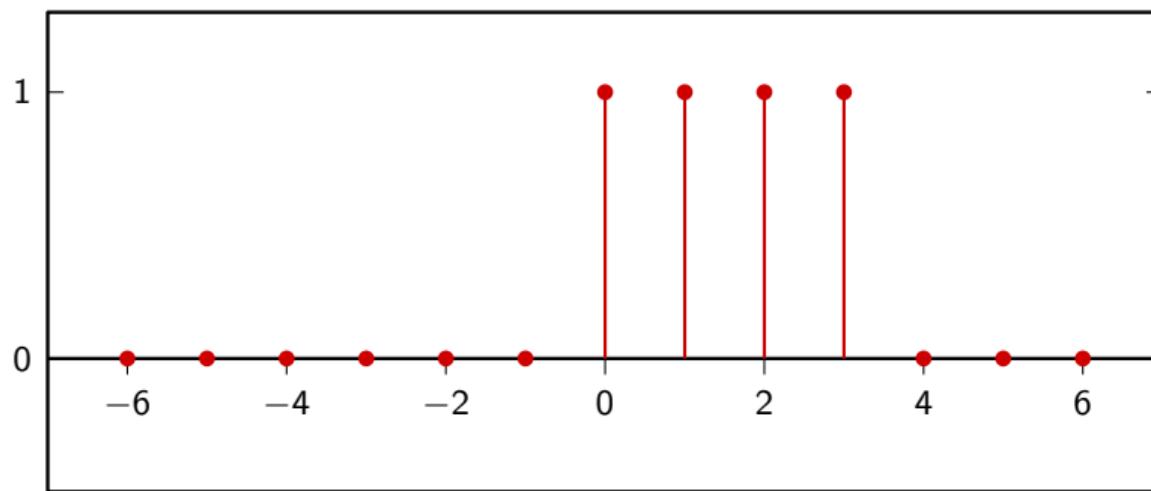


Zero-order interpolator



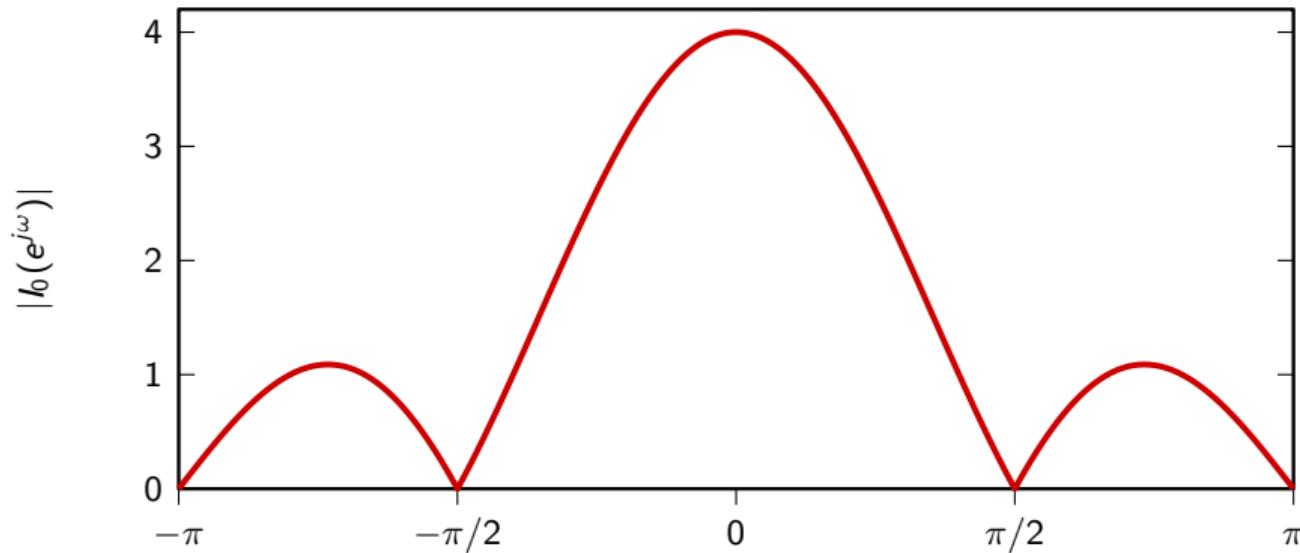
Zero-order interpolator for 4-upsampling

$$i_0[n] = u[n] - u[n - 4]$$

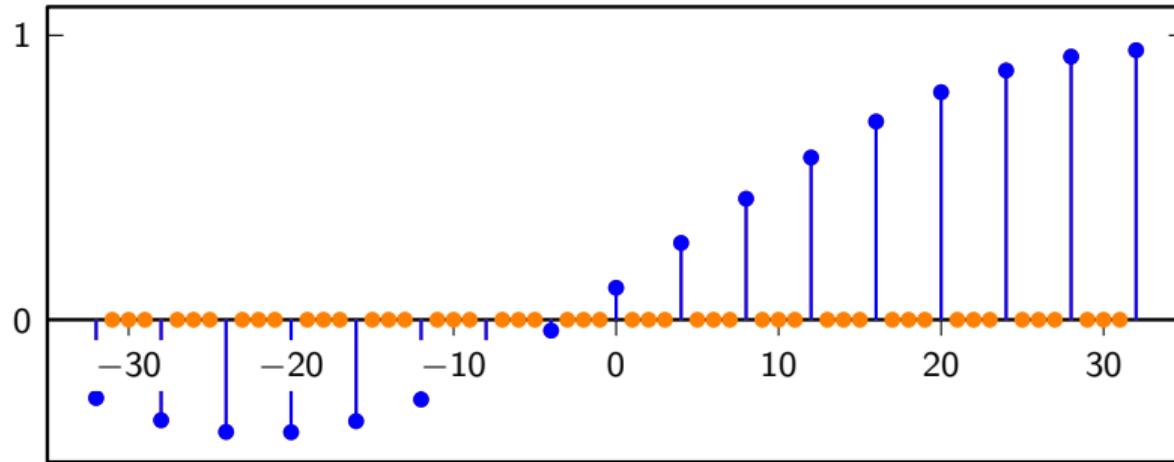


Zero-order interpolator for 4-upsampling

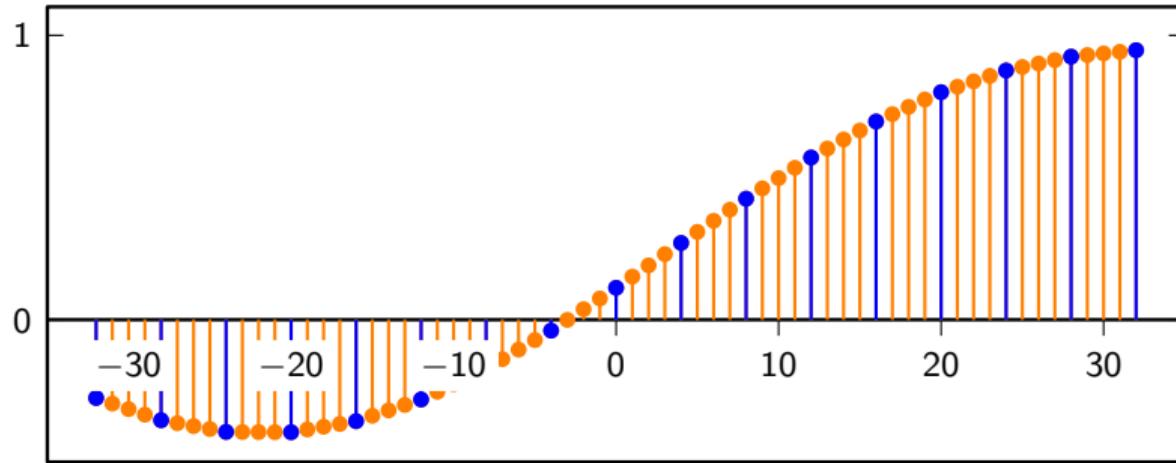
$$|I_0(e^{j\omega})| = \left| \frac{\sin\left(\frac{\omega}{2}N\right)}{\sin\left(\frac{\omega}{2}\right)} \right| \quad N = 4$$



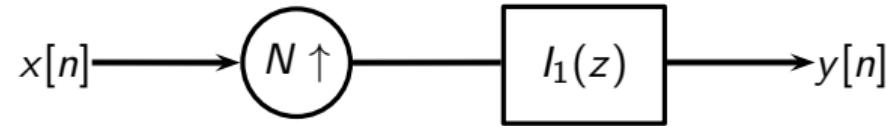
Upsampling in the time domain, revisited



Upsampling in the time domain, revisited

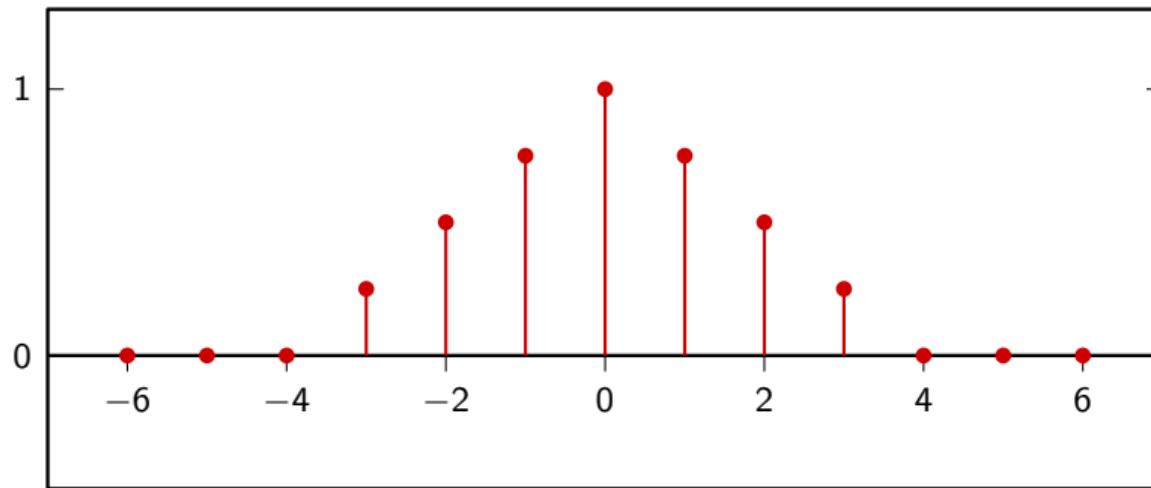


first-order interpolator



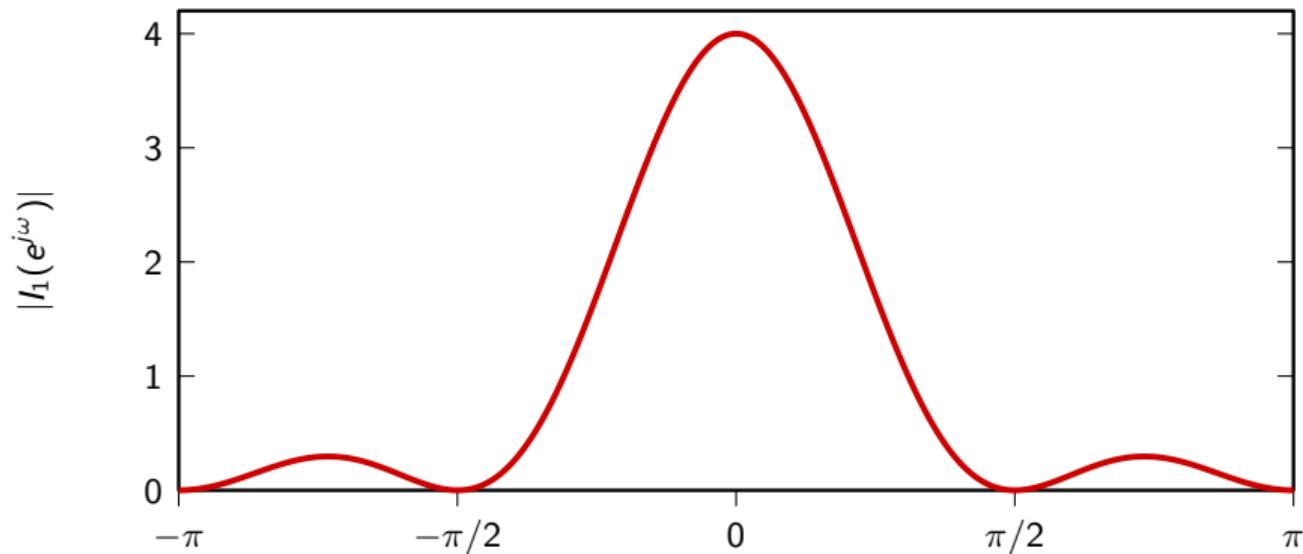
first-order interpolator for 4-upsampling

$$i_1[n] = (i_0[n] * i_0[n])/N$$

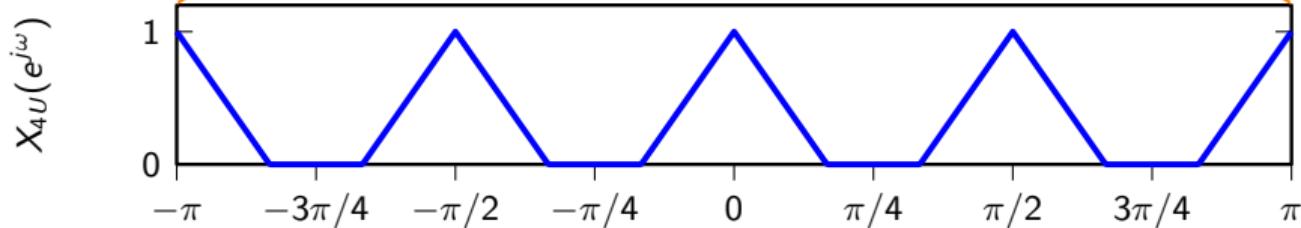
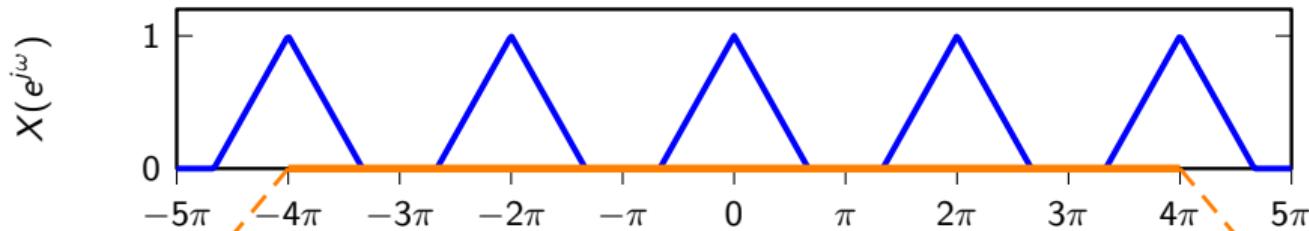
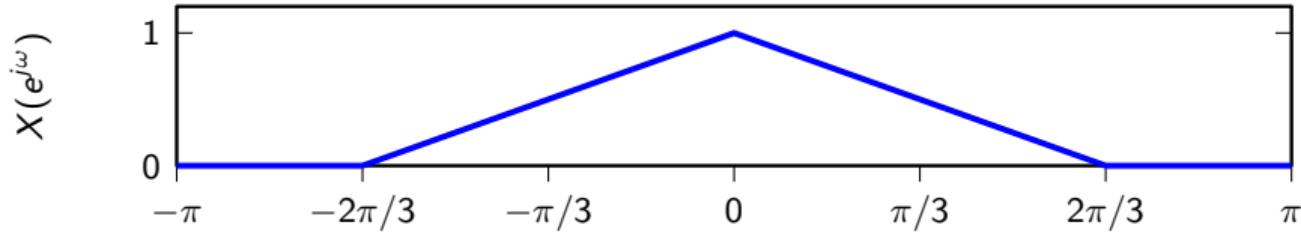


first-order interpolator for 4-upampling

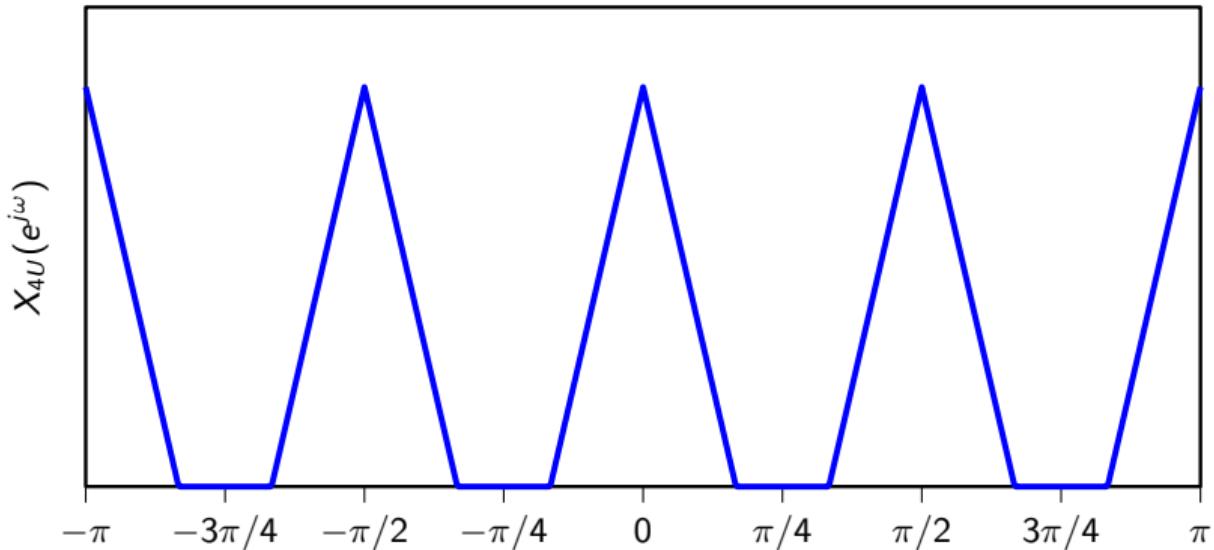
$$|I_1(e^{j\omega})| = |I_0(e^{j\omega})|^2 / N \quad N = 4$$



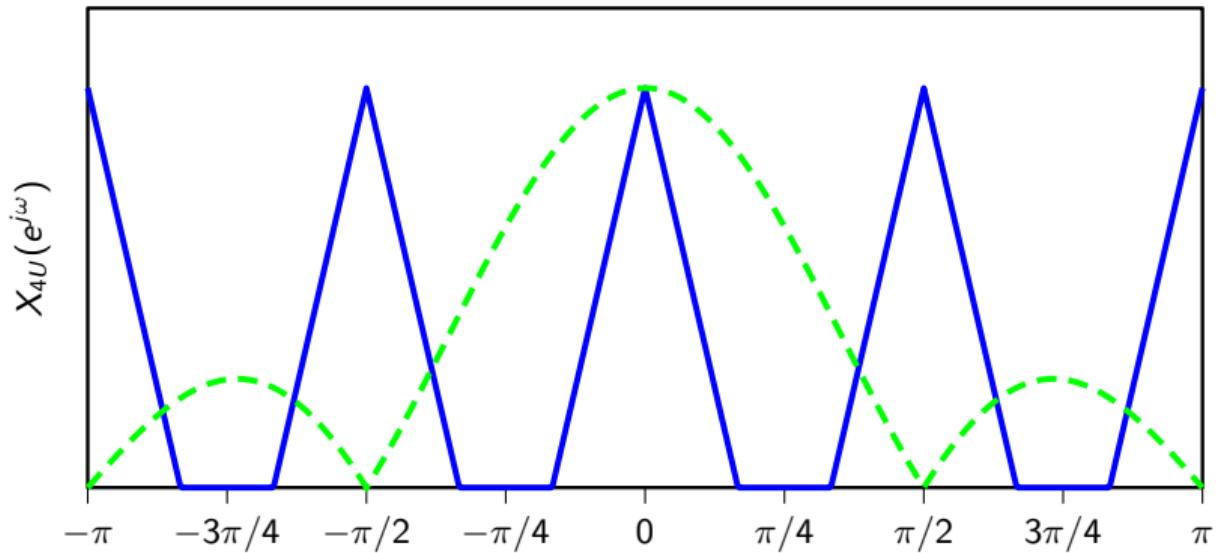
in the frequency domain...



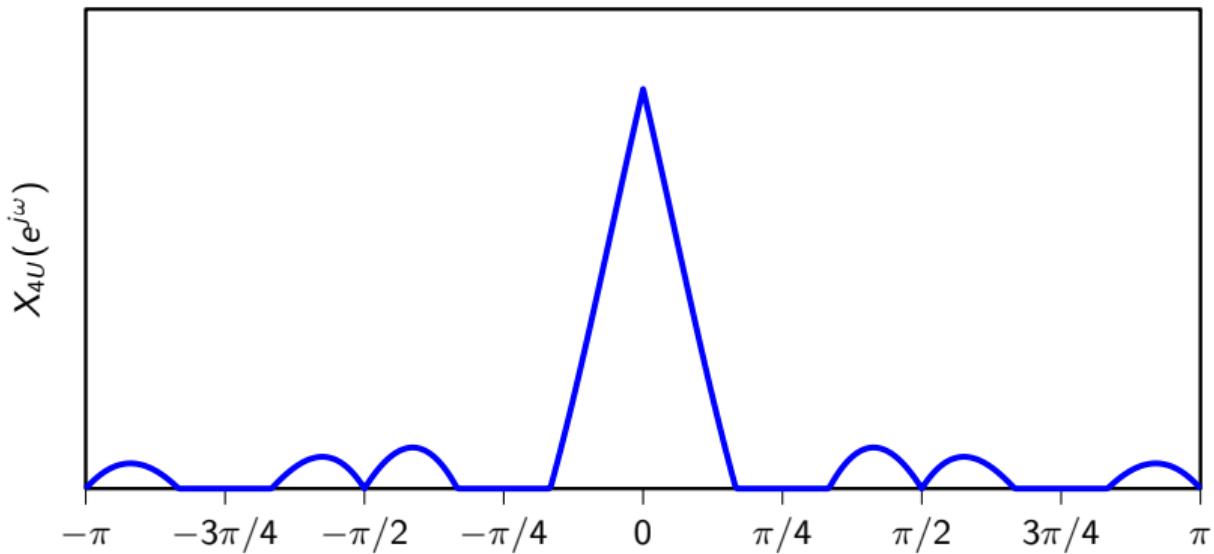
in the frequency domain...



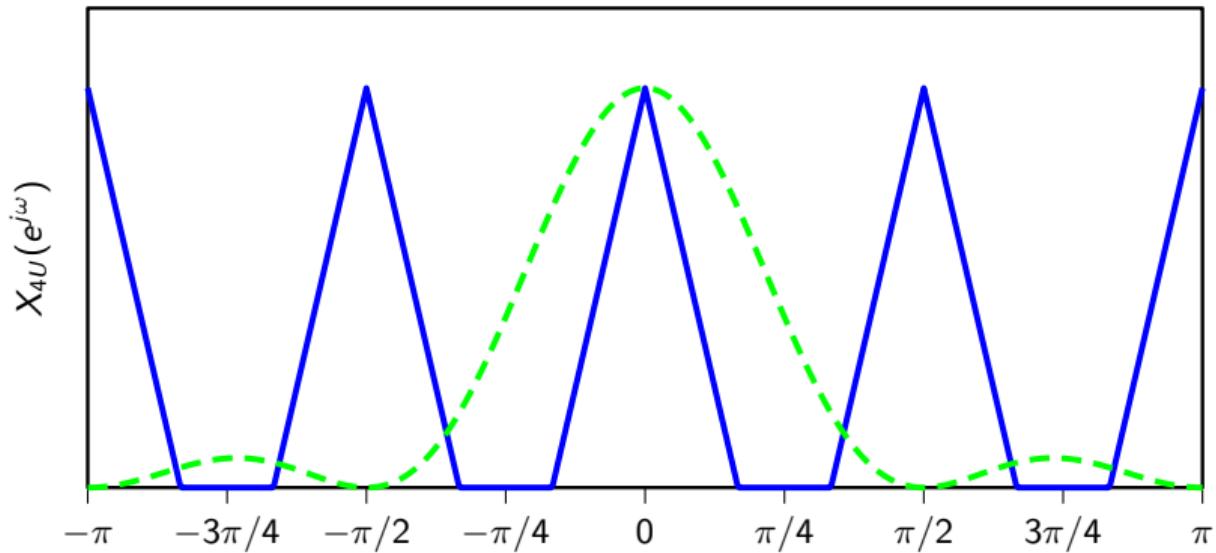
in the frequency domain...



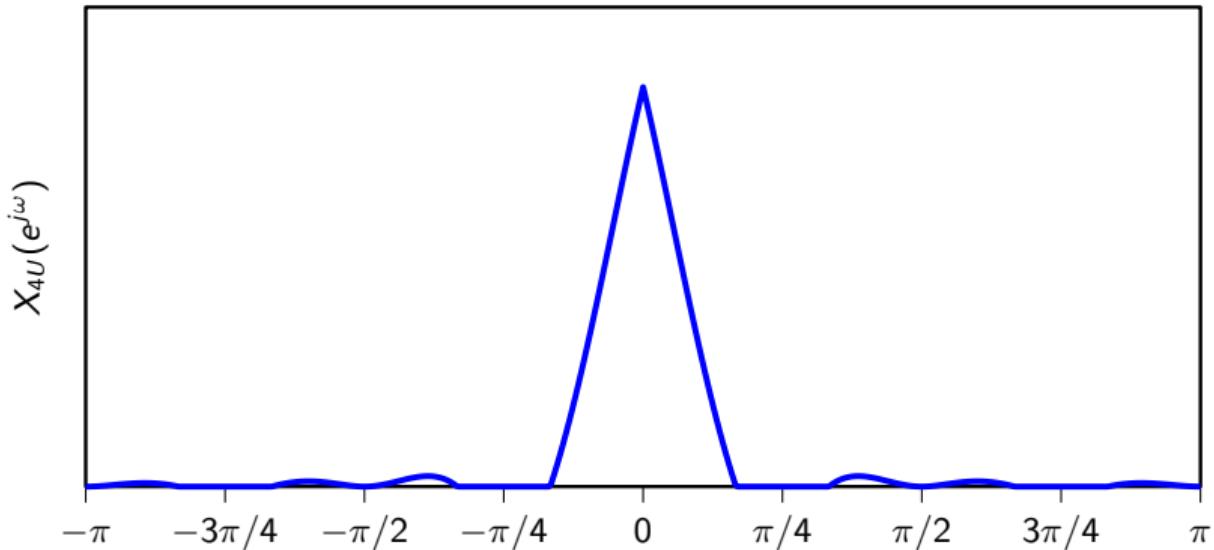
in the frequency domain...



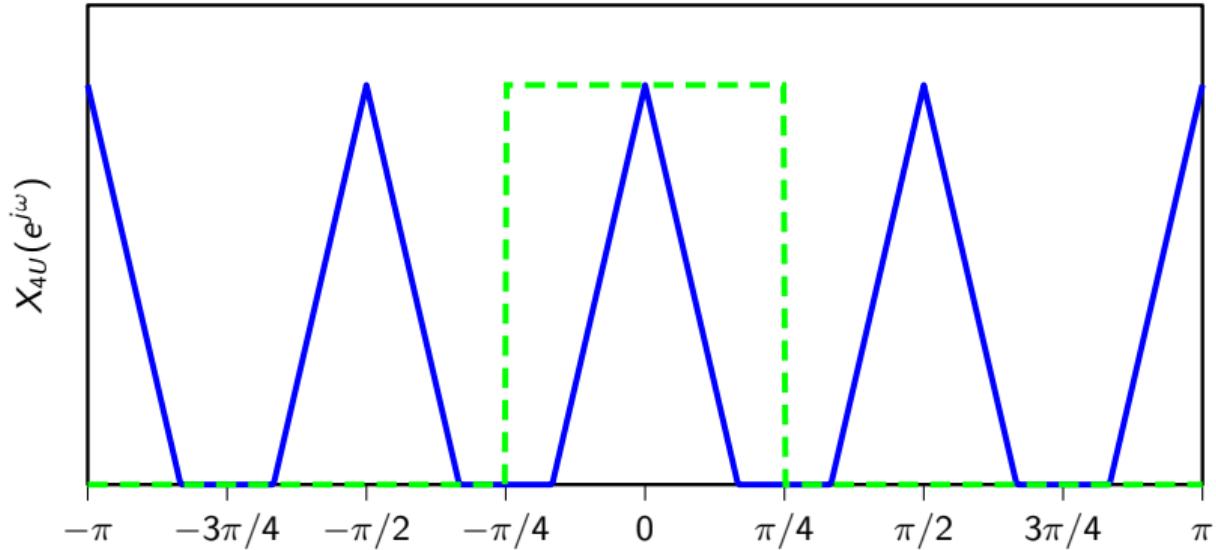
in the frequency domain...



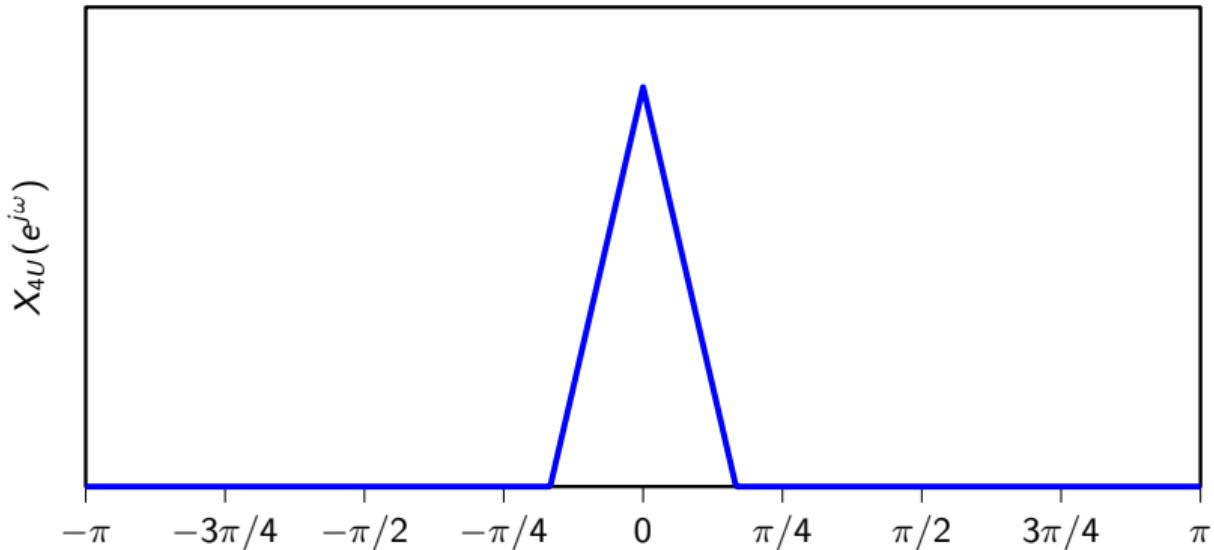
in the frequency domain...



in the frequency domain...



in the frequency domain...



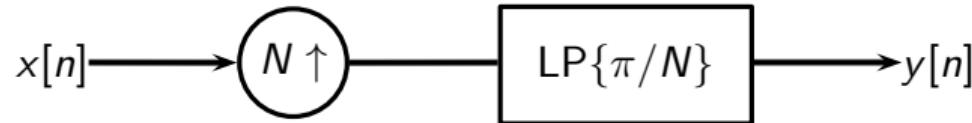
ideal digital interpolator



$$H(e^{j\omega}) = \text{rect}(\omega N/2\pi)$$

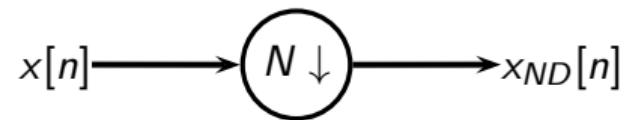
$$h[n] = (1/N) \text{sinc}(n/N)$$

ideal digital interpolator

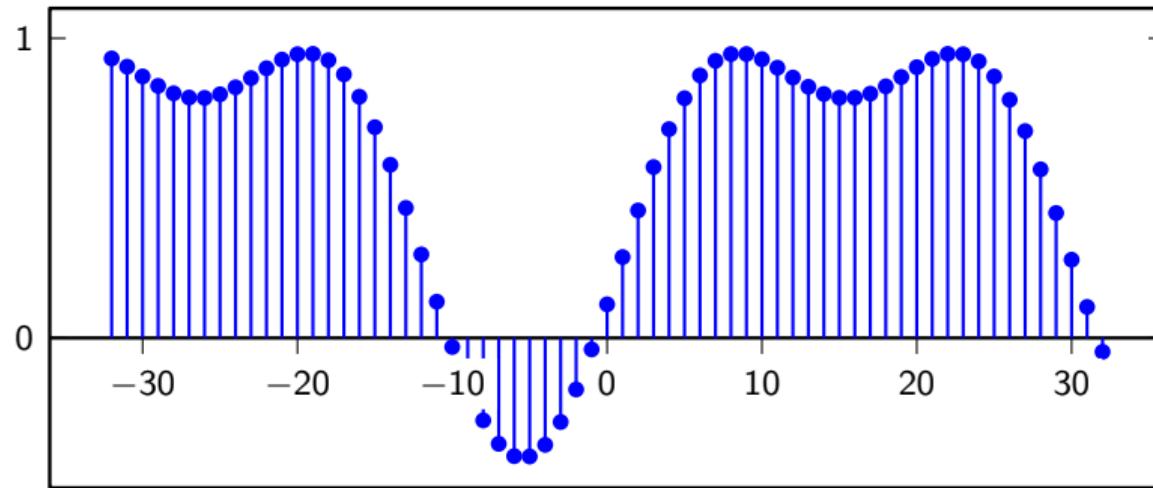


Downsampling

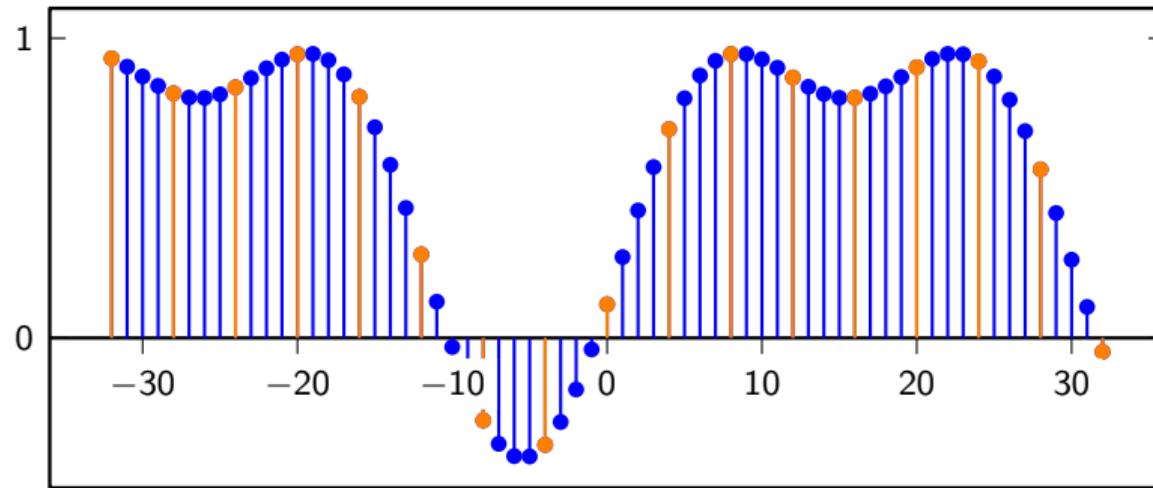
$$x_{ND}[n] = x[nN]$$



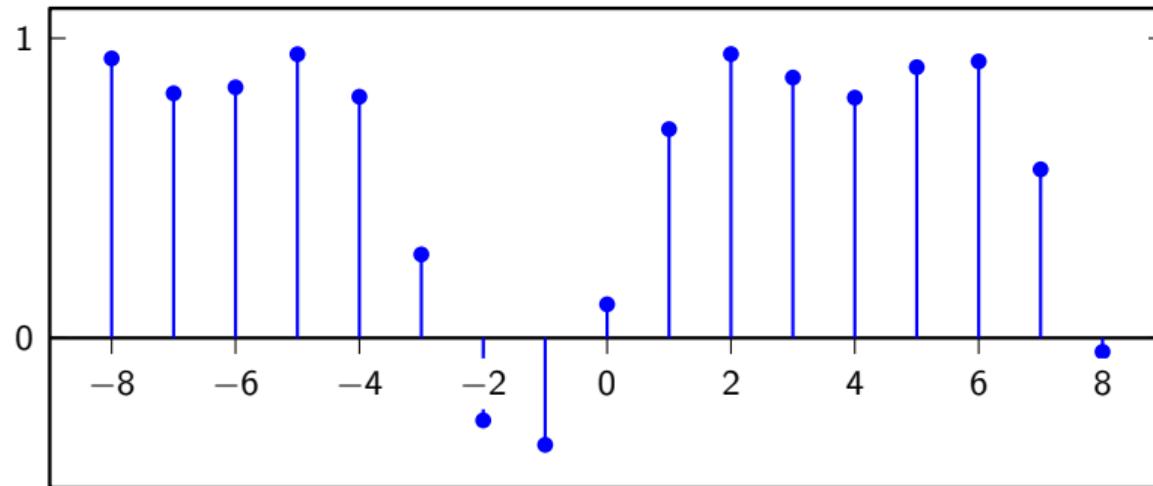
Example: downsampling by 4



Example: downsampling by 4



Example: downsampling by 4



Spectral representation

$$X_{ND}(z) = \sum_{k=-\infty}^{\infty} x[kN]z^{-k} = ?$$

if we can compute

$$A(z) = \sum_{k=-\infty}^{\infty} x[kN]z^{-kN}$$

then

$$X_{ND}(z) = A(z^{1/N})$$

Spectral representation

$$X_{ND}(z) = \sum_{k=-\infty}^{\infty} x[kN]z^{-k} = ?$$

if we can compute

$$A(z) = \sum_{k=-\infty}^{\infty} x[kN]z^{-kN}$$

then

$$X_{ND}(z) = A(z^{1/N})$$

Spectral representation

$$\begin{aligned} A(z) &= \sum_{k=-\infty}^{\infty} x[kN]z^{-kN} \\ &= \sum_{k=-\infty}^{\infty} \xi[k]x[k]z^{-k} \end{aligned}$$

$$\xi[n] = \begin{cases} 1 & \text{for } n = kN \\ 0 & \text{otherwise} \end{cases}$$

Spectral representation

$$\begin{aligned} A(z) &= \sum_{k=-\infty}^{\infty} x[kN]z^{-kN} \\ &= \sum_{k=-\infty}^{\infty} \xi[k]x[k]z^{-k} \end{aligned}$$

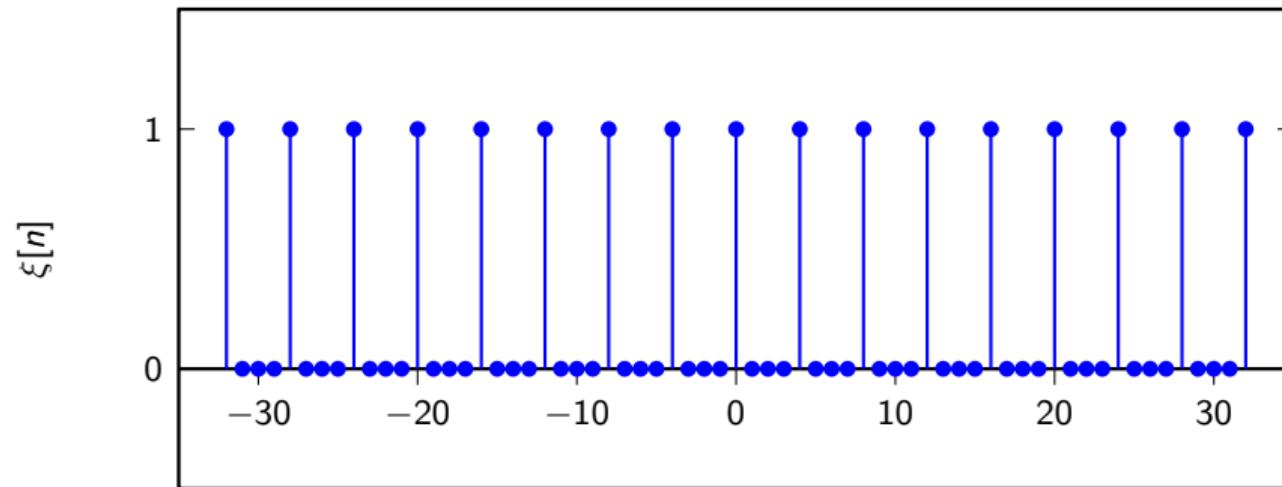
$$\xi[n] = \begin{cases} 1 & \text{for } n = kN \\ 0 & \text{otherwise} \end{cases}$$

Spectral representation

$$\begin{aligned} A(z) &= \sum_{k=-\infty}^{\infty} x[kN]z^{-kN} \\ &= \sum_{k=-\infty}^{\infty} \xi[k]x[k]z^{-k} \end{aligned}$$

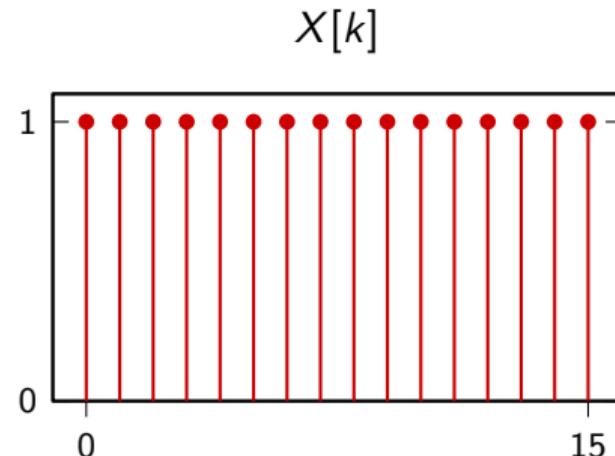
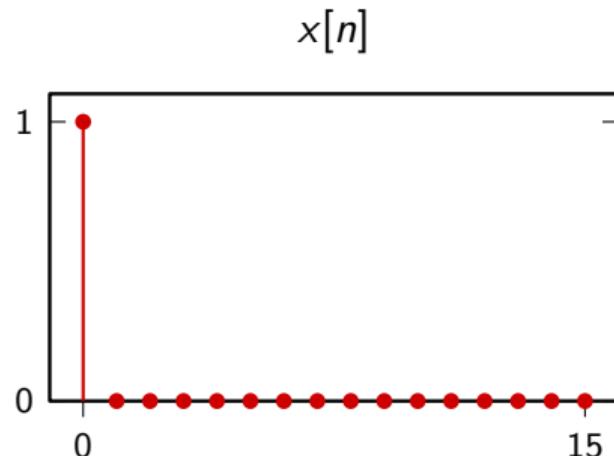
$$\xi[n] = \begin{cases} 1 & \text{for } n = kN \\ 0 & \text{otherwise} \end{cases}$$

$\xi[n]$ for $N = 4$



Blast from the past: DFT of $x[n] = \delta[n]$, $x[n] \in \mathbb{C}^N$

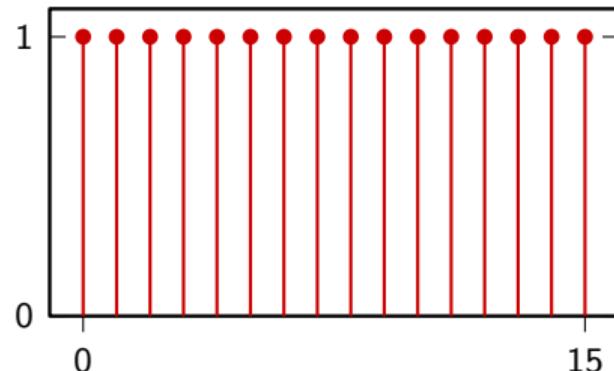
$$X[k] = \sum_{n=0}^{N-1} \delta[n] e^{-j\frac{2\pi}{N}nk} = 1$$



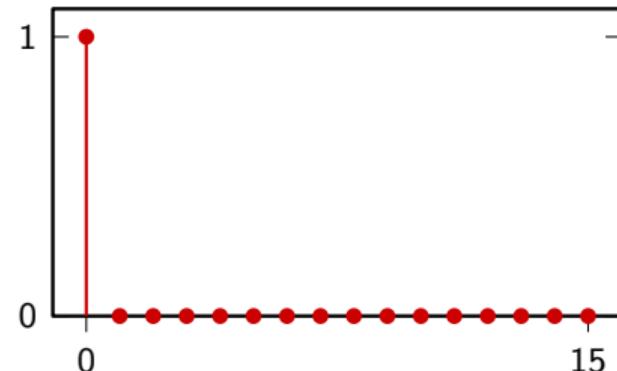
From the other side:

$$\text{IDFT } \{1\} = \frac{1}{N} \sum_{m=0}^{N-1} e^{j \frac{2\pi}{N} mn} = \delta[n]$$

$X[k]$



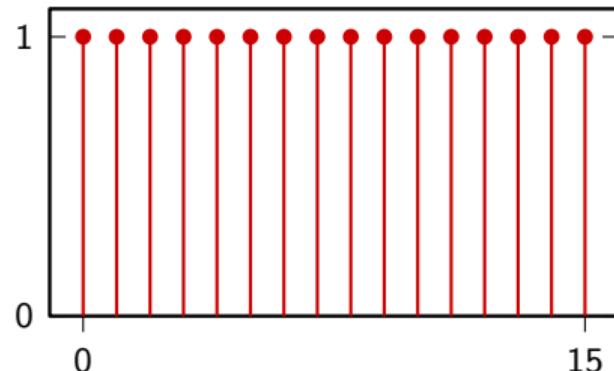
$x[n]$



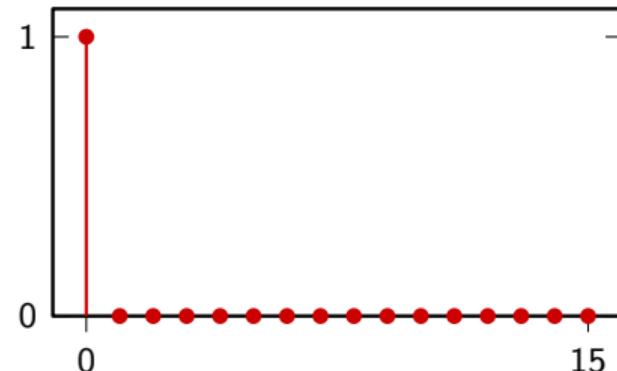
From the other side:

$$\text{IDFT } \{1\} = \frac{1}{N} \sum_{m=0}^{N-1} e^{j \frac{2\pi}{N} mn} = \begin{cases} 1 & \text{for } n = 0 \\ 0 & \text{otherwise} \end{cases} \quad n = 0, \dots, N-1$$

$X[k]$

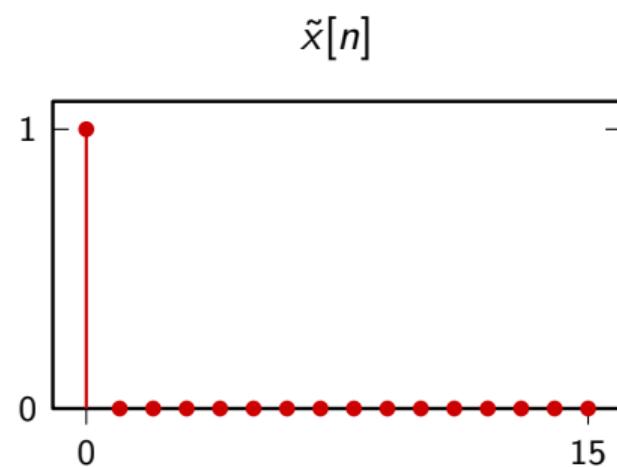
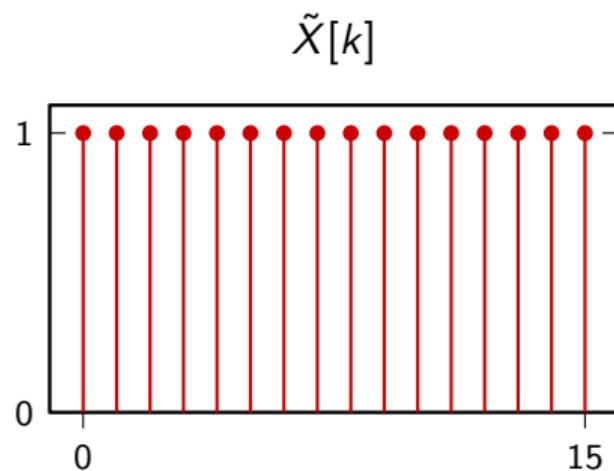


$x[n]$



From the other side:

$$\frac{1}{N} \sum_{m=0}^{N-1} e^{j\frac{2\pi}{N} mn} = \begin{cases} 1 & \text{for } n \bmod N = 0 \\ 0 & \text{otherwise} \end{cases}$$



Spectral representation

$$\xi[n] = \frac{1}{N} \sum_{m=0}^{N-1} e^{j\frac{2\pi}{N} mn}$$

$$A(z) = \sum_{k=-\infty}^{\infty} \xi[k] x[k] z^{-k}$$

$$= \frac{1}{N} \sum_{m=0}^{N-1} \sum_{k=-\infty}^{\infty} x[k] e^{j\frac{2\pi}{N} mk} z^{-k}$$

$$= \frac{1}{N} \sum_{m=0}^{N-1} X(e^{-j\frac{2\pi}{N} m} z)$$

Spectral representation

$$\xi[n] = \frac{1}{N} \sum_{m=0}^{N-1} e^{j\frac{2\pi}{N} mn}$$

$$A(z) = \sum_{k=-\infty}^{\infty} \xi[k] x[k] z^{-k}$$

$$= \frac{1}{N} \sum_{m=0}^{N-1} \sum_{k=-\infty}^{\infty} x[k] e^{j\frac{2\pi}{N} mk} z^{-k}$$

$$= \frac{1}{N} \sum_{m=0}^{N-1} X(e^{-j\frac{2\pi}{N} m} z)$$

Spectral representation

$$\xi[n] = \frac{1}{N} \sum_{m=0}^{N-1} e^{j\frac{2\pi}{N} mn}$$

$$A(z) = \sum_{k=-\infty}^{\infty} \xi[k] x[k] z^{-k}$$

$$= \frac{1}{N} \sum_{m=0}^{N-1} \sum_{k=-\infty}^{\infty} x[k] e^{j\frac{2\pi}{N} mk} z^{-k}$$

$$= \frac{1}{N} \sum_{m=0}^{N-1} X(e^{-j\frac{2\pi}{N} m} z)$$

Spectral representation

$$\xi[n] = \frac{1}{N} \sum_{m=0}^{N-1} e^{j\frac{2\pi}{N} mn}$$

$$A(z) = \sum_{k=-\infty}^{\infty} \xi[k] x[k] z^{-k}$$

$$= \frac{1}{N} \sum_{m=0}^{N-1} \sum_{k=-\infty}^{\infty} x[k] e^{j\frac{2\pi}{N} mk} z^{-k}$$

$$= \frac{1}{N} \sum_{m=0}^{N-1} X(e^{-j\frac{2\pi}{N} m} z)$$

Spectral representation

$$A(e^{j\omega}) = \frac{1}{N} \sum_{m=0}^{N-1} X(e^{j(\omega - \frac{2\pi}{N} m)})$$

Spectral representation

$$X_{ND}(z) = A(z^{1/N}) = \frac{1}{N} \sum_{m=0}^{N-1} X(e^{-j\frac{2\pi}{N}m} z^{\frac{1}{N}})$$

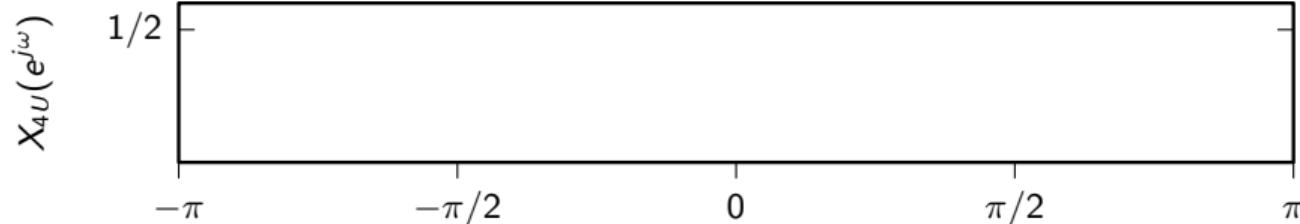
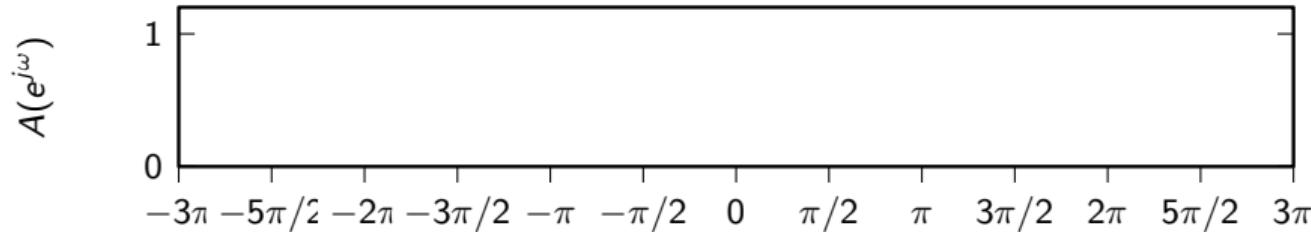
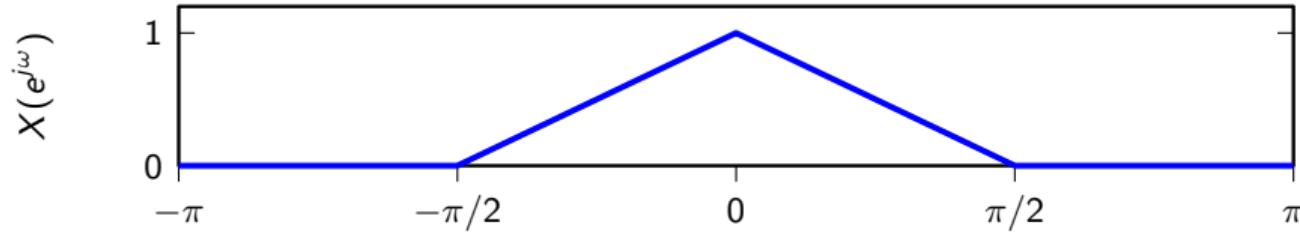
$$X_{ND}(e^{j\omega}) = \frac{1}{N} \sum_{m=0}^{N-1} X(e^{j(\frac{\omega-2\pi m}{N})})$$

Spectral representation

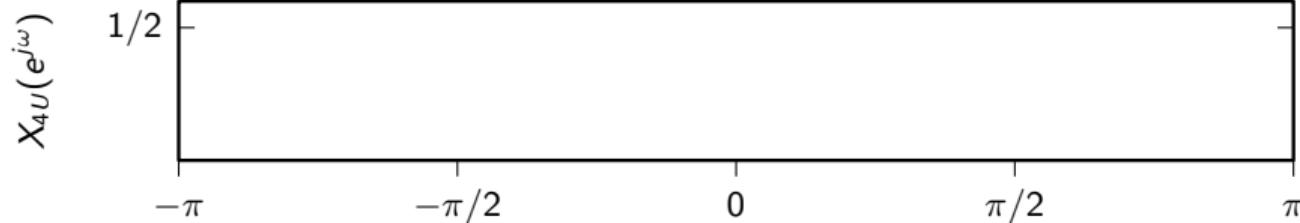
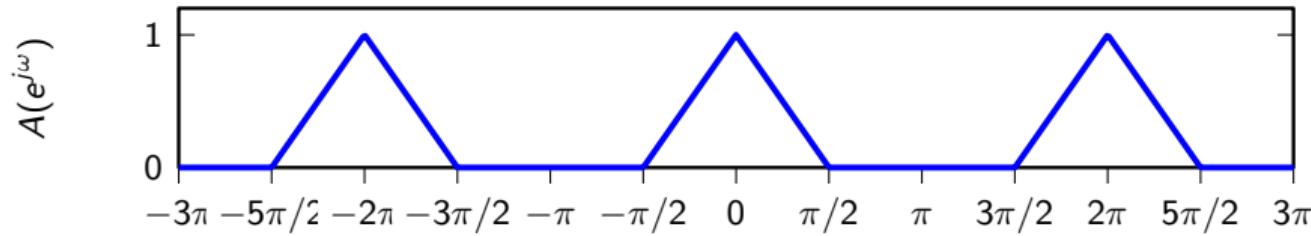
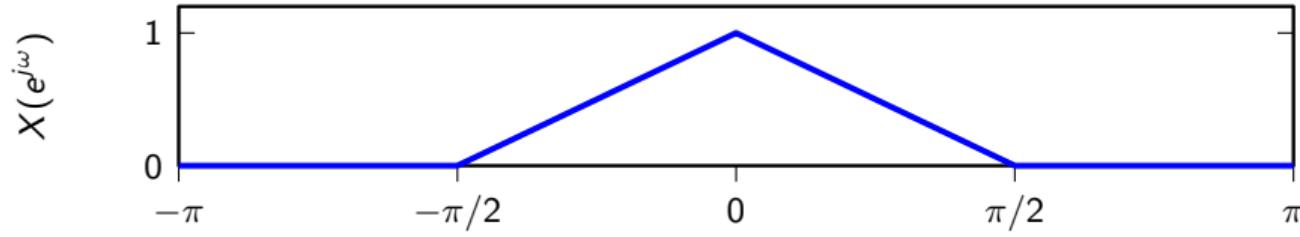
$$X_{ND}(z) = A(z^{1/N}) = \frac{1}{N} \sum_{m=0}^{N-1} X(e^{-j\frac{2\pi}{N}m} z^{\frac{1}{N}})$$

$$X_{ND}(e^{j\omega}) = \frac{1}{N} \sum_{m=0}^{N-1} X(e^{j(\frac{\omega-2\pi m}{N})})$$

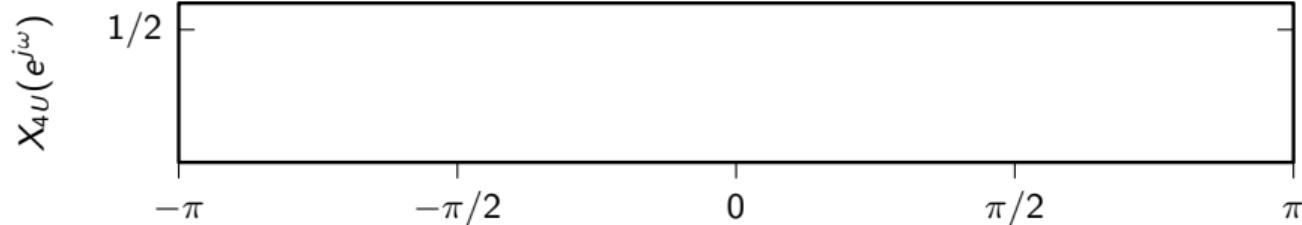
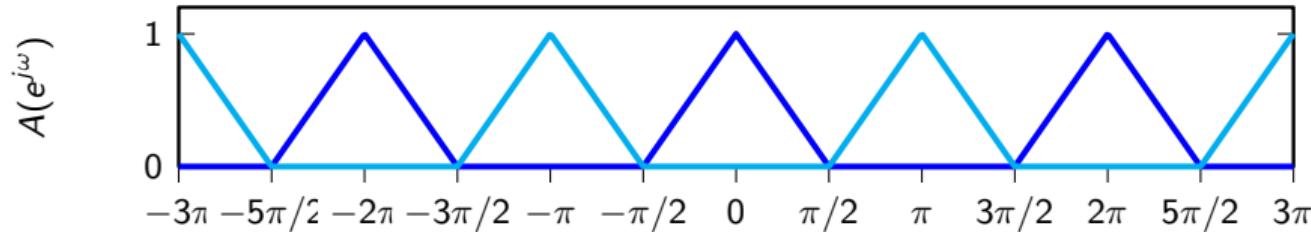
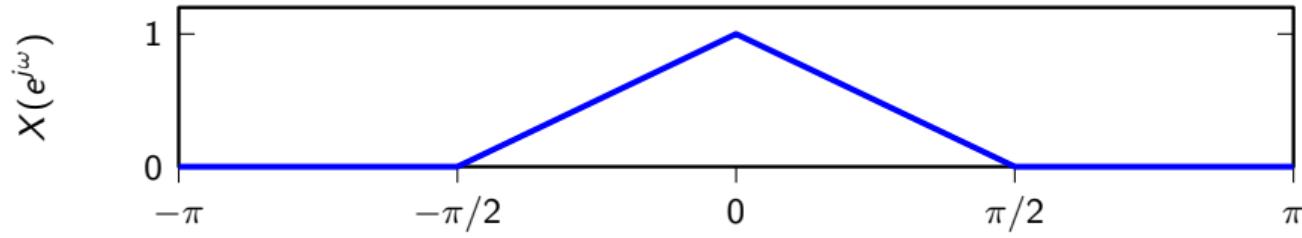
Downsampling by 2



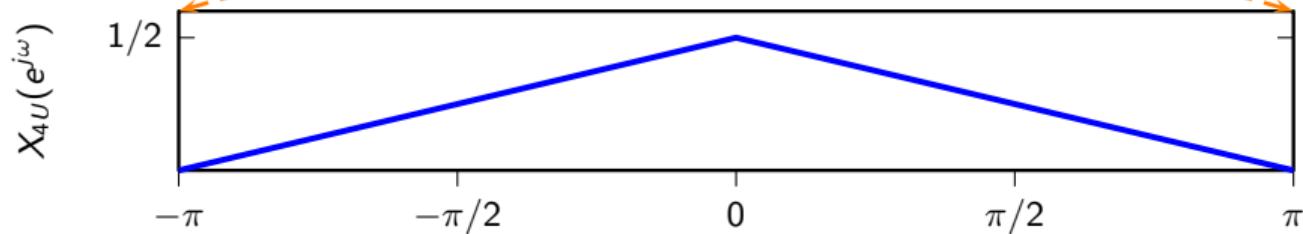
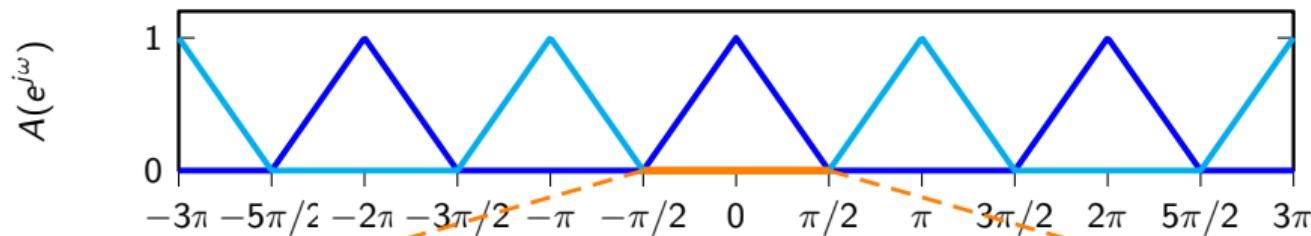
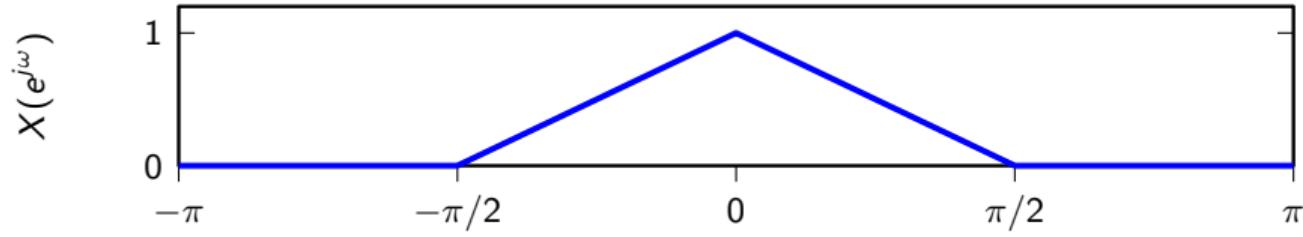
Downsampling by 2



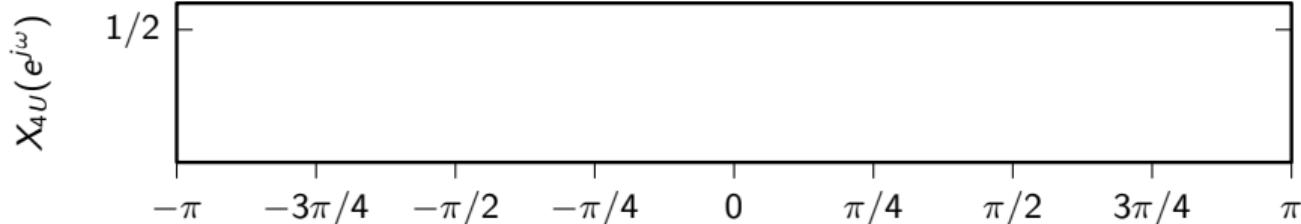
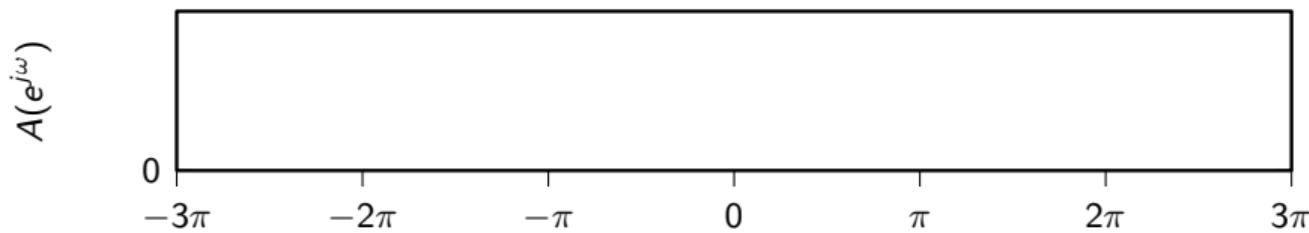
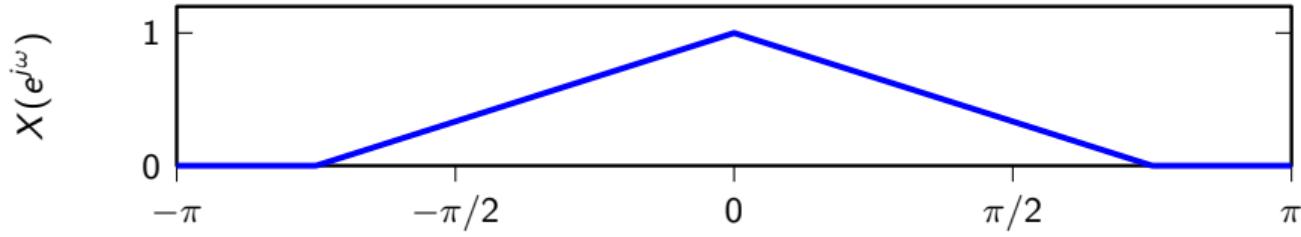
Downsampling by 2



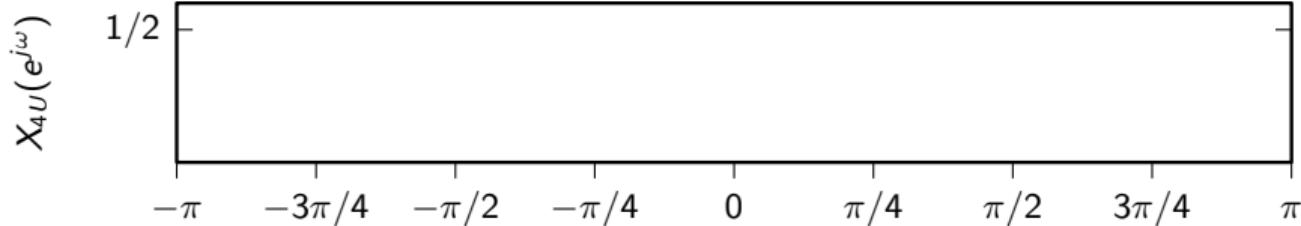
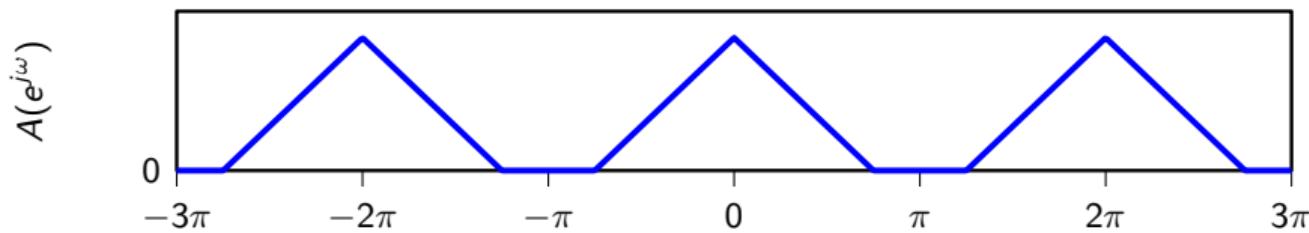
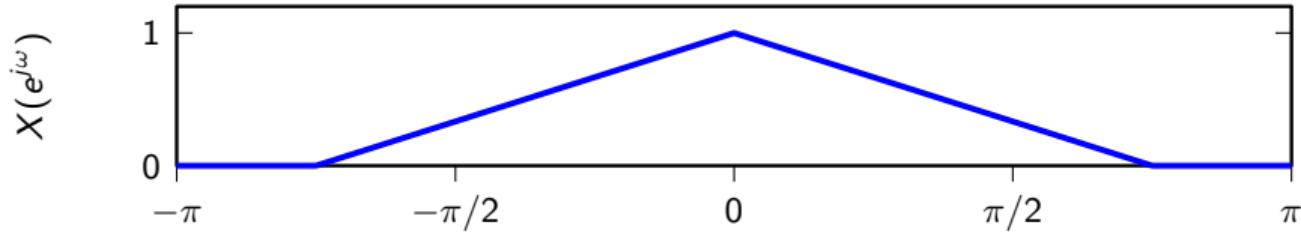
Downsampling by 2



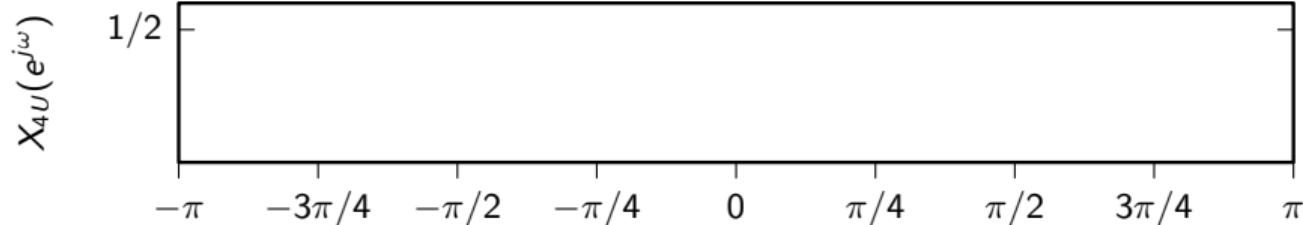
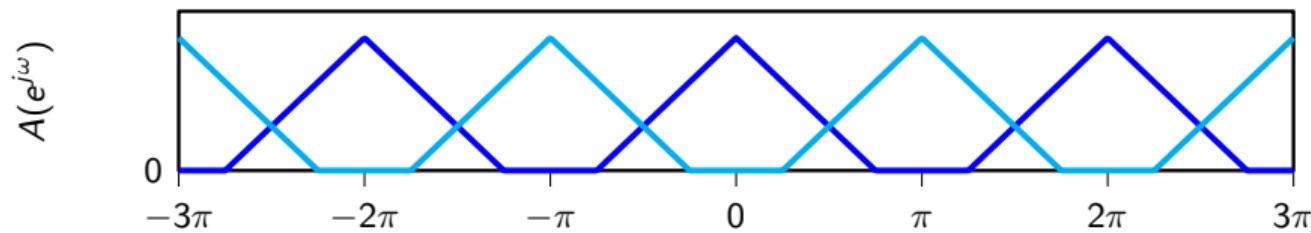
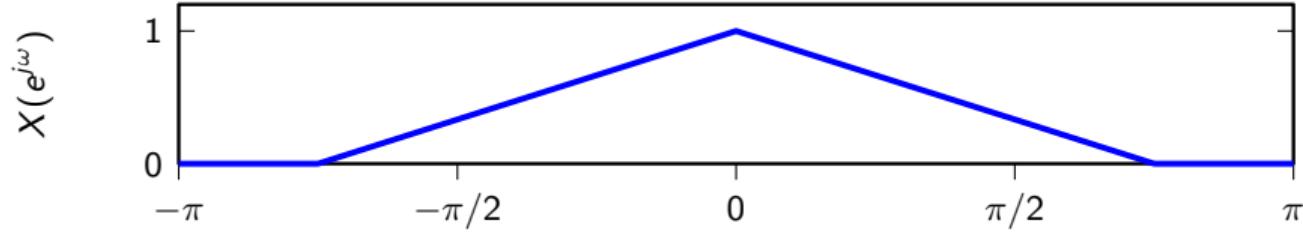
Downsampling by 2 with aliasing



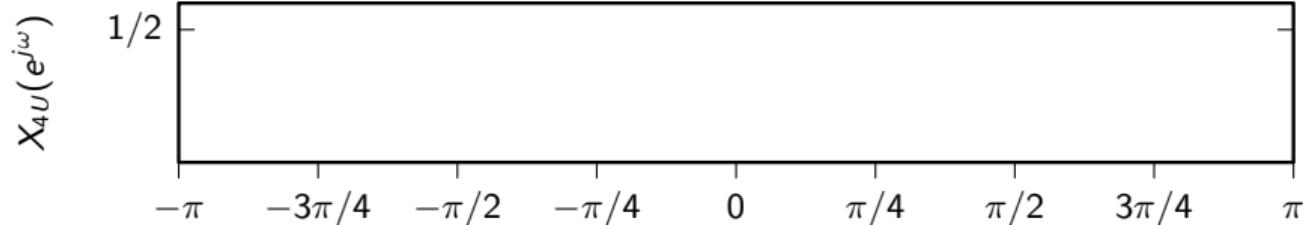
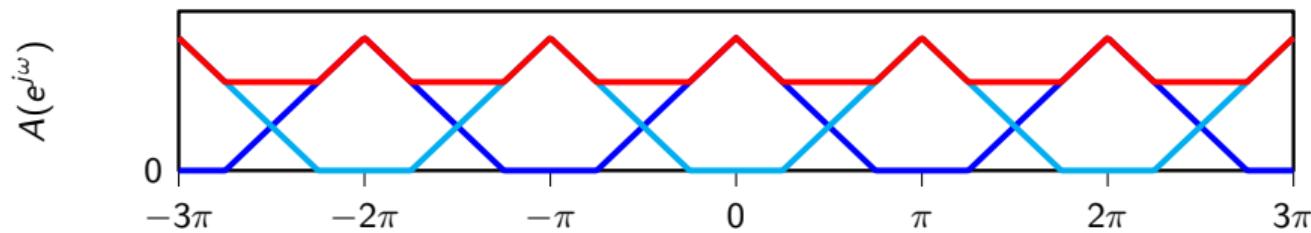
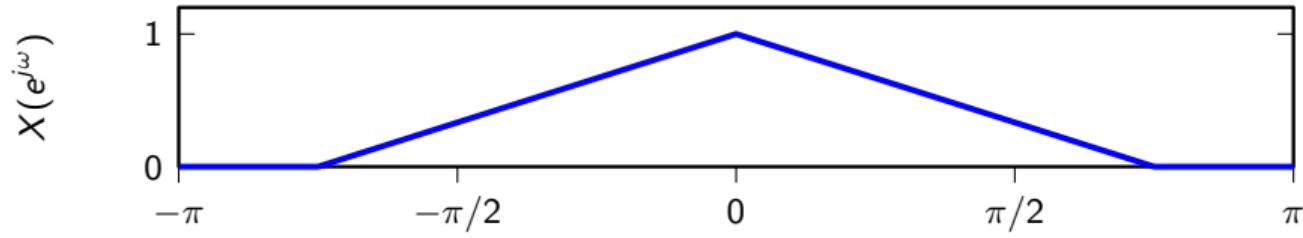
Downsampling by 2 with aliasing



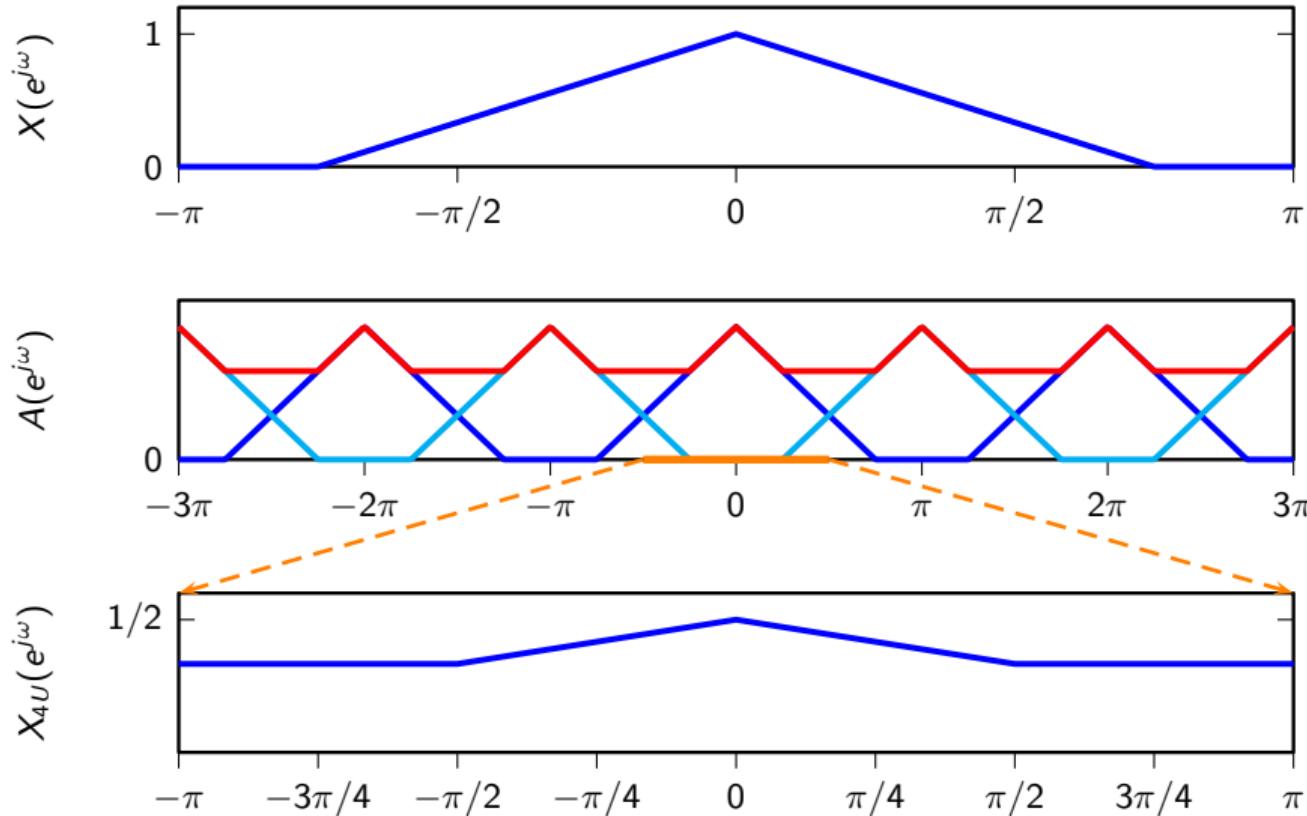
Downsampling by 2 with aliasing



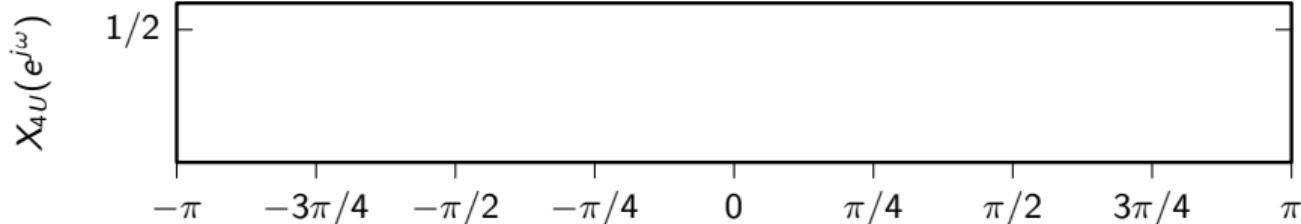
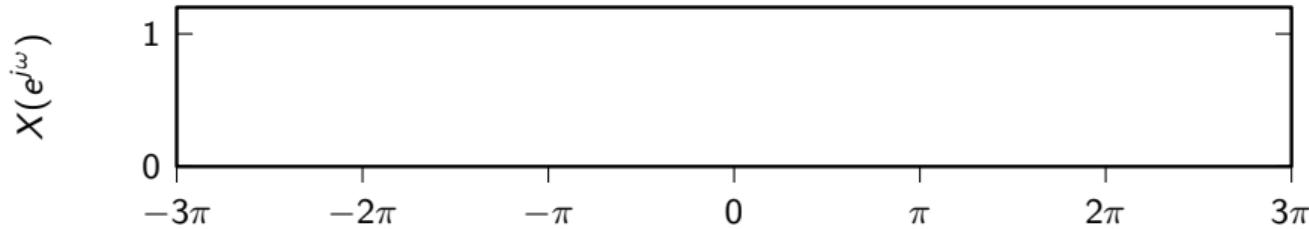
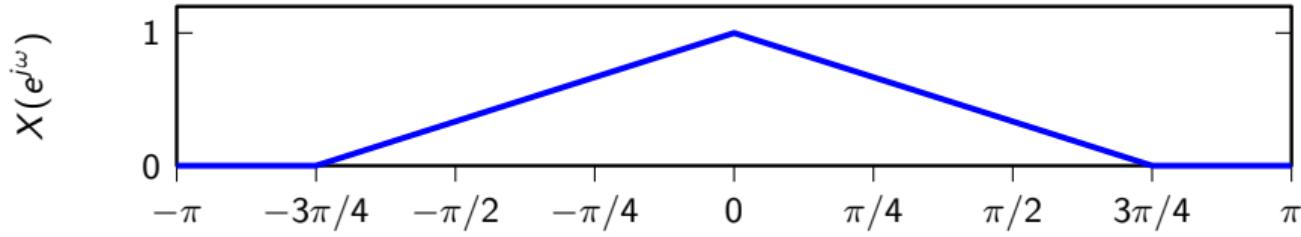
Downsampling by 2 with aliasing



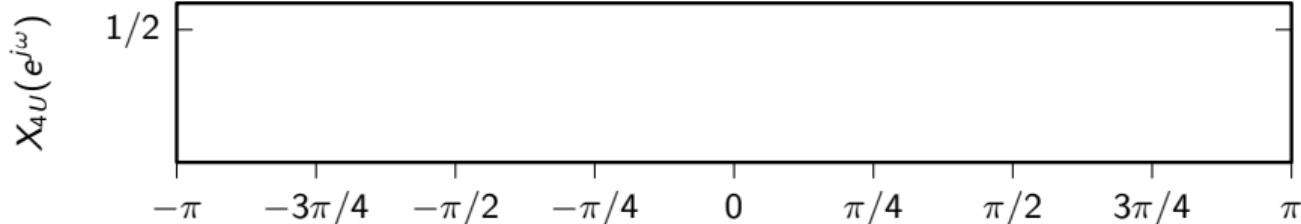
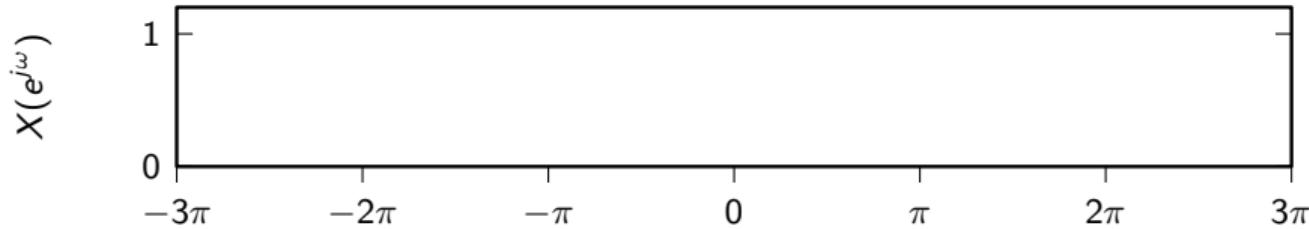
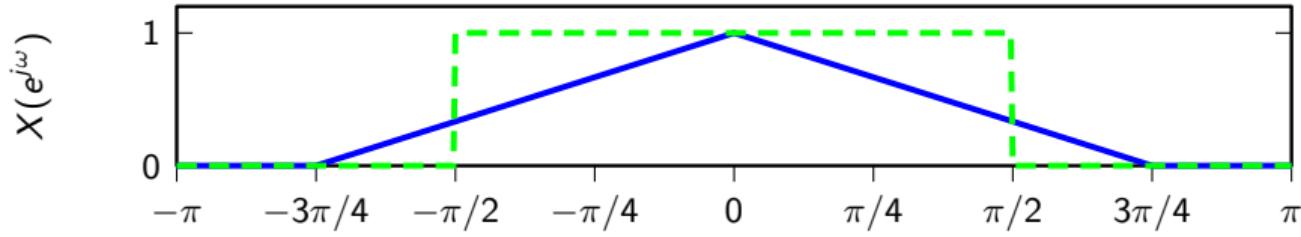
Downsampling by 2 with aliasing



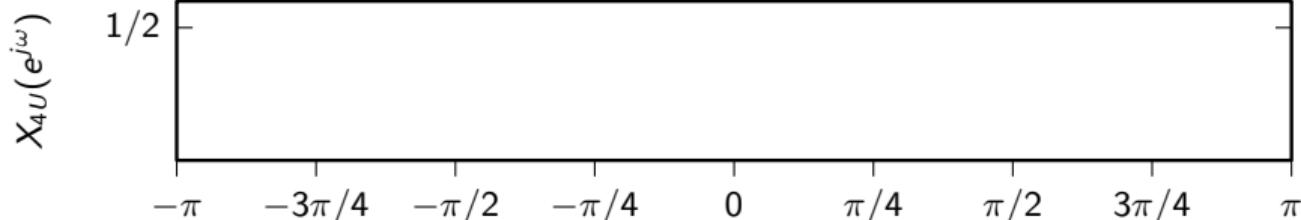
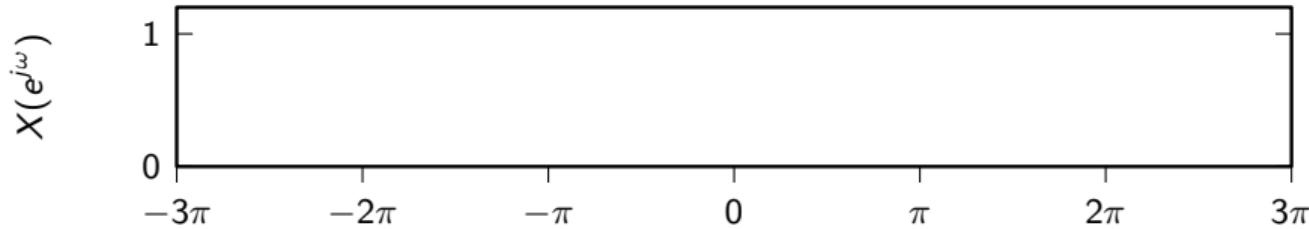
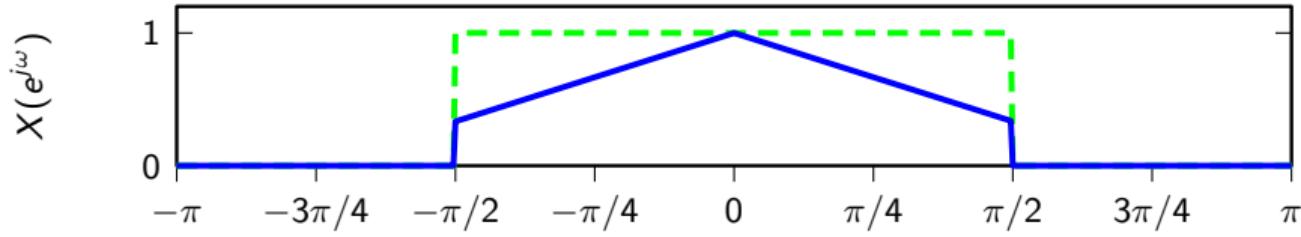
Downsampling by 2 with antialiasing filter



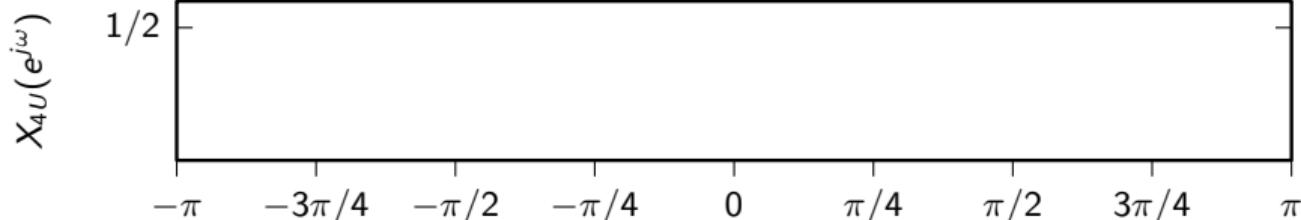
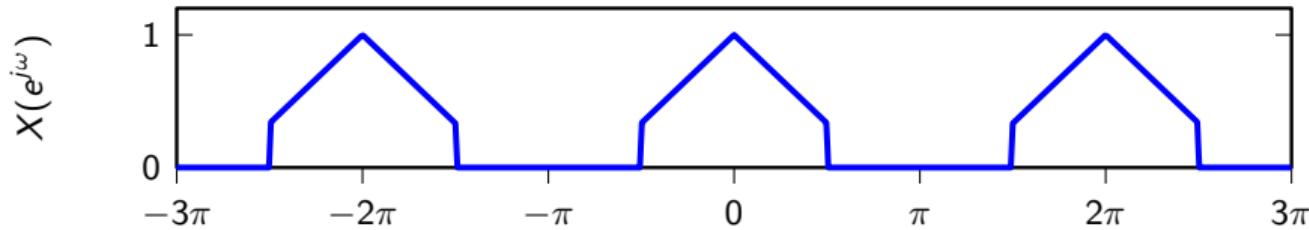
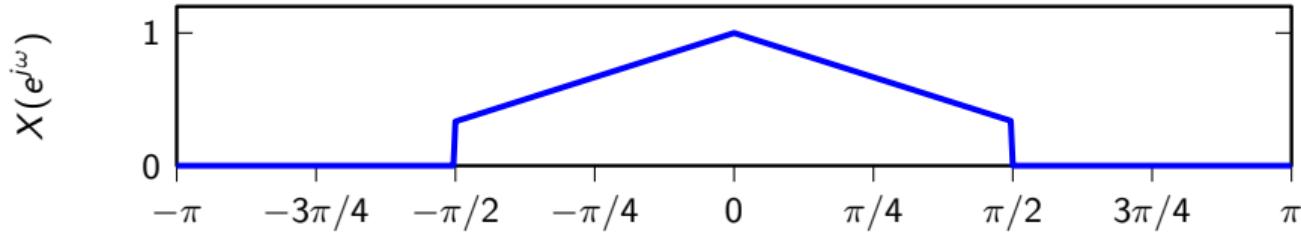
Downsampling by 2 with antialiasing filter



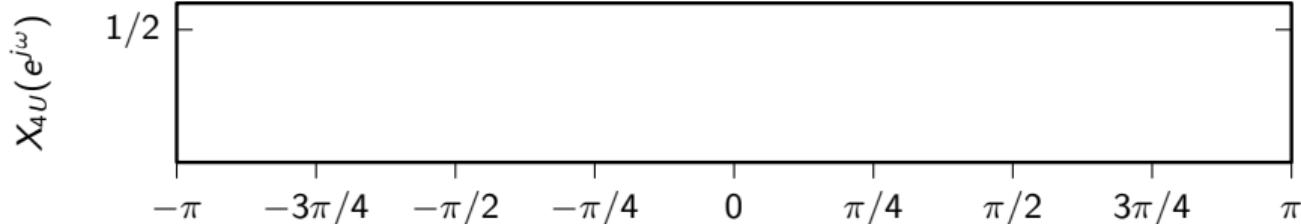
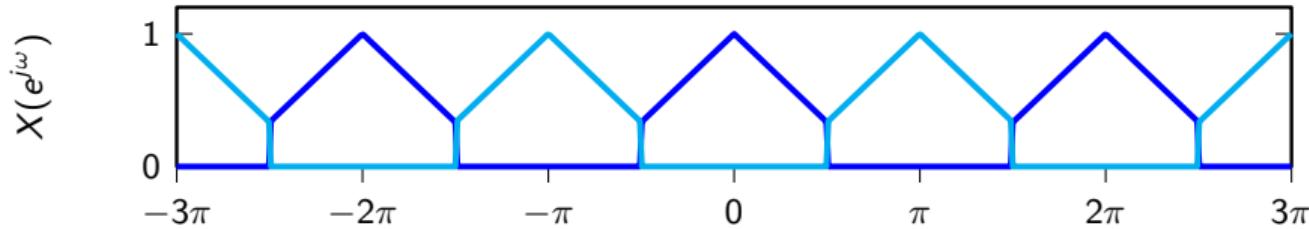
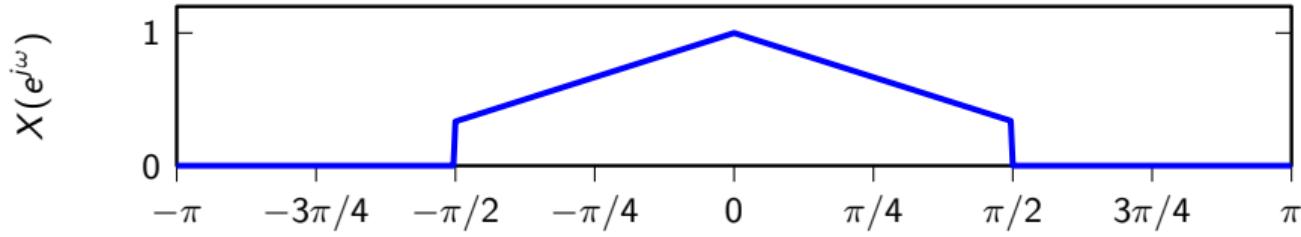
Downsampling by 2 with antialiasing filter



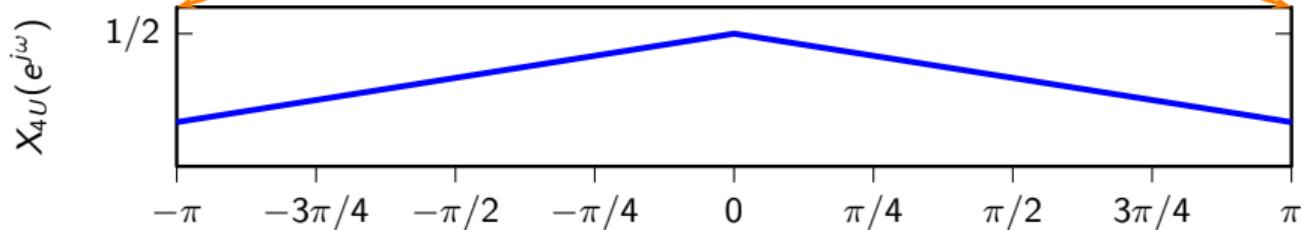
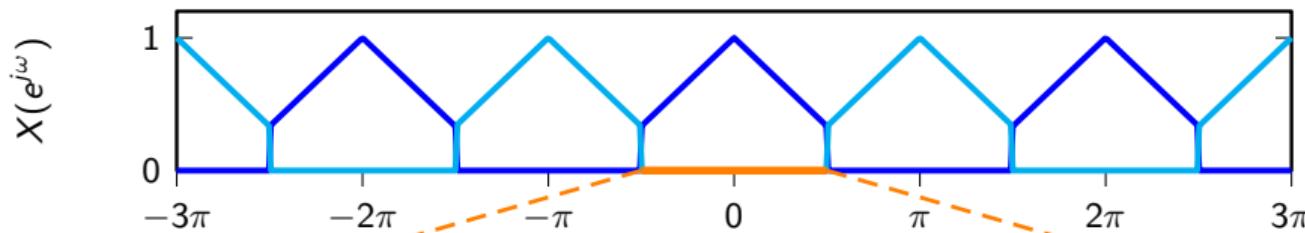
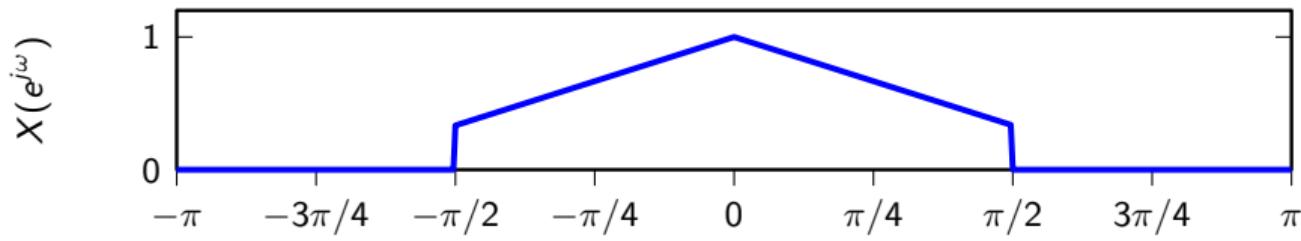
Downsampling by 2 with antialiasing filter



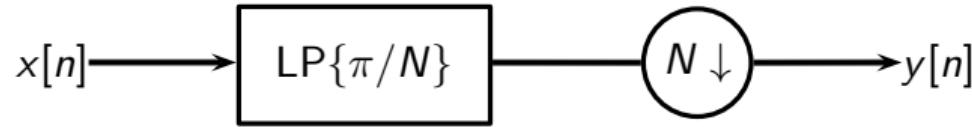
Downsampling by 2 with antialiasing filter



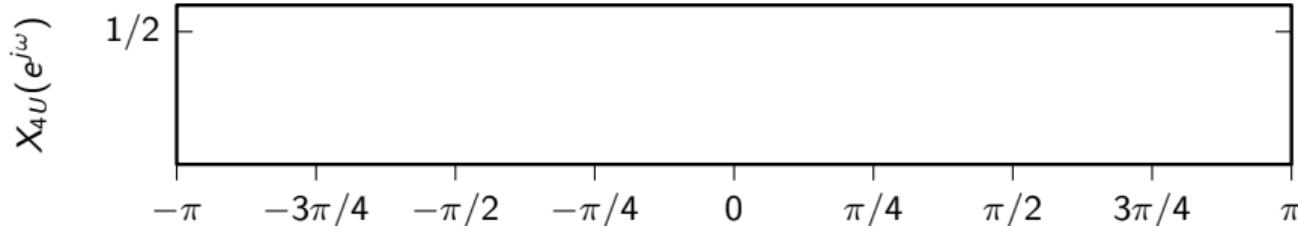
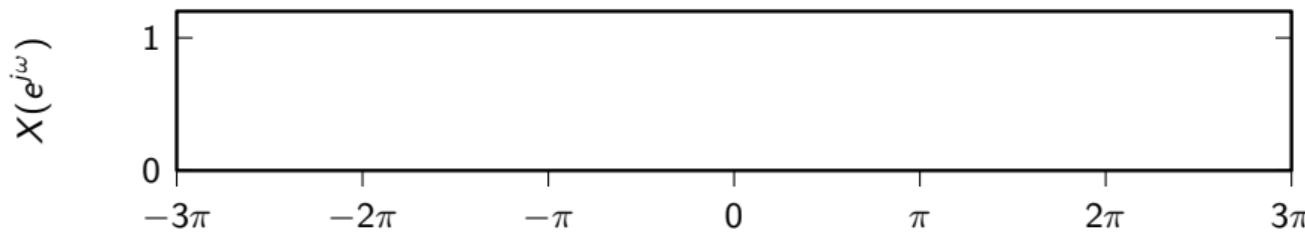
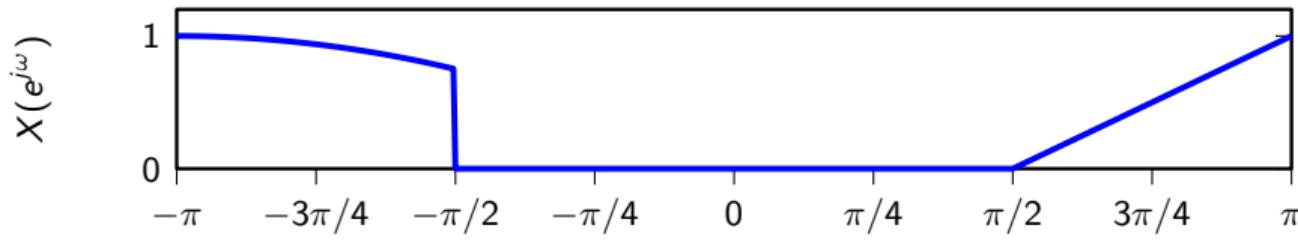
Downsampling by 2 with antialiasing filter



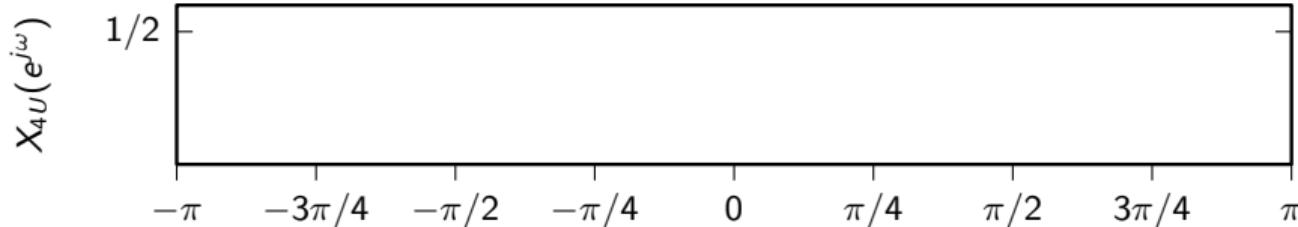
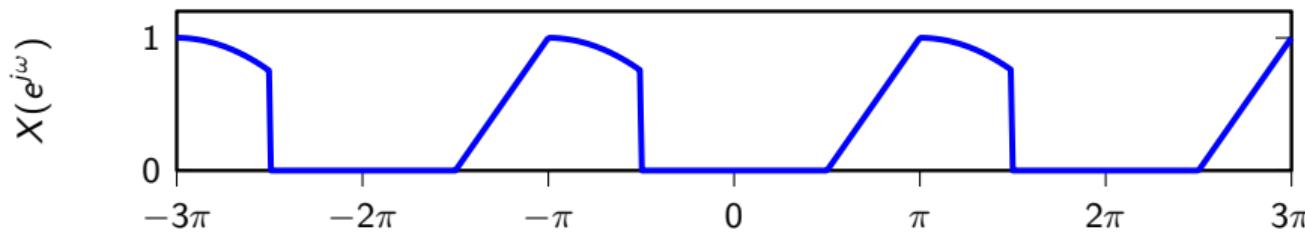
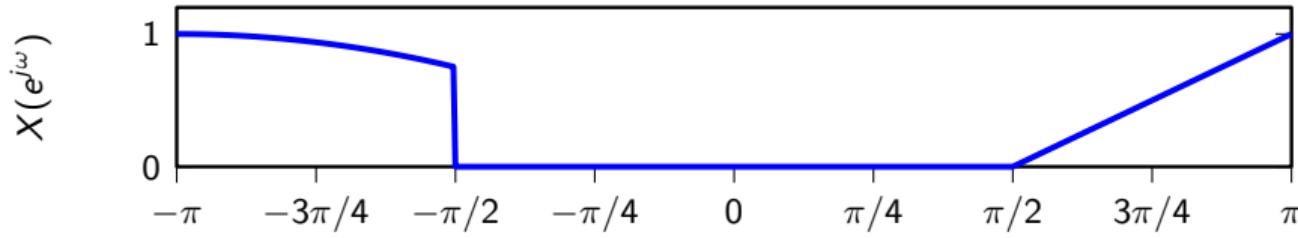
Downsampling



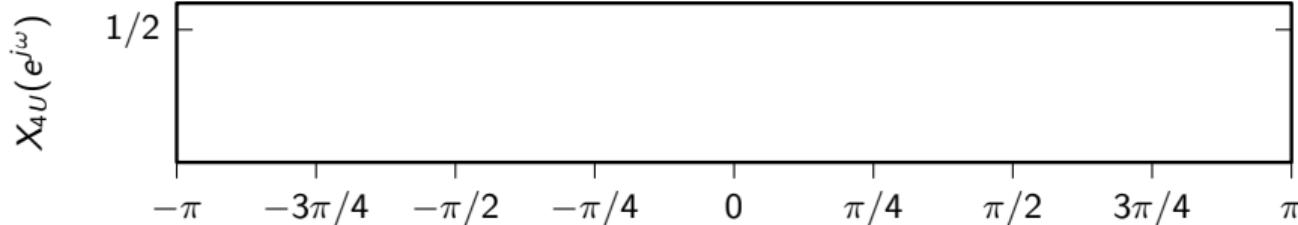
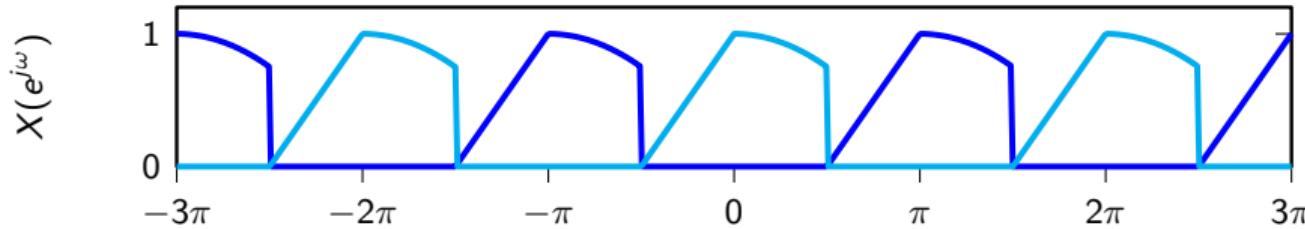
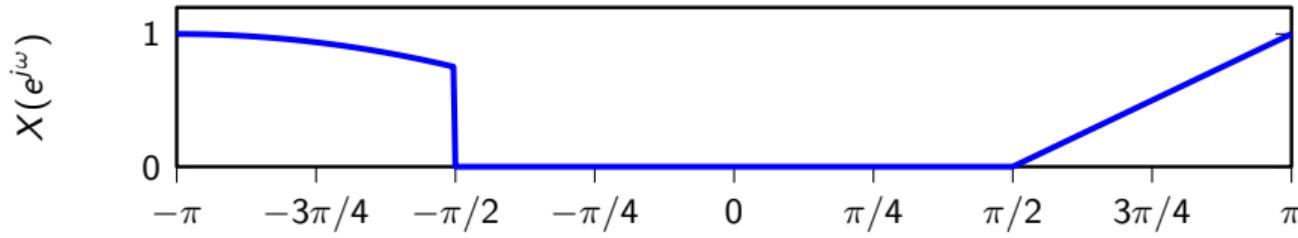
Downsampling by 2, careful with highpass signals



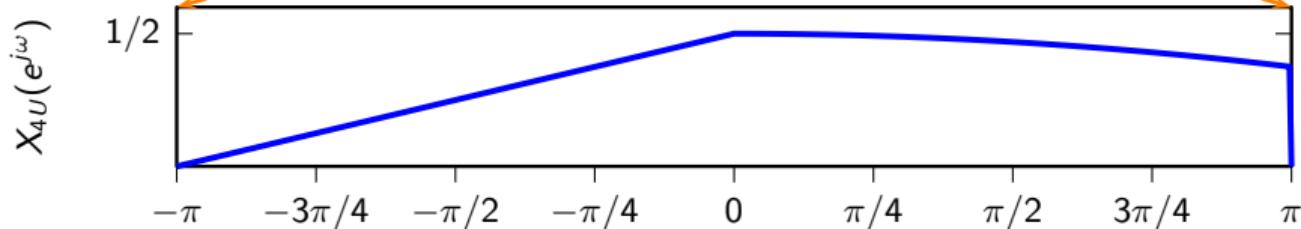
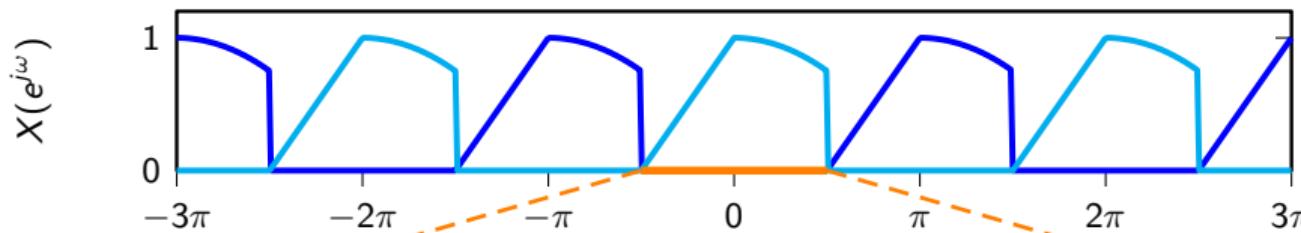
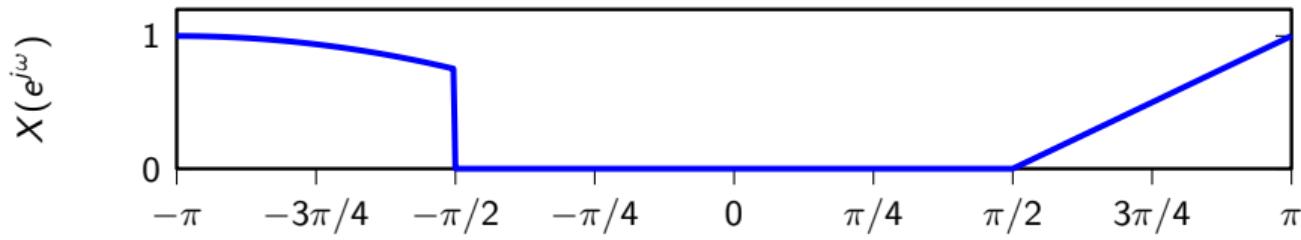
Downsampling by 2, careful with highpass signals



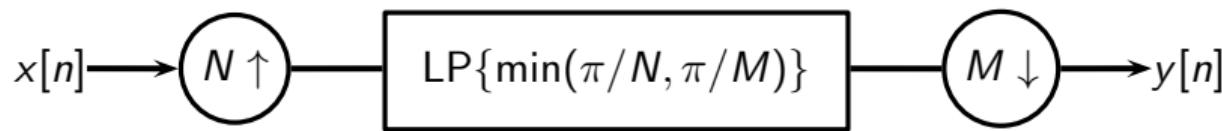
Downsampling by 2, careful with highpass signals



Downsampling by 2, careful with highpass signals



Rational Sampling Rate Change

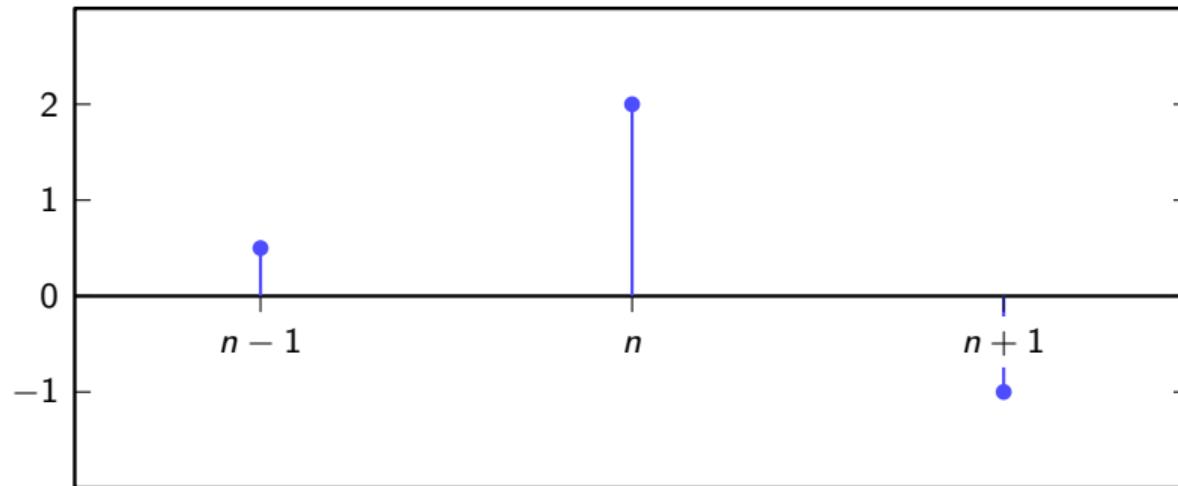


Rational Sampling Rate Change

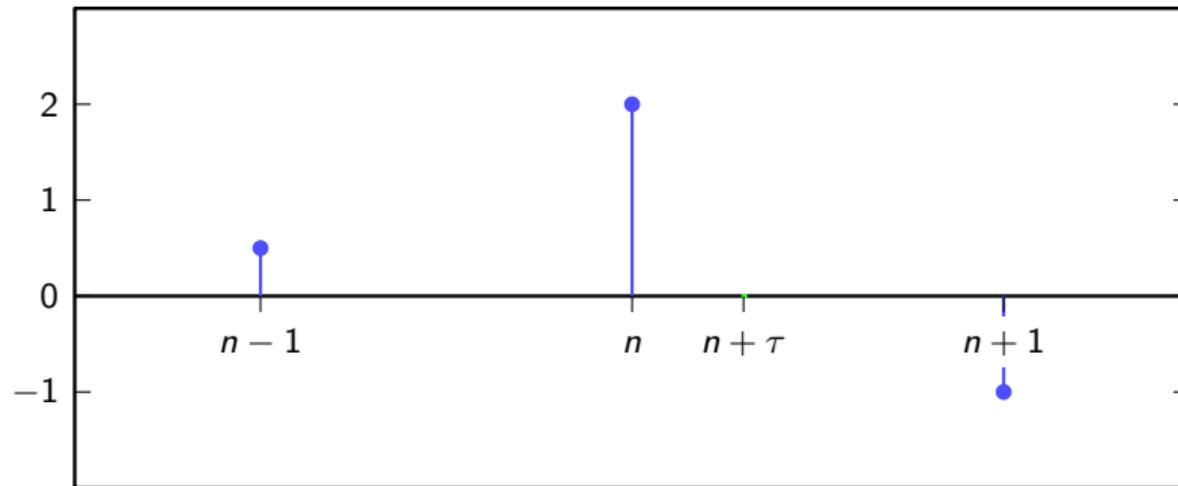
Example CD to DVD:

- ▶ CD: $F_s = 44100\text{Hz}$
- ▶ DVD: $F_s = 48000\text{Hz}$
- ▶ $\frac{N}{M} = \frac{160}{147}$
- ▶ in practice, we use time-varying local interpolation

Subsample Interpolation



Subsample Interpolation



Subsample Interpolation

- ▶ we want to compute $x(n + \tau)$, with $|\tau| < 1/2$

- ▶ local Lagrange approximation around n

$$x_L(n; t) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(t)$$

$$L_k^{(N)}(t) = \prod_{\substack{i=-N \\ i \neq n}}^N \frac{t-i}{k-i} \quad k = -N, \dots, N$$

- ▶ $x(n + \tau) \approx x_L(n; \tau)$

Subsample Interpolation

- ▶ we want to compute $x(n + \tau)$, with $|\tau| < 1/2$
- ▶ local Lagrange approximation around n

$$x_L(n; t) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(t)$$

$$L_k^{(N)}(t) = \prod_{\substack{i=-N \\ i \neq n}}^N \frac{t - i}{k - i} \quad k = -N, \dots, N$$

- ▶ $x(n + \tau) \approx x_L(n; \tau)$

Subsample Interpolation

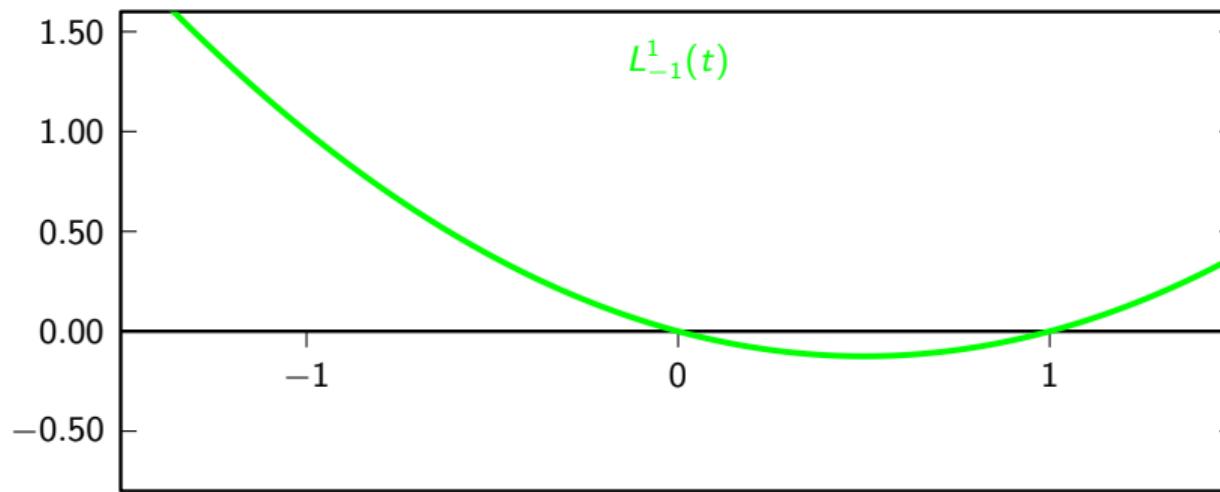
- ▶ we want to compute $x(n + \tau)$, with $|\tau| < 1/2$
- ▶ local Lagrange approximation around n

$$x_L(n; t) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(t)$$

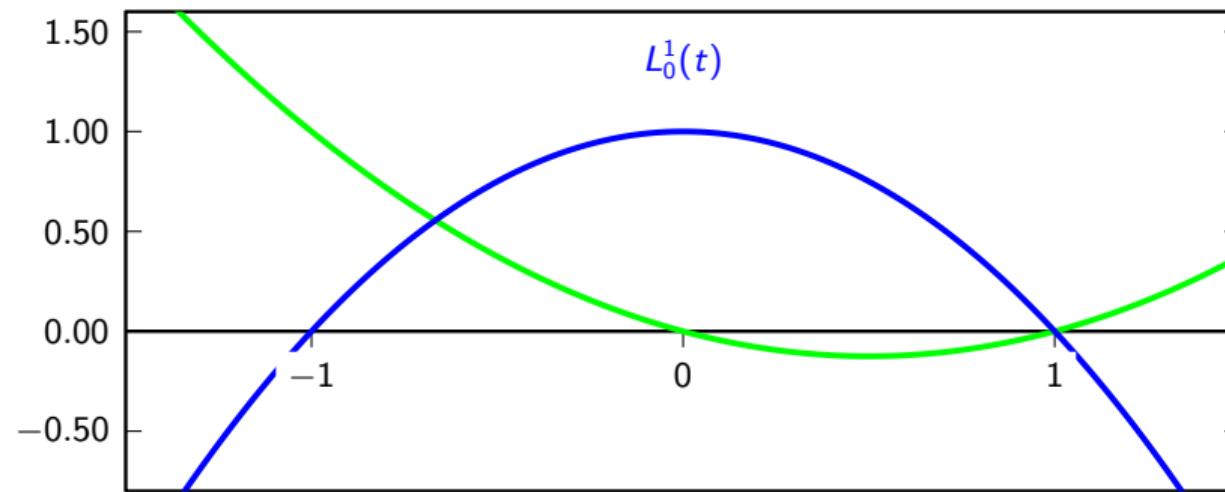
$$L_k^{(N)}(t) = \prod_{\substack{i=-N \\ i \neq n}}^N \frac{t - i}{k - i} \quad k = -N, \dots, N$$

- ▶ $x(n + \tau) \approx x_L(n; \tau)$

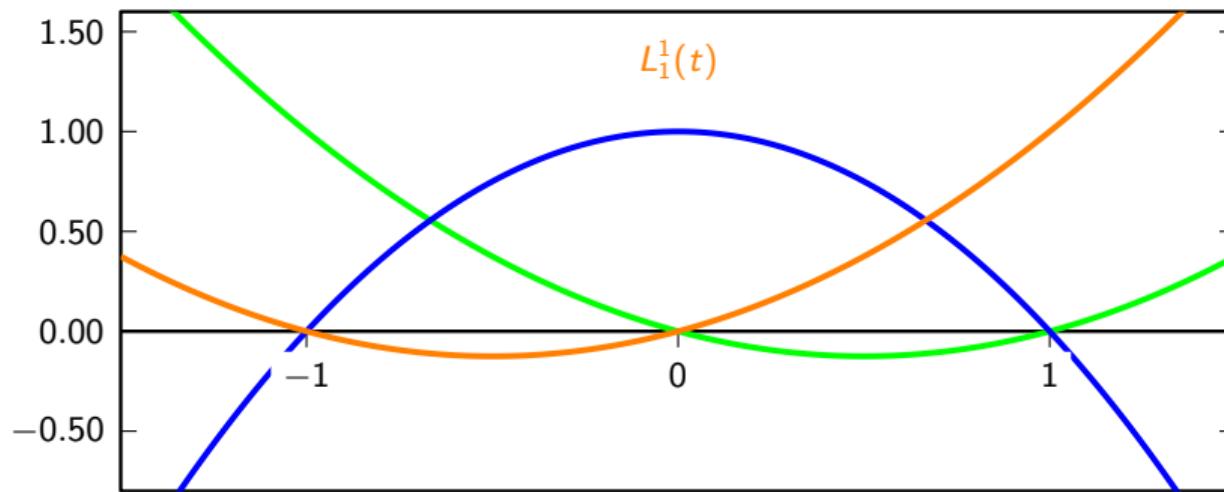
2nd-order Lagrange interpolation polynomials ($N = 1$)



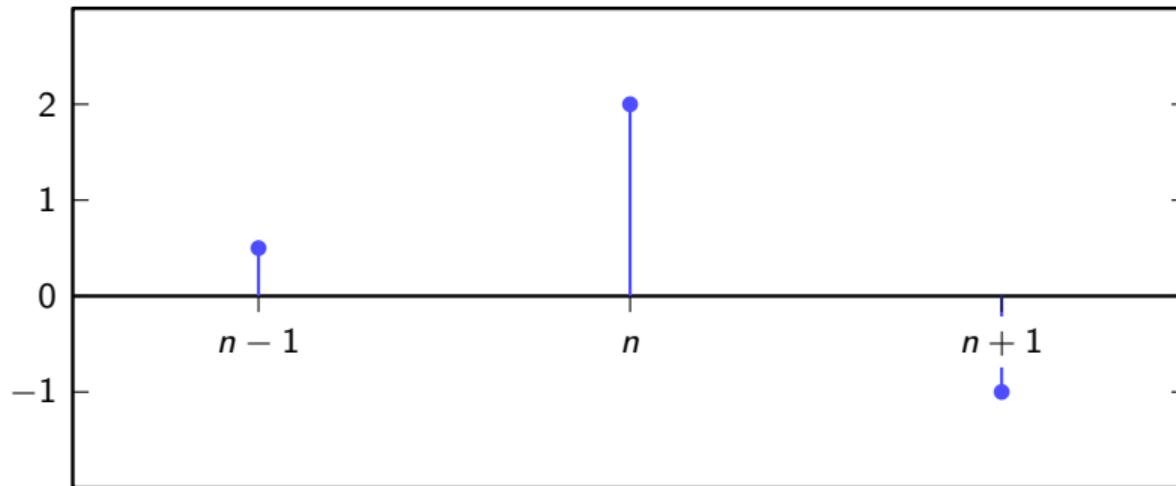
2nd-order Lagrange interpolation polynomials ($N = 1$)



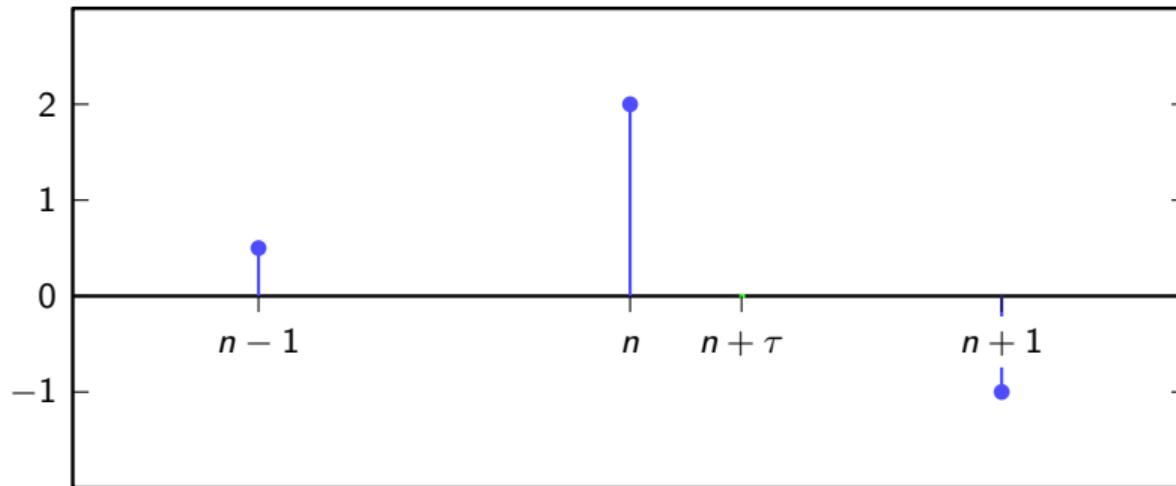
2nd-order Lagrange interpolation polynomials ($N = 1$)



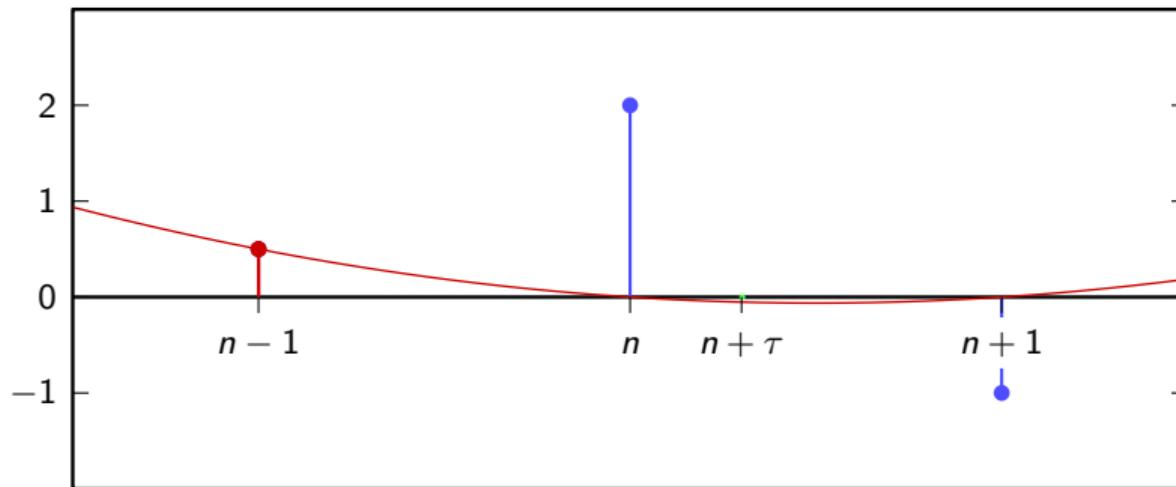
Lagrange interpolation ($N = 1$)



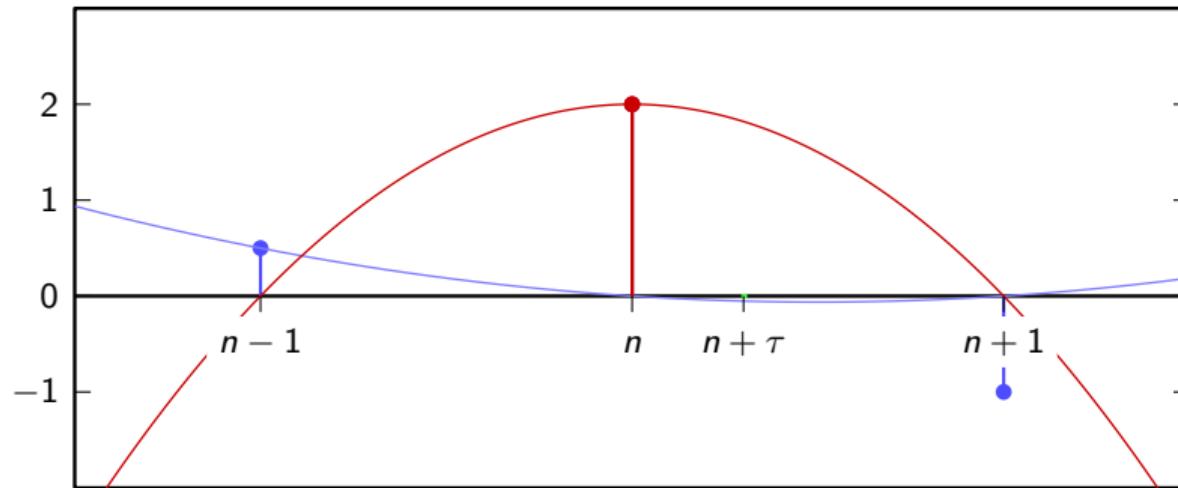
Lagrange interpolation ($N = 1$)



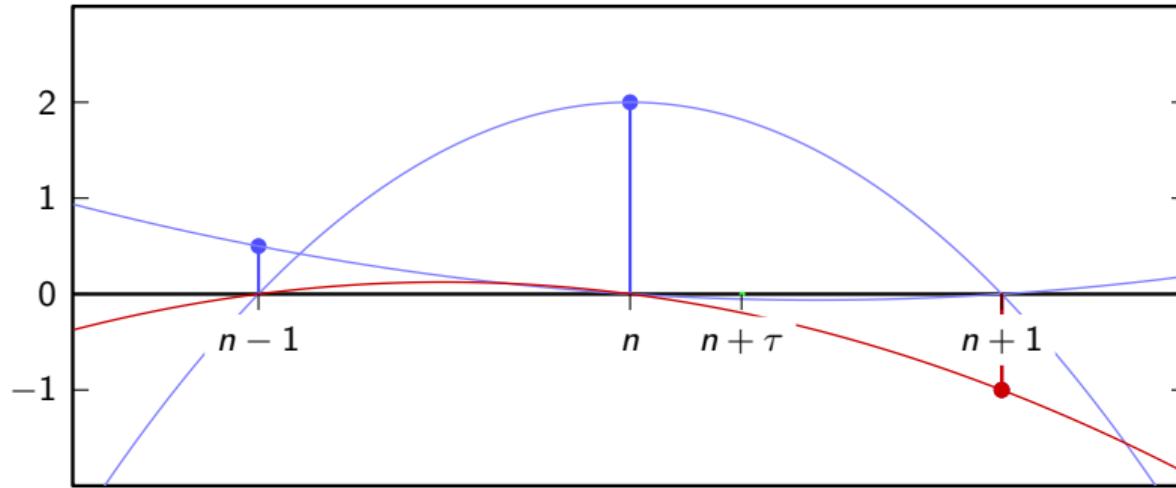
Lagrange interpolation ($N = 1$)



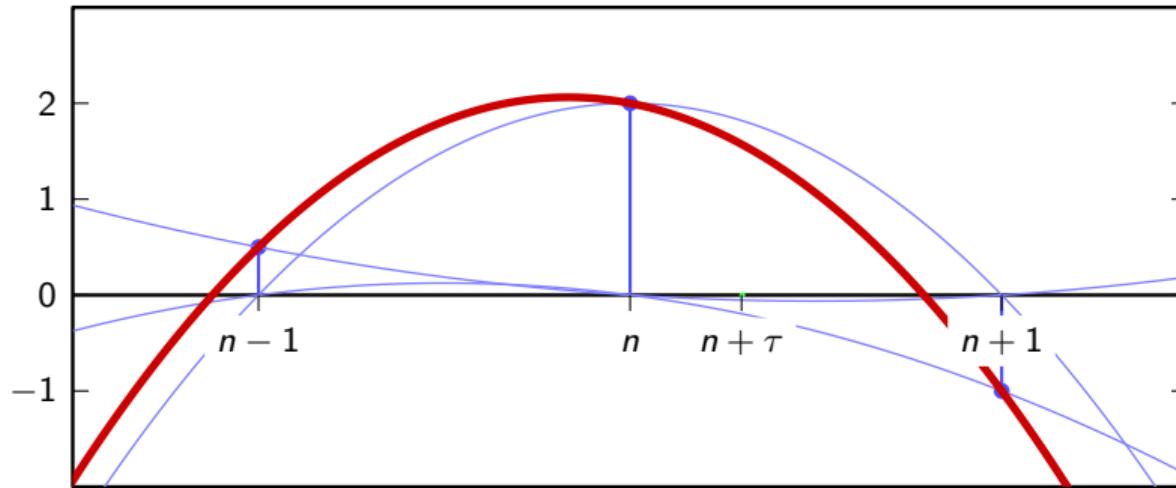
Lagrange interpolation ($N = 1$)



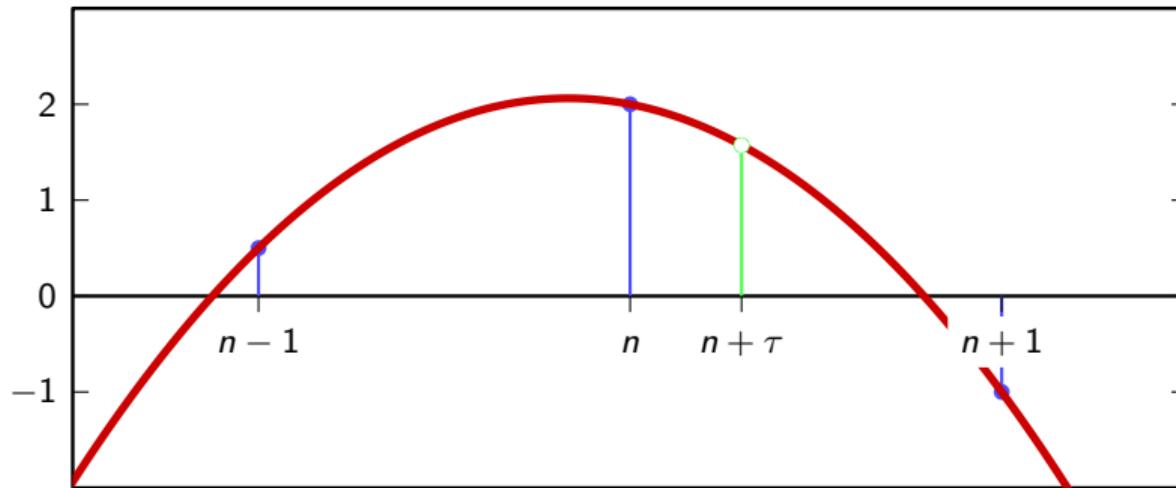
Lagrange interpolation ($N = 1$)



Lagrange interpolation ($N = 1$)



Lagrange interpolation ($N = 1$)



Lagrange interpolation as an FIR

- ▶ $x(n + \tau) \approx x_L(n; \tau)$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^N x[n-k] L_{-k}^{(N)}(\tau)$
- ▶ define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \dots, N$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n-k] d_\tau[k]$
- ▶ $x_L(n; \tau) = (x * d_\tau)[n]$
- ▶ $d_\tau[k]$ is a $(2N + 1)$ -tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- ▶ $x(n + \tau) \approx x_L(n; \tau)$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^N x[n-k] L_{-k}^{(N)}(\tau)$
- ▶ define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \dots, N$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n-k] d_\tau[k]$
- ▶ $x_L(n; \tau) = (x * d_\tau)[n]$
- ▶ $d_\tau[k]$ is a $(2N + 1)$ -tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- ▶ $x(n + \tau) \approx x_L(n; \tau)$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^N x[n-k] L_{-k}^{(N)}(\tau)$
- ▶ define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \dots, N$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n-k] d_\tau[k]$
- ▶ $x_L(n; \tau) = (x * d_\tau)[n]$
- ▶ $d_\tau[k]$ is a $(2N+1)$ -tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- ▶ $x(n + \tau) \approx x_L(n; \tau)$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^N x[n-k] L_{-k}^{(N)}(\tau)$
- ▶ define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \dots, N$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n-k] d_\tau[k]$
- ▶ $x_L(n; \tau) = (x * d_\tau)[n]$
- ▶ $d_\tau[k]$ is a $(2N + 1)$ -tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- ▶ $x(n + \tau) \approx x_L(n; \tau)$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^N x[n-k] L_{-k}^{(N)}(\tau)$
- ▶ define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \dots, N$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n-k] d_\tau[k]$
- ▶ $x_L(n; \tau) = (x * d_\tau)[n]$
- ▶ $d_\tau[k]$ is a $(2N+1)$ -tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- ▶ $x(n + \tau) \approx x_L(n; \tau)$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^N x[n-k] L_{-k}^{(N)}(\tau)$
- ▶ define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \dots, N$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n-k] d_\tau[k]$
- ▶ $x_L(n; \tau) = (x * d_\tau)[n]$
- ▶ $d_\tau[k]$ is a $(2N + 1)$ -tap FIR (dependent on τ)

Example ($N = 1$, second order approximation)

$$L_{-1}^{(1)}(t) = t \frac{t - 1}{2}$$

$$L_0^{(1)}(t) = (1 - t)(1 + t)$$

$$L_1^{(1)}(t) = t \frac{t + 1}{2}$$

Example ($N = 1$, second order approximation)

$$d_{0.2}[n] = \begin{cases} 0.12 & n = -1 \\ 0.96 & n = 0 \\ -0.08 & n = 1 \\ 0 & \text{otherwise} \end{cases}$$

CD to DVD, revisited

for every 147 CD samples, generate 160 DVD samples

Fractional resampling: algorithm

sampling rate change A/B : for every B input point, generate A output points

Initialization (pattern will repeat every A output points)

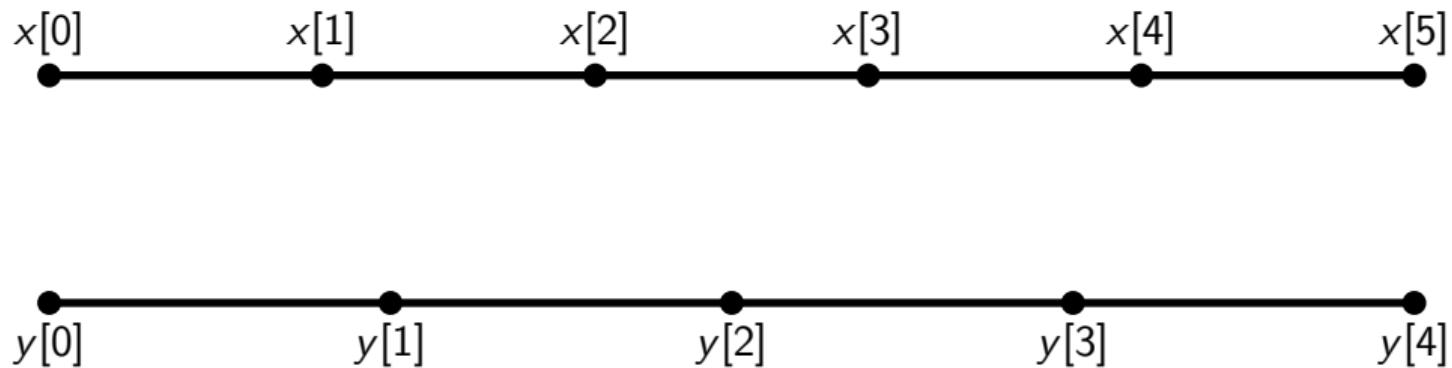
- ▶ for each output index $0 \leq m < A$ find closest (< 0.5) input point
- ▶ generate a FIR interpolation filter using difference between output and anchor

For every block of A output values $y[n]$:

- ▶ use all filters in turn
- ▶ generate output points

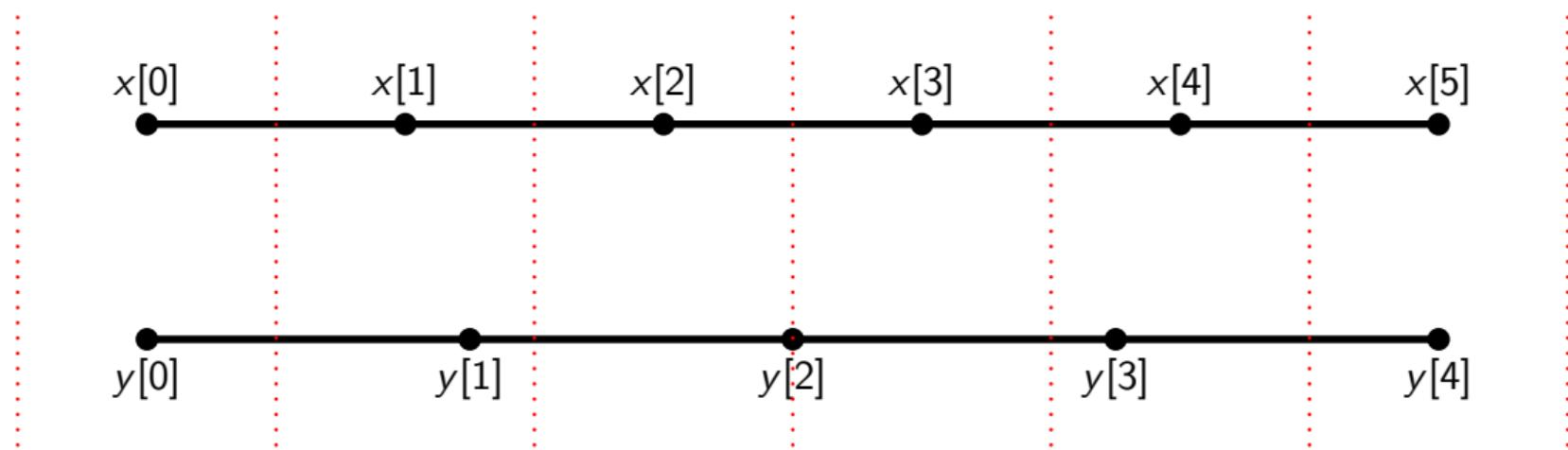
Fractional resampling: sample rate reduction

Downsampling: $\frac{A}{B} = \frac{4}{5}$



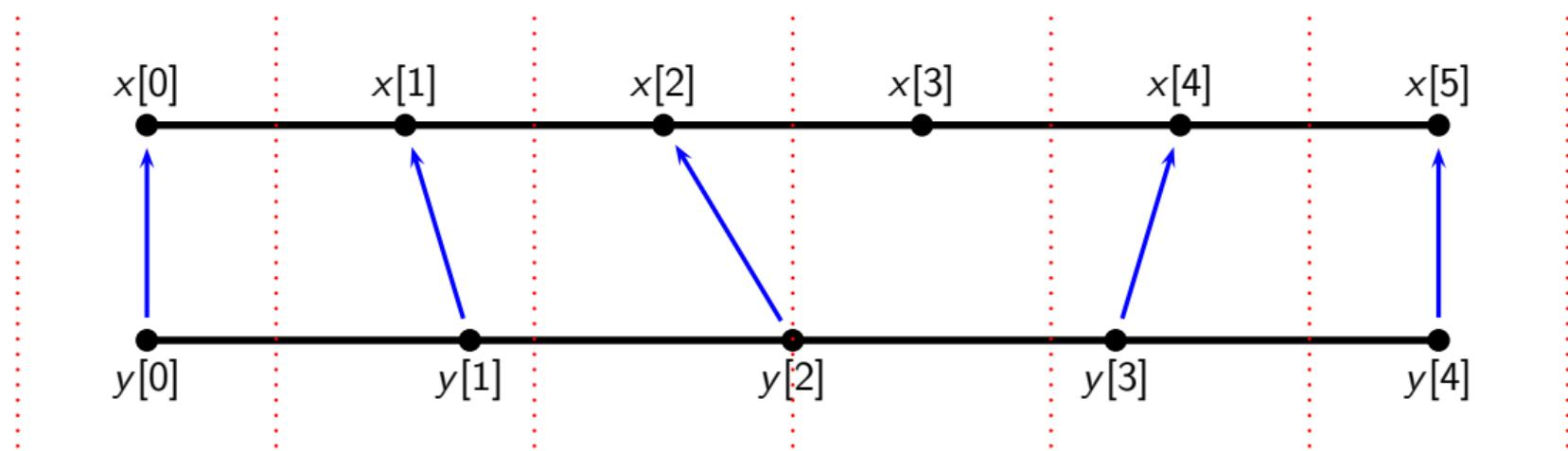
Fractional resampling: sample rate reduction

Downsampling: $\frac{A}{B} = \frac{4}{5}$



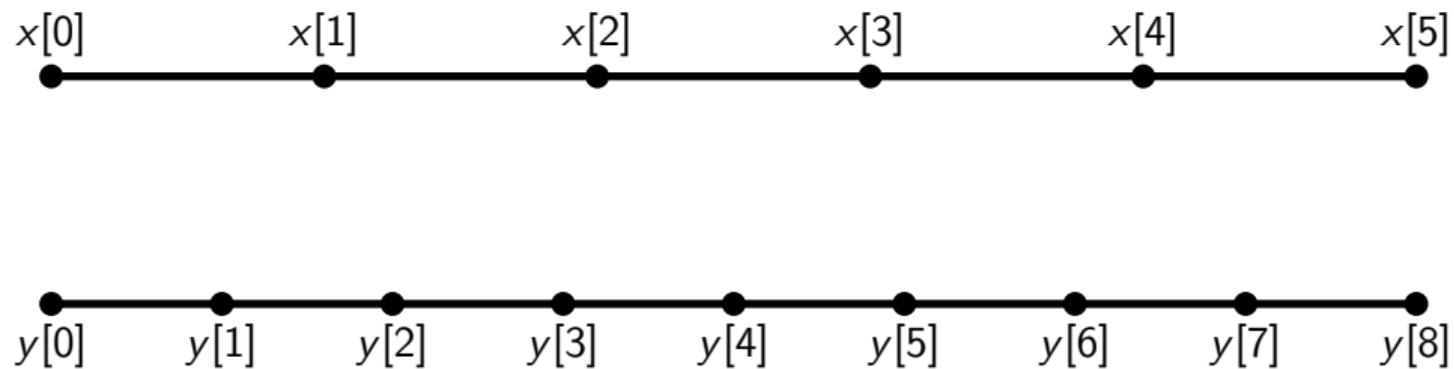
Fractional resampling: sample rate reduction

Downsampling: $\frac{A}{B} = \frac{4}{5}$



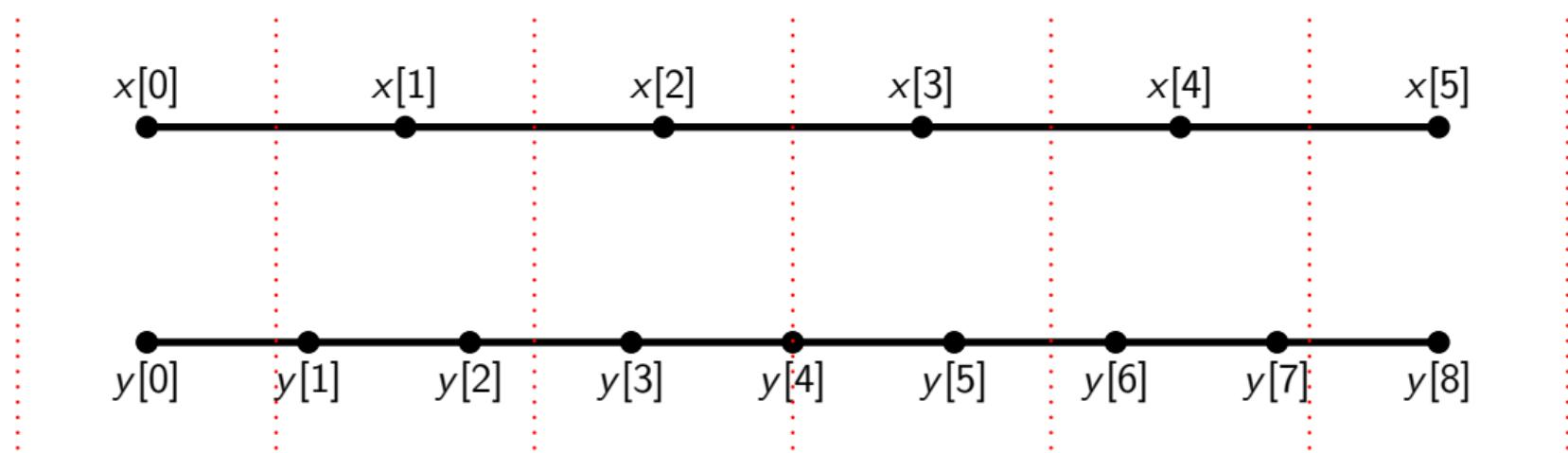
Fractional resampling: sample rate increase

Upsampling: $\frac{A}{B} = \frac{8}{5}$



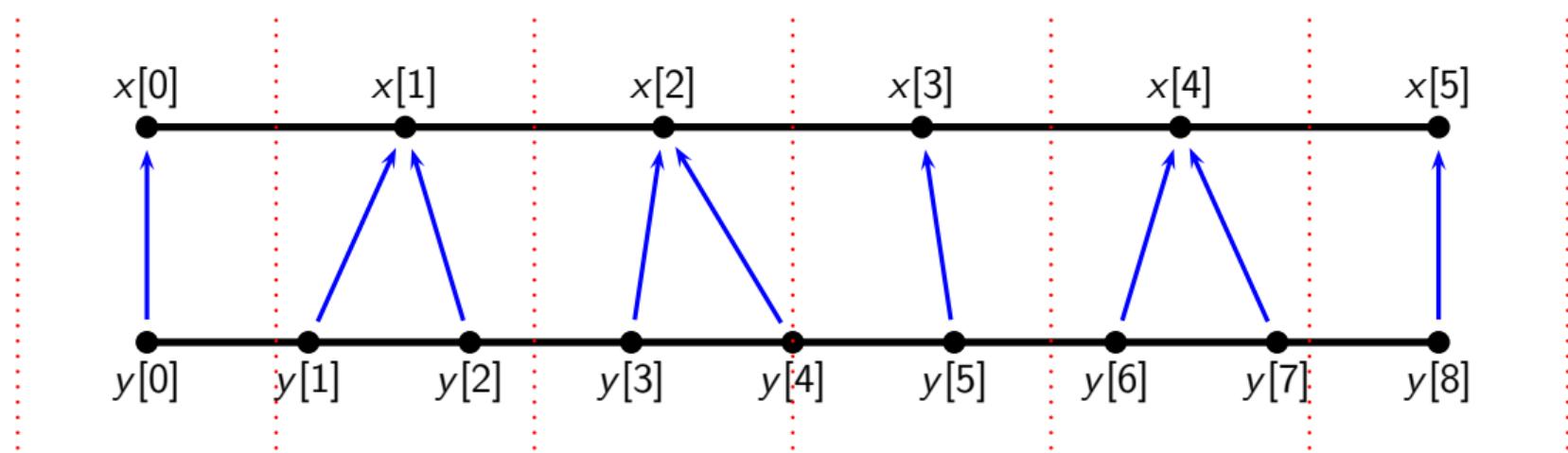
Fractional resampling: sample rate increase

Upsampling: $\frac{A}{B} = \frac{8}{5}$

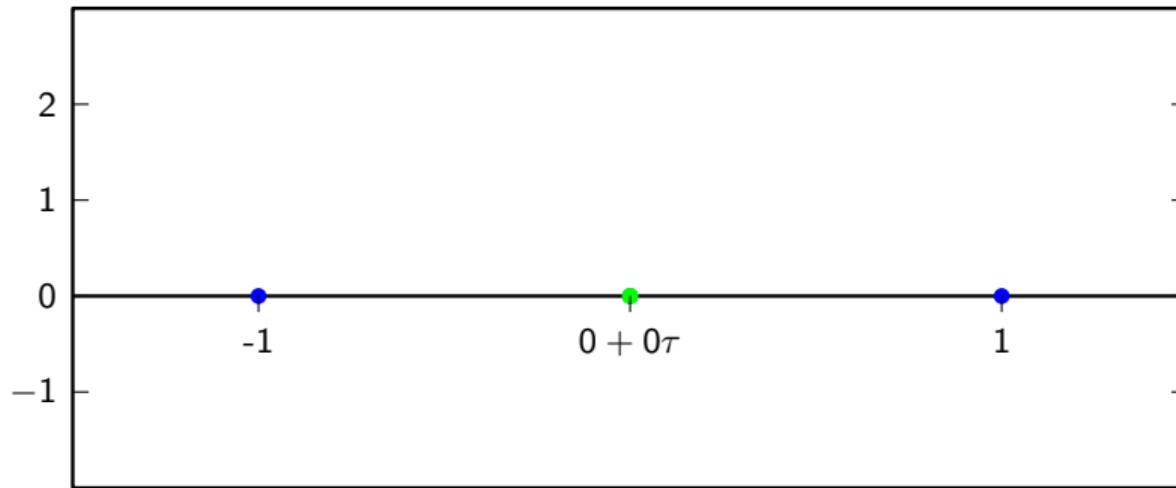


Fractional resampling: sample rate increase

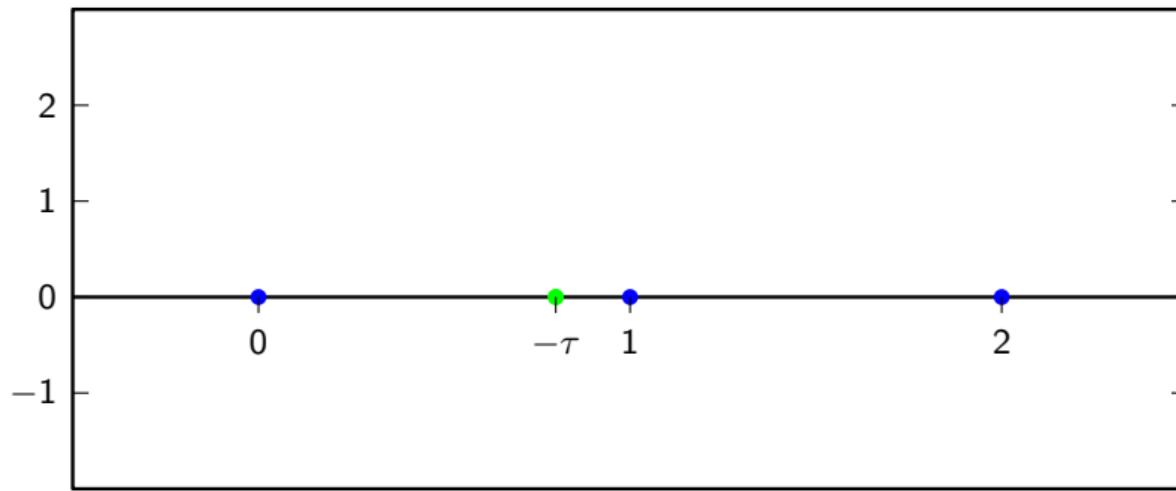
Upsampling: $\frac{A}{B} = \frac{8}{5}$



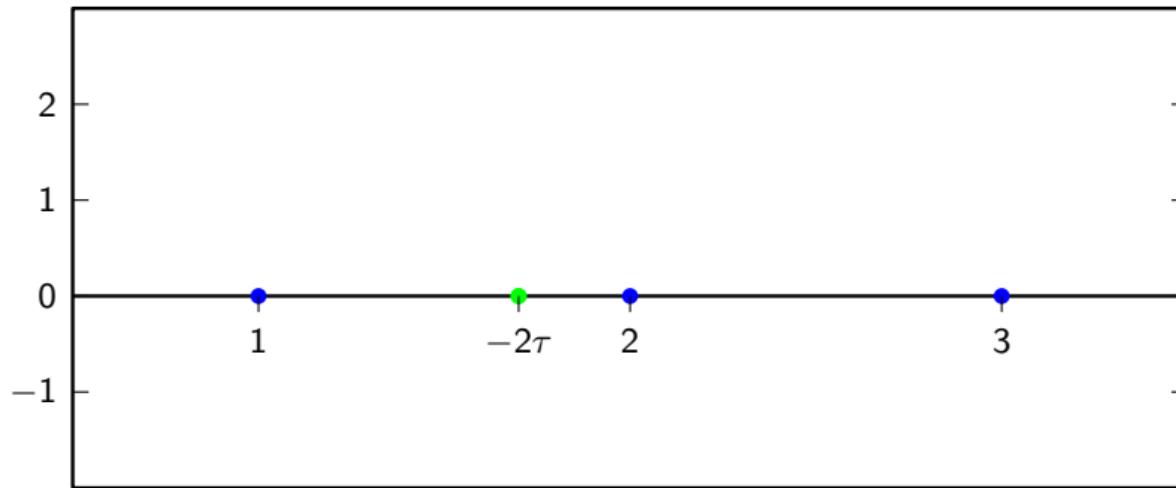
CD to DVD, revisited



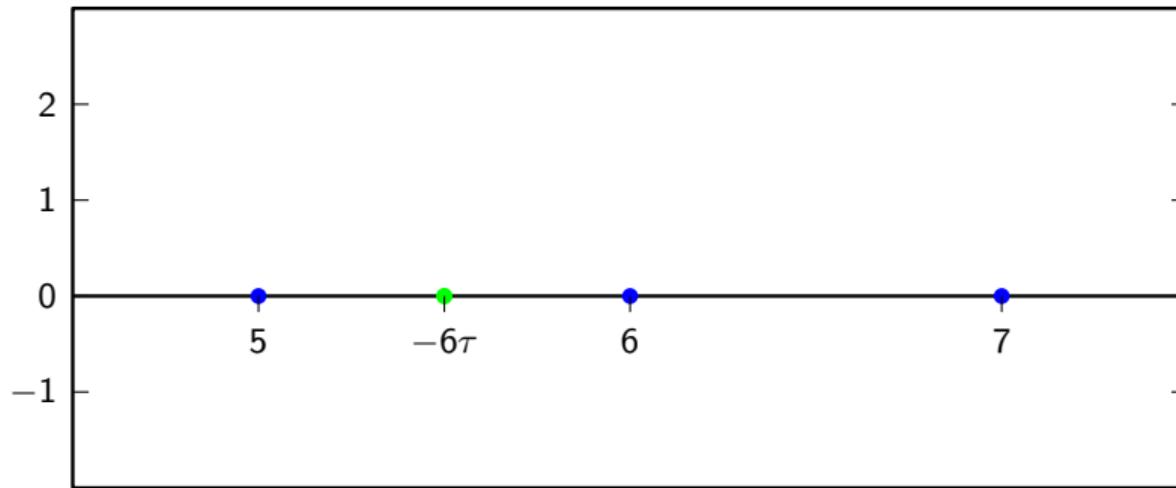
CD to DVD, revisited, $\tau[1] = 0.06875$



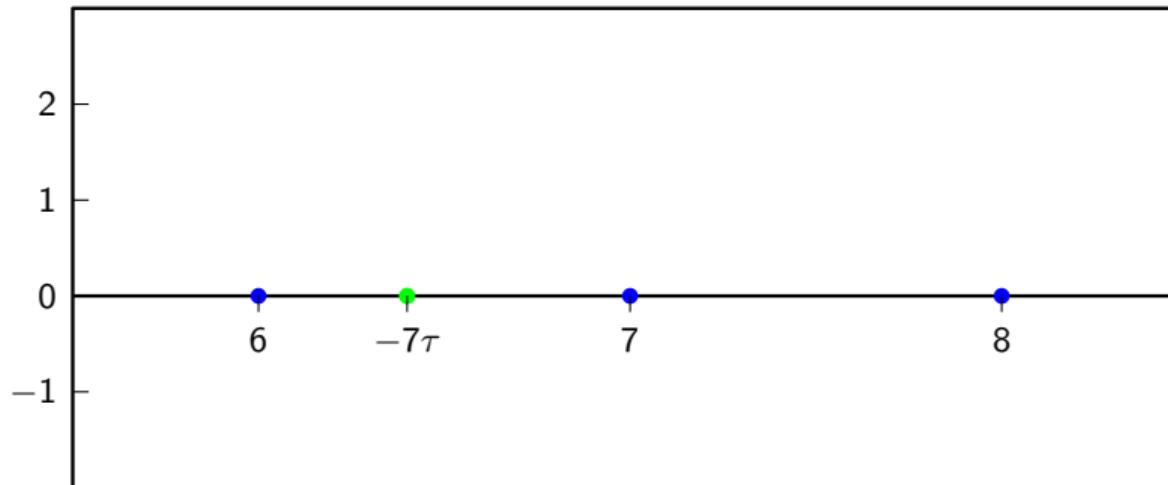
CD to DVD, revisited



CD to DVD, revisited

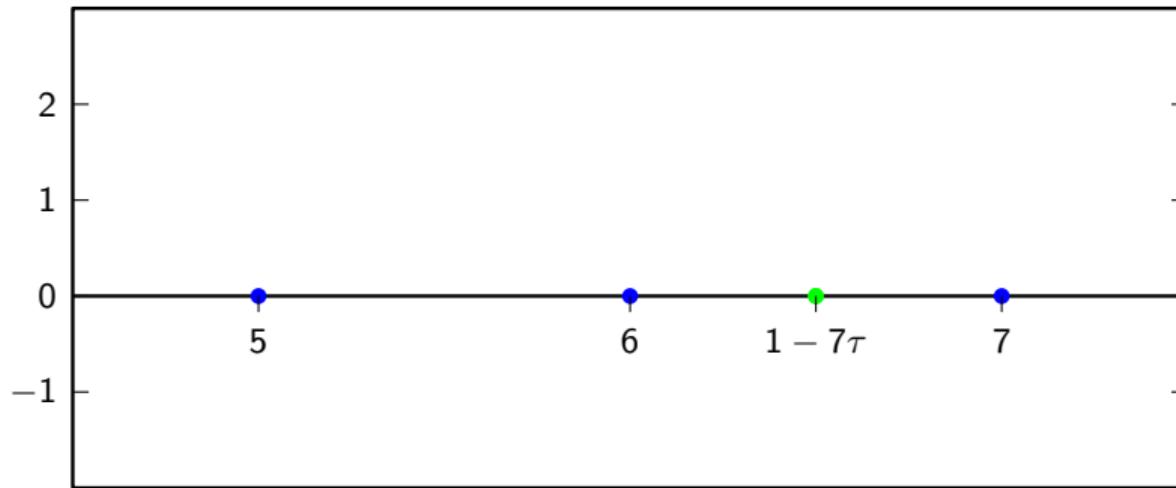


CD to DVD, revisited

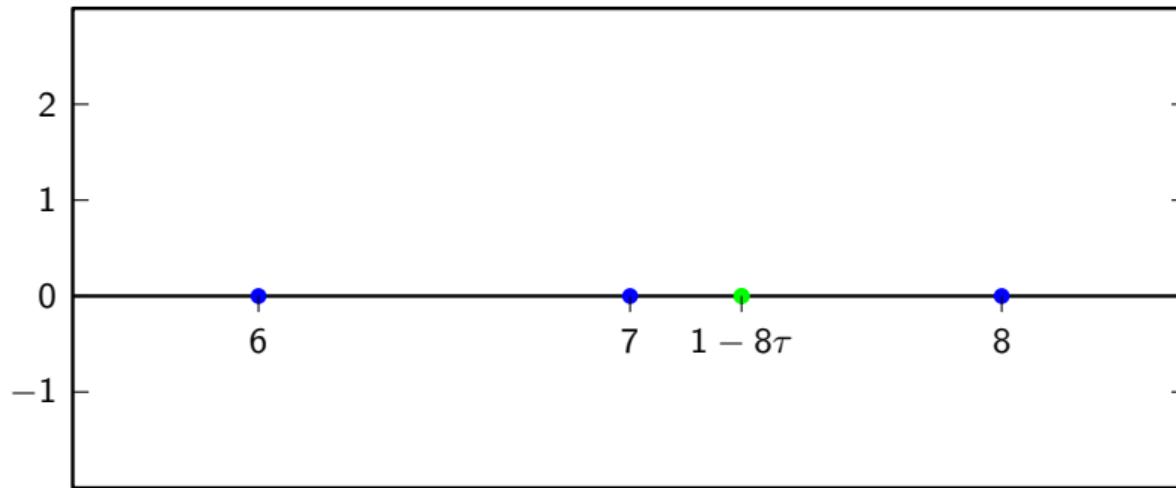


but $-7\tau < -0.5$

CD to DVD, revisited: repeat a sample



CD to DVD, revisited



CD to DVD, revisited

efficient local interpolation with 160 3-tap filters, used in sequence

COM303: Digital Signal Processing

Lecture 19: Quantization

overview

- ▶ quantization
- ▶ A/D and D/A converters
- ▶ oversampling

quantization

Overview:

- ▶ Quantization
- ▶ Uniform quantization and error analysis
- ▶ Clipping, saturation, companding

Overview:

- ▶ Quantization
- ▶ Uniform quantization and error analysis
- ▶ Clipping, saturation, companding

Overview:

- ▶ Quantization
- ▶ Uniform quantization and error analysis
- ▶ Clipping, saturation, companding

Quantization

- ▶ digital devices can only deal with integers (b bits per sample)
- ▶ we need to map the range of a signal onto a finite set of values
- ▶ irreversible loss of information → quantization noise

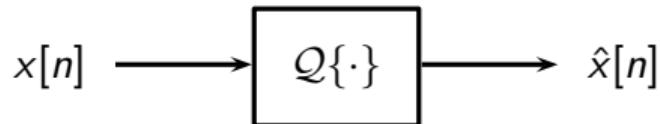
Quantization

- ▶ digital devices can only deal with integers (b bits per sample)
- ▶ we need to map the range of a signal onto a finite set of values
- ▶ irreversible loss of information → quantization noise

Quantization

- ▶ digital devices can only deal with integers (b bits per sample)
- ▶ we need to map the range of a signal onto a finite set of values
- ▶ irreversible loss of information → quantization noise

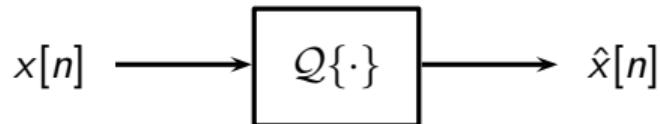
Quantization schemes



Several factors at play:

- ▶ storage budget (bits per sample)
- ▶ storage scheme (fixed point, floating point)
- ▶ properties of the input
 - range
 - probability distribution

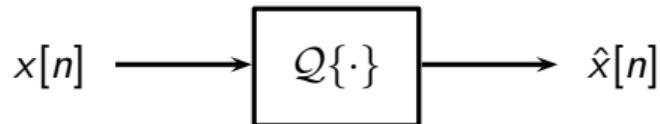
Quantization schemes



Several factors at play:

- ▶ storage budget (bits per sample)
- ▶ storage scheme (fixed point, floating point)
- ▶ properties of the input
 - range
 - probability distribution

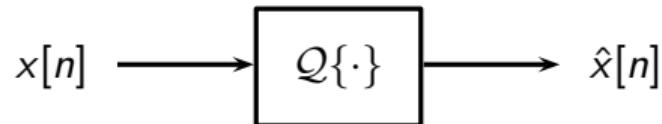
Quantization schemes



Several factors at play:

- ▶ storage budget (bits per sample)
- ▶ storage scheme (fixed point, floating point)
- ▶ properties of the input
 - range
 - probability distribution

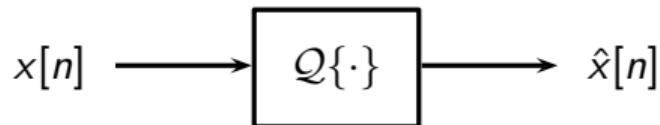
Quantization schemes



Several factors at play:

- ▶ storage budget (bits per sample)
- ▶ storage scheme (fixed point, floating point)
- ▶ properties of the input
 - range
 - probability distribution

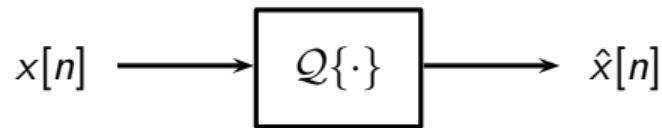
Quantization schemes



Several factors at play:

- ▶ storage budget (bits per sample)
- ▶ storage scheme (fixed point, floating point)
- ▶ properties of the input
 - range
 - probability distribution

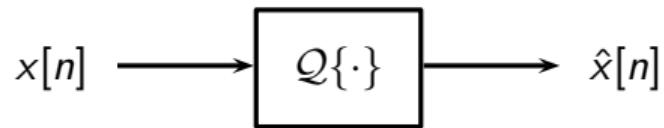
Scalar quantization



The simplest quantizer:

- ▶ each sample is encoded individually (hence *scalar*)
- ▶ each sample is quantized independently (memoryless quantization)
- ▶ each sample is encoded using R bits

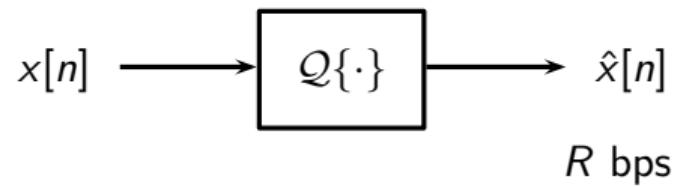
Scalar quantization



The simplest quantizer:

- ▶ each sample is encoded individually (hence *scalar*)
- ▶ each sample is quantized independently (memoryless quantization)
- ▶ each sample is encoded using R bits

Scalar quantization



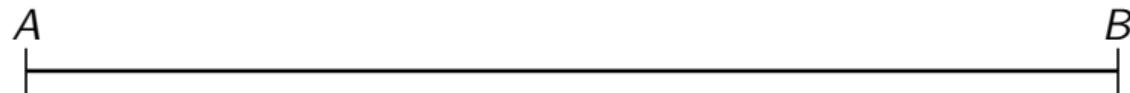
The simplest quantizer:

- ▶ each sample is encoded individually (hence *scalar*)
- ▶ each sample is quantized independently (memoryless quantization)
- ▶ each sample is encoded using R bits

Scalar quantization

Input signal: $A \leq x[n] \leq B$ (A, B can be ∞)

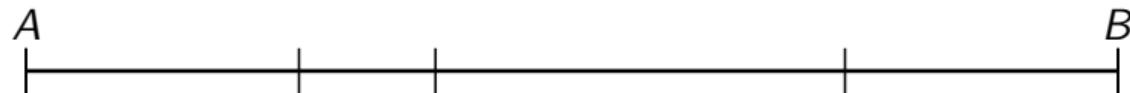
- ▶ each sample quantized over 2^R possible values $\Rightarrow 2^R$ intervals.
- ▶ each interval associated to a quantization value



Scalar quantization

Input signal: $A \leq x[n] \leq B$ (A, B can be ∞)

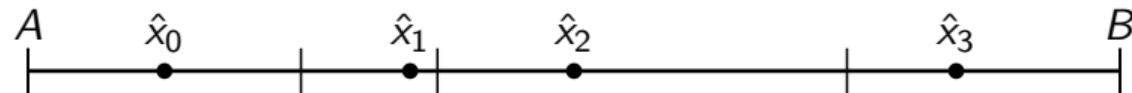
- ▶ each sample quantized over 2^R possible values $\Rightarrow 2^R$ intervals.
- ▶ each interval associated to a quantization value



Scalar quantization

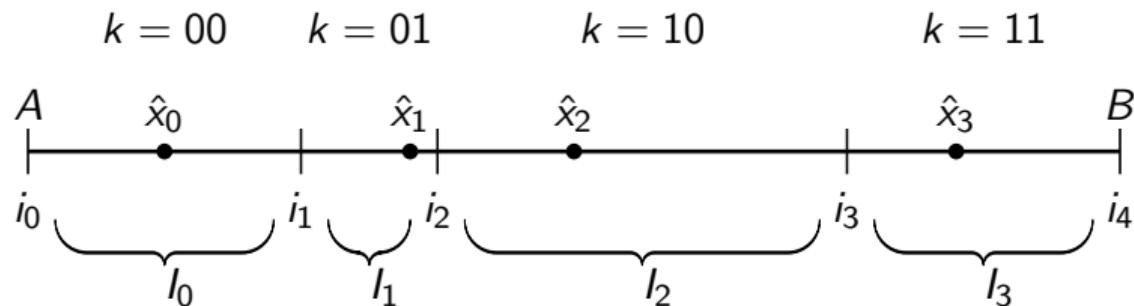
Input signal: $A \leq x[n] \leq B$ (A, B can be ∞)

- ▶ each sample quantized over 2^R possible values $\Rightarrow 2^R$ intervals.
- ▶ each interval associated to a quantization value



Scalar quantization

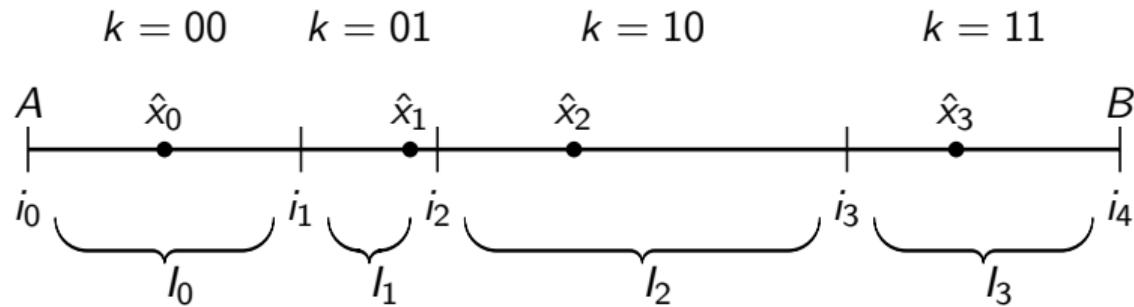
Example for $R = 2$:



- ▶ what are the optimal interval boundaries i_k ?
- ▶ what are the optimal quantization values \hat{x}_k ?

Scalar quantization

Example for $R = 2$:



- ▶ what are the optimal interval boundaries i_k ?
- ▶ what are the optimal quantization values \hat{x}_k ?

Optimal Quantization

The optimal quantizer minimizes the energy of the quantization error:

$$e[n] = Q\{x[n]\} - x[n] = \hat{x}[n] - x[n]$$

- ▶ model $x[n]$ as a stochastic process
- ▶ find the optimal i_k and \hat{x}_k that minimize $\sigma_e^2 = E[e^2[n]]$
- ▶ optimal quantizer will depend on the input's statistics

Optimal Quantization

The optimal quantizer minimizes the energy of the quantization error:

$$e[n] = Q\{x[n]\} - x[n] = \hat{x}[n] - x[n]$$

- ▶ model $x[n]$ as a stochastic process
- ▶ find the optimal i_k and \hat{x}_k that minimize $\sigma_e^2 = E[e^2[n]]$
- ▶ optimal quantizer will depend on the input's statistics

Optimal Quantization

The optimal quantizer minimizes the energy of the quantization error:

$$e[n] = Q\{x[n]\} - x[n] = \hat{x}[n] - x[n]$$

- ▶ model $x[n]$ as a stochastic process
- ▶ find the optimal i_k and \hat{x}_k that minimize $\sigma_e^2 = E[e^2[n]]$
- ▶ optimal quantizer will depend on the input's statistics

Quantization MSE

$$\begin{aligned}\sigma_e^2 &= \int_{-\infty}^{\infty} (x - Q\{x\})^2 f_x(x) dx \\ &= \sum_{k=0}^{2^R-1} \int_{i_k}^{i_{k+1}} (x - \hat{x}_k)^2 f_x(x) dx\end{aligned}$$

find global minimum wrt i_k, \hat{x}_k

Quantization MSE

$$\begin{aligned}\sigma_e^2 &= \int_{-\infty}^{\infty} (x - Q\{x\})^2 f_x(x) dx \\ &= \sum_{k=0}^{2^R-1} \int_{i_k}^{i_{k+1}} (x - \hat{x}_k)^2 f_x(x) dx\end{aligned}$$

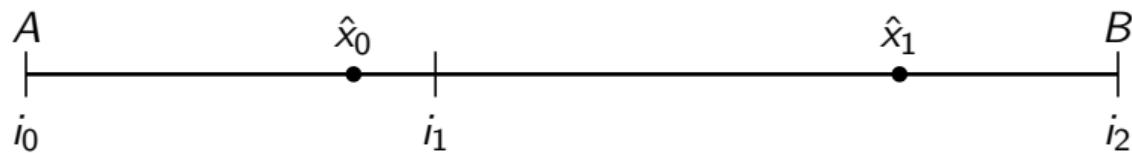
find global minimum wrt i_k, \hat{x}_k

Quantization MSE

$$\begin{aligned}\sigma_e^2 &= \int_{-\infty}^{\infty} (x - Q\{x\})^2 f_x(x) dx \\ &= \sum_{k=0}^{2^R-1} \int_{i_k}^{i_{k+1}} (x - \hat{x}_k)^2 f_x(x) dx\end{aligned}$$

find global minimum wrt i_k, \hat{x}_k

Simple example: optimal one-bit quantizer



3 free parameters: $i_1, \hat{x}_0, \hat{x}_1$

Simple example: optimal one-bit quantizer

$$\sigma_e^2 = \int_A^{i_1} (x - \hat{x}_0)^2 f_x(x) dx + \int_{i_1}^B (x - \hat{x}_1)^2 f_x(x) dx$$

find $i_1, \hat{x}_0, \hat{x}_1$ such that

$$\frac{\partial \sigma_e^2}{\partial i_1} = \frac{\partial \sigma_e^2}{\partial \hat{x}_0} = \frac{\partial \sigma_e^2}{\partial \hat{x}_1} = 0$$

Simple example: optimal one-bit quantizer

$$\sigma_e^2 = \int_A^{i_1} (x - \hat{x}_0)^2 f_x(x) dx + \int_{i_1}^B (x - \hat{x}_1)^2 f_x(x) dx$$

find $i_1, \hat{x}_0, \hat{x}_1$ such that

$$\frac{\partial \sigma_e^2}{\partial i_1} = \frac{\partial \sigma_e^2}{\partial \hat{x}_0} = \frac{\partial \sigma_e^2}{\partial \hat{x}_1} = 0$$

little calculus reminder

$$\frac{\partial}{\partial t} \int_{\alpha}^t f(\tau) d\tau = \frac{\partial}{\partial t} [F(t) - F(\alpha)] = f(t)$$

Optimal one-bit quantizer: threshold

$$\begin{aligned}\frac{\partial \sigma_e^2}{\partial i_1} &= \frac{\partial}{\partial i_1} \left[\int_A^{i_1} (x - \hat{x}_0)^2 f_x(x) dx + \int_{i_1}^B (x - \hat{x}_1)^2 f_x(x) dx \right] \\ &= (i_1 - \hat{x}_0)^2 f_x(i_1) - (i_1 - \hat{x}_1)^2 f_x(i_1) = 0\end{aligned}$$

$$\Rightarrow (i_1 - \hat{x}_0)^2 - (i_1 - \hat{x}_1)^2 = 0$$

$$\Rightarrow i_1 = \frac{\hat{x}_0 + \hat{x}_1}{2}$$

Optimal one-bit quantizer: threshold

$$\begin{aligned}\frac{\partial \sigma_e^2}{\partial i_1} &= \frac{\partial}{\partial i_1} \left[\int_A^{i_1} (x - \hat{x}_0)^2 f_x(x) dx + \int_{i_1}^B (x - \hat{x}_1)^2 f_x(x) dx \right] \\ &= (i_1 - \hat{x}_0)^2 f_x(i_1) - (i_1 - \hat{x}_1)^2 f_x(i_1) = 0\end{aligned}$$

$$\Rightarrow (i_1 - \hat{x}_0)^2 - (i_1 - \hat{x}_1)^2 = 0$$

$$\Rightarrow i_1 = \frac{\hat{x}_0 + \hat{x}_1}{2}$$

Optimal one-bit quantizer: threshold

$$\begin{aligned}\frac{\partial \sigma_e^2}{\partial i_1} &= \frac{\partial}{\partial i_1} \left[\int_A^{i_1} (x - \hat{x}_0)^2 f_x(x) dx + \int_{i_1}^B (x - \hat{x}_1)^2 f_x(x) dx \right] \\ &= (i_1 - \hat{x}_0)^2 f_x(i_1) - (i_1 - \hat{x}_1)^2 f_x(i_1) = 0\end{aligned}$$

$$\Rightarrow (i_1 - \hat{x}_0)^2 - (i_1 - \hat{x}_1)^2 = 0$$

$$\Rightarrow i_1 = \frac{\hat{x}_0 + \hat{x}_1}{2}$$

Optimal one-bit quantizer: threshold

$$\begin{aligned}\frac{\partial \sigma_e^2}{\partial i_1} &= \frac{\partial}{\partial i_1} \left[\int_A^{i_1} (x - \hat{x}_0)^2 f_x(x) dx + \int_{i_1}^B (x - \hat{x}_1)^2 f_x(x) dx \right] \\ &= (i_1 - \hat{x}_0)^2 f_x(i_1) - (i_1 - \hat{x}_1)^2 f_x(i_1) = 0\end{aligned}$$

$$\Rightarrow (i_1 - \hat{x}_0)^2 - (i_1 - \hat{x}_1)^2 = 0$$

$$\Rightarrow i_1 = \frac{\hat{x}_0 + \hat{x}_1}{2}$$

Optimal one-bit quantizer: values

$$\begin{aligned}\frac{\partial \sigma_e^2}{\partial \hat{x}_0} &= \frac{\partial}{\partial x_0} \int_A^{i_1} (x - \hat{x}_0)^2 f_x(x) dx \\ &= \int_A^{i_1} 2(\hat{x}_0 - x) f_x(x) dx = 0\end{aligned}$$

$$\Rightarrow \hat{x}_0 = \frac{\int_A^{i_1} x f_x(x) dx}{\int_A^{i_1} f_x(x) dx} \quad (\textit{center of mass})$$

$$\Rightarrow \hat{x}_1 = \frac{\int_{i_1}^B x f_x(x) dx}{\int_{i_1}^B f_x(x) dx}$$

Optimal one-bit quantizer: values

$$\begin{aligned}\frac{\partial \sigma_e^2}{\partial \hat{x}_0} &= \frac{\partial}{\partial x_0} \int_A^{i_1} (x - \hat{x}_0)^2 f_x(x) dx \\ &= \int_A^{i_1} 2(\hat{x}_0 - x) f_x(x) dx = 0\end{aligned}$$

$$\Rightarrow \hat{x}_0 = \frac{\int_A^{i_1} x f_x(x) dx}{\int_A^{i_1} f_x(x) dx} \quad (\text{center of mass})$$

$$\Rightarrow \hat{x}_1 = \frac{\int_{i_1}^B x f_x(x) dx}{\int_{i_1}^B f_x(x) dx}$$

Optimal one-bit quantizer: values

$$\frac{\partial \sigma_e^2}{\partial \hat{x}_0} = \frac{\partial}{\partial x_0} \int_A^{i_1} (x - \hat{x}_0)^2 f_x(x) dx$$

$$= \int_A^{i_1} 2(\hat{x}_0 - x) f_x(x) dx = 0$$

$$\Rightarrow \hat{x}_0 = \frac{\int_A^{i_1} x f_x(x) dx}{\int_A^{i_1} f_x(x) dx} \quad (\text{center of mass})$$

$$\Rightarrow \hat{x}_1 = \frac{\int_{i_1}^B x f_x(x) dx}{\int_{i_1}^B f_x(x) dx}$$

For uniformly-distributed input

$$f_x(x) = \frac{1}{B - A}$$

$$\hat{x}_0 = \frac{\int_A^{i_1} x \, dx}{\int_A^{i_1} dx} = \frac{A + i_1}{2}$$

$$\hat{x}_1 = \frac{\int_{i_1}^B x \, dx}{\int_{i_1}^B dx} = \frac{i_1 + B}{2}$$

$$i_1 = \frac{\hat{x}_0 + \hat{x}_1}{2} = \frac{A + B}{2}$$

For uniformly-distributed input

$$f_x(x) = \frac{1}{B - A}$$

$$\hat{x}_0 = \frac{\int_A^{i_1} x \, dx}{\int_A^{i_1} dx} = \frac{A + i_1}{2}$$

$$\hat{x}_1 = \frac{\int_{i_1}^B x \, dx}{\int_{i_1}^B dx} = \frac{i_1 + B}{2}$$

$$i_1 = \frac{\hat{x}_0 + \hat{x}_1}{2} = \frac{A + B}{2}$$

For uniformly-distributed input

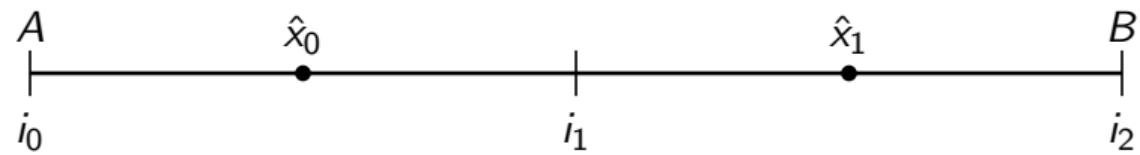
$$f_x(x) = \frac{1}{B - A}$$

$$\hat{x}_0 = \frac{\int_A^{i_1} x \, dx}{\int_A^{i_1} dx} = \frac{A + i_1}{2}$$

$$\hat{x}_1 = \frac{\int_{i_1}^B x \, dx}{\int_{i_1}^B dx} = \frac{i_1 + B}{2}$$

$$i_1 = \frac{\hat{x}_0 + \hat{x}_1}{2} = \frac{A + B}{2}$$

Optimal one-bit quantizer



Uniform quantization of uniform input

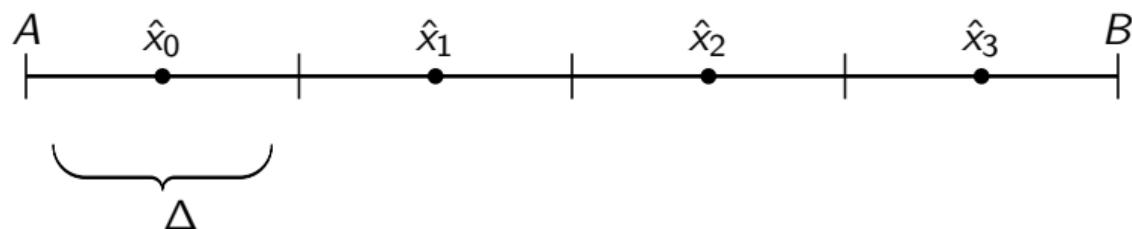
- ▶ simple but very general case
- ▶ optimal subdivision: 2^R equal intervals of width $\Delta = (B - A)2^{-R}$
- ▶ optimal quantization values: interval's midpoint

Uniform quantization of uniform input

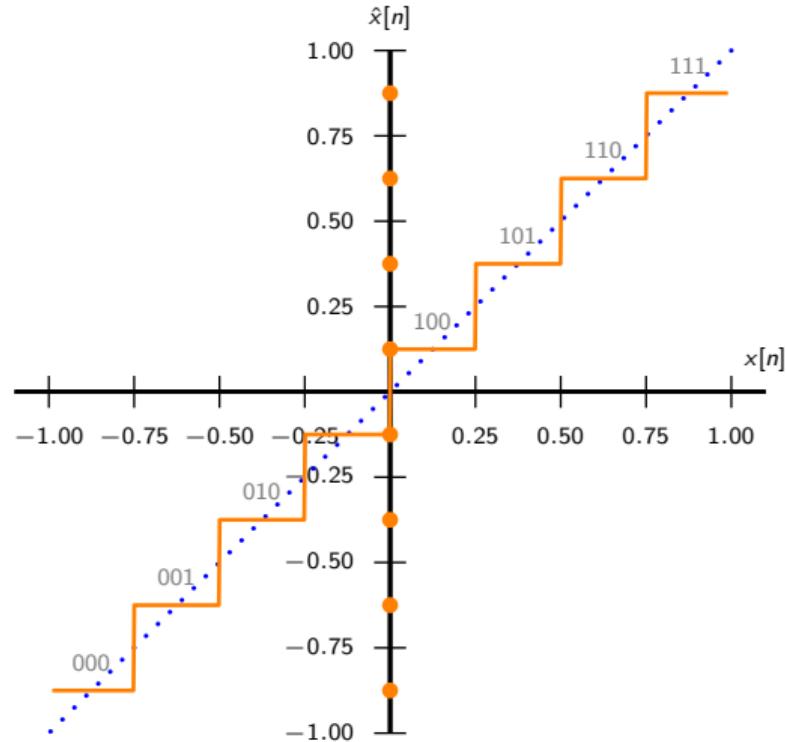
- ▶ simple but very general case
- ▶ optimal subdivision: 2^R *equal* intervals of width $\Delta = (B - A)2^{-R}$
- ▶ optimal quantization values: interval's midpoint

Uniform quantization of uniform input

- ▶ simple but very general case
- ▶ optimal subdivision: 2^R equal intervals of width $\Delta = (B - A)2^{-R}$
- ▶ optimal quantization values: interval's midpoint



Uniform 3-Bit quantization function



Uniform quantization of uniform input: error analysis

$$\begin{aligned}\sigma_e^2 &= \int_A^B f_x(x)(\mathcal{Q}\{x\} - x)^2 dx \\ &= \sum_{k=0}^{2^R-1} \int_{I_k} f_x(x)(\hat{x}_k - x)^2 dx\end{aligned}$$

$$f_x(s) = \frac{1}{B-A}$$

$$\Delta = \frac{B-A}{2^R}$$

$$I_k = [A + k\Delta, A + (k+1)\Delta]$$

$$\hat{x}_k = A + k\Delta + \Delta/2$$

Uniform quantization of uniform input: error analysis

$$\begin{aligned}\sigma_e^2 &= \int_A^B f_x(x)(\mathcal{Q}\{x\} - x)^2 dx \\ &= \sum_{k=0}^{2^R-1} \int_{I_k} f_x(x)(\hat{x}_k - x)^2 dx\end{aligned}$$

$$f_x(s) = \frac{1}{B-A}$$

$$\Delta = \frac{B-A}{2^R}$$

$$I_k = [A + k\Delta, A + (k+1)\Delta]$$

$$\hat{x}_k = A + k\Delta + \Delta/2$$

Uniform quantization of uniform input: error analysis

$$\begin{aligned}\sigma_e^2 &= \int_A^B f_x(x)(\mathcal{Q}\{x\} - x)^2 dx \\ &= \sum_{k=0}^{2^R-1} \int_{I_k} f_x(x)(\hat{x}_k - x)^2 dx\end{aligned}$$

$$f_x(s) = \frac{1}{B-A}$$

$$\Delta = \frac{B-A}{2^R}$$

$$I_k = [A + k\Delta, A + (k+1)\Delta]$$

$$\hat{x}_k = A + k\Delta + \Delta/2$$

Uniform quantization of uniform input: error analysis

$$\begin{aligned}\sigma_e^2 &= \sum_{k=0}^{2^R-1} \int_{A+k\Delta}^{A+k\Delta+\Delta} \frac{(A + k\Delta + \Delta/2 - x)^2}{B - A} dx \\ &= 2^R \int_0^\Delta \frac{(\Delta/2 - x)^2}{B - A} dx \\ &= \frac{\Delta^2}{12}\end{aligned}$$

Uniform quantization of uniform input: error analysis

$$\begin{aligned}\sigma_e^2 &= \sum_{k=0}^{2^R-1} \int_{A+k\Delta}^{A+k\Delta+\Delta} \frac{(A + k\Delta + \Delta/2 - x)^2}{B - A} dx \\ &= 2^R \int_0^\Delta \frac{(\Delta/2 - x)^2}{B - A} dx \\ &= \frac{\Delta^2}{12}\end{aligned}$$

Uniform quantization of uniform input: error analysis

$$\begin{aligned}\sigma_e^2 &= \sum_{k=0}^{2^R-1} \int_{A+k\Delta}^{A+k\Delta+\Delta} \frac{(A + k\Delta + \Delta/2 - x)^2}{B - A} dx \\ &= 2^R \int_0^\Delta \frac{(\Delta/2 - x)^2}{B - A} dx \\ &= \frac{\Delta^2}{12}\end{aligned}$$

Error analysis of the quantization error

fundamental assumptions:

- ▶ signal and quantization error are uncorrelated (ok-ish)
- ▶ quantization error process is white (stretch)

quantization noise acts as additive white noise

Error analysis

- ▶ error energy

$$\sigma_e^2 = \Delta^2/12, \quad \Delta = (B - A)/2^R$$

- ▶ signal energy

$$\sigma_x^2 = (B - A)^2/12$$

- ▶ signal to noise ratio

$$\text{SNR} = 2^{2R}$$

- ▶ in dB

$$\text{SNR}_{\text{dB}} = 10 \log_{10} 2^{2R} \approx 6R \text{ dB}$$

Error analysis

- ▶ error energy

$$\sigma_e^2 = \Delta^2/12, \quad \Delta = (B - A)/2^R$$

- ▶ signal energy

$$\sigma_x^2 = (B - A)^2/12$$

- ▶ signal to noise ratio

$$\text{SNR} = 2^{2R}$$

- ▶ in dB

$$\text{SNR}_{\text{dB}} = 10 \log_{10} 2^{2R} \approx 6R \text{ dB}$$

Error analysis

- ▶ error energy

$$\sigma_e^2 = \Delta^2/12, \quad \Delta = (B - A)/2^R$$

- ▶ signal energy

$$\sigma_x^2 = (B - A)^2/12$$

- ▶ signal to noise ratio

$$\text{SNR} = 2^{2R}$$

- ▶ in dB

$$\text{SNR}_{\text{dB}} = 10 \log_{10} 2^{2R} \approx 6R \text{ dB}$$

Error analysis

- ▶ error energy

$$\sigma_e^2 = \Delta^2/12, \quad \Delta = (B - A)/2^R$$

- ▶ signal energy

$$\sigma_x^2 = (B - A)^2/12$$

- ▶ signal to noise ratio

$$\text{SNR} = 2^{2R}$$

- ▶ in dB

$$\text{SNR}_{\text{dB}} = 10 \log_{10} 2^{2R} \approx 6R \text{ dB}$$

The “6dB/bit” rule of thumb

- ▶ a compact disk has 16 bits/sample:

$$\text{max SNR} = 96\text{dB}$$

- ▶ a DVD has 24 bits/sample:

$$\text{max SNR} = 144\text{dB}$$

The “6dB/bit” rule of thumb

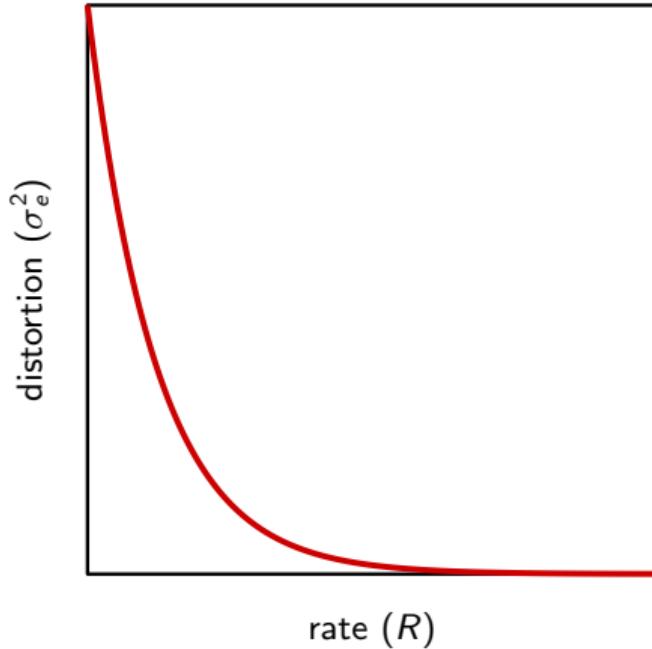
- ▶ a compact disk has 16 bits/sample:

$$\text{max SNR} = 96\text{dB}$$

- ▶ a DVD has 24 bits/sample:

$$\text{max SNR} = 144\text{dB}$$

Rate/Distortion Curve



Other quantization errors

If input is not bounded to $[A, B]$:

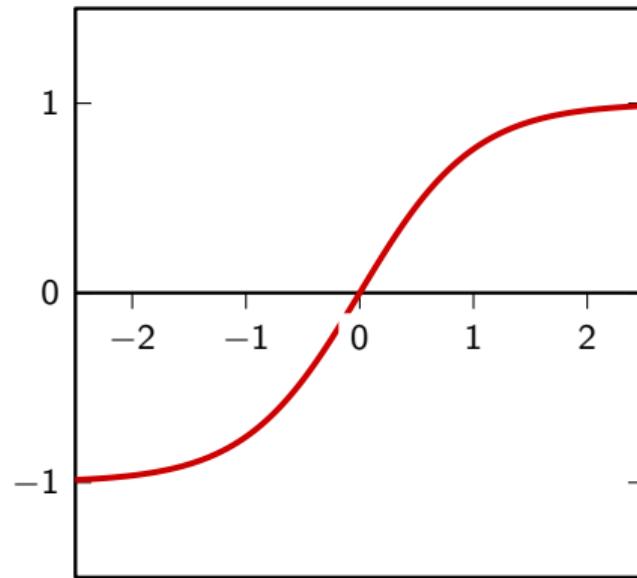
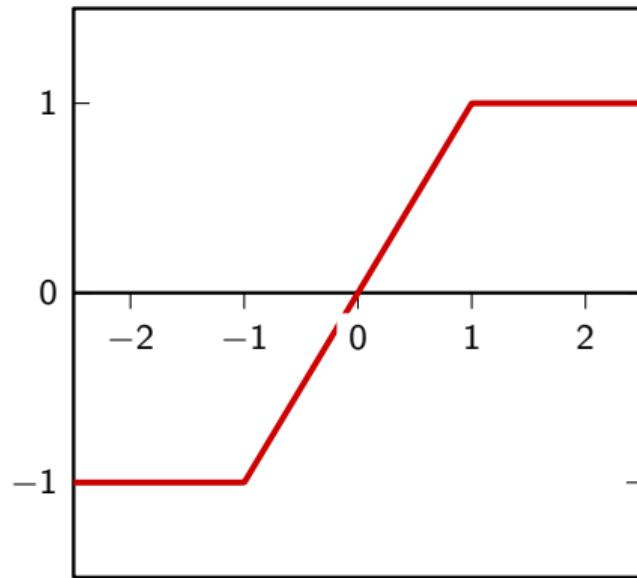
- ▶ clip samples to $[A, B]$: linear distortion (can be put to good use in guitar effects!)
- ▶ smoothly saturate input: this simulates the saturation curves of analog electronics

Other quantization errors

If input is not bounded to $[A, B]$:

- ▶ clip samples to $[A, B]$: linear distortion (can be put to good use in guitar effects!)
- ▶ smoothly saturate input: this simulates the saturation curves of analog electronics

Clipping vs saturation



Other quantization errors

If input is not uniform:

- ▶ use uniform quantizer and accept increased error.

For instance, if input is Gaussian:

$$\sigma_e^2 = \frac{\sqrt{3}\pi}{2} \sigma^2 \Delta^2$$

- ▶ use “companders”
- ▶ design optimal quantizer for input distribution, if known (Lloyd-Max algorithm)

Other quantization errors

If input is not uniform:

- ▶ use uniform quantizer and accept increased error.

For instance, if input is Gaussian:

$$\sigma_e^2 = \frac{\sqrt{3}\pi}{2} \sigma^2 \Delta^2$$

- ▶ use “companders”
- ▶ design optimal quantizer for input distribution, if known (Lloyd-Max algorithm)

Other quantization errors

If input is not uniform:

- ▶ use uniform quantizer and accept increased error.

For instance, if input is Gaussian:

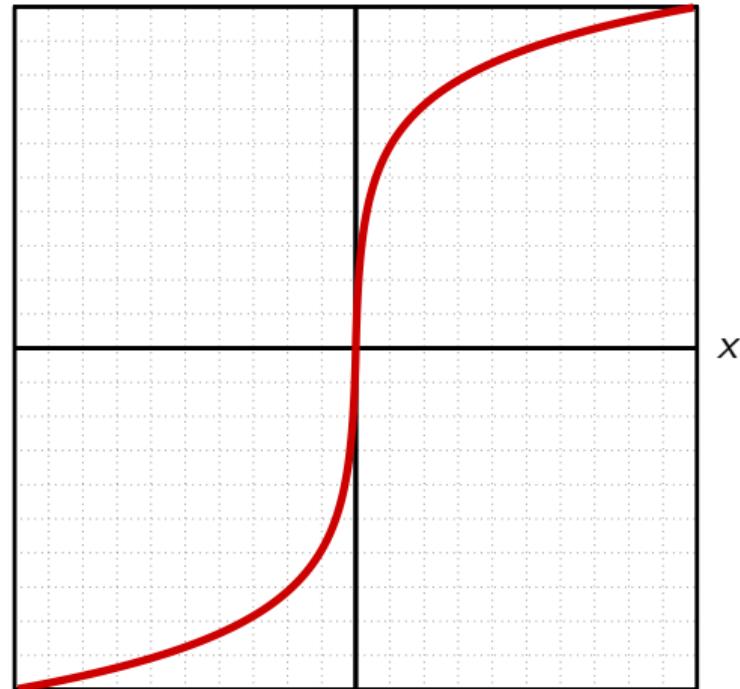
$$\sigma_e^2 = \frac{\sqrt{3}\pi}{2} \sigma^2 \Delta^2$$

- ▶ use “companders”
- ▶ design optimal quantizer for input distribution, if known (Lloyd-Max algorithm)

μ -law compander

$$\mathcal{C}\{x[n]\} = \text{sgn}(x[n]) \frac{\ln(1 + \mu|x[n]|)}{\ln(1 + \mu)}$$

$\mathcal{C}\{x\}$



Lloyd-Max Quantizer design

$$\sigma_e^2 = \sum_{k=0}^{2^R-1} \int_{i_k}^{i_{k+1}} (x - \hat{x}_k)^2 f_x(x) dx$$

A) $\frac{\partial \sigma_e^2}{\partial \hat{x}_k} = 0 \Rightarrow \hat{x}_k = \frac{\int_{i_{k-1}}^{i_k} x f_x(x) dx}{\int_{i_{k-1}}^{i_k} f_x(x) dx}$

B) $\frac{\partial \sigma_e^2}{\partial i_k} = 0 \Rightarrow i_k = \frac{\hat{x}_{k-1} + \hat{x}_k}{2}$

Lloyd-Max Quantizer design

- ▶ start with a guess for the i_k
- ▶ solve A and B iteratively until convergence

A/D and D/A converters

Overview:

- ▶ Analog-to-digital (A/D) conversion
- ▶ Digital-to-analog (D/A) conversion

Overview:

- ▶ Analog-to-digital (A/D) conversion
- ▶ Digital-to-analog (D/A) conversion

From analog to digital

- ▶ sampling discretizes time
- ▶ quantization discretized amplitude
- ▶ how is it done in practice?

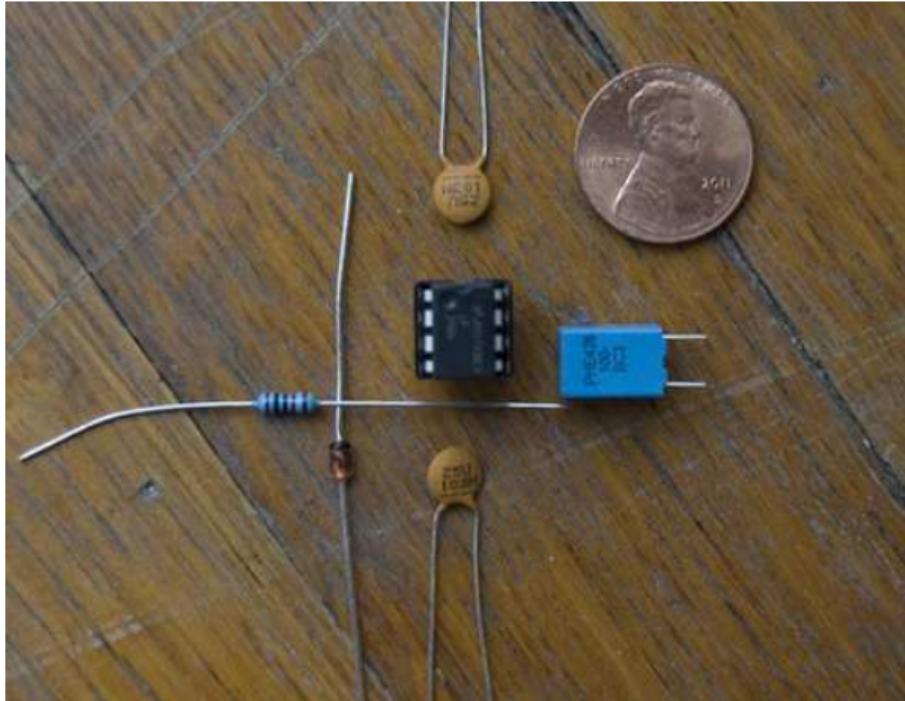
From analog to digital

- ▶ sampling discretizes time
- ▶ quantization discretized amplitude
- ▶ how is it done in practice?

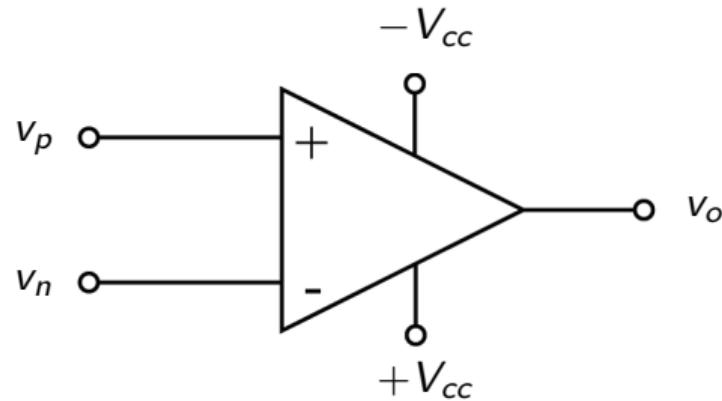
From analog to digital

- ▶ sampling discretizes time
- ▶ quantization discretized amplitude
- ▶ how is it done in practice?

From analog to digital

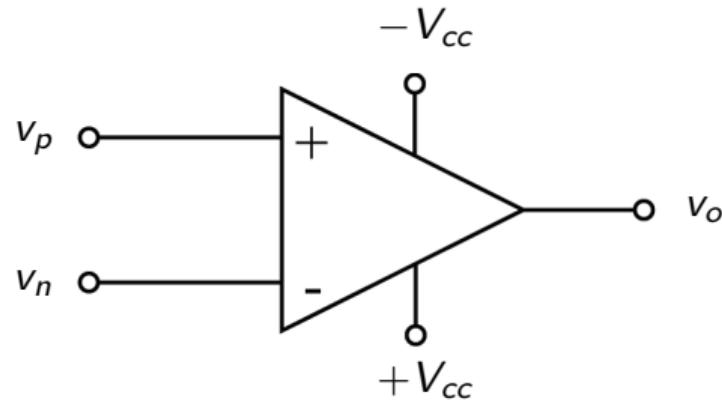


A tiny bit of electronics: the op-amp



$$v_o = G(v_p - v_n)$$

A tiny bit of electronics: the op-amp



$$v_o = G(v_p - v_n)$$

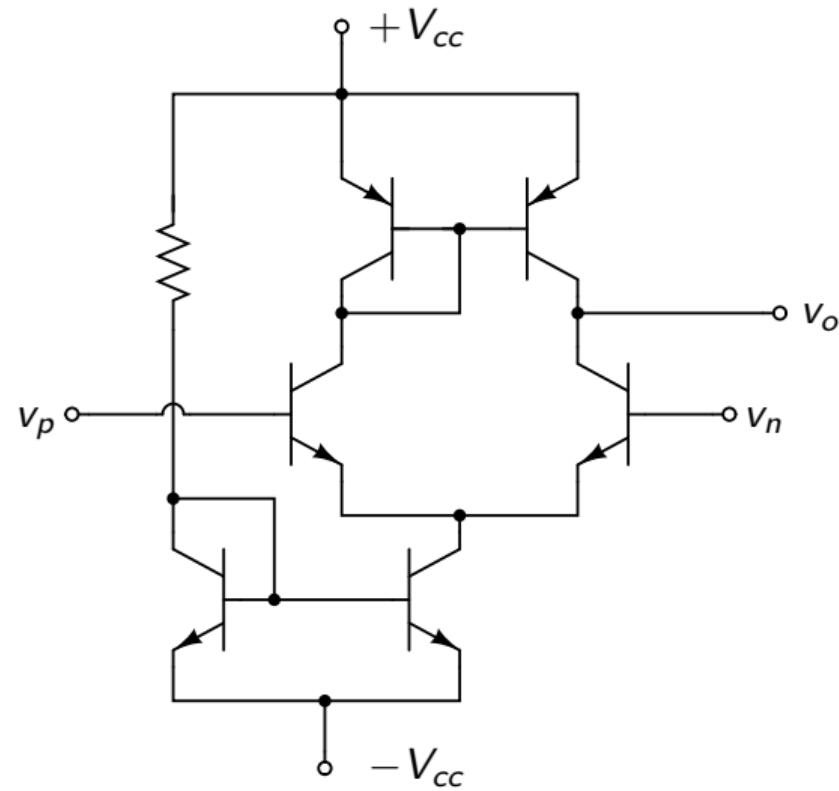
The two key properties

- ▶ infinite input gain ($G \approx \infty$)
- ▶ zero input current

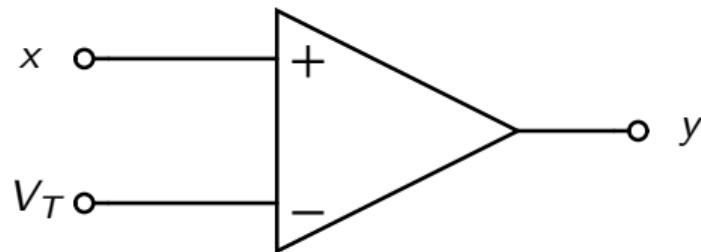
The two key properties

- ▶ infinite input gain ($G \approx \infty$)
- ▶ zero input current

Inside the box

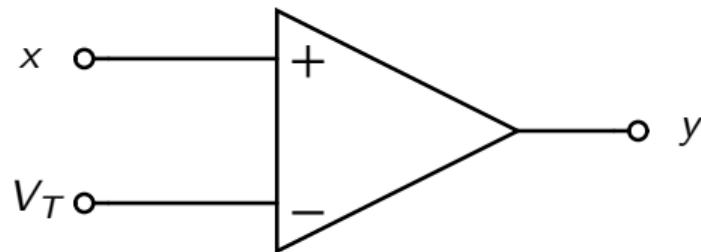


The op-amp in open loop: comparator



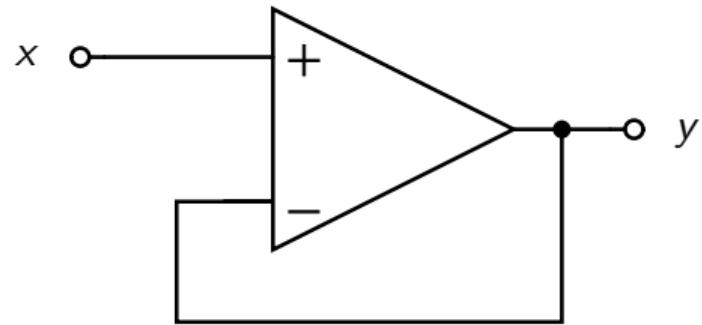
$$y = \begin{cases} +V_{cc} & \text{if } x > V_T \\ -V_{cc} & \text{if } x < V_T \end{cases}$$

The op-amp in open loop: comparator



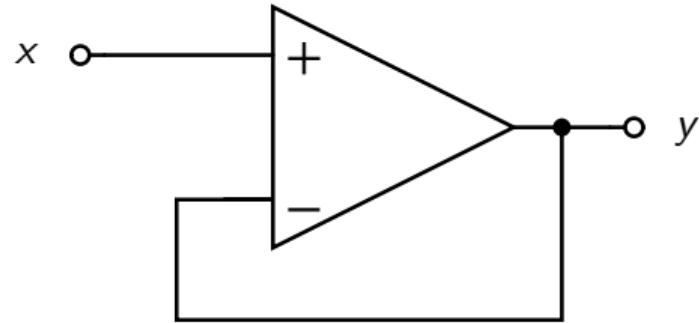
$$y = \begin{cases} +V_{cc} & \text{if } x > V_T \\ -V_{cc} & \text{if } x < V_T \end{cases}$$

The op-amp in closed loop: buffer



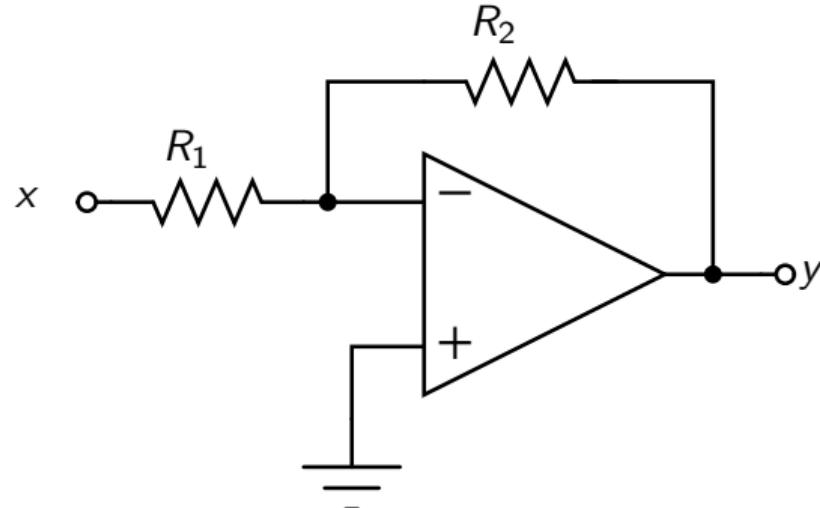
$$y = x$$

The op-amp in closed loop: buffer

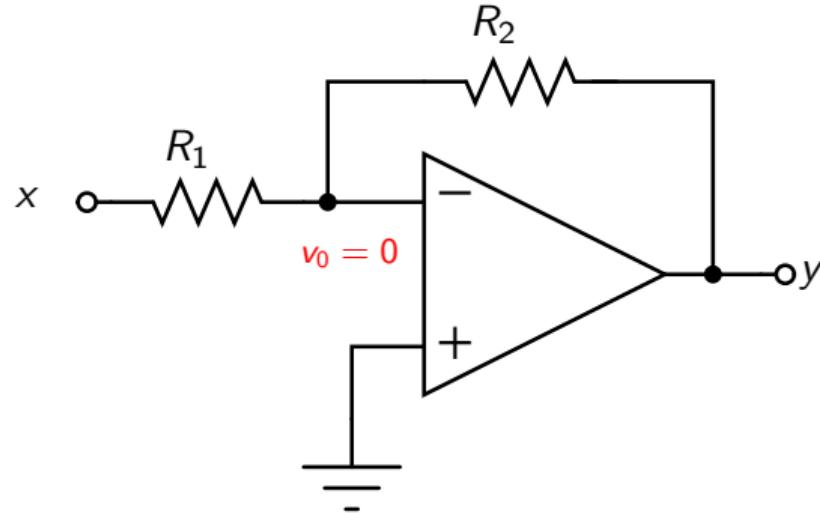


$$y = x$$

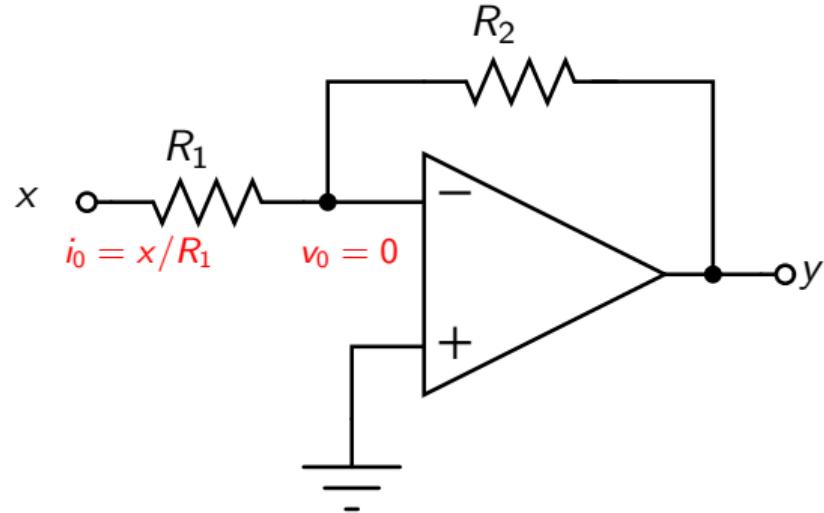
The op-amp in closed loop: inverting amplifier



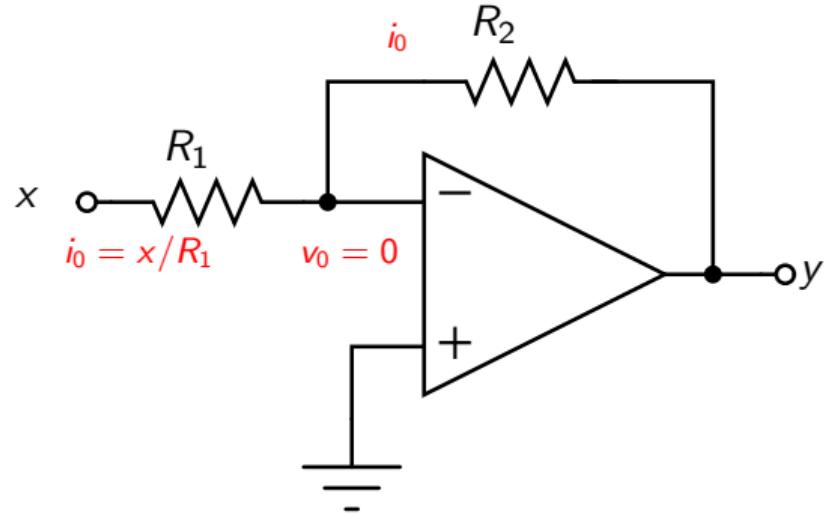
The op-amp in closed loop: inverting amplifier



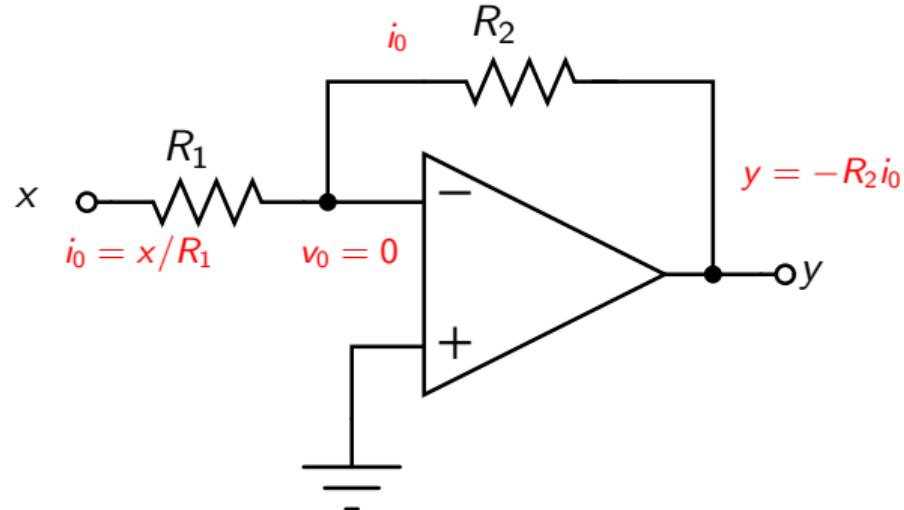
The op-amp in closed loop: inverting amplifier



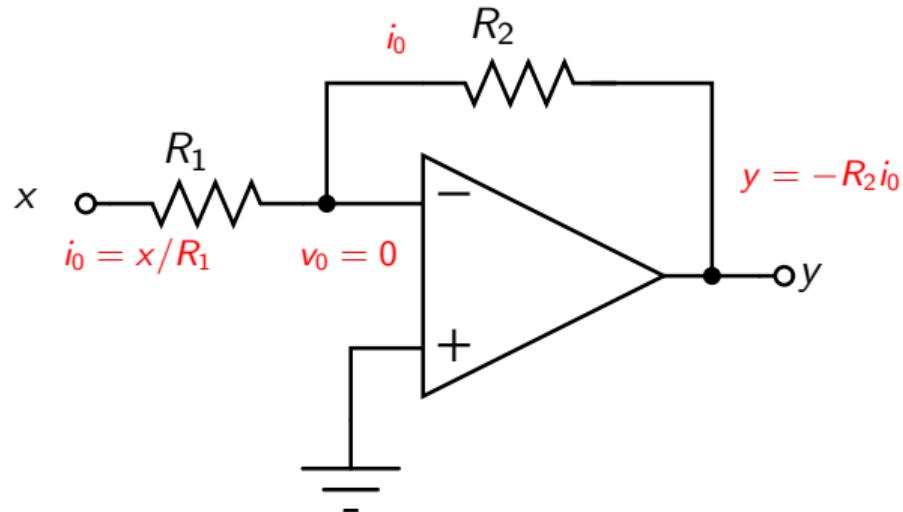
The op-amp in closed loop: inverting amplifier



The op-amp in closed loop: inverting amplifier



The op-amp in closed loop: inverting amplifier

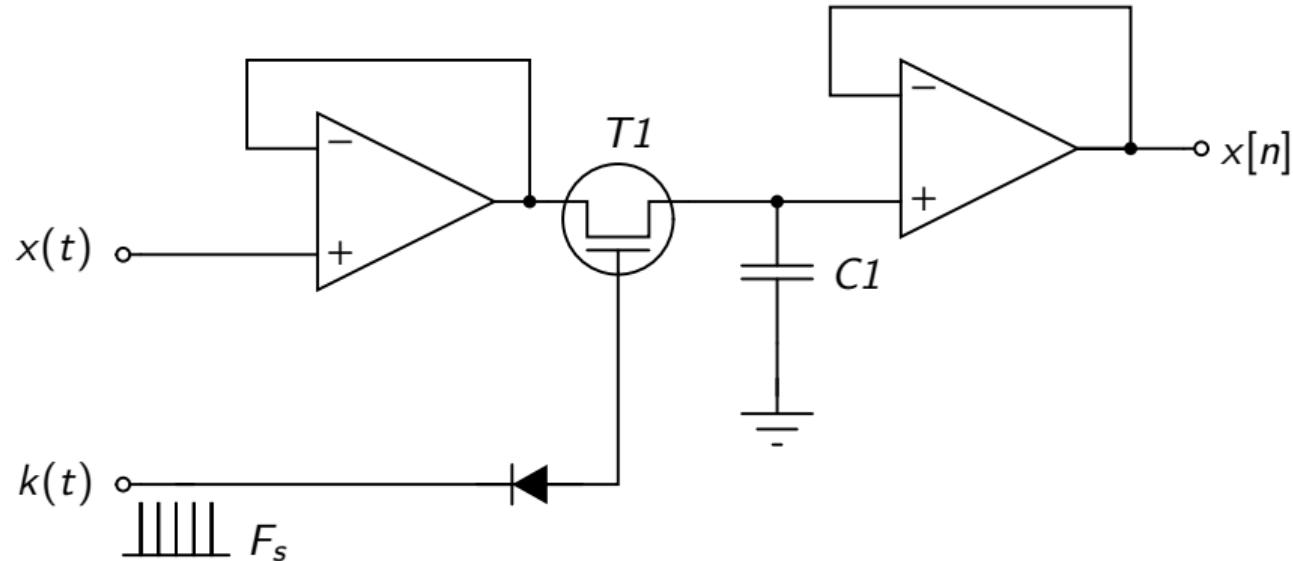


$$y = -(R_2/R_1)x$$

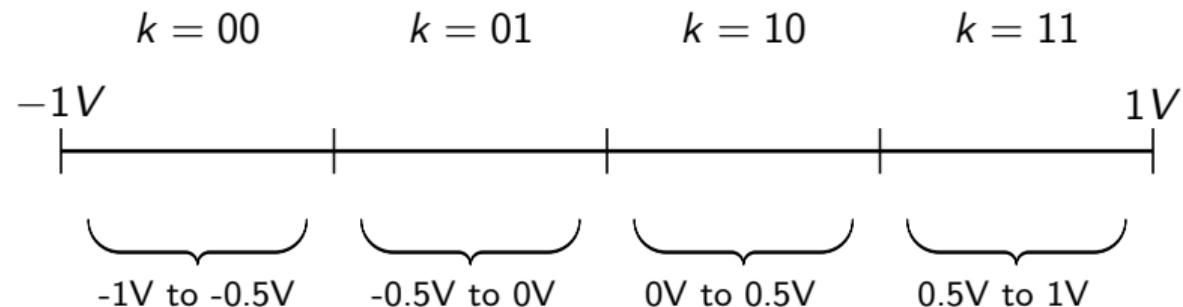
A/D Converter: Sampling



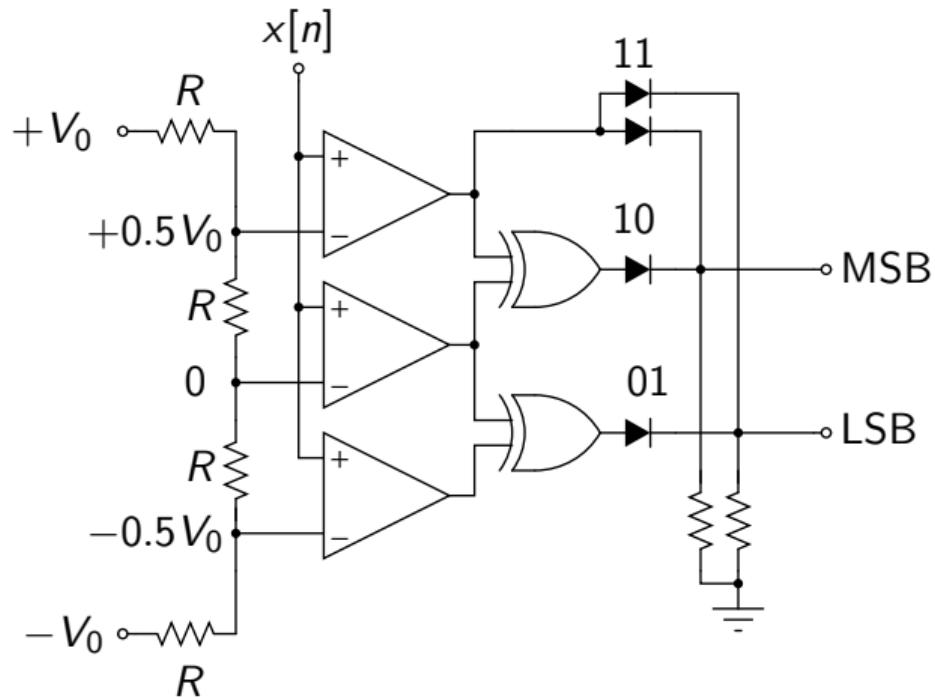
A/D Converter: Sample & Hold



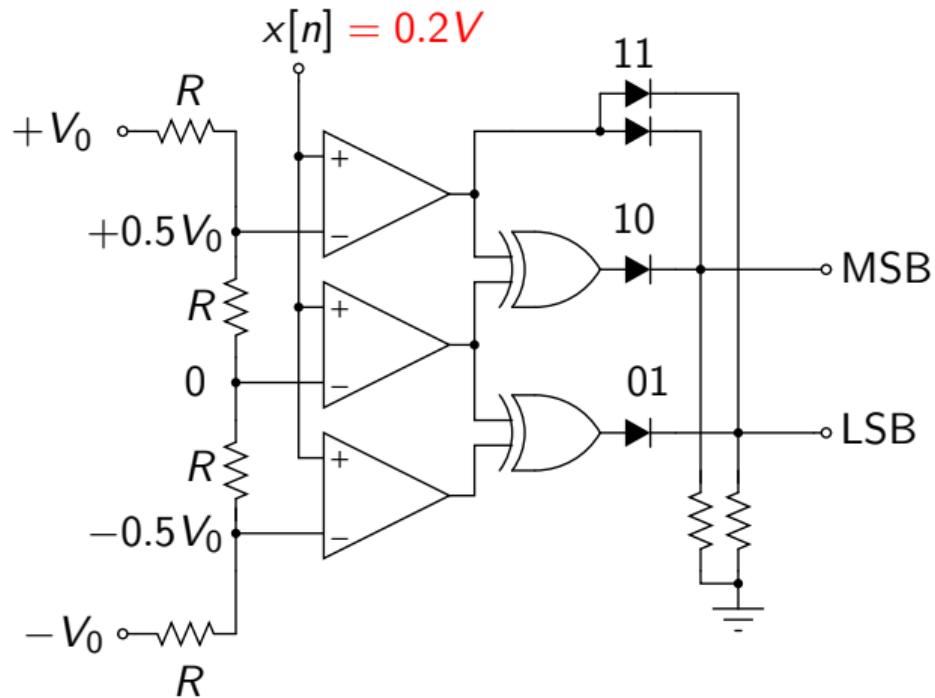
A/D Converter: Quantization



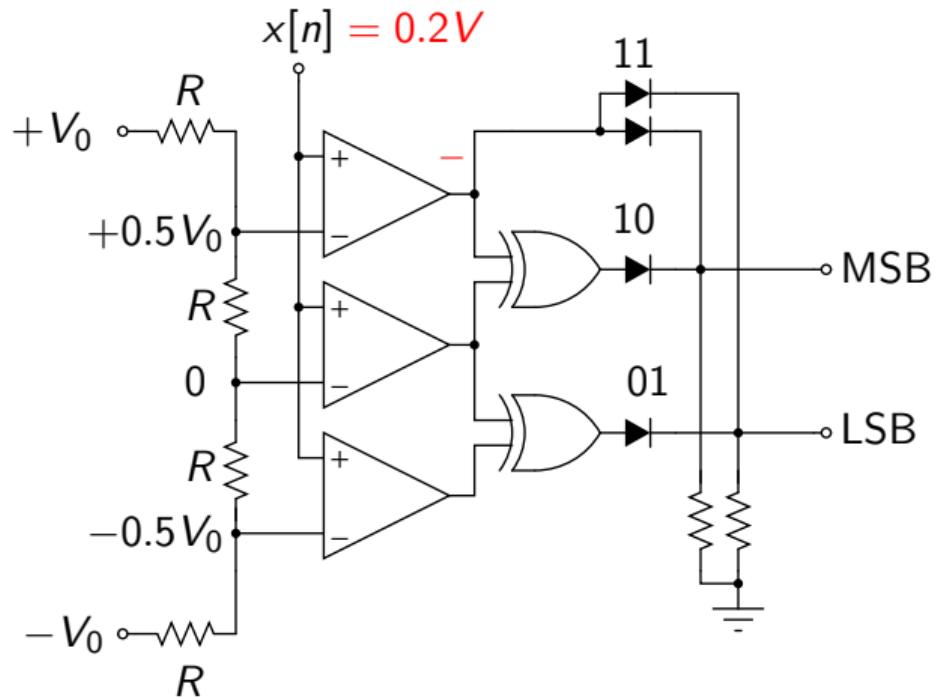
A/D Converter: 2-Bit Quantizer



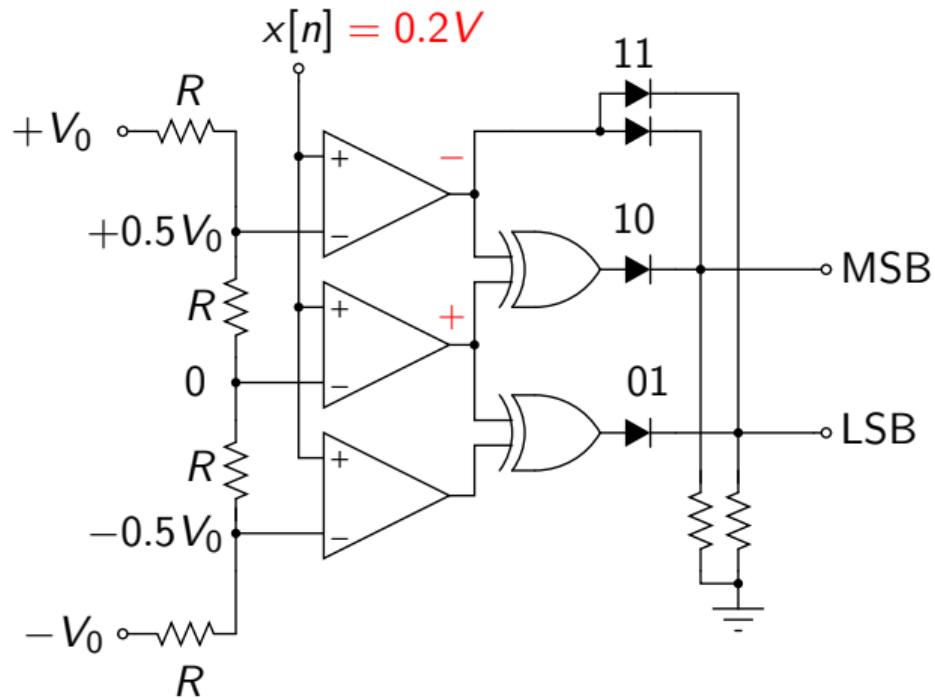
A/D Converter: 2-Bit Quantizer



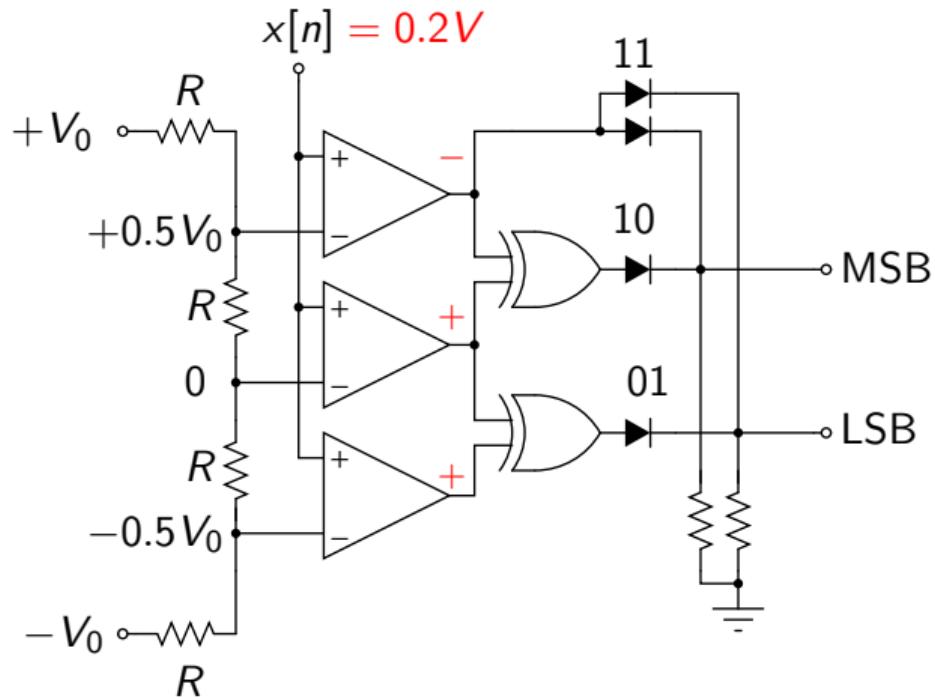
A/D Converter: 2-Bit Quantizer



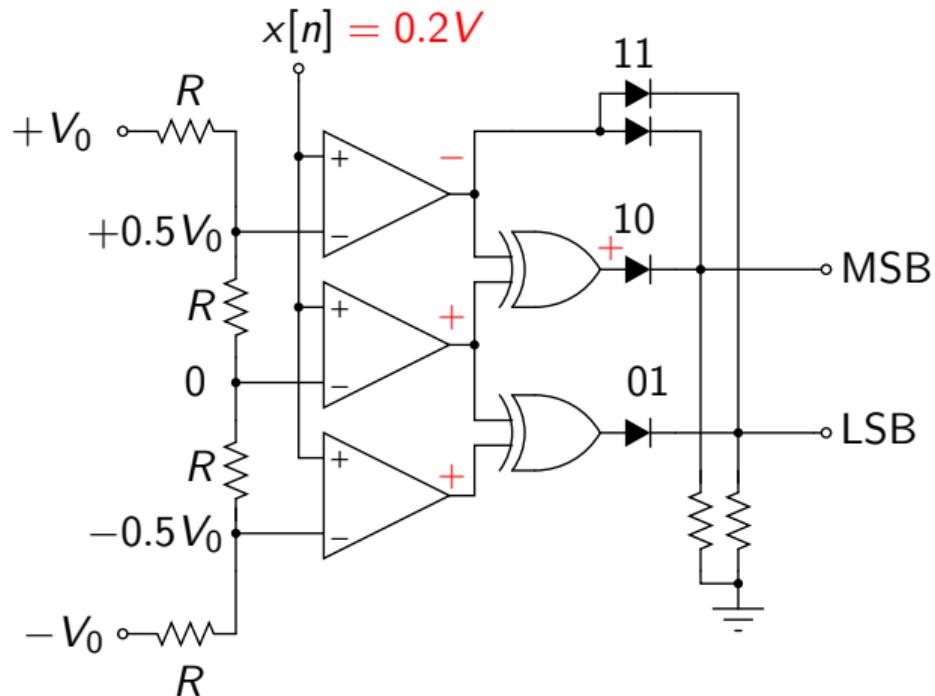
A/D Converter: 2-Bit Quantizer



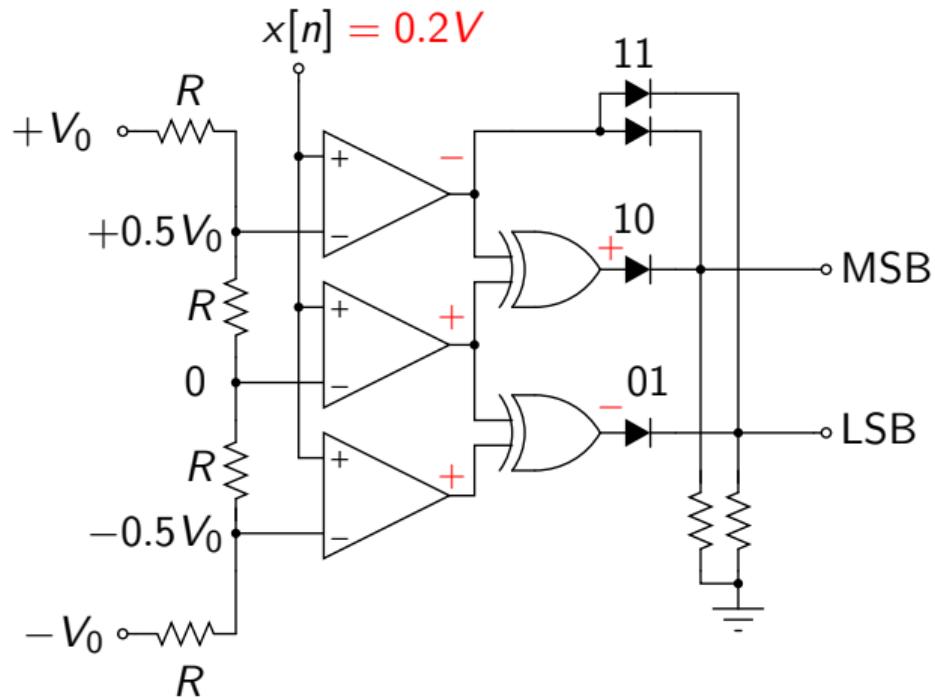
A/D Converter: 2-Bit Quantizer



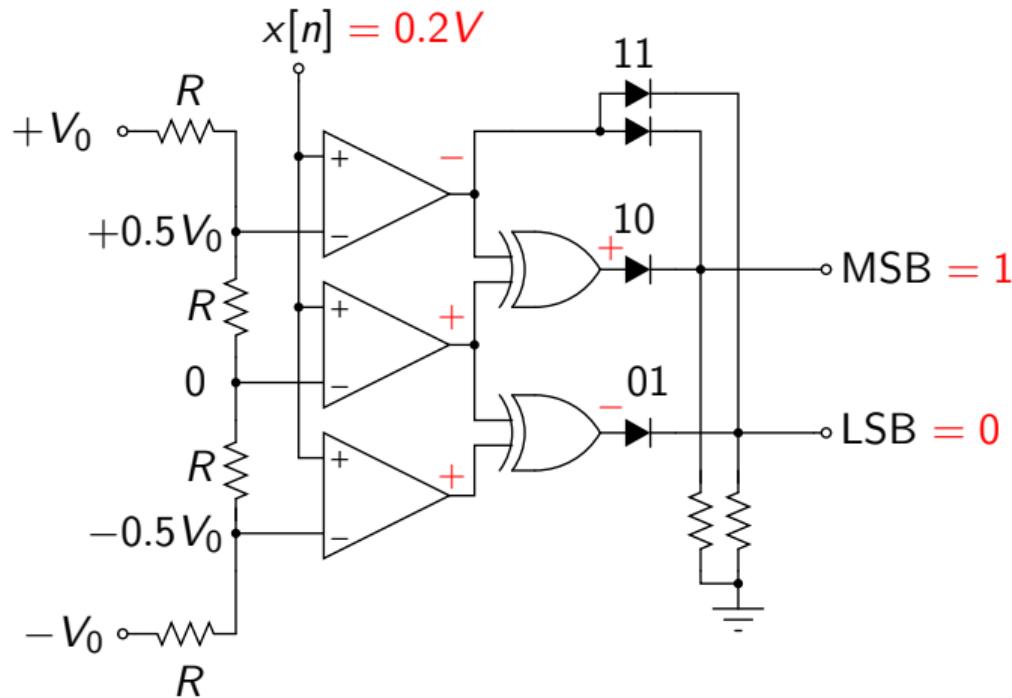
A/D Converter: 2-Bit Quantizer



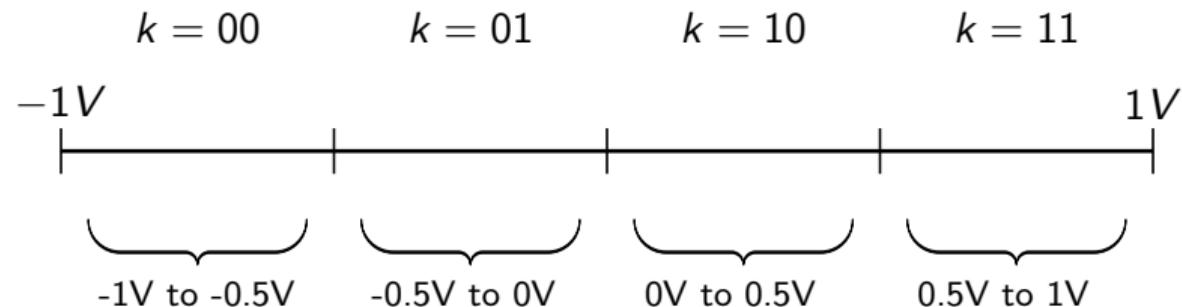
A/D Converter: 2-Bit Quantizer



A/D Converter: 2-Bit Quantizer



A/D Converter: Quantization

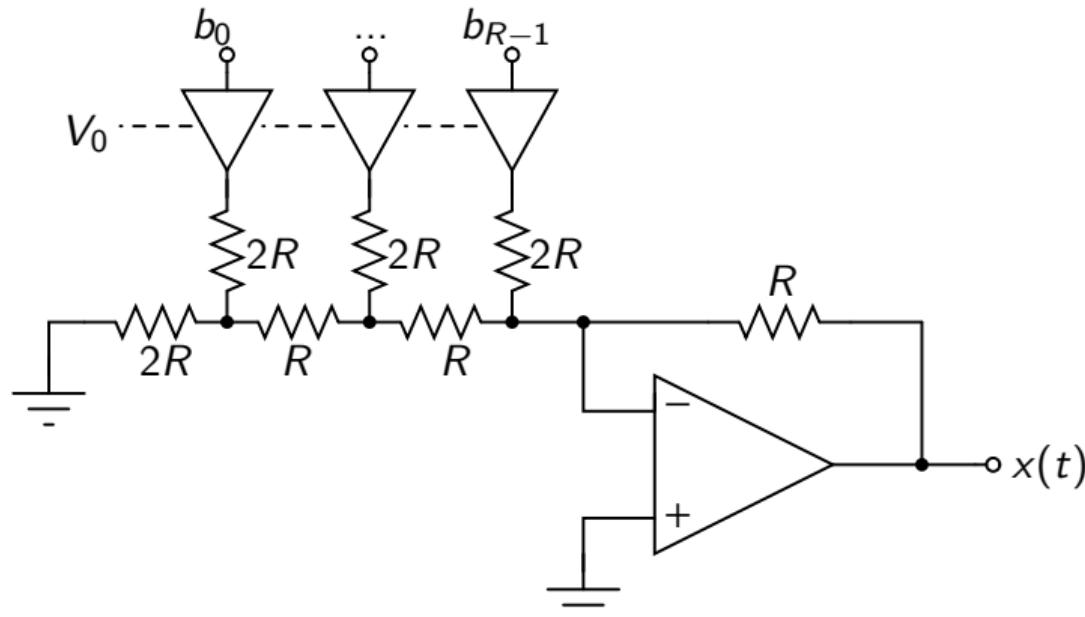


D/A Converter

$$x_B[n] = b_{R-1}b_{R-2}\dots b_1b_0$$

$$\hat{x}[n] = \sum_{k=0}^{R-1} \frac{V_0}{2^k} b_k$$

D/A Converter



oversampling

Oversampling

► oversampled A/D

- reduce quantization error

► oversampled D/A

- use cheaper hardware for interpolation

Oversampling

- ▶ oversampled A/D

- reduce quantization error

- ▶ oversampled D/A

- use cheaper hardware for interpolation

Oversampling

- ▶ oversampled A/D

- reduce quantization error

- ▶ oversampled D/A

- use cheaper hardware for interpolation

Oversampling

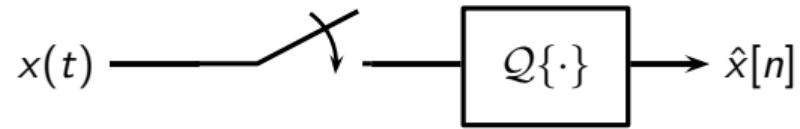
- ▶ oversampled A/D

- reduce quantization error

- ▶ oversampled D/A

- use cheaper hardware for interpolation

Oversampled A/D



$$T_s = 1/F_s$$

$$\hat{x}[n] = x[n] + e[n]$$

Oversampled A/D

Key assumptions:

$e[n]$ i.i.d. process, **independent of $x[n]$**

$$P_e(e^{j\omega}) = \frac{\Delta^2}{12} \quad \text{over } [-\pi, \pi] \text{ (no aliasing)}$$

Key observation:

$$X(e^{j\omega}) = F_s X\left(\frac{\omega}{2\pi} F_s\right)$$

Oversampled A/D

Key assumptions:

$e[n]$ i.i.d. process, **independent of $x[n]$**

$$P_e(e^{j\omega}) = \frac{\Delta^2}{12} \quad \text{over } [-\pi, \pi] \text{ (no aliasing)}$$

Key observation:

$$X(e^{j\omega}) = F_s X\left(\frac{\omega}{2\pi} F_s\right)$$

Oversampled A/D

Key assumptions:

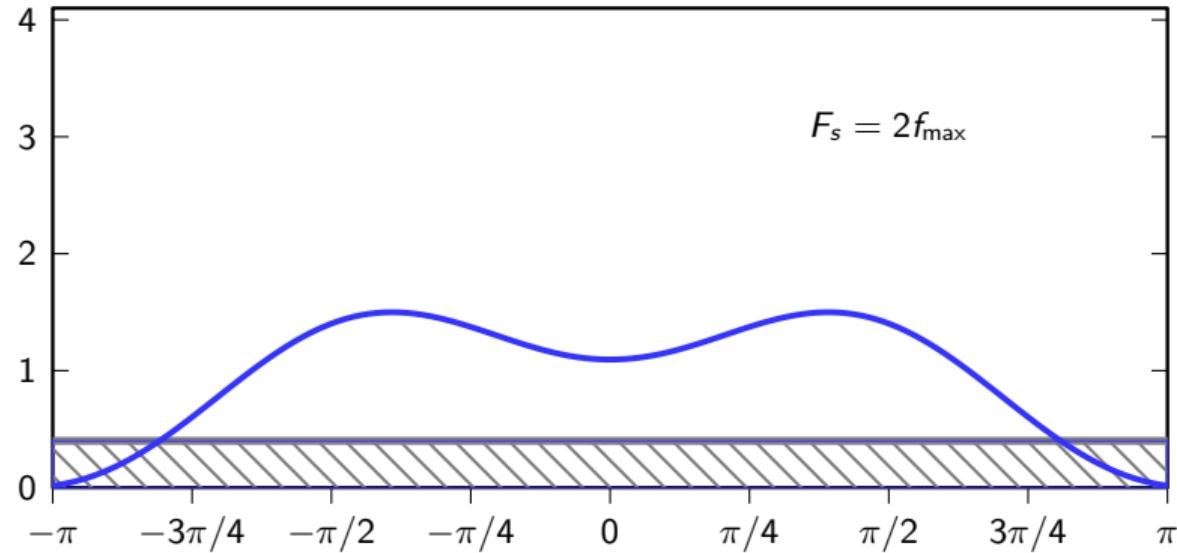
$e[n]$ i.i.d. process, **independent of $x[n]$**

$$P_e(e^{j\omega}) = \frac{\Delta^2}{12} \quad \text{over } [-\pi, \pi] \text{ (no aliasing)}$$

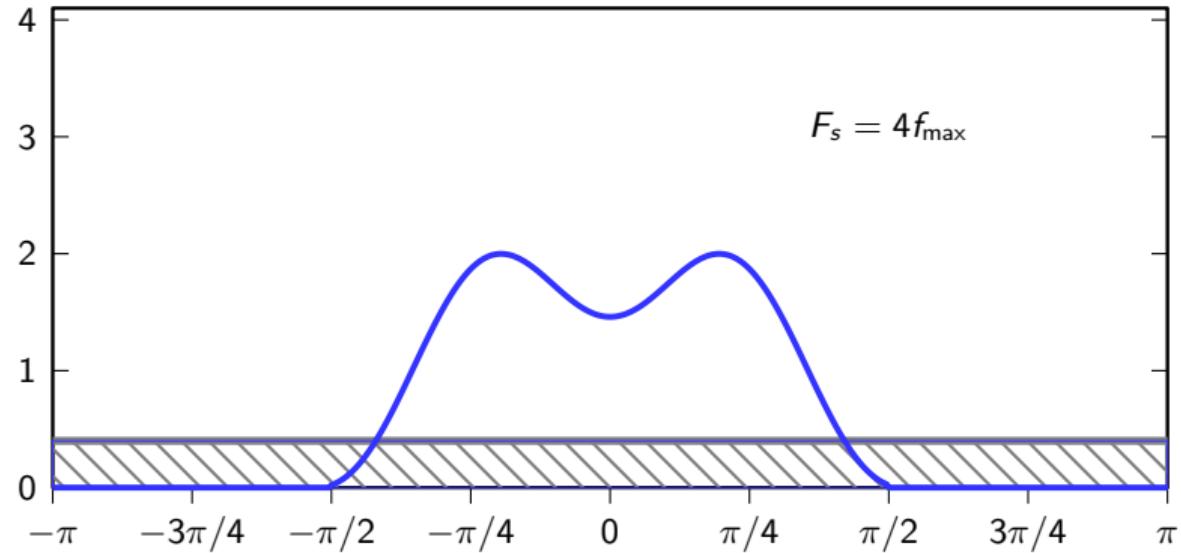
Key observation:

$$X(e^{j\omega}) = F_s X\left(\frac{\omega}{2\pi} F_s\right)$$

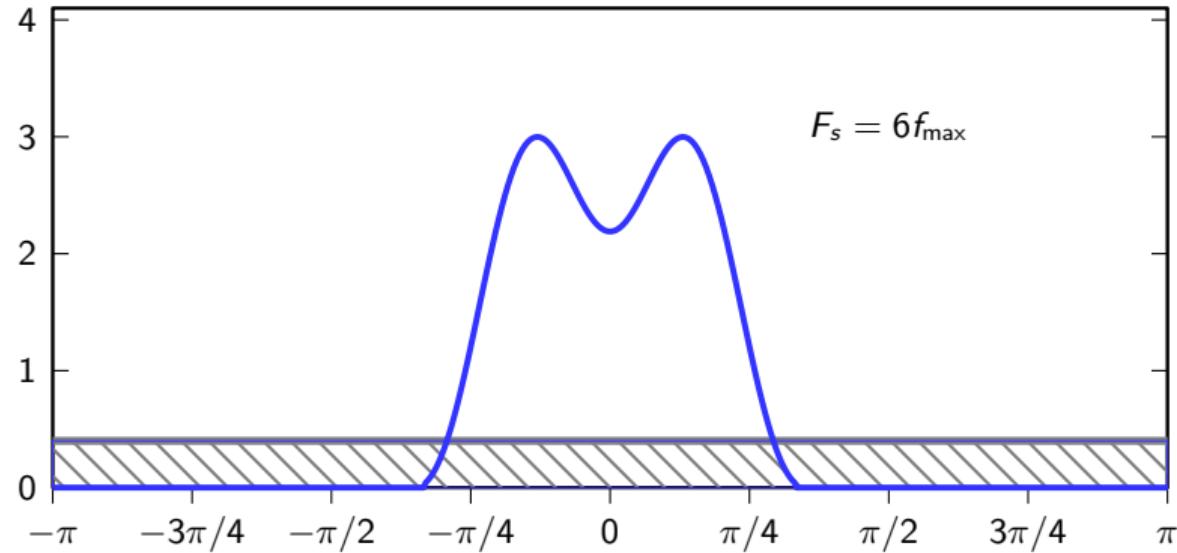
Oversampled A/D



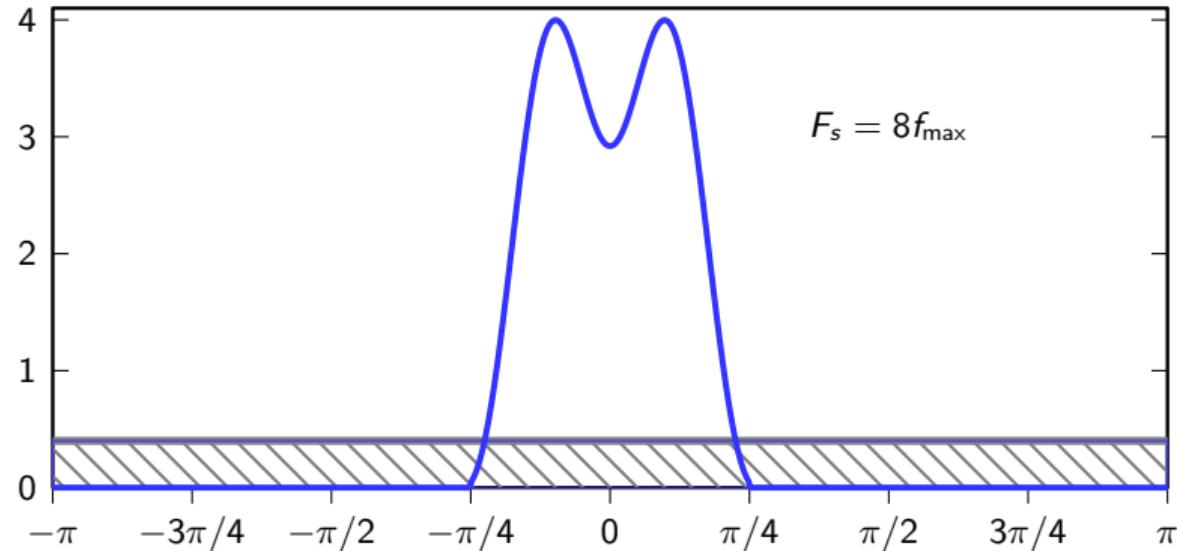
Oversampled A/D



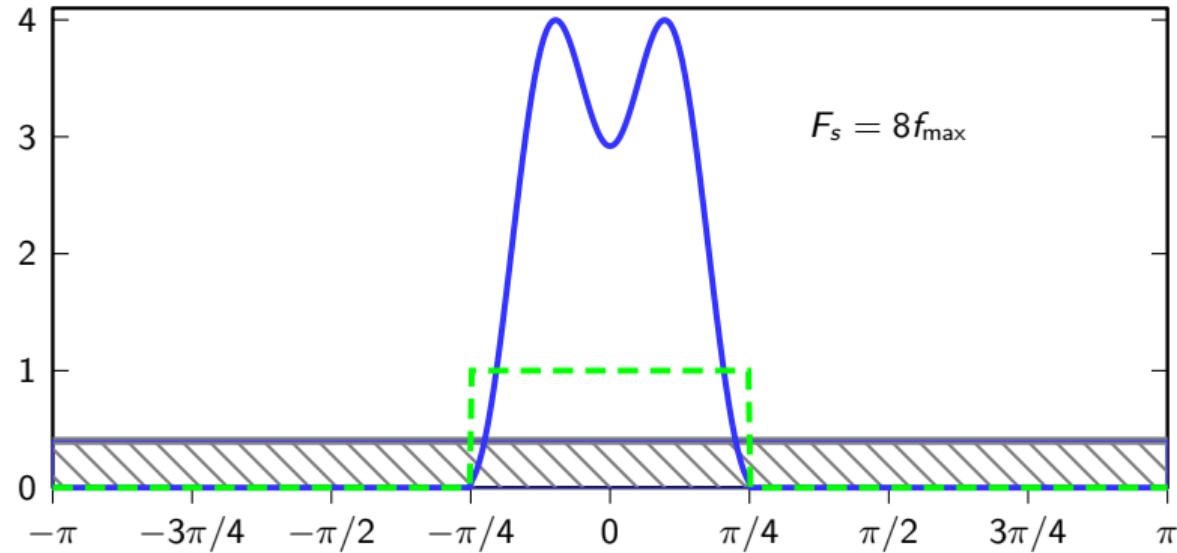
Oversampled A/D



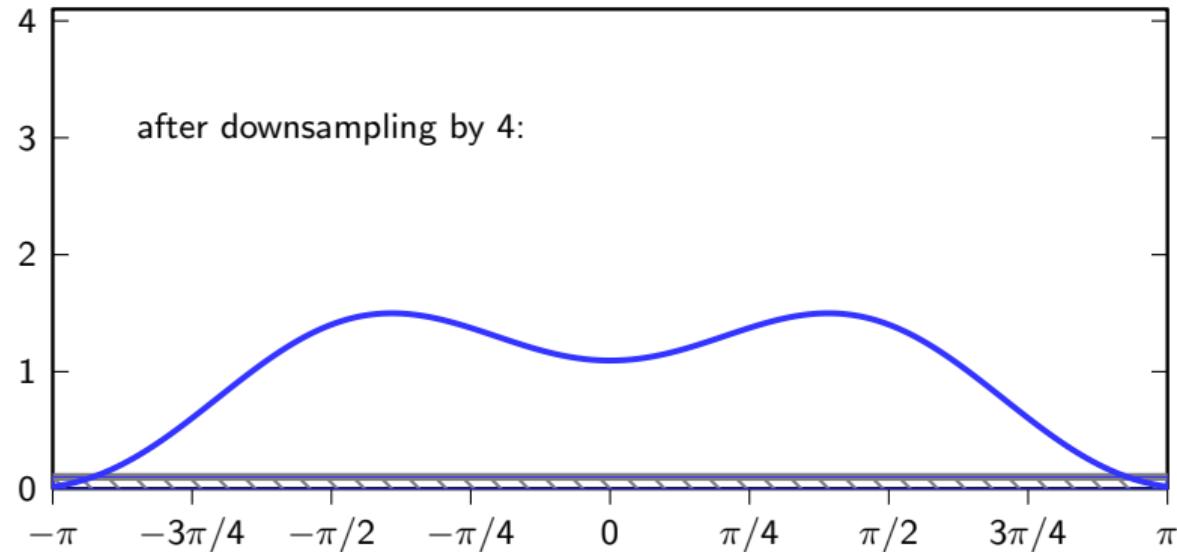
Oversampled A/D



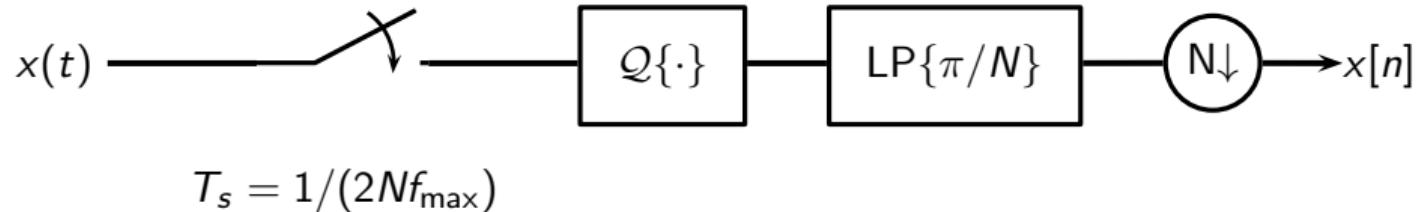
Oversampled A/D



Oversampled A/D



Oversampled A/D



- ▶ $\text{SNR}_O \approx N \text{SNR}$
- ▶ 3dB per octave (doubling of F_s)
- ▶ but key assumption (independence) breaks down fast...

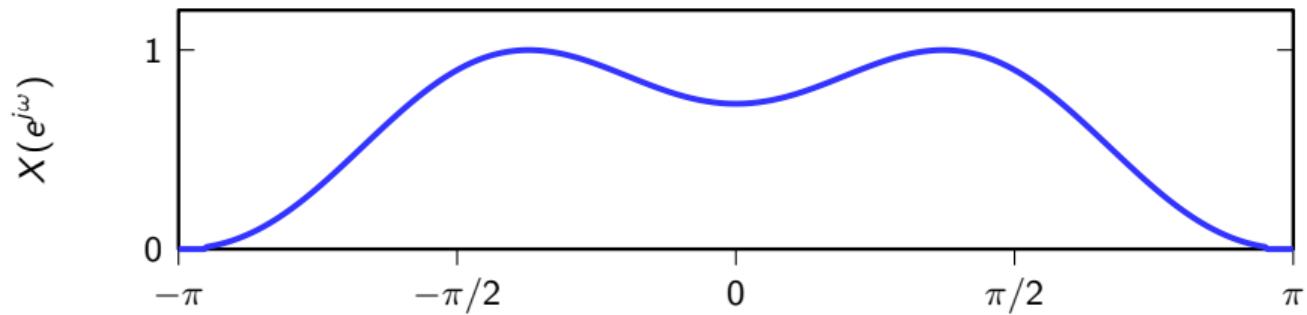
Oversampled D/A

Oversampled D/A

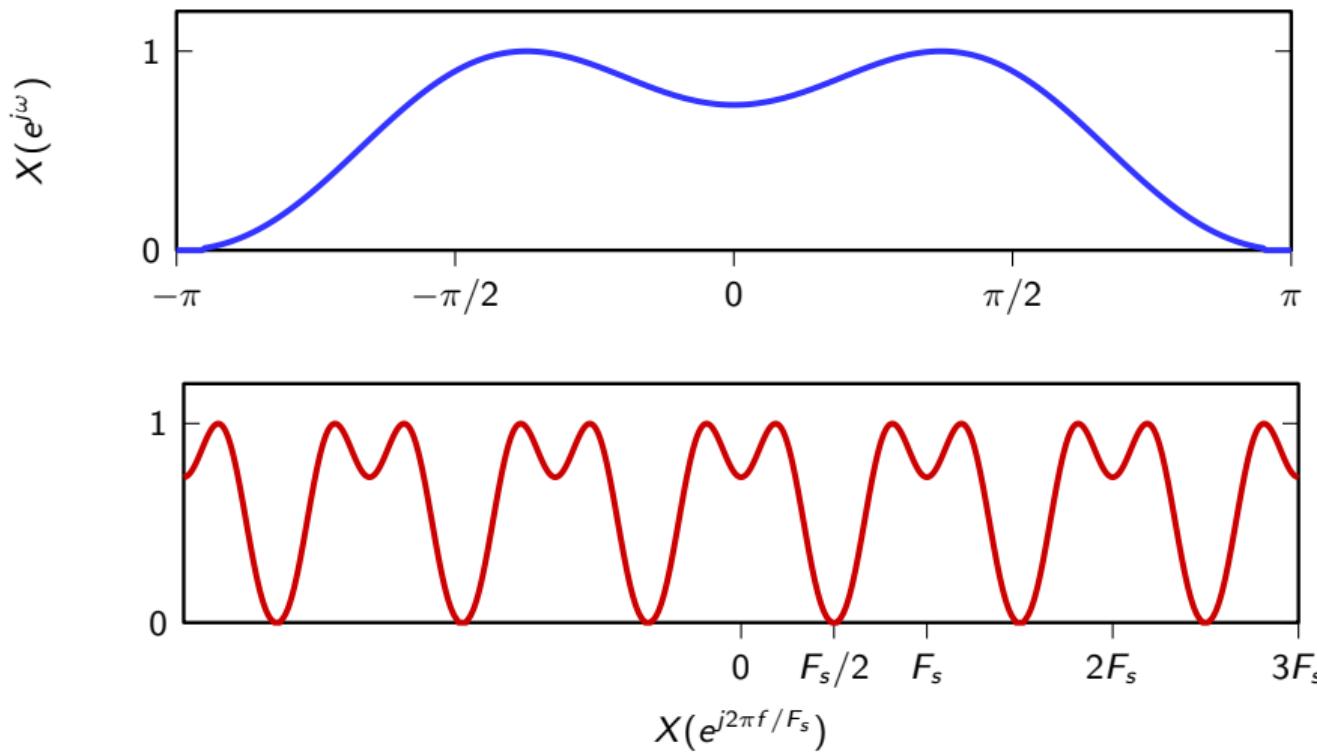
Sinc interpolation:

$$X_c(f) = \frac{1}{F_s} X(e^{j2\pi f/F_s}) \operatorname{rect}\left(\frac{f}{F_s}\right)$$

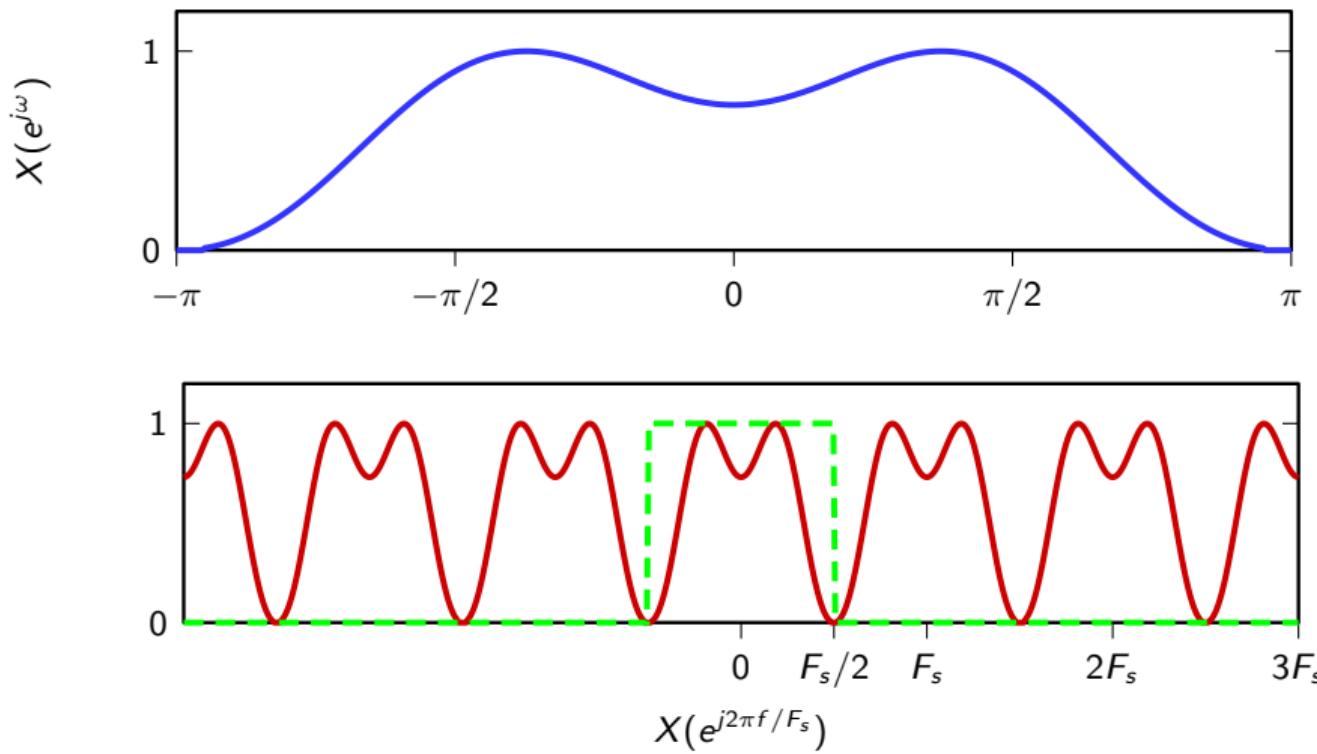
Sinc interpolation



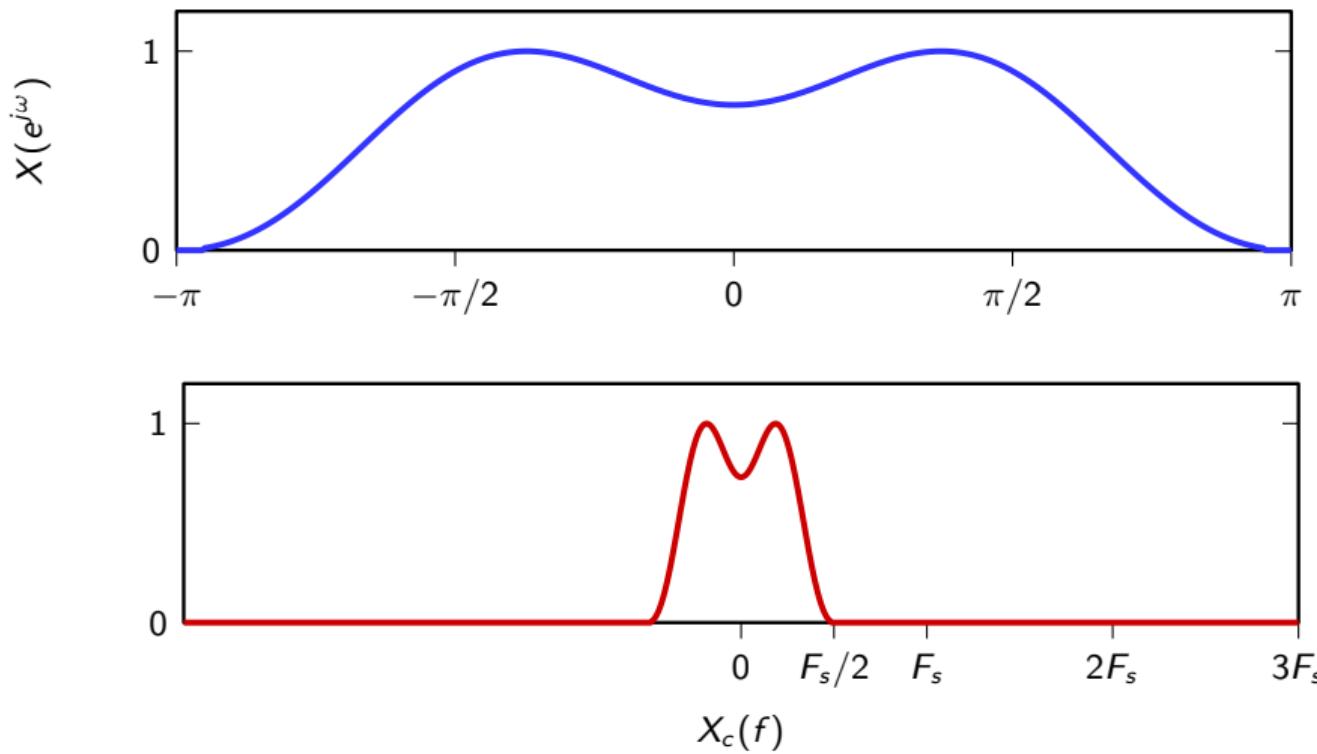
Sinc interpolation



Sinc interpolation



Sinc interpolation



Oversampled D/A

In general:

$$X_c(f) = \frac{1}{F_s} X(e^{j2\pi f/F_s}) I\left(\frac{f}{F_s}\right)$$

The cheapest (hence most common) interpolator is the zero-order hold:

$$i_0(t) = \text{rect}(t)$$

$$I_0(f) = \text{sinc}(f)$$

Oversampled D/A

In general:

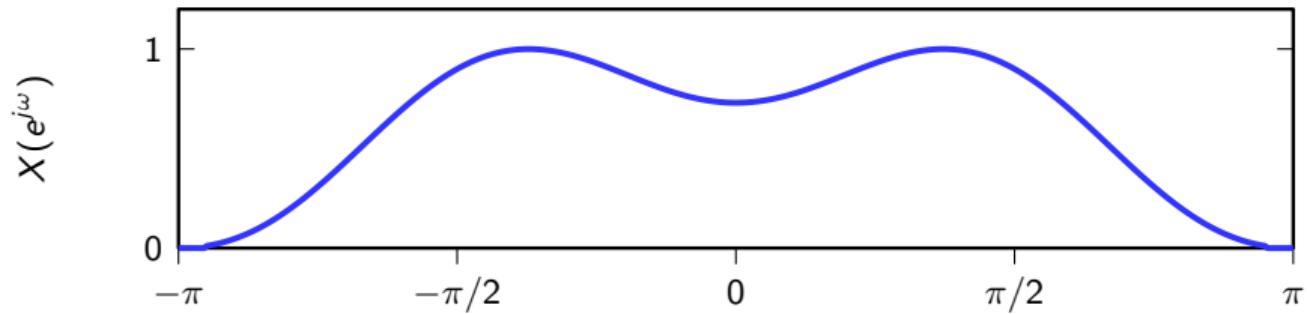
$$X_c(f) = \frac{1}{F_s} X(e^{j2\pi f/F_s}) I\left(\frac{f}{F_s}\right)$$

The cheapest (hence most common) interpolator is the zero-order hold:

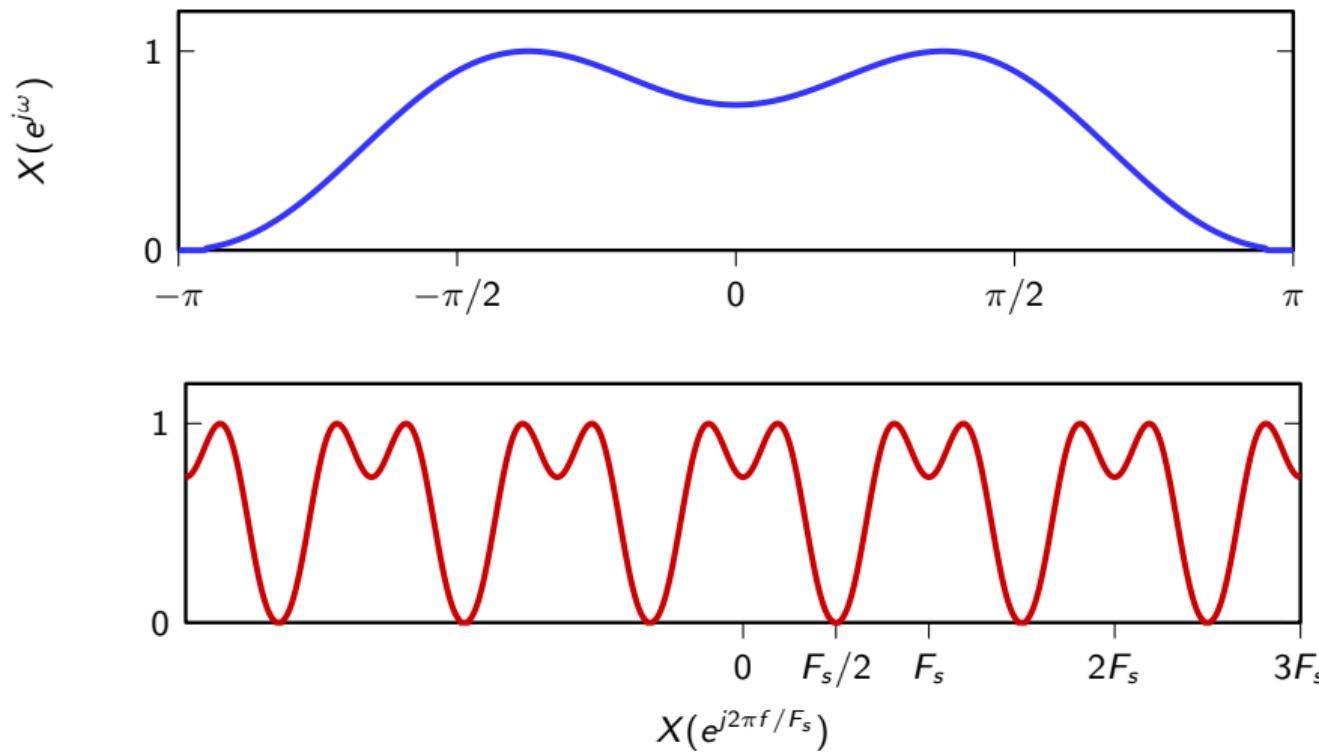
$$i_0(t) = \text{rect}(t)$$

$$I_0(f) = \text{sinc}(f)$$

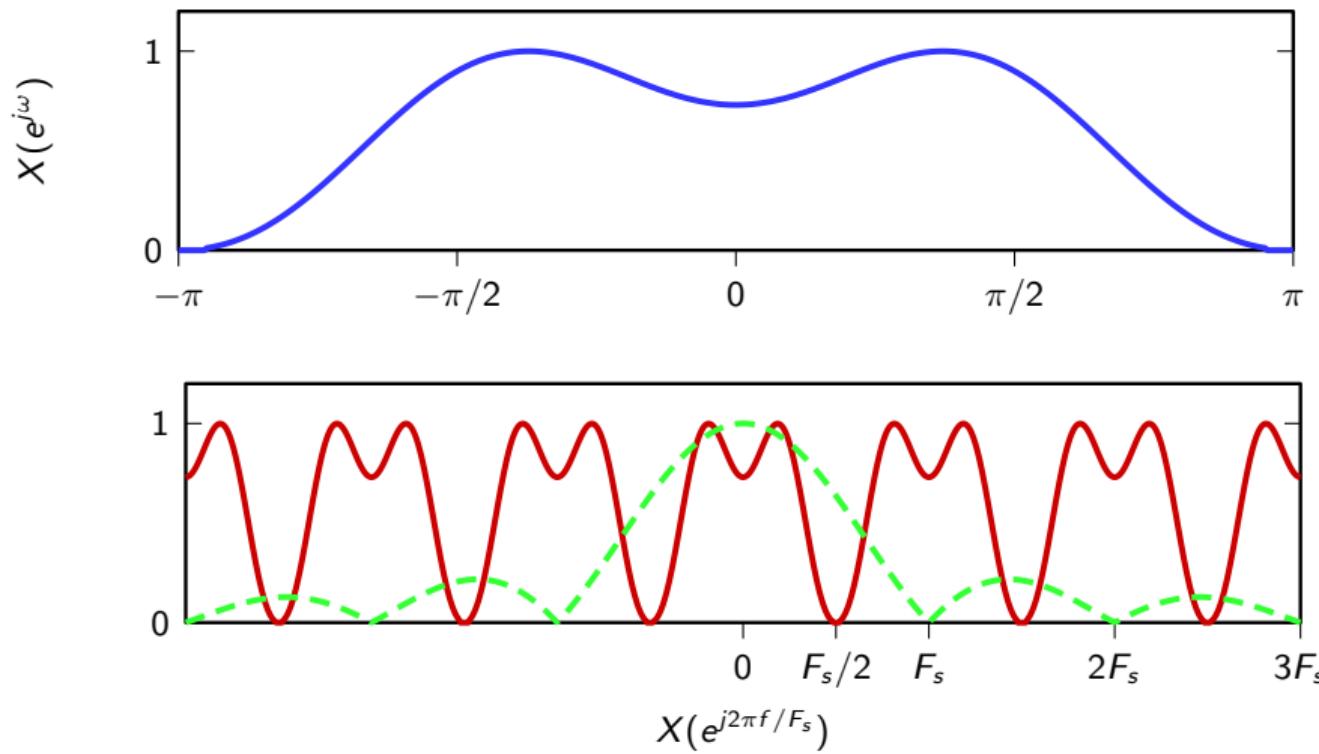
ZOH interpolation



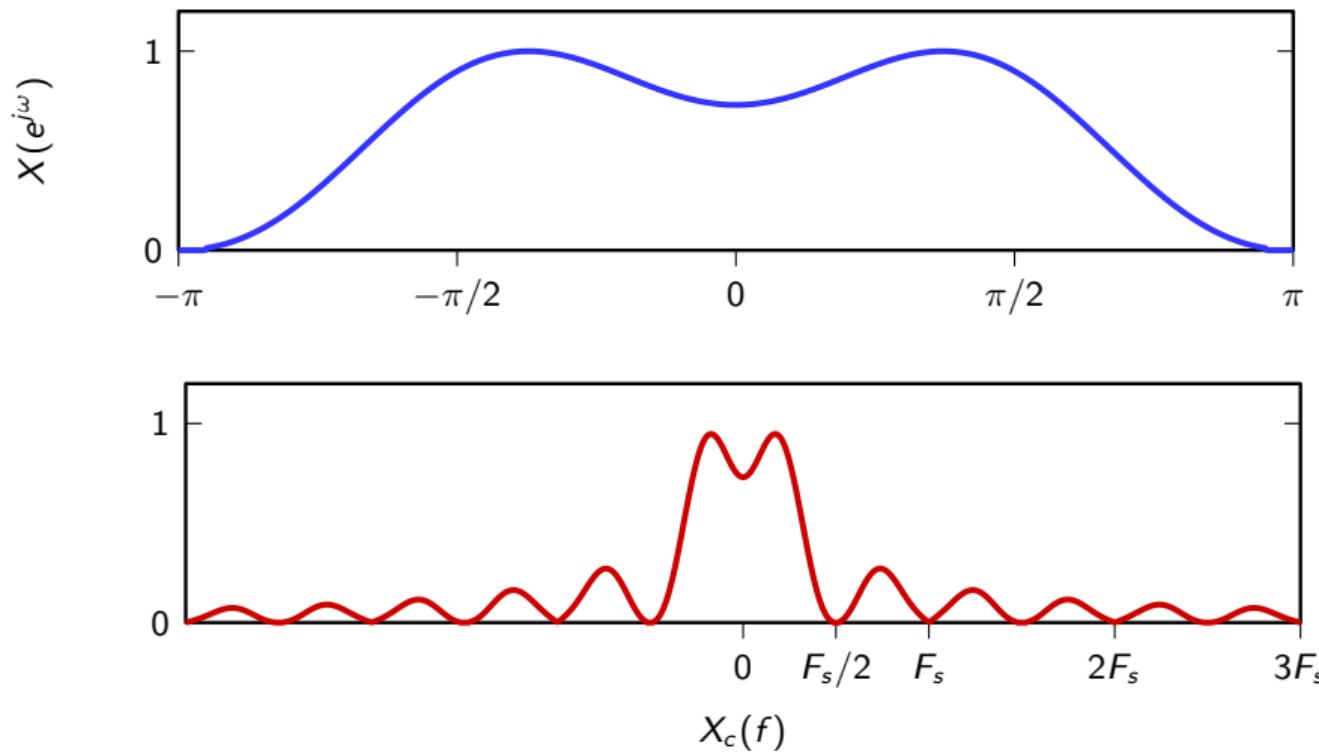
ZOH interpolation



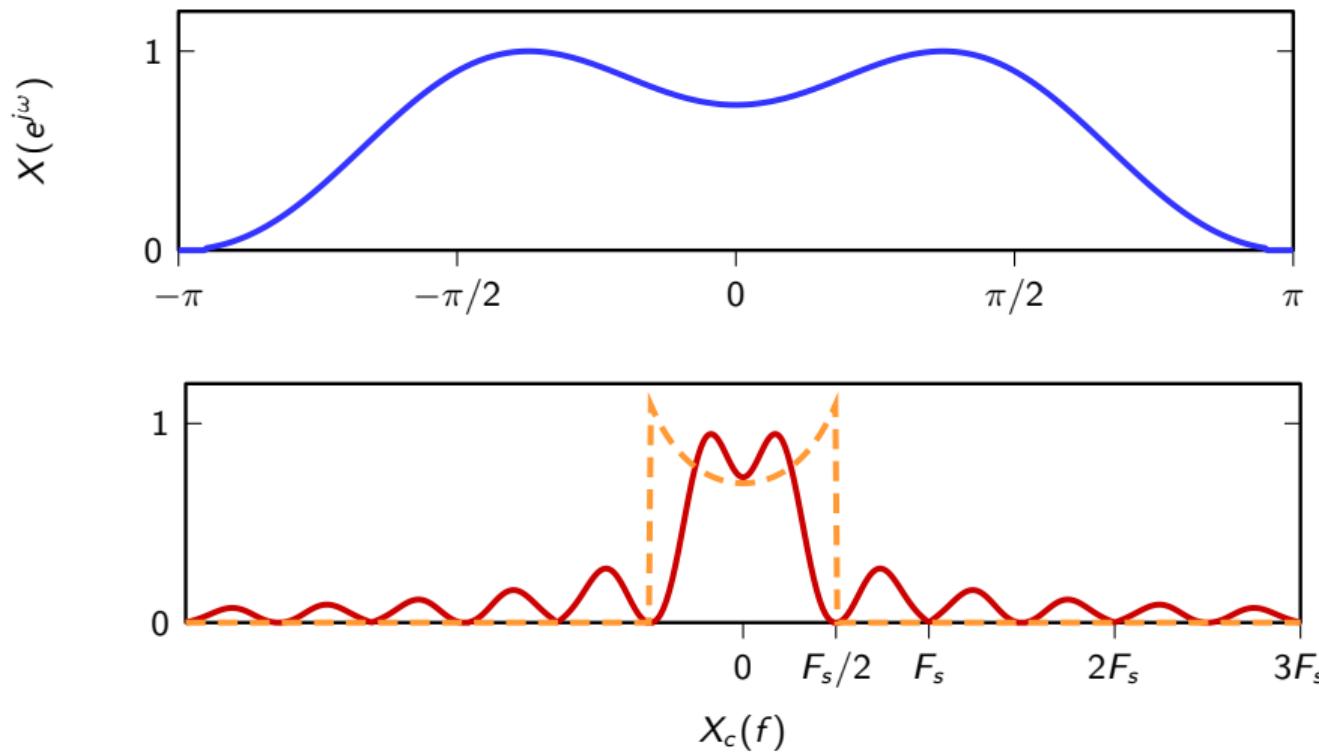
ZOH interpolation



ZOH interpolation



ZOH interpolation

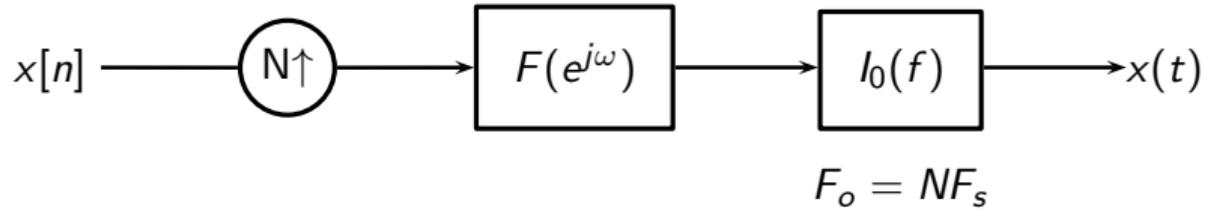


Oversampled A/D

key problems:

- ▶ we need to undo the in-band distortion in the analog domain
- ▶ we have a significant out-of-band distortion
- ▶ only advantage: minimal D/A rate

Oversampled D/A



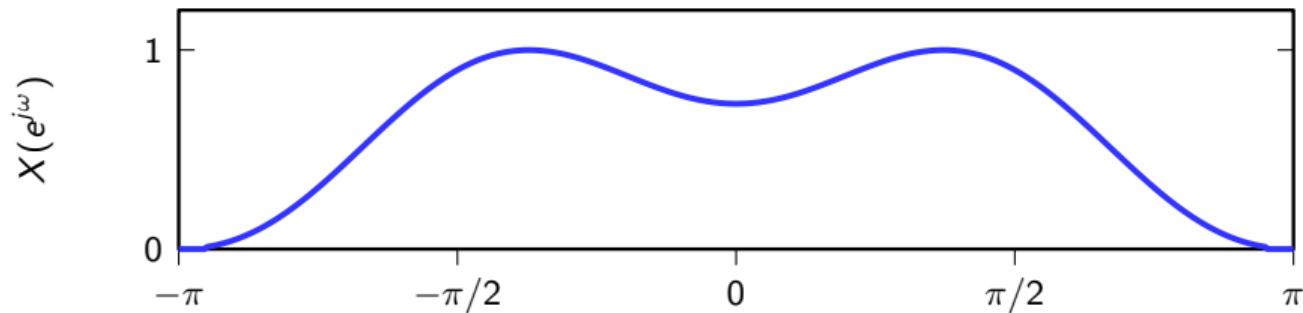
Oversampled D/A

consider a N -upsampled and interpolated version of $x[n]$:

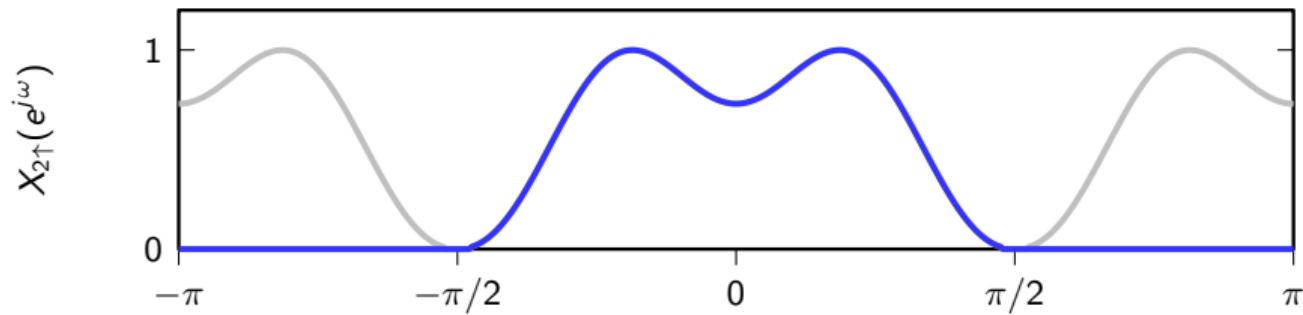
$$X_o(e^{j\omega}) = X_{N\uparrow}(e^{j\omega}) F(e^{j\omega}) = X(e^{j\omega N}) F(e^{j\omega})$$

- ▶ $F(e^{j\omega}) = N \operatorname{rect}(\omega N/(2\pi)) C(e^{j\omega})$
- ▶ rect matches the upsampler
- ▶ $C(e^{j\omega})$ compensates for zoh in-band distortion

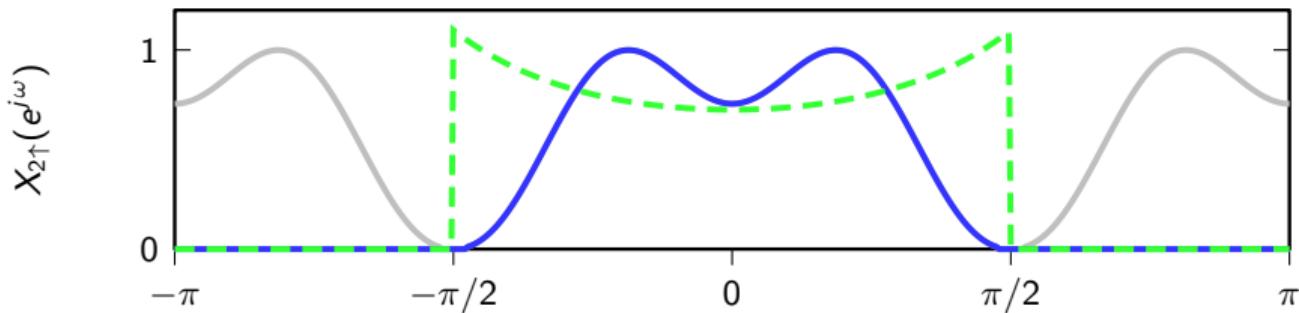
Oversampled D/A ($N = 2$)



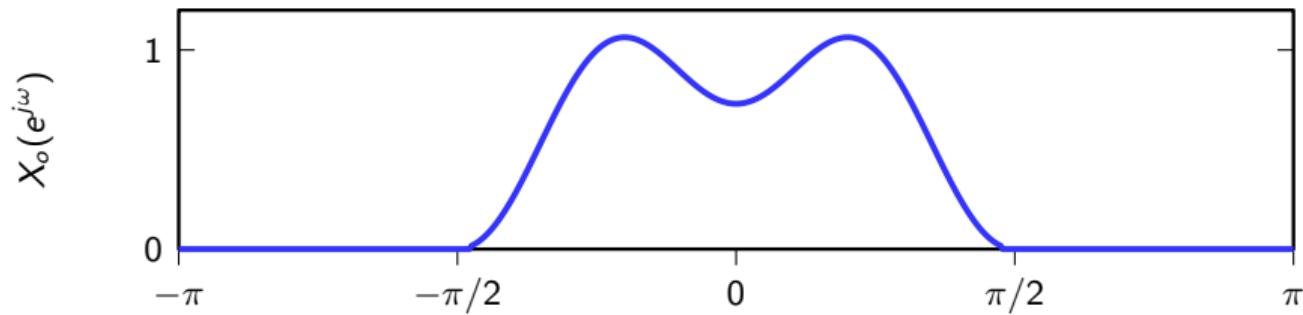
Oversampled D/A ($N = 2$)



Oversampled D/A ($N = 2$)



Oversampled D/A ($N = 2$)



Oversampled D/A

interpolate $x_o[n]$ with $F_o = NF_s$:

$$\begin{aligned} X_o(f) &= \frac{1}{F_o} X_o(e^{j2\pi f/F_o}) I_0\left(\frac{f}{F_o}\right) \\ &= \frac{1}{F_o} \left[X(e^{j\omega N}) F(e^{j\omega}) \right]_{\omega=2\pi f/F_o} I_0\left(\frac{f}{F_o}\right) \\ &= \frac{N}{F_o} \left[X(e^{j\omega N}) \operatorname{rect}\left(\frac{\omega N}{2\pi}\right) C(e^{j\omega}) \right]_{\omega=2\pi f/F_o} I_0\left(\frac{f}{F_o}\right) \\ &= \frac{1}{F_s} X(e^{j2\pi f/F_s}) \operatorname{rect}\left(\frac{f}{F_s}\right) C(e^{j2\pi f/F_o}) I_0\left(\frac{f}{F_o}\right) \\ &= X(f) \quad \text{for } |f| < F_s/2 \end{aligned}$$

Oversampled D/A

interpolate $x_o[n]$ with $F_o = NF_s$:

$$\begin{aligned} X_o(f) &= \frac{1}{F_o} X_o(e^{j2\pi f/F_o}) I_0\left(\frac{f}{F_o}\right) \\ &= \frac{1}{F_o} \left[X(e^{j\omega N}) F(e^{j\omega}) \right]_{\omega=2\pi f/F_o} I_0\left(\frac{f}{F_o}\right) \\ &= \frac{N}{F_o} \left[X(e^{j\omega N}) \operatorname{rect}\left(\frac{\omega N}{2\pi}\right) C(e^{j\omega}) \right]_{\omega=2\pi f/F_o} I_0\left(\frac{f}{F_o}\right) \\ &= \frac{1}{F_s} X(e^{j2\pi f/F_s}) \operatorname{rect}\left(\frac{f}{F_s}\right) C(e^{j2\pi f/F_o}) I_0\left(\frac{f}{F_o}\right) \\ &= X(f) \quad \text{for } |f| < F_s/2 \end{aligned}$$

Oversampled D/A

interpolate $x_o[n]$ with $F_o = NF_s$:

$$\begin{aligned} X_o(f) &= \frac{1}{F_o} X_o(e^{j2\pi f/F_o}) I_0\left(\frac{f}{F_o}\right) \\ &= \frac{1}{F_o} \left[X(e^{j\omega N}) F(e^{j\omega}) \right]_{\omega=2\pi f/F_o} I_0\left(\frac{f}{F_o}\right) \\ &= \frac{N}{F_o} \left[X(e^{j\omega N}) \operatorname{rect}\left(\frac{\omega N}{2\pi}\right) C(e^{j\omega}) \right]_{\omega=2\pi f/F_o} I_0\left(\frac{f}{F_o}\right) \\ &= \frac{1}{F_s} X(e^{j2\pi f/F_s}) \operatorname{rect}\left(\frac{f}{F_s}\right) C(e^{j2\pi f/F_o}) I_0\left(\frac{f}{F_o}\right) \\ &= X(f) \quad \text{for } |f| < F_s/2 \end{aligned}$$

Oversampled D/A

interpolate $x_o[n]$ with $F_o = NF_s$:

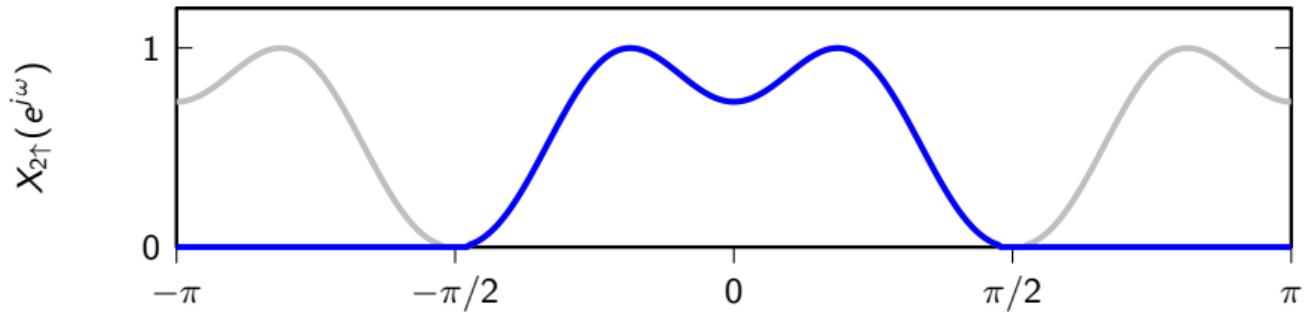
$$\begin{aligned} X_o(f) &= \frac{1}{F_o} X_o(e^{j2\pi f/F_o}) I_0\left(\frac{f}{F_o}\right) \\ &= \frac{1}{F_o} \left[X(e^{j\omega N}) F(e^{j\omega}) \right]_{\omega=2\pi f/F_o} I_0\left(\frac{f}{F_o}\right) \\ &= \frac{N}{F_o} \left[X(e^{j\omega N}) \operatorname{rect}\left(\frac{\omega N}{2\pi}\right) C(e^{j\omega}) \right]_{\omega=2\pi f/F_o} I_0\left(\frac{f}{F_o}\right) \\ &= \frac{1}{F_s} X(e^{j2\pi f/F_s}) \operatorname{rect}\left(\frac{f}{F_s}\right) C(e^{j2\pi f/F_o}) I_0\left(\frac{f}{F_o}\right) \\ &= X(f) \quad \text{for } |f| < F_s/2 \end{aligned}$$

Oversampled D/A

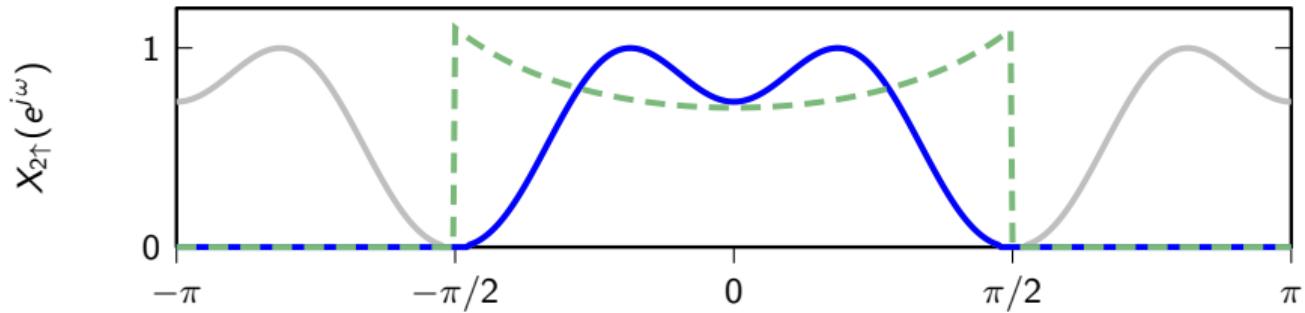
interpolate $x_o[n]$ with $F_o = NF_s$:

$$\begin{aligned} X_o(f) &= \frac{1}{F_o} X_o(e^{j2\pi f/F_o}) I_0\left(\frac{f}{F_o}\right) \\ &= \frac{1}{F_o} \left[X(e^{j\omega N}) F(e^{j\omega}) \right]_{\omega=2\pi f/F_o} I_0\left(\frac{f}{F_o}\right) \\ &= \frac{N}{F_o} \left[X(e^{j\omega N}) \operatorname{rect}\left(\frac{\omega N}{2\pi}\right) C(e^{j\omega}) \right]_{\omega=2\pi f/F_o} I_0\left(\frac{f}{F_o}\right) \\ &= \frac{1}{F_s} X(e^{j2\pi f/F_s}) \operatorname{rect}\left(\frac{f}{F_s}\right) C(e^{j2\pi f/F_o}) I_0\left(\frac{f}{F_o}\right) \\ &= X(f) \quad \text{for } |f| < F_s/2 \end{aligned}$$

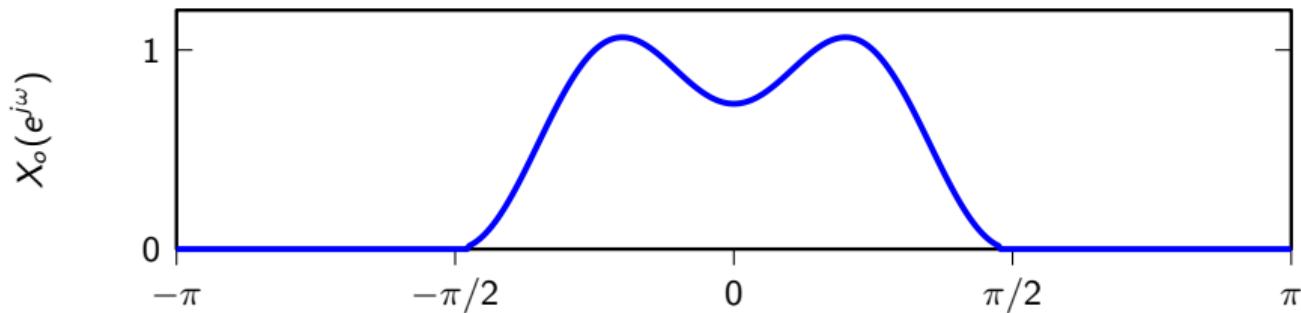
Oversampled D/A, using a ZOH ($N = 2$)



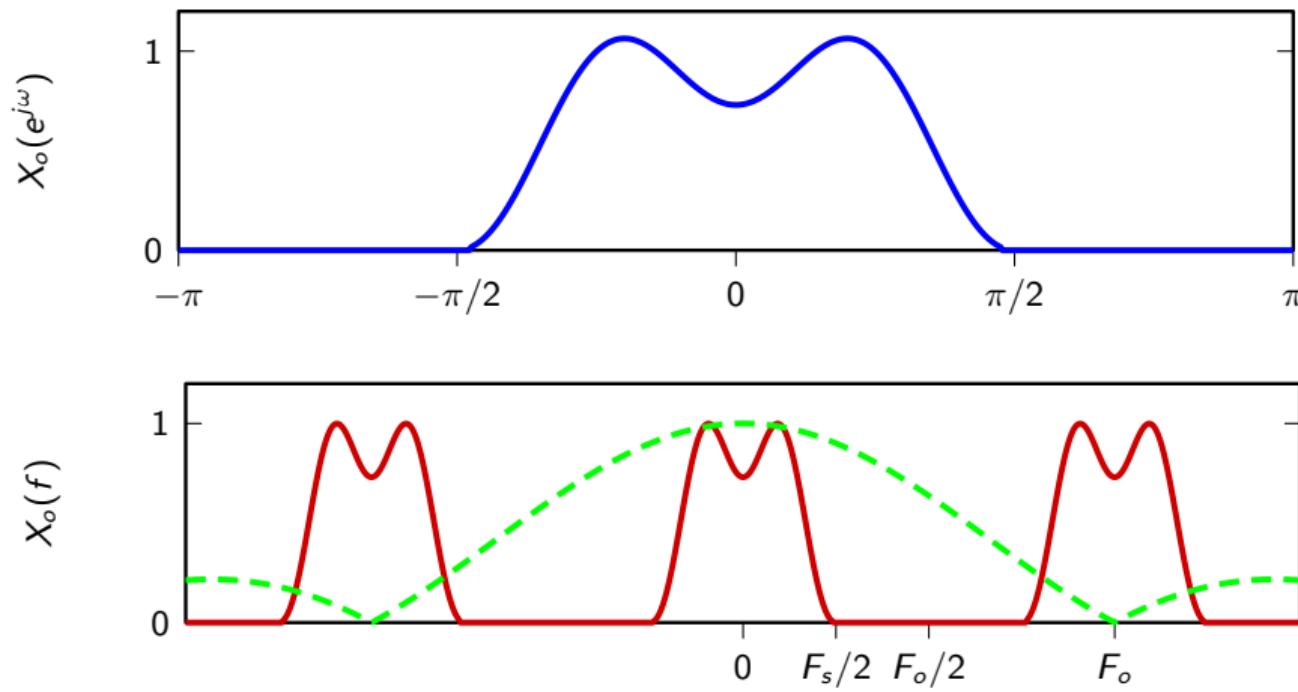
Oversampled D/A, using a ZOH ($N = 2$)



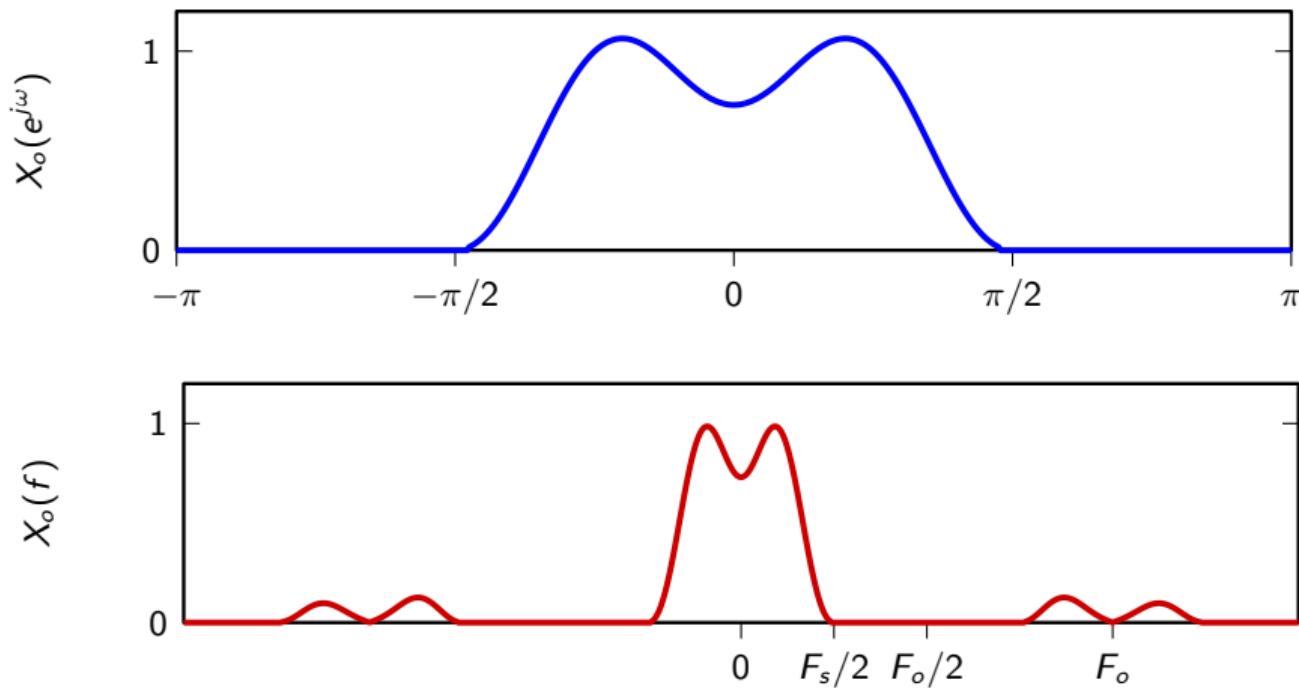
Oversampled D/A, using a ZOH ($N = 2$)



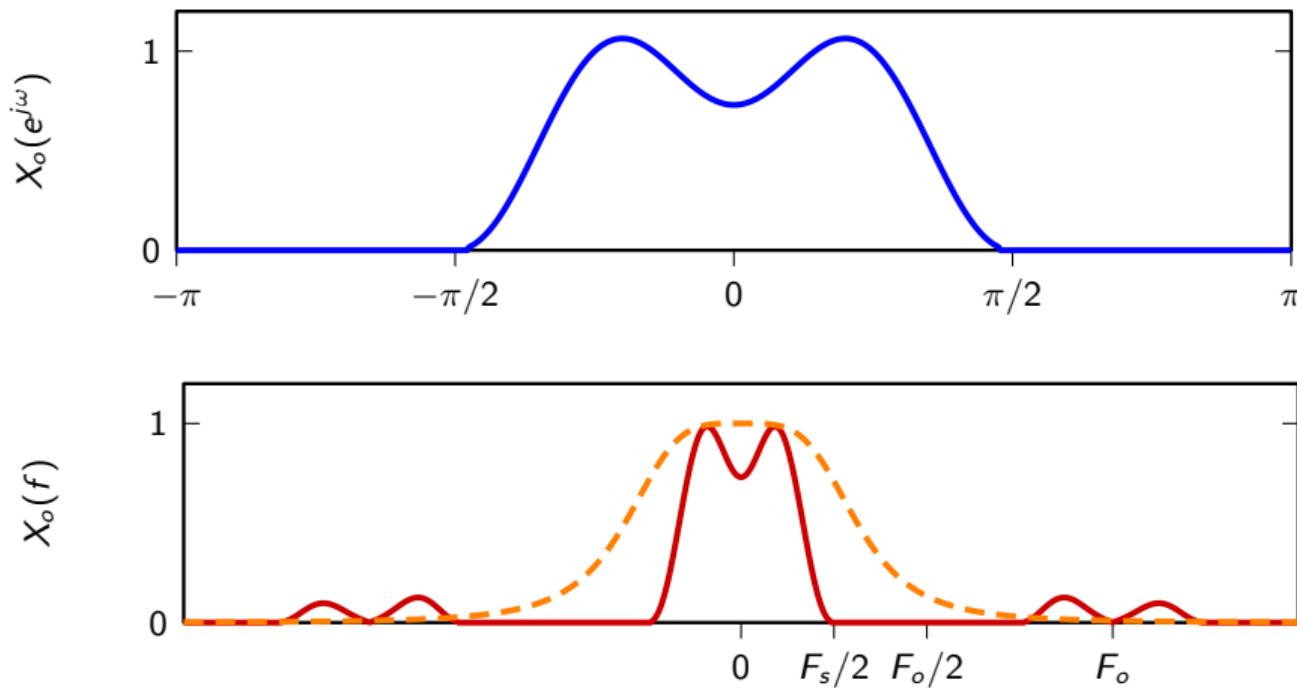
Oversampled D/A, using a ZOH ($N = 2$)



Oversampled D/A, using a ZOH ($N = 2$)



Oversampled D/A, using a ZOH ($N = 2$)



Oversampled A/D

key points:

- ▶ we can pre-compensate the in-band distortion in the digital domain
- ▶ we can interpolate with a cheap ZOH
- ▶ the higher the upsampling, the cheaper the analog lowpass needed to eliminate out-of-band distortion
- ▶ only price: higher D/A rate

COM303: Digital Signal Processing

Lecture 20: Image Processing

overview

- ▶ Introduction to Images and Image Processing
- ▶ Affine Transforms
- ▶ 2D Fourier Analysis
- ▶ Image Filters

Overview:

- ▶ Images as multidimensional digital signals
- ▶ 2D signal representations
- ▶ Basic signals and operators

Overview:

- ▶ Images as multidimensional digital signals
- ▶ 2D signal representations
- ▶ Basic signals and operators

Overview:

- ▶ Images as multidimensional digital signals
- ▶ 2D signal representations
- ▶ Basic signals and operators

In the old, non-PC days...



Please meet ...



Digital images

- ▶ two-dimensional signal $x[n_1, n_2]$, $n_1, n_2 \in \mathbb{Z}$
- ▶ indices locate a point on a grid → pixel
- ▶ grid is usually regularly spaced
- ▶ values $x[n_1, n_2]$ refer to the pixel's appearance

Digital images

- ▶ two-dimensional signal $x[n_1, n_2]$, $n_1, n_2 \in \mathbb{Z}$
- ▶ indices locate a point on a grid → pixel
- ▶ grid is usually regularly spaced
- ▶ values $x[n_1, n_2]$ refer to the pixel's appearance

Digital images

- ▶ two-dimensional signal $x[n_1, n_2]$, $n_1, n_2 \in \mathbb{Z}$
- ▶ indices locate a point on a grid → pixel
- ▶ grid is usually regularly spaced
- ▶ values $x[n_1, n_2]$ refer to the pixel's appearance

Digital images

- ▶ two-dimensional signal $x[n_1, n_2]$, $n_1, n_2 \in \mathbb{Z}$
- ▶ indices locate a point on a grid → pixel
- ▶ grid is usually regularly spaced
- ▶ values $x[n_1, n_2]$ refer to the pixel's appearance

Digital images: grayscale vs color

- ▶ grayscale images: scalar pixel values
- ▶ color images: multidimensional pixel values in a color space (RGB, HSV, YUV, etc)
- ▶ we can consider the single components separately:

Digital images: grayscale vs color

- ▶ grayscale images: scalar pixel values
- ▶ color images: multidimensional pixel values in a color space (RGB, HSV, YUV, etc)
- ▶ we can consider the single components separately:

Digital images: grayscale vs color

- ▶ grayscale images: scalar pixel values
- ▶ color images: multidimensional pixel values in a color space (RGB, HSV, YUV, etc)
- ▶ we can consider the single components separately:

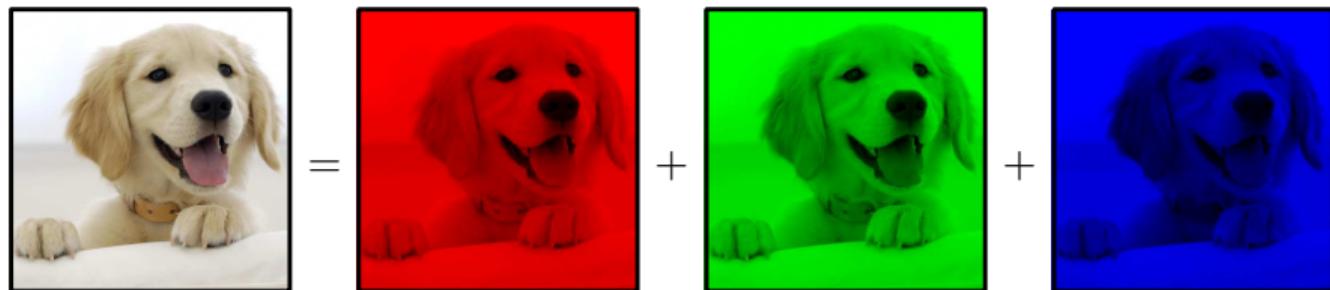


Image processing

From one to two dimensions...

- ▶ something still works
- ▶ something breaks down
- ▶ something is new

Image processing

From one to two dimensions...

- ▶ something still works
- ▶ something breaks down
- ▶ something is new

Image processing

From one to two dimensions...

- ▶ something still works
- ▶ something breaks down
- ▶ something is new

A thought experiment

- ▶ consider all possible 256×256 , 8bpp “images”
- ▶ each image is 524,288 bits
- ▶ total number of possible images: $2^{524,288} \approx 10^{157,826}$
- ▶ number of atoms in the universe: 10^{82}
- ▶ how many “images” are proper images?

A thought experiment

- ▶ consider all possible 256×256 , 8bpp “images”
- ▶ each image is 524,288 bits
- ▶ total number of possible images: $2^{524,288} \approx 10^{157,826}$
- ▶ number of atoms in the universe: 10^{82}
- ▶ how many “images” are proper images?

A thought experiment

- ▶ consider all possible 256×256 , 8bpp “images”
- ▶ each image is 524,288 bits
- ▶ total number of possible images: $2^{524,288} \approx 10^{157,826}$
- ▶ number of atoms in the universe: 10^{82}
- ▶ how many “images” are proper images?

A thought experiment

- ▶ consider all possible 256×256 , 8bpp “images”
- ▶ each image is 524,288 bits
- ▶ total number of possible images: $2^{524,288} \approx 10^{157,826}$
- ▶ number of atoms in the universe: 10^{82}
- ▶ how many “images” are proper images?

A thought experiment

- ▶ consider all possible 256×256 , 8bpp “images”
- ▶ each image is 524,288 bits
- ▶ total number of possible images: $2^{524,288} \approx 10^{157,826}$
- ▶ number of atoms in the universe: 10^{82}
- ▶ how many “images” are proper images?

Image processing

images are very specialized signals,
“designed” for a very specific processing system: the human brain!

visual semantics is extremely hard to deal with

Image processing

What works:

- ▶ the maths: linearity, space-invariance, convolution
- ▶ Fourier transform formulas
- ▶ interpolation, sampling, quantization

Image processing

What works:

- ▶ the maths: linearity, space-invariance, convolution
- ▶ Fourier transform formulas
- ▶ interpolation, sampling, quantization

Image processing

What works:

- ▶ the maths: linearity, space-invariance, convolution
- ▶ Fourier transform formulas
- ▶ interpolation, sampling, quantization

Image processing

What breaks down:

- ▶ Fourier analysis less relevant
- ▶ filter design hard, IIRs rare
- ▶ linear, space-invariant operators only mildly useful because of their isotropy

Image processing

What breaks down:

- ▶ Fourier analysis less relevant
- ▶ filter design hard, IIRs rare
- ▶ linear, space-invariant operators only mildly useful because of their isotropy

Image processing

What breaks down:

- ▶ Fourier analysis less relevant
- ▶ filter design hard, IIRs rare
- ▶ linear, space-invariant operators only mildly useful because of their isotropy

Image processing

What's new:

- ▶ new manipulations: affine transforms
- ▶ images are finite-support signals
- ▶ images are usually available in their entirety → causality mostly irrelevant

Image processing

What's new:

- ▶ new manipulations: affine transforms
- ▶ images are finite-support signals
- ▶ images are usually available in their entirety → causality mostly irrelevant

Image processing

What's new:

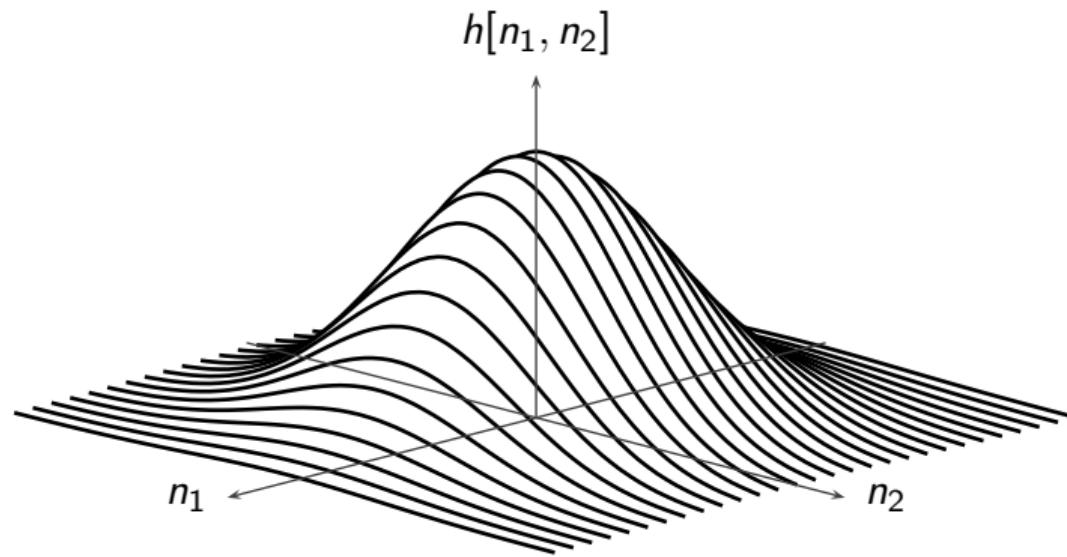
- ▶ new manipulations: affine transforms
- ▶ images are finite-support signals
- ▶ images are usually available in their entirety → causality mostly irrelevant

2D signal processing: the basics

A two-dimensional discrete-space signal:

$$x[n_1, n_2], \quad n_1, n_2 \in \mathbb{Z}$$

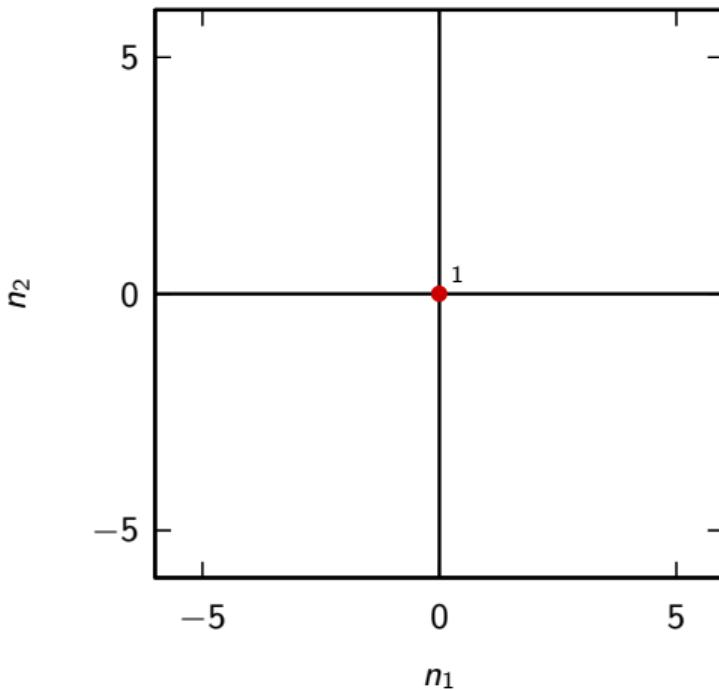
2D signals: Cartesian representation



2D signals: support representation

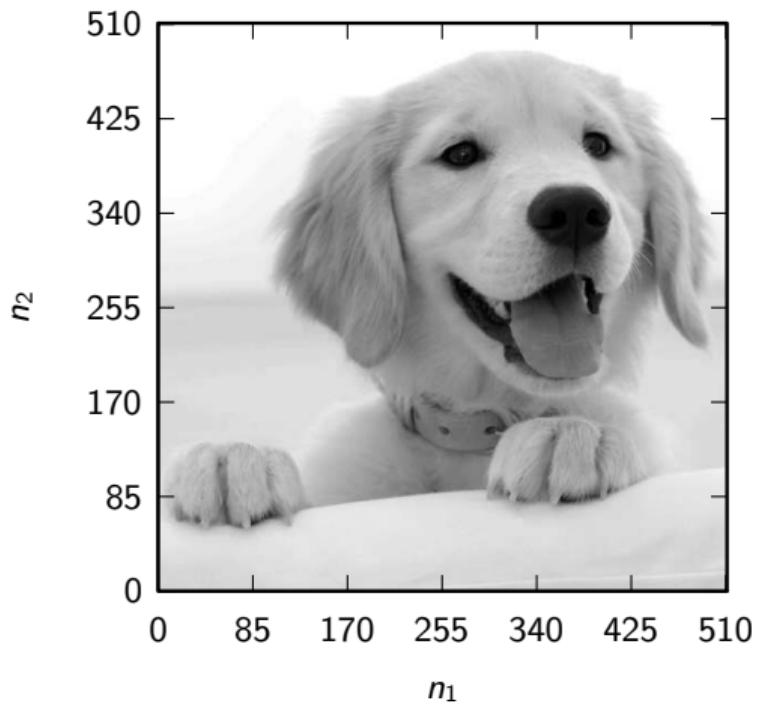
- ▶ just show coordinates of nonzero samples
- ▶ amplitude may be written along explicitly
- ▶ example:

$$\delta[n_1, n_2] = \begin{cases} 1 & \text{if } n_1 = n_2 = 0 \\ 0 & \text{otherwise.} \end{cases}$$



2D signals: image representation

- ▶ medium has a certain dynamic range (paper, screen)
- ▶ image values are quantized (usually to 8 bits, or 256 levels)
- ▶ the eye does the interpolation in space provided the pixel density is high enough



About dynamic ranges...

Images:

- ▶ human eye: 120dB
- ▶ prints: 12dB to 36dB
- ▶ LCD: 60dB
- ▶ digital cinema: 90dB

Sounds:

- ▶ human ear: 140dB
- ▶ speech: 40dB
- ▶ vinyl, tape: 50dB
- ▶ CD: 96dB

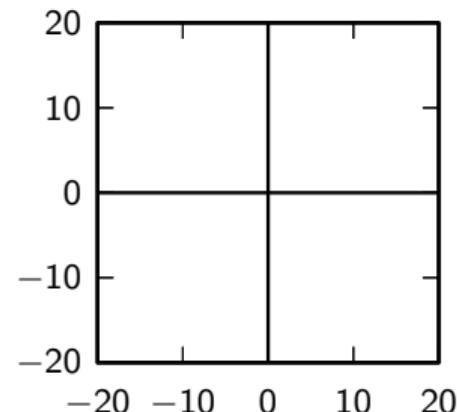
Why 2D?

- ▶ images could be unrolled (printers, fax)
- ▶ but what about spatial correlation?

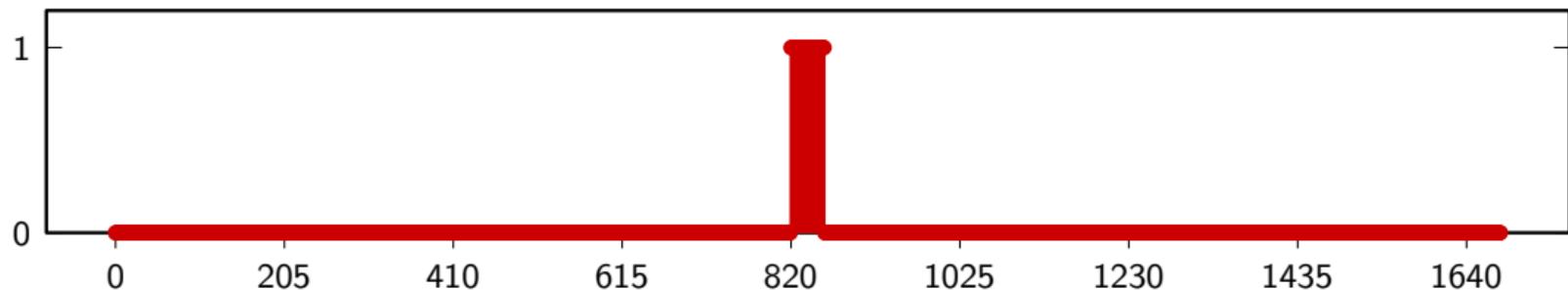
Why 2D?

- ▶ images could be unrolled (printers, fax)
- ▶ but what about spatial correlation?

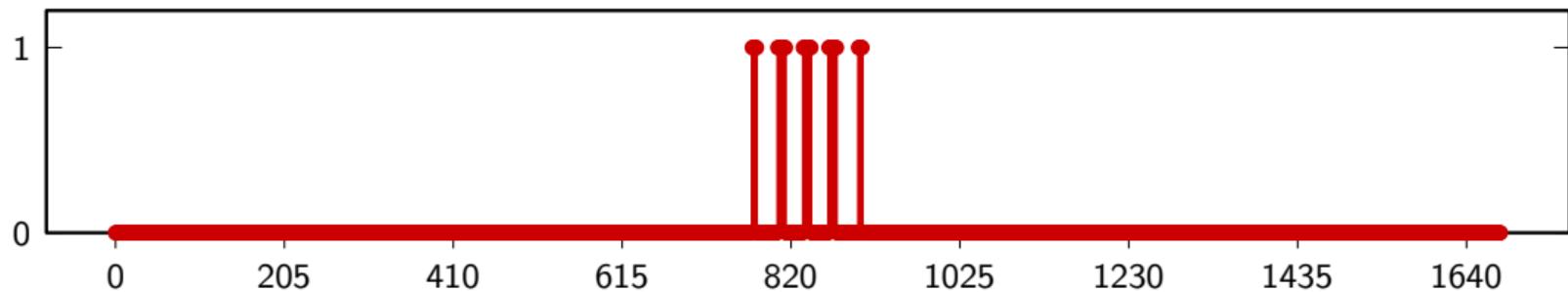
2D vs raster scan



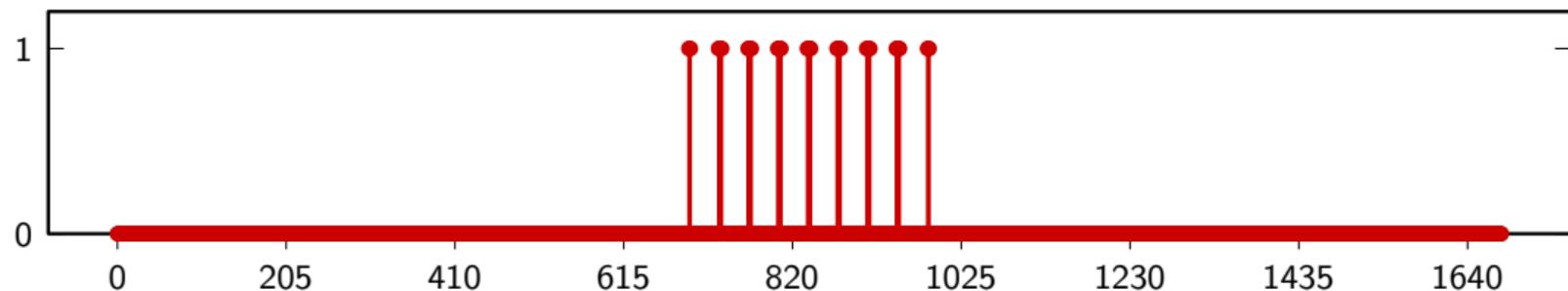
2D vs raster scan



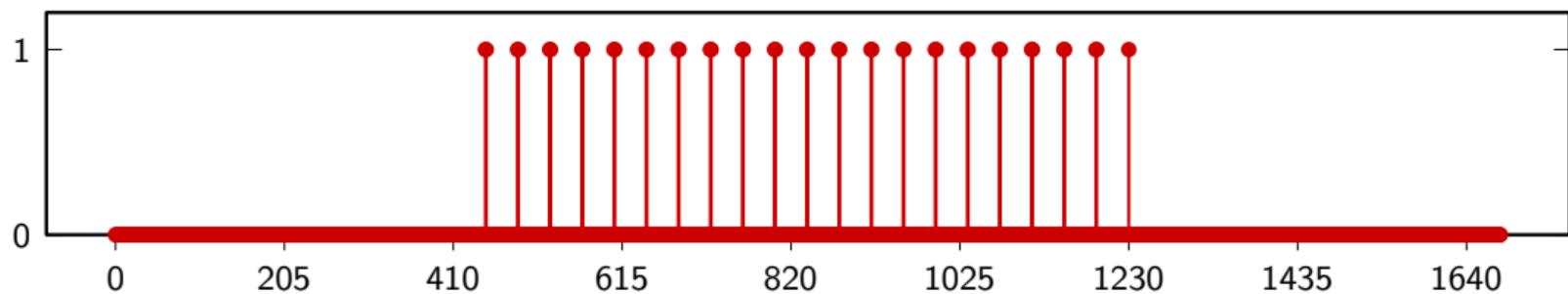
2D vs raster scan



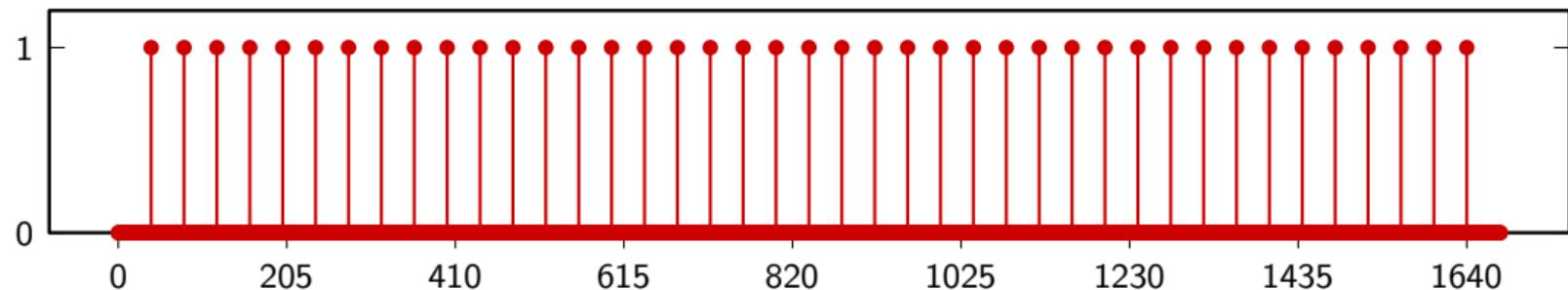
2D vs raster scan



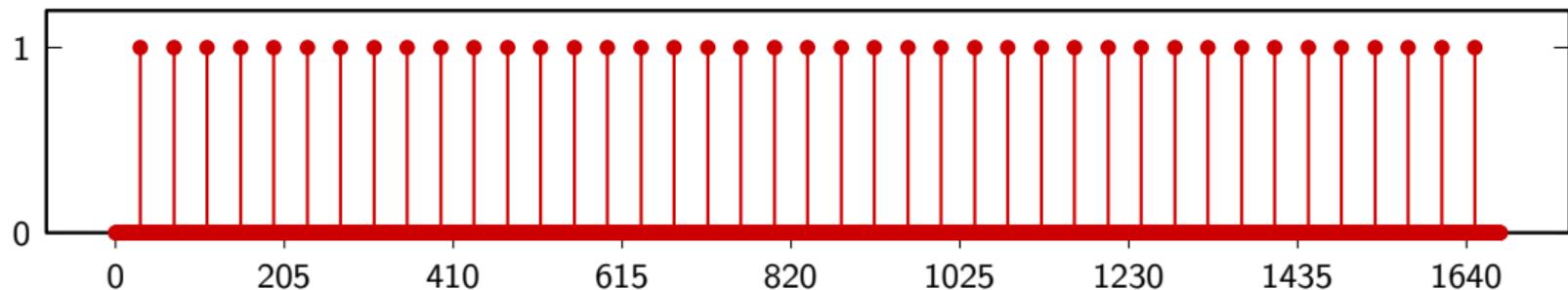
2D vs raster scan



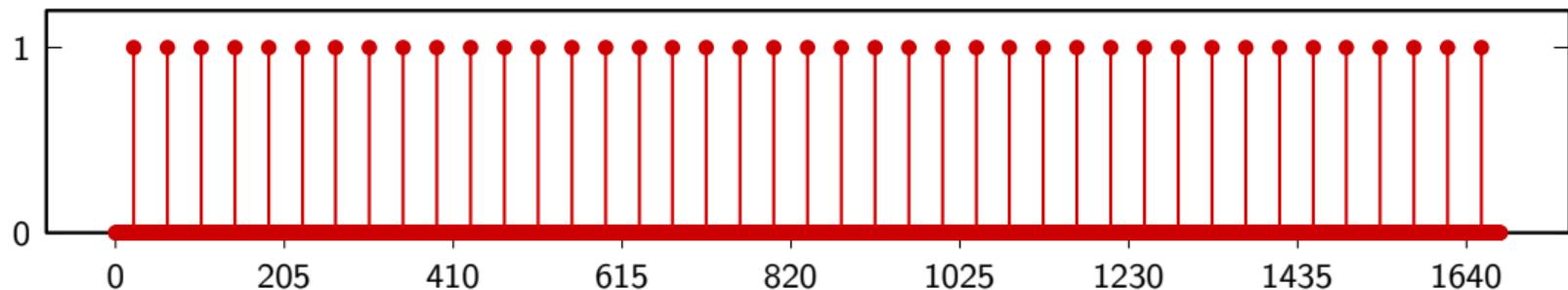
2D vs raster scan



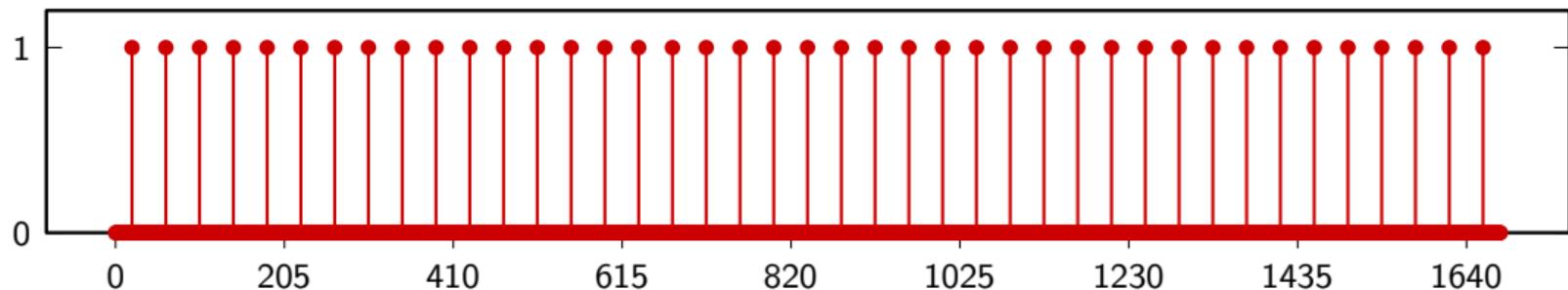
2D vs raster scan



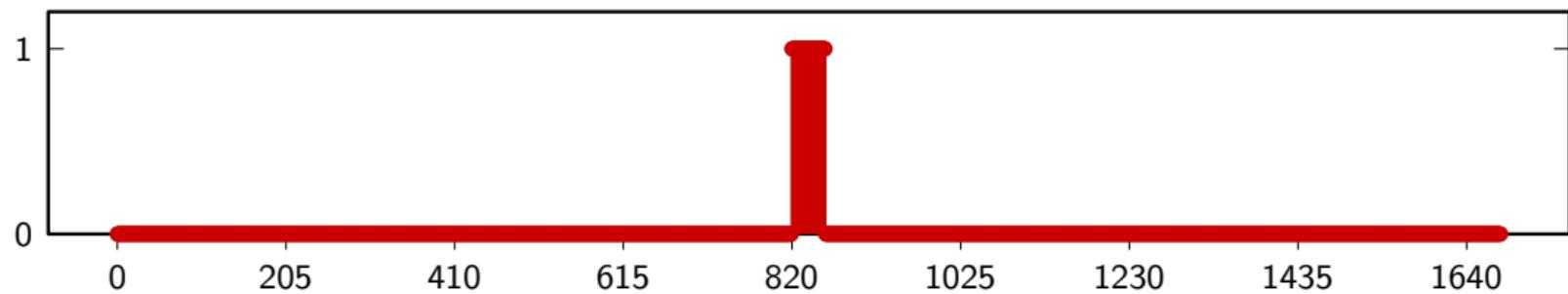
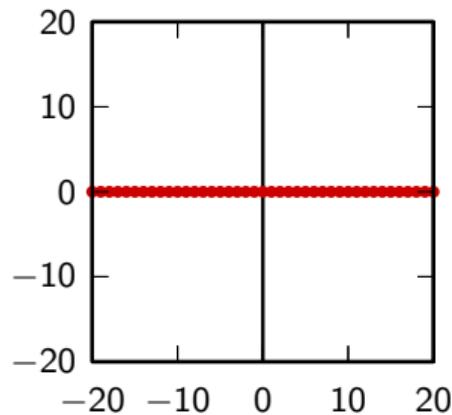
2D vs raster scan



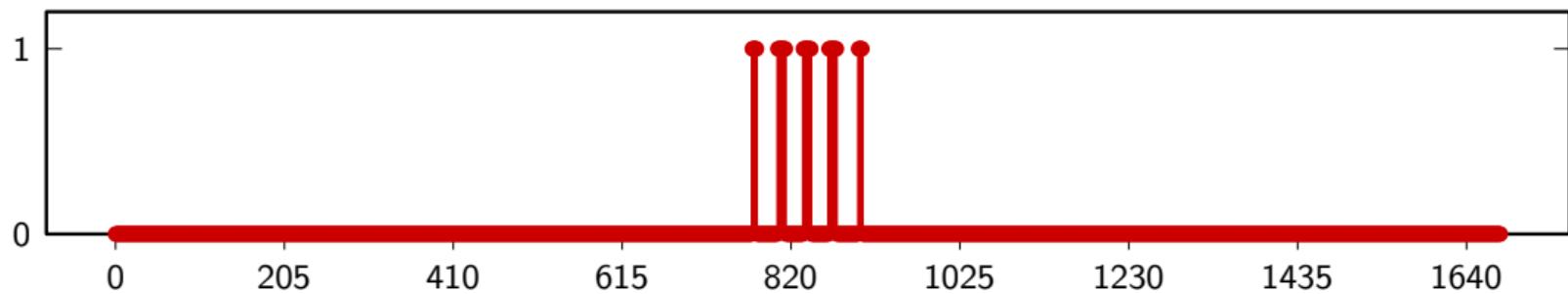
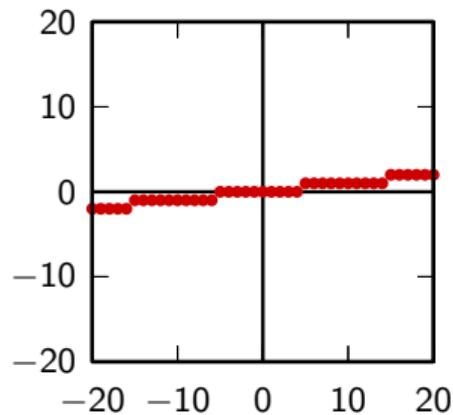
2D vs raster scan



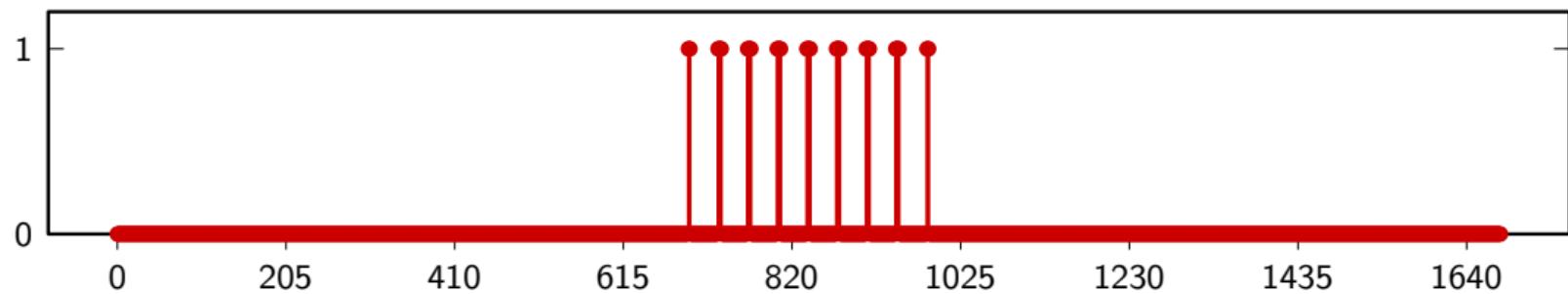
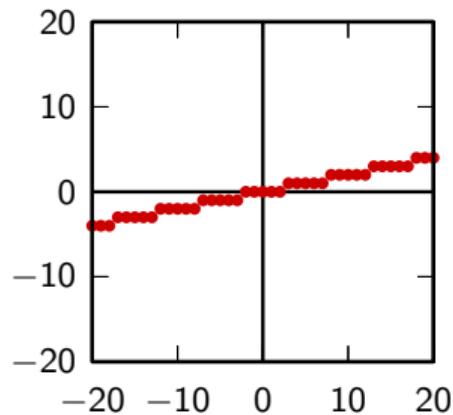
2D vs raster scan



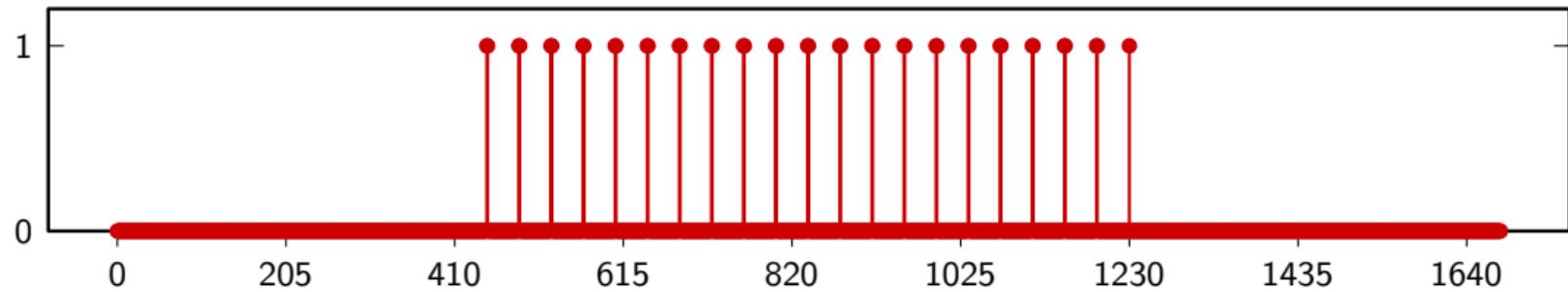
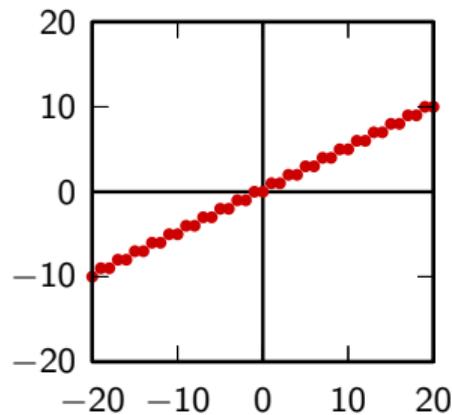
2D vs raster scan



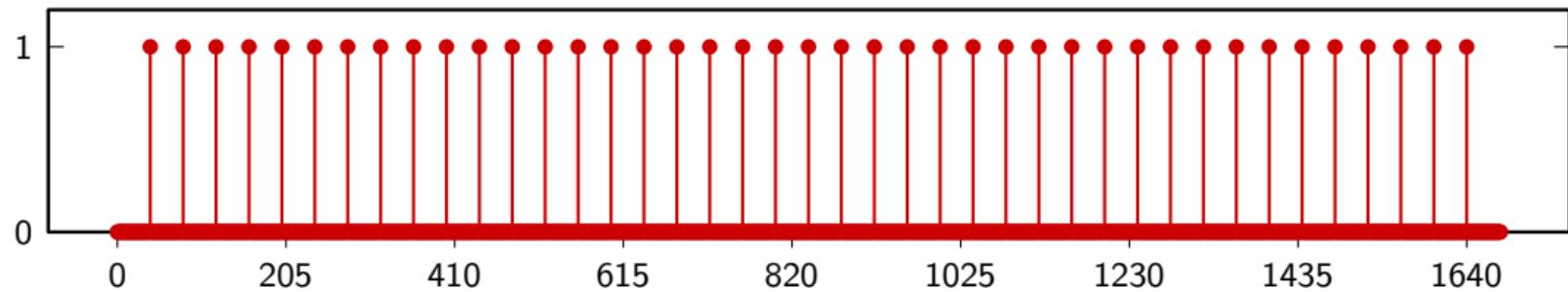
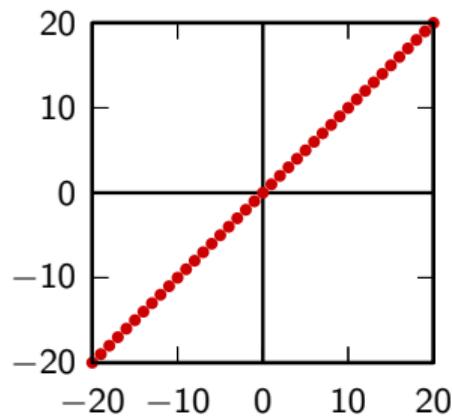
2D vs raster scan



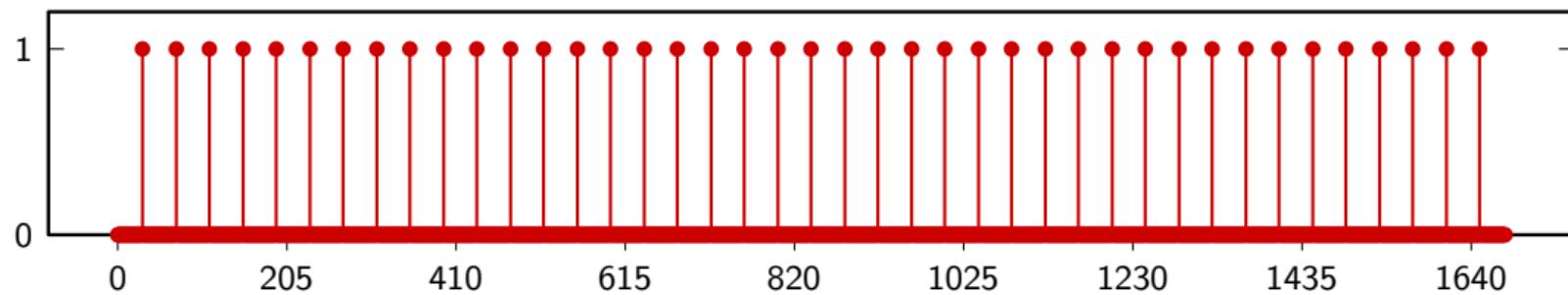
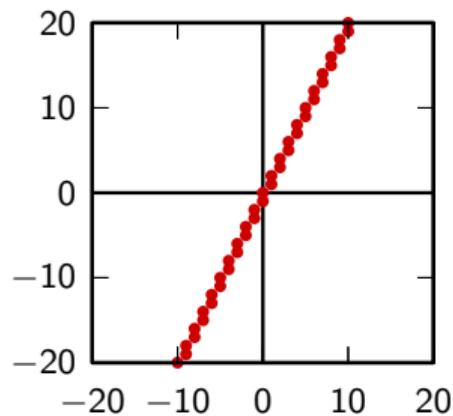
2D vs raster scan



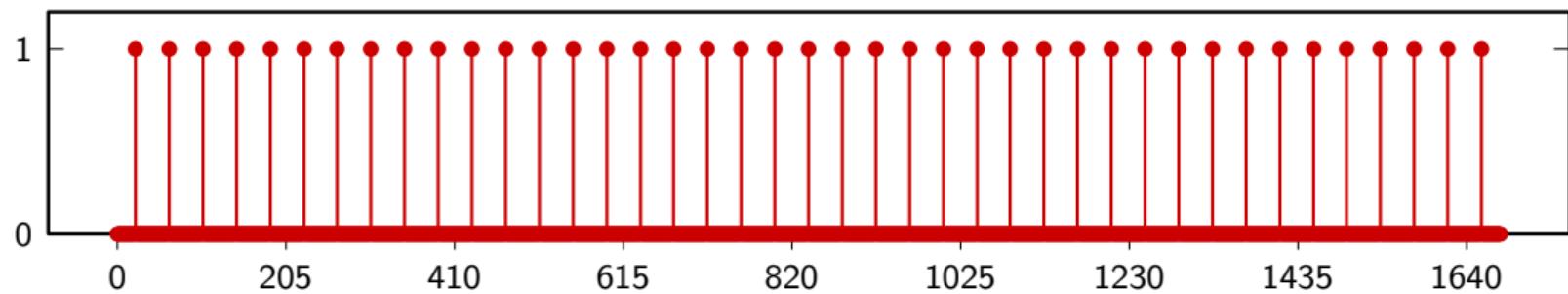
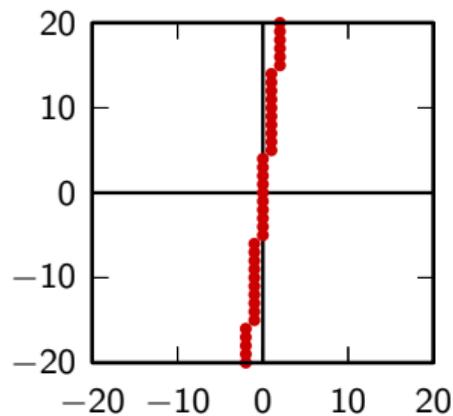
2D vs raster scan



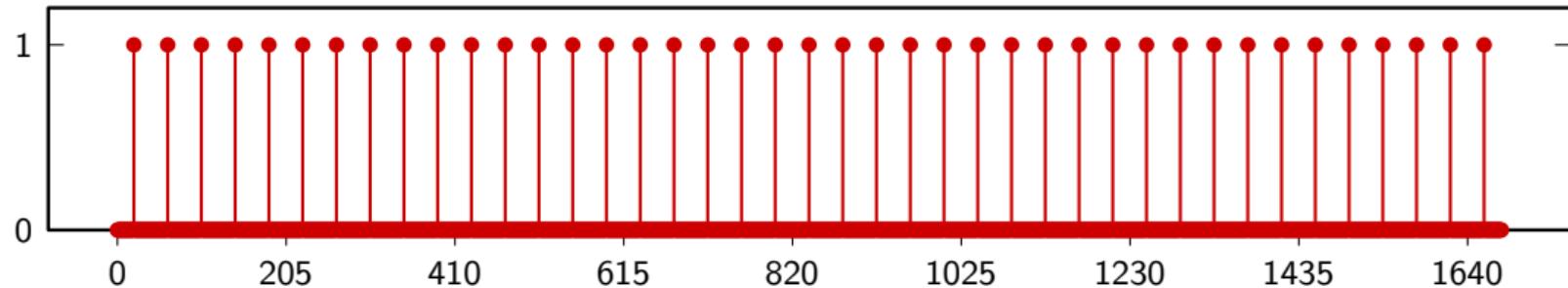
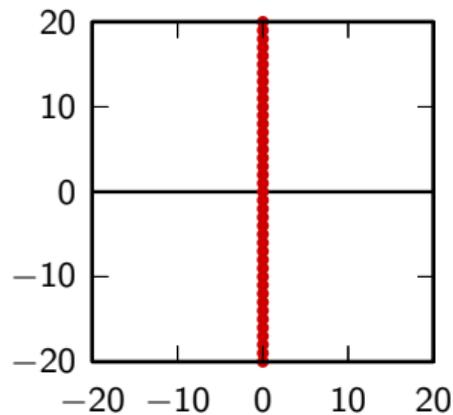
2D vs raster scan



2D vs raster scan

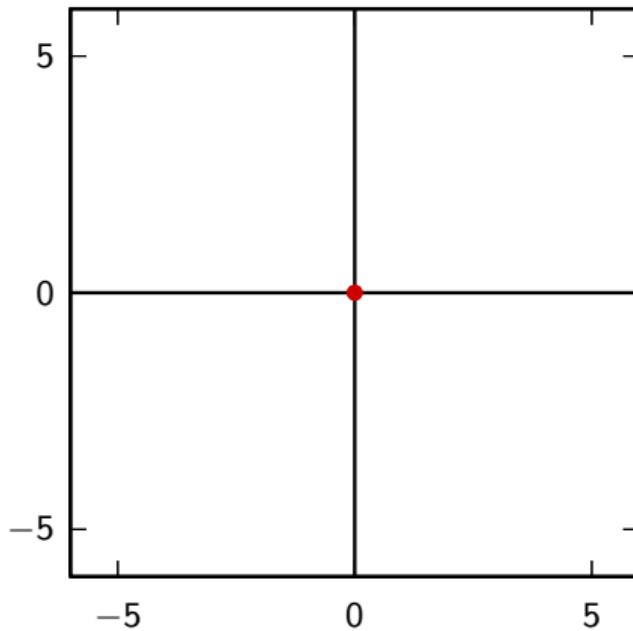


2D vs raster scan



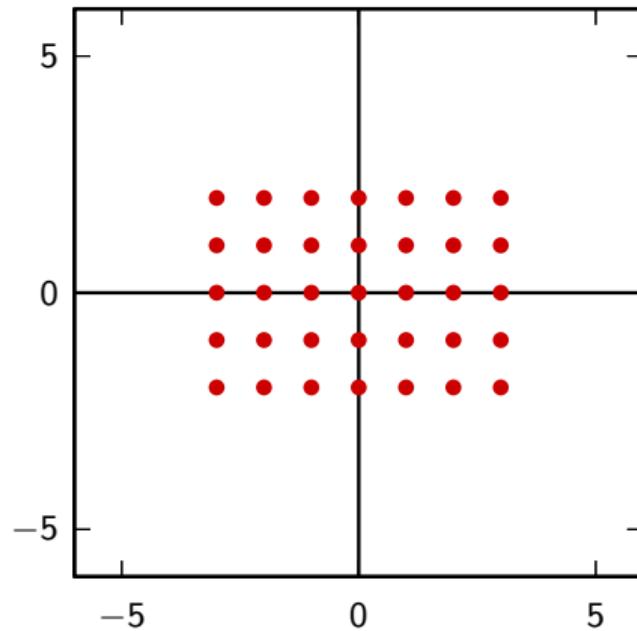
Basic 2D signals: the impulse

$$\delta[n_1, n_2] = \begin{cases} 1 & \text{if } n_1 = n_2 = 0 \\ 0 & \text{otherwise.} \end{cases}$$



Basic 2D signals: the rect

$$\text{rect}\left(\frac{n_1}{2N_1}, \frac{n_2}{2N_2}\right) = \begin{cases} 1 & \text{if } |n_1| < N_1 \\ & \text{and } |n_2| < N_2 \\ 0 & \text{otherwise;} \end{cases}$$



Separability

$$x[n_1, n_2] = x_1[n_1]x_2[n_2]$$

Separable signals

$$\delta[n_1, n_2] = \delta[n_1]\delta[n_2]$$

$$\text{rect}\left(\frac{n_1}{2N_1}, \frac{n_2}{2N_2}\right) = \text{rect}\left(\frac{n_1}{2N_1}\right)\text{rect}\left(\frac{n_2}{2N_2}\right).$$

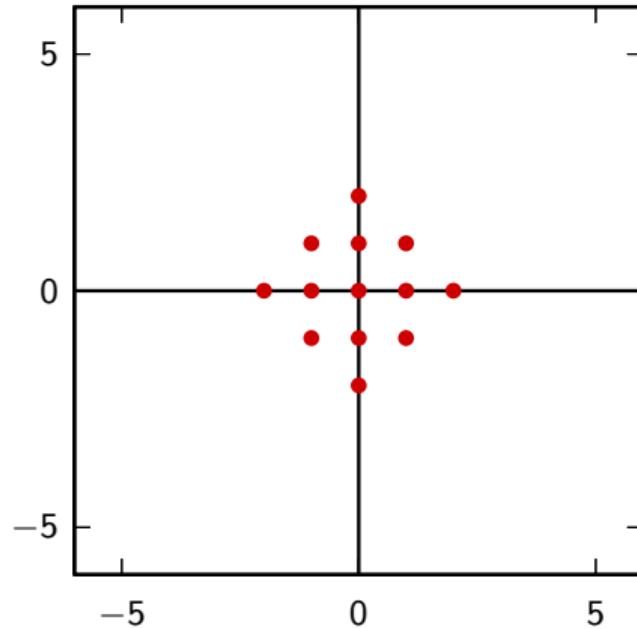
Separable signals

$$\delta[n_1, n_2] = \delta[n_1]\delta[n_2]$$

$$\text{rect}\left(\frac{n_1}{2N_1}, \frac{n_2}{2N_2}\right) = \text{rect}\left(\frac{n_1}{2N_1}\right) \text{rect}\left(\frac{n_2}{2N_2}\right).$$

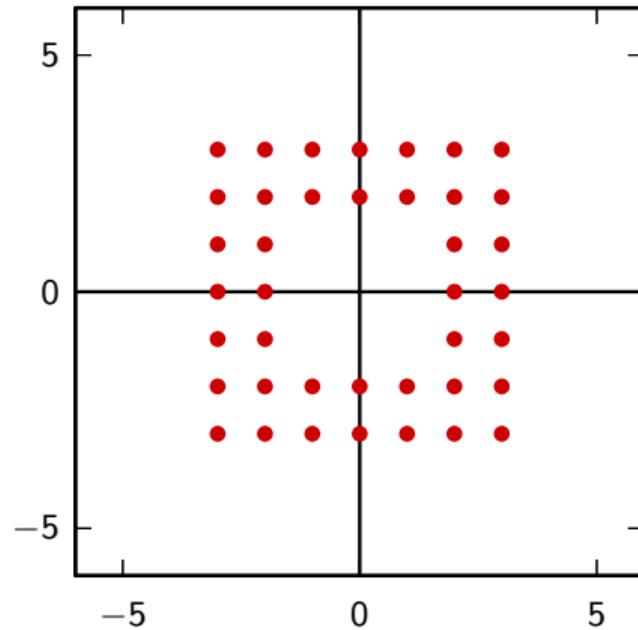
Nonseparable signal

$$x[n_1, n_2] = \begin{cases} 1 & \text{if } |n_1| + |n_2| < N \\ 0 & \text{otherwise} \end{cases}$$



Nonseparable signal

$$x[n_1, n_2] = \text{rect}\left(\frac{n_1}{2N_1}, \frac{n_2}{2N_2}\right) - \text{rect}\left(\frac{n_1}{2M_1}, \frac{n_2}{2M_2}\right)$$



2D convolution

$$x[n_1, n_2] * h[n_1, n_2] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x[k_1, k_2] h[n_1 - k_1, n_2 - k_2]$$

2D convolution for separable signals

If $h[n_1, n_2] = h_1[n_1]h_2[n_2]$:

$$\begin{aligned}x[n_1, n_2] * h[n_1, n_2] &= \sum_{k_1=-\infty}^{\infty} h_1[n_1 - k_1] \sum_{k_2=-\infty}^{\infty} x[k_1, k_2]h_2[n_2 - k_2] \\&= h_1[n_1] * (h_2[n_2] * x[n_1, n_2]).\end{aligned}$$

2D convolution for separable signals

If $h[n_1, n_2]$ is an $M_1 \times M_2$ finite-support signal:

- ▶ non-separable convolution: $M_1 M_2$ operations per output sample
- ▶ separable convolution: $M_1 + M_2$ operations per output sample!

affine transforms

Affine transforms

mapping $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ that reshapes the coordinate system (in continuous space):

$$\begin{bmatrix} t'_1 \\ t'_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} - \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

$$\begin{bmatrix} t'_1 \\ t'_2 \end{bmatrix} = \mathbf{A} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} - \mathbf{d}$$

Affine transforms

mapping $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ that reshapes the coordinate system (in continuous space):

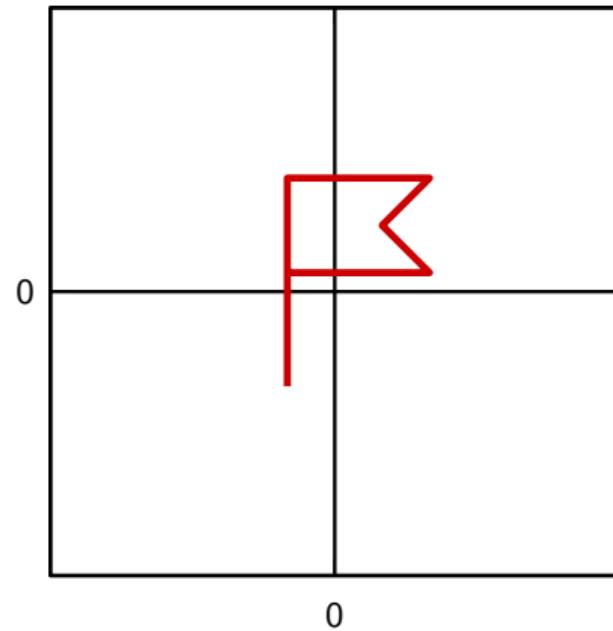
$$\begin{bmatrix} t'_1 \\ t'_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} - \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

$$\begin{bmatrix} t'_1 \\ t'_2 \end{bmatrix} = \mathbf{A} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} - \mathbf{d}$$

Translation

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}$$

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

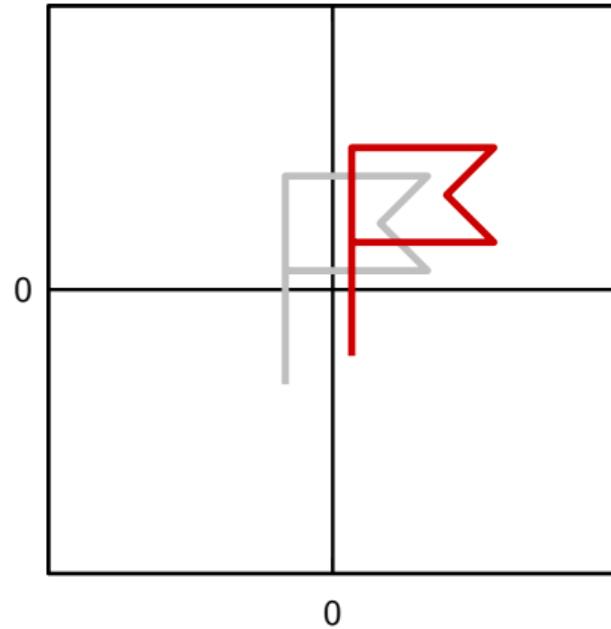


Translation

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}$$

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

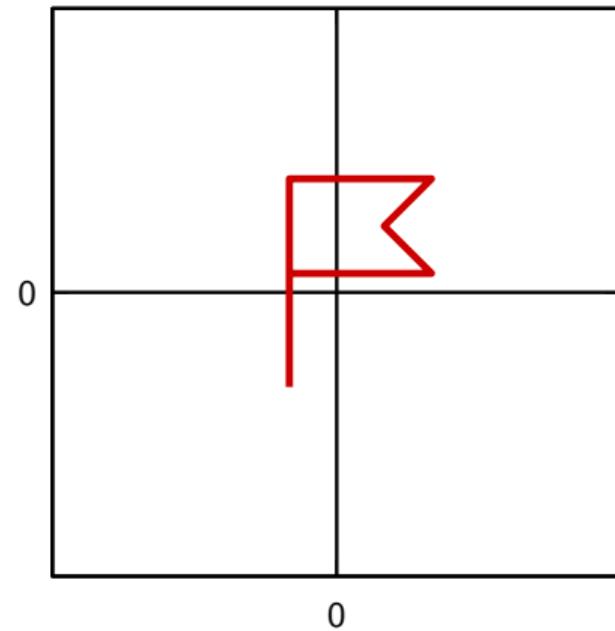
$$d_1 = -0.7, d_2 = -0.3$$



Scaling

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix}$$

$$\mathbf{d} = 0$$



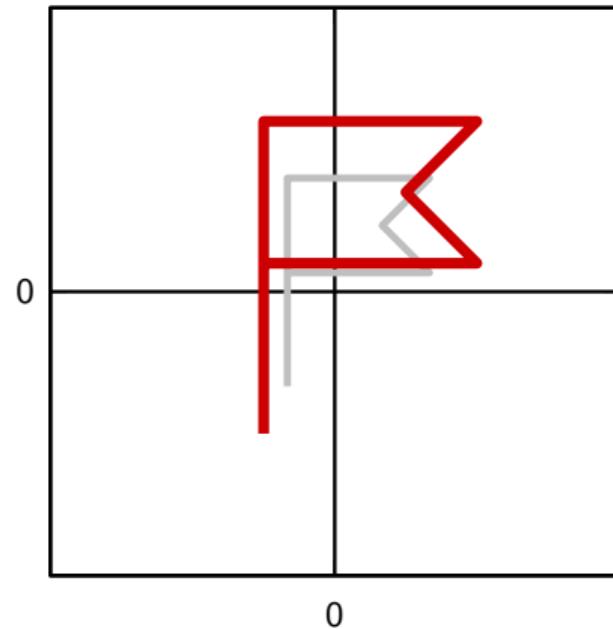
Scaling

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix}$$

$$\mathbf{d} = 0$$

if $a_1 = a_2$ the *aspect ratio* is preserved

$$a_1 = a_2 = 1.5$$

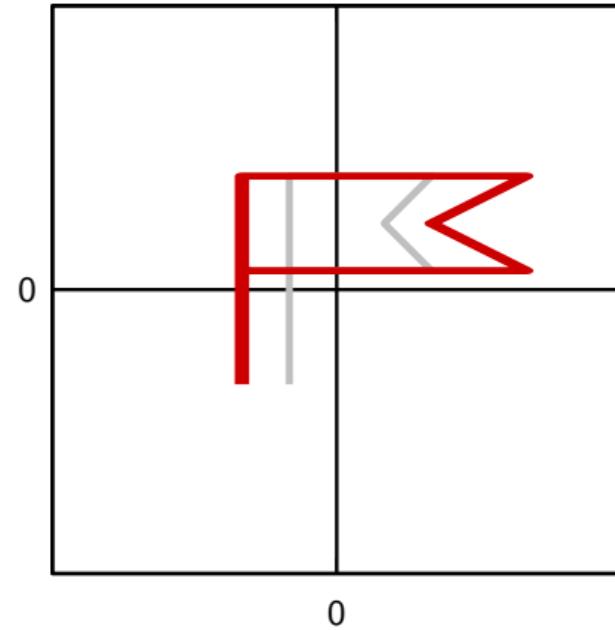


Scaling

$$a_1 = 2, a_2 = 1$$

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix}$$

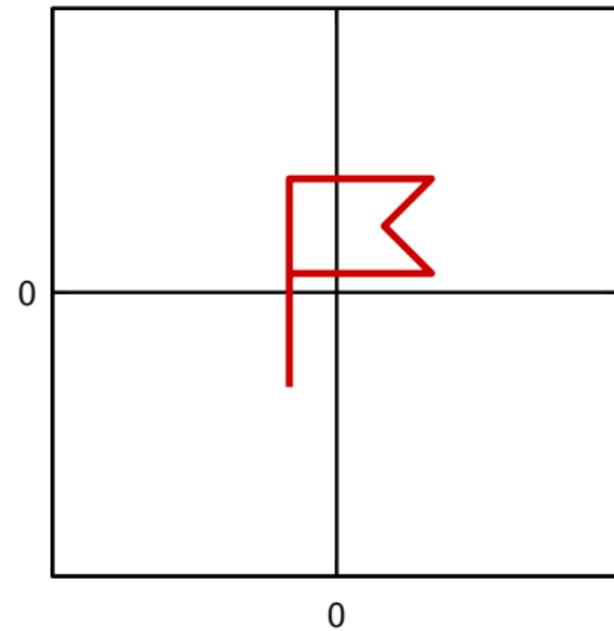
$$\mathbf{d} = 0$$



Rotation

$$\mathbf{A} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

$$\mathbf{d} = 0$$

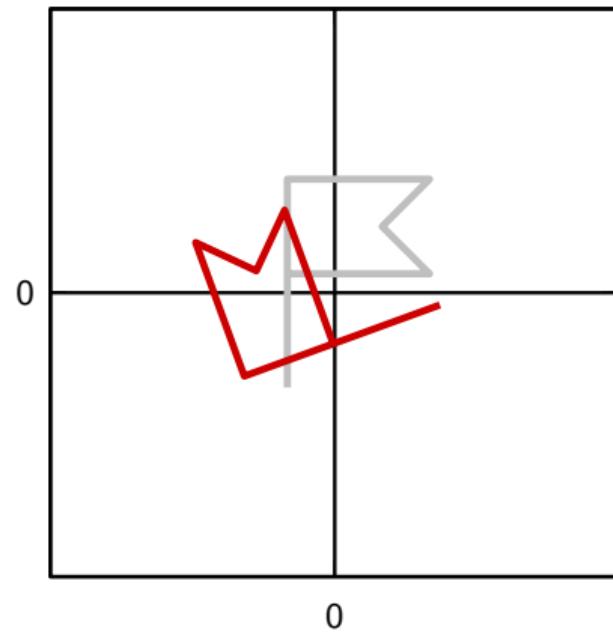


Rotation

$$\mathbf{A} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

$$\mathbf{d} = 0$$

$$\theta = -0.6\pi$$

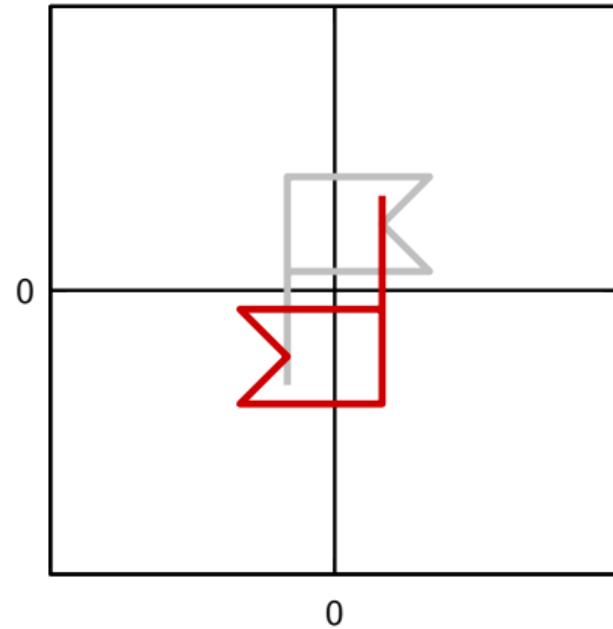


Rotation

$$\mathbf{A} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

$$\mathbf{d} = 0$$

$$\theta = \pi$$



Flips

Horizontal:

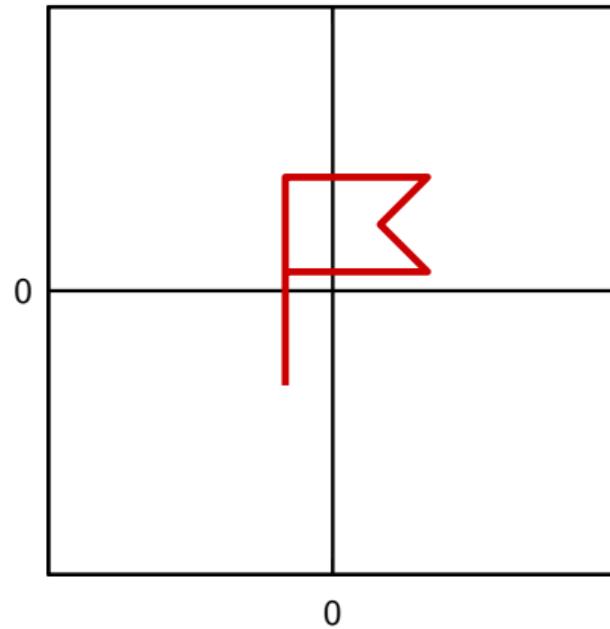
$$\mathbf{A} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{d} = 0$$

Vertical:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\mathbf{d} = 0$$



Flips

Horizontal:

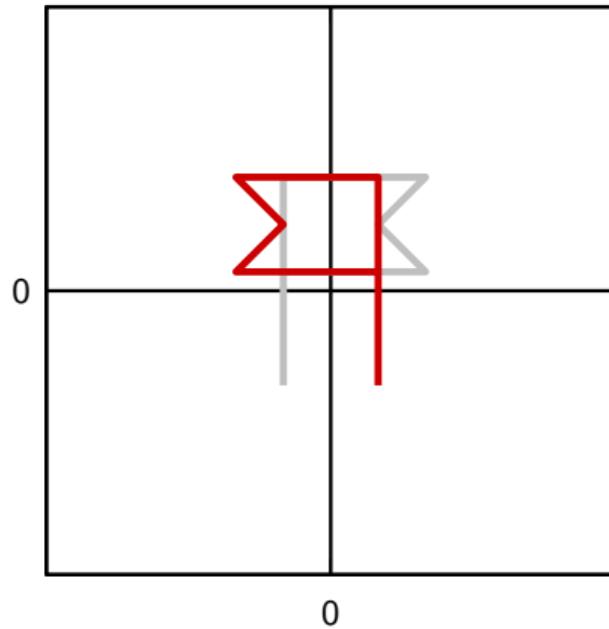
$$\mathbf{A} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{d} = 0$$

Vertical:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\mathbf{d} = 0$$



Shear

Horizontal:

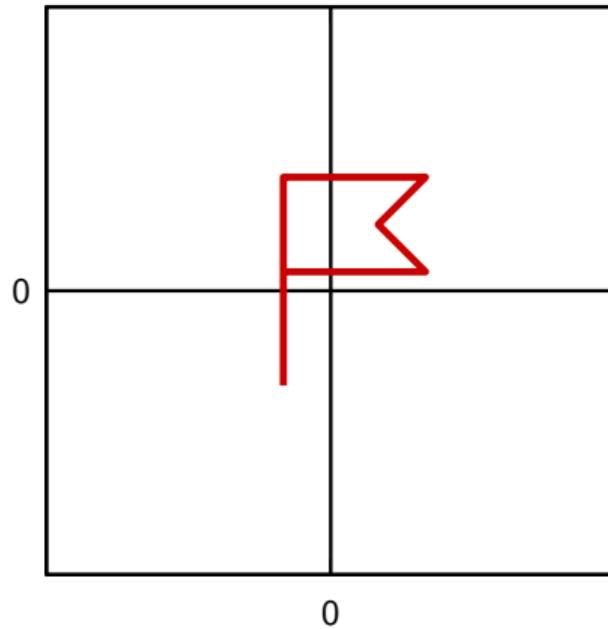
$$\mathbf{A} = \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{d} = 0$$

Vertical:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix}$$

$$\mathbf{d} = 0$$



Shear

Horizontal:

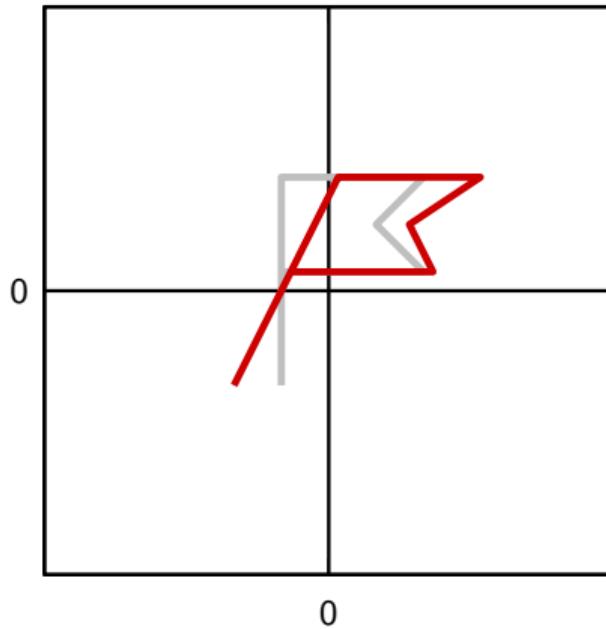
$$\mathbf{A} = \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{d} = 0$$

Vertical:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix}$$

$$\mathbf{d} = 0$$



Affine transforms in discrete-space

$$\begin{bmatrix} t'_1 \\ t'_2 \end{bmatrix} = \mathbf{A} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} - \mathbf{d} \in \mathbb{R}^2 \neq \mathbb{Z}^2$$

Solution for images

- ▶ take each *output point* $y[m_1, m_2]$
- ▶ apply the *inverse* transform to $[m_1, m_2]$ and find the *source* point's coordinates:

$$\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} m_1 + d_1 \\ m_2 + d_2 \end{bmatrix};$$

- ▶ if source point not on source grid, write

$$(t_1, t_2) = (\eta_1 + \tau_1, \eta_2 + \tau_2), \quad \eta_{1,2} \in \mathbb{Z}, \quad 0 \leq \tau_{1,2} < 1$$

and interpolate from the surrounding original grid points

Solution for images

- ▶ take each *output point* $y[m_1, m_2]$
- ▶ apply the *inverse* transform to $[m_1, m_2]$ and find the *source* point's coordinates:

$$\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} m_1 + d_1 \\ m_2 + d_2 \end{bmatrix};$$

- ▶ if source point not on source grid, write

$$(t_1, t_2) = (\eta_1 + \tau_1, \eta_2 + \tau_2), \quad \eta_{1,2} \in \mathbb{Z}, \quad 0 \leq \tau_{1,2} < 1$$

and interpolate from the surrounding original grid points

Solution for images

- ▶ take each *output point* $y[m_1, m_2]$
- ▶ apply the *inverse* transform to $[m_1, m_2]$ and find the *source* point's coordinates:

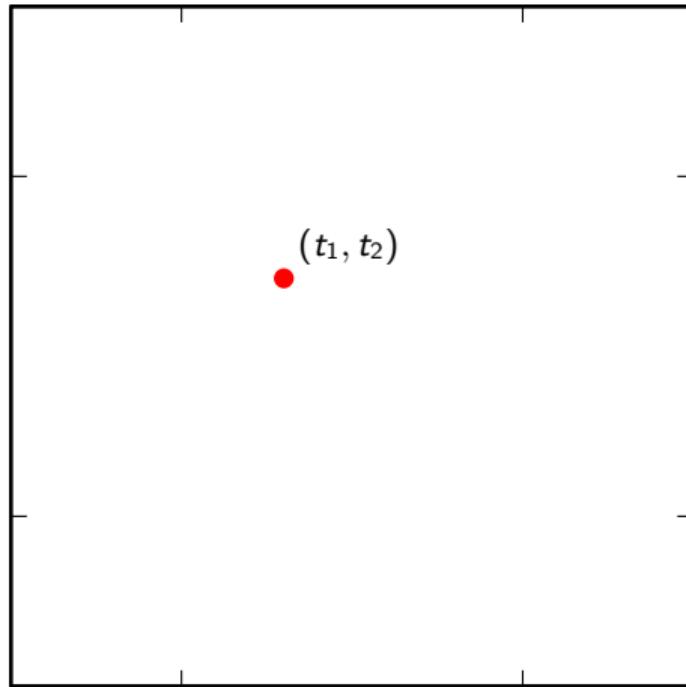
$$\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} m_1 + d_1 \\ m_2 + d_2 \end{bmatrix};$$

- ▶ if source point not on source grid, write

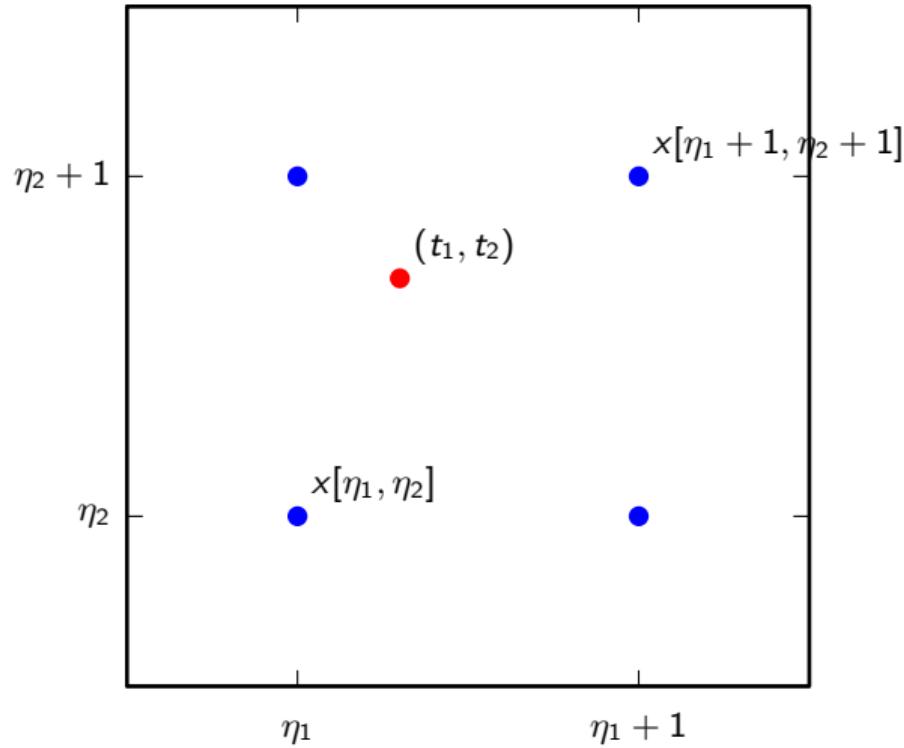
$$(t_1, t_2) = (\eta_1 + \tau_1, \eta_2 + \tau_2), \quad \eta_{1,2} \in \mathbb{Z}, \quad 0 \leq \tau_{1,2} < 1$$

and interpolate from the surrounding original grid points

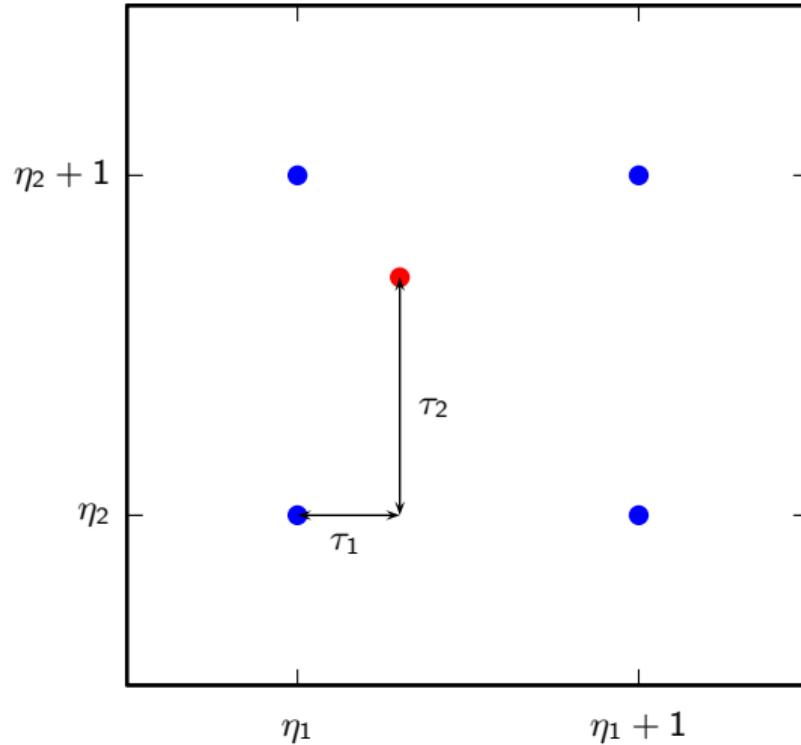
Bilinear Interpolation



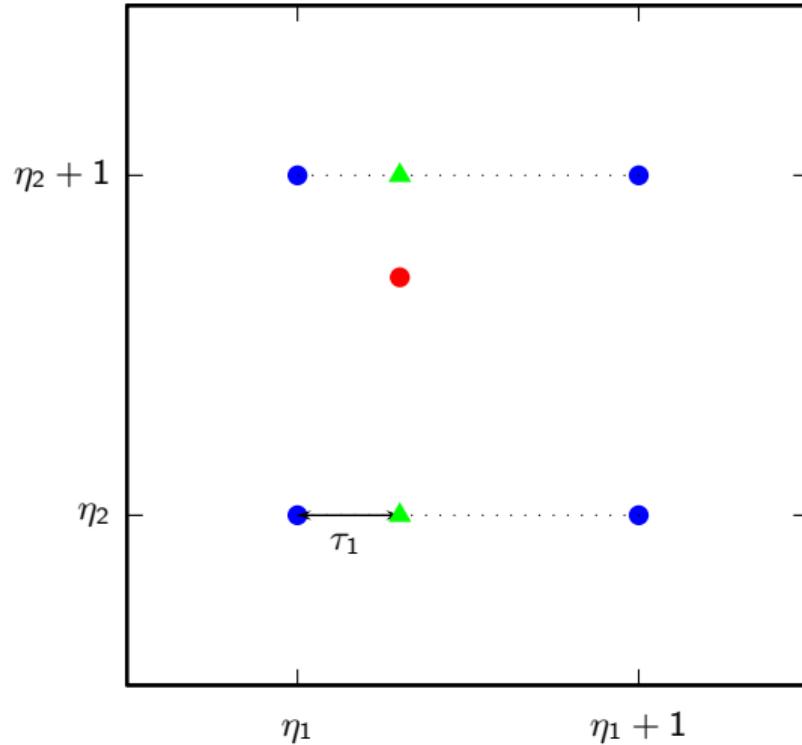
Bilinear Interpolation



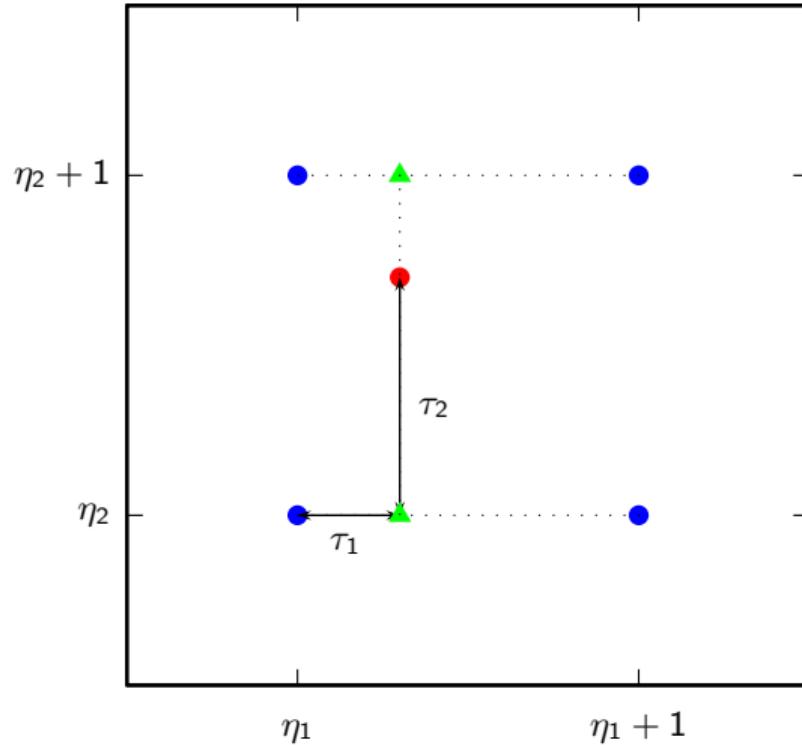
Bilinear Interpolation



Bilinear Interpolation



Bilinear Interpolation



Bilinear Interpolation

If we use a first-order interpolator:

$$\begin{aligned}y[m_1, m_2] &= (1 - \tau_1)(1 - \tau_2)x[\eta_1, \eta_2] + \tau_1(1 - \tau_2)x[\eta_1 + 1, \eta_2] \\&\quad + (1 - \tau_1)\tau_2x[\eta_1, \eta_2 + 1] + \tau_1\tau_2x[\eta_1 + 1, \eta_2 + 1]\end{aligned}$$

Shearing



2D Fourier Analysis

2D DFT

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j \frac{2\pi}{N_1} n_1 k_1} e^{-j \frac{2\pi}{N_2} n_2 k_2}$$

$$x[n_1, n_2] = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X[k_1, k_2] e^{j \frac{2\pi}{N_1} n_1 k_1} e^{j \frac{2\pi}{N_2} n_2 k_2}$$

2D DFT

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j \frac{2\pi}{N_1} n_1 k_1} e^{-j \frac{2\pi}{N_2} n_2 k_2}$$

$$x[n_1, n_2] = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X[k_1, k_2] e^{j \frac{2\pi}{N_1} n_1 k_1} e^{j \frac{2\pi}{N_2} n_2 k_2}$$

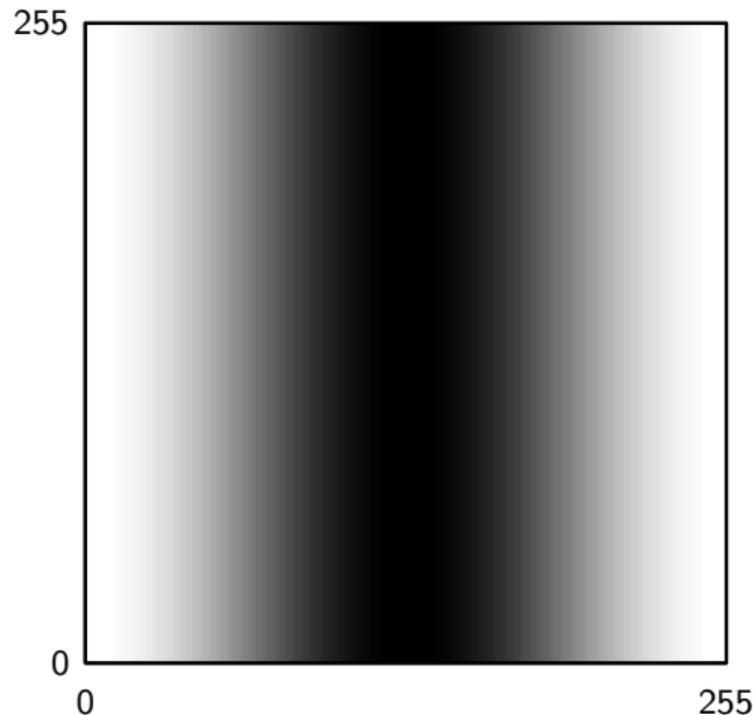
2D-DFT Basis Vectors

There are $N_1 N_2$ orthogonal basis vectors for an $N_1 \times N_2$ image:

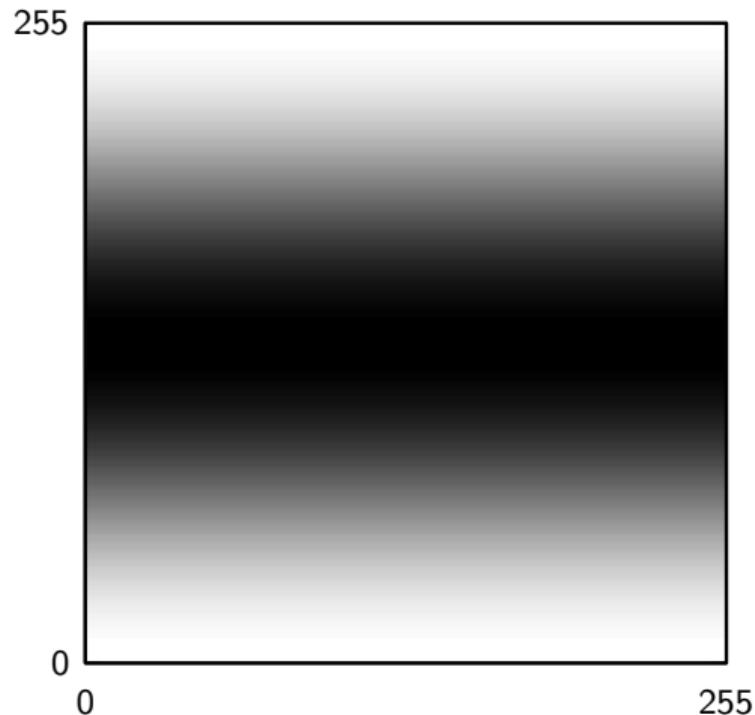
$$w_{k_1, k_2}[n_1, n_2] = e^{j\frac{2\pi}{N_1} n_1 k_1} e^{j\frac{2\pi}{N_2} n_2 k_2}$$

for $n_1, k_1 = 0, 1, \dots, N_1 - 1$ and $n_2, k_2 = 0, 1, \dots, N_2 - 1$

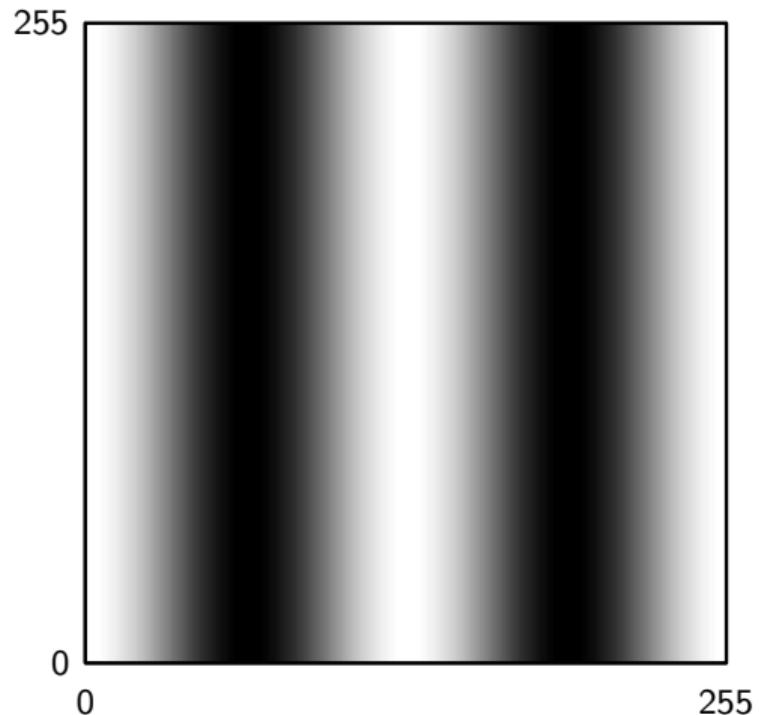
2D-DFT basis vectors for $k_1 = 1, k_2 = 0$ (real part)



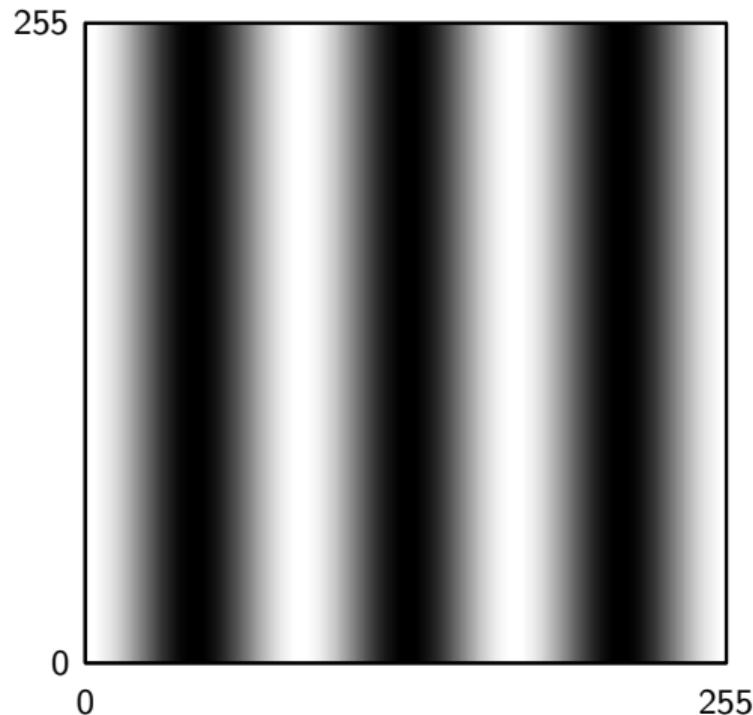
2D-DFT basis vectors for $k_1 = 0, k_2 = 1$ (real part)



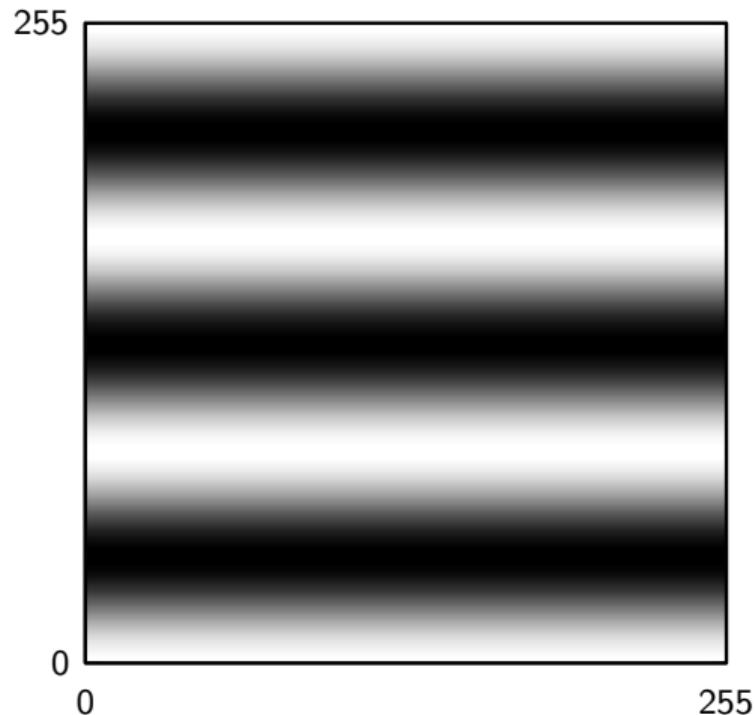
2D-DFT basis vectors for $k_1 = 2, k_2 = 0$ (real part)



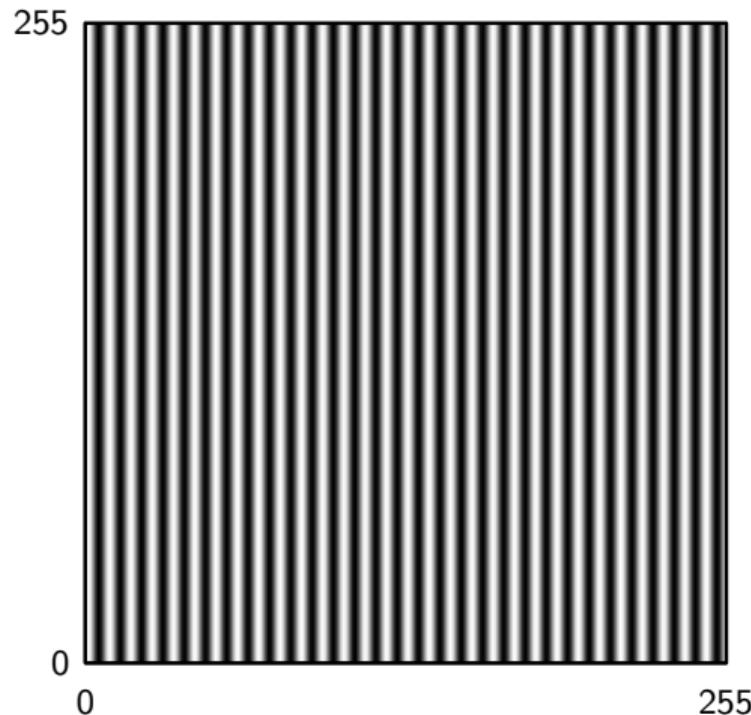
2D-DFT basis vectors for $k_1 = 3, k_2 = 0$ (real part)



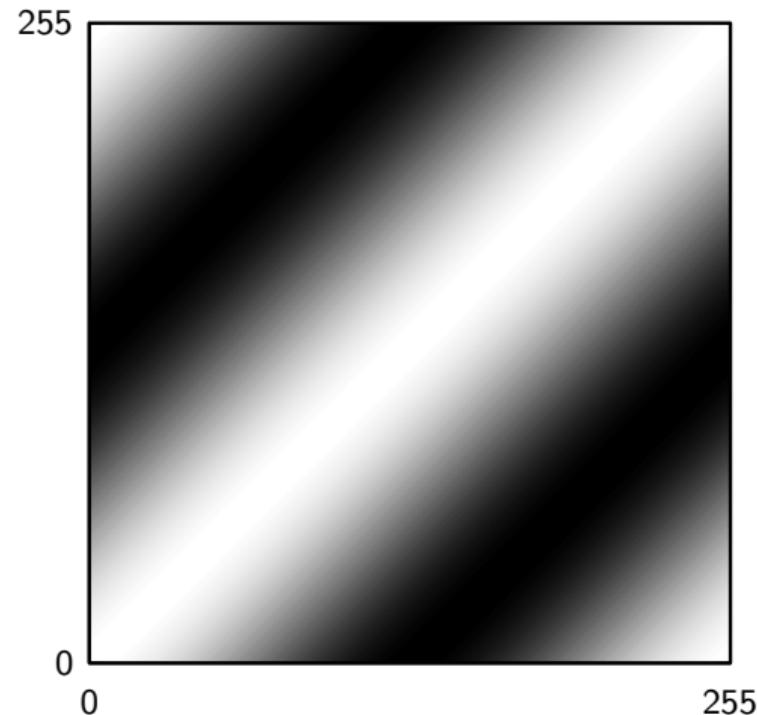
2D-DFT basis vectors for $k_1 = 0, k_2 = 3$ (real part)



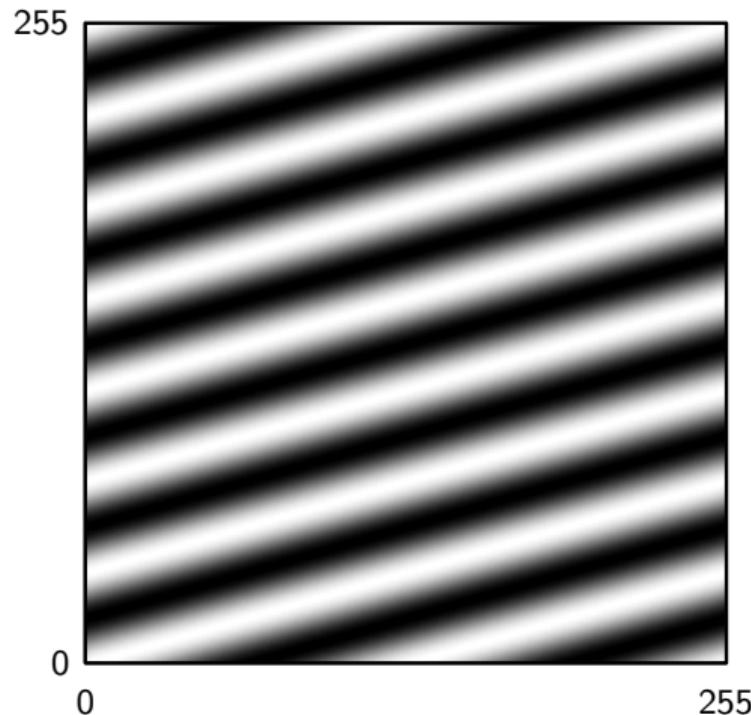
2D-DFT basis vectors for $k_1 = 30, k_2 = 0$ (real part)



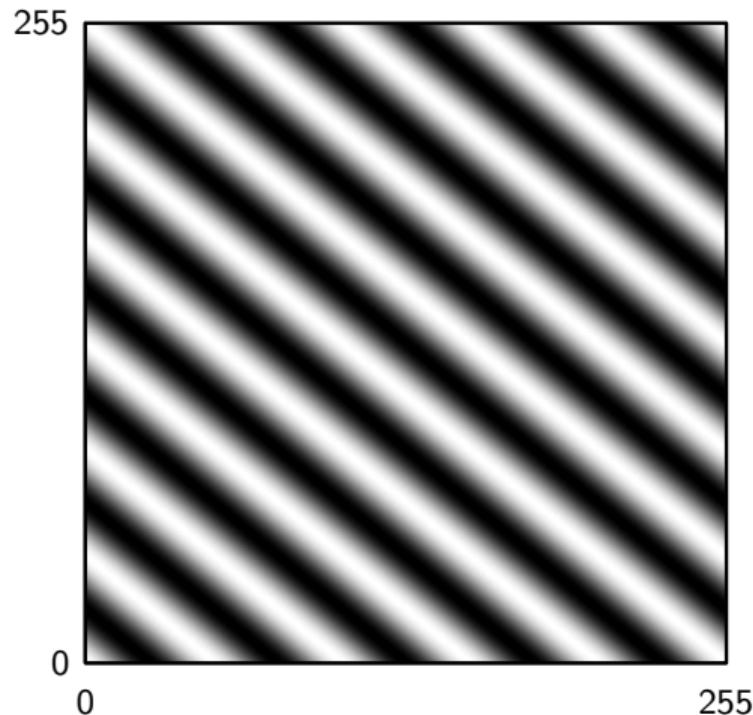
2D-DFT basis vectors for $k_1 = 1, k_2 = 1$ (real part)



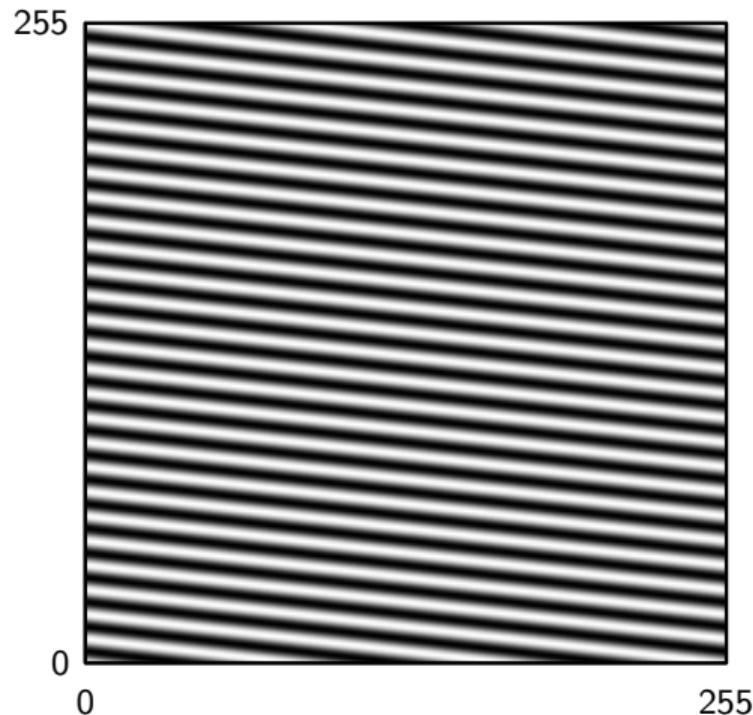
2D-DFT basis vectors for $k_1 = 2, k_2 = 7$ (real part)



2D-DFT basis vectors for $k_1 = 5, k_2 = 250$ (real part)



2D-DFT basis vectors for $k_1 = 3, k_2 = 230$ (real part)



2D DFT

2D-DFT basis functions are separable, and so is the 2D-DFT:

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_1} n_1 k_1} e^{-j\frac{2\pi}{N_2} n_2 k_2}$$

- ▶ 1D-DFT along n_2 (the columns)
- ▶ 1D-DFT along n_1 (the rows)

2D DFT

2D-DFT basis functions are separable, and so is the 2D-DFT:

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j \frac{2\pi}{N_2} n_2 k_2} \right] e^{-j \frac{2\pi}{N_1} n_1 k_1}$$

- ▶ 1D-DFT along n_2 (the columns)
- ▶ 1D-DFT along n_1 (the rows)

2D DFT

2D-DFT basis functions are separable, and so is the 2D-DFT:

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j \frac{2\pi}{N_2} n_2 k_2} \right] e^{-j \frac{2\pi}{N_1} n_1 k_1}$$

- ▶ 1D-DFT along n_2 (the columns)
- ▶ 1D-DFT along n_1 (the rows)



2D DFT

2D-DFT basis functions are separable, and so is the 2D-DFT:

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_2} n_2 k_2} \right] e^{-j\frac{2\pi}{N_1} n_1 k_1}$$

- ▶ 1D-DFT along n_2 (the columns)
- ▶ 1D-DFT along n_1 (the rows)



2D DFT in matrix form

- ▶ finite-support 2D signal can be written as a matrix \mathbf{x}
- ▶ $N_1 \times N_2$ image is an $N_2 \times N_1$ matrix (n_1 spans the columns, n_2 spans the rows)
- ▶ recall also the $N \times N$ DFT matrix:

$$\mathbf{W}_N = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^1 & W_N^2 & W_N^3 & \dots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & W_N^6 & \dots & W_N^{2(N-1)} \\ & & & \ddots & & \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & W_N^{3(N-1)} & \dots & W_N^{(N-1)^2} \end{bmatrix}$$

2D DFT in matrix form

- ▶ finite-support 2D signal can be written as a matrix \mathbf{x}
- ▶ $N_1 \times N_2$ image is an $N_2 \times N_1$ matrix (n_1 spans the columns, n_2 spans the rows)
- ▶ recall also the $N \times N$ DFT matrix:

$$\mathbf{W}_N = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^1 & W_N^2 & W_N^3 & \dots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & W_N^6 & \dots & W_N^{2(N-1)} \\ & & & \ddots & & \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & W_N^{3(N-1)} & \dots & W_N^{(N-1)^2} \end{bmatrix}$$

2D DFT in matrix form

- ▶ finite-support 2D signal can be written as a matrix \mathbf{x}
- ▶ $N_1 \times N_2$ image is an $N_2 \times N_1$ matrix (n_1 spans the columns, n_2 spans the rows)
- ▶ recall also the $N \times N$ DFT matrix:

$$\mathbf{W}_N = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^1 & W_N^2 & W_N^3 & \dots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & W_N^6 & \dots & W_N^{2(N-1)} \\ & & & \ddots & & \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & W_N^{3(N-1)} & \dots & W_N^{(N-1)^2} \end{bmatrix}$$

2D DFT in matrix form

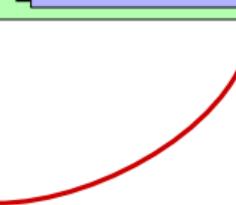
$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_2} n_2 k_2} \right] e^{-j\frac{2\pi}{N_1} n_1 k_1}$$

2D DFT in matrix form

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_2} n_2 k_2} \right] e^{-j\frac{2\pi}{N_1} n_1 k_1}$$

$$\mathbf{V} = \mathbf{W}_{N_2} \mathbf{x}$$

$$\mathbf{V} \in \mathbb{C}^{N_2 \times N_1}$$



2D DFT in matrix form

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_2} n_2 k_2} \right] e^{-j\frac{2\pi}{N_1} n_1 k_1}$$
$$\mathbf{V} = \mathbf{W}_{N_2} \mathbf{x}$$
$$\mathbf{V} \in \mathbb{C}^{N_2 \times N_1}$$
$$\mathbf{X} = \mathbf{V} \mathbf{W}_{N_1}$$
$$\mathbf{X} \in \mathbb{C}^{N_2 \times N_1}$$

2D DFT in matrix form

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \left[\sum_{n_2=0}^{N_2-1} x[n_1, n_2] e^{-j\frac{2\pi}{N_2} n_2 k_2} \right] e^{-j\frac{2\pi}{N_1} n_1 k_1}$$

$$\mathbf{V} = \mathbf{W}_{N_2} \mathbf{x}$$

$$\mathbf{V} \in \mathbb{C}^{N_2 \times N_1}$$

$$\mathbf{X} = \mathbf{V} \mathbf{W}_{N_1}$$

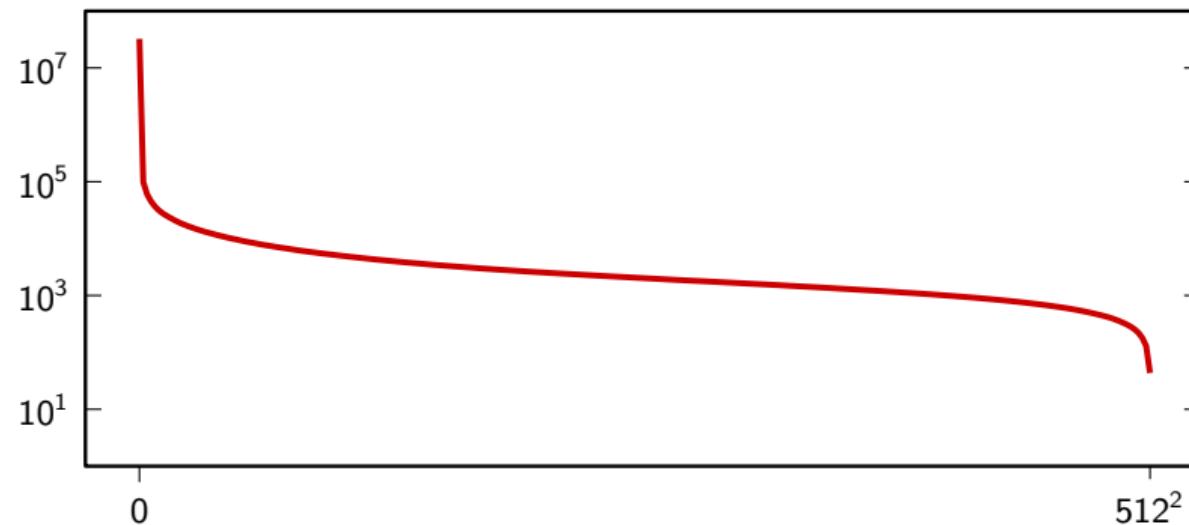
$$\mathbf{X} \in \mathbb{C}^{N_2 \times N_1}$$

$$\mathbf{X} = \mathbf{W}_{N_2} \mathbf{x} \mathbf{W}_{N_1}$$

How does a 2D-DFT look like?

- ▶ try to show the magnitude as an image
- ▶ problem: the range is too big for the grayscale range of paper or screen
- ▶ try to normalize: $|X'[n_1, n_2]| = |X[n_1, n_2]| / \max |X[n_1, n_2]|$
- ▶ but it doesn't work...

DFT coefficients sorted by magnitude



Dealing with HDR images

if the image is high dynamic range we need to compress the levels

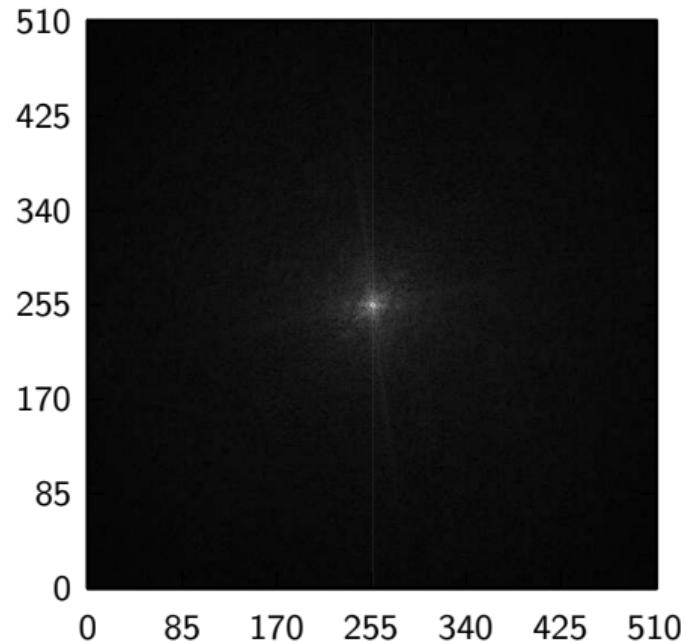
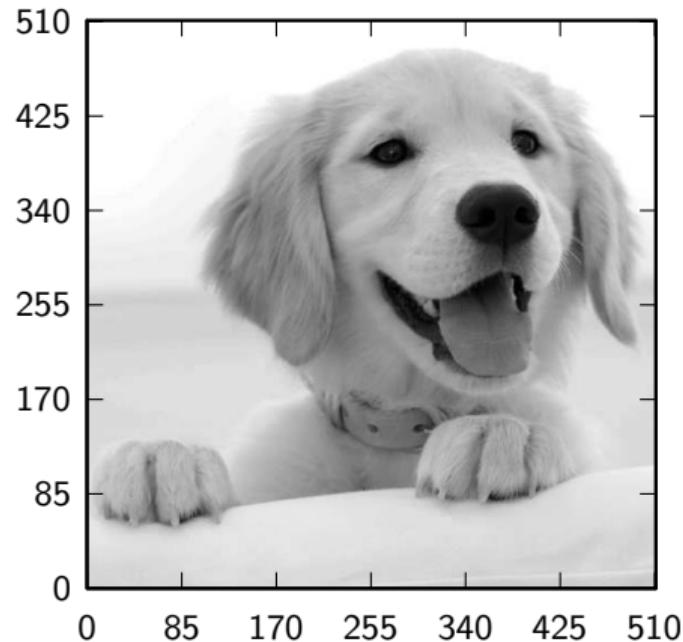
- ▶ remove flagrant outliers (e.g. $X[0,0] = \sum \sum x[n_1, n_2]$)
- ▶ use a nonlinear mapping: e.g. $y = x^{1/3}$ after normalization ($x \leq 1$)

Dealing with HDR images

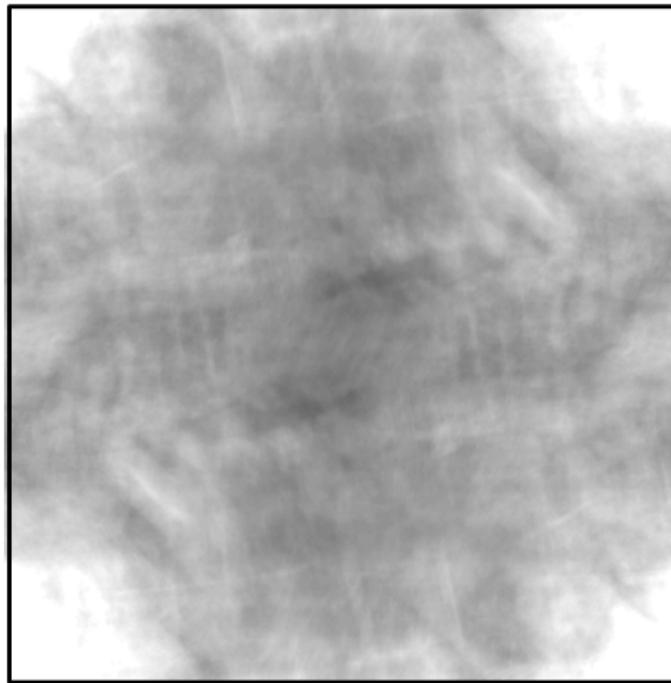
if the image is high dynamic range we need to compress the levels

- ▶ remove flagrant outliers (e.g. $X[0, 0] = \sum \sum x[n_1, n_2]$)
- ▶ use a nonlinear mapping: e.g. $y = x^{1/3}$ after normalization ($x \leq 1$)

How does a 2D-DFT look like?



DFT magnitude doesn't carry much information



DFT phase, on the other hand...



Image frequency analysis

- ▶ most of the information is contained in image's *edges*
- ▶ edges are points of abrupt change in signal's values
- ▶ edges are a “space-domain” feature → not captured by DFT's magnitude
- ▶ phase alignment is important for reproducing edges

image filtering

Overview:

- ▶ Filters for image processing
- ▶ Classification
- ▶ Examples

Overview:

- ▶ Filters for image processing
- ▶ Classification
- ▶ Examples

Overview:

- ▶ Filters for image processing
- ▶ Classification
- ▶ Examples

Analogies with 1D filters

- ▶ linearity
- ▶ *space* invariance
- ▶ impulse response
- ▶ frequency response
- ▶ stability
- ▶ 2D CCDE

The problem with LSI operators

- ▶ interesting images contain lots of *semantics*: different information in different areas
- ▶ space-invariant filters process everything in the same way
- ▶ but we should process things differently
 - edges
 - gradients
 - textures
 - ...

The problem with LSI operators

- ▶ interesting images contain lots of *semantics*: different information in different areas
- ▶ space-invariant filters process everything in the same way
- ▶ but we should process things differently
 - edges
 - gradients
 - textures
 - ...

The problem with LSI operators

- ▶ interesting images contain lots of *semantics*: different information in different areas
- ▶ space-invariant filters process everything in the same way
- ▶ but we should process things differently
 - edges
 - gradients
 - textures
 - ...

The problem with LSI operators

- ▶ interesting images contain lots of *semantics*: different information in different areas
- ▶ space-invariant filters process everything in the same way
- ▶ but we should process things differently
 - edges
 - gradients
 - textures
 - ...

The problem with LSI operators

- ▶ interesting images contain lots of *semantics*: different information in different areas
- ▶ space-invariant filters process everything in the same way
- ▶ but we should process things differently
 - edges
 - gradients
 - textures
 - ...

The problem with LSI operators

- ▶ interesting images contain lots of *semantics*: different information in different areas
- ▶ space-invariant filters process everything in the same way
- ▶ but we should process things differently
 - edges
 - gradients
 - textures
 - ...

The problem with LSI operators

- ▶ interesting images contain lots of *semantics*: different information in different areas
- ▶ space-invariant filters process everything in the same way
- ▶ but we should process things differently
 - edges
 - gradients
 - textures
 - ...

Filter types

- ▶ IIR, FIR
- ▶ causal or noncausal
- ▶ highpass, lowpass, ...
 - lowpass → image smoothing
 - highpass → enhancement, edge detection

Filter types

- ▶ IIR, FIR
- ▶ causal or noncausal
- ▶ highpass, lowpass, ...
 - lowpass → image smoothing
 - highpass → enhancement, edge detection

Filter types

- ▶ IIR, FIR
- ▶ causal or noncausal
- ▶ highpass, lowpass, ...
 - lowpass → image smoothing
 - highpass → enhancement, edge detection

Filter types

- ▶ IIR, FIR
- ▶ causal or noncausal
- ▶ highpass, lowpass, ...
 - lowpass → image smoothing
 - highpass → enhancement, edge detection

Filter types

- ▶ IIR, FIR
- ▶ causal or noncausal
- ▶ highpass, lowpass, ...
 - lowpass → image smoothing
 - highpass → enhancement, edge detection

The problems with 2D IIRs

- ▶ nonlinear phase (edges!)
- ▶ border effects
- ▶ stability: the fundamental theorem of algebra doesn't hold in multiple dimensions!
- ▶ computability

The problems with 2D IIRs

- ▶ nonlinear phase (edges!)
- ▶ border effects
- ▶ stability: the fundamental theorem of algebra doesn't hold in multiple dimensions!
- ▶ computability

The problems with 2D IIRs

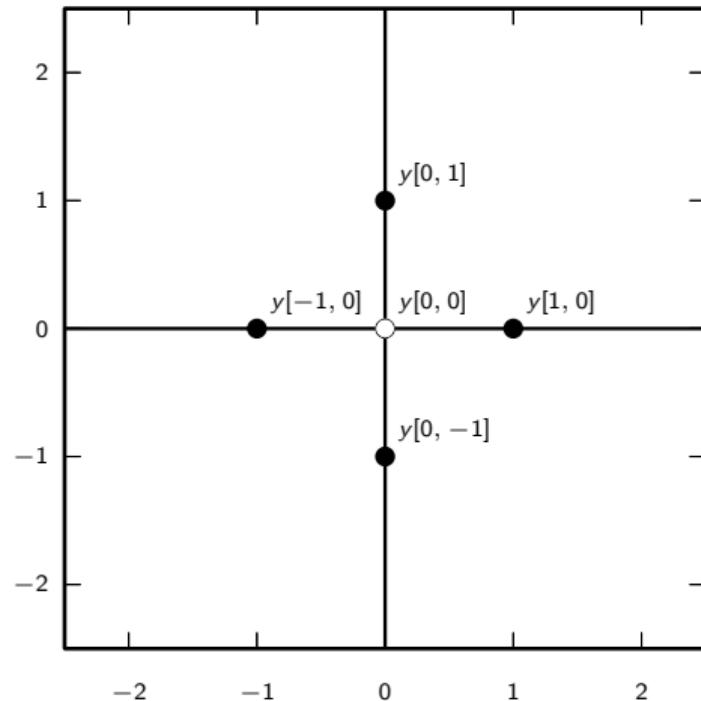
- ▶ nonlinear phase (edges!)
- ▶ border effects
- ▶ stability: the fundamental theorem of algebra doesn't hold in multiple dimensions!
- ▶ computability

The problems with 2D IIRs

- ▶ nonlinear phase (edges!)
- ▶ border effects
- ▶ stability: the fundamental theorem of algebra doesn't hold in multiple dimensions!
- ▶ computability

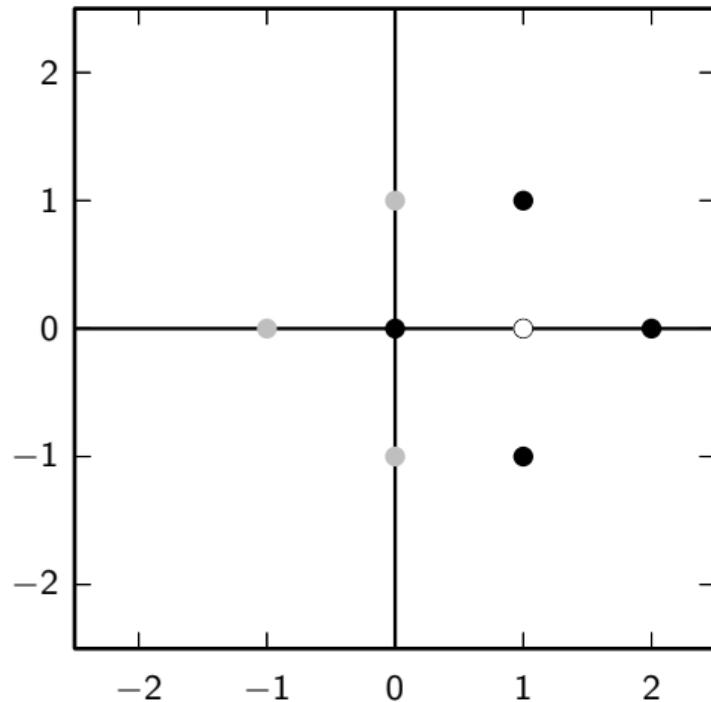
A noncomputable CCDE

$$y[n_1, n_2] = a_0 y[n_1 + 1, n_2] + a_1 y[n_1, n_2 - 1] + a_2 y[n_1 - 1, n_2] + a_3 y[n_1, n_2 + 1] + x[n_1, n_2];$$



A noncomputable CCDE

$$y[n_1, n_2] = a_0 y[n_1 + 1, n_2] + a_1 y[n_1, n_2 - 1] + a_2 y[n_1 - 1, n_2] + a_3 y[n_1, n_2 + 1] + x[n_1, n_2];$$



Practical FIR filters

- ▶ generally zero centered (causality not an issue) \Rightarrow odd number of taps in both directions
- ▶ per-sample complexity:
 - $M_1 M_2$ for nonseparable impulse responses
 - $M_1 + M_2$ for separable impulse responses
- ▶ obviously always stable

Practical FIR filters

- ▶ generally zero centered (causality not an issue) \Rightarrow
odd number of taps in both directions
- ▶ per-sample complexity:
 - $M_1 M_2$ for nonseparable impulse responses
 - $M_1 + M_2$ for separable impulse responses
- ▶ obviously always stable

Practical FIR filters

- ▶ generally zero centered (causality not an issue) \Rightarrow
odd number of taps in both directions
- ▶ per-sample complexity:
 - $M_1 M_2$ for nonseparable impulse responses
 - $M_1 + M_2$ for separable impulse responses
- ▶ obviously always stable

Practical FIR filters

- ▶ generally zero centered (causality not an issue) \Rightarrow
odd number of taps in both directions
- ▶ per-sample complexity:
 - $M_1 M_2$ for nonseparable impulse responses
 - $M_1 + M_2$ for separable impulse responses
- ▶ obviously always stable

Practical FIR filters

- ▶ generally zero centered (causality not an issue) \Rightarrow
odd number of taps in both directions
- ▶ per-sample complexity:
 - $M_1 M_2$ for nonseparable impulse responses
 - $M_1 + M_2$ for separable impulse responses
- ▶ obviously always stable

Moving Average

$$y[n_1, n_2] = \frac{1}{(2N+1)^2} \sum_{k_1=-N}^N \sum_{k_2=-N}^N x[n_1 - k_1, n_2 - k_2]$$

$$h[n_1, n_2] = \frac{1}{(2N+1)^2} \text{rect}\left(\frac{n_1}{2N}, \frac{n_2}{2N}\right)$$

Moving Average

$$y[n_1, n_2] = \frac{1}{(2N+1)^2} \sum_{k_1=-N}^N \sum_{k_2=-N}^N x[n_1 - k_1, n_2 - k_2]$$

$$h[n_1, n_2] = \frac{1}{(2N+1)^2} \text{rect}\left(\frac{n_1}{2N}, \frac{n_2}{2N}\right)$$

Moving Average

$$h[n_1, n_2] = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Moving Average



original



11×11 MA

Moving Average



original



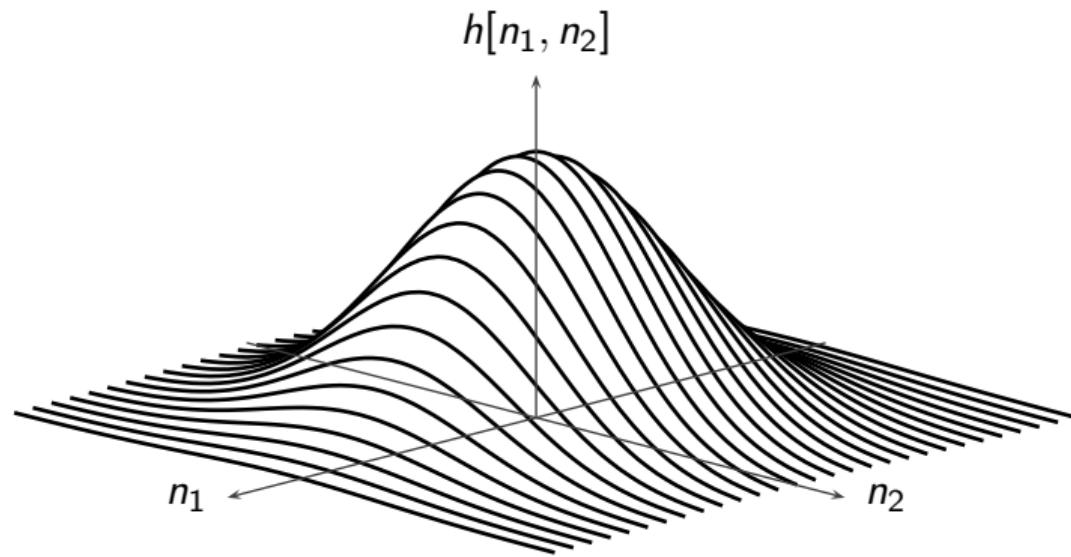
51×51 MA

Gaussian Blur

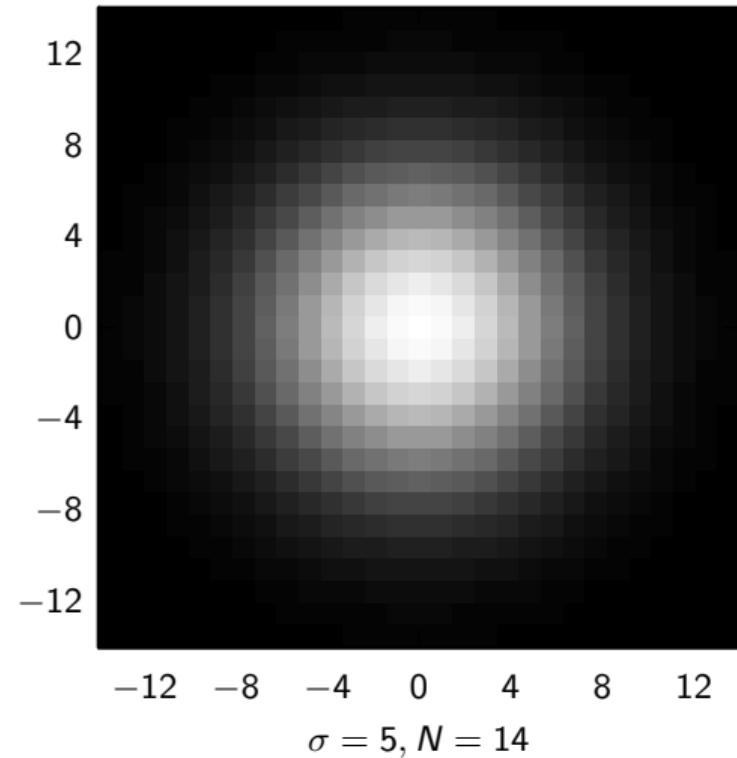
$$h[n_1, n_2] = \frac{1}{2\pi\sigma^2} e^{-\frac{n_1^2+n_2^2}{2\sigma^2}}, \quad |n_1, n_2| < N$$

with $N \approx 3\sigma$

Gaussian Blur



Gaussian Blur



Gaussian Blur



original



$\sigma = 1.8, 11 \times 11$ blur

Gaussian Blur



original



$\sigma = 8.7, 51 \times 51$ blur

Gaussian blur more “photographic” than moving average



11×11 MA



$\sigma = 1.8, 11 \times 11$ blur

Gaussian blur more “photographic” than moving average



51×51 MA

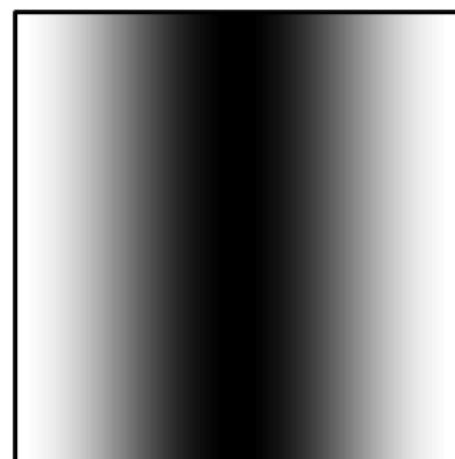
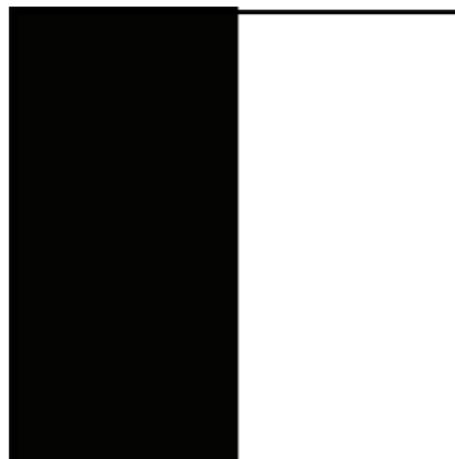


$\sigma = 8.7, 51 \times 51$ blur

Filters for edge detection

What is an edge? Very complicated but, simplifying:

- ▶ points of “discontinuity” in intensity
- ▶ points of inflection in intensity



Sobel filter

Goal: find points where derivative is large.

$$\nabla f(t_1, t_2) = \begin{bmatrix} \frac{\partial f}{\partial t_1} & \frac{\partial f}{\partial t_2} \end{bmatrix}^T$$

Sobel filter

approximating the first derivative on the discrete grid in a circularly symmetric way:

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x & x_5 \\ x_6 & x_7 & x_8 \end{bmatrix}$$

$$x' \approx \sum_{i=1}^8 \frac{x - x_i}{d(x, x_i)} x_i$$

Sobel filter

approximating the first derivative on the discrete grid in a circularly symmetric way:

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x & x_5 \\ x_6 & x_7 & x_8 \end{bmatrix}$$

$$x' \approx \sum_{i=1}^8 \frac{x - x_i}{d(x, x_i)} x_i$$

Sobel filter

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x & x_5 \\ x_6 & x_7 & x_8 \end{bmatrix}$$

$$d(x, x_i) = \begin{cases} 2 & \text{for } i = 2, 4, 5, 7 \\ 4 & \text{for } i = 1, 3, 6, 8 \end{cases}$$

$$\mathbf{x}_1 = [-1 \ 1]^T, \quad \mathbf{x}_2 = [0 \ 1]^T, \quad \dots, \quad \mathbf{x}_8 = [1 \ -1]^T$$

Sobel filter

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x & x_5 \\ x_6 & x_7 & x_8 \end{bmatrix}$$

$$4\nabla x \approx \begin{bmatrix} x_3 - x_1 + 2(x_4 - x_8) + x_5 - x_7 \\ x_7 - x_1 + 2(x_6 - x_2) + x_5 - x_3 \end{bmatrix}$$

Sobel filter

approximation of the first derivative in the horizontal direction:

$$s_h[n_1, n_2] = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

approximation of the first derivative in the vertical direction:

$$s_v[n_1, n_2] = \begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel filter

approximation of the first derivative in the horizontal direction:

$$s_h[n_1, n_2] = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

approximation of the first derivative in the vertical direction:

$$s_v[n_1, n_2] = \begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel filter

filter is separable, e.g.:

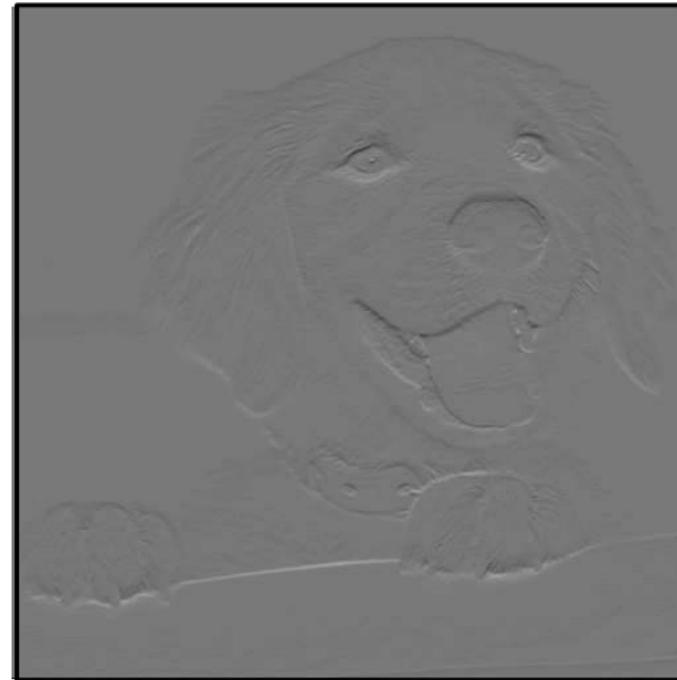
$$s_h[n_1, n_2] = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} [-1 \quad 0 \quad 1]$$

horizontal gradient = vertical averaging followed by horizontal differentiation

Sobel filter



horizontal Sobel filter



vertical Sobel filter

Sobel operator

approximation for the square magnitude of the gradient:

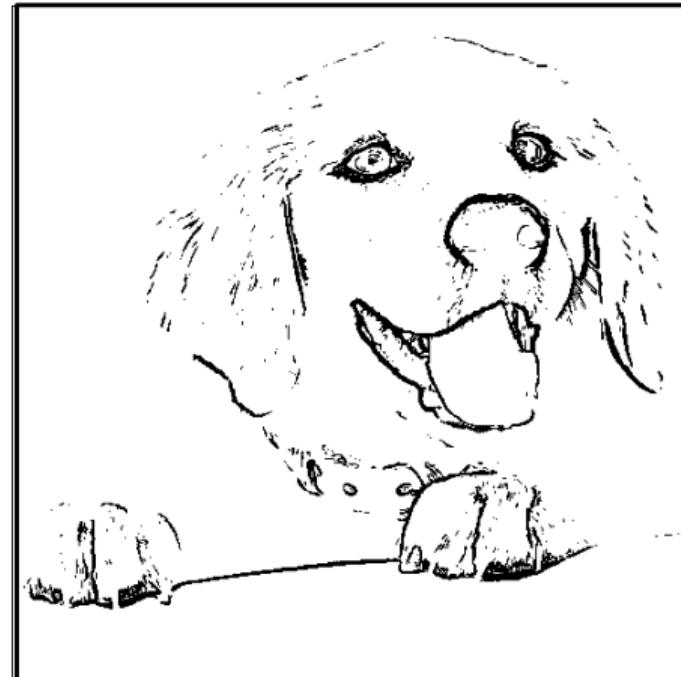
$$|\nabla x[n_1, n_2]|^2 = |s_h[n_1, n_2] * x[n_1, n_2]|^2 + |s_v[n_1, n_2] * x[n_1, n_2]|^2$$

("operator" because it's nonlinear)

Gradient approximation for edge detection



Sobel operator



thresholded Sobel operator

Laplacian operator

Laplacian of a function in continuous-space:

$$\Delta f(t_1, t_2) = \frac{\partial^2 f}{\partial t_1^2} + \frac{\partial^2 f}{\partial t_2^2}$$

Laplacian operator

approximating the Laplacian; start with a Taylor expansion

$$f(t + \tau) = \sum_{n=0}^{\infty} \frac{f^{(n)}(t)}{n!} \tau^n$$

and compute the expansion in $(t + \tau)$ and $(t - \tau)$:

$$f(t + \tau) = f(t) + f'(t)\tau + \frac{1}{2}f''(t)\tau^2$$

$$f(t - \tau) = f(t) - f'(t)\tau + \frac{1}{2}f''(t)\tau^2$$

Laplacian operator

by rearranging terms:

$$f''(t) = \frac{1}{\tau^2}(f(t - \tau) - 2f(t) + f(t + \tau))$$

which, on the discrete grid, is the FIR $h[n] = [1 \ -2 \ 1]$

Laplacian

summing the horizontal and vertical components:

$$h[n_1, n_2] = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Laplacian

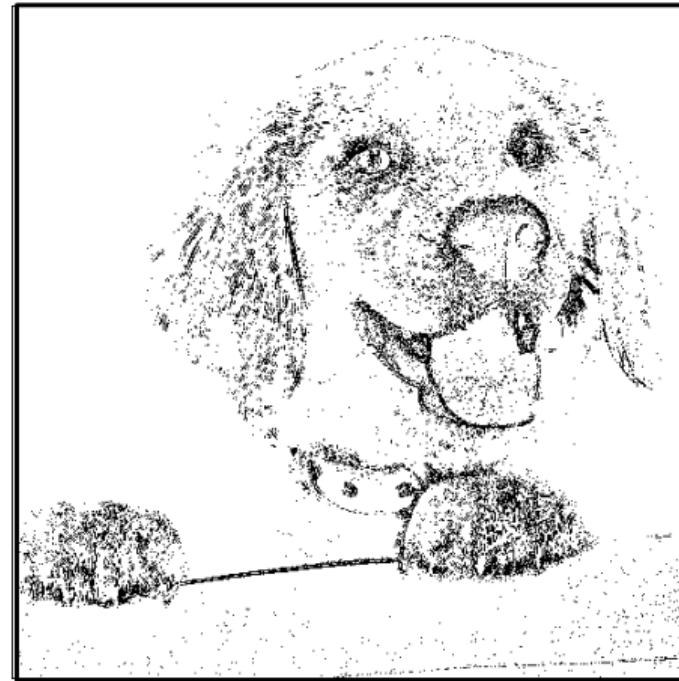
If we use the diagonals too:

$$h[n_1, n_2] = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Laplacian for Edge Detection



Laplacian operator



thresholded Laplacian operator

COM303: Digital Signal Processing

Lecture 21: Image Compression

overview

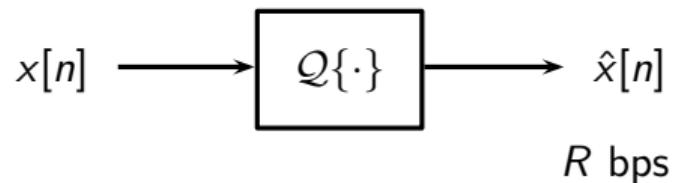
- ▶ introduction to quantization
- ▶ the problem of image compression
- ▶ the JPEG standard

quantization: the basics

Quantization

- ▶ digital devices can only deal with integers (R bits per sample)
- ▶ we need to map the range of a signal onto a finite set of values
- ▶ e.g. pixel sensor level is represented with 8 bits
- ▶ irreversible loss of information → quantization noise

Uniform scalar quantization

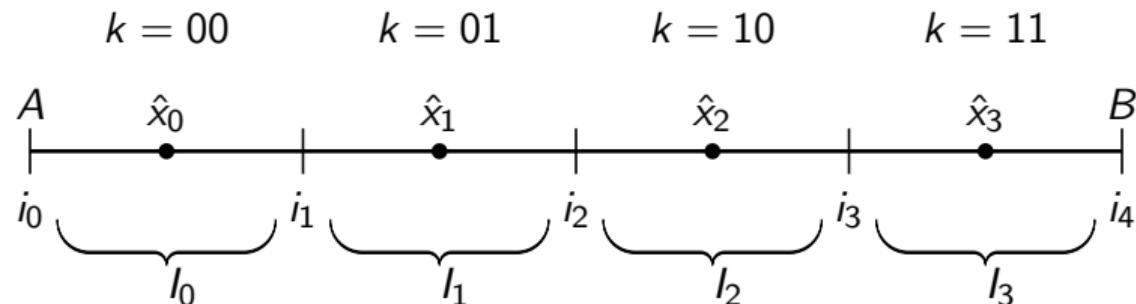


The simplest quantizer:

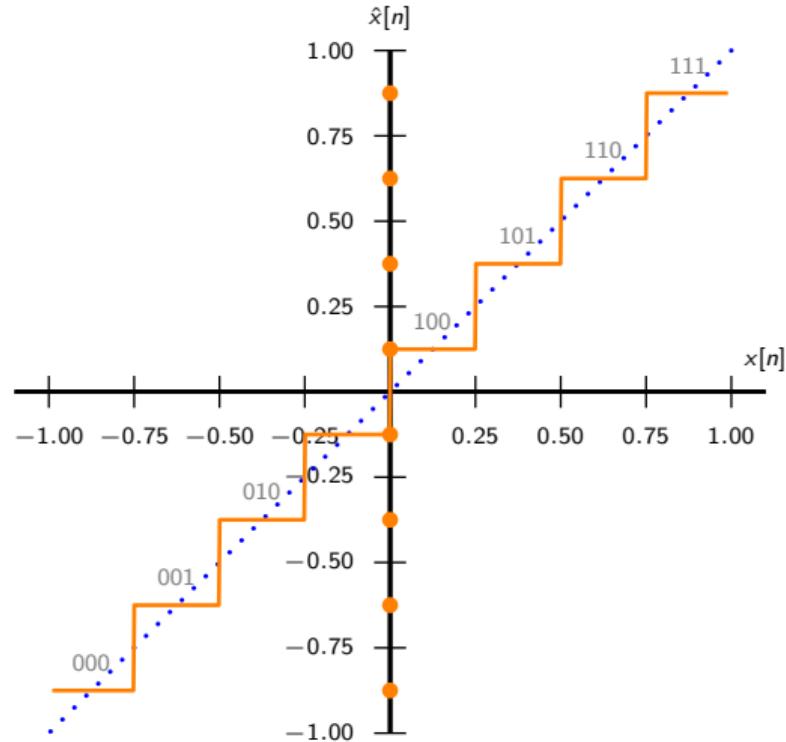
- ▶ each sample is encoded individually (hence *scalar*) using R bits
- ▶ each sample is quantized independently (memoryless quantization)
- ▶ the input range is divided into 2^R equal-size intervals

Uniform scalar quantization

- ▶ input assumed uniformly distributed over $[A, B]$
- ▶ range is split into 2^R *equal* intervals of width $\Delta = (B - A)2^{-R}$
- ▶ quantized value is interval's midpoint



Uniform 3-Bit quantization function

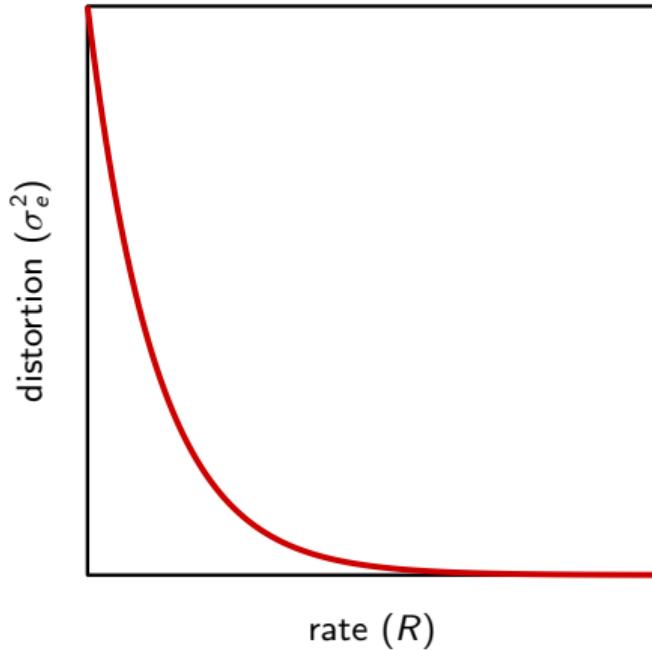


Quantization Error

$$e[n] = Q\{x[n]\} - x[n] = \hat{x}[n] - x[n]$$

- ▶ model $x[n]$ as a stochastic process
- ▶ model error as a white noise sequence:
 - error samples are uncorrelated
 - all error samples have the same distribution

Rate/Distortion Curve



Dynamic range after quantization

Convenient rule of thumb:

- ▶ signal to noise ratio:

$$\text{SNR} = 2^{2R}$$

- ▶ in dBs:

$$\text{SNR}_{\text{dB}} = 10 \log_{10} 2^{2R} \approx 6R \text{ dB}$$

The “6dB/bit” rule of thumb

- ▶ a compact disk has 16 bits/sample:

$$\text{max SNR} = 96\text{dB}$$

- ▶ a DVD has 24 bits/sample:

$$\text{max SNR} = 144\text{dB}$$

The “6dB/bit” rule of thumb

- ▶ a compact disk has 16 bits/sample:

$$\text{max SNR} = 96\text{dB}$$

- ▶ a DVD has 24 bits/sample:

$$\text{max SNR} = 144\text{dB}$$

image compression fundamentals

Remember our thought experiment...

- ▶ consider all possible 256×256 , 8bpp images
- ▶ each image is 524,288 bits
- ▶ total number of possible images: $2^{524,288} \approx 10^{157,826}$
- ▶ number of atoms in the universe: 10^{82}

Remember our thought experiment...

- ▶ consider all possible 256×256 , 8bpp images
- ▶ each image is 524,288 bits
- ▶ total number of possible images: $2^{524,288} \approx 10^{157,826}$
- ▶ number of atoms in the universe: 10^{82}

Remember our thought experiment...

- ▶ consider all possible 256×256 , 8bpp images
- ▶ each image is 524,288 bits
- ▶ total number of possible images: $2^{524,288} \approx 10^{157,826}$
- ▶ number of atoms in the universe: 10^{82}

Remember our thought experiment...

- ▶ consider all possible 256×256 , 8bpp images
- ▶ each image is 524,288 bits
- ▶ total number of possible images: $2^{524,288} \approx 10^{157,826}$
- ▶ number of atoms in the universe: 10^{82}

How many bits per image?

Another thought experiment

- ▶ take *all* images in the world and list them in an “encyclopedia of images”
- ▶ to indicate an image, simply give its number
- ▶ on the Internet: $M = 50$ billion
- ▶ raw encoding: 524,288 bits per image
- ▶ enumeration-based encoding: $\log_2 M \approx 36$ bits per image
- ▶ (of course, side information is HUGE)

How many bits per image?

Another thought experiment

- ▶ take *all* images in the world and list them in an “encyclopedia of images”
- ▶ to indicate an image, simply give its number

- ▶ on the Internet: $M = 50$ billion
- ▶ raw encoding: 524,288 bits per image
- ▶ enumeration-based encoding: $\log_2 M \approx 36$ bits per image
- ▶ (of course, side information is HUGE)

How many bits per image?

Another thought experiment

- ▶ take *all* images in the world and list them in an “encyclopedia of images”
- ▶ to indicate an image, simply give its number

- ▶ on the Internet: $M = 50$ billion
- ▶ raw encoding: 524,288 bits per image
- ▶ enumeration-based encoding: $\log_2 M \approx 36$ bits per image
- ▶ (of course, side information is HUGE)

How many bits per image?

Another thought experiment

- ▶ take *all* images in the world and list them in an “encyclopedia of images”
- ▶ to indicate an image, simply give its number

- ▶ on the Internet: $M = 50$ billion
- ▶ raw encoding: 524,288 bits per image
- ▶ enumeration-based encoding: $\log_2 M \approx 36$ bits per image
- ▶ (of course, side information is HUGE)

How many bits per image?

Another thought experiment

- ▶ take *all* images in the world and list them in an “encyclopedia of images”
- ▶ to indicate an image, simply give its number

- ▶ on the Internet: $M = 50$ billion
- ▶ raw encoding: 524,288 bits per image
- ▶ enumeration-based encoding: $\log_2 M \approx 36$ bits per image
- ▶ (of course, side information is HUGE)

How many bits per image?

Another thought experiment

- ▶ take *all* images in the world and list them in an “encyclopedia of images”
- ▶ to indicate an image, simply give its number

- ▶ on the Internet: $M = 50$ billion
- ▶ raw encoding: 524,288 bits per image
- ▶ enumeration-based encoding: $\log_2 M \approx 36$ bits per image
- ▶ (of course, side information is HUGE)

Compression

Another approach:

- ▶ exploit “physical” redundancy
- ▶ allocate bits for things that matter (e.g. edges)
- ▶ use psychovisual experiments to find out what matters
- ▶ some information is discarded: *lossy* compression

Compression

Another approach:

- ▶ exploit “physical” redundancy
- ▶ allocate bits for things that matter (e.g. edges)
- ▶ use psychovisual experiments to find out what matters
- ▶ some information is discarded: *lossy* compression

Compression

Another approach:

- ▶ exploit “physical” redundancy
- ▶ allocate bits for things that matter (e.g. edges)
- ▶ use psychovisual experiments to find out what matters
- ▶ some information is discarded: *lossy* compression

Compression

Another approach:

- ▶ exploit “physical” redundancy
- ▶ allocate bits for things that matter (e.g. edges)
- ▶ use psychovisual experiments to find out what matters
- ▶ some information is discarded: *lossy* compression

Key ingredients

- ▶ compressing at block level
- ▶ using a suitable transform (i.e., a change of basis)
- ▶ smart quantization
- ▶ entropy coding

Key ingredients

- ▶ compressing at block level
- ▶ using a suitable transform (i.e., a change of basis)
- ▶ smart quantization
- ▶ entropy coding

Key ingredients

- ▶ compressing at block level
- ▶ using a suitable transform (i.e., a change of basis)
- ▶ smart quantization
- ▶ entropy coding

Key ingredients

- ▶ compressing at block level
- ▶ using a suitable transform (i.e., a change of basis)
- ▶ smart quantization
- ▶ entropy coding

Compressing at pixel level

- ▶ reduce number bits per pixel
- ▶ equivalent to coarser quantization
- ▶ in the limit, 1bpp

Compressing at pixel level

- ▶ reduce number bits per pixel
- ▶ equivalent to coarser quantization
- ▶ in the limit, 1bpp

Compressing at pixel level

- ▶ reduce number bits per pixel
- ▶ equivalent to coarser quantization
- ▶ in the limit, 1bpp



Compressing at block level

- ▶ divide the image in blocks
- ▶ code the average value with 8 bits
- ▶ 3×3 blocks at 8 bits per block gives less than 1bpp

Compressing at block level

- ▶ divide the image in blocks
- ▶ code the average value with 8 bits
- ▶ 3×3 blocks at 8 bits per block gives less than 1bpp

Compressing at block level

- ▶ divide the image in blocks
- ▶ code the average value with 8 bits
- ▶ 3×3 blocks at 8 bits per block gives less than 1bpp



Compressing at block level

- ▶ exploit the local spatial correlation
- ▶ compress remote regions independently

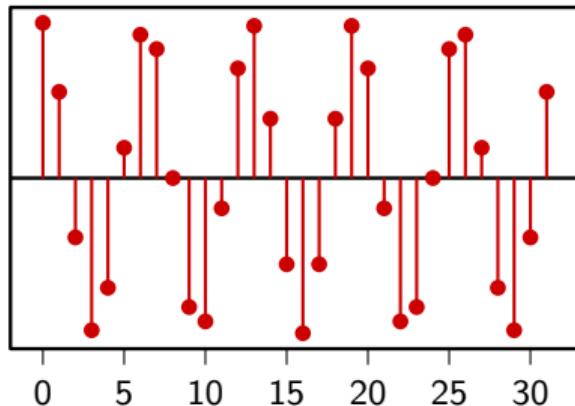
Compressing at block level

- ▶ exploit the local spatial correlation
- ▶ compress remote regions independently

Transform coding

A simple example:

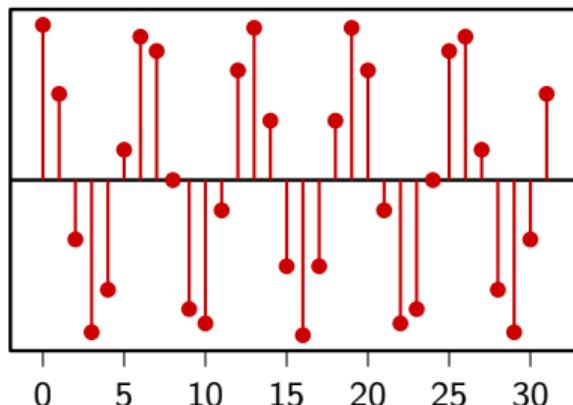
- ▶ take a DT signal, assume R bits per sample
- ▶ storing the signal requires NR bits
- ▶ now you take the DFT and it looks like this
- ▶ in theory, we can just code the two nonzero DFT coefficients!



Transform coding

A simple example:

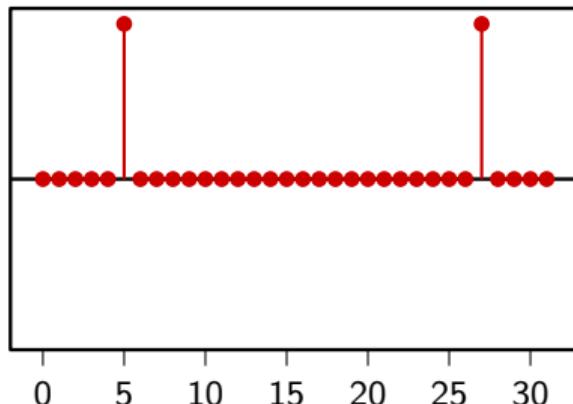
- ▶ take a DT signal, assume R bits per sample
- ▶ storing the signal requires NR bits
- ▶ now you take the DFT and it looks like this
- ▶ in theory, we can just code the two nonzero DFT coefficients!



Transform coding

A simple example:

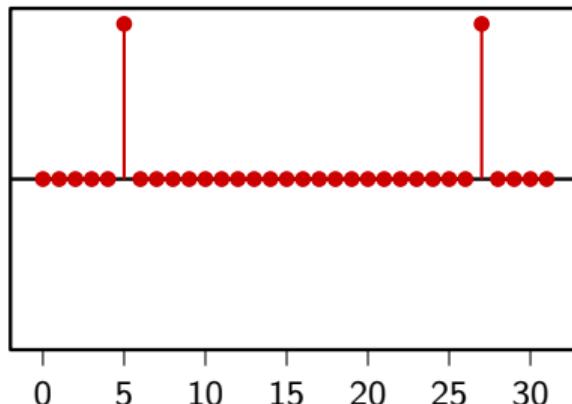
- ▶ take a DT signal, assume R bits per sample
- ▶ storing the signal requires NR bits
- ▶ now you take the DFT and it looks like this
- ▶ in theory, we can just code the two nonzero DFT coefficients!



Transform coding

A simple example:

- ▶ take a DT signal, assume R bits per sample
- ▶ storing the signal requires NR bits
- ▶ now you take the DFT and it looks like this
- ▶ in theory, we can just code the two nonzero DFT coefficients!



Transform coding

Ideally, we would like a transform that:

- ▶ captures the important features of an image block in a few coefficients
- ▶ is efficient to compute
- ▶ answer: the Discrete Cosine Transform

Transform coding

Ideally, we would like a transform that:

- ▶ captures the important features of an image block in a few coefficients
- ▶ is efficient to compute
- ▶ answer: the Discrete Cosine Transform

Transform coding

Ideally, we would like a transform that:

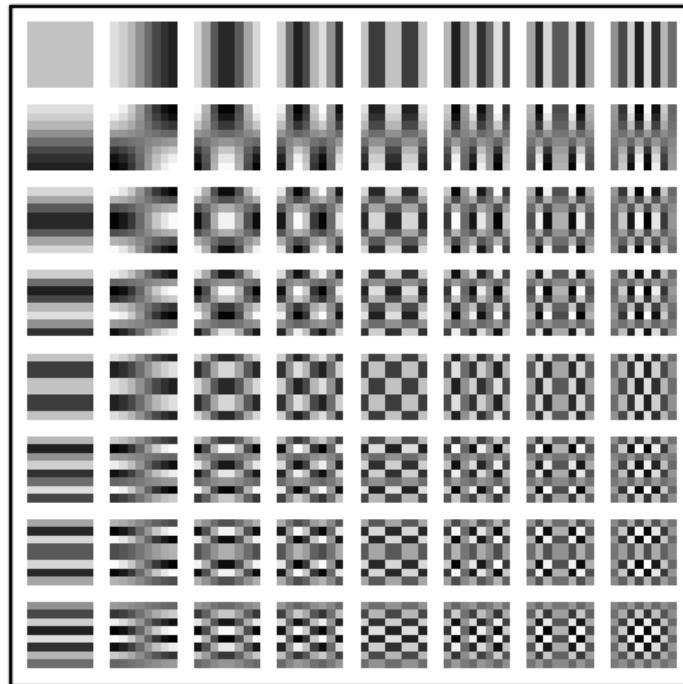
- ▶ captures the important features of an image block in a few coefficients
- ▶ is efficient to compute
- ▶ answer: the Discrete Cosine Transform

2D-DCT

$$C[k_1, k_2] = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x[n_1, n_2] \cos \left[\frac{\pi}{N} \left(n_1 + \frac{1}{2} \right) k_1 \right] \cos \left[\frac{\pi}{N} \left(n_2 + \frac{1}{2} \right) k_2 \right]$$

$$C[k_1, k_2] \in \mathbb{R}$$

DCT basis vectors for an 8×8 image



Smart quantization of the DCT coefficients

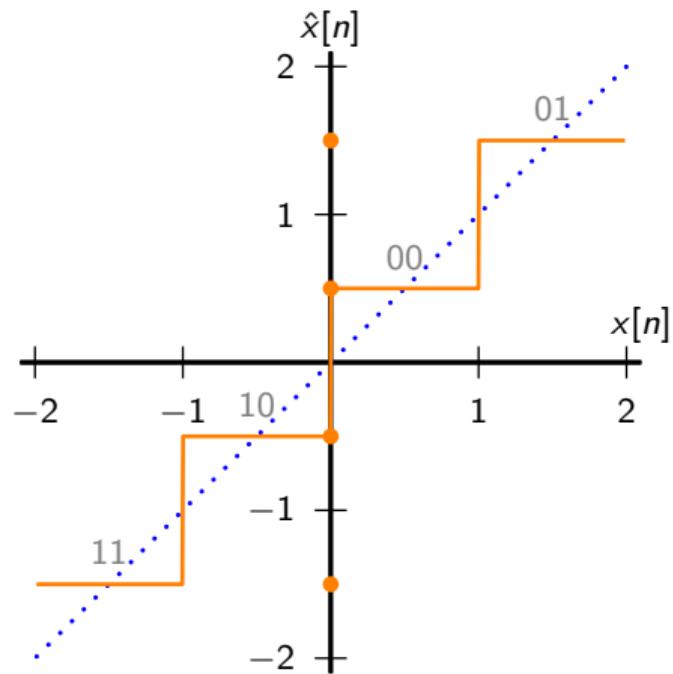
- ▶ deadzone
- ▶ variable step (fine to coarse)

Smart quantization of the DCT coefficients

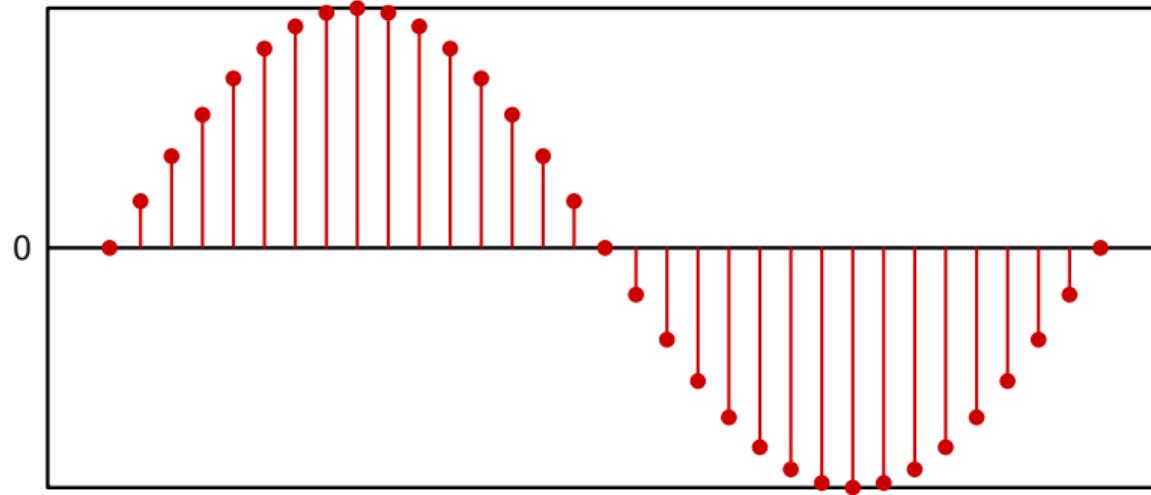
- ▶ deadzone
- ▶ variable step (fine to coarse)

Standard Uniform Quantization

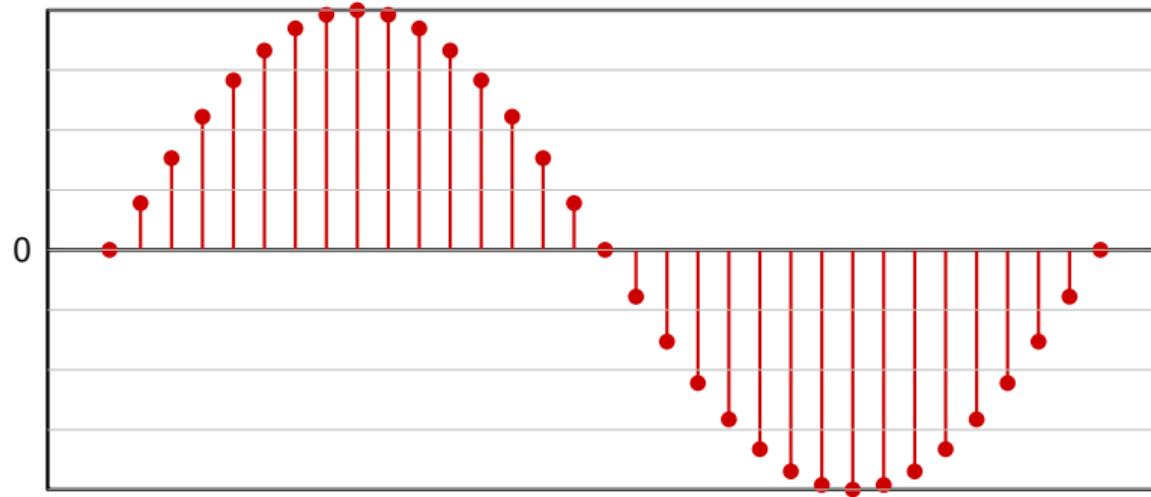
$$\hat{x} = \text{floor}(x) + 0.5$$



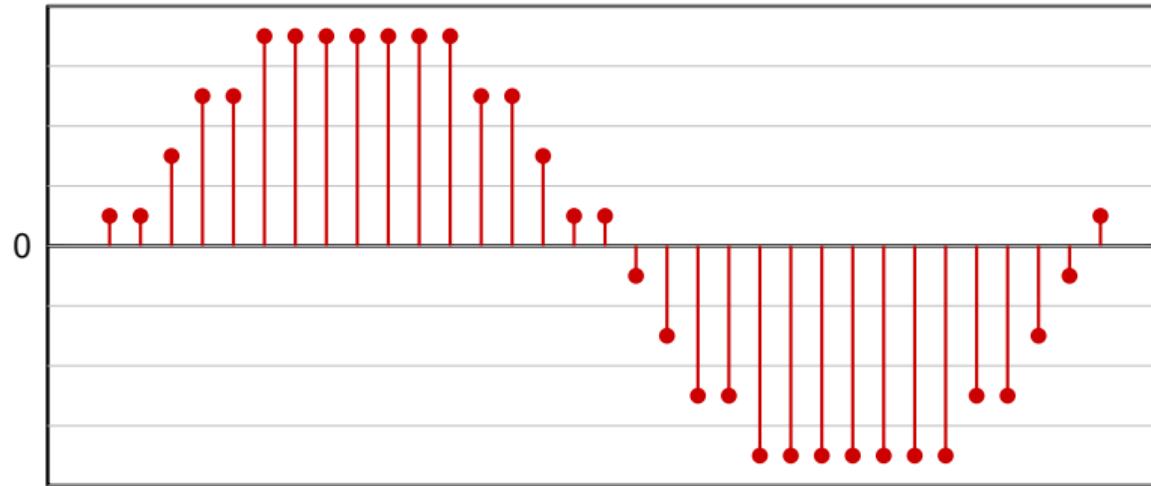
Quantizing a full-range signal



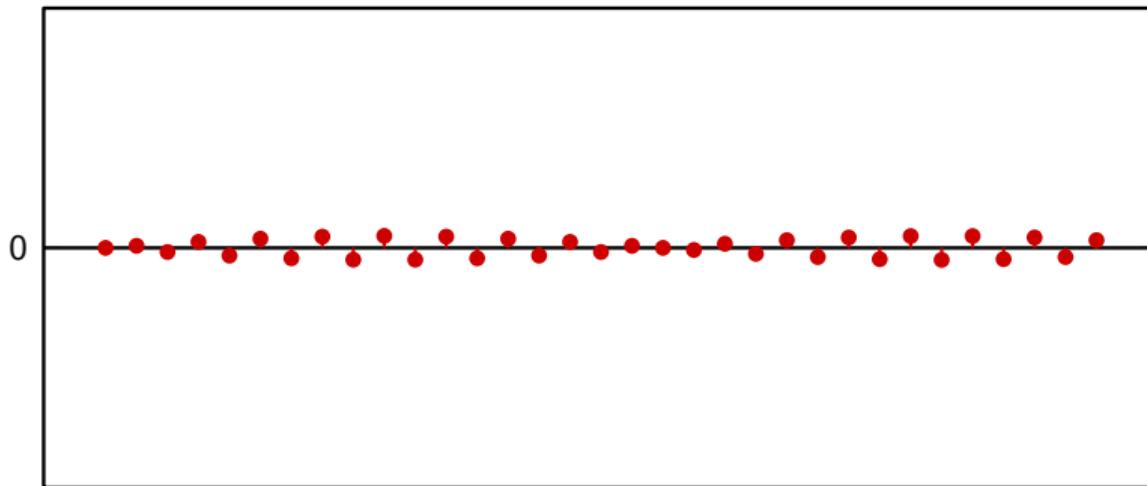
Quantizing a full-range signal



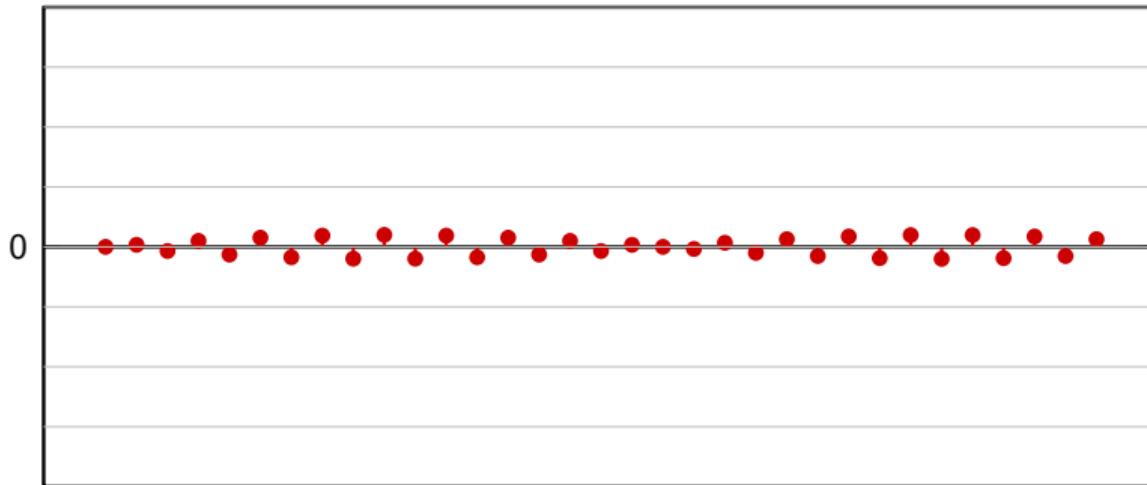
Quantizing a full-range signal



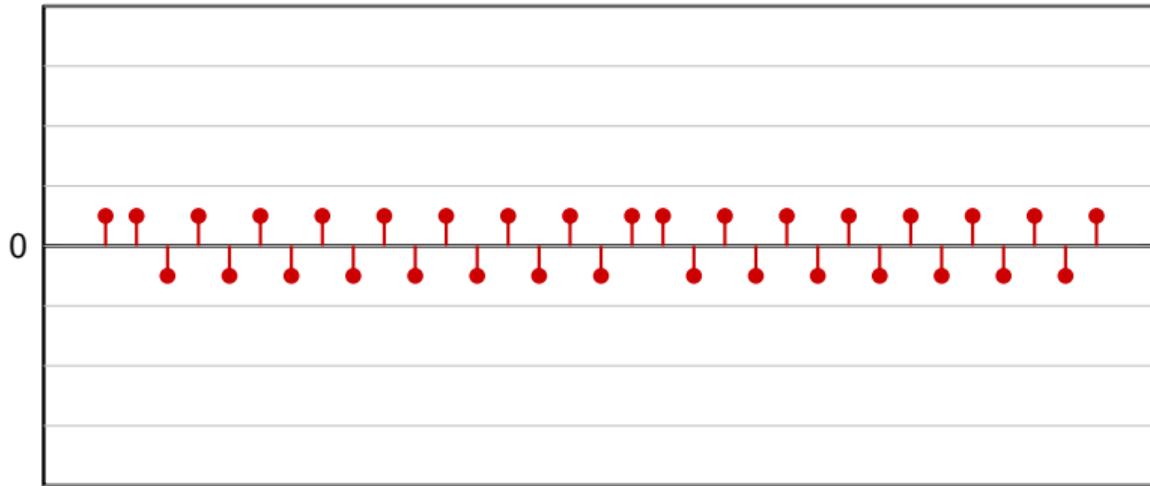
Quantizing a small, noise-like signal



Quantizing a small, noise-like signal

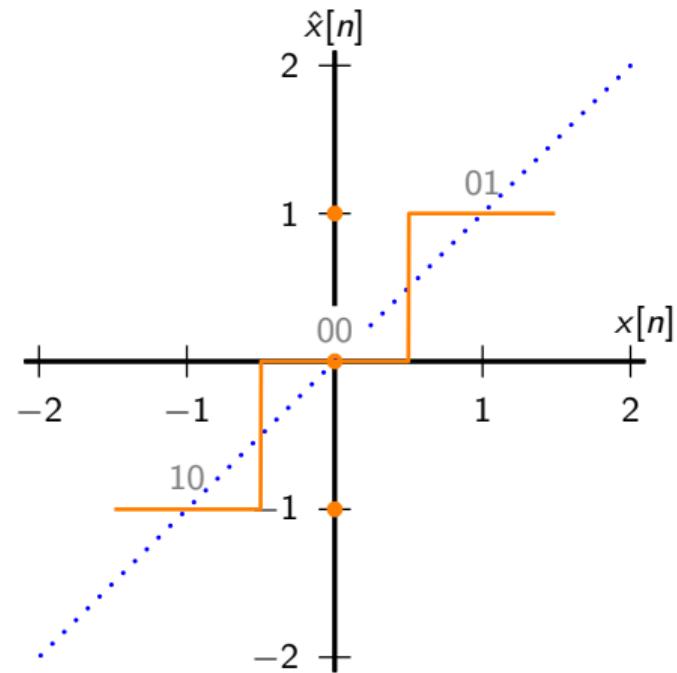


Quantizing a small, noise-like signal

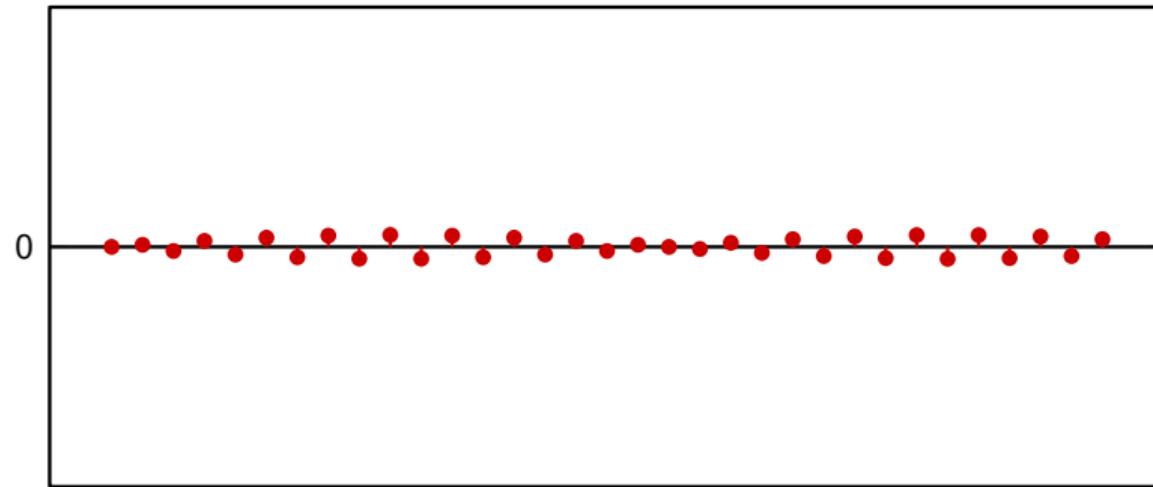


Deadzone Quantization

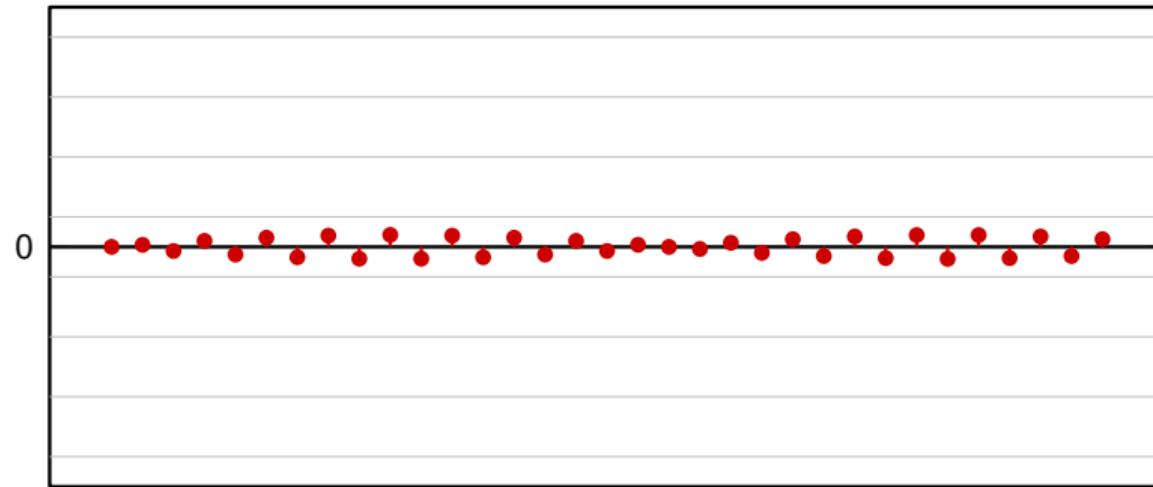
$$\hat{x} = \text{round}(x)$$



Quantizing a small, noise-like signal, take 2



Quantizing a small, noise-like signal, take 2



Quantizing a small, noise-like signal, take 2



Entropy coding

- ▶ minimize the effort to encode a certain amount of information
- ▶ associate short symbols to frequent values and vice-versa
- ▶ if it sounds familiar it's because it is...

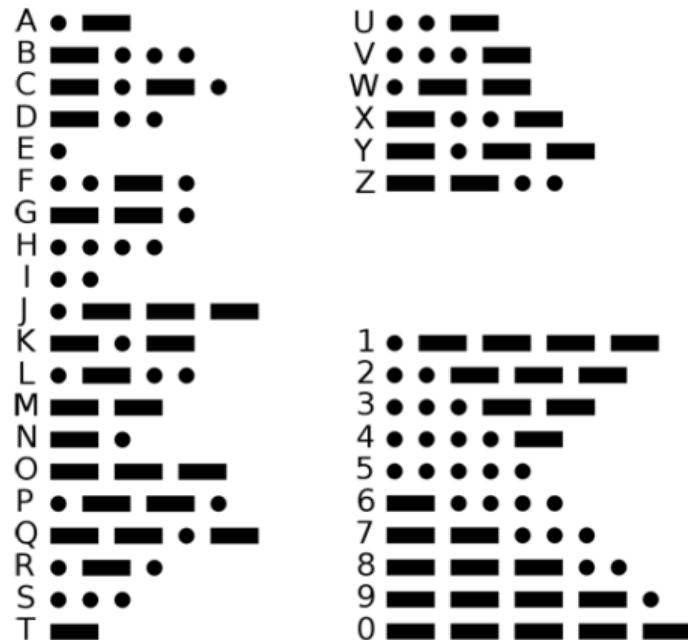
Entropy coding

- ▶ minimize the effort to encode a certain amount of information
- ▶ associate short symbols to frequent values and vice-versa
- ▶ if it sounds familiar it's because it is...

Entropy coding

- ▶ minimize the effort to encode a certain amount of information
- ▶ associate short symbols to frequent values and vice-versa
- ▶ if it sounds familiar it's because it is...

Entropy coding



the JPEG standard

Key ingredients

- ▶ compressing at block level
- ▶ using a suitable transform (i.e., a change of basis)
- ▶ smart quantization
- ▶ entropy coding

Key ingredients

- ▶ split image into 8×8 non-overlapping blocks
- ▶ using a suitable transform (i.e., a change of basis)
- ▶ smart quantization
- ▶ entropy coding

Key ingredients

- ▶ split image into 8×8 non-overlapping blocks
- ▶ compute the DCT of each block
- ▶ smart quantization
- ▶ entropy coding

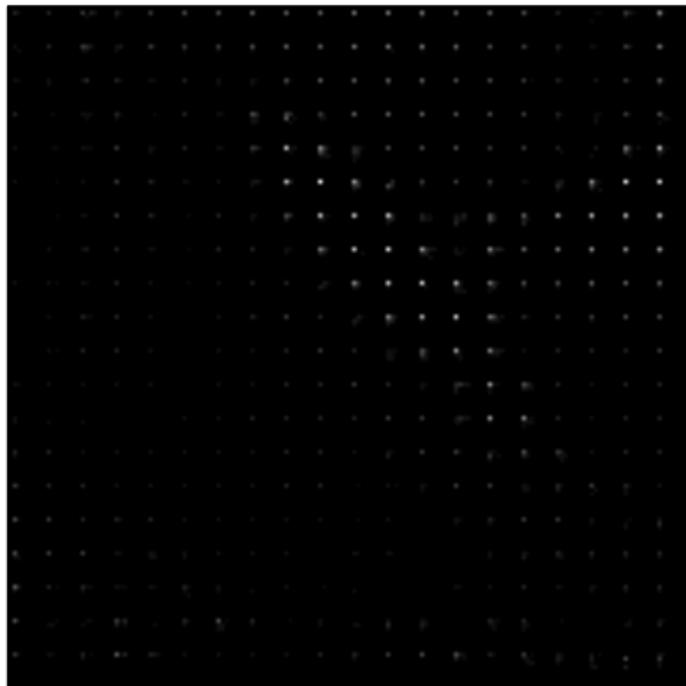
Key ingredients

- ▶ split image into 8×8 non-overlapping blocks
- ▶ compute the DCT of each block
- ▶ quantize DCT coefficients according to psycovisually-tuned tables
- ▶ entropy coding

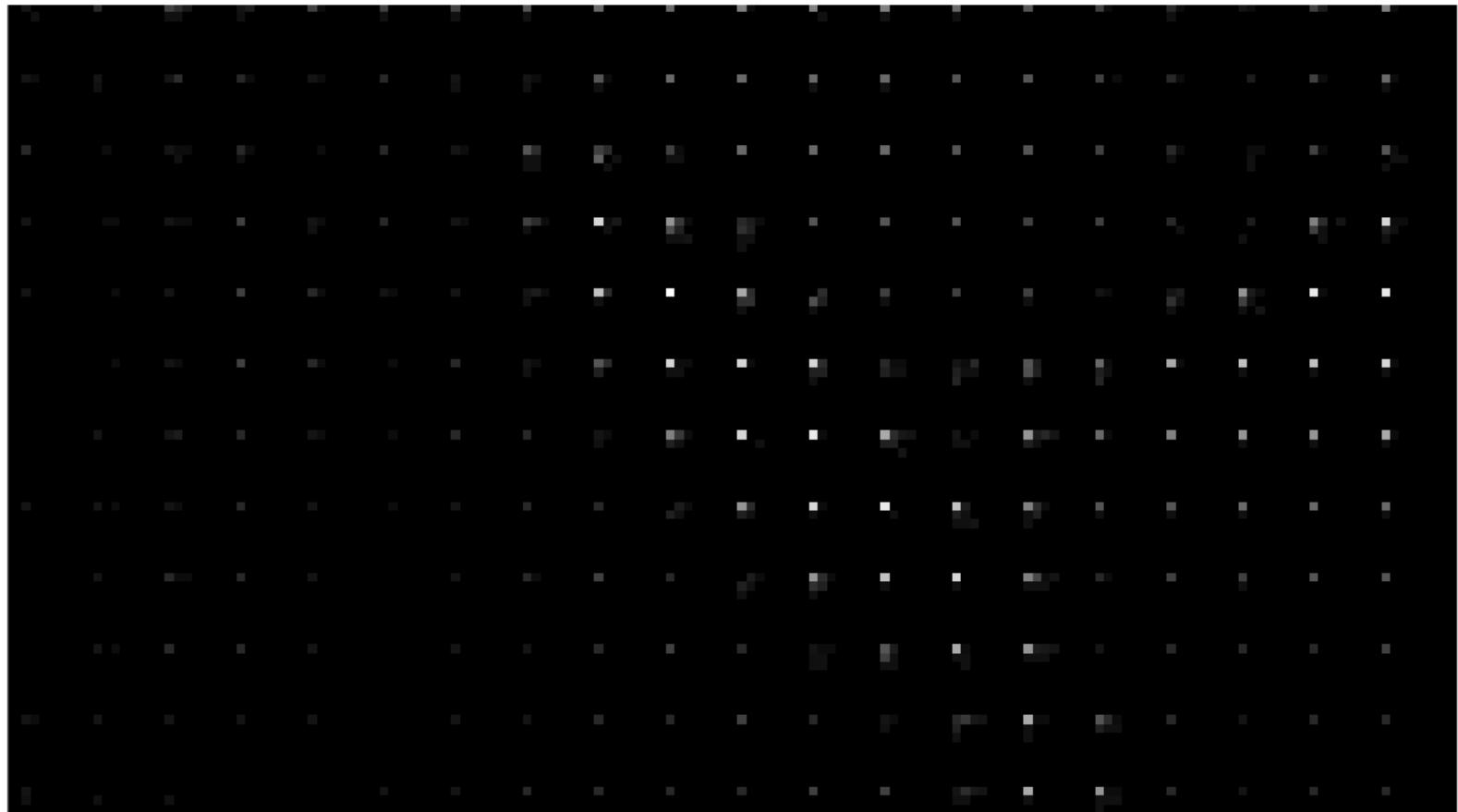
Key ingredients

- ▶ split image into 8×8 non-overlapping blocks
- ▶ compute the DCT of each block
- ▶ quantize DCT coefficients according to psychovisually-tuned tables
- ▶ run-length encoding and Huffman coding

DCT coefficients of image blocks (detail)



DCT coefficients of image blocks (detail)



Smart quantization

- ▶ most coefficients are negligible → captured by the deadzone
- ▶ some coefficients have a higher visual impact
- ▶ find out the critical coefficients by experimentation
- ▶ use smaller quantization intervals (i.e. use more bits) for the important coefficients

Smart quantization

- ▶ most coefficients are negligible → captured by the deadzone
- ▶ some coefficients have a higher visual impact
- ▶ find out the critical coefficients by experimentation
- ▶ use smaller quantization intervals (i.e. use more bits) for the important coefficients

Smart quantization

- ▶ most coefficients are negligible → captured by the deadzone
- ▶ some coefficients have a higher visual impact
- ▶ find out the critical coefficients by experimentation
- ▶ use smaller quantization intervals (i.e. use more bits) for the important coefficients

Smart quantization

- ▶ most coefficients are negligible → captured by the deadzone
- ▶ some coefficients have a higher visual impact
- ▶ find out the critical coefficients by experimentation
- ▶ use smaller quantization intervals (i.e. use more bits) for the important coefficients

Psychovisually-tuned quantization table

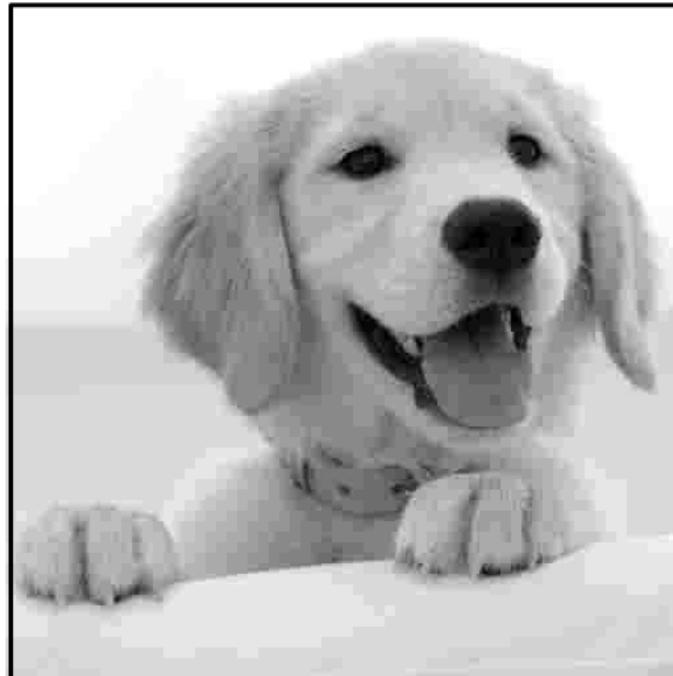
$$\hat{c}[k_1, k_2] = \text{round}(c[k_1, k_2]/Q[k_1, k_2])$$

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Advantages of tuned quantization intervals at 0.2bpp



uniform



tuned

Advantages of tuned quantization intervals (detail)



uniform



tuned

Efficient coding

- ▶ most coefficients are small, decreasing with index
- ▶ use zigzag scan to maximize ordering
- ▶ quantization will create long series of zeros

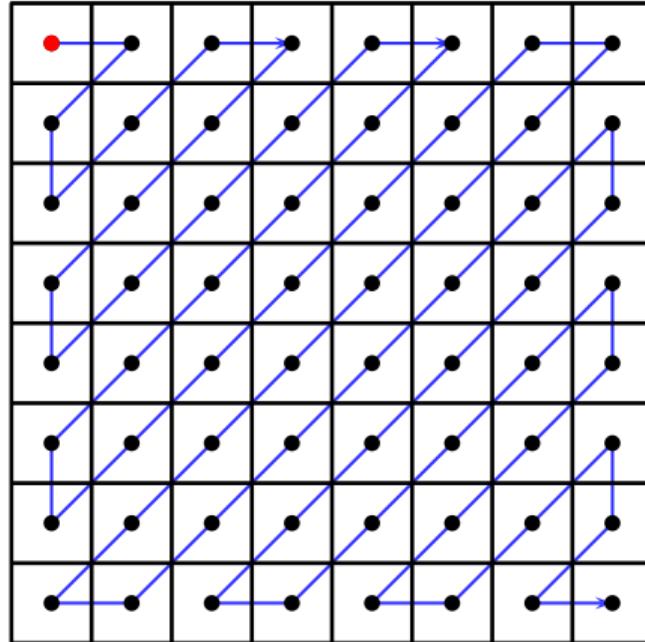
Efficient coding

- ▶ most coefficients are small, decreasing with index
- ▶ use zigzag scan to maximize ordering
- ▶ quantization will create long series of zeros

Efficient coding

- ▶ most coefficients are small, decreasing with index
- ▶ use zigzag scan to maximize ordering
- ▶ quantization will create long series of zeros

Zigzag scan



Example

$$\begin{bmatrix} 100 & -60 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Example

Runlength encoding

- ▶ the DC value is encoded differentially wrt previous block
- ▶ each nonzero AC value is encoded as the triple

$$[(r, s), c]$$

- r is the *runlength* i.e. the number of zeros before the current value
- s is the *category* i.e. the number of bits needed to encode the value
- c is the actual value
- $(0, 0)$ indicates that from now on it's only zeros (end of block)

Runlength encoding

- ▶ the DC value is encoded differentially wrt previous block
- ▶ each nonzero AC value is encoded as the triple

$$[(r, s), c]$$

- r is the *runlength* i.e. the number of zeros before the current value
- s is the *category* i.e. the number of bits needed to encode the value
- c is the actual value
- $(0, 0)$ indicates that from now on it's only zeros (end of block)

Runlength encoding

- ▶ the DC value is encoded differentially wrt previous block
- ▶ each nonzero AC value is encoded as the triple

$$[(r, s), c]$$

- r is the *runlength* i.e. the number of zeros before the current value
 - s is the *category* i.e. the number of bits needed to encode the value
 - c is the actual value
 - $(0, 0)$ indicates that from now on it's only zeros (end of block)

Runlength encoding

- ▶ the DC value is encoded differentially wrt previous block
- ▶ each nonzero AC value is encoded as the triple

$$[(r, s), c]$$

- r is the *runlength* i.e. the number of zeros before the current value
- s is the *category* i.e. the number of bits needed to encode the value
 - c is the actual value
 - $(0, 0)$ indicates that from now on it's only zeros (end of block)

Runlength encoding

- ▶ the DC value is encoded differentially wrt previous block
- ▶ each nonzero AC value is encoded as the triple

$$[(r, s), c]$$

- r is the *runlength* i.e. the number of zeros before the current value
- s is the *category* i.e. the number of bits needed to encode the value
- c is the actual value
- $(0, 0)$ indicates that from now on it's only zeros (end of block)

Runlength encoding

- ▶ the DC value is encoded differentially wrt previous block
- ▶ each nonzero AC value is encoded as the triple

$$[(r, s), c]$$

- r is the *runlength* i.e. the number of zeros before the current value
- s is the *category* i.e. the number of bits needed to encode the value
- c is the actual value
- $(0, 0)$ indicates that from now on it's only zeros (end of block)

Example

$$\begin{bmatrix} 100 & -60 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

[100], [(0, 6), -60], [(4, 3), 6], [(3, 4), 13], [(8, 1), -1], [(0, 0)]

Example

$$\begin{bmatrix} 100 & -60 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

[100], [(0, 6), -60], [(4, 3), 6], [(3, 4), 13], [(8, 1), -1], [(0, 0)]

The runlength-size pairs

- ▶ by design, $(r, s) \in \mathcal{A}$ with $|\mathcal{A}| = 256$
- ▶ in theory, 8 bits per pair
- ▶ some pairs are much more common than others!
- ▶ a lot of space can be saved by being smart

The runlength-size pairs

- ▶ by design, $(r, s) \in \mathcal{A}$ with $|\mathcal{A}| = 256$
- ▶ in theory, 8 bits per pair
- ▶ some pairs are much more common than others!
- ▶ a lot of space can be saved by being smart

The runlength-size pairs

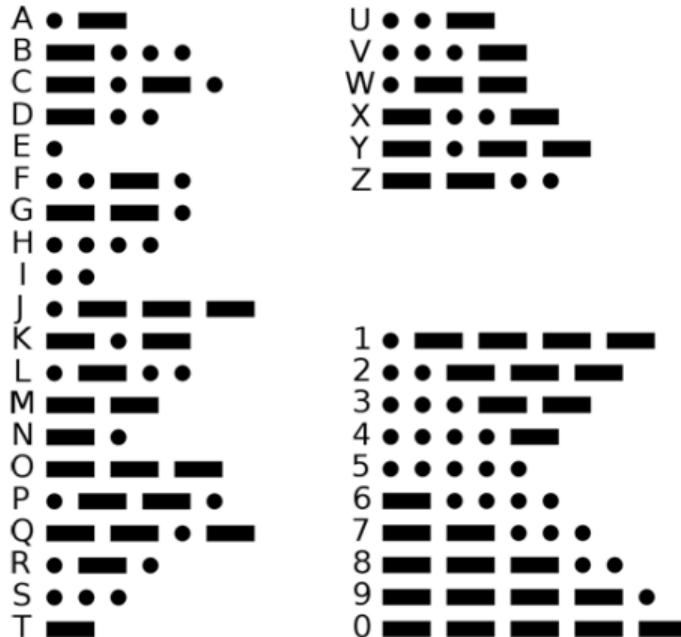
- ▶ by design, $(r, s) \in \mathcal{A}$ with $|\mathcal{A}| = 256$
- ▶ in theory, 8 bits per pair
- ▶ some pairs are much more common than others!
- ▶ a lot of space can be saved by being smart

The runlength-size pairs

- ▶ by design, $(r, s) \in \mathcal{A}$ with $|\mathcal{A}| = 256$
- ▶ in theory, 8 bits per pair
- ▶ some pairs are much more common than others!
- ▶ a lot of space can be saved by being smart

Variable-length encoding

great idea: shorter binary sequences for common symbols



Variable-length encoding

however: if symbols have different lengths, we must know how to parse them!

- ▶ in English, spaces separate words → extra symbol (wasteful)
- ▶ in Morse code, pauses separate letters and words (wasteful)
- ▶ can we do away with separators?

Variable-length encoding

however: if symbols have different lengths, we must know how to parse them!

- ▶ in English, spaces separate words → extra symbol (wasteful)
- ▶ in Morse code, pauses separate letters and words (wasteful)
- ▶ can we do away with separators?

Variable-length encoding

however: if symbols have different lengths, we must know how to parse them!

- ▶ in English, spaces separate words → extra symbol (wasteful)
- ▶ in Morse code, pauses separate letters and words (wasteful)
- ▶ can we do away with separators?

Prefix-free codes

- ▶ no valid sequence can be the beginning of another valid sequence
- ▶ can parse a bitstream sequentially with no look-ahead
- ▶ extremely easy to understand graphically...

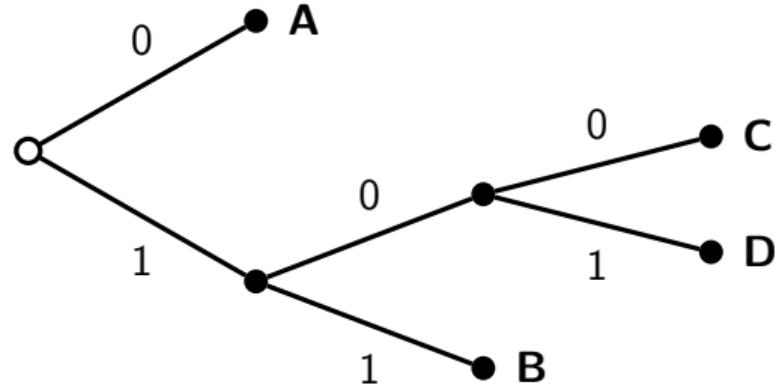
Prefix-free codes

- ▶ no valid sequence can be the beginning of another valid sequence
- ▶ can parse a bitstream sequentially with no look-ahead
- ▶ extremely easy to understand graphically...

Prefix-free codes

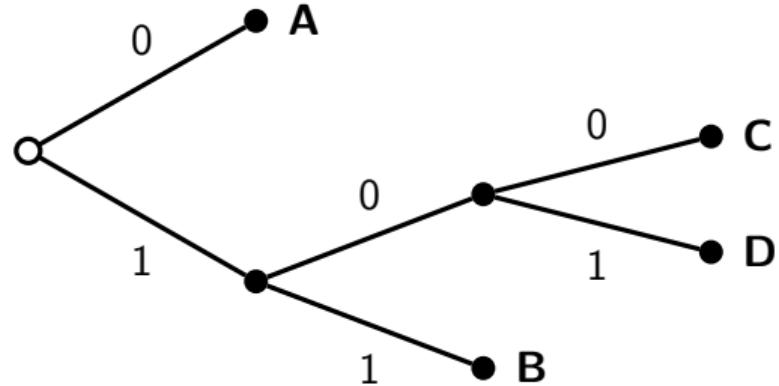
- ▶ no valid sequence can be the beginning of another valid sequence
- ▶ can parse a bitstream sequentially with no look-ahead
- ▶ extremely easy to understand graphically...

Prefix-free code



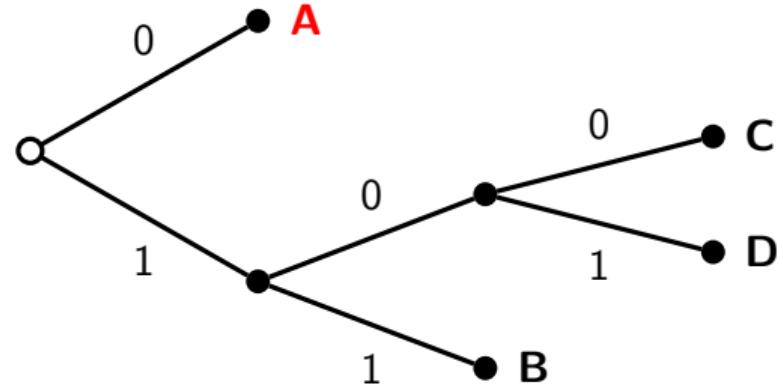
001100110101100

Prefix-free code



001100110101100

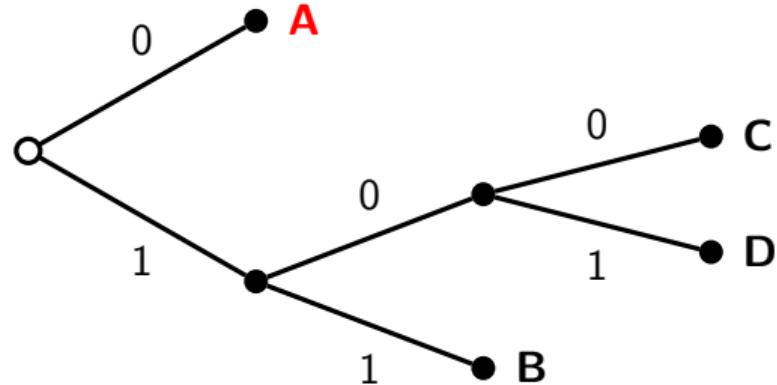
Prefix-free code



001100110101100

A

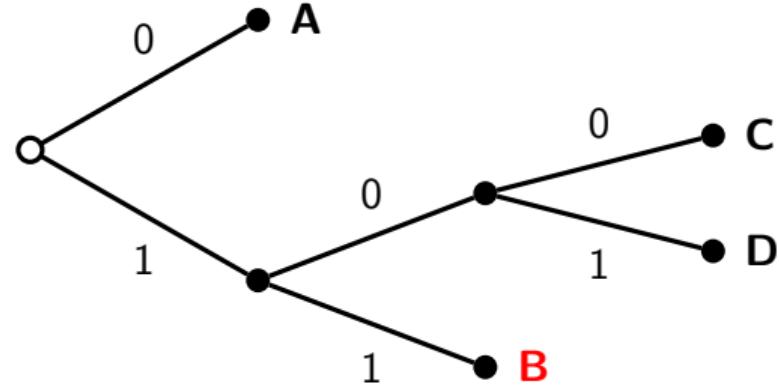
Prefix-free code



001100110101100

AA

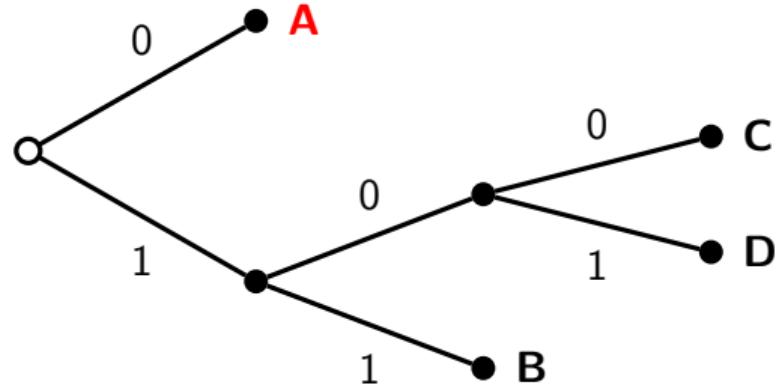
Prefix-free code



00**11**00110101100

AAB

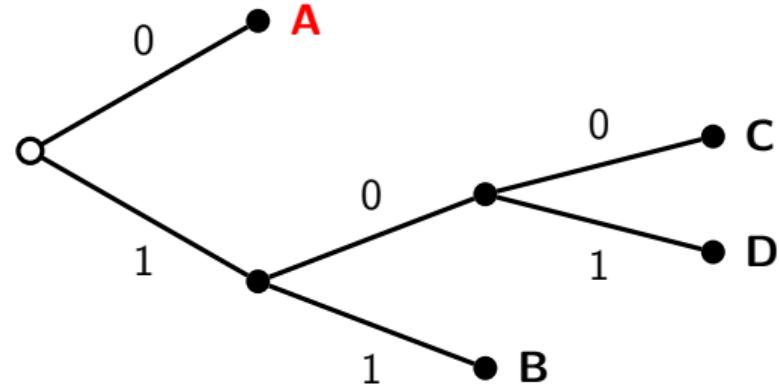
Prefix-free code



001100110101100

AABA

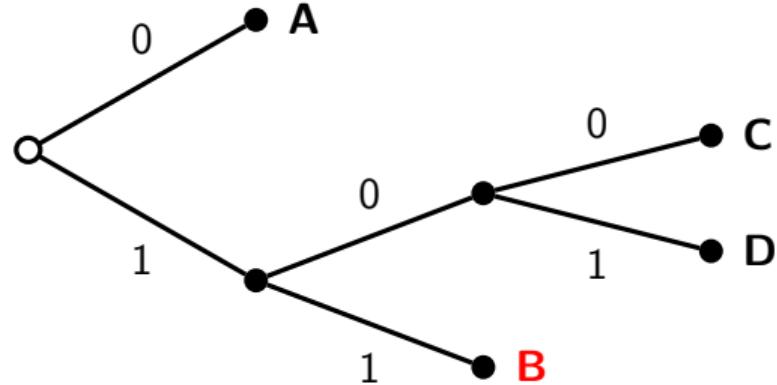
Prefix-free code



001100110101100

AABAA

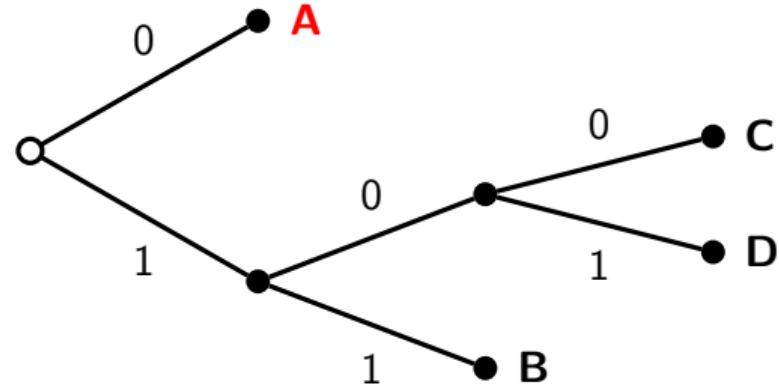
Prefix-free code



001100**11**0101100

AABAAB

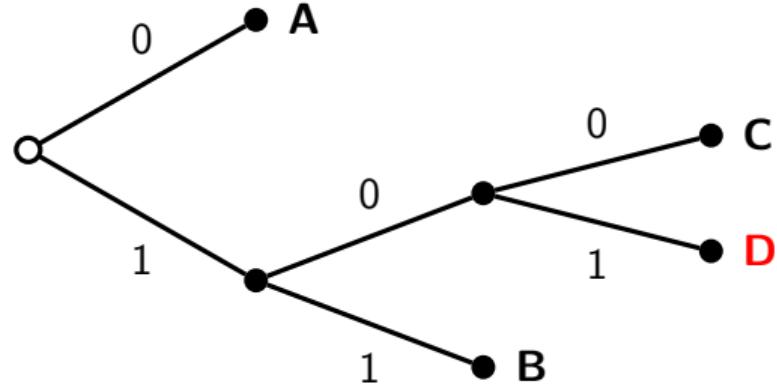
Prefix-free code



001100110101100

AABAABA

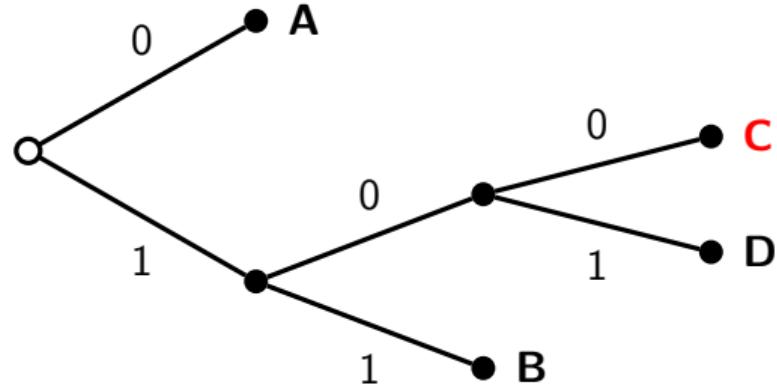
Prefix-free code



001100110**101**100

AABAABAD

Prefix-free code



001100110101100

AABAABADC

Entropy coding

goal: minimize message length

- ▶ assign short sequences to more frequent symbols
- ▶ the Huffman algorithm builds the optimal code for a set of symbol probabilities
- ▶ in JPEG, you can use a “general-purpose” Huffman code or build your own
(but then you pay a “side-information” price)

Entropy coding

goal: minimize message length

- ▶ assign short sequences to more frequent symbols
- ▶ the Huffman algorithm builds the optimal code for a set of symbol probabilities
- ▶ in JPEG, you can use a “general-purpose” Huffman code or build your own
(but then you pay a “side-information” price)

Entropy coding

goal: minimize message length

- ▶ assign short sequences to more frequent symbols
- ▶ the Huffman algorithm builds the optimal code for a set of symbol probabilities
- ▶ in JPEG, you can use a “general-purpose” Huffman code or build your own
(but then you pay a “side-information” price)

Example

- ▶ four symbols: A, B, C, D
- ▶ probability table:

$$p(A) = 0.38$$

$$p(B) = 0.32$$

$$p(C) = 0.1$$

$$p(D) = 0.2$$

Example

- ▶ four symbols: A, B, C, D
- ▶ probability table:

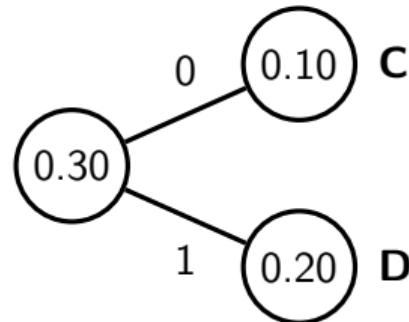
$$p(A) = 0.38$$

$$p(B) = 0.32$$

$$p(C) = 0.1$$

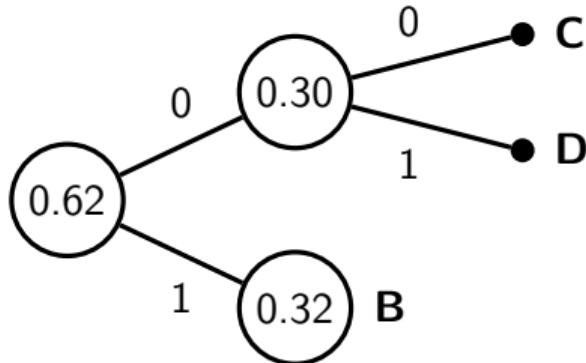
$$p(D) = 0.2$$

Building the Huffman code



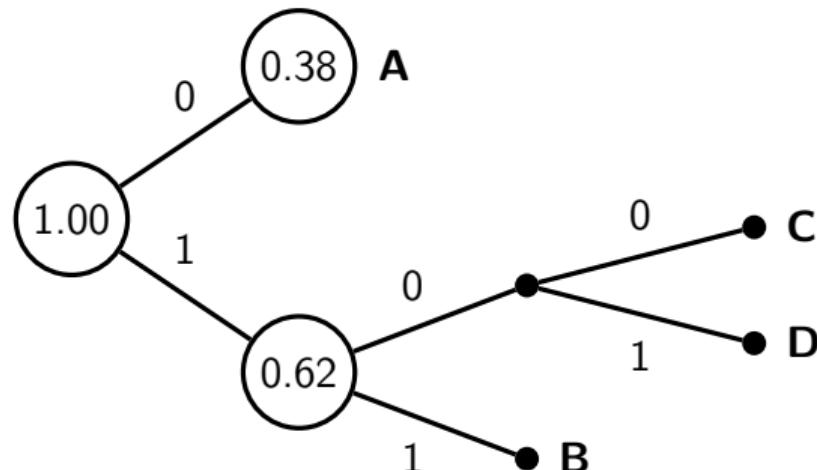
$$p(A) = 0.38 \quad p(B) = 0.32 \quad p(C) = 0.1 \quad p(D) = 0.2$$

Building the Huffman code



$$p(A) = 0.38 \quad p(B) = 0.32 \quad p(C + D) = 0.3$$

Huffman Coding



$$p(A) = 0.38 \quad p(B + C + D) = 0.62$$

Conclusions

- ▶ JPEG is a very complex and comprehensive standard:
 - lossless, lossy
 - color, B&W
 - progressive encoding
 - HDR (12bpp) for medical imaging
- ▶ JPEG is VERY good:
 - compression factor of 10:1 virtually indistinguishable
 - rates of 1bpp for RGB images acceptable (25:1 compression ratio)
- ▶ other important compression schemes:
 - TIFF, JPEG2000
 - MPEG (MP3)

COM303: Digital Signal Processing

Lecture 22: Digital Communication Systems (I)

overview

- ▶ the communication channel
- ▶ bandwidth constraints
- ▶ power constraints

a comparison of data rates

► Transatlantic cable:

- 1866: 8 words per minute (≈ 5 bps)
- 1956: AT&T, coax, 48 voice channels (≈ 3 Mbps)
- 2005: Alcatel Tera10, fiber, 8.4 Tbps (8.4×10^{12} bps)
- 2012: fiber, 60 Tbps

► Voiceband modems

- 1950s: Bell 202, 1200 bps
- 1990s: V90, 56Kbps
- 2008: ADSL2+, 24Mbps

a comparison of data rates

► Transatlantic cable:

- 1866: 8 words per minute (≈ 5 bps)
- 1956: AT&T, coax, 48 voice channels (≈ 3 Mbps)
- 2005: Alcatel Tera10, fiber, 8.4 Tbps (8.4×10^{12} bps)
- 2012: fiber, 60 Tbps

► Voiceband modems

- 1950s: Bell 202, 1200 bps
- 1990s: V90, 56Kbps
- 2008: ADSL2+, 24Mbps

Success factors for digital communications

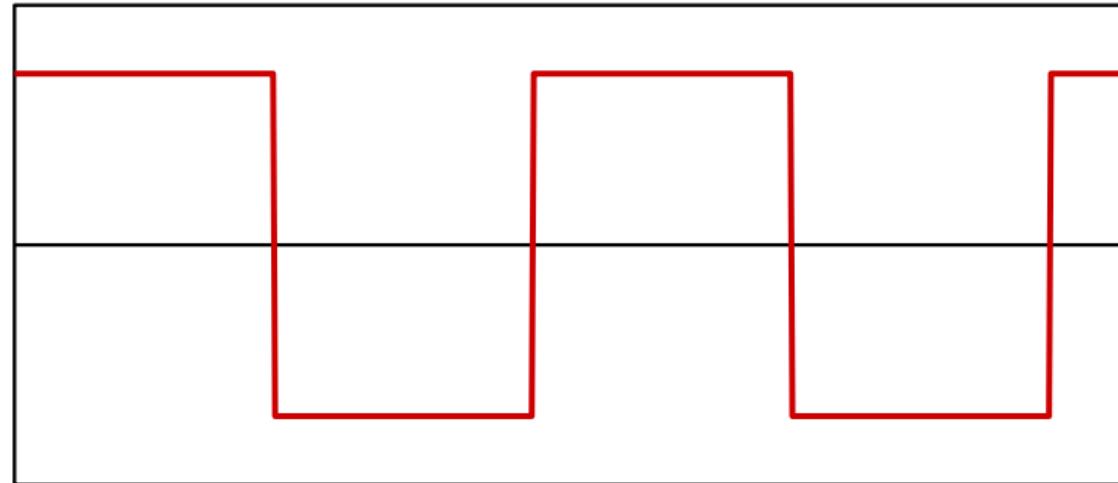
1. power of the digital paradigm
2. natural integration with information theory
3. hardware advancement

Success factors for digital communications

1) power of the DSP paradigm:

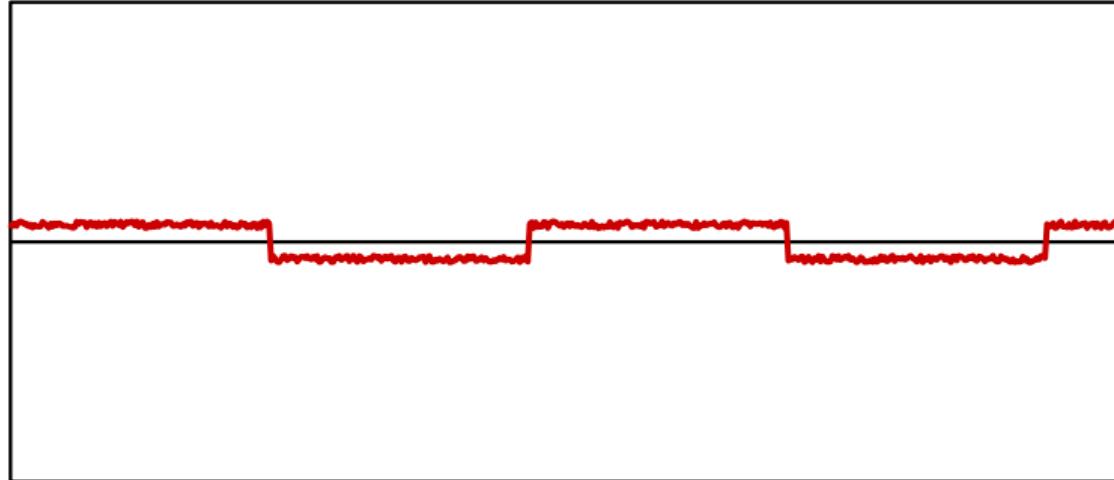
- ▶ integers are “easy” to regenerate
- ▶ good phase control
- ▶ adaptive algorithms

Regenerating signals



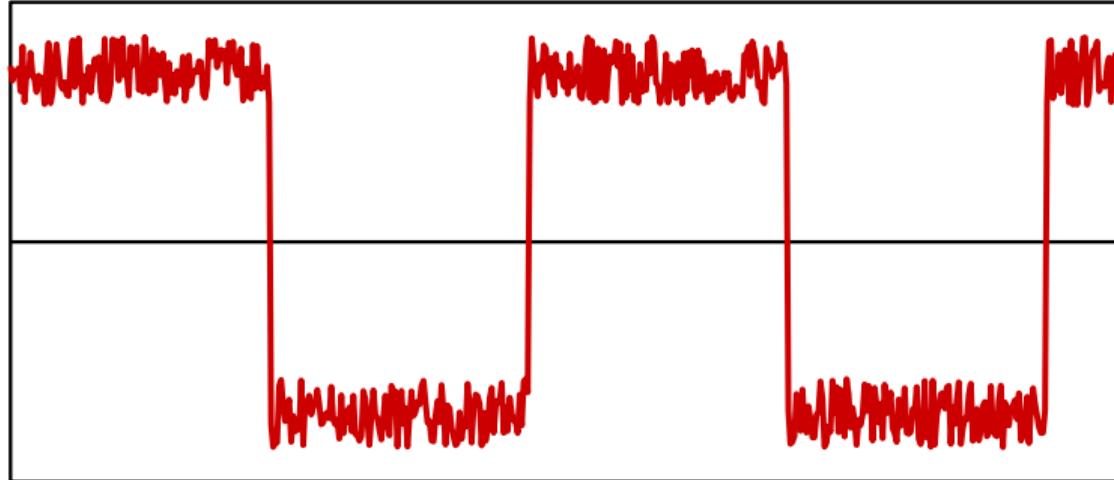
$$x(t)$$

Regenerating signals



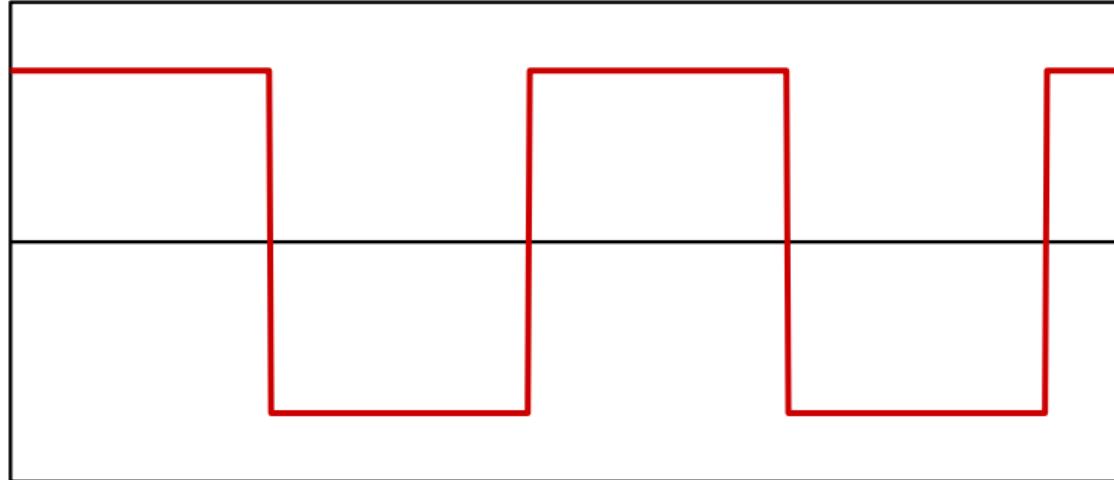
$$x(t)/G + \sigma(t)$$

Regenerating signals



$$G[x(t)/G + \sigma(t)] = x(t) + G\sigma(t)$$

Regenerating signals



$$\hat{x}_1(t) = G \text{sgn}[x(t) + \sigma(t)]$$

Success factors for digital communications

2) algorithmic nature of DSP is a perfect match with information theory:

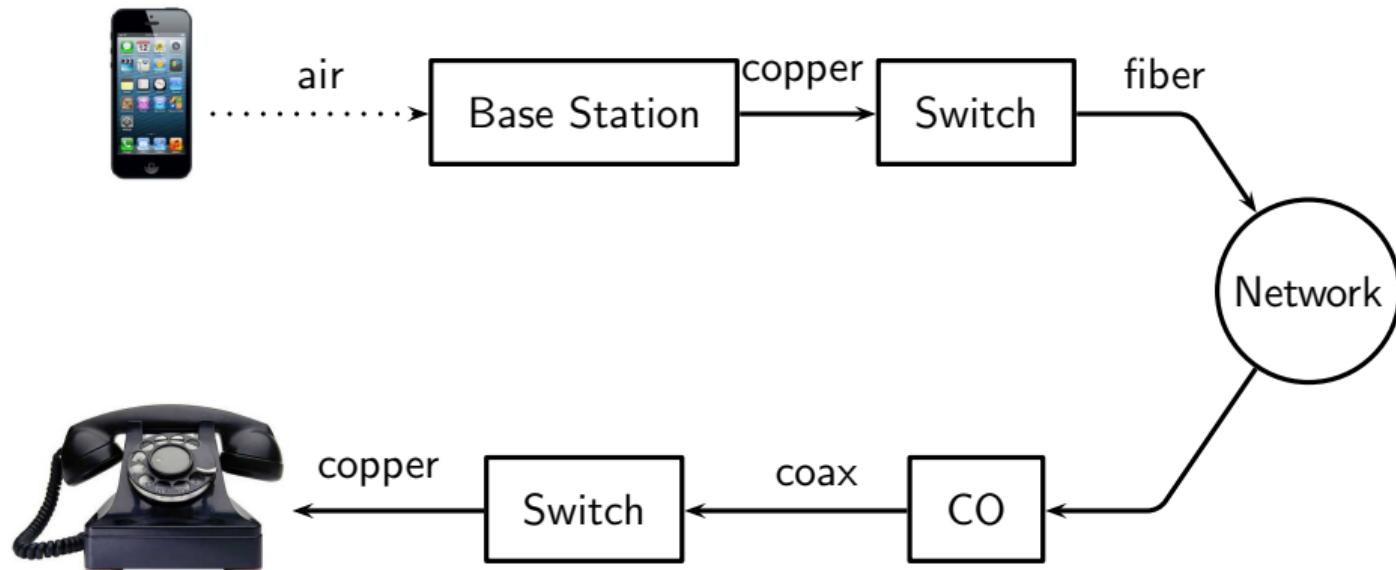
- ▶ error correction (CD's and DVD's)
- ▶ entropy coding (JPEG)

Success factors for digital communications

3) hardware advancement

- ▶ general-purpose platforms
- ▶ miniaturization
- ▶ power efficiency

The many incarnations of a conversation



The analog channel

unescapable “limits” of physical channels:

- ▶ bandwidth constraint
- ▶ power constraint

both constraints will affect the final *capacity* of the channel

The analog channel

unescapable “limits” of physical channels:

- ▶ bandwidth constraint
- ▶ power constraint

both constraints will affect the final *capacity* of the channel

The analog channel

unescapable “limits” of physical channels:

- ▶ bandwidth constraint
- ▶ power constraint

both constraints will affect the final *capacity* of the channel

The analog channel's capacity

maximum amount of information that can be *reliably* delivered over a channel
(bits per second)

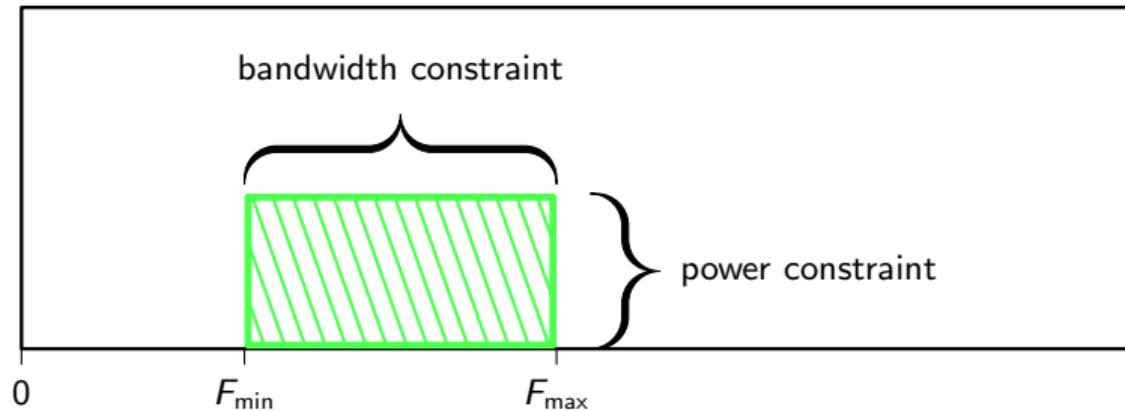
The harsh truth about reliability

we cannot design a perfect (error-free) communication system because of noise

but

we can design a system with arbitrary small error rate (e.g. 10^{-6})

The channel constraints



The design problem

- ▶ transmitted data is a sequence of digital symbols $a[n] \in \mathcal{A}$
- ▶ we will model $a[n]$ as a zero-mean white process
- ▶ we need to transform $a[n]$ into an analog signal $s(t)$ that fulfills both bandwidth and power constraints

Bandwidth vs capacity: intuition

- ▶ we want to transmit a data sequence $a[n]$ over an analog channel
- ▶ we sinc-interpolate $a[n]$ with a period T_s
- ▶ if we make T_s small we can send more symbols per unit of time...
- ▶ ... but the bandwidth of the signal will grow as $1/T_s$

Bandwidth vs capacity: intuition

- ▶ we want to transmit a data sequence $a[n]$ over an analog channel
- ▶ we sinc-interpolate $a[n]$ with a period T_s
- ▶ if we make T_s small we can send more symbols per unit of time...
- ▶ ... but the bandwidth of the signal will grow as $1/T_s$

Bandwidth vs capacity: intuition

- ▶ we want to transmit a data sequence $a[n]$ over an analog channel
- ▶ we sinc-interpolate $a[n]$ with a period T_s
- ▶ if we make T_s small we can send more symbols per unit of time...
- ▶ ... but the bandwidth of the signal will grow as $1/T_s$

Bandwidth vs capacity: intuition

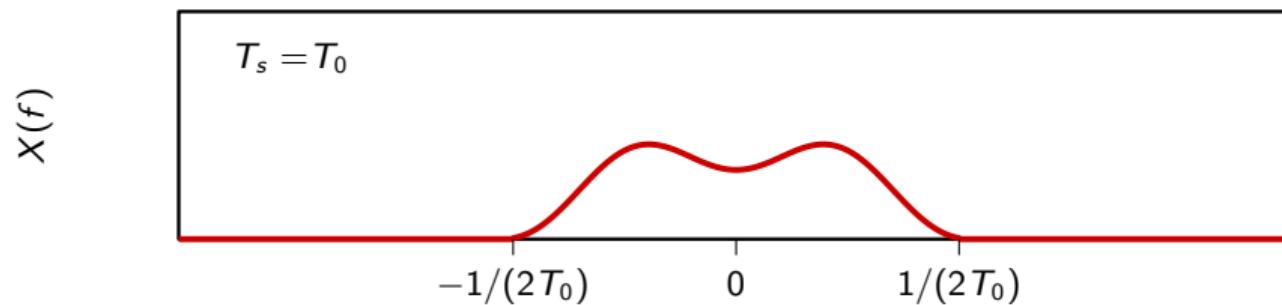
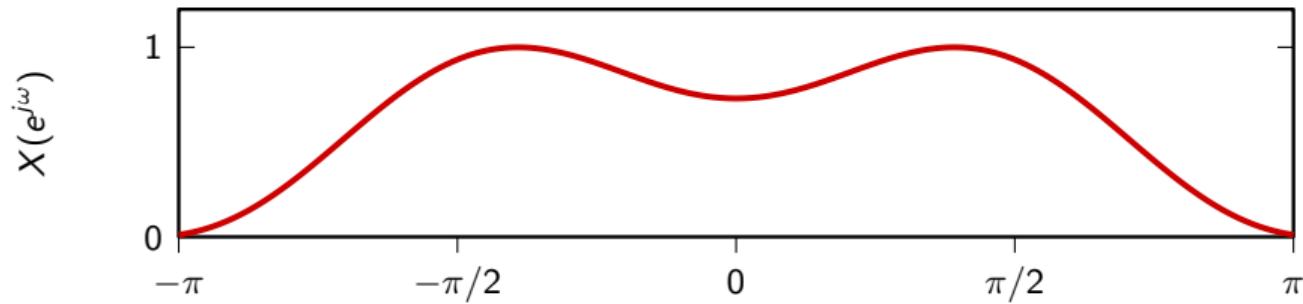
- ▶ we want to transmit a data sequence $a[n]$ over an analog channel
- ▶ we sinc-interpolate $a[n]$ with a period T_s
- ▶ if we make T_s small we can send more symbols per unit of time...
- ▶ ... but the bandwidth of the signal will grow as $1/T_s$

Sinc interpolation

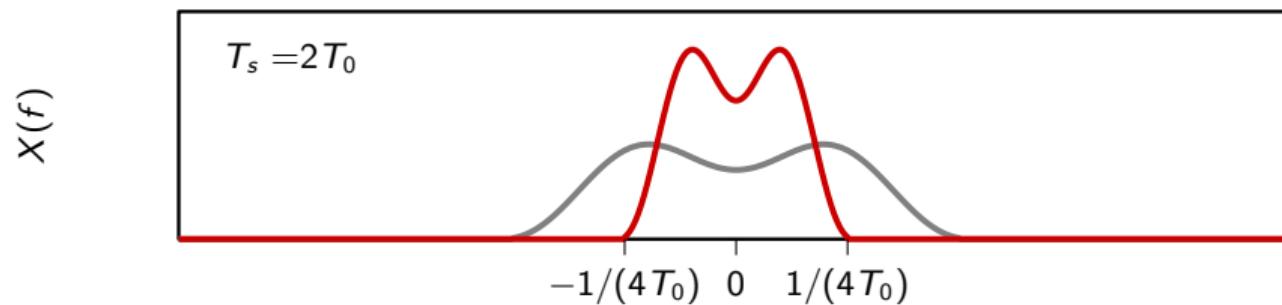
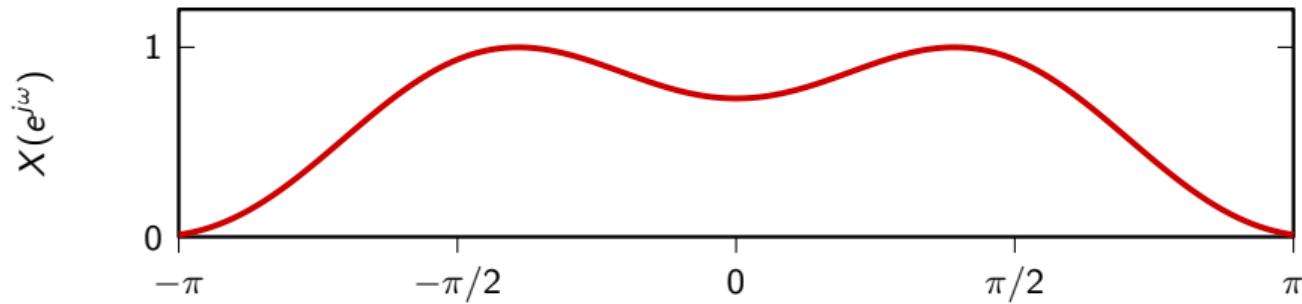
$$x(t) = \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right)$$

$$X(f) = \begin{cases} (1/F_s)X(e^{j2\pi f/F_s}) & \text{for } |f| \leq \frac{F_s}{2} = \frac{1}{2T_s} \\ 0 & \text{otherwise} \end{cases}$$

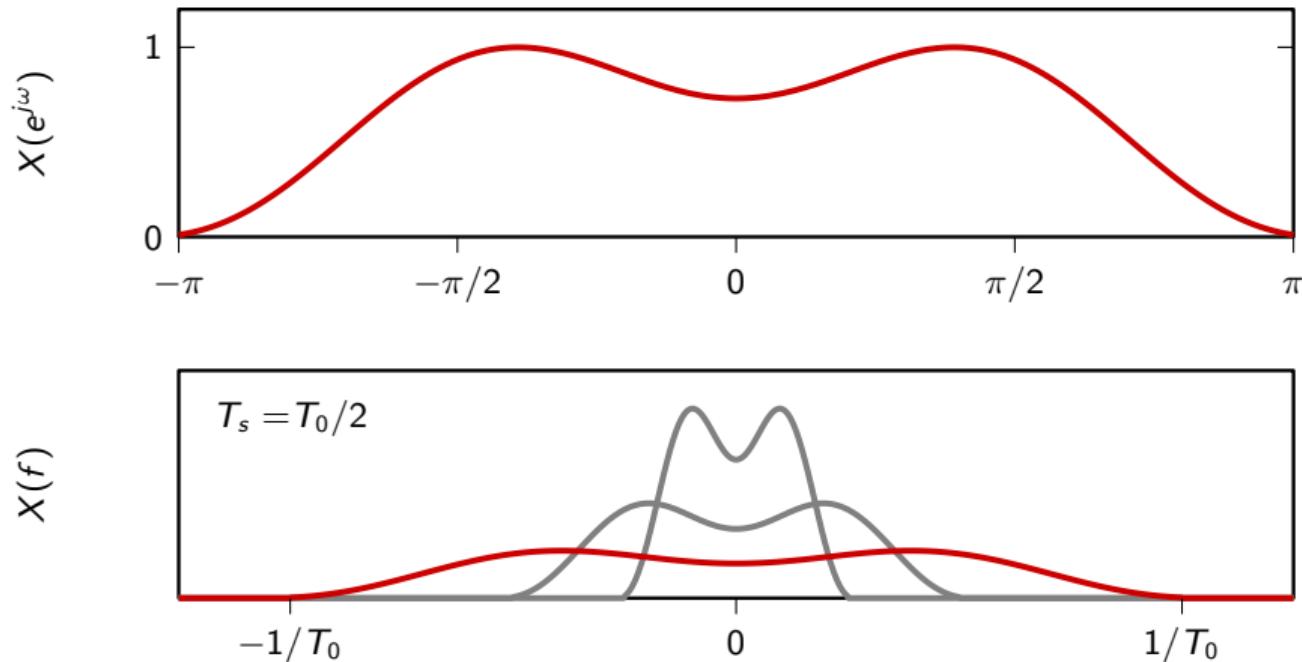
Spectrum of interpolated signals



Spectrum of interpolated signals



Spectrum of interpolated signals



Power vs capacity: intuition

- ▶ number of symbols per second is determined by bandwidth constraint
- ▶ to increase information rate we must increase the number of possible symbols
- ▶ power is proportional to the square of the max symbol
- ▶ to keep power limited we need to pack symbols closer together
- ▶ all channels introduce noise; at the receiver we have to “guess” what was transmitted
- ▶ closer symbols have less *noise margin*, i.e., we lose reliability

Power vs capacity: intuition

- ▶ number of symbols per second is determined by bandwidth constraint
- ▶ to increase information rate we must increase the number of possible symbols
- ▶ power is proportional to the square of the max symbol
- ▶ to keep power limited we need to pack symbols closer together
- ▶ all channels introduce noise; at the receiver we have to “guess” what was transmitted
- ▶ closer symbols have less *noise margin*, i.e., we lose reliability

Power vs capacity: intuition

- ▶ number of symbols per second is determined by bandwidth constraint
- ▶ to increase information rate we must increase the number of possible symbols
- ▶ power is proportional to the square of the max symbol
- ▶ to keep power limited we need to pack symbols closer together
- ▶ all channels introduce noise; at the receiver we have to “guess” what was transmitted
- ▶ closer symbols have less *noise margin*, i.e., we lose reliability

Power vs capacity: intuition

- ▶ number of symbols per second is determined by bandwidth constraint
- ▶ to increase information rate we must increase the number of possible symbols
- ▶ power is proportional to the square of the max symbol
- ▶ to keep power limited we need to pack symbols closer together
- ▶ all channels introduce noise; at the receiver we have to “guess” what was transmitted
- ▶ closer symbols have less *noise margin*, i.e., we lose reliability

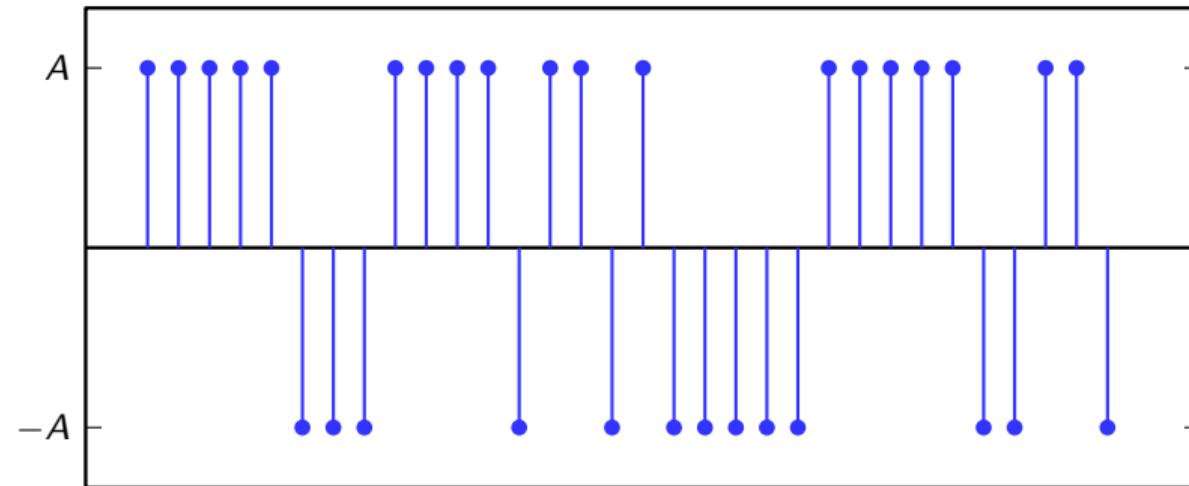
Power vs capacity: intuition

- ▶ number of symbols per second is determined by bandwidth constraint
- ▶ to increase information rate we must increase the number of possible symbols
- ▶ power is proportional to the square of the max symbol
- ▶ to keep power limited we need to pack symbols closer together
- ▶ all channels introduce noise; at the receiver we have to “guess” what was transmitted
- ▶ closer symbols have less *noise margin*, i.e., we lose reliability

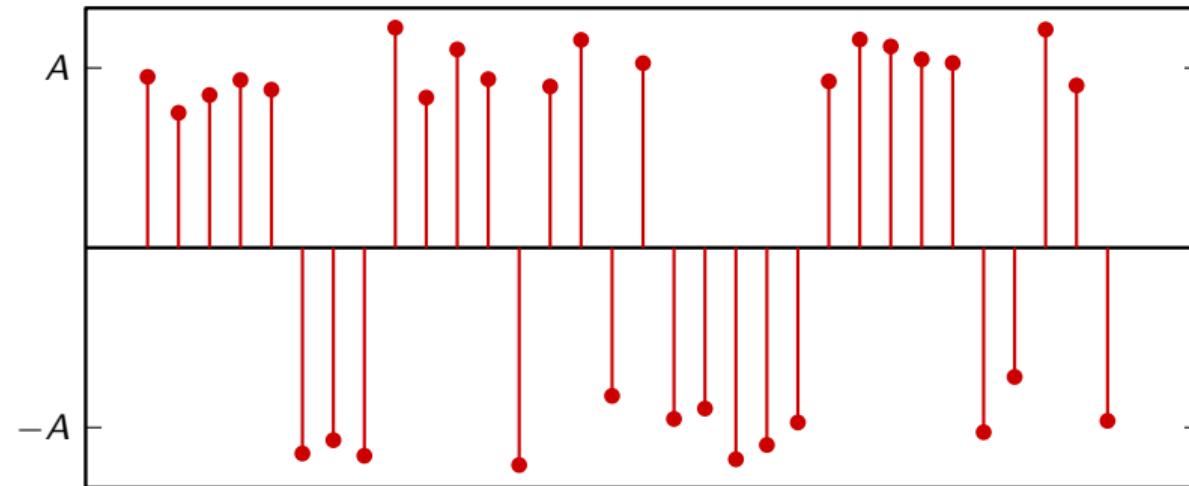
Power vs capacity: intuition

- ▶ number of symbols per second is determined by bandwidth constraint
- ▶ to increase information rate we must increase the number of possible symbols
- ▶ power is proportional to the square of the max symbol
- ▶ to keep power limited we need to pack symbols closer together
- ▶ all channels introduce noise; at the receiver we have to “guess” what was transmitted
- ▶ closer symbols have less *noise margin*, i.e., we lose reliability

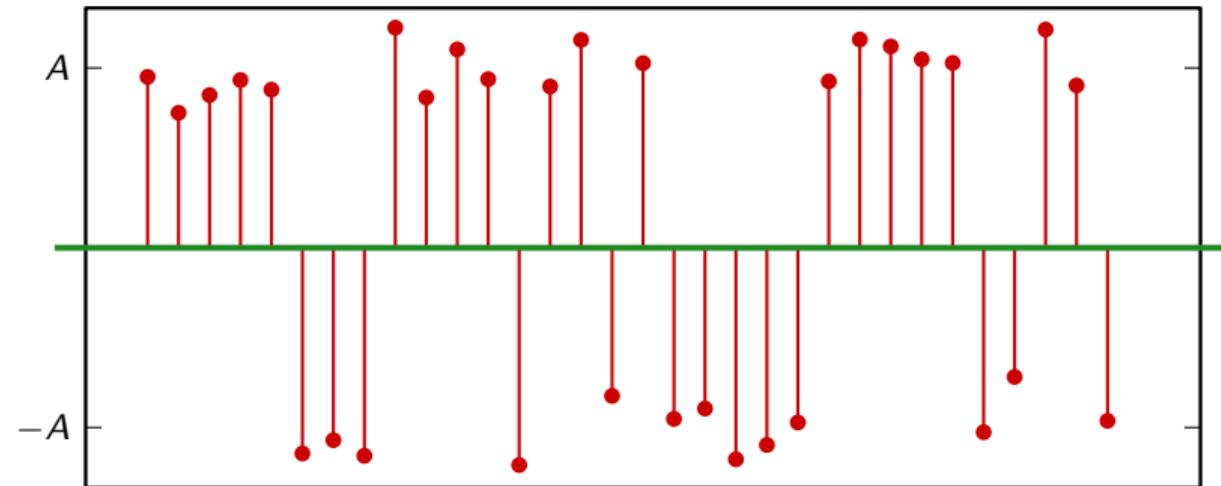
1 bit per symbol



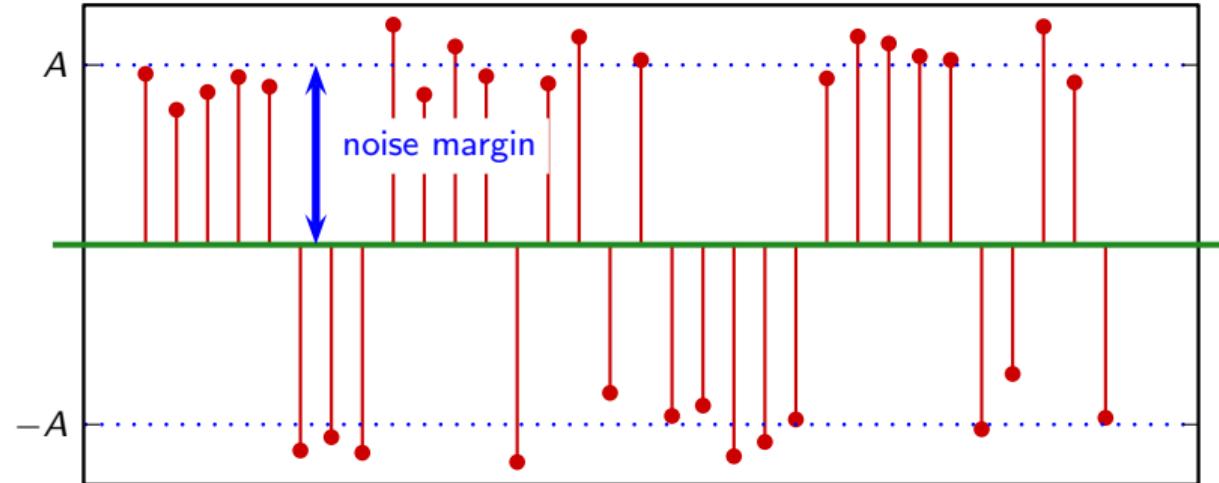
1 bit per symbol



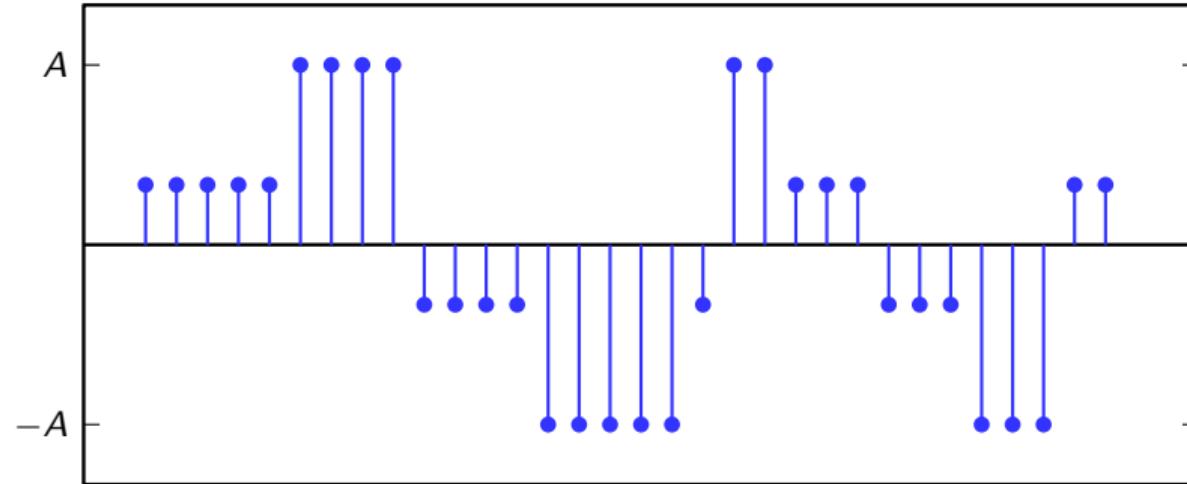
1 bit per symbol



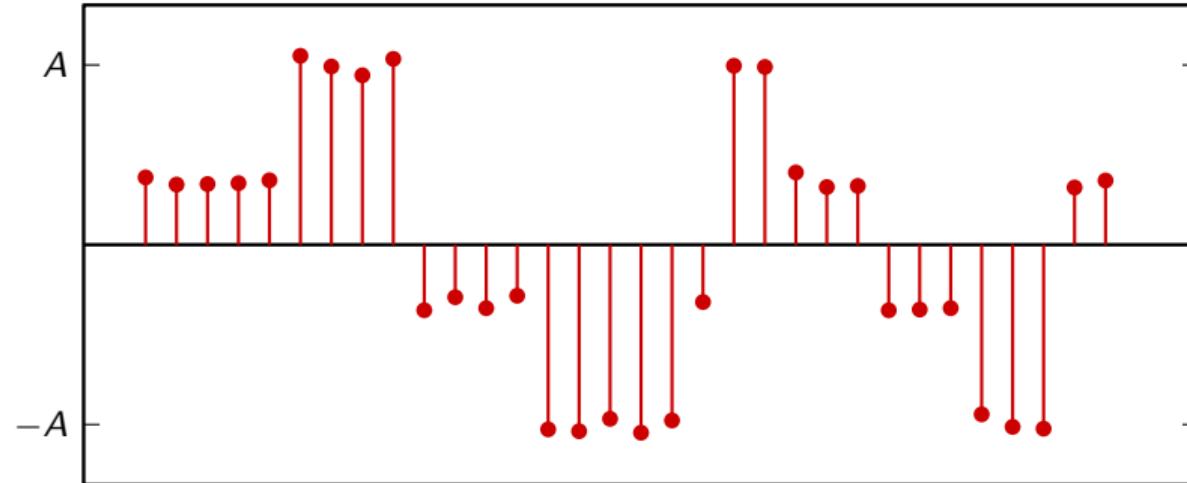
1 bit per symbol



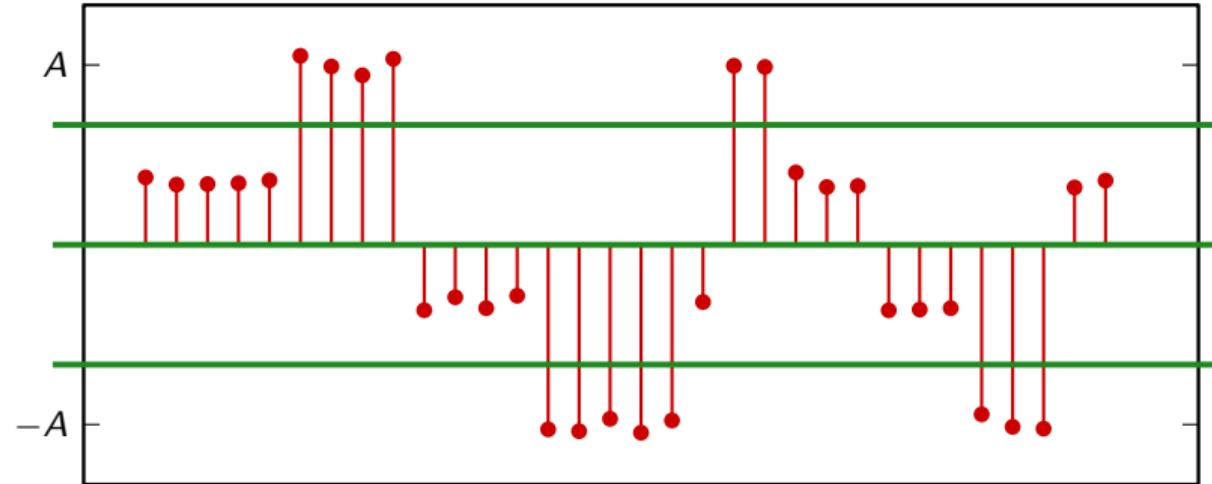
2 bits per symbol



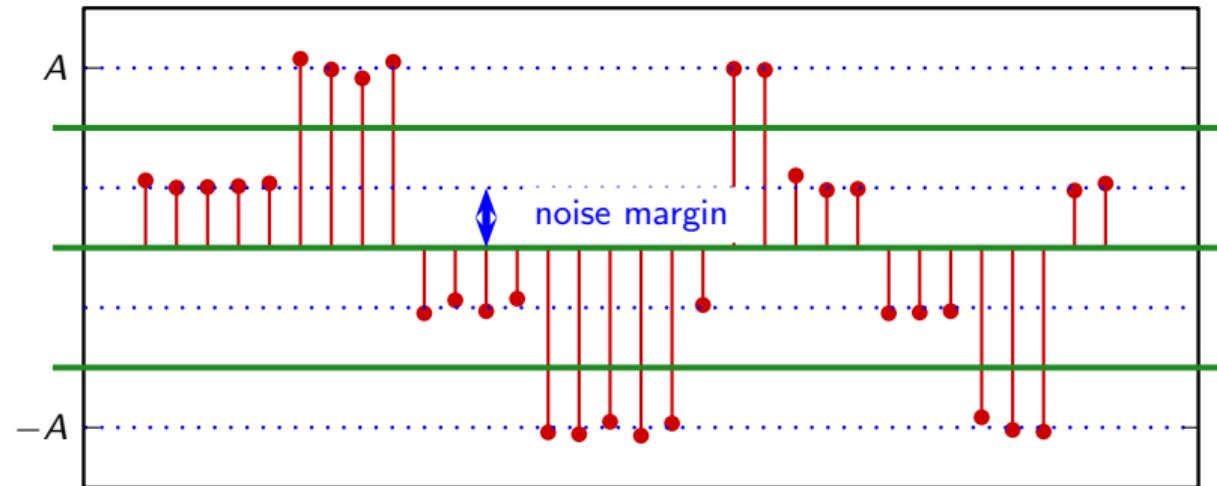
2 bits per symbol



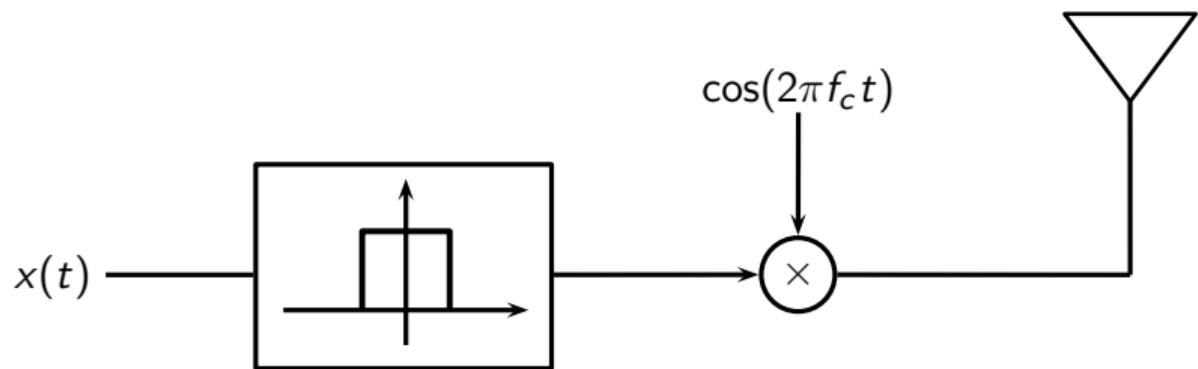
2 bits per symbol



2 bits per symbol



Example: the AM radio channel



Example: the AM radio channel

- ▶ from 530kHz to 1.7MHz
- ▶ each channel is 8KHz
- ▶ power limited by law:
 - daytime/nighttime
 - interference
 - health hazards

Example: the AM radio channel

- ▶ from 530kHz to 1.7MHz
- ▶ each channel is 8KHz
- ▶ power limited by law:
 - daytime/nighttime
 - interference
 - health hazards

Example: the AM radio channel

- ▶ from 530kHz to 1.7MHz
- ▶ each channel is 8KHz
- ▶ power limited by law:
 - daytime/nighttime
 - interference
 - health hazards

Example: the AM radio channel

- ▶ from 530kHz to 1.7MHz
- ▶ each channel is 8KHz
- ▶ power limited by law:
 - daytime/nighttime
 - interference
 - health hazards

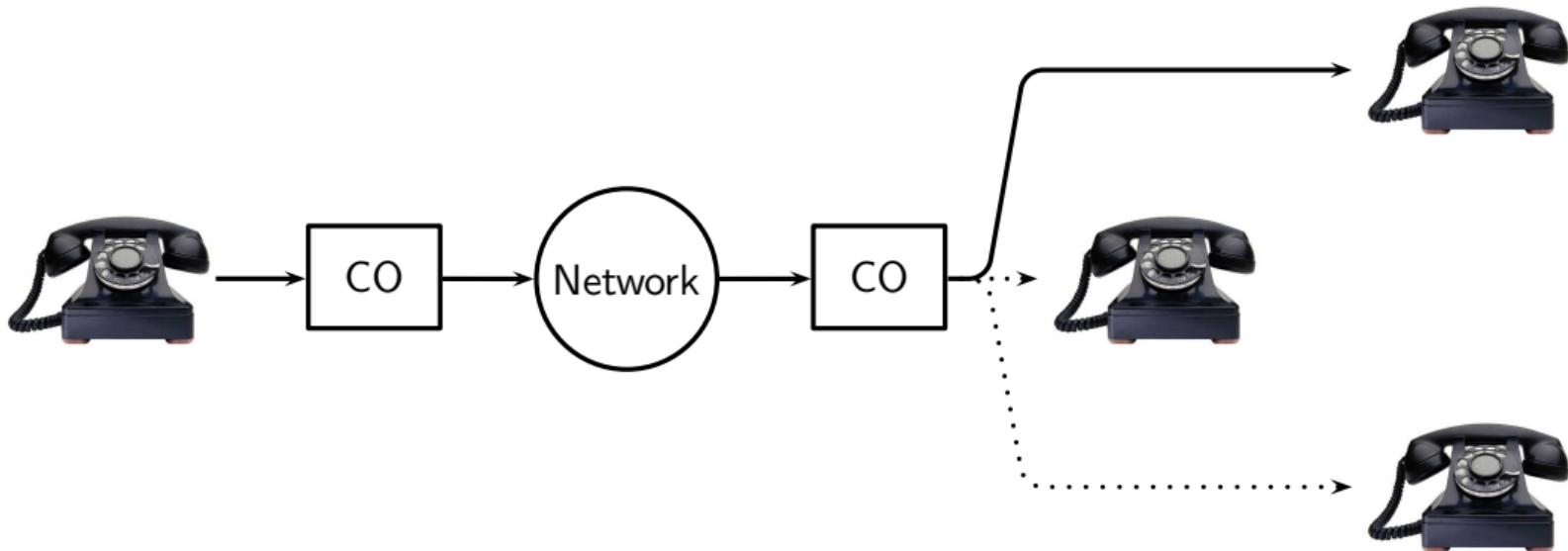
Example: the AM radio channel

- ▶ from 530kHz to 1.7MHz
- ▶ each channel is 8KHz
- ▶ power limited by law:
 - daytime/nighttime
 - interference
 - health hazards

Example: the AM radio channel

- ▶ from 530kHz to 1.7MHz
- ▶ each channel is 8KHz
- ▶ power limited by law:
 - daytime/nighttime
 - interference
 - health hazards

Example: the telephone channel



Example: the telephone channel

- ▶ from around 300Hz to around 3500Hz
- ▶ power limited by law to 0.2-0.7V rms
- ▶ noise is rather low: SNR usually 30dB or more

Example: the telephone channel

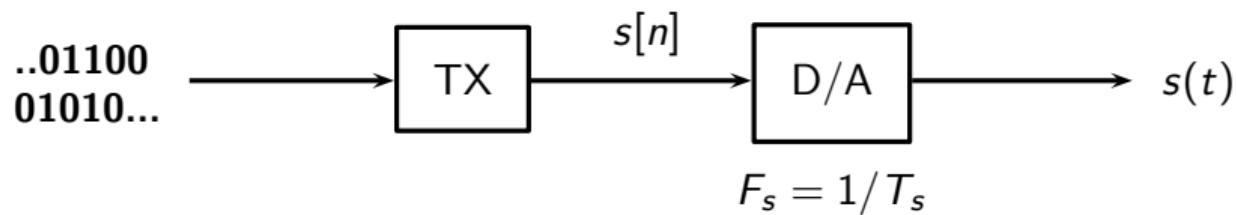
- ▶ from around 300Hz to around 3500Hz
- ▶ power limited by law to 0.2-0.7V rms
- ▶ noise is rather low: SNR usually 30dB or more

Example: the telephone channel

- ▶ from around 300Hz to around 3500Hz
- ▶ power limited by law to 0.2-0.7V rms
- ▶ noise is rather low: SNR usually 30dB or more

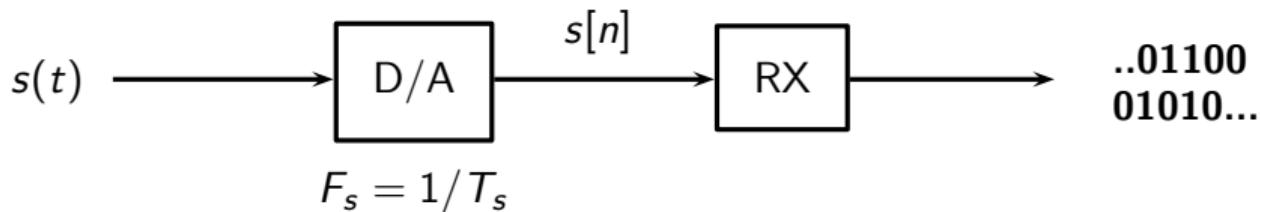
The all-digital paradigm

keep everything digital until we hit the physical channel

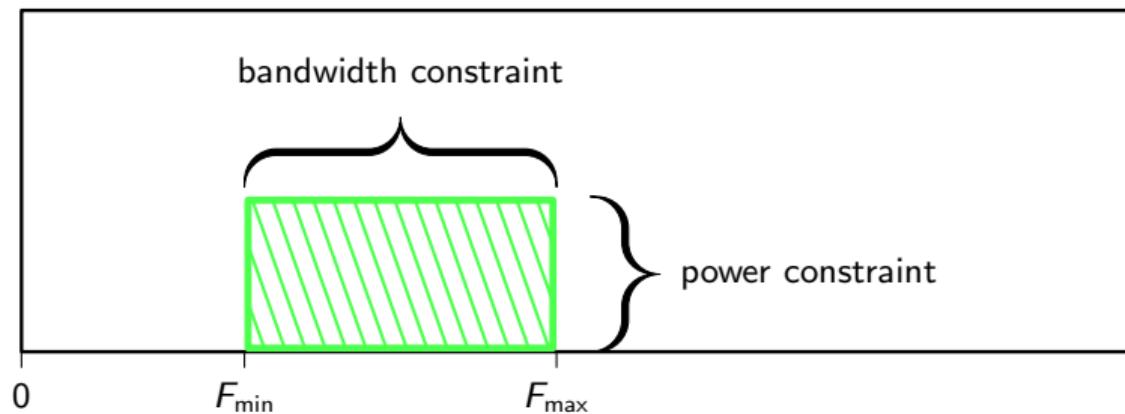


The all-digital paradigm

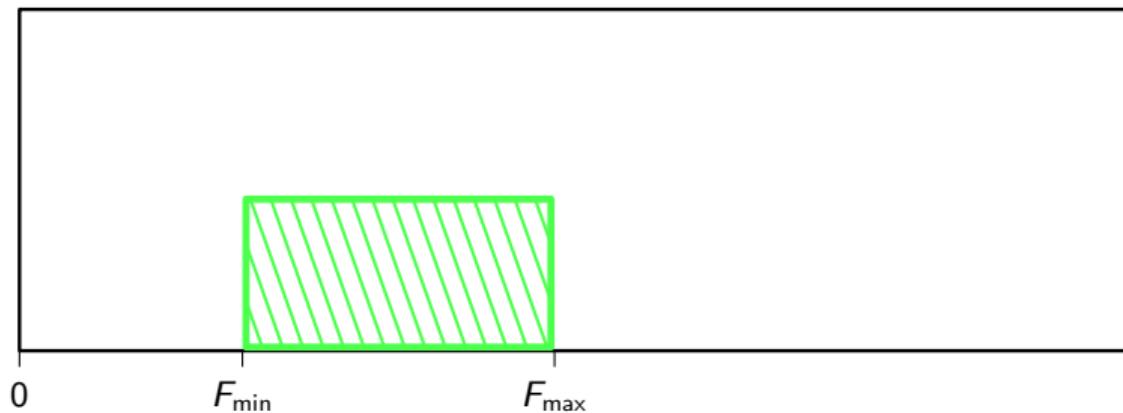
... and of course do the same for the receiver



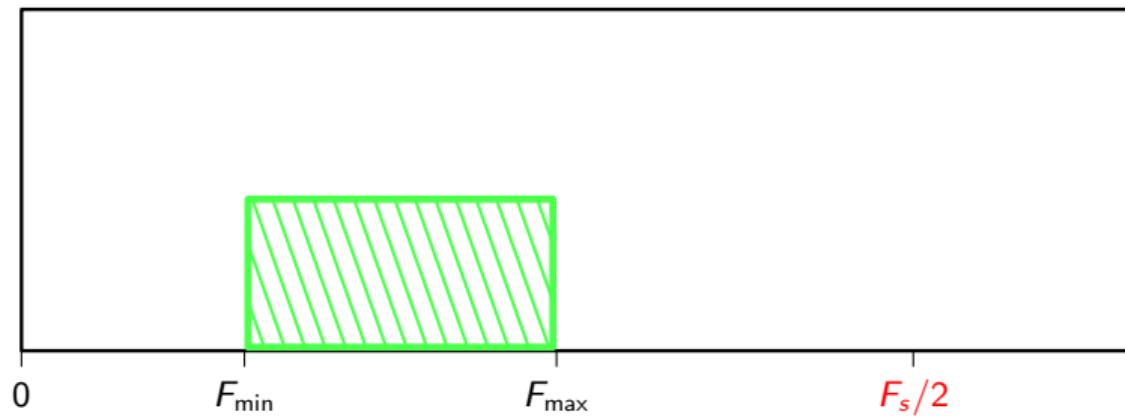
Let's look again at the channel constraints



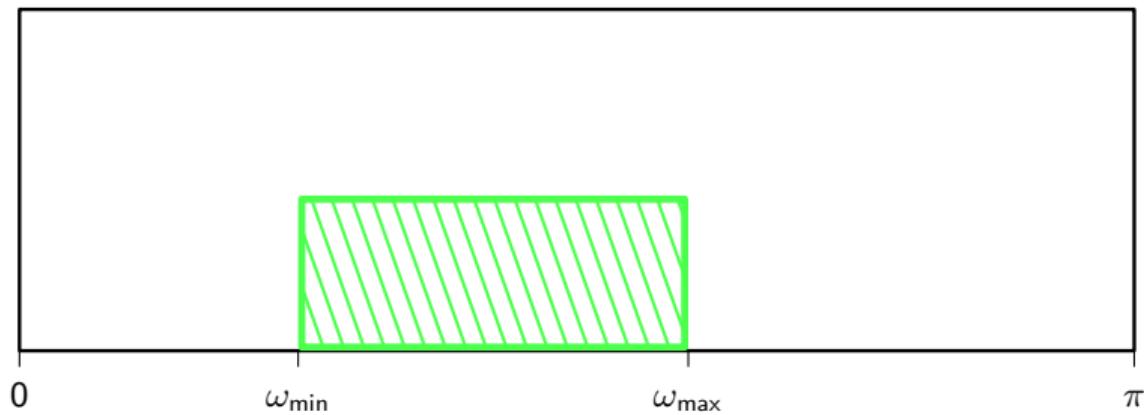
Converting the specs to a digital design



Converting the specs to a digital design



Converting the specs to a digital design



Transmitter design

some working hypotheses:

- ▶ convert the bitstream into a sequence of symbols $a[n]$ via a mapper
- ▶ model $a[n]$ as a white random sequence (add a scrambler on the bitstream to make sure)
- ▶ now we need to convert $a[n]$ into a continuous-time signal within the constraints

Transmitter design

some working hypotheses:

- ▶ convert the bitstream into a sequence of symbols $a[n]$ via a mapper
- ▶ model $a[n]$ as a white random sequence (add a scrambler on the bitstream to make sure)
- ▶ now we need to convert $a[n]$ into a continuous-time signal within the constraints

Transmitter design

some working hypotheses:

- ▶ convert the bitstream into a sequence of symbols $a[n]$ via a mapper
- ▶ model $a[n]$ as a white random sequence (add a scrambler on the bitstream to make sure)
- ▶ now we need to convert $a[n]$ into a continuous-time signal within the constraints

Transmitter design

some working hypotheses:

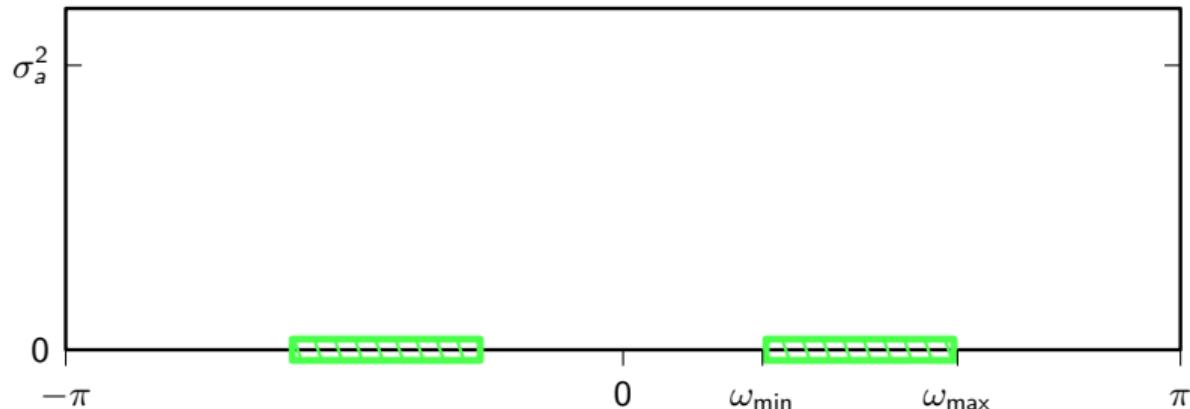
- ▶ convert the bitstream into a sequence of symbols $a[n]$ via a mapper
- ▶ model $a[n]$ as a white random sequence (add a scrambler on the bitstream to make sure)
- ▶ now we need to convert $a[n]$ into a continuous-time signal within the constraints



the bandwidth constraint

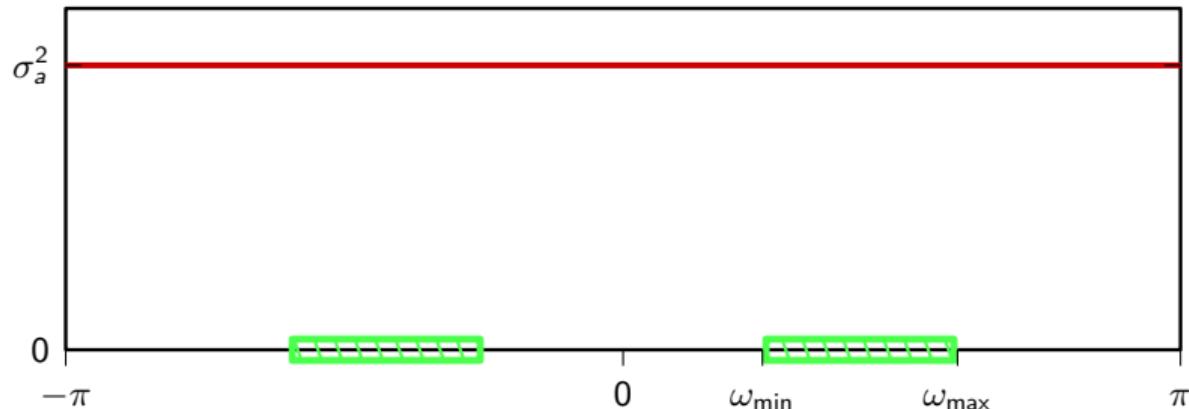
First problem: the bandwidth constraint

$a[n]$ white: $P_a(e^{j\omega}) = \sigma_a^2$



First problem: the bandwidth constraint

$$a[n] \text{ white: } P_a(e^{j\omega}) = \sigma_a^2$$



Shaping the bandwidth

Our problem:

- ▶ bandwidth constraint requires us to control the spectral support of a signal
- ▶ we need to be able to “shrink” the support of a full-band signal
- ▶ the answer is *multirate* techniques

Shaping the bandwidth

Our problem:

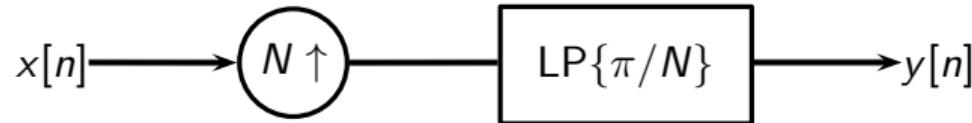
- ▶ bandwidth constraint requires us to control the spectral support of a signal
- ▶ we need to be able to “shrink” the support of a full-band signal
- ▶ the answer is *multirate* techniques

Shaping the bandwidth

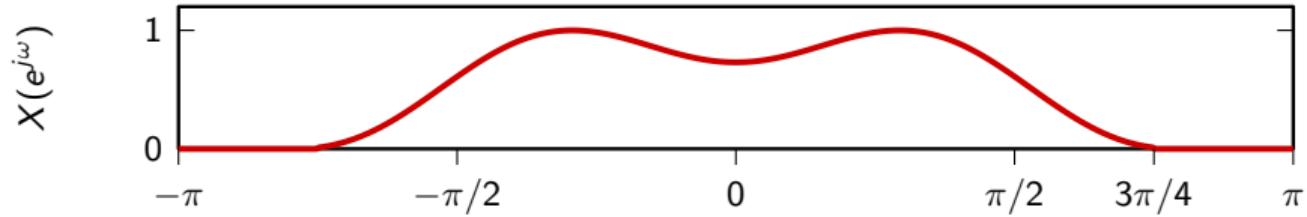
Our problem:

- ▶ bandwidth constraint requires us to control the spectral support of a signal
- ▶ we need to be able to “shrink” the support of a full-band signal
- ▶ the answer is *multirate* techniques

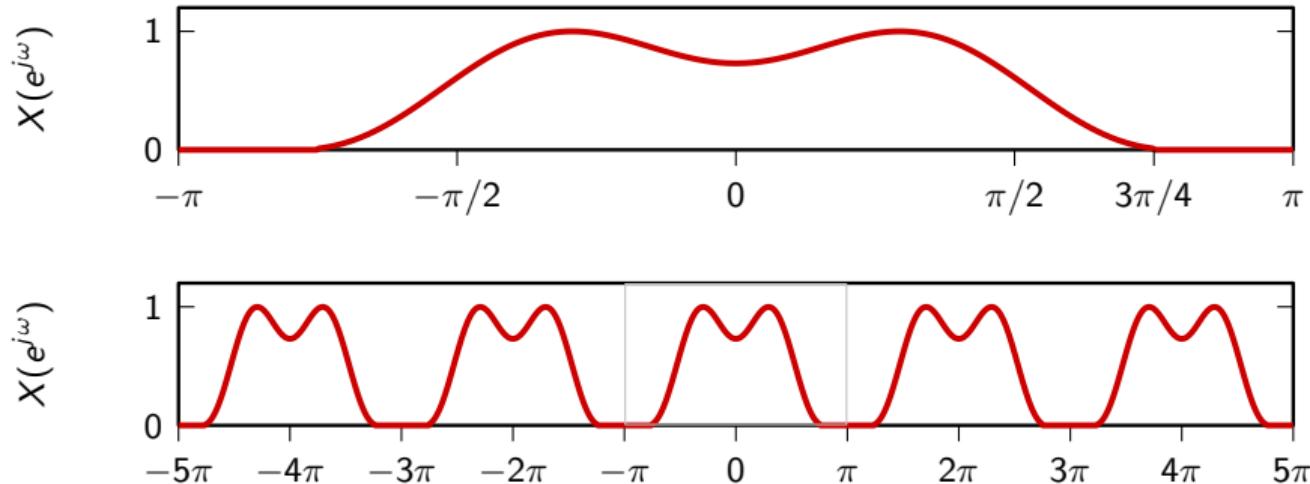
ideal digital interpolator



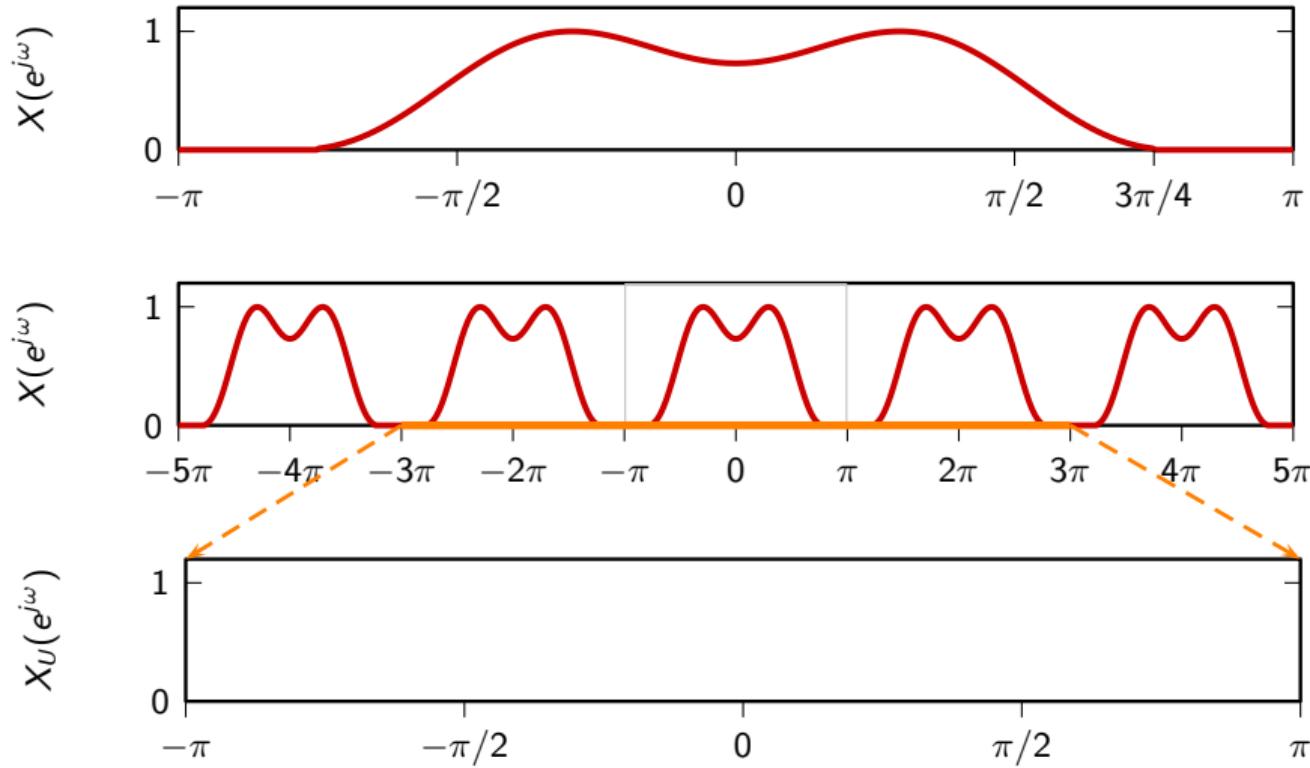
Upsampling by $K = 3$



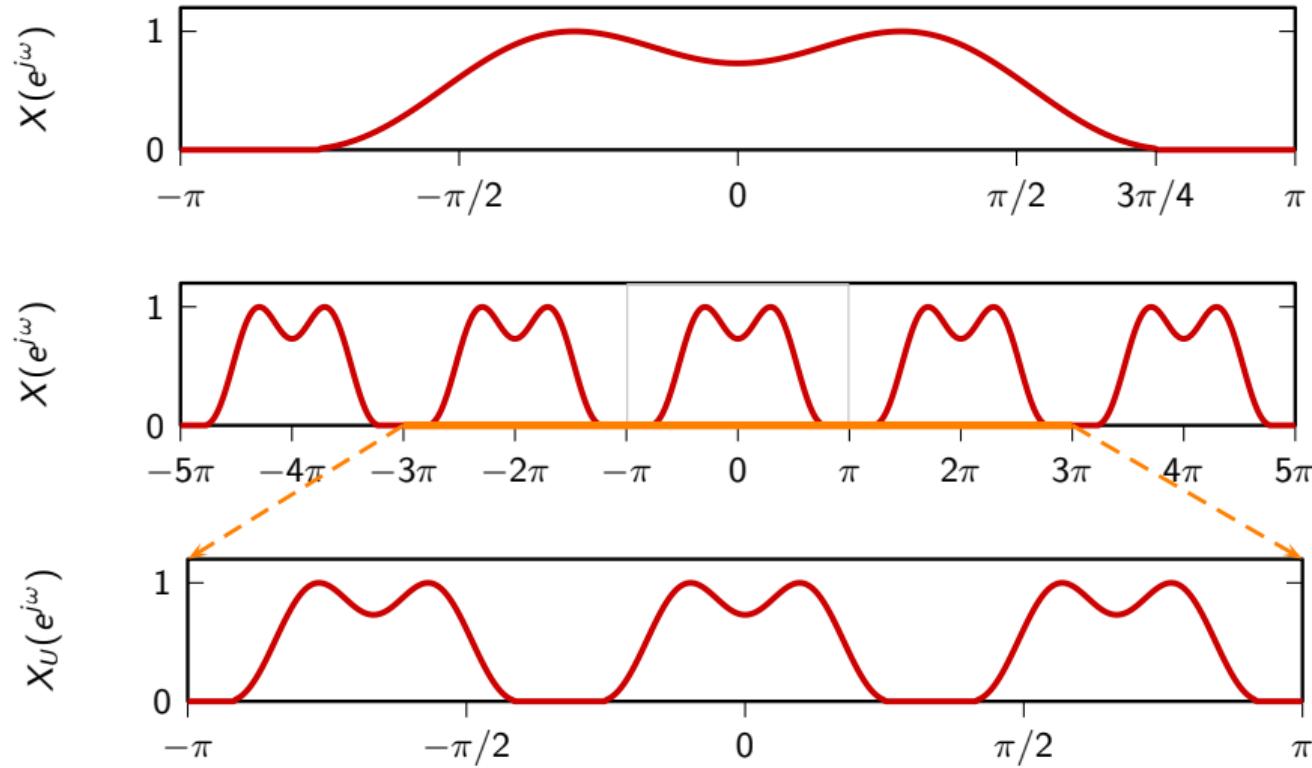
Upsampling by $K = 3$



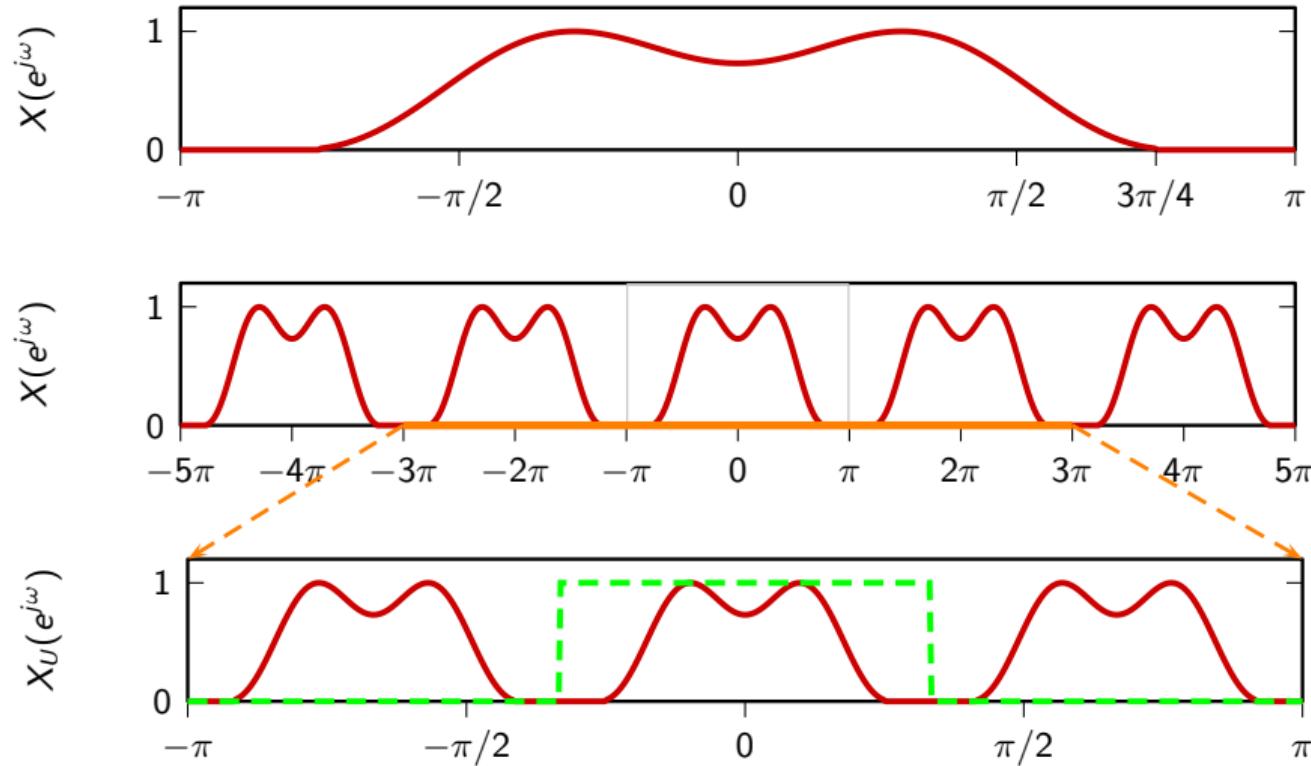
Upsampling by $K = 3$



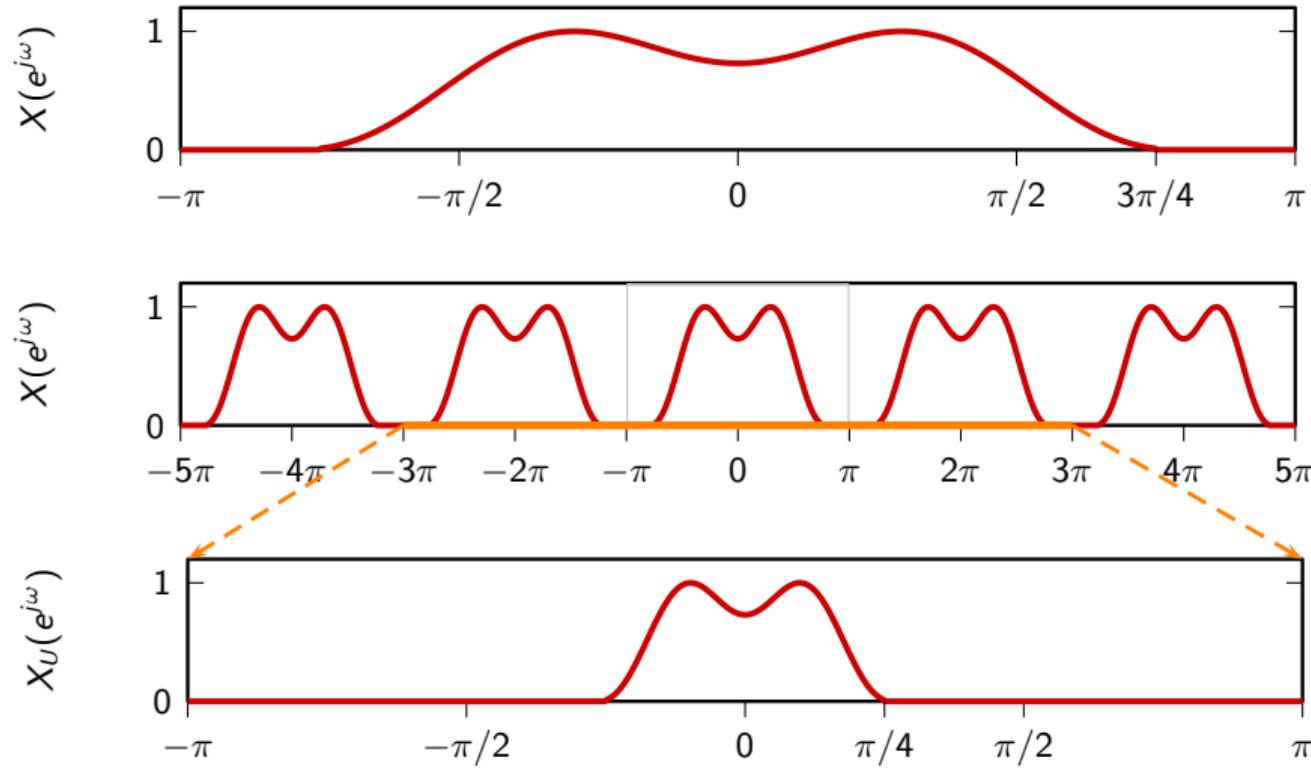
Upsampling by $K = 3$



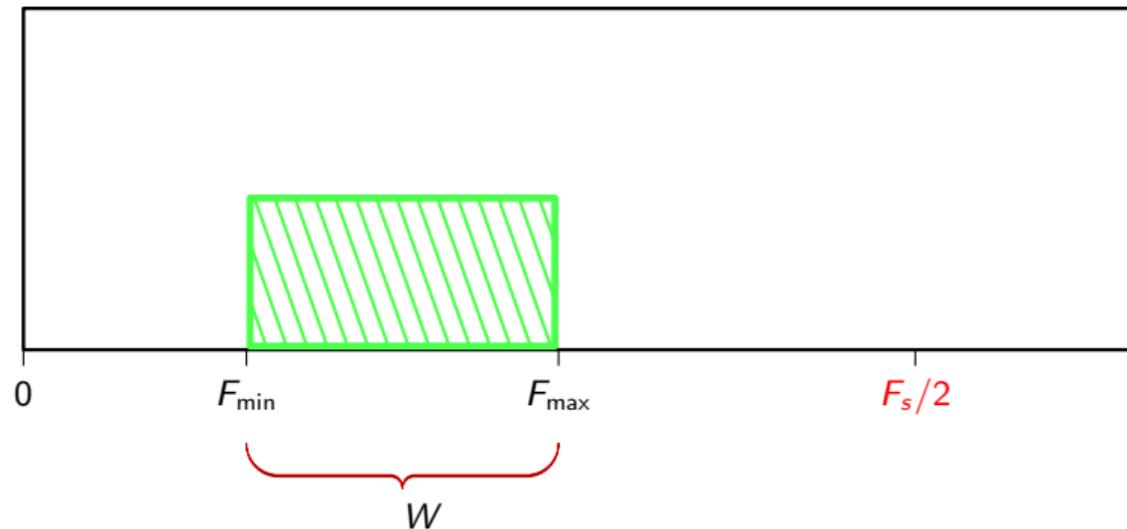
Upsampling by $K = 3$



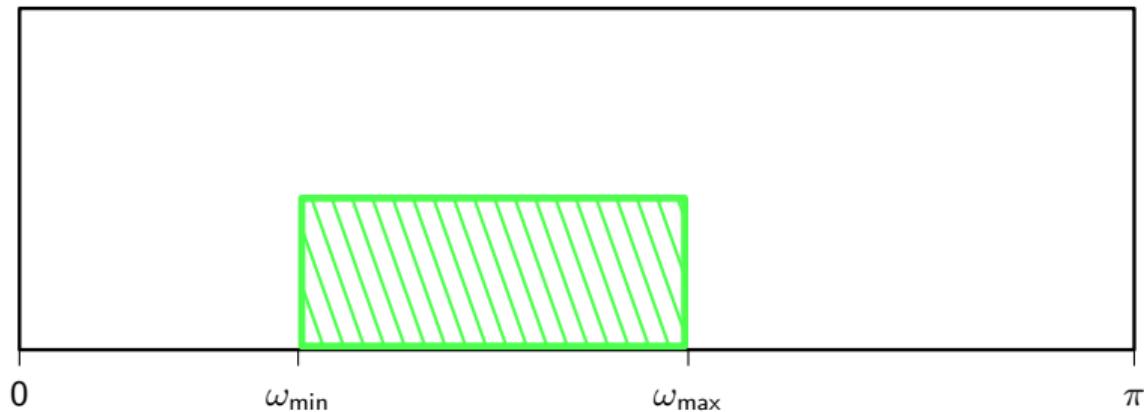
Upsampling by $K = 3$



Fulfilling the bandwidth constraint



Fulfilling the bandwidth constraint



Here's a neat trick

let $W = F_{\max} - F_{\min}$; pick F_s so that:

- ▶ $F_s > 2F_{\max}$ (obviously)
 - ▶ $F_s = KW, K \in \mathbb{N}$
-
- ▶ $\omega_{\max} - \omega_{\min} = 2\pi \frac{W}{F_s} = \frac{2\pi}{K}$
 - ▶ we can simply upsample by K

Here's a neat trick

let $W = F_{\max} - F_{\min}$; pick F_s so that:

- ▶ $F_s > 2F_{\max}$ (obviously)
 - ▶ $F_s = KW, K \in \mathbb{N}$
-
- ▶ $\omega_{\max} - \omega_{\min} = 2\pi \frac{W}{F_s} = \frac{2\pi}{K}$
 - ▶ we can simply upsample by K

Here's a neat trick

let $W = F_{\max} - F_{\min}$; pick F_s so that:

- ▶ $F_s > 2F_{\max}$ (obviously)
 - ▶ $F_s = KW, K \in \mathbb{N}$
-
- ▶ $\omega_{\max} - \omega_{\min} = 2\pi \frac{W}{F_s} = \frac{2\pi}{K}$
 - ▶ we can simply upsample by K

Here's a neat trick

let $W = F_{\max} - F_{\min}$; pick F_s so that:

- ▶ $F_s > 2F_{\max}$ (obviously)
 - ▶ $F_s = KW, K \in \mathbb{N}$
-
- ▶ $\omega_{\max} - \omega_{\min} = 2\pi \frac{W}{F_s} = \frac{2\pi}{K}$
 - ▶ we can simply upsample by K

Baud rate

- ▶ upsampling does not change the *data rate*, only the sample rate
- ▶ we produce (and transmit) W symbols per second
- ▶ W is sometimes called the Baud rate of the system and is equal to the available bandwidth

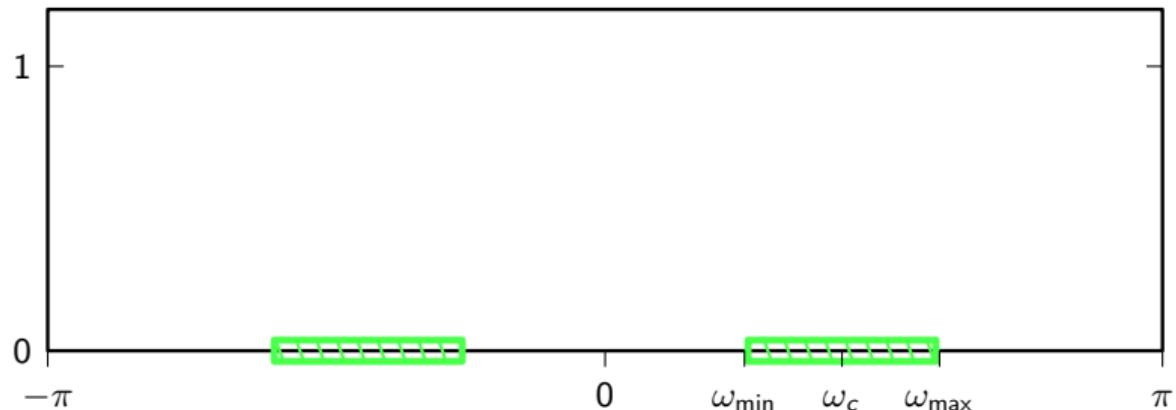
Baud rate

- ▶ upsampling does not change the *data* rate, only the sample rate
- ▶ we produce (and transmit) W symbols per second
- ▶ W is sometimes called the Baud rate of the system and is equal to the available bandwidth

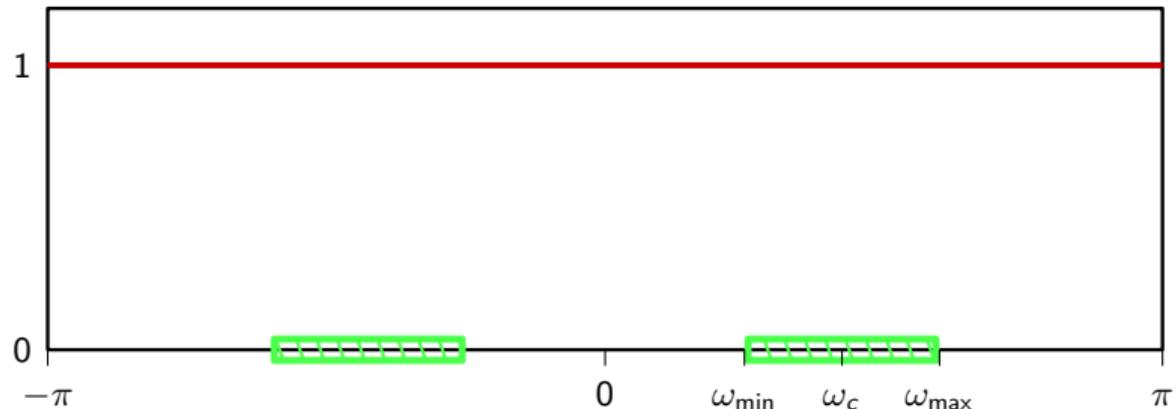
Baud rate

- ▶ upsampling does not change the *data* rate, only the sample rate
- ▶ we produce (and transmit) W symbols per second
- ▶ W is sometimes called the Baud rate of the system and is equal to the available bandwidth

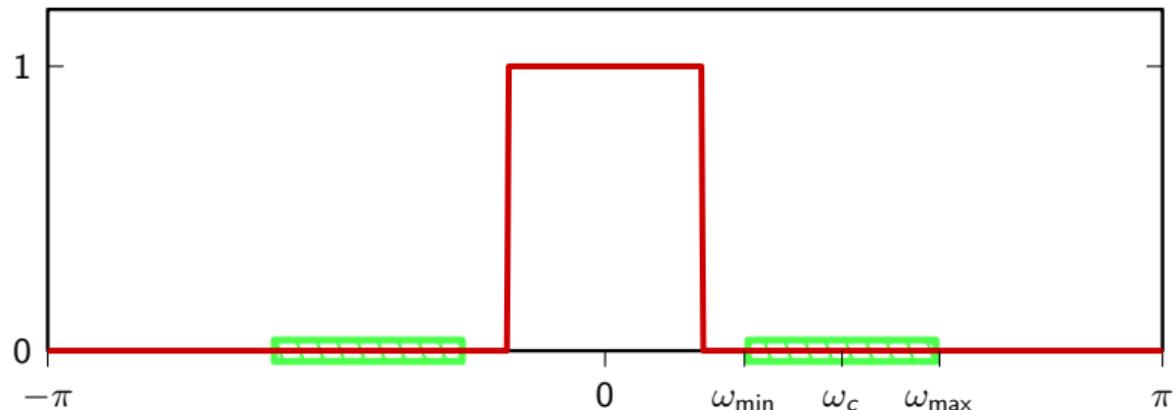
Fulfilling the bandwidth constraint



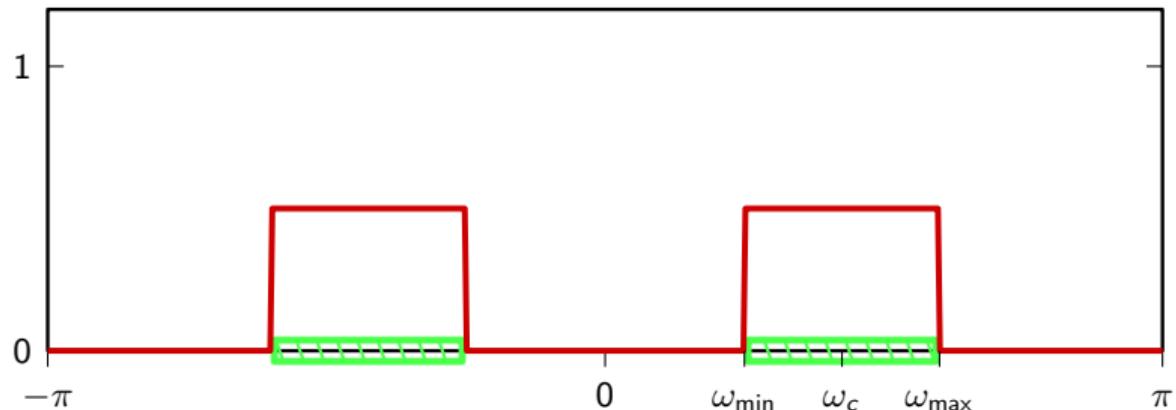
Fulfilling the bandwidth constraint



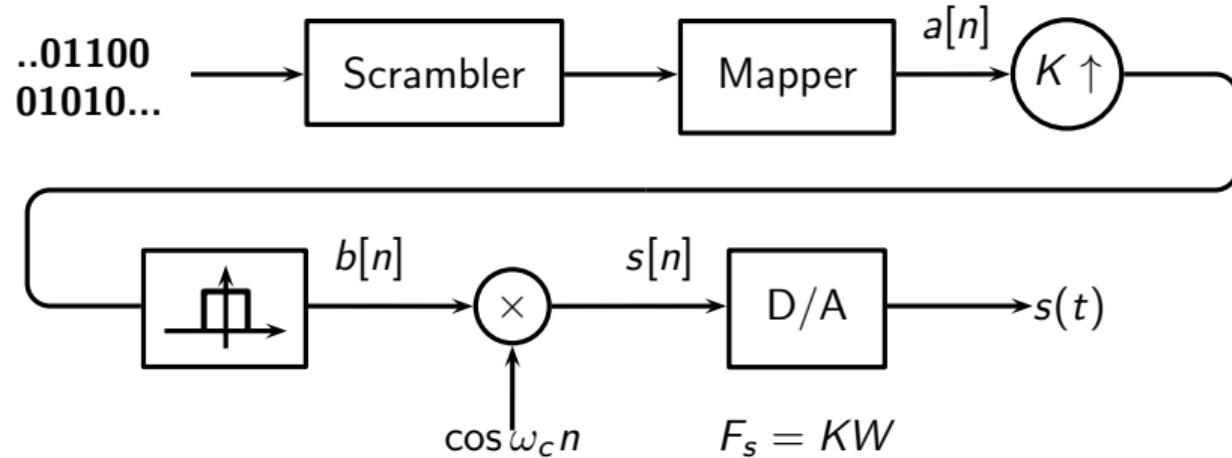
Fulfilling the bandwidth constraint



Fulfilling the bandwidth constraint



Transmitter design, continued

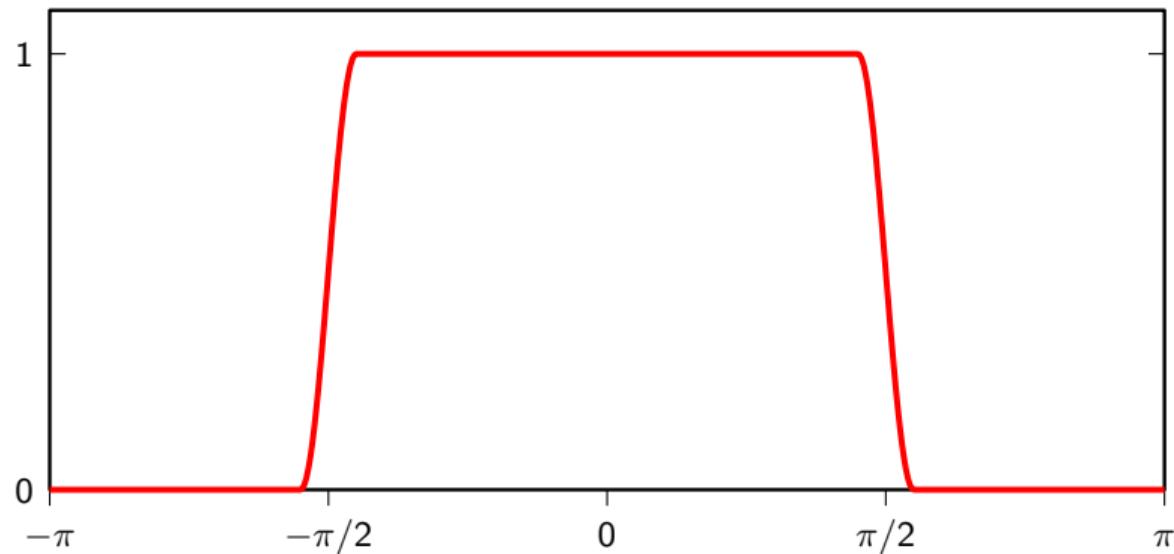


Raised Cosine

$$H_{K,\beta}(e^{j\omega}) = \begin{cases} 1 & |\omega| \leq \frac{\pi(1-\beta)}{K} \\ \frac{1}{2} \left(1 + \cos \left(\frac{K|\omega| - (1-\beta)\pi}{\beta} \right) \right) & \frac{\pi(1-\beta)}{K} \leq |\omega| \leq \frac{\pi(1+\beta)}{K} \\ 0 & |\omega| > \frac{\pi(1+\beta)}{K} \end{cases}$$

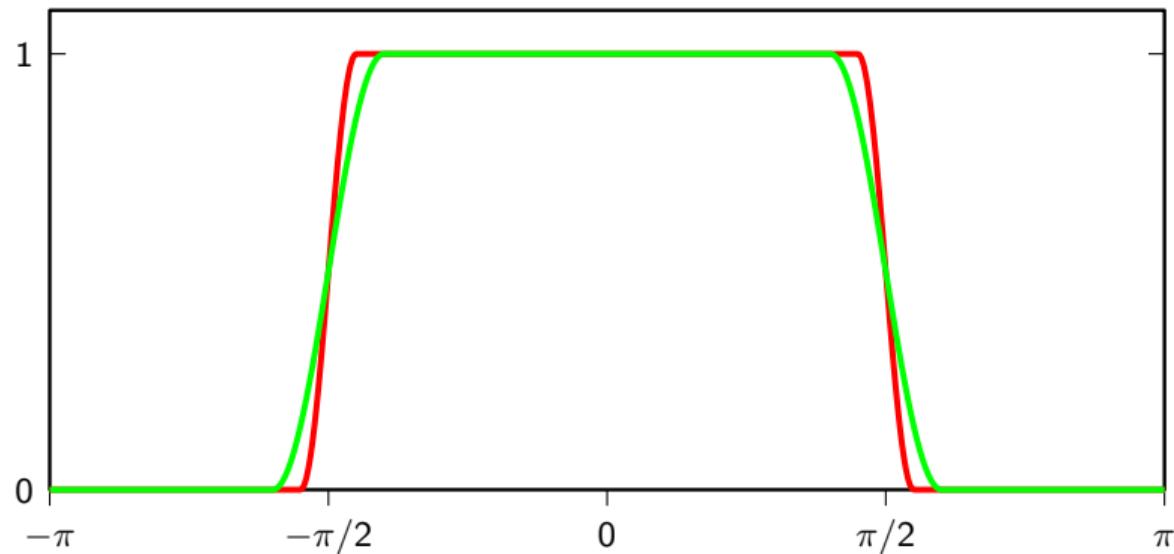
Raised Cosine

$$H_{2,\beta}(e^{j\omega})$$



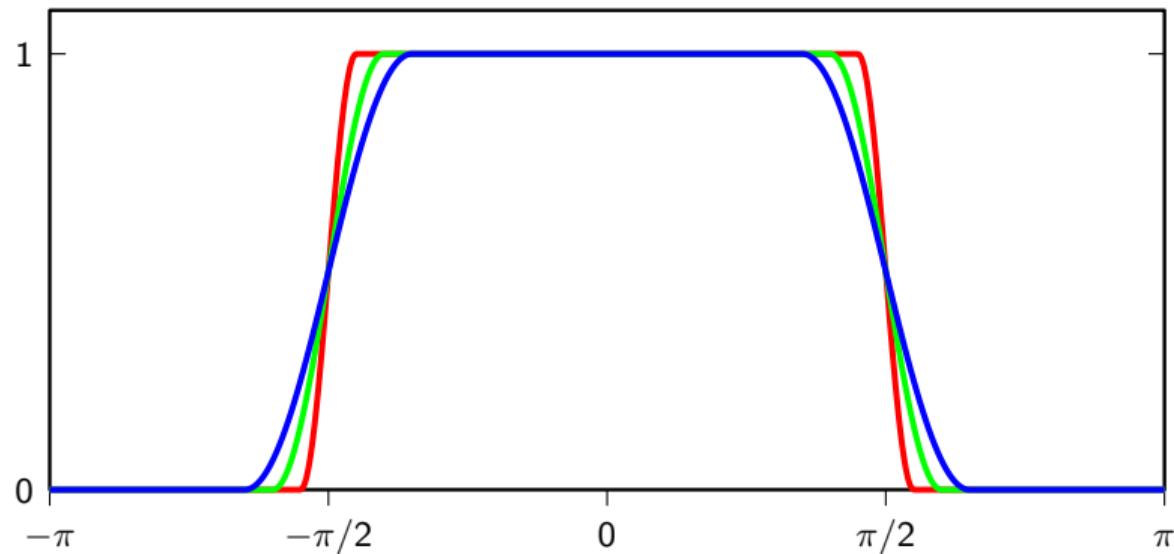
Raised Cosine

$$H_{2,\beta}(e^{j\omega})$$



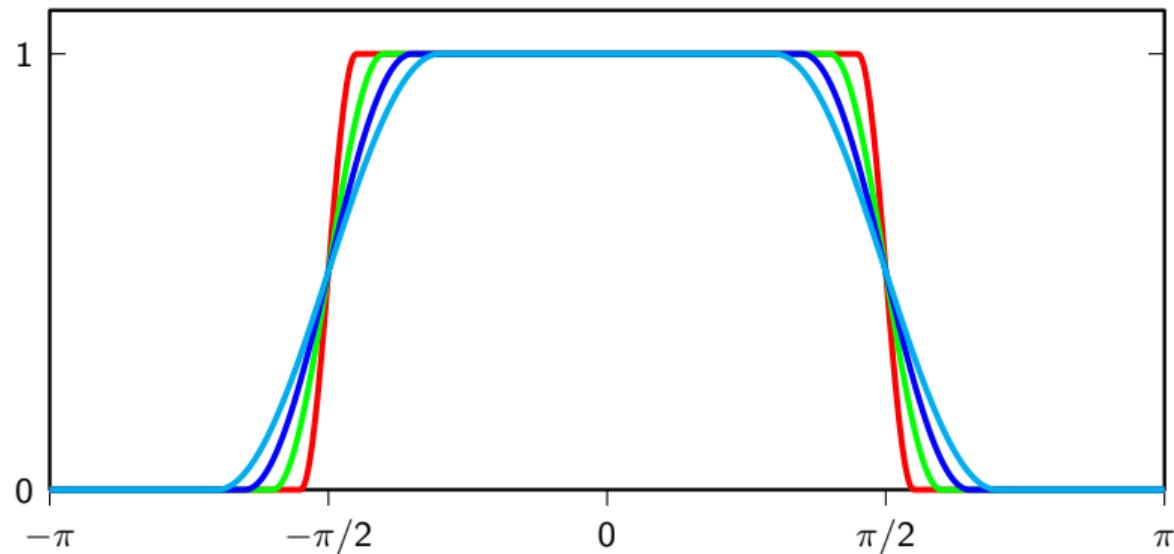
Raised Cosine

$$H_{2,\beta}(e^{j\omega})$$



Raised Cosine

$$H_{2,\beta}(e^{j\omega})$$



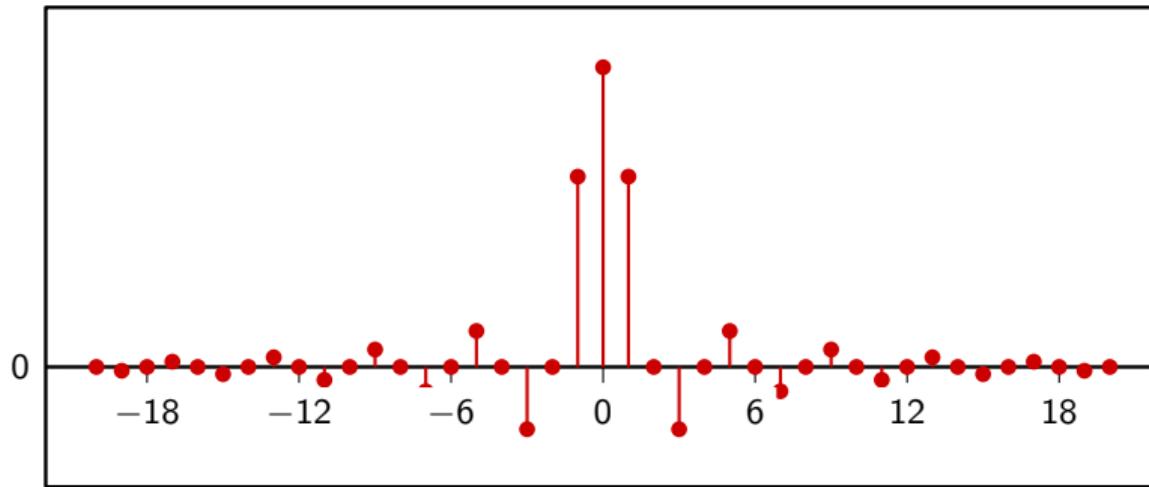
Raised Cosine

$$h_{K,\beta}[nK] = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}$$

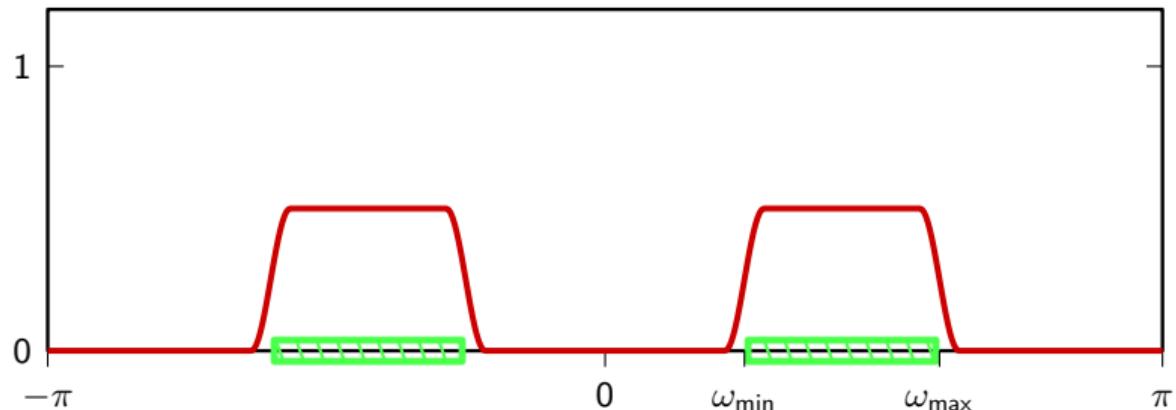
$$h_{K,\beta}[nK] \propto \frac{1}{(\beta n)^2}$$

Raised Cosine: $1/n^2$ decay

$$h_{2,\beta}[n]$$



Spectral shaping with raised cosine



the power constraint

Overview:

- ▶ Noise and probability of error
- ▶ Signaling alphabet and power
- ▶ QAM signaling

Overview:

- ▶ Noise and probability of error
- ▶ Signaling alphabet and power
- ▶ QAM signaling

Overview:

- ▶ Noise and probability of error
- ▶ Signaling alphabet and power
- ▶ QAM signaling

Transmission reliability

- ▶ transmitter sends a sequence of symbols $a[n]$
- ▶ receiver obtains a sequence $\hat{a}[n]$
- ▶ even if no distortion we can't avoid noise: $\hat{a}[n] = a[n] + \eta[n]$
- ▶ when noise is large, we make an error

Transmission reliability

- ▶ transmitter sends a sequence of symbols $a[n]$
- ▶ receiver obtains a sequence $\hat{a}[n]$
- ▶ even if no distortion we can't avoid noise: $\hat{a}[n] = a[n] + \eta[n]$
- ▶ when noise is large, we make an error

Transmission reliability

- ▶ transmitter sends a sequence of symbols $a[n]$
- ▶ receiver obtains a sequence $\hat{a}[n]$
- ▶ even if no distortion we can't avoid noise: $\hat{a}[n] = a[n] + \eta[n]$
- ▶ when noise is large, we make an error

Transmission reliability

- ▶ transmitter sends a sequence of symbols $a[n]$
- ▶ receiver obtains a sequence $\hat{a}[n]$
- ▶ even if no distortion we can't avoid noise: $\hat{a}[n] = a[n] + \eta[n]$
- ▶ when noise is large, we make an error

Probability of error

depends on:

- ▶ power of the noise wrt power of the signal
- ▶ decoding strategy
- ▶ *alphabet* of transmission symbols

Probability of error

depends on:

- ▶ power of the noise wrt power of the signal
- ▶ decoding strategy
- ▶ *alphabet* of transmission symbols

Probability of error

depends on:

- ▶ power of the noise wrt power of the signal
- ▶ decoding strategy
- ▶ *alphabet* of transmission symbols

Signaling alphabets

- ▶ we have a (randomized) bitstream coming in
- ▶ we want to send some upsampled and interpolated samples over the channel
- ▶ how do we go from bitstream to samples?

Signaling alphabets

- ▶ we have a (randomized) bitstream coming in
- ▶ we want to send some upsampled and interpolated samples over the channel
- ▶ how do we go from bitstream to samples?

Signaling alphabets

- ▶ we have a (randomized) bitstream coming in
- ▶ we want to send some upsampled and interpolated samples over the channel
- ▶ how do we go from bitstream to samples?

Mappers and slicers

mapper:

- ▶ split incoming bitstream into chunks
- ▶ assign a symbol $a[n]$ from a finite alphabet \mathcal{A} to each chunk

slicer:

- ▶ receive a value $\hat{a}[n]$
- ▶ decide which symbol from \mathcal{A} is “closest” to $\hat{a}[n]$
- ▶ piece back together the corresponding bitstream

Example: two-level signaling

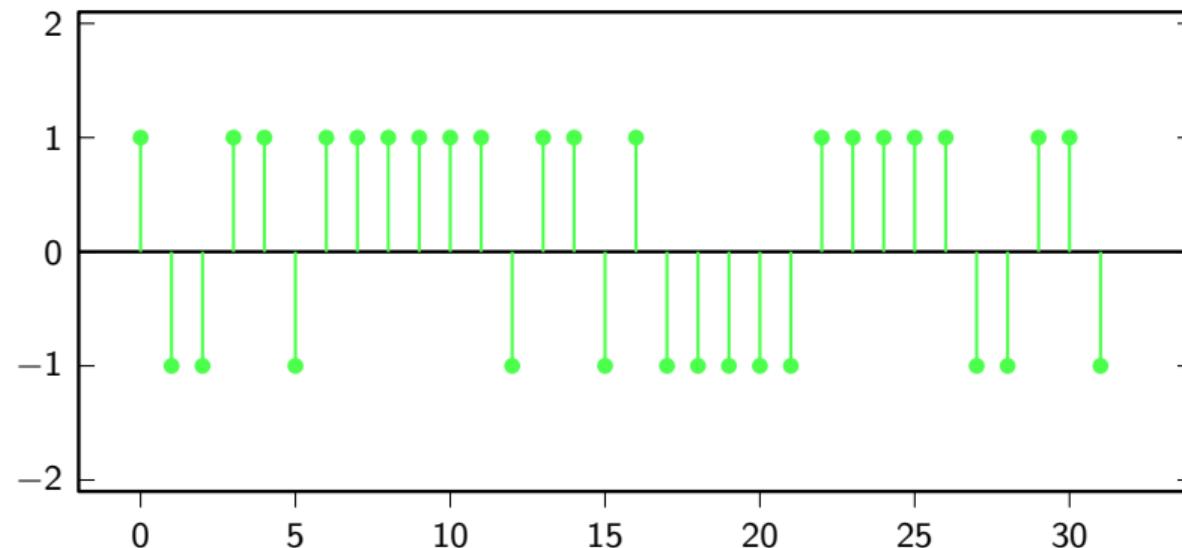
mapper:

- ▶ split incoming bitstream into single bits
- ▶ $a[n] = G$ if the bit is 1, $a[n] = -G$ if the bit is 0

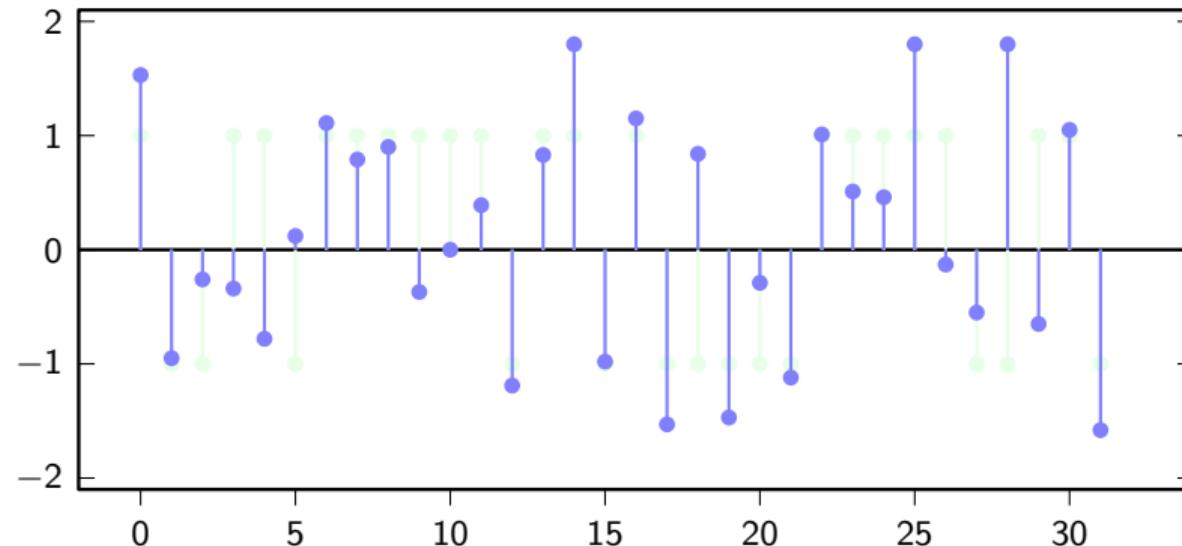
slicer:

- ▶ n -th bit = $\begin{cases} 1 & \text{if } \hat{a}[n] > 0 \\ 0 & \text{otherwise} \end{cases}$

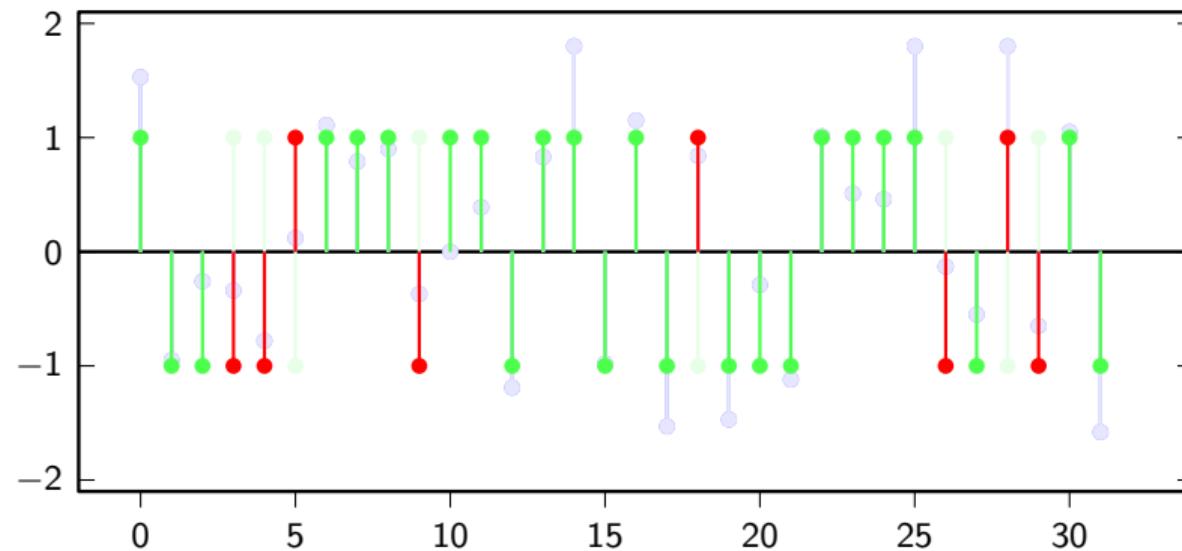
Example: two-level signaling



Example: two-level signaling



Example: two-level signaling



Example: two-level signaling

let's look at the probability of error after making some hypotheses:

- ▶ $\hat{a}[n] = a[n] + \eta[n]$
- ▶ bits in bitstream are equiprobable
- ▶ noise and signal are independent
- ▶ noise is additive white Gaussian noise with zero mean and variance σ_0

Example: two-level signaling

$$\begin{aligned} P_{\text{err}} &= P[\eta[n] < -G \mid n\text{-th bit is } 1]P[\text{n-th bit is } 1] + \\ &\quad P[\eta[n] > G \mid n\text{-th bit is } 0]P[\text{n-th bit is } 0] \\ &= (P[\eta[n] < -G] + P[\eta[n] > G])/2 \\ &= P[\eta[n] > G] \\ &= \int_G^{\infty} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\tau^2}{2\sigma_0^2}} d\tau \\ &= Q(G/\sigma_0) = \frac{1}{2}\operatorname{erfc}((G/\sigma_0)/\sqrt{2}) \end{aligned}$$

Example: two-level signaling

$$\begin{aligned} P_{\text{err}} &= P[\eta[n] < -G \mid n\text{-th bit is 1}] P[n\text{-th bit is 1}] + \\ &\quad P[\eta[n] > G \mid n\text{-th bit is 0}] P[n\text{-th bit is 0}] \\ &= (P[\eta[n] < -G] + P[\eta[n] > G])/2 \\ &= P[\eta[n] > G] \\ &= \int_G^{\infty} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\tau^2}{2\sigma_0^2}} d\tau \\ &= Q(G/\sigma_0) = \frac{1}{2} \operatorname{erfc}((G/\sigma_0)/\sqrt{2}) \end{aligned}$$

Example: two-level signaling

$$\begin{aligned} P_{\text{err}} &= P[\ \eta[n] < -G \mid n\text{-th bit is 1}] P[\ n\text{-th bit is 1}] + \\ &\quad P[\ \eta[n] > G \mid n\text{-th bit is 0}] P[\ n\text{-th bit is 0}] \\ &= (P[\ \eta[n] < -G] + P[\ \eta[n] > G]) / 2 \\ &= P[\ \eta[n] > G] \\ &= \int_G^{\infty} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\tau^2}{2\sigma_0^2}} d\tau \\ &= Q(G/\sigma_0) = \frac{1}{2} \operatorname{erfc}((G/\sigma_0)/\sqrt{2}) \end{aligned}$$

Example: two-level signaling

$$\begin{aligned} P_{\text{err}} &= P[\eta[n] < -G \mid n\text{-th bit is 1}] P[n\text{-th bit is 1}] + \\ &\quad P[\eta[n] > G \mid n\text{-th bit is 0}] P[n\text{-th bit is 0}] \\ &= (P[\eta[n] < -G] + P[\eta[n] > G])/2 \\ &= P[\eta[n] > G] \\ &= \int_G^{\infty} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\tau^2}{2\sigma_0^2}} d\tau \\ &= Q(G/\sigma_0) = \frac{1}{2} \operatorname{erfc}((G/\sigma_0)/\sqrt{2}) \end{aligned}$$

Example: two-level signaling

$$\begin{aligned} P_{\text{err}} &= P[\eta[n] < -G \mid n\text{-th bit is 1}] P[n\text{-th bit is 1}] + \\ &\quad P[\eta[n] > G \mid n\text{-th bit is 0}] P[n\text{-th bit is 0}] \\ &= (P[\eta[n] < -G] + P[\eta[n] > G])/2 \\ &= P[\eta[n] > G] \\ &= \int_G^{\infty} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{\tau^2}{2\sigma_0^2}} d\tau \\ &= Q(G/\sigma_0) = \frac{1}{2} \operatorname{erfc}((G/\sigma_0)/\sqrt{2}) \end{aligned}$$

Example: two-level signaling

transmitted power

$$\begin{aligned}\sigma_s^2 &= G^2 P[n\text{-th bit is } 1] + G^2 P[n\text{-th bit is } 0] \\ &= G^2\end{aligned}$$

$$P_{\text{err}} = Q(\sigma_s/\sigma_0) = Q(\sqrt{\text{SNR}})$$

Example: two-level signaling

transmitted power

$$\begin{aligned}\sigma_s^2 &= G^2 P[n\text{-th bit is } 1] + G^2 P[n\text{-th bit is } 0] \\ &= G^2\end{aligned}$$

$$P_{\text{err}} = Q(\sigma_s/\sigma_0) = Q(\sqrt{\text{SNR}})$$

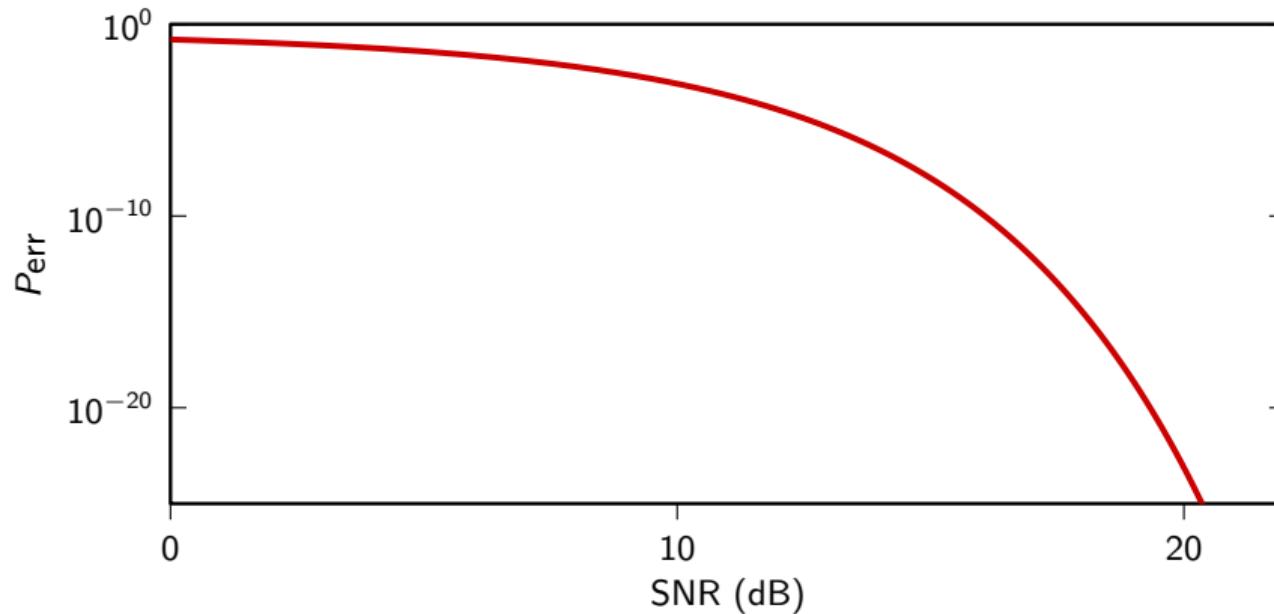
Example: two-level signaling

transmitted power

$$\begin{aligned}\sigma_s^2 &= G^2 P[n\text{-th bit is } 1] + G^2 P[n\text{-th bit is } 0] \\ &= G^2\end{aligned}$$

$$P_{\text{err}} = Q(\sigma_s/\sigma_0) = Q(\sqrt{\text{SNR}})$$

Probability of error



Lesson learned:

- ▶ to reduce the probability of error increase SNR
- ▶ can't control noise so we must increase G
- ▶ increasing G increases the power
- ▶ we can't go above the channel's power constraint!

Lesson learned:

- ▶ to reduce the probability of error increase SNR
- ▶ can't control noise so we must increase G
- ▶ increasing G increases the power
- ▶ we can't go above the channel's power constraint!

Lesson learned:

- ▶ to reduce the probability of error increase SNR
- ▶ can't control noise so we must increase G
- ▶ increasing G increases the power
- ▶ we can't go above the channel's power constraint!

Lesson learned:

- ▶ to reduce the probability of error increase SNR
- ▶ can't control noise so we must increase G
- ▶ increasing G increases the power
- ▶ we can't go above the channel's power constraint!

Multilevel signaling

- ▶ binary signaling is not very efficient (one bit at a time)
- ▶ to increase the throughput we can use multilevel signaling
- ▶ many ways to do so, we will just scratch the surface

Multilevel signaling

- ▶ binary signaling is not very efficient (one bit at a time)
- ▶ to increase the throughput we can use multilevel signaling
- ▶ many ways to do so, we will just scratch the surface

Multilevel signaling

- ▶ binary signaling is not very efficient (one bit at a time)
- ▶ to increase the throughput we can use multilevel signaling
- ▶ many ways to do so, we will just scratch the surface

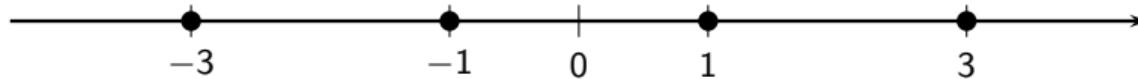
mapper:

- ▶ split incoming bitstream into chunks of M bits
- ▶ chunks define a sequence of integers $k[n] \in \{0, 1, \dots, 2^M - 1\}$
- ▶ $a[n] = G((-2^M + 1) + 2k[n])$ (odd integers around zero)

slicer:

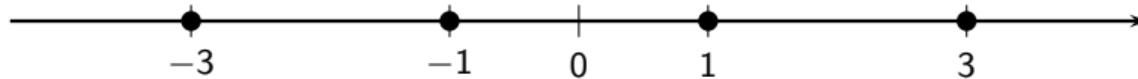
- ▶ $a'[n] = \arg \min_{a \in \mathcal{A}} [|\hat{a}[n] - a|]$

PAM, $M = 2$, $G = 1$



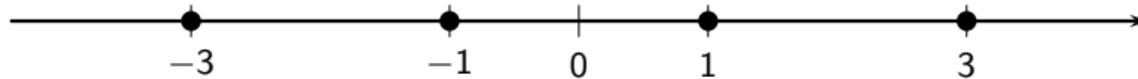
- ▶ distance between points is $2G$
- ▶ using odd integers creates a zero-mean sequence

PAM, $M = 2$, $G = 1$



- ▶ distance between points is $2G$
- ▶ using odd integers creates a zero-mean sequence

PAM, $M = 2$, $G = 1$



- ▶ distance between points is $2G$
- ▶ using odd integers creates a zero-mean sequence

From PAM to QAM

- ▶ error analysis for PAM along the lines of binary signaling
- ▶ can we increase the throughput even further?
- ▶ here's a wild idea, let's use complex numbers

From PAM to QAM

- ▶ error analysis for PAM along the lines of binary signaling
- ▶ can we increase the throughput even further?
- ▶ here's a wild idea, let's use complex numbers

From PAM to QAM

- ▶ error analysis for PAM along the lines of binary signaling
- ▶ can we increase the throughput even further?
- ▶ here's a wild idea, let's use complex numbers

QAM

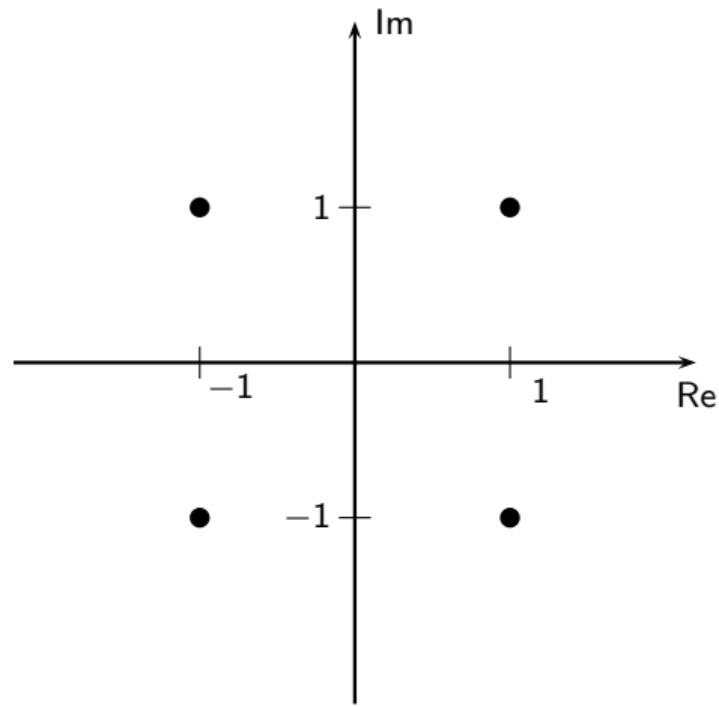
mapper:

- ▶ split incoming bitstream into chunks of M bits, M even
- ▶ use $M/2$ bits to define a PAM sequence $a_r[n]$
- ▶ use the remaining $M/2$ bits to define an independent PAM sequence $a_i[n]$
- ▶ $a[n] = G(a_r[n] + ja_i[n])$

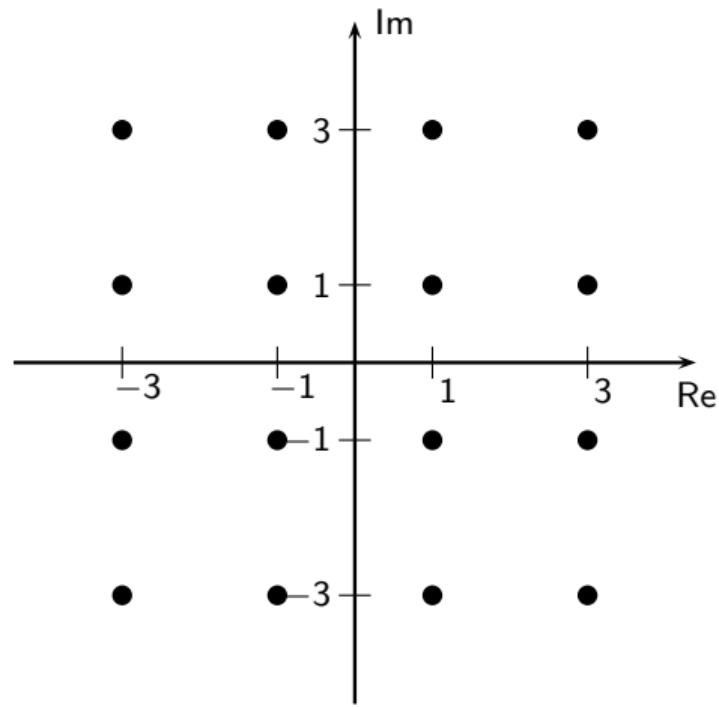
slicer:

- ▶ $a'[n] = \arg \min_{a \in \mathcal{A}} [|\hat{a}[n] - a|]$

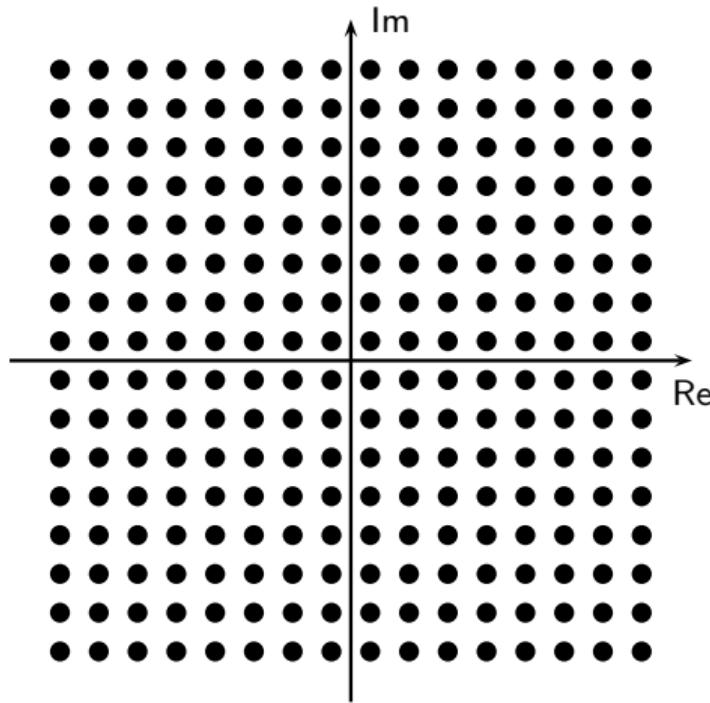
QAM, $M = 2, G = 1$



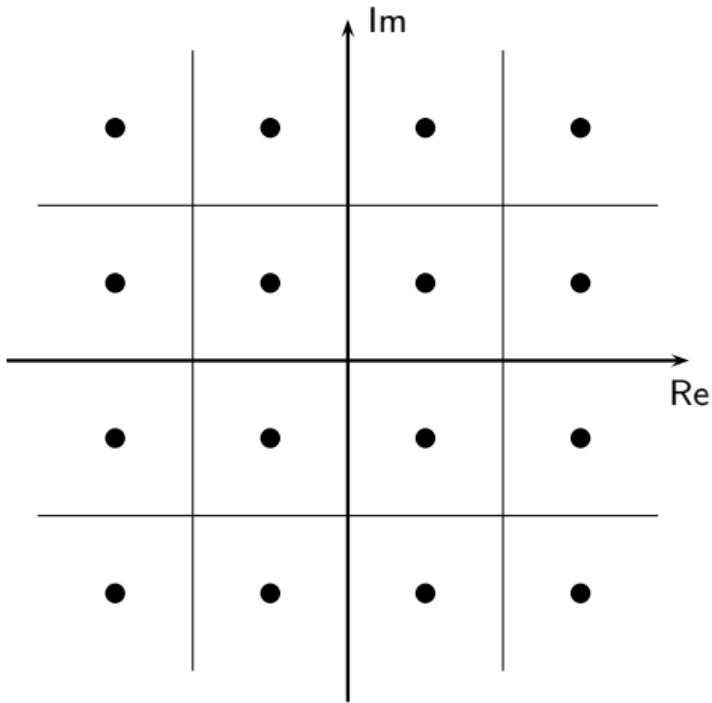
QAM, $M = 4, G = 1$



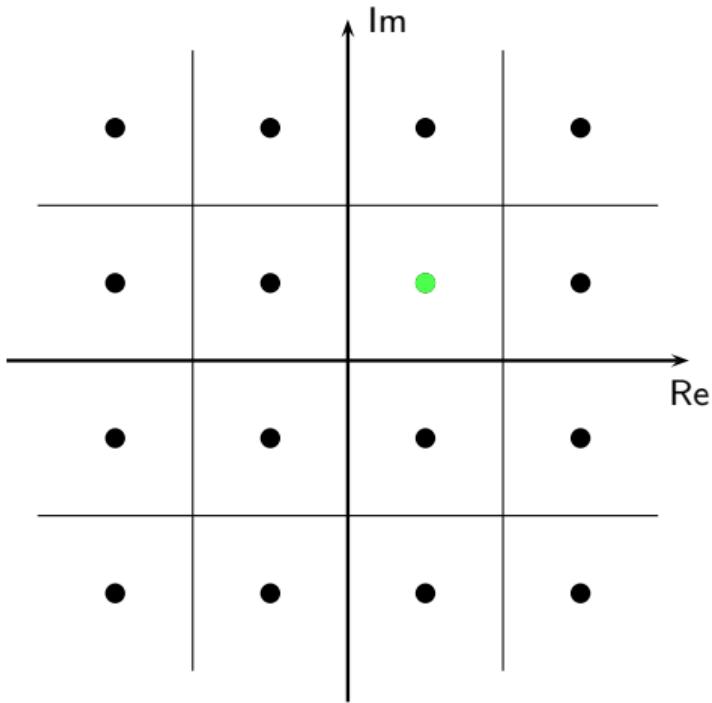
QAM, $M = 8, G = 1$



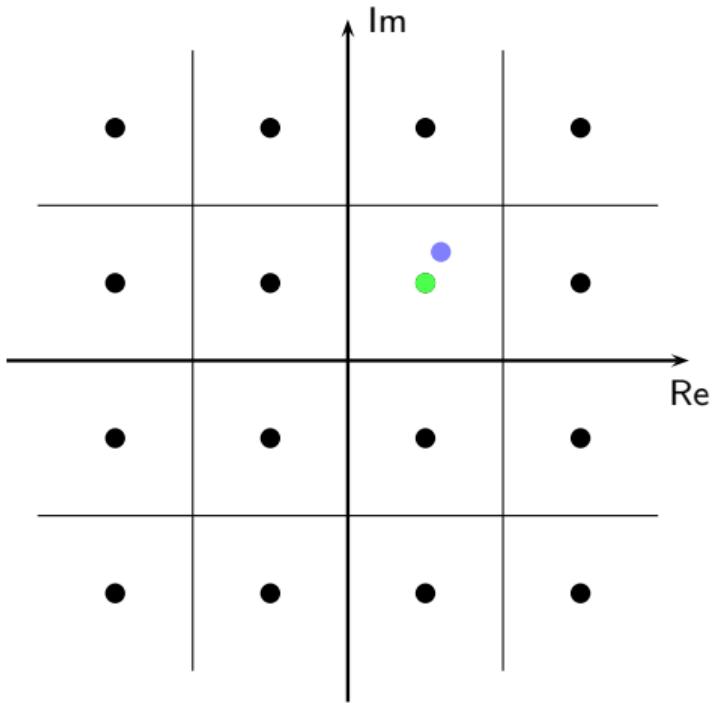
QAM



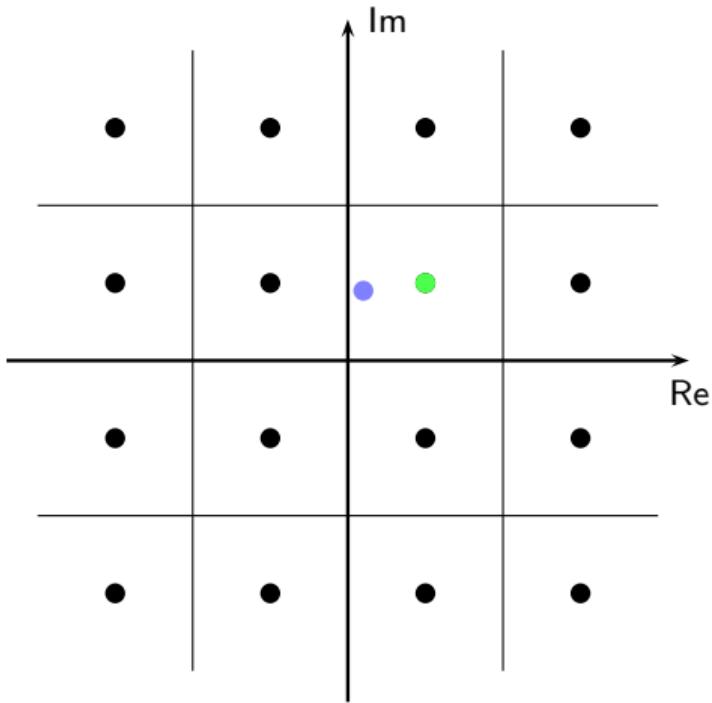
QAM



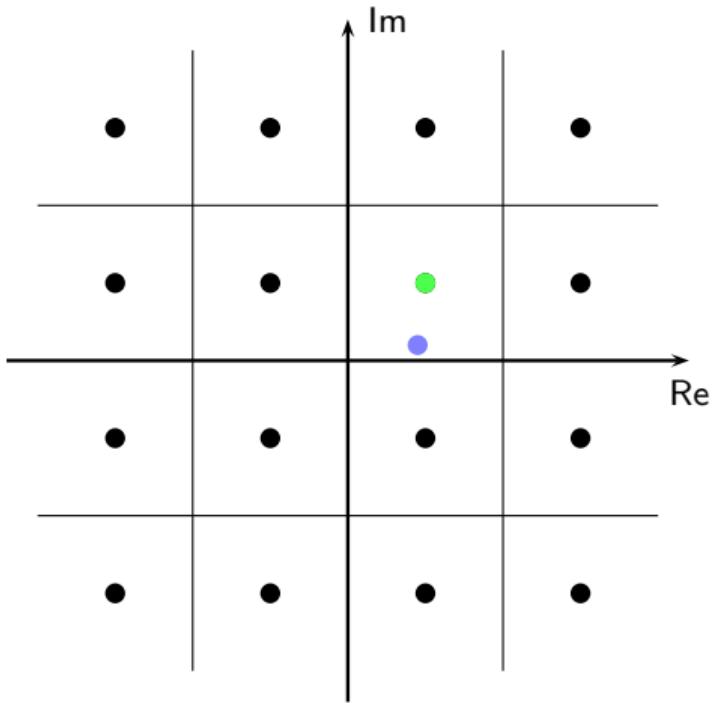
QAM



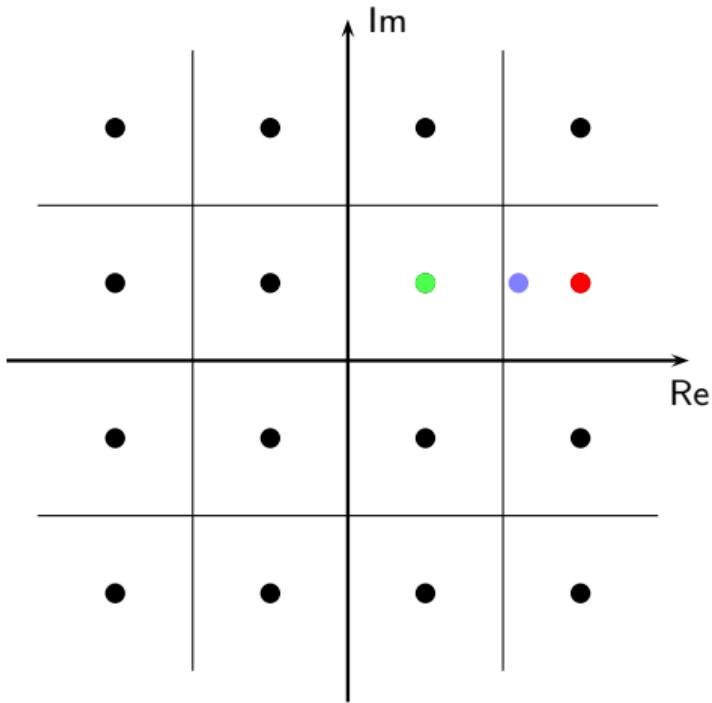
QAM



QAM



QAM



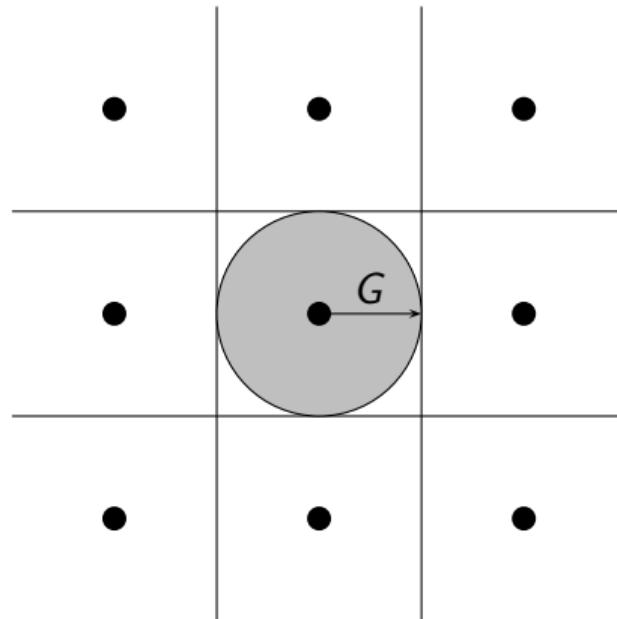
QAM, probability of error

$$\hat{a}[n] = a[n] + \eta[n]$$

$$P_{\text{err}} = 1 - P[|\operatorname{Re}(\eta[n])| < G \wedge |\operatorname{Im}(\eta[n])| < G]$$

$$= 1 - \int_D f_\eta(z) dz$$

QAM, probability of error, circular approximation



QAM, probability of error

$$P_{\text{err}} \approx 1 - \int_{|z| < G} f_\eta(z) dz \quad f_\eta(z) = \frac{1}{\pi\sigma_0^2} e^{-\frac{|z|^2}{\sigma_0^2}}$$

$$z \rightarrow \rho e^{j\theta}$$

$$= 1 - \int_0^{2\pi} d\theta \int_0^G \frac{\rho}{\pi\sigma_0^2} e^{-\frac{\rho^2}{\sigma_0^2}} d\rho$$

$$= e^{-\frac{G^2}{\sigma_0^2}}$$

QAM, probability of error

$$P_{\text{err}} \approx 1 - \int_{|z| < G} f_\eta(z) dz$$
$$f_\eta(z) = \frac{1}{\pi \sigma_0^2} e^{-\frac{|z|^2}{\sigma_0^2}}$$
$$z \rightarrow \rho e^{j\theta}$$
$$= 1 - \int_0^{2\pi} d\theta \int_0^G \frac{\rho}{\pi \sigma_0^2} e^{-\frac{\rho^2}{\sigma_0^2}} d\rho$$
$$= e^{-\frac{G^2}{\sigma_0^2}}$$

QAM, probability of error

$$\begin{aligned} P_{\text{err}} &\approx 1 - \int_{|z|<G} f_\eta(z) dz & f_\eta(z) &= \frac{1}{\pi\sigma_0^2} e^{-\frac{|z|^2}{\sigma_0^2}} \\ z &\rightarrow \rho e^{j\theta} \\ &= 1 - \int_0^{2\pi} d\theta \int_0^G \frac{\rho}{\pi\sigma_0^2} e^{-\frac{\rho^2}{\sigma_0^2}} d\rho \\ &= e^{-\frac{G^2}{\sigma_0^2}} \end{aligned}$$

QAM, probability of error

$$\begin{aligned} P_{\text{err}} &\approx 1 - \int_{|z| < G} f_\eta(z) dz & f_\eta(z) &= \frac{1}{\pi\sigma_0^2} e^{-\frac{|z|^2}{\sigma_0^2}} \\ z &\rightarrow \rho e^{j\theta} \\ &= 1 - \int_0^{2\pi} d\theta \int_0^G \frac{\rho}{\pi\sigma_0^2} e^{-\frac{\rho^2}{\sigma_0^2}} d\rho \\ &= e^{-\frac{G^2}{\sigma_0^2}} \end{aligned}$$

QAM, probability of error

transmitted power (all symbols equiprobable and independent):

$$\begin{aligned}\sigma_s^2 &= G^2 \frac{1}{2^M} \sum_{a \in \mathcal{A}} |a|^2 \\ &= G^2 \frac{2}{3} (2^M - 1)\end{aligned}$$

$$P_{\text{err}} \approx e^{-\frac{G^2}{\sigma_0^2}} \approx e^{-3 \text{SNR}/2^{(M+1)}}$$

QAM, probability of error

transmitted power (all symbols equiprobable and independent):

$$\begin{aligned}\sigma_s^2 &= G^2 \frac{1}{2^M} \sum_{a \in \mathcal{A}} |a|^2 \\ &= G^2 \frac{2}{3} (2^M - 1)\end{aligned}$$

$$P_{\text{err}} \approx e^{-\frac{G^2}{\sigma_0^2}} \approx e^{-3 \text{SNR}/2^{(M+1)}}$$

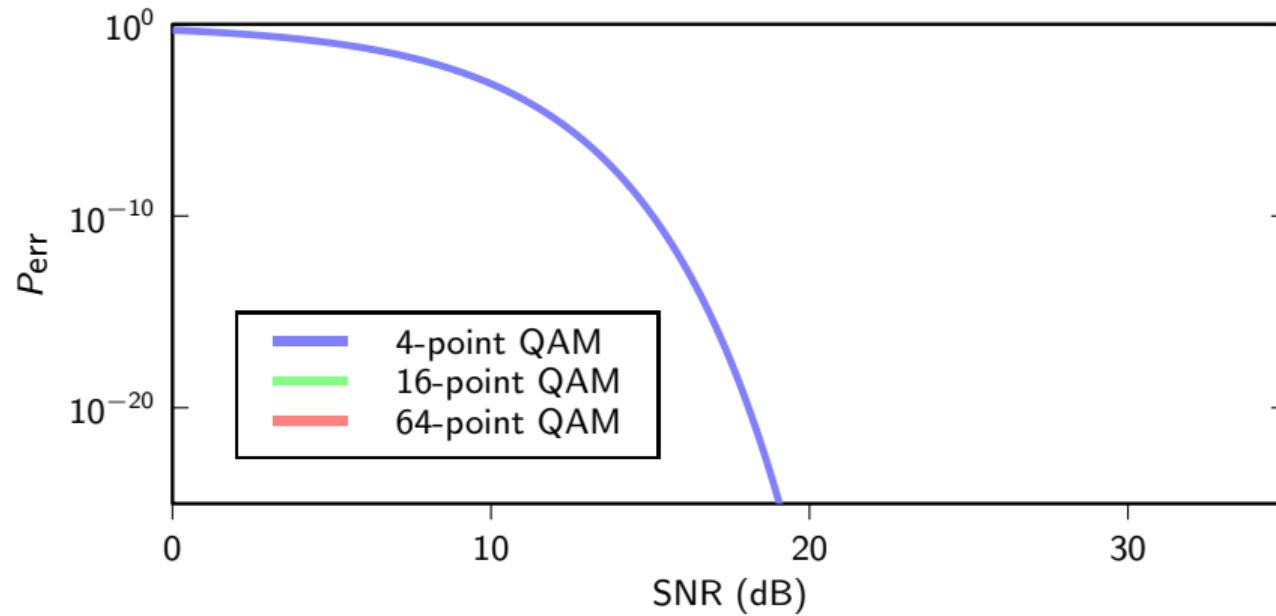
QAM, probability of error

transmitted power (all symbols equiprobable and independent):

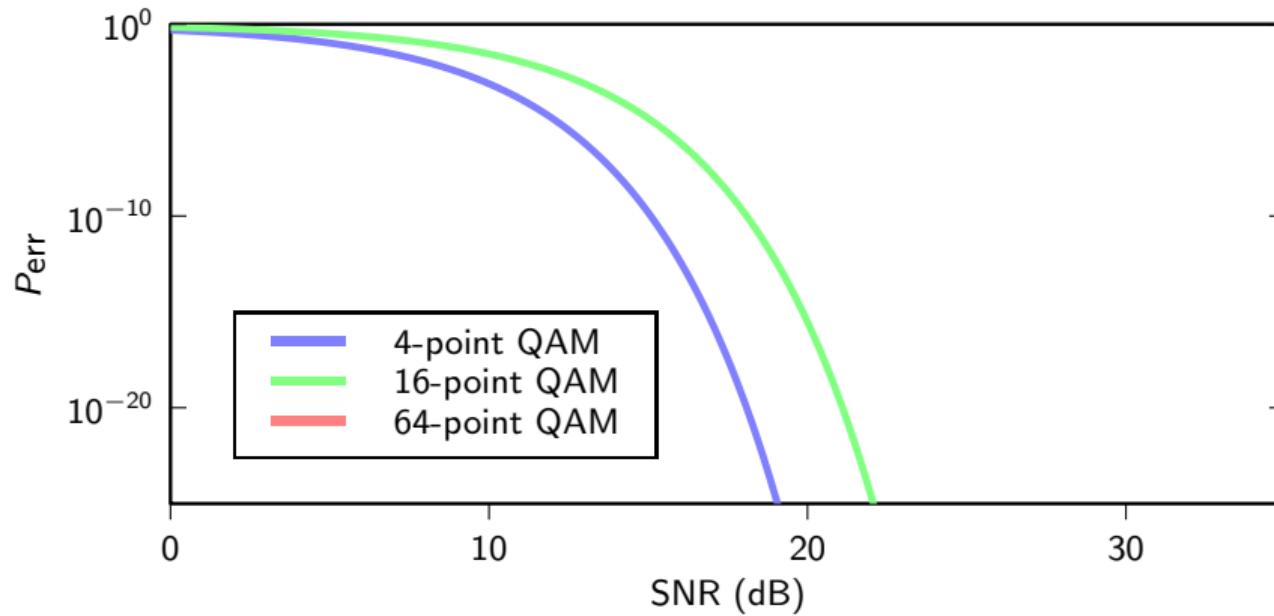
$$\begin{aligned}\sigma_s^2 &= G^2 \frac{1}{2^M} \sum_{a \in \mathcal{A}} |a|^2 \\ &= G^2 \frac{2}{3} (2^M - 1)\end{aligned}$$

$$P_{\text{err}} \approx e^{-\frac{G^2}{\sigma_0^2}} \approx e^{-3 \text{SNR}/2^{(M+1)}}$$

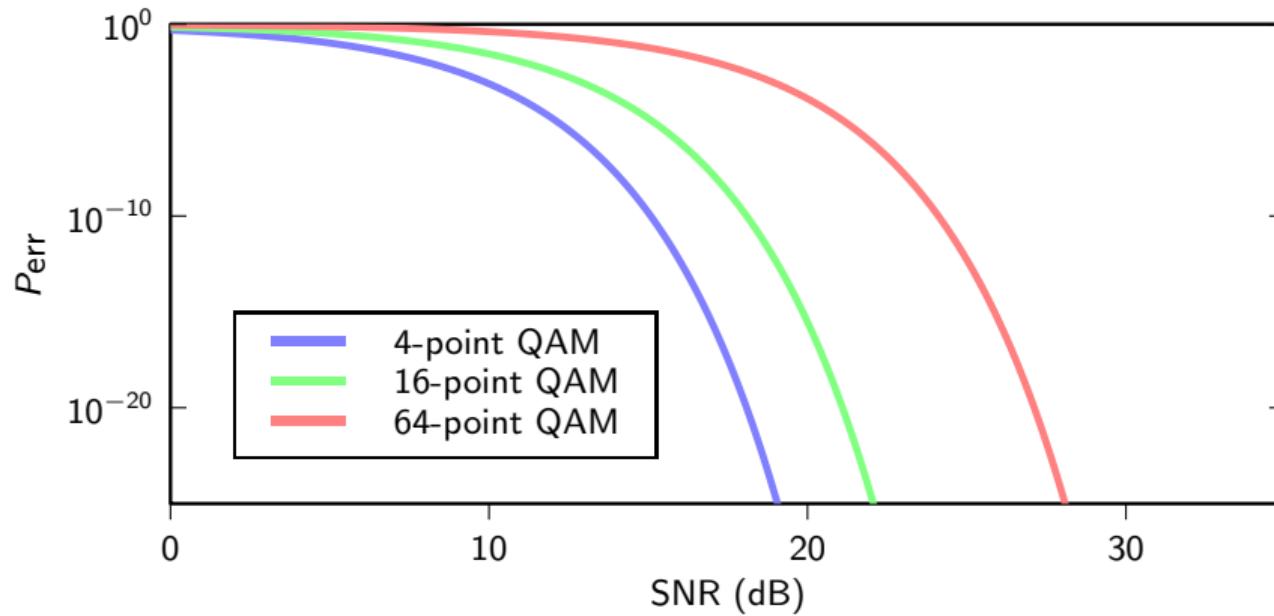
Probability of error



Probability of error



Probability of error



QAM, the recipe

- ▶ pick a probability of error you can live with (e.g. 10^{-6})
- ▶ find out the SNR imposed by the channel's power constraint
- ▶ $M = \log_2 \left(1 - \frac{3}{2} \frac{\text{SNR}}{\ln(p_e)} \right)$
- ▶ final throughput will be MW

QAM, the recipe

- ▶ pick a probability of error you can live with (e.g. 10^{-6})
- ▶ find out the SNR imposed by the channel's power constraint
- ▶ $M = \log_2 \left(1 - \frac{3}{2} \frac{\text{SNR}}{\ln(p_e)} \right)$
- ▶ final throughput will be MW

QAM, the recipe

- ▶ pick a probability of error you can live with (e.g. 10^{-6})
- ▶ find out the SNR imposed by the channel's power constraint
- ▶ $M = \log_2 \left(1 - \frac{3}{2} \frac{\text{SNR}}{\ln(p_e)} \right)$
- ▶ final throughput will be MW

QAM, the recipe

- ▶ pick a probability of error you can live with (e.g. 10^{-6})
- ▶ find out the SNR imposed by the channel's power constraint
- ▶ $M = \log_2 \left(1 - \frac{3}{2} \frac{\text{SNR}}{\ln(p_e)} \right)$
- ▶ final throughput will be MW

QAM

where we stand:

- ▶ we know how to fit the bandwidth constraint
- ▶ with QAM, we know how many bits per symbol we can use given the power constraint
- ▶ we know the theoretical throughput of the transmitter

but how do we transmit *complex* symbols over a real channel?

QAM

where we stand:

- ▶ we know how to fit the bandwidth constraint
- ▶ with QAM, we know how many bits per symbol we can use given the power constraint
- ▶ we know the theoretical throughput of the transmitter

but how do we transmit *complex* symbols over a real channel?

QAM

where we stand:

- ▶ we know how to fit the bandwidth constraint
- ▶ with QAM, we know how many bits per symbol we can use given the power constraint
- ▶ we know the theoretical throughput of the transmitter

but how do we transmit *complex* symbols over a real channel?

QAM

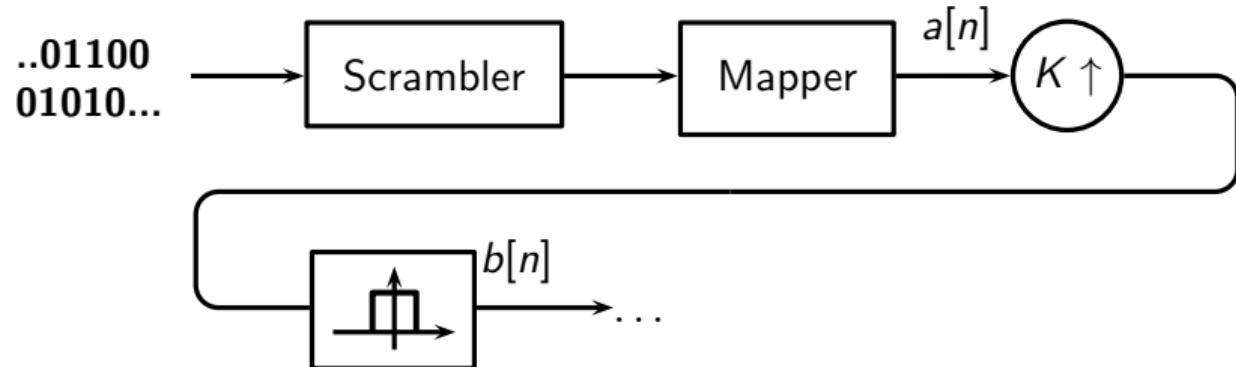
where we stand:

- ▶ we know how to fit the bandwidth constraint
- ▶ with QAM, we know how many bits per symbol we can use given the power constraint
- ▶ we know the theoretical throughput of the transmitter

but how do we transmit *complex* symbols over a real channel?

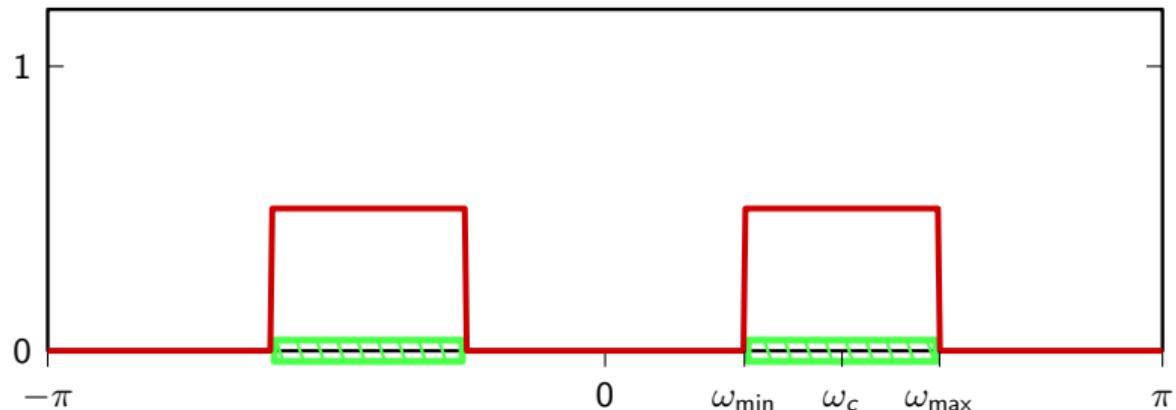
a QAM transmitter

QAM transmitter design

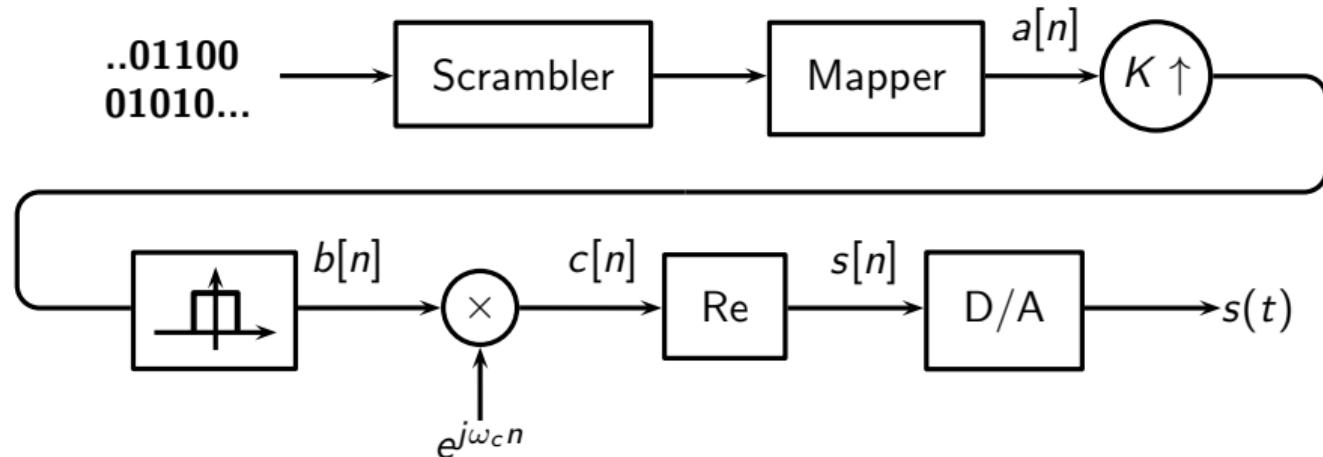


$b[n] = b_r[n] + jb_i[n]$ is a complex-valued baseband signal...

Fulfilling the bandwidth constraint



QAM transmitter, final design



The transmitted passband signal

$$c[n] = b[n] e^{j\omega_c n}$$

$$\begin{aligned} s[n] &= \operatorname{Re}\{c[n]\} \\ &= \operatorname{Re}\{(b_r[n] + jb_i[n])(\cos \omega_c n + j \sin \omega_c n)\} \\ &= b_r[n] \cos \omega_c n - b_i[n] \sin \omega_c n \end{aligned}$$

The transmitted passband signal

$$c[n] = b[n] e^{j\omega_c n}$$

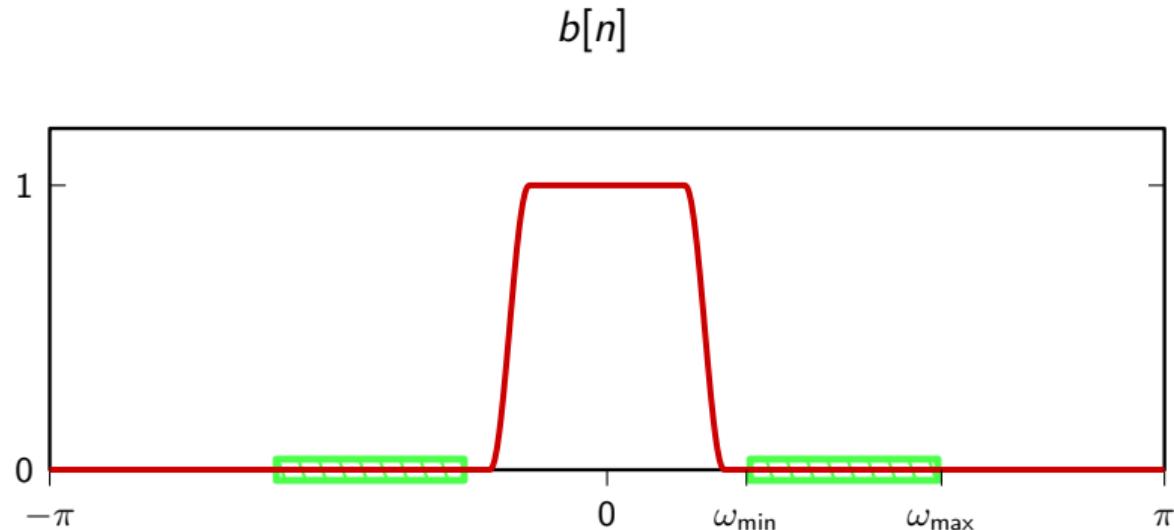
$$\begin{aligned} s[n] &= \operatorname{Re}\{c[n]\} \\ &= \operatorname{Re}\{(b_r[n] + jb_i[n])(\cos \omega_c n + j \sin \omega_c n)\} \\ &= b_r[n] \cos \omega_c n - b_i[n] \sin \omega_c n \end{aligned}$$

The transmitted passband signal

$$c[n] = b[n] e^{j\omega_c n}$$

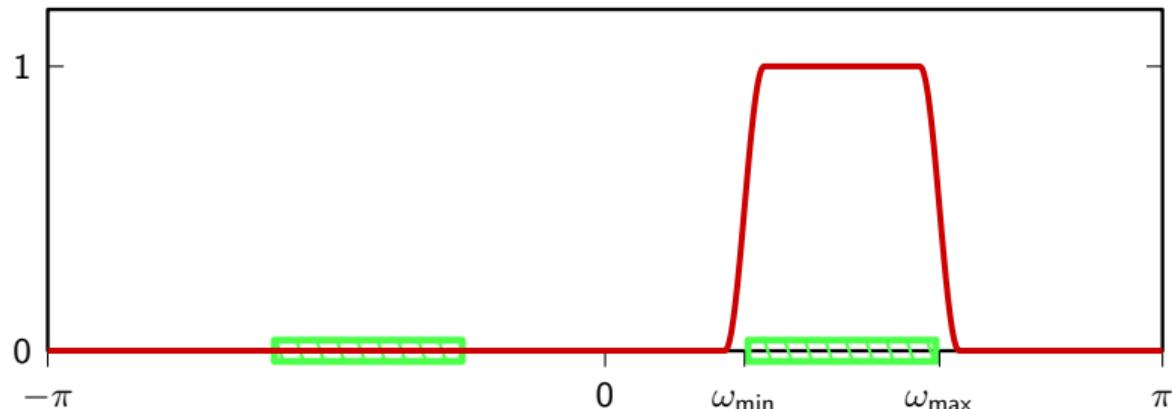
$$\begin{aligned} s[n] &= \operatorname{Re}\{c[n]\} \\ &= \operatorname{Re}\{(b_r[n] + jb_i[n])(\cos \omega_c n + j \sin \omega_c n)\} \\ &= b_r[n] \cos \omega_c n - b_i[n] \sin \omega_c n \end{aligned}$$

From complex baseband to real passband signal



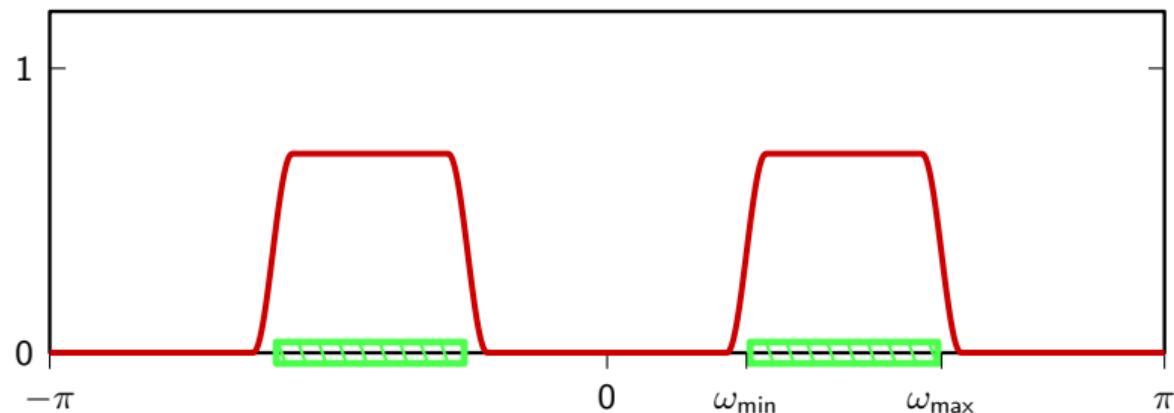
From complex baseband to real passband signal

$$c[n] = b[n] e^{j\omega_c n}$$



From complex baseband to real passband signal

$$s[n] = \operatorname{Re}\{c[n]\}$$

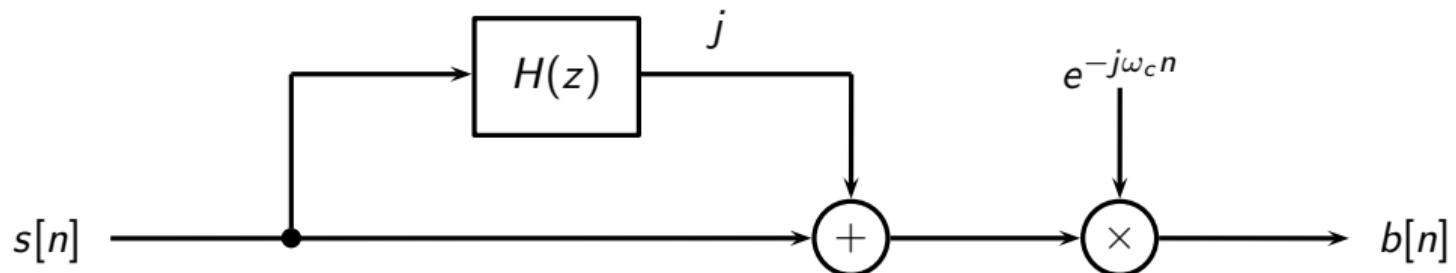


Can we go back?

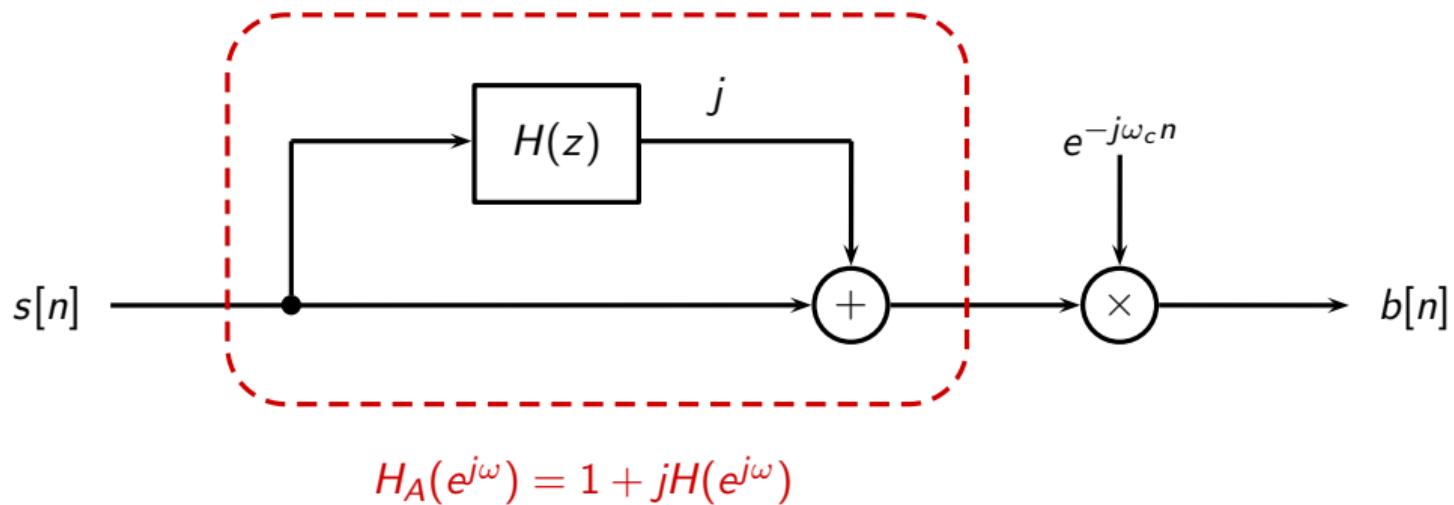
receiver obtains $s[n]$
can we recover the complex baseband $b[n]$ from $s[n]$?

easiest way: Hilbert demodulation

Hilbert demodulation



Hilbert demodulation

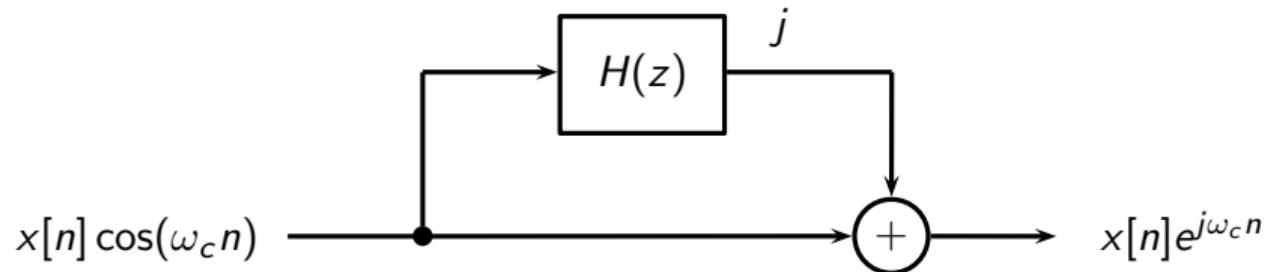


Hilbert demodulation

$$H(e^{j\omega}) = \begin{cases} -j & \omega \geq 0 \\ j & \omega < 0 \end{cases}$$

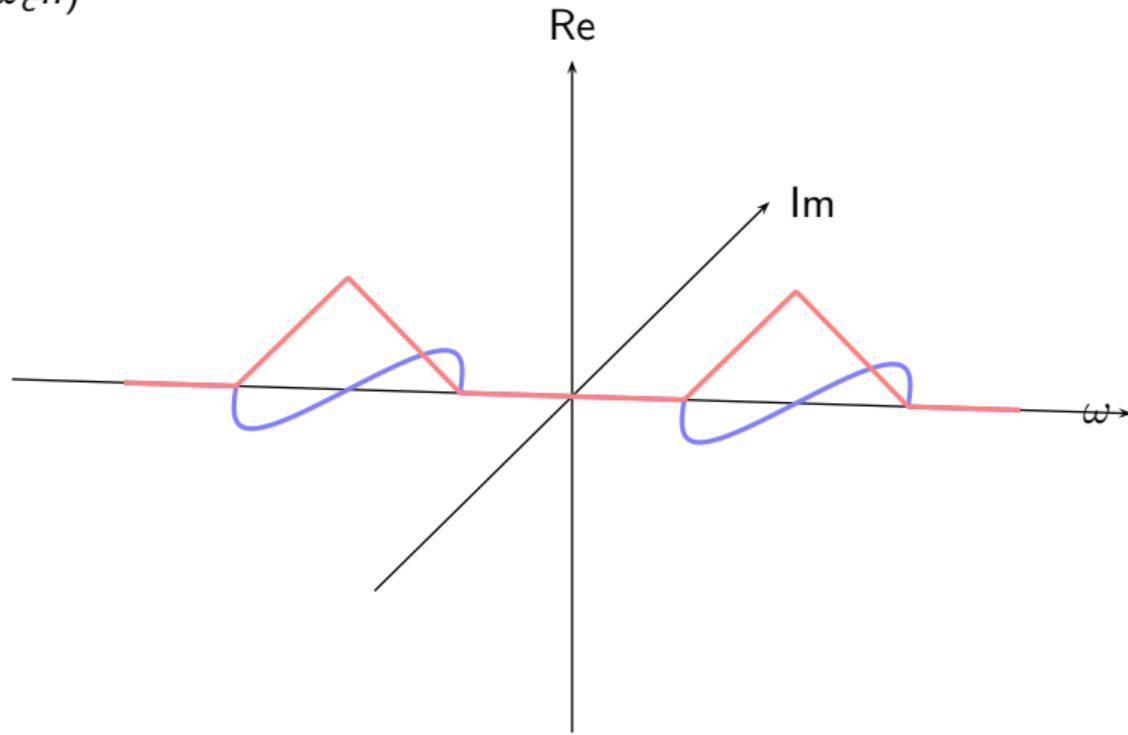
$$H_A(e^{j\omega}) = 1 + jH(e^{j\omega}) = \begin{cases} 2 & \omega \geq 0 \\ 0 & \omega < 0 \end{cases}$$

Hilbert demodulation (recap)



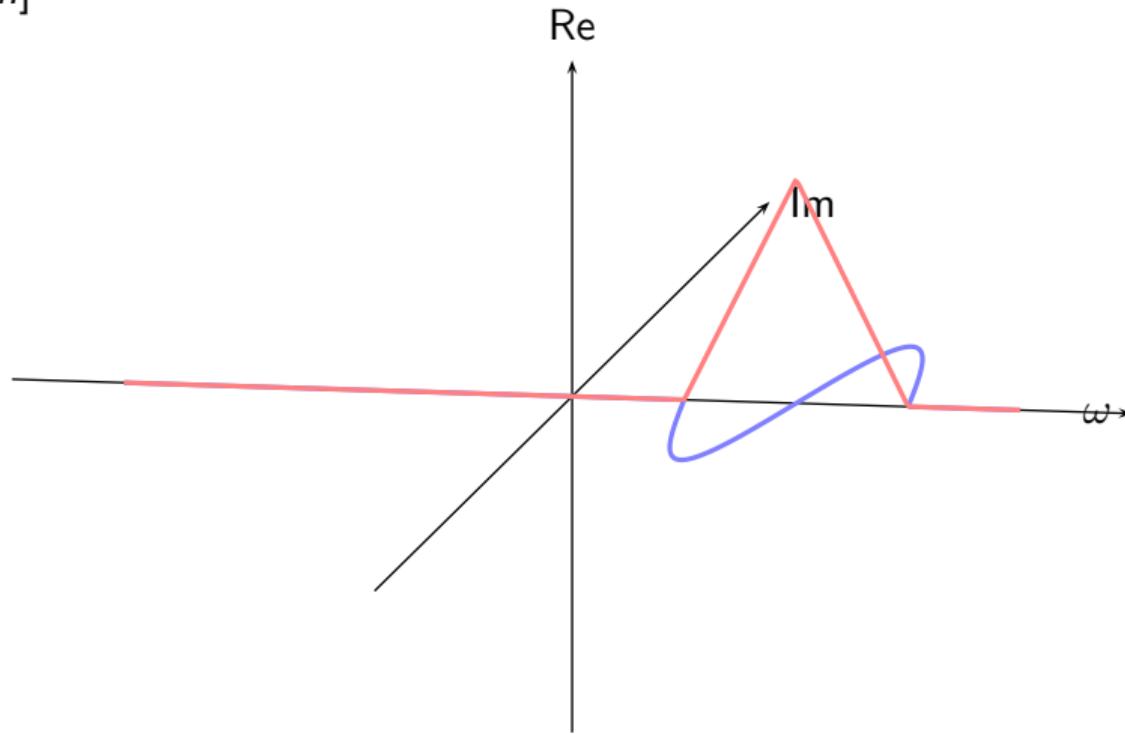
Hilbert demodulation

$$y[n] = x[n] \cos(\omega_c n)$$

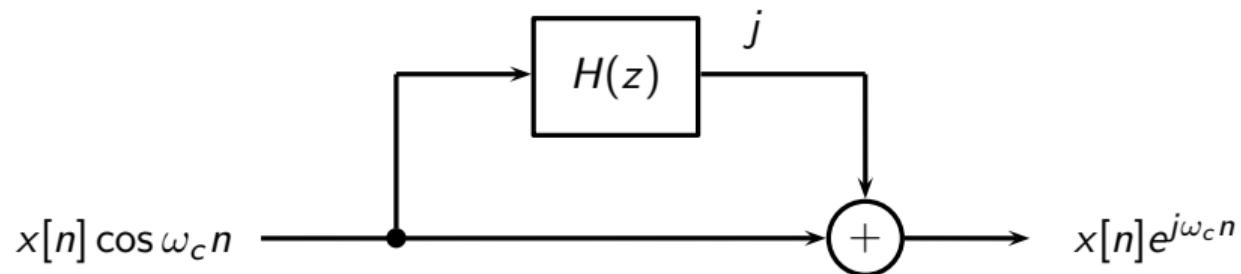


Hilbert demodulation

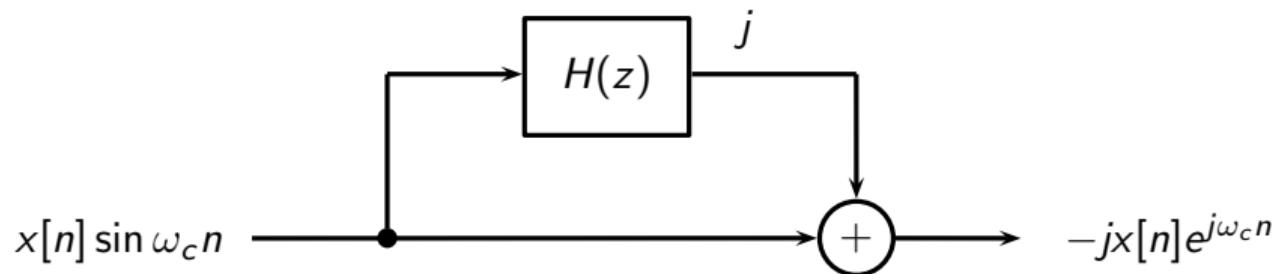
$$jy[n] * h[n] + y[n]$$



Hilbert demodulation



Hilbert demodulation

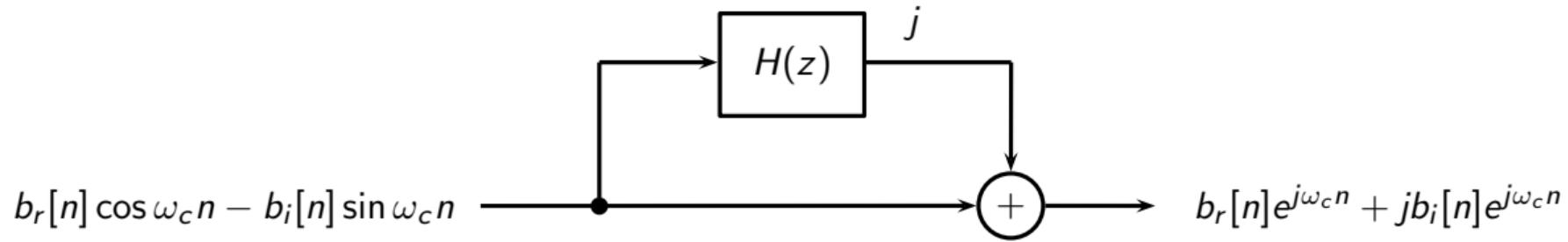


The transmitted passband signal

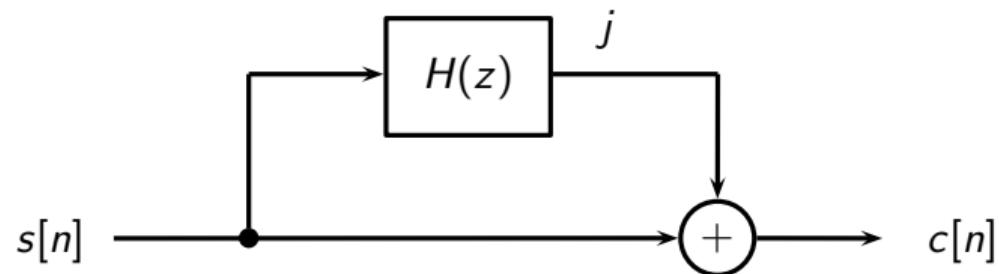
$$\begin{aligned} c[n] &= b[n] e^{j\omega_c n} \\ &= (b_r[n] + j b_i[n]) e^{j\omega_c n} \end{aligned}$$

$$\begin{aligned} s[n] &= \operatorname{Re}\{c[n]\} \\ &= b_r[n] \cos \omega_c n - b_i[n] \sin \omega_c n \end{aligned}$$

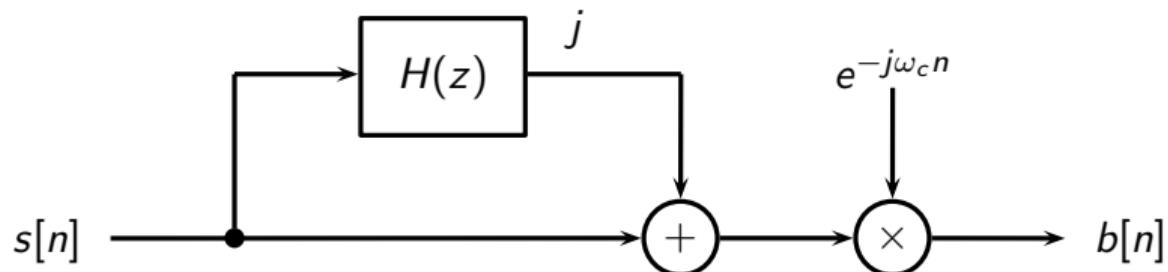
Hilbert demodulation



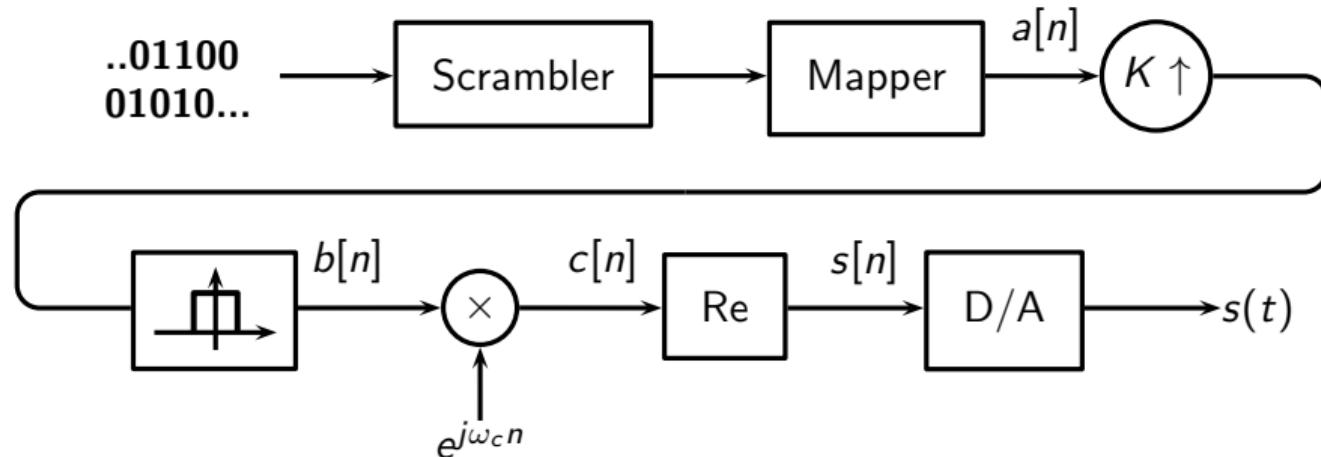
Hilbert demodulation



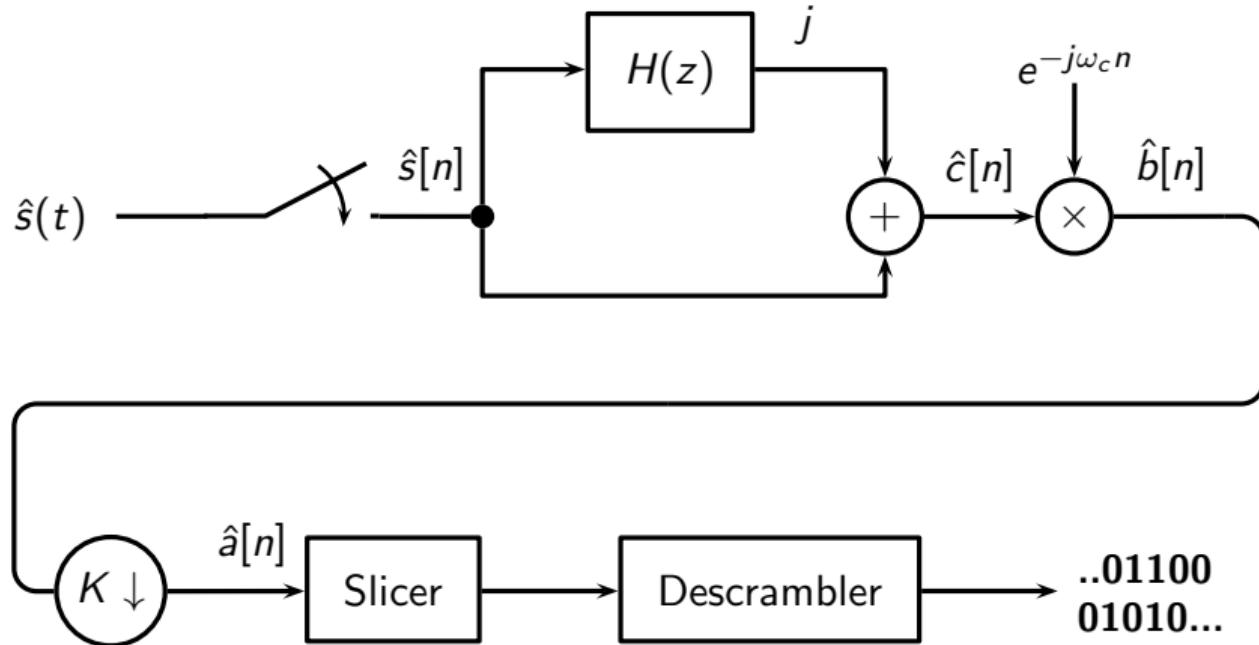
Hilbert demodulation



QAM transmitter, final design



QAM receiver, idealized design



Example: the V.32 voiceband modem

- ▶ analog telephone channel: $F_{\min} = 450\text{Hz}$, $F_{\max} = 2850\text{Hz}$
- ▶ usable bandwidth: $W = 2400\text{Hz}$, center frequency $F_c = 1650\text{Hz}$
- ▶ pick $F_s = 3 \cdot 2400 = 7200\text{Hz}$, so that $K = 3$
- ▶ $\omega_c = 0.458\pi$

Example: the V.32 voiceband modem

- ▶ analog telephone channel: $F_{\min} = 450\text{Hz}$, $F_{\max} = 2850\text{Hz}$
- ▶ usable bandwidth: $W = 2400\text{Hz}$, center frequency $F_c = 1650\text{Hz}$
- ▶ pick $F_s = 3 \cdot 2400 = 7200\text{Hz}$, so that $K = 3$
- ▶ $\omega_c = 0.458\pi$

Example: the V.32 voiceband modem

- ▶ analog telephone channel: $F_{\min} = 450\text{Hz}$, $F_{\max} = 2850\text{Hz}$
- ▶ usable bandwidth: $W = 2400\text{Hz}$, center frequency $F_c = 1650\text{Hz}$
- ▶ pick $F_s = 3 \cdot 2400 = 7200\text{Hz}$, so that $K = 3$
- ▶ $\omega_c = 0.458\pi$

Example: the V.32 voiceband modem

- ▶ analog telephone channel: $F_{\min} = 450\text{Hz}$, $F_{\max} = 2850\text{Hz}$
- ▶ usable bandwidth: $W = 2400\text{Hz}$, center frequency $F_c = 1650\text{Hz}$
- ▶ pick $F_s = 3 \cdot 2400 = 7200\text{Hz}$, so that $K = 3$
- ▶ $\omega_c = 0.458\pi$

Example: the V.32 voiceband modem

- ▶ maximum SNR: 22dB
- ▶ pick $P_{\text{err}} = 10^{-6}$
- ▶ using QAM, we find

$$M = \log_2 \left(1 - \frac{3}{2} \frac{10^{22/10}}{\ln(10^{-6})} \right) \approx 4.1865$$

so we pick $M = 4$ and use a 16-point constellation

- ▶ final data rate is $WM = 9600$ bits per second

Example: the V.32 voiceband modem

- ▶ maximum SNR: 22dB
- ▶ pick $P_{\text{err}} = 10^{-6}$
- ▶ using QAM, we find

$$M = \log_2 \left(1 - \frac{3}{2} \frac{10^{22/10}}{\ln(10^{-6})} \right) \approx 4.1865$$

so we pick $M = 4$ and use a 16-point constellation

- ▶ final data rate is $WM = 9600$ bits per second

Example: the V.32 voiceband modem

- ▶ maximum SNR: 22dB
- ▶ pick $P_{\text{err}} = 10^{-6}$
- ▶ using QAM, we find

$$M = \log_2 \left(1 - \frac{3}{2} \frac{10^{22/10}}{\ln(10^{-6})} \right) \approx 4.1865$$

so we pick $M = 4$ and use a 16-point constellation

- ▶ final data rate is $WM = 9600$ bits per second

Example: the V.32 voiceband modem

- ▶ maximum SNR: 22dB
- ▶ pick $P_{\text{err}} = 10^{-6}$
- ▶ using QAM, we find

$$M = \log_2 \left(1 - \frac{3}{2} \frac{10^{22/10}}{\ln(10^{-6})} \right) \approx 4.1865$$

so we pick $M = 4$ and use a 16-point constellation

- ▶ final data rate is $WM = 9600$ bits per second

Theoretical channel capacity

- ▶ we used very specific design choices to derive the throughput
- ▶ what is the best one can do?
- ▶ Shannon's capacity formula is the upper bound

$$C = W \log_2 (1 + SNR)$$

- ▶ for instance, for the previous example $C \approx 17500$ bps
- ▶ the gap can be narrowed by more advanced coding techniques

Theoretical channel capacity

- ▶ we used very specific design choices to derive the throughput
- ▶ what is the best one can do?
- ▶ Shannon's capacity formula is the upper bound

$$C = W \log_2 (1 + SNR)$$

- ▶ for instance, for the previous example $C \approx 17500$ bps
- ▶ the gap can be narrowed by more advanced coding techniques

Theoretical channel capacity

- ▶ we used very specific design choices to derive the throughput
- ▶ what is the best one can do?
- ▶ Shannon's capacity formula is the upper bound

$$C = W \log_2 (1 + SNR)$$

- ▶ for instance, for the previous example $C \approx 17500$ bps
- ▶ the gap can be narrowed by more advanced coding techniques

Theoretical channel capacity

- ▶ we used very specific design choices to derive the throughput
- ▶ what is the best one can do?
- ▶ Shannon's capacity formula is the upper bound

$$C = W \log_2 (1 + SNR)$$

- ▶ for instance, for the previous example $C \approx 17500$ bps
- ▶ the gap can be narrowed by more advanced coding techniques

Theoretical channel capacity

- ▶ we used very specific design choices to derive the throughput
- ▶ what is the best one can do?
- ▶ Shannon's capacity formula is the upper bound

$$C = W \log_2 (1 + SNR)$$

- ▶ for instance, for the previous example $C \approx 17500$ bps
- ▶ the gap can be narrowed by more advanced coding techniques

COM303: Digital Signal Processing

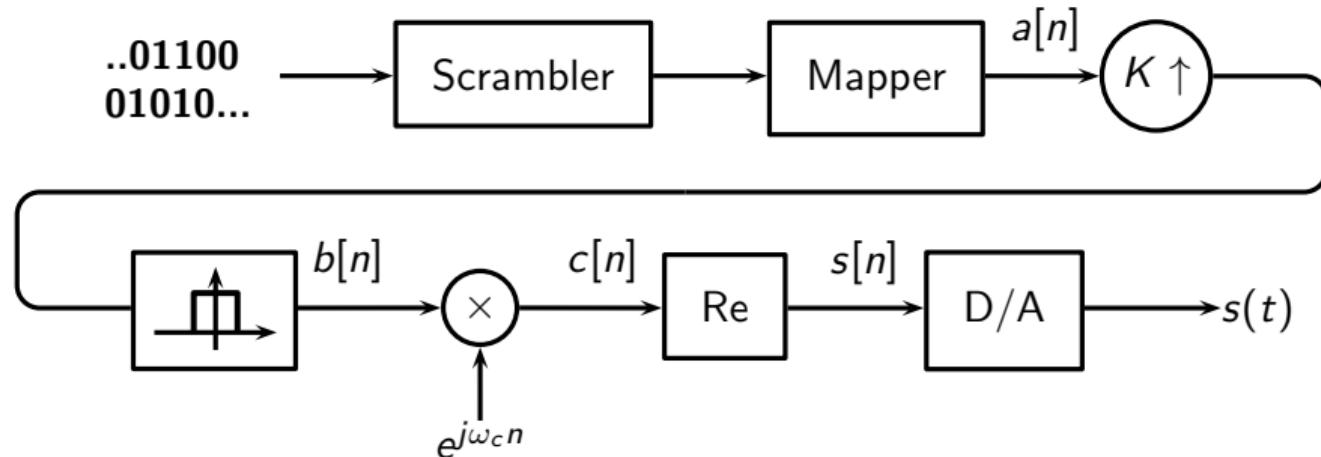
Lecture 23: Digital Communication Systems (II)

overview

- ▶ QAM receiver design
- ▶ ADSL

receiver design

QAM transmitter, final design



It's a dirty job...

but a receiver has to do it:

- ▶ propagation delay
- ▶ channel effects
- ▶ interference
- ▶ clock drift

It's a dirty job...

but a receiver has to do it:

- ▶ propagation delay
- ▶ channel effects
- ▶ interference
- ▶ clock drift

It's a dirty job...

but a receiver has to do it:

- ▶ propagation delay
- ▶ channel effects
- ▶ interference
- ▶ clock drift

It's a dirty job...

but a receiver has to do it:

- ▶ propagation delay
- ▶ channel effects
- ▶ interference
- ▶ clock drift

It's a dirty job...

but a receiver has to do it:

- ▶ propagation delay → handshake and delay estimation
- ▶ channel effects
- ▶ interference
- ▶ clock drift

It's a dirty job...

but a receiver has to do it:

- ▶ propagation delay → handshake and delay estimation
- ▶ channel effects → adaptive equalization
- ▶ interference
- ▶ clock drift

It's a dirty job...

but a receiver has to do it:

- ▶ propagation delay → handshake and delay estimation
- ▶ channel effects → adaptive equalization
- ▶ interference → line probing
- ▶ clock drift

It's a dirty job...

but a receiver has to do it:

- ▶ propagation delay → handshake and delay estimation
- ▶ channel effects → adaptive equalization
- ▶ interference → line probing
- ▶ clock drift → timing recovery

A blast from the past

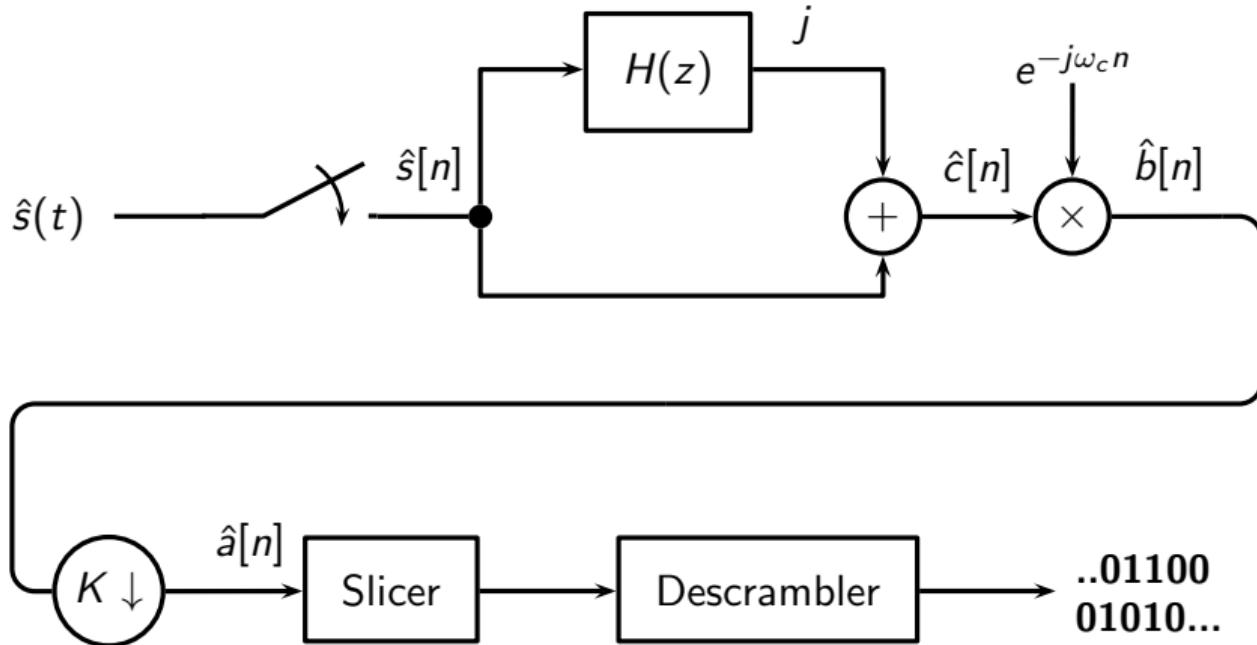
Play

A blast from the past

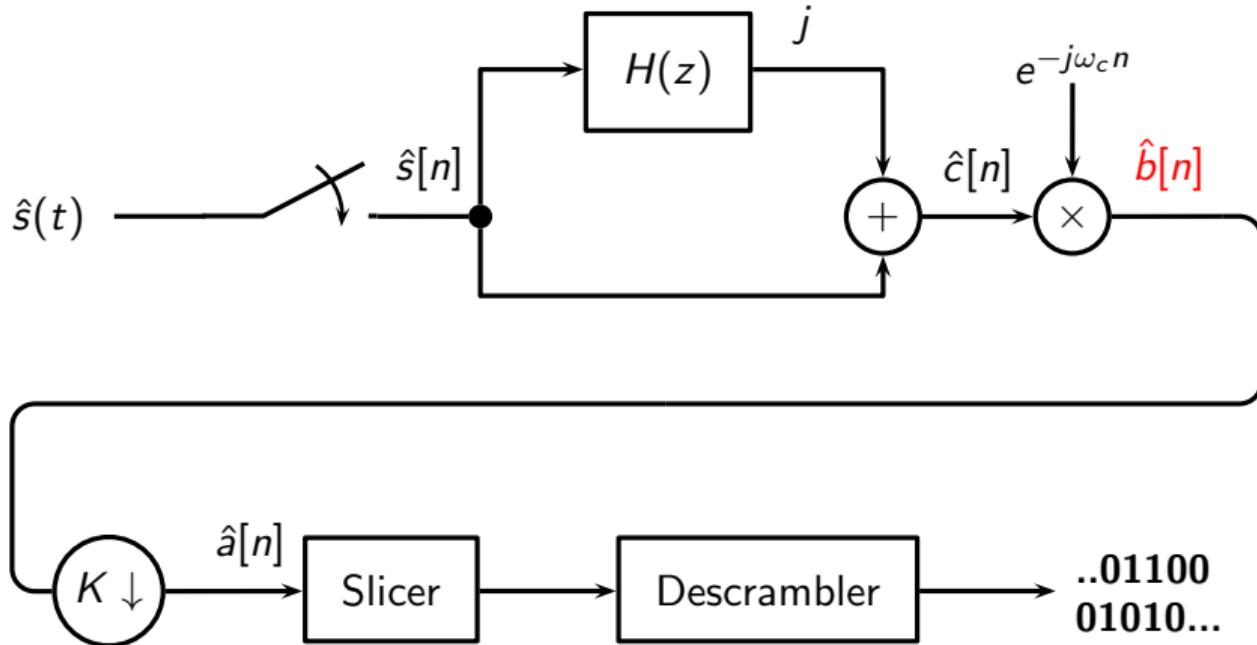
Play

- ▶ a sound familiar to anyone who's used a modem or a fax machine
- ▶ what's going on here?

Remember the (simplified) receiver



Remember the (simplified) receiver

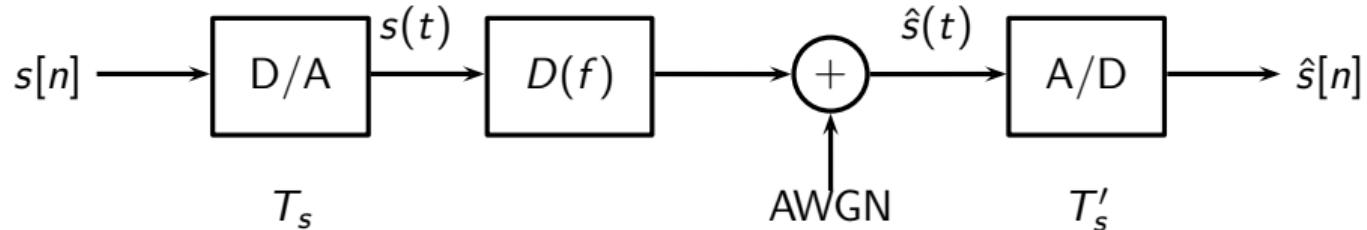


Pilot tones

if $\hat{s}[n] = \cos((\omega_c + \omega_0)n)$ then $\hat{b}[n] = e^{j\omega_0 n}$

Visually, in slow motion, plotting $b[n]$

The main problems



- ▶ noise
- ▶ propagation delay
- ▶ channel distortion $D(f)$
- ▶ different clocks ($T'_s \neq T_s$)

Delay compensation

Assume the channel is a simple delay: $\hat{s}(t) = s(t - d) \Rightarrow D(f) = e^{-j2\pi f d}$

- ▶ channel introduces a delay of d seconds
- ▶ we can write $d = (L + \tau)T_s$ with $L \in \mathbb{N}$ and $|\tau| < 1/2$
- ▶ L is called the *bulk delay*
- ▶ τ is the fractional delay

Delay compensation

Assume the channel is a simple delay: $\hat{s}(t) = s(t - d) \Rightarrow D(f) = e^{-j2\pi f d}$

- ▶ channel introduces a delay of d seconds
- ▶ we can write $d = (L + \tau)T_s$ with $L \in \mathbb{N}$ and $|\tau| < 1/2$
- ▶ L is called the *bulk delay*
- ▶ τ is the fractional delay

Delay compensation

Assume the channel is a simple delay: $\hat{s}(t) = s(t - d) \Rightarrow D(f) = e^{-j2\pi f d}$

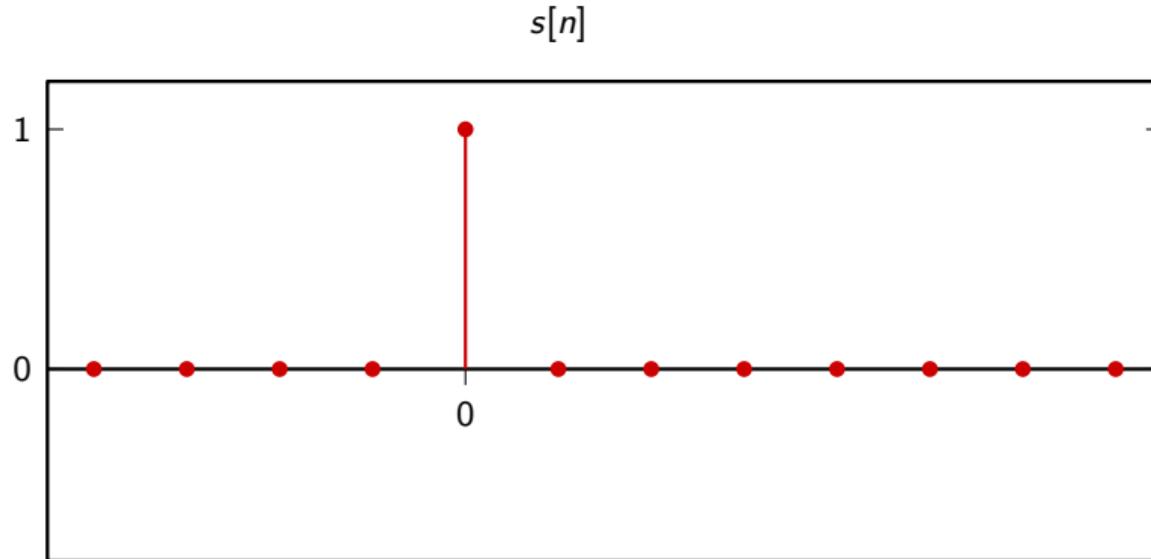
- ▶ channel introduces a delay of d seconds
- ▶ we can write $d = (L + \tau)T_s$ with $L \in \mathbb{N}$ and $|\tau| < 1/2$
- ▶ L is called the *bulk delay*
- ▶ τ is the fractional delay

Delay compensation

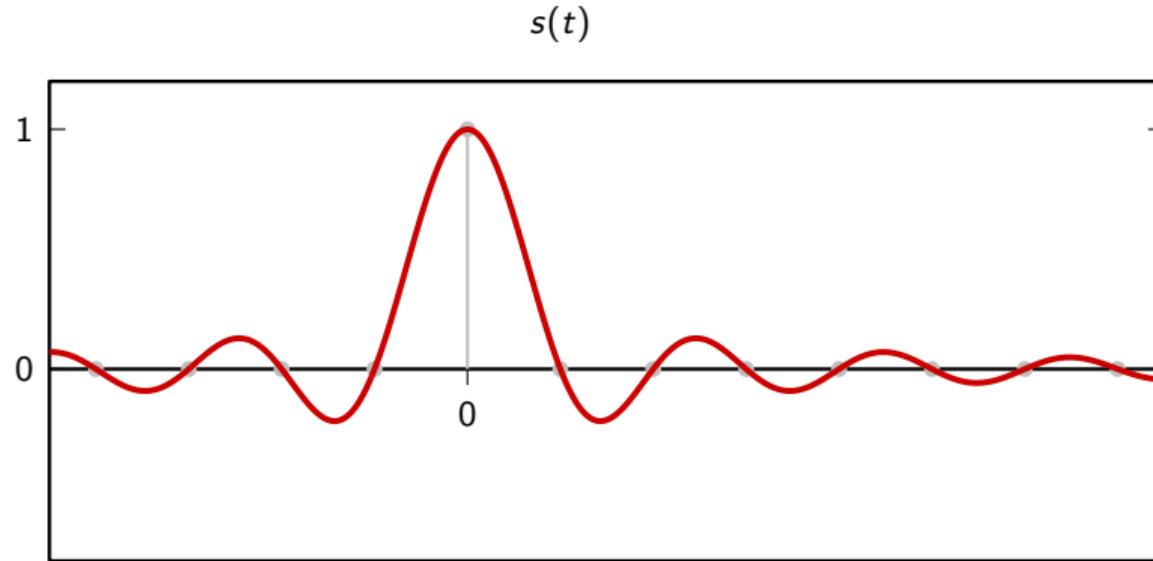
Assume the channel is a simple delay: $\hat{s}(t) = s(t - d) \Rightarrow D(f) = e^{-j2\pi f d}$

- ▶ channel introduces a delay of d seconds
- ▶ we can write $d = (L + \tau)T_s$ with $L \in \mathbb{N}$ and $|\tau| < 1/2$
- ▶ L is called the *bulk delay*
- ▶ τ is the fractional delay

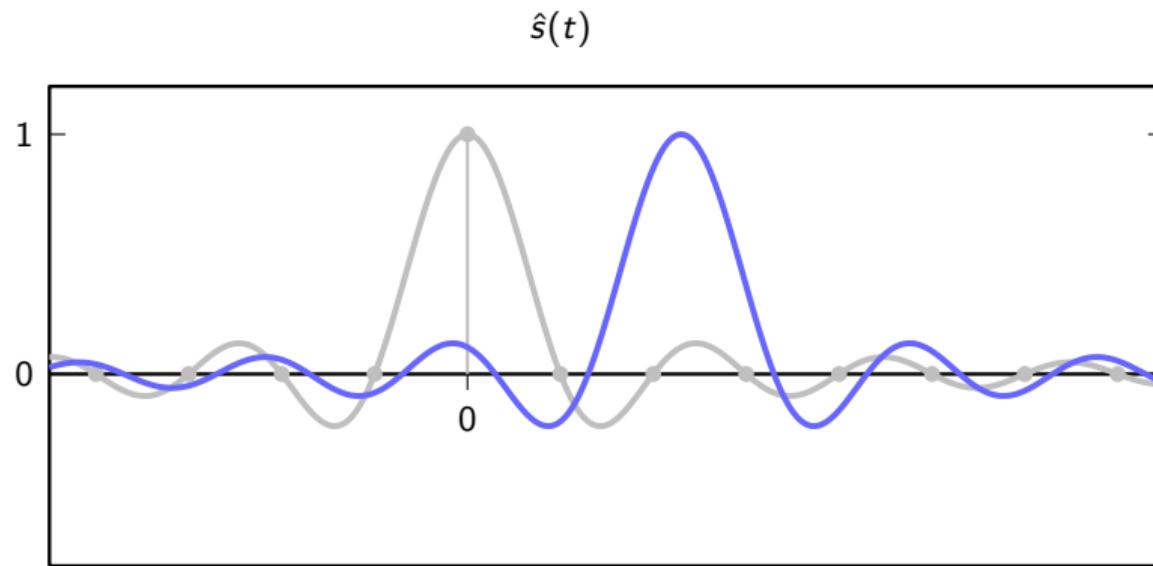
Estimating the bulk delay ($T_s = 1$)



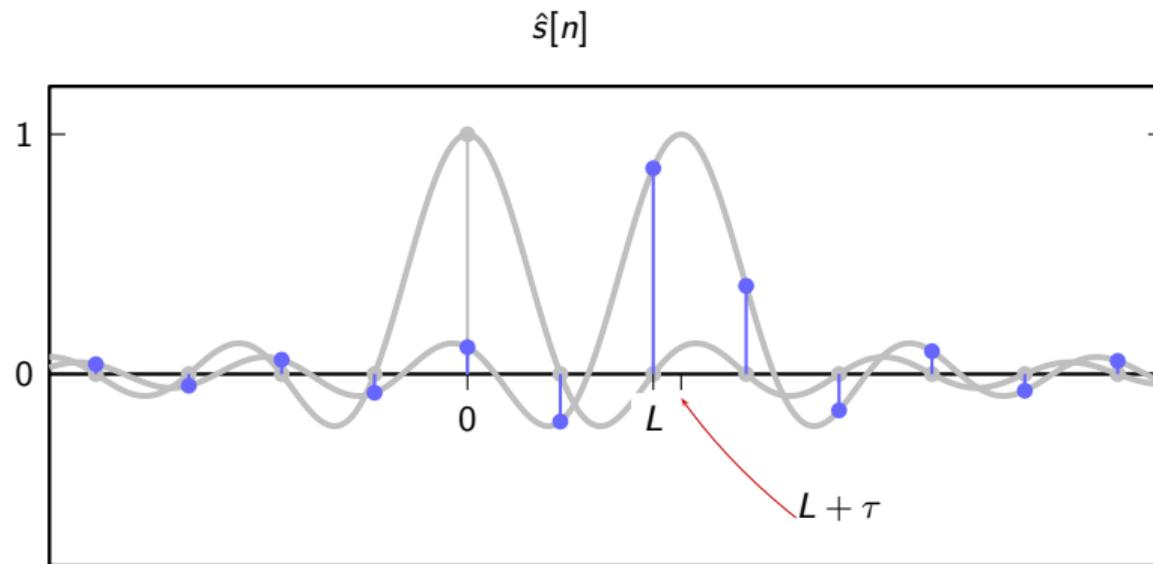
Estimating the bulk delay ($T_s = 1$)



Estimating the bulk delay ($T_s = 1$)



Estimating the bulk delay ($T_s = 1$)



Estimating the fractional delay

► transmit $b[n] = e^{j\omega_0 n}$ (i.e. $s[n] = \cos((\omega_c + \omega_0)n)$)

► receive $\hat{s}[n] = \cos((\omega_c + \omega_0)(n - L - \tau))$

► after demodulation and bulk delay offset:

$$\hat{b}[n] = e^{j\omega_0(n-\tau)}$$

► multiply by known frequency

$$\hat{b}[n] e^{-j\omega_0 n} = e^{-j\omega_0 \tau}$$

Estimating the fractional delay

- ▶ transmit $b[n] = e^{j\omega_0 n}$ (i.e. $s[n] = \cos((\omega_c + \omega_0)n)$)
- ▶ receive $\hat{s}[n] = \cos((\omega_c + \omega_0)(n - L - \tau))$
- ▶ after demodulation and bulk delay offset:

$$\hat{b}[n] = e^{j\omega_0(n-\tau)}$$

- ▶ multiply by known frequency

$$\hat{b}[n] e^{-j\omega_0 n} = e^{-j\omega_0 \tau}$$

Estimating the fractional delay

- ▶ transmit $b[n] = e^{j\omega_0 n}$ (i.e. $s[n] = \cos((\omega_c + \omega_0)n)$)
- ▶ receive $\hat{s}[n] = \cos((\omega_c + \omega_0)(n - L - \tau))$
- ▶ after demodulation and bulk delay offset:

$$\hat{b}[n] = e^{j\omega_0(n-\tau)}$$

- ▶ multiply by known frequency

$$\hat{b}[n] e^{-j\omega_0 n} = e^{-j\omega_0 \tau}$$

Estimating the fractional delay

- ▶ transmit $b[n] = e^{j\omega_0 n}$ (i.e. $s[n] = \cos((\omega_c + \omega_0)n)$)
- ▶ receive $\hat{s}[n] = \cos((\omega_c + \omega_0)(n - L - \tau))$
- ▶ after demodulation and bulk delay offset:

$$\hat{b}[n] = e^{j\omega_0(n-\tau)}$$

- ▶ multiply by known frequency

$$\hat{b}[n] e^{-j\omega_0 n} = e^{-j\omega_0 \tau}$$

Compensating for the fractional delay

- ▶ $\hat{s}[n] = s((n - \tau)T_s)$ (after offsetting bulk delay)
- ▶ we need to compute subsample values
- ▶ in theory, compensate with a sinc fractional delay $h[n] = \text{sinc}(n + \tau)$
- ▶ in practice, use local Lagrange approximation

$$x_L(n; t) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(t)$$

$$L_k^{(N)}(t) = \prod_{\substack{i=-N \\ i \neq n}}^N \frac{t-i}{k-i} \quad k = -N, \dots, N$$

Compensating for the fractional delay

- ▶ $\hat{s}[n] = s((n - \tau)T_s)$ (after offsetting bulk delay)
- ▶ we need to compute subsample values
- ▶ in theory, compensate with a sinc fractional delay $h[n] = \text{sinc}(n + \tau)$
- ▶ in practice, use local Lagrange approximation

$$x_L(n; t) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(t)$$

$$L_k^{(N)}(t) = \prod_{\substack{i=-N \\ i \neq n}}^N \frac{t-i}{k-i} \quad k = -N, \dots, N$$

Compensating for the fractional delay

- ▶ $\hat{s}[n] = s((n - \tau)T_s)$ (after offsetting bulk delay)
- ▶ we need to compute subsample values
- ▶ in theory, compensate with a sinc fractional delay $h[n] = \text{sinc}(n + \tau)$
- ▶ in practice, use local Lagrange approximation

$$x_L(n; t) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(t)$$

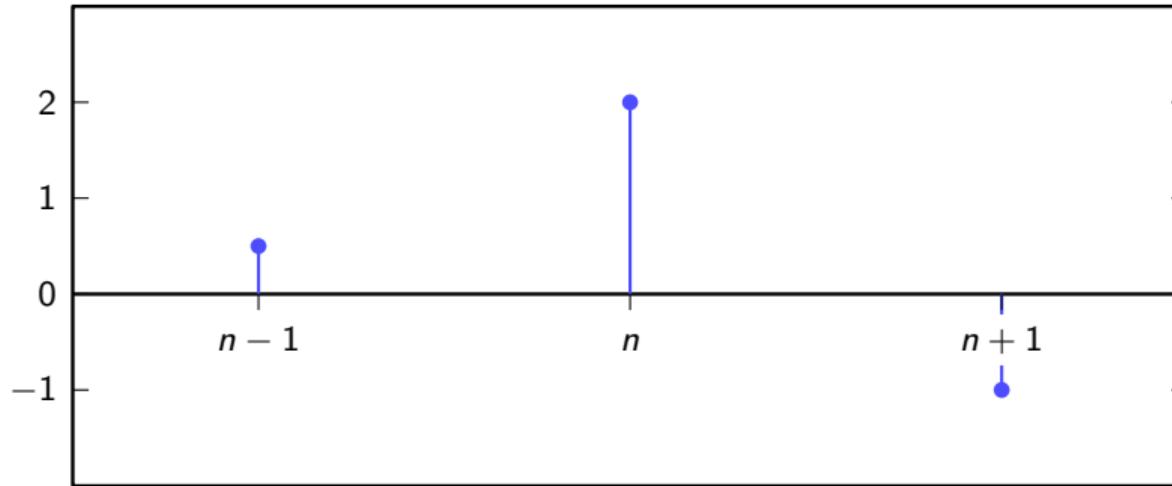
$$L_k^{(N)}(t) = \prod_{\substack{i=-N \\ i \neq n}}^N \frac{t-i}{k-i} \quad k = -N, \dots, N$$

Compensating for the fractional delay

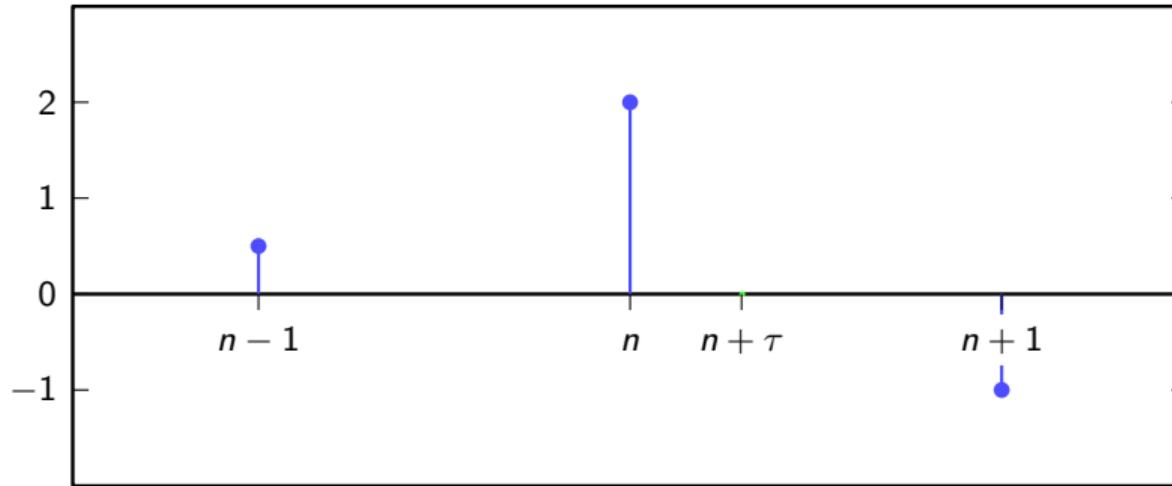
- ▶ $\hat{s}[n] = s((n - \tau)T_s)$ (after offsetting bulk delay)
- ▶ we need to compute subsample values
- ▶ in theory, compensate with a sinc fractional delay $h[n] = \text{sinc}(n + \tau)$
- ▶ in practice, use local Lagrange approximation

$$x_L(n; t) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(t)$$
$$L_k^{(N)}(t) = \prod_{\substack{i=-N \\ i \neq n}}^N \frac{t-i}{k-i} \quad k = -N, \dots, N$$

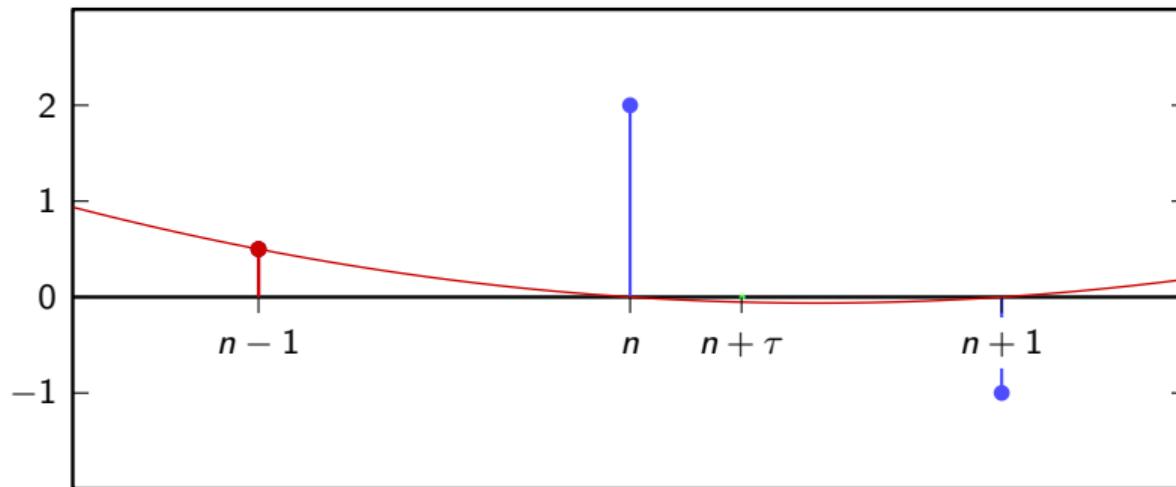
Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



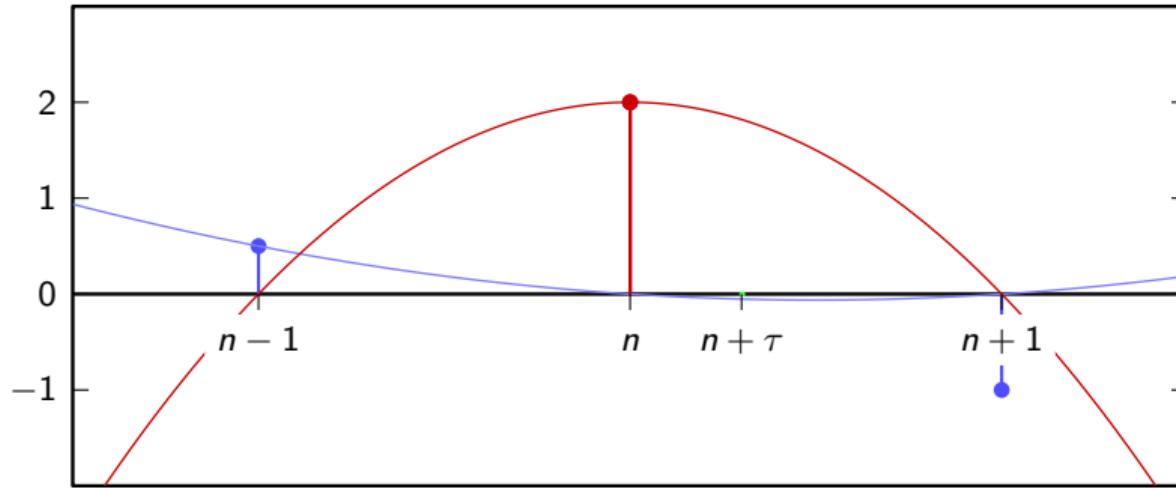
Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



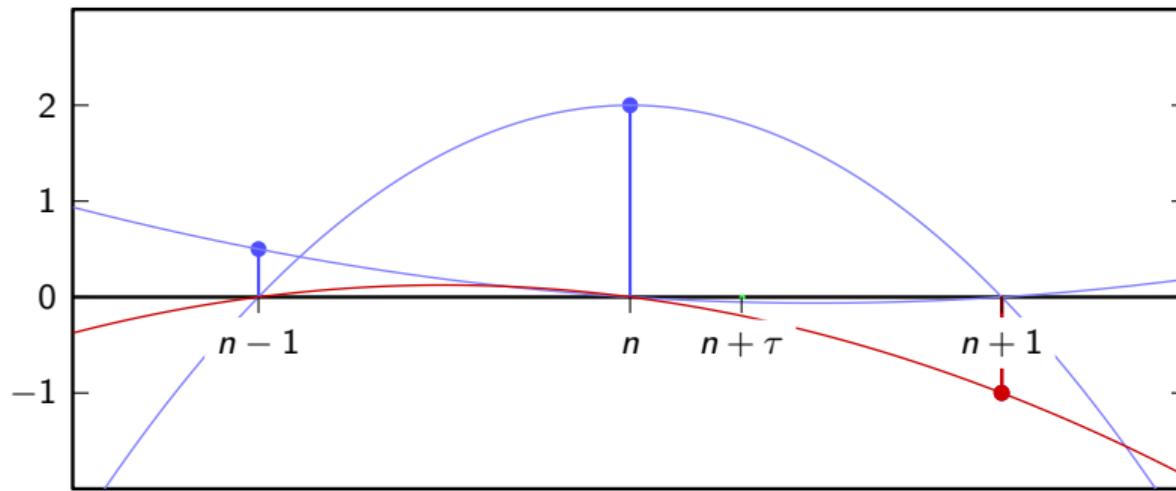
Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



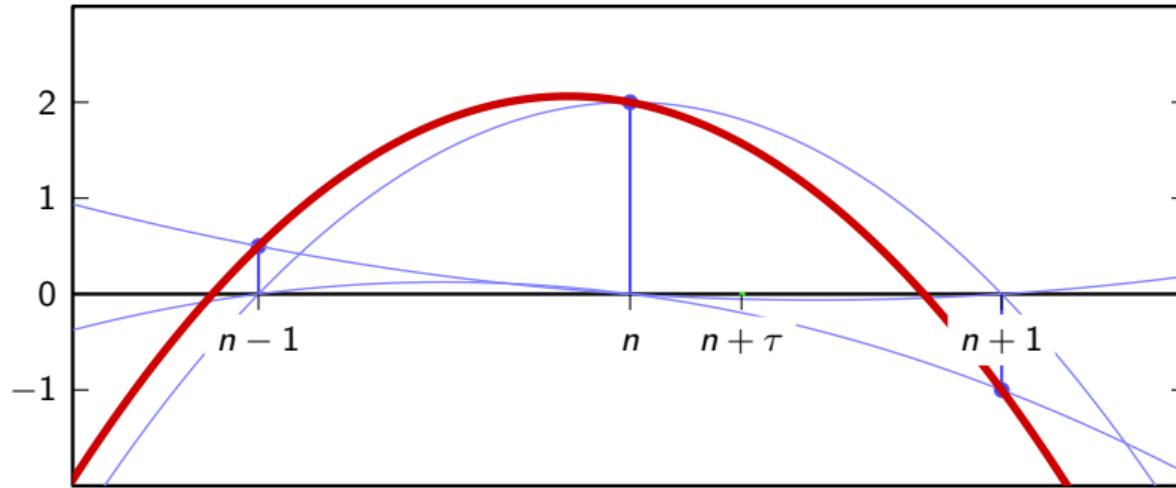
Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



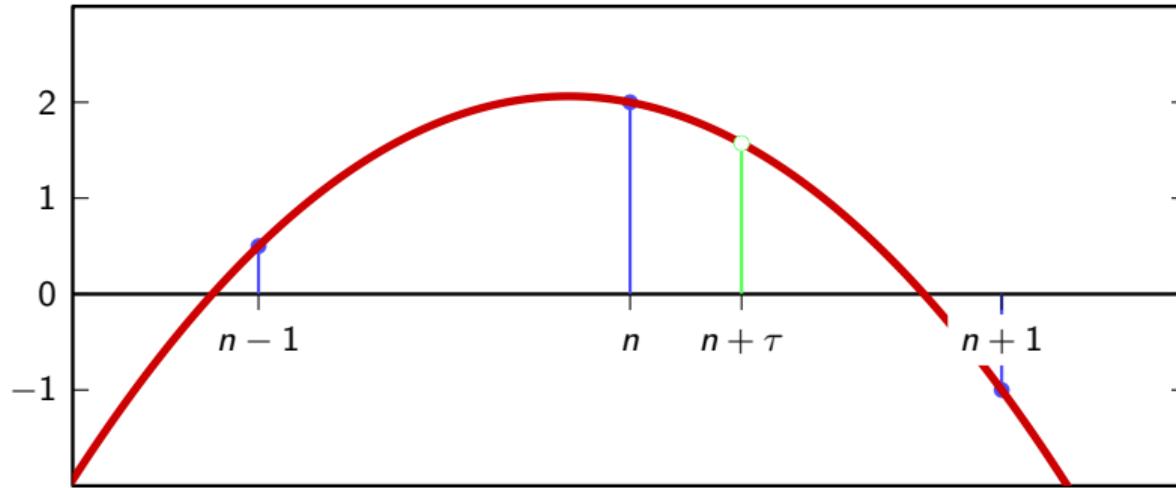
Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



Compensating for the fractional delay: Lagrange interpolation ($N = 1$)



Lagrange interpolation as an FIR

- ▶ $x(n + \tau) \approx x_L(n; \tau)$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^N x[n-k] L_{-k}^{(N)}(\tau)$
- ▶ define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \dots, N$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n-k] d_\tau[k]$
- ▶ $x_L(n; \tau) = (x * d_\tau)[n]$
- ▶ $d_\tau[k]$ is a $(2N + 1)$ -tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- ▶ $x(n + \tau) \approx x_L(n; \tau)$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^N x[n-k] L_{-k}^{(N)}(\tau)$
- ▶ define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \dots, N$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n-k] d_\tau[k]$
- ▶ $x_L(n; \tau) = (x * d_\tau)[n]$
- ▶ $d_\tau[k]$ is a $(2N + 1)$ -tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- ▶ $x(n + \tau) \approx x_L(n; \tau)$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^N x[n-k] L_{-k}^{(N)}(\tau)$
- ▶ define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \dots, N$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n-k] d_\tau[k]$
- ▶ $x_L(n; \tau) = (x * d_\tau)[n]$
- ▶ $d_\tau[k]$ is a $(2N + 1)$ -tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- ▶ $x(n + \tau) \approx x_L(n; \tau)$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^N x[n-k] L_{-k}^{(N)}(\tau)$
- ▶ define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \dots, N$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n-k] d_\tau[k]$
- ▶ $x_L(n; \tau) = (x * d_\tau)[n]$
- ▶ $d_\tau[k]$ is a $(2N + 1)$ -tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- ▶ $x(n + \tau) \approx x_L(n; \tau)$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^N x[n-k] L_{-k}^{(N)}(\tau)$
- ▶ define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \dots, N$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n-k] d_\tau[k]$
- ▶ $x_L(n; \tau) = (x * d_\tau)[n]$
- ▶ $d_\tau[k]$ is a $(2N+1)$ -tap FIR (dependent on τ)

Lagrange interpolation as an FIR

- ▶ $x(n + \tau) \approx x_L(n; \tau)$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n+k] L_k^{(N)}(\tau) = \sum_{k=-N}^N x[n-k] L_{-k}^{(N)}(\tau)$
- ▶ define $d_\tau[k] = L_{-k}^{(N)}(\tau)$, $k = -N, \dots, N$
- ▶ $x_L(n; \tau) = \sum_{k=-N}^N x[n-k] d_\tau[k]$
- ▶ $x_L(n; \tau) = (x * d_\tau)[n]$
- ▶ $d_\tau[k]$ is a $(2N + 1)$ -tap FIR (dependent on τ)

Example ($N = 1$, second order approximation)

$$L_{-1}^{(1)}(t) = t \frac{t - 1}{2}$$

$$L_0^{(1)}(t) = (1 - t)(1 + t)$$

$$L_1^{(1)}(t) = t \frac{t + 1}{2}$$

Example ($N = 1$, second order approximation)

$$d_{0.2}[n] = \begin{cases} 0.12 & n = -1 \\ 0.96 & n = 0 \\ -0.08 & n = 1 \\ 0 & \text{otherwise} \end{cases}$$

Delay compensation algorithm

- ▶ estimate the delay τ
- ▶ compute the $2N + 1$ Lagrangian coefficients
- ▶ filter with the resulting FIR

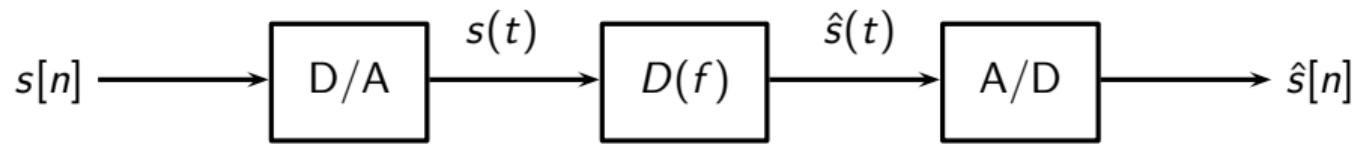
Delay compensation algorithm

- ▶ estimate the delay τ
- ▶ compute the $2N + 1$ Lagrangian coefficients
- ▶ filter with the resulting FIR

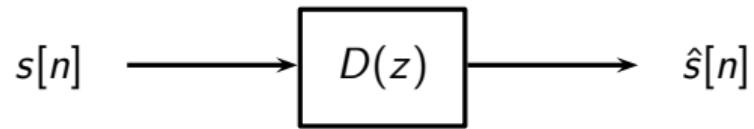
Delay compensation algorithm

- ▶ estimate the delay τ
- ▶ compute the $2N + 1$ Lagrangian coefficients
- ▶ filter with the resulting FIR

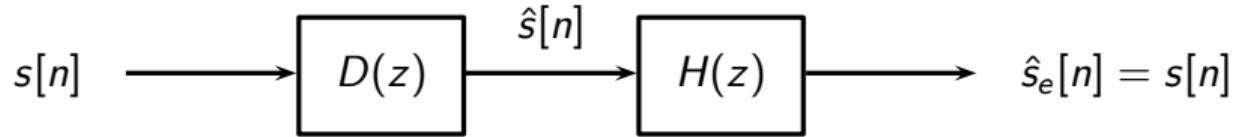
Compensating for the distortion



Compensating for the distortion



Example: adaptive equalization



Example: adaptive equalization

- ▶ in theory, $H(z) = 1/D(z)$
- ▶ but we don't know $D(z)$ in advance
- ▶ $D(z)$ may change over time

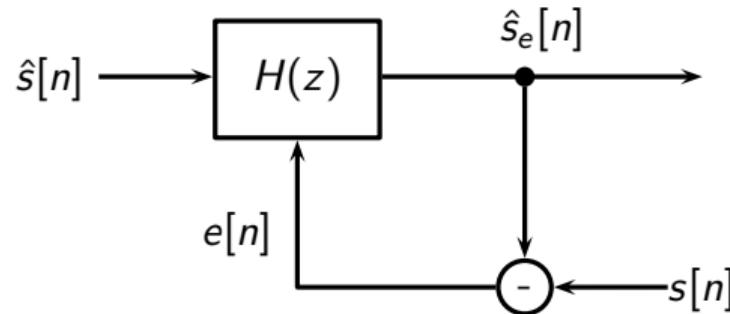
Example: adaptive equalization

- ▶ in theory, $H(z) = 1/D(z)$
- ▶ but we don't know $D(z)$ in advance
- ▶ $D(z)$ may change over time

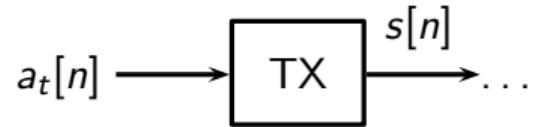
Example: adaptive equalization

- ▶ in theory, $H(z) = 1/D(z)$
- ▶ but we don't know $D(z)$ in advance
- ▶ $D(z)$ may change over time

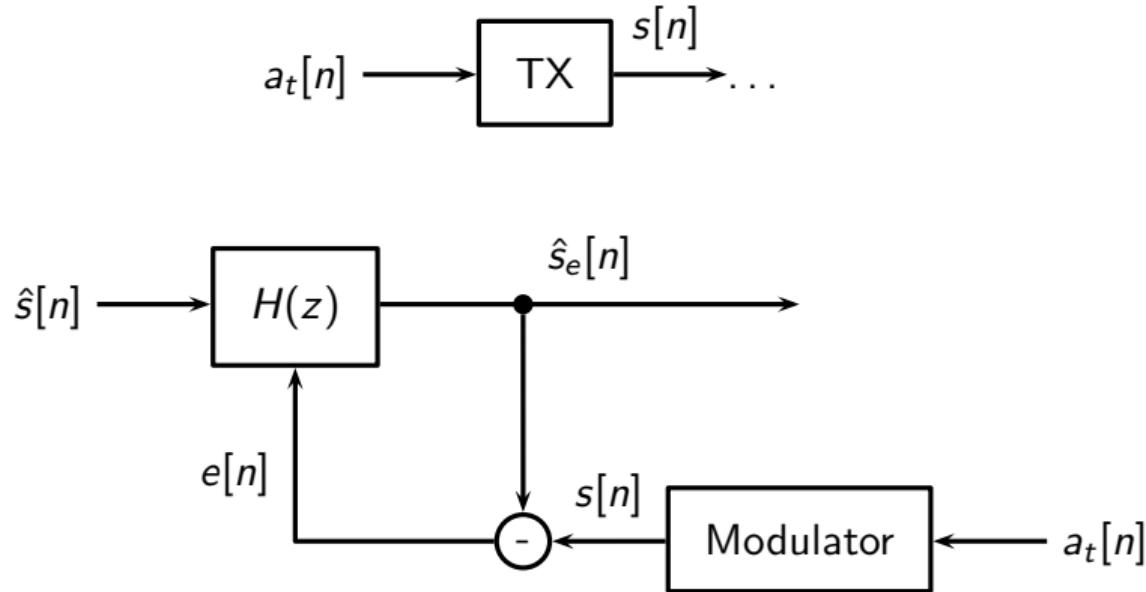
Adaptive equalization



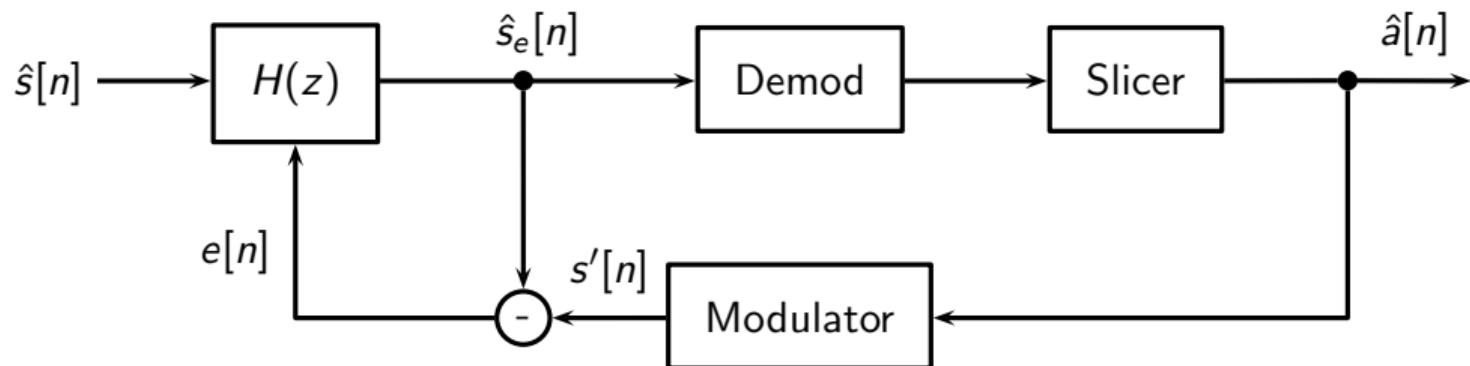
Adaptive equalization: bootstrapping via a training sequence



Adaptive equalization: bootstrapping via a training sequence



Adaptive equalization: online mode



Adaptive equalization: the LMS algorithm

FIR equalization:

$$\begin{aligned}\hat{s}_e[n] &= \sum_{k=0}^{N-1} h_n[k] \hat{s}[n - k] \\ e[n] &= \hat{s}_e[n] - s[n]\end{aligned}$$

adapting the coefficients:

$$h_{n+1}[k] = h_n[k] + \alpha e[n] x[n - k], \quad k = 0, 1, \dots, N - 1$$

So much more to do...

- ▶ how do we compensate for differences in clocks?
- ▶ how do we recover from interference?
- ▶ how do we improve resilience to noise?

advanced topics in communication system design

So much more to do...

- ▶ how do we compensate for differences in clocks?
- ▶ how do we recover from interference?
- ▶ how do we improve resilience to noise?

advanced topics in communication system design

So much more to do...

- ▶ how do we compensate for differences in clocks?
- ▶ how do we recover from interference?
- ▶ how do we improve resilience to noise?

advanced topics in communication system design

So much more to do...

- ▶ how do we compensate for differences in clocks?
- ▶ how do we recover from interference?
- ▶ how do we improve resilience to noise?

advanced topics in communication system design

So much more to do...

- ▶ how do we compensate for differences in clocks?
- ▶ how do we recover from interference?
- ▶ how do we improve resilience to noise?

advanced topics in communication system design

ADSL

Overview:

- ▶ Channel
- ▶ Signaling strategy
- ▶ Discrete Multitone Modulation (DMT)

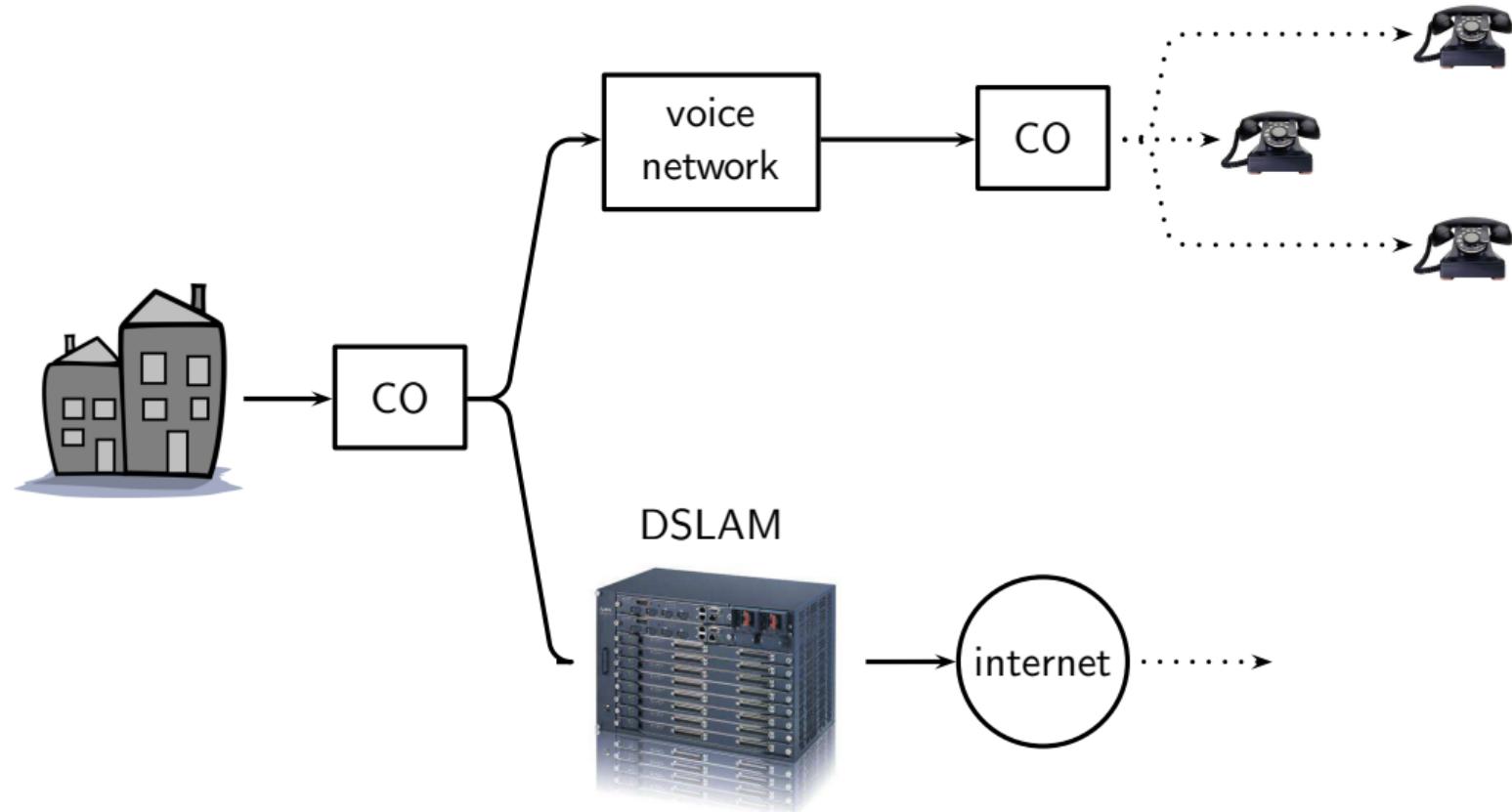
Overview:

- ▶ Channel
- ▶ Signaling strategy
- ▶ Discrete Multitone Modulation (DMT)

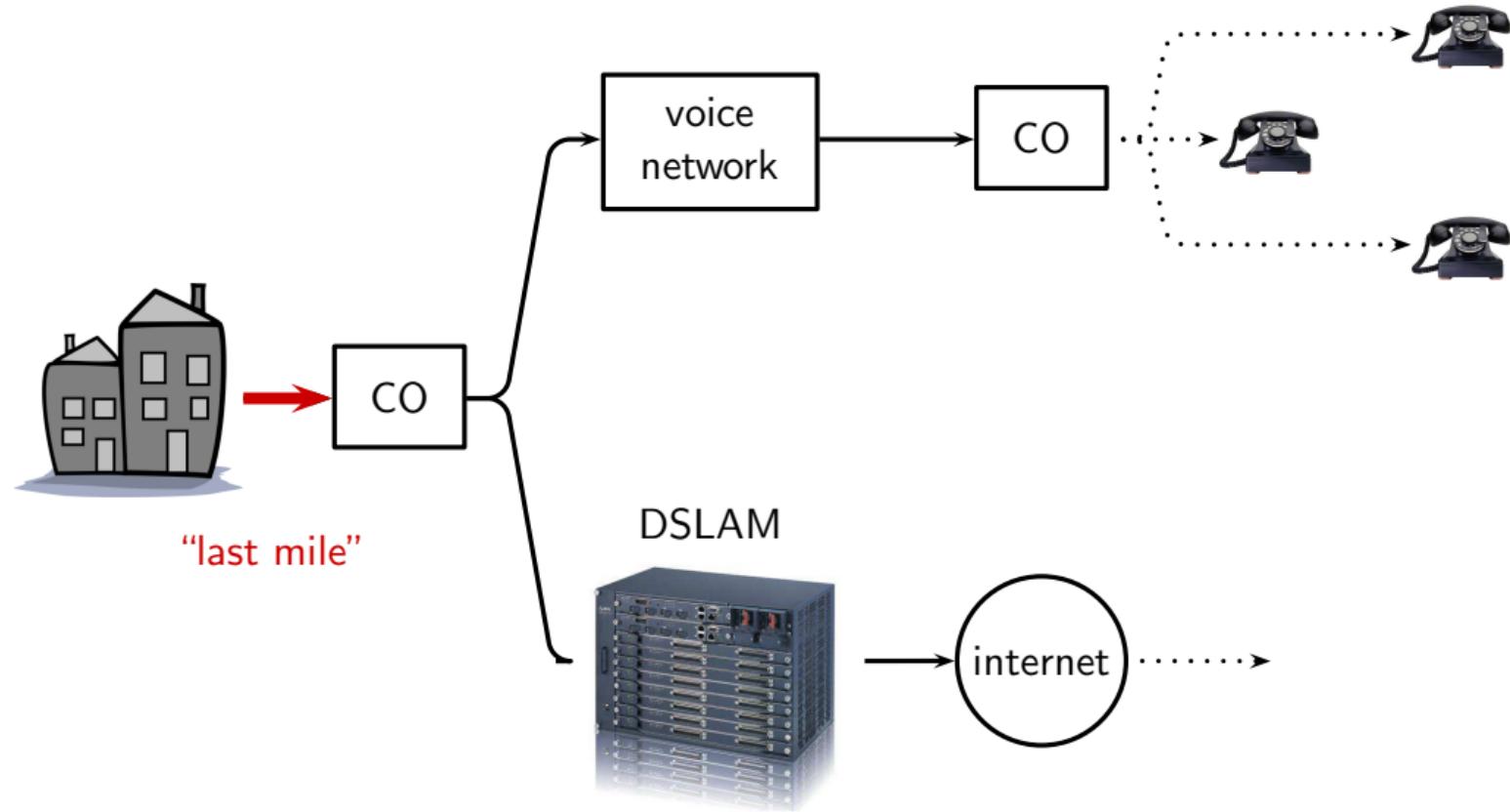
Overview:

- ▶ Channel
- ▶ Signaling strategy
- ▶ Discrete Multitone Modulation (DMT)

The telephone network today



The telephone network today



The last mile

- ▶ copper wire (twisted pair) between home and nearest CO
- ▶ very large bandwidth (well over 1MHz)
- ▶ very uneven spectrum: noise, attenuation, interference, etc.

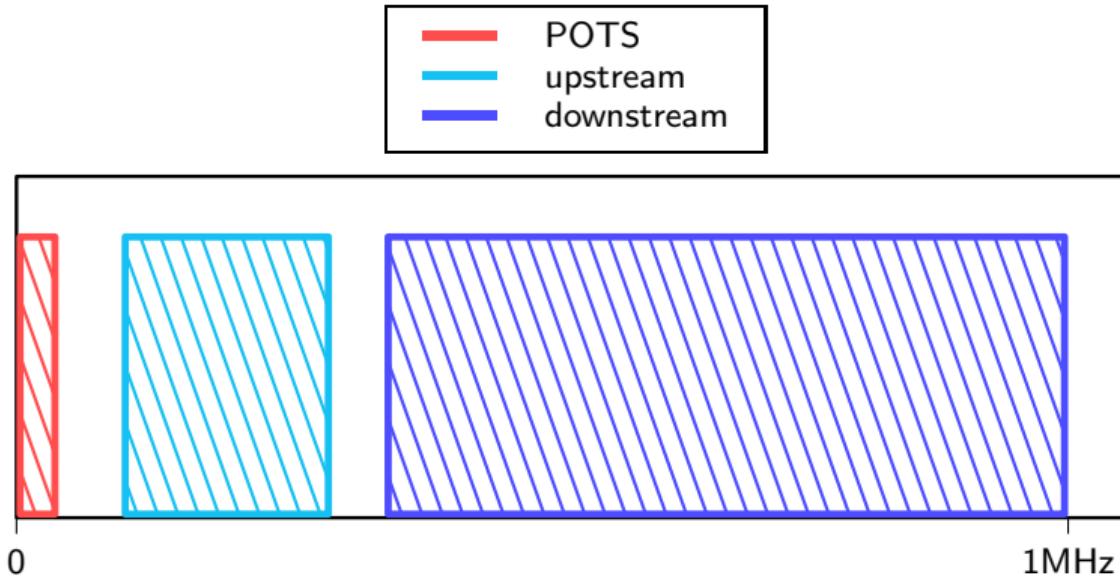
The last mile

- ▶ copper wire (twisted pair) between home and nearest CO
- ▶ very large bandwidth (well over 1MHz)
- ▶ very uneven spectrum: noise, attenuation, interference, etc.

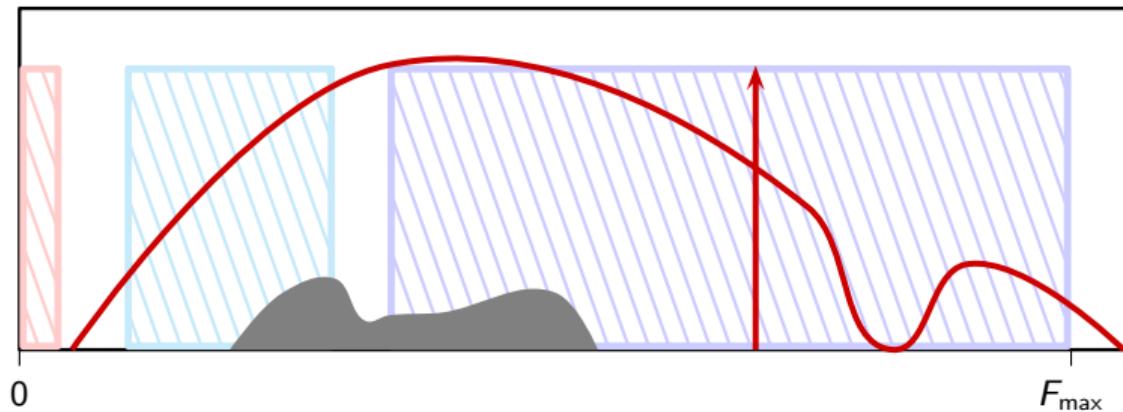
The last mile

- ▶ copper wire (twisted pair) between home and nearest CO
- ▶ very large bandwidth (well over 1MHz)
- ▶ very uneven spectrum: noise, attenuation, interference, etc.

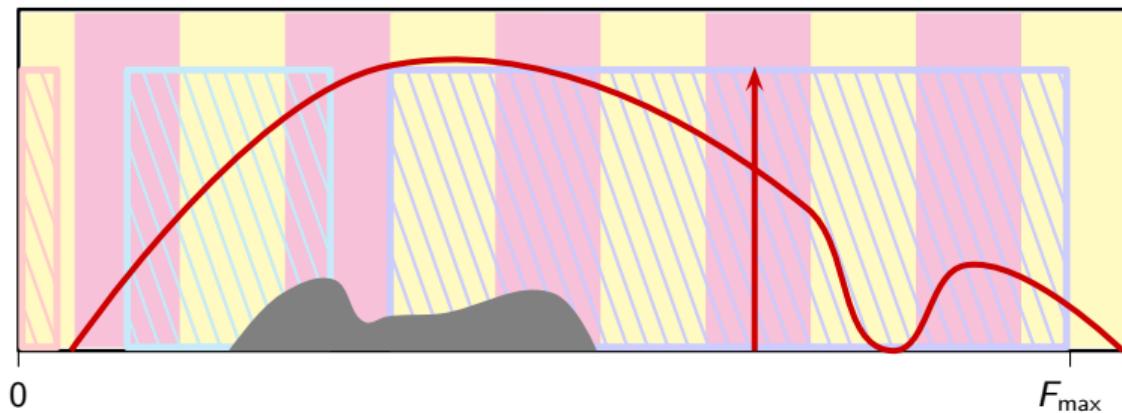
The ADSL channel



The ADSL channel



Idea: split the band into independent subchannels



Subchannel structure

- ▶ allocate N subchannels over the total positive bandwidth
- ▶ equal subchannel bandwidth $W = F_{\max}/N$
- ▶ equally spaced subchannels with center frequency kF_{\max}/N , $k = 0, \dots, N - 1$

Subchannel structure

- ▶ allocate N subchannels over the total positive bandwidth
- ▶ equal subchannel bandwidth $W = F_{\max}/N$
- ▶ equally spaced subchannels with center frequency kF_{\max}/N , $k = 0, \dots, N - 1$

Subchannel structure

- ▶ allocate N subchannels over the total positive bandwidth
- ▶ equal subchannel bandwidth $W = F_{\max}/N$
- ▶ equally spaced subchannels with center frequency kF_{\max}/N , $k = 0, \dots, N - 1$

The digital design

- ▶ pick $F_s = 2F_{\max}$ ($F_s = 2NW = 2F_{\max}$. i.e. F_s multiple of W)
- ▶ center frequency for each subchannel $\omega_k = 2\pi \frac{kW}{F_s} = \frac{2\pi}{2N}k$
- ▶ bandwidth of each subchannel $\frac{2\pi}{2N}$
- ▶ to send symbols over a subchannel: upsampling factor $K = 2N$

The digital design

- ▶ pick $F_s = 2F_{\max}$ ($F_s = 2NW = 2F_{\max}$. i.e. F_s multiple of W)
- ▶ center frequency for each subchannel $\omega_k = 2\pi \frac{kW}{F_s} = \frac{2\pi}{2N}k$
- ▶ bandwidth of each subchannel $\frac{2\pi}{2N}$
- ▶ to send symbols over a subchannel: upsampling factor $K = 2N$

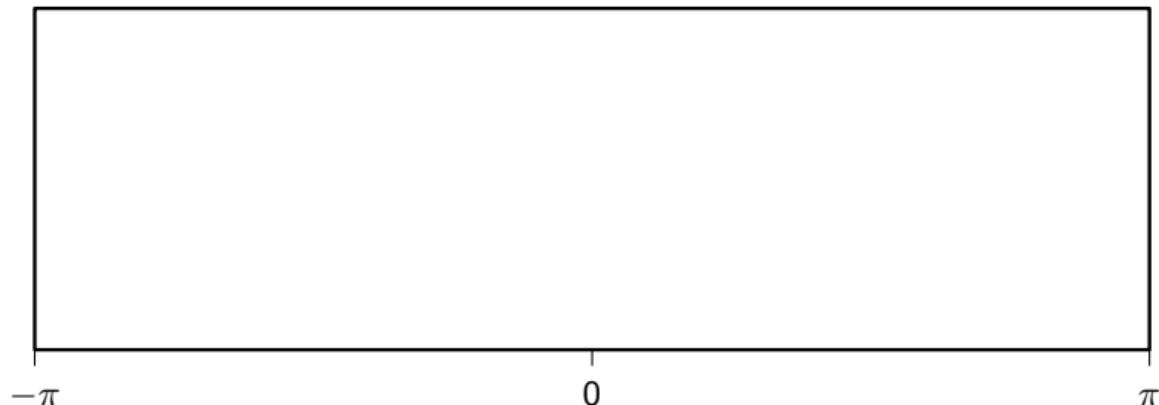
The digital design

- ▶ pick $F_s = 2F_{\max}$ ($F_s = 2NW = 2F_{\max}$. i.e. F_s multiple of W)
- ▶ center frequency for each subchannel $\omega_k = 2\pi \frac{kW}{F_s} = \frac{2\pi}{2N}k$
- ▶ bandwidth of each subchannel $\frac{2\pi}{2N}$
- ▶ to send symbols over a subchannel: upsampling factor $K = 2N$

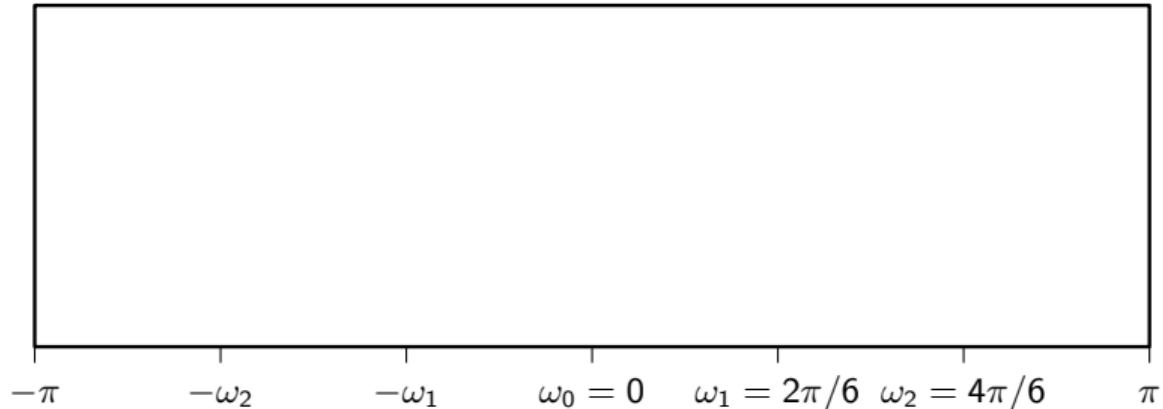
The digital design

- ▶ pick $F_s = 2F_{\max}$ ($F_s = 2NW = 2F_{\max}$. i.e. F_s multiple of W)
- ▶ center frequency for each subchannel $\omega_k = 2\pi \frac{kW}{F_s} = \frac{2\pi}{2N}k$
- ▶ bandwidth of each subchannel $\frac{2\pi}{2N}$
- ▶ to send symbols over a subchannel: upsampling factor $K = 2N$

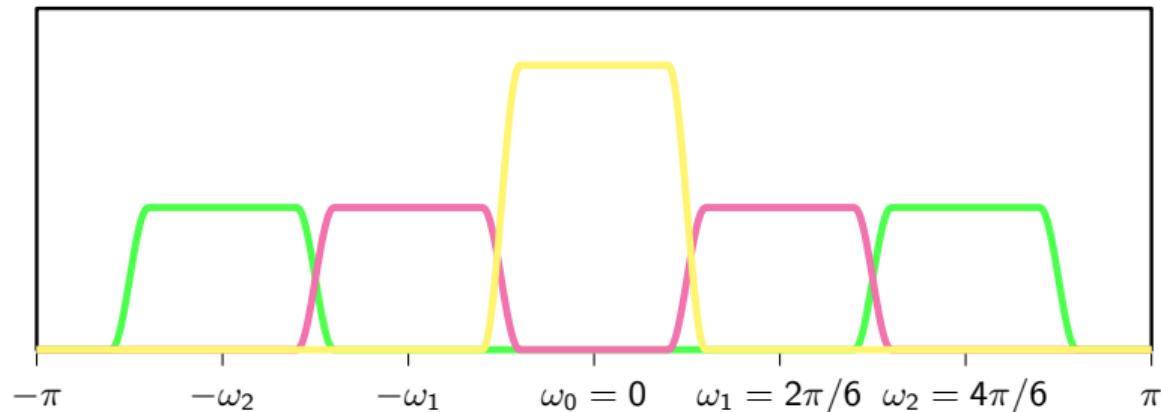
The digital design ($N = 3$)



The digital design ($N = 3$)



The digital design ($N = 3$)



The digital design

- ▶ put a QAM modem on each channel
- ▶ decide on constellation size independently
- ▶ noisy or forbidden subchannels send zeros

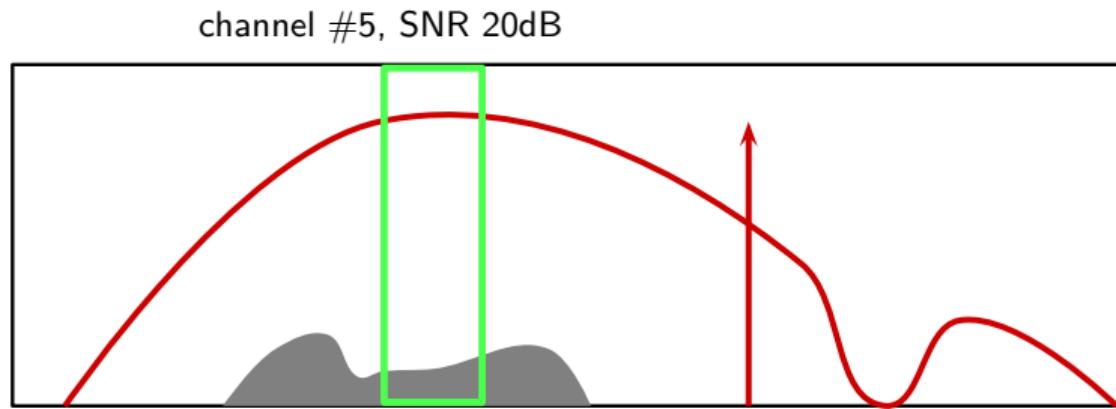
The digital design

- ▶ put a QAM modem on each channel
- ▶ decide on constellation size independently
- ▶ noisy or forbidden subchannels send zeros

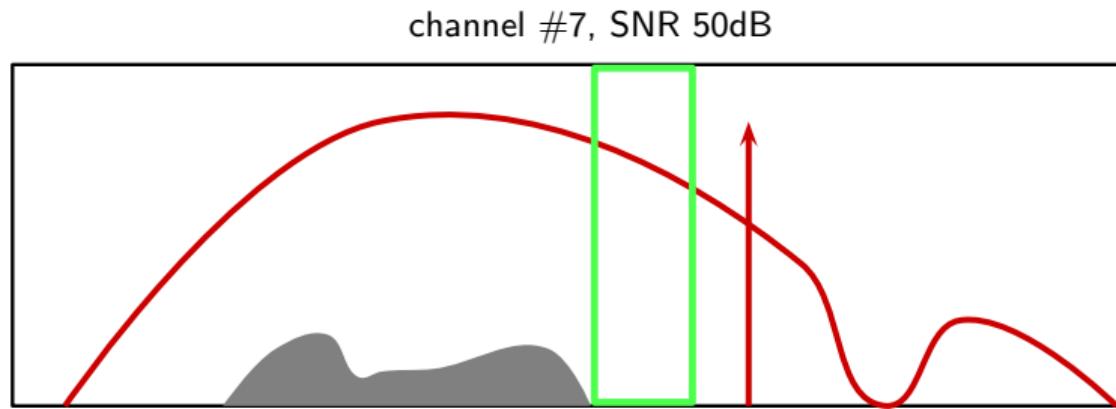
The digital design

- ▶ put a QAM modem on each channel
- ▶ decide on constellation size independently
- ▶ noisy or forbidden subchannels send zeros

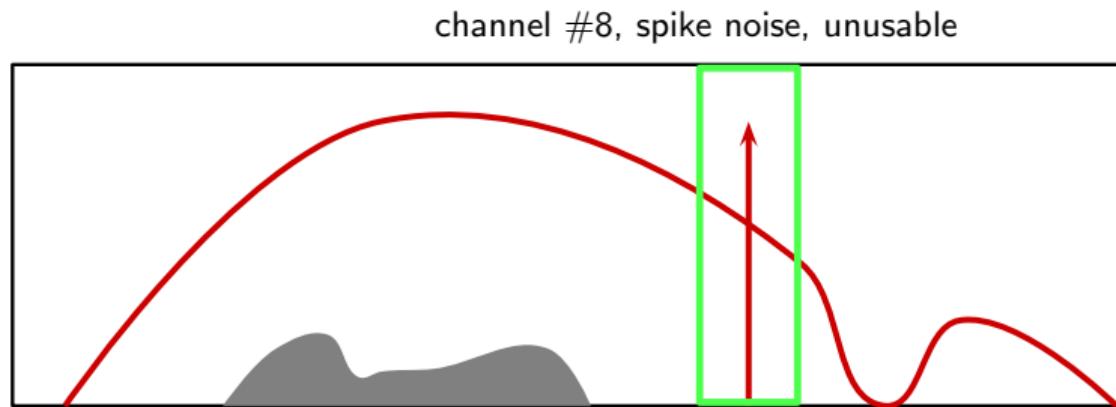
Idea: split the band into independent subchannels



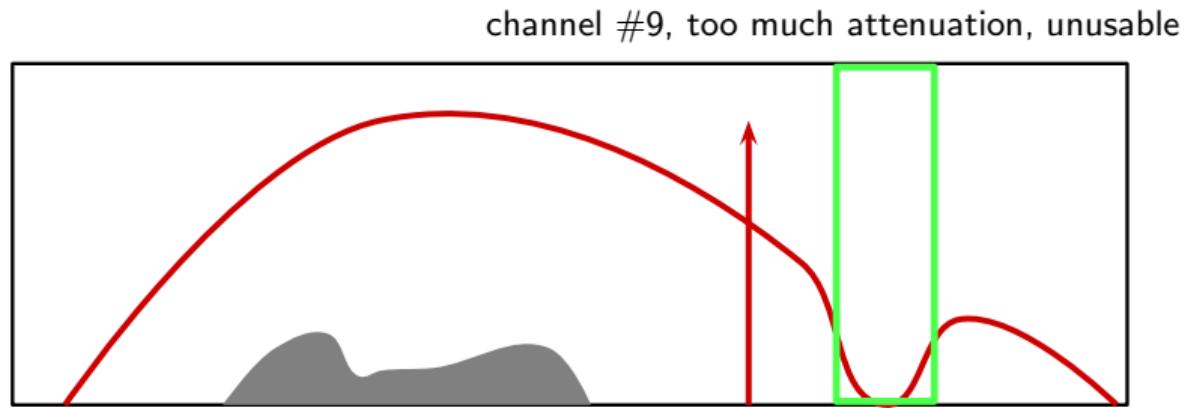
Idea: split the band into independent subchannels



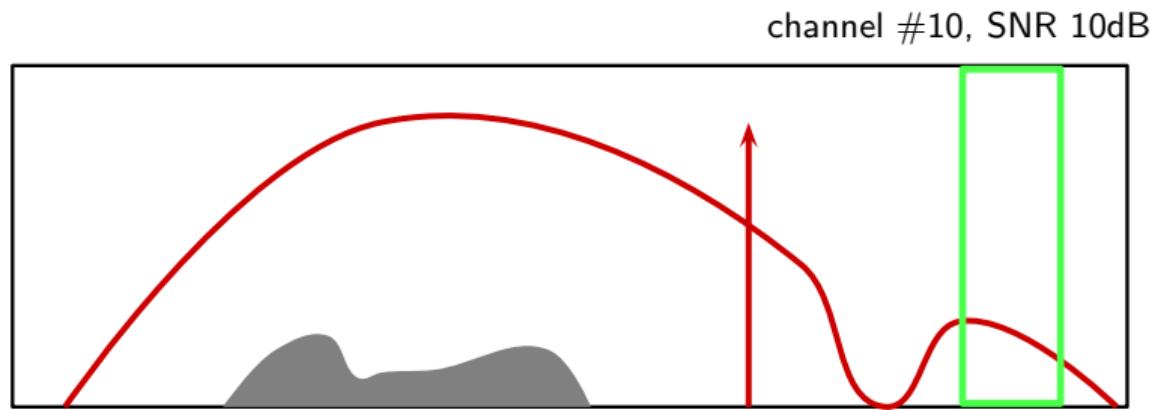
Idea: split the band into independent subchannels



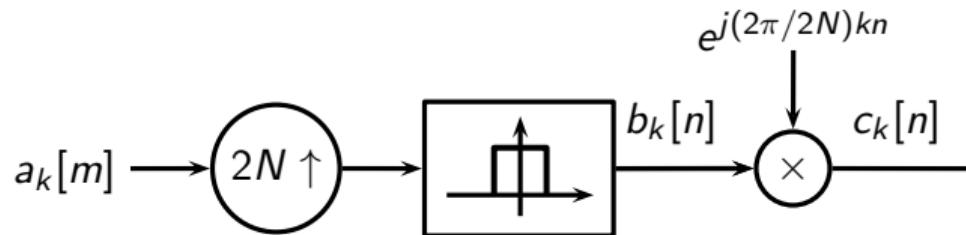
Idea: split the band into independent subchannels



Idea: split the band into independent subchannels



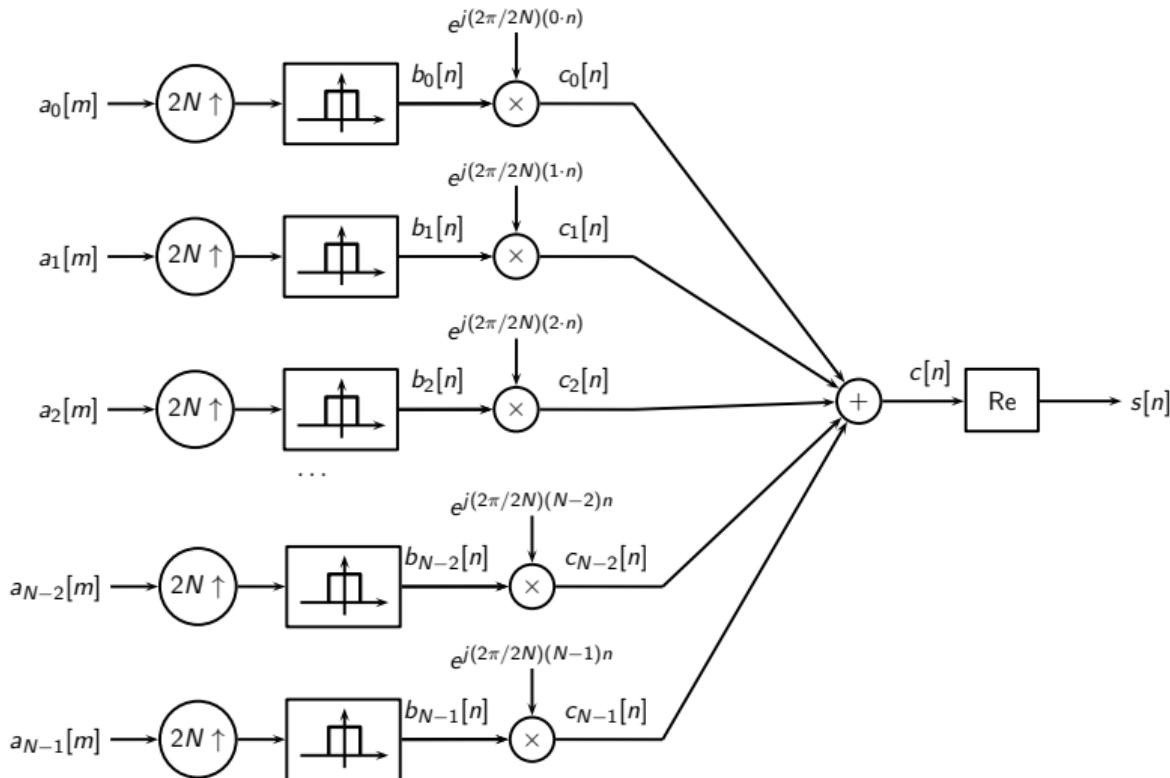
The subchannel modem



rate: W symbols/sec

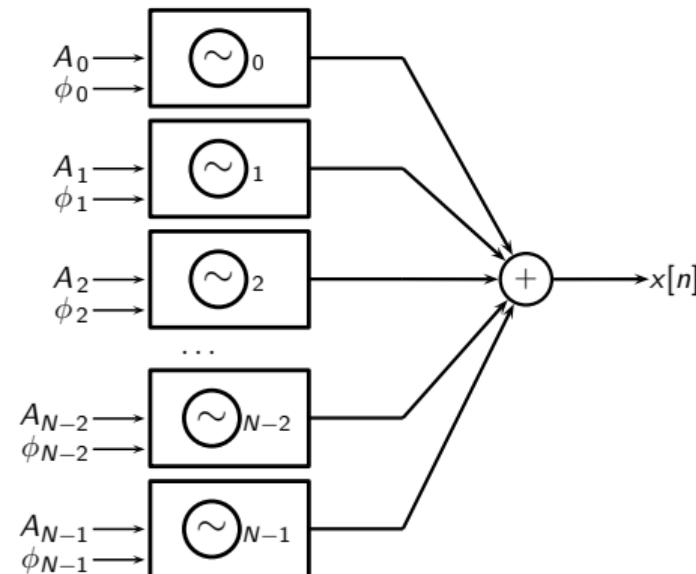
$2NW = F_s$ samples/sec

The bank of modems



If it looks familiar...

remember the DFT reconstruction formula?



DMT via IFFT

- ▶ we will show that transmission can be implemented efficiently via an IFFT
- ▶ Discrete Multitone Modulation

DMT via IFFT

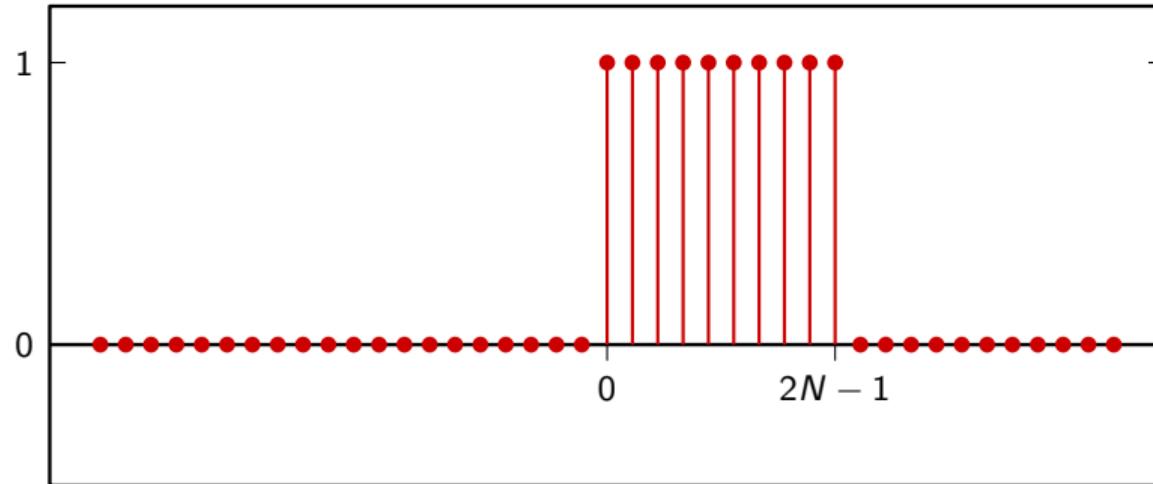
- ▶ we will show that transmission can be implemented efficiently via an IFFT
- ▶ Discrete Multitone Modulation

The great ADSL trick

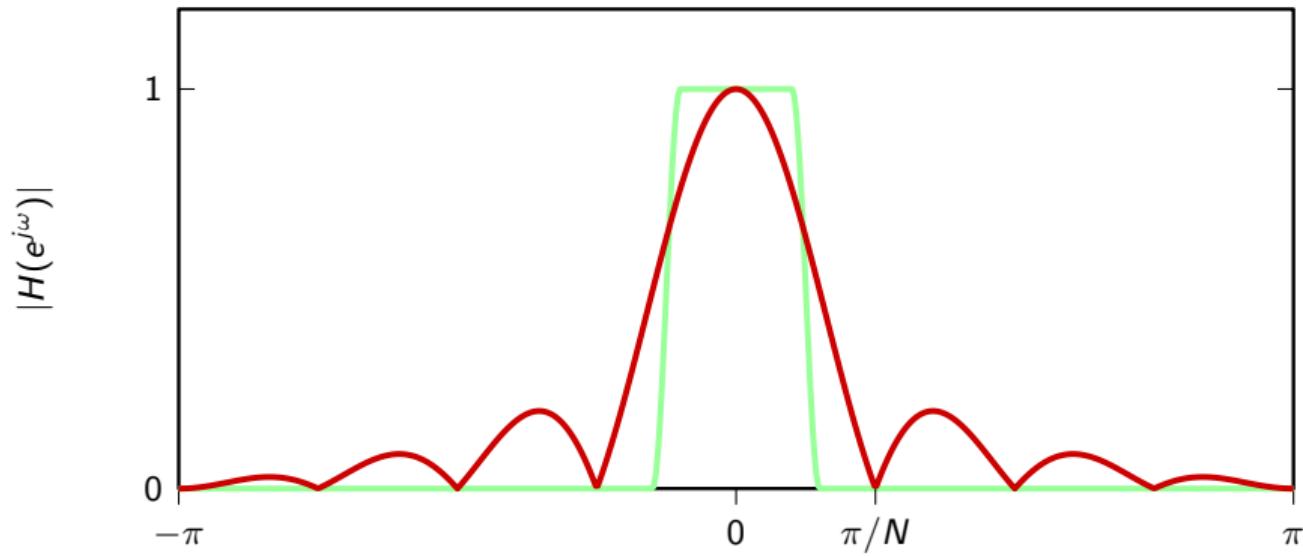
instead of using a “semi-ideal” digital interpolator (e.g. raised cosine), use a zero-order hold:

$$h[n] = \begin{cases} 1 & \text{for } 0 \leq n < 2N \\ 0 & \text{otherwise} \end{cases}$$

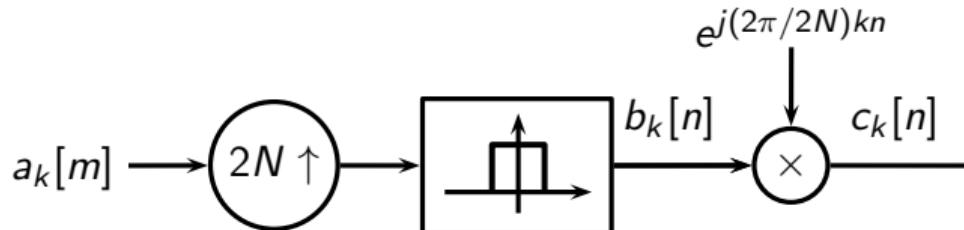
Digital ZOH: interval indicator signal



DTFT of interval signal



Back to the subchannel modem



rate: W symbols/sec

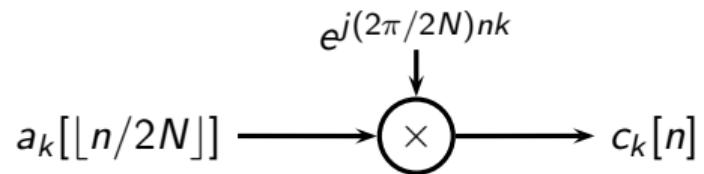
$2NW = F_s$ samples/sec

m changes every $1/W$ sec

n changes every $1/F_s = (1/W)/(2N)$

Back to the subchannel modem

by using the ZOH interpolator:

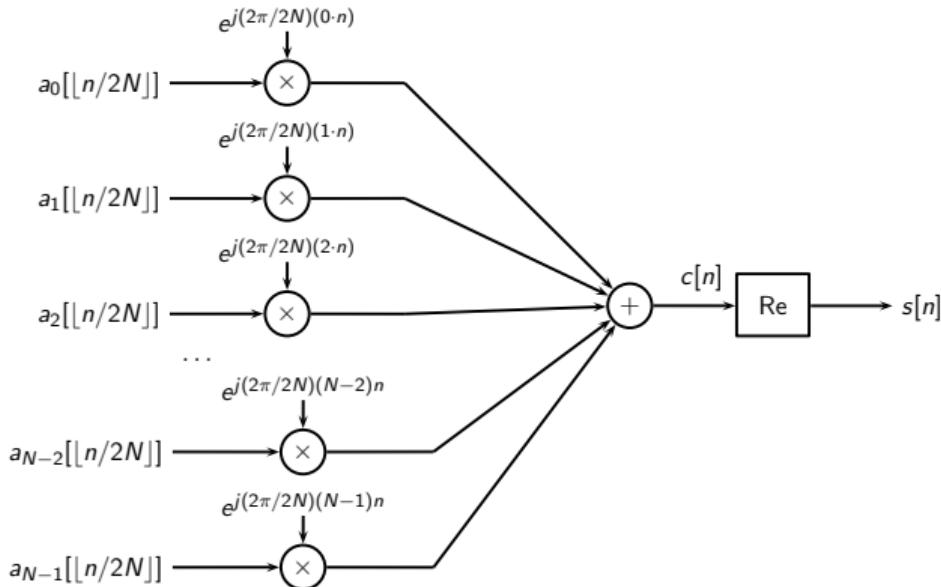


rate: W symbols/sec

$2NW = F_s$ samples/sec

n changes every $1/F_s$

The bank of modems, revisited



rate: $NW = F_{\max}$ symbols/sec

$2NW = F_s$ samples/sec

The complex output signal

$$c[n] = \sum_{k=0}^{N-1} a_k[\lfloor n/2N \rfloor] e^{j \frac{2\pi}{2N} nk}$$

for every successive block of $2N$ output samples:

- ▶ $a_k[\lfloor n/2N \rfloor] = a_k[p]$ doesn't change
- ▶ $c[n] = 2N \cdot \text{IDFT}_{2N} \left\{ [a_0[p] \ a_1[p] \ \dots \ a_{N-1}[p] \ 0 \ 0 \ \dots \ 0] \right\} [n]$

The complex output signal

$$c[n] = \sum_{k=0}^{N-1} a_k[\lfloor n/2N \rfloor] e^{j \frac{2\pi}{2N} nk}$$

for every successive block of $2N$ output samples:

- ▶ $a_k[\lfloor n/2N \rfloor] = a_k[p]$ doesn't change
- ▶ $c[n] = 2N \cdot \text{IDFT}_{2N} \left\{ [a_0[p] \ a_1[p] \ \dots \ a_{N-1}[p] \ 0 \ 0 \ \dots \ 0] \right\} [n]$

The complex output signal: vector space approach

for every successive block of $2N$ output samples:

$$\mathbf{c} = \sum_{k=0}^{N-1} a_k \mathbf{w}^{(k)}$$

- ▶ $\mathbf{c}, \mathbf{w}^{(k)} \in \mathbb{C}^{2N}$
- ▶ $\mathbf{w}^{(k)}$ Fourier basis vector for \mathbb{C}^{2N}
- ▶ orthogonality of basis vector makes detection easy: $a_k = \langle \mathbf{w}^{(k)}, \mathbf{c} \rangle / (2N)$
- ▶ can obtain all a_k 's with a single DFT

Neat implementation detail

- ▶ we are interested in $s[n] = \operatorname{Re}\{c[n]\} = (c[n] + c^*[n])/2$
- ▶ it is easy to prove (exercise) that:

$$\text{IDFT} \left\{ [x_0 \ x_1 \ x_2 \ \dots \ x_{N-2} \ x_{N-1}] \right\}^* = \text{IDFT} \left\{ [x_0 \ x_{N-1} \ x_{N-2} \ \dots \ x_2 \ x_1]^* \right\}$$

- ▶ $c[n] = 2N \cdot \text{IDFT} \left\{ [a_0[p] \ a_1[p] \ \dots \ a_{N-1}[p] \ 0 \ 0 \ \dots \ 0] \right\} [n]$
- ▶ $c^*[n] = 2N \cdot \text{IDFT} \left\{ [a_0[p] \ 0 \ 0 \ \dots \ 0 \ a_{N-1}^*[p] \ \dots \ a_1^*[p]] \right\} [n]$
- ▶ therefore

$$s[n] = N \cdot \text{IDFT} \left\{ [2a_0[p] \ a_1[p] \ \dots \ a_{N-1}[p] \ a_{N-1}^*[p] \ a_{N-2}^*[p] \ \dots \ a_1^*[p]] \right\} [n]$$

Neat implementation detail

- ▶ we are interested in $s[n] = \operatorname{Re}\{c[n]\} = (c[n] + c^*[n])/2$
- ▶ it is easy to prove (exercise) that:

$$\text{IDFT} \left\{ [x_0 \ x_1 \ x_2 \ \dots \ x_{N-2} \ x_{N-1}] \right\}^* = \text{IDFT} \left\{ [x_0 \ x_{N-1} \ x_{N-2} \ \dots \ x_2 \ x_1]^* \right\}$$

- ▶ $c[n] = 2N \cdot \text{IDFT} \left\{ [a_0[p] \ a_1[p] \ \dots \ a_{N-1}[p] \ 0 \ 0 \ \dots \ 0] \right\} [n]$
- ▶ $c^*[n] = 2N \cdot \text{IDFT} \left\{ [a_0[p] \ 0 \ 0 \ \dots \ 0 \ a_{N-1}^*[p] \ \dots \ a_1^*[p]] \right\} [n]$
- ▶ therefore

$$s[n] = N \cdot \text{IDFT} \left\{ [2a_0[p] \ a_1[p] \ \dots \ a_{N-1}[p] \ a_{N-1}^*[p] \ a_{N-2}^*[p] \ \dots \ a_1^*[p]] \right\} [n]$$

Neat implementation detail

- ▶ we are interested in $s[n] = \operatorname{Re}\{c[n]\} = (c[n] + c^*[n])/2$
- ▶ it is easy to prove (exercise) that:

$$\text{IDFT} \left\{ [x_0 \ x_1 \ x_2 \ \dots \ x_{N-2} \ x_{N-1}] \right\}^* = \text{IDFT} \left\{ [x_0 \ x_{N-1} \ x_{N-2} \ \dots \ x_2 \ x_1]^* \right\}$$

- ▶ $c[n] = 2N \cdot \text{IDFT} \left\{ [a_0[p] \ a_1[p] \ \dots \ a_{N-1}[p] \ 0 \ 0 \ \dots \ 0] \right\} [n]$
- ▶ $c^*[n] = 2N \cdot \text{IDFT} \left\{ [a_0[p] \ 0 \ 0 \ \dots \ 0 \ a_{N-1}^*[p] \ \dots \ a_1^*[p]] \right\} [n]$
- ▶ therefore

$$s[n] = N \cdot \text{IDFT} \left\{ [2a_0[p] \ a_1[p] \ \dots \ a_{N-1}[p] \ a_{N-1}^*[p] \ a_{N-2}^*[p] \ \dots \ a_1^*[p]] \right\} [n]$$

Neat implementation detail

- ▶ we are interested in $s[n] = \operatorname{Re}\{c[n]\} = (c[n] + c^*[n])/2$
- ▶ it is easy to prove (exercise) that:

$$\text{IDFT} \left\{ [x_0 \ x_1 \ x_2 \ \dots \ x_{N-2} \ x_{N-1}] \right\}^* = \text{IDFT} \left\{ [x_0 \ x_{N-1} \ x_{N-2} \ \dots \ x_2 \ x_1]^* \right\}$$

- ▶ $c[n] = 2N \cdot \text{IDFT} \left\{ [a_0[p] \ a_1[p] \ \dots \ a_{N-1}[p] \ 0 \ 0 \ \dots \ 0] \right\} [n]$
- ▶ $c^*[n] = 2N \cdot \text{IDFT} \left\{ [a_0[p] \ 0 \ 0 \ \dots \ 0 \ a_{N-1}^*[p] \ \dots \ a_1^*[p]] \right\} [n]$
- ▶ therefore

$$s[n] = N \cdot \text{IDFT} \left\{ [2a_0[p] \ a_1[p] \ \dots \ a_{N-1}[p] \ a_{N-1}^*[p] \ a_{N-2}^*[p] \ \dots \ a_1^*[p]] \right\} [n]$$

Neat implementation detail

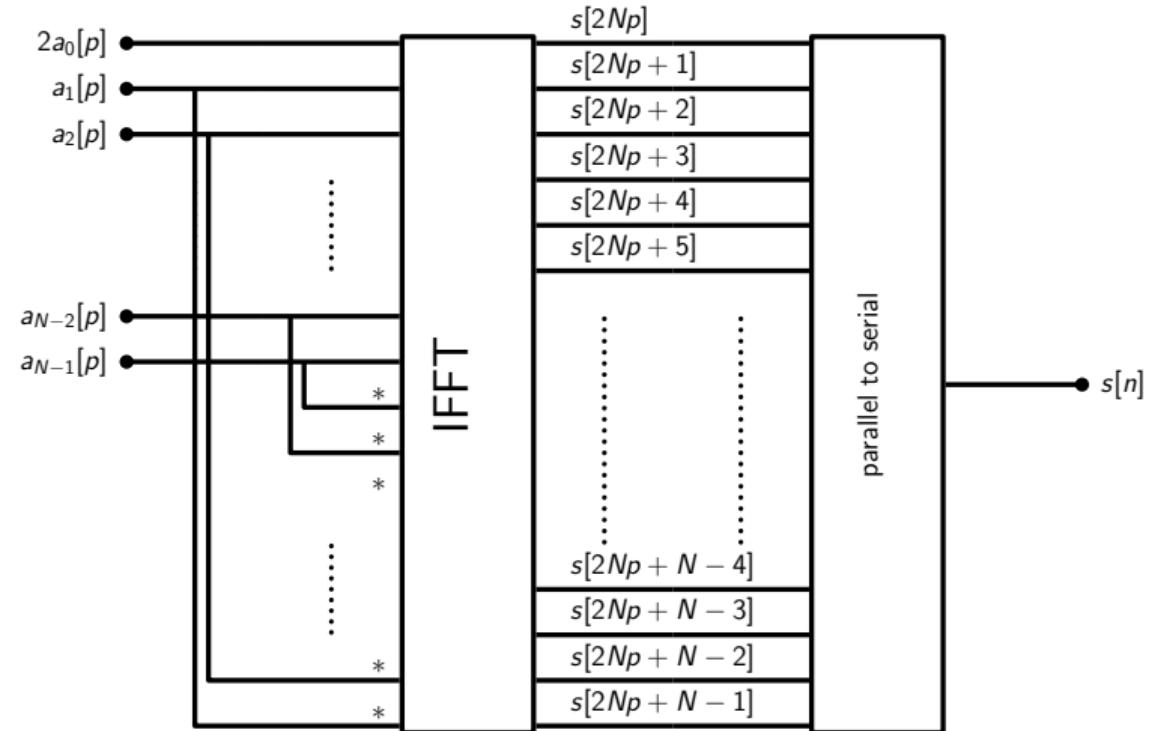
- ▶ we are interested in $s[n] = \operatorname{Re}\{c[n]\} = (c[n] + c^*[n])/2$
- ▶ it is easy to prove (exercise) that:

$$\text{IDFT} \left\{ [x_0 \ x_1 \ x_2 \ \dots \ x_{N-2} \ x_{N-1}] \right\}^* = \text{IDFT} \left\{ [x_0 \ x_{N-1} \ x_{N-2} \ \dots \ x_2 \ x_1]^* \right\}$$

- ▶ $c[n] = 2N \cdot \text{IDFT} \left\{ [a_0[p] \ a_1[p] \ \dots \ a_{N-1}[p] \ 0 \ 0 \ \dots \ 0] \right\} [n]$
- ▶ $c^*[n] = 2N \cdot \text{IDFT} \left\{ [a_0[p] \ 0 \ 0 \ \dots \ 0 \ a_{N-1}^*[p] \ \dots \ a_1^*[p]] \right\} [n]$
- ▶ therefore

$$s[n] = N \cdot \text{IDFT} \left\{ [2a_0[p] \ a_1[p] \ \dots \ a_{N-1}[p] \ a_{N-1}^*[p] \ a_{N-2}^*[p] \ \dots \ a_1^*[p]] \right\} [n]$$

ADSL transmitter



ADSL specs

- ▶ $F_{\max} = 1104 \text{ KHz}$
- ▶ $N = 256$
- ▶ subchannel width $W = 4312.5 \text{ Hz}$
- ▶ each QAM can send from 0 to 15 bits per symbol
- ▶ forbidden channels: 0 to 7 (voice)
- ▶ channels 7 to 31: upstream data
- ▶ max theoretical throughput: 14.9 Mbps (downstream)

ADSL specs

- ▶ $F_{\max} = 1104 \text{ KHz}$
- ▶ $N = 256$
- ▶ subchannel width $W = 4312.5 \text{ Hz}$
- ▶ each QAM can send from 0 to 15 bits per symbol
- ▶ forbidden channels: 0 to 7 (voice)
- ▶ channels 7 to 31: upstream data
- ▶ max theoretical throughput: 14.9 Mbps (downstream)

ADSL specs

- ▶ $F_{\max} = 1104 \text{ KHz}$
- ▶ $N = 256$
- ▶ subchannel width $W = 4312.5 \text{ Hz}$
- ▶ each QAM can send from 0 to 15 bits per symbol
- ▶ forbidden channels: 0 to 7 (voice)
- ▶ channels 7 to 31: upstream data
- ▶ max theoretical throughput: 14.9 Mbps (downstream)

ADSL specs

- ▶ $F_{\max} = 1104 \text{ KHz}$
- ▶ $N = 256$
- ▶ subchannel width $W = 4312.5 \text{ Hz}$
- ▶ each QAM can send from 0 to 15 bits per symbol
- ▶ forbidden channels: 0 to 7 (voice)
- ▶ channels 7 to 31: upstream data
- ▶ max theoretical throughput: 14.9 Mbps (downstream)

ADSL specs

- ▶ $F_{\max} = 1104 \text{ KHz}$
- ▶ $N = 256$
- ▶ subchannel width $W = 4312.5 \text{ Hz}$
- ▶ each QAM can send from 0 to 15 bits per symbol
- ▶ forbidden channels: 0 to 7 (voice)
- ▶ channels 7 to 31: upstream data
- ▶ max theoretical throughput: 14.9 Mbps (downstream)

ADSL specs

- ▶ $F_{\max} = 1104 \text{ KHz}$
- ▶ $N = 256$
- ▶ subchannel width $W = 4312.5 \text{ Hz}$
- ▶ each QAM can send from 0 to 15 bits per symbol
- ▶ forbidden channels: 0 to 7 (voice)
- ▶ channels 7 to 31: upstream data
- ▶ max theoretical throughput: 14.9 Mbps (downstream)

ADSL specs

- ▶ $F_{\max} = 1104 \text{ KHz}$
- ▶ $N = 256$
- ▶ subchannel width $W = 4312.5 \text{ Hz}$
- ▶ each QAM can send from 0 to 15 bits per symbol
- ▶ forbidden channels: 0 to 7 (voice)
- ▶ channels 7 to 31: upstream data
- ▶ max theoretical throughput: 14.9 Mbps (downstream)