EPFL

COM-202: Signal Processing

Chapter 6.a: Discrete-Time Filters

Overview

- linear time-invariant systems in discrete time
- filters for denoising
- impulse response
- convolution
- frequency response
- modulation and demodulation
- ideal filters and approximations

linear time-invariant systems

A single-input single-output signal processing device

$$\mathbf{y} = \mathcal{H}\mathbf{x}$$

- all signals assumed to be infinite-length
- we need some restrictions on $\mathcal{H}$ to proceed

Linearity

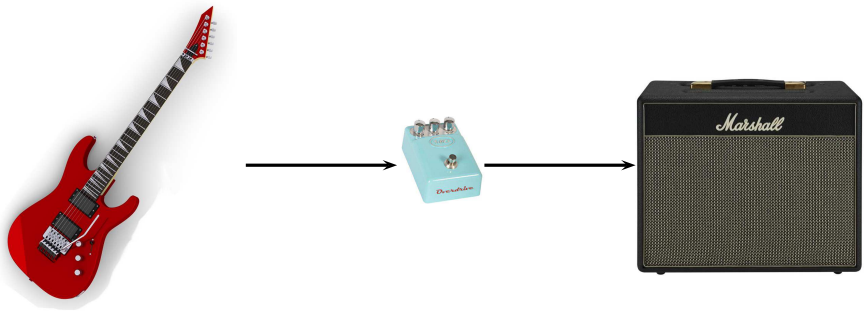
$$\mathcal{H}(\alpha\,\mathbf{x}_1 + \beta\,\mathbf{x}_2) = \alpha\,\mathcal{H}\mathbf{x}_1 + \beta\,\mathcal{H}\mathbf{x}_2$$

- realistic requirement: sum of inputs leads to sum of outputs
- in practice systems are linear until they aren't (eg. volume too loud)

Linearity

Play

Nonlinearity



Play

Time invariance

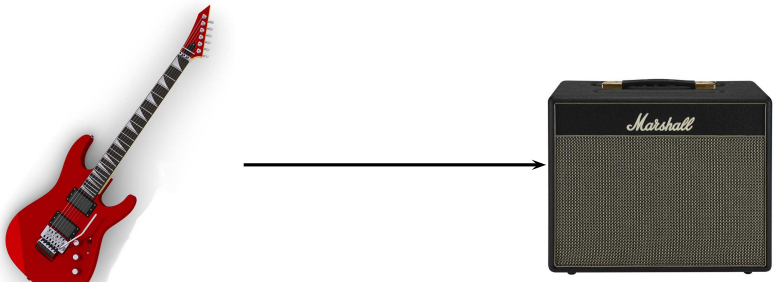
$$\mathcal{H}\mathcal{S}^k\mathbf{x} = \mathcal{S}^k\mathcal{H}\mathbf{x}$$

- more explicitly:

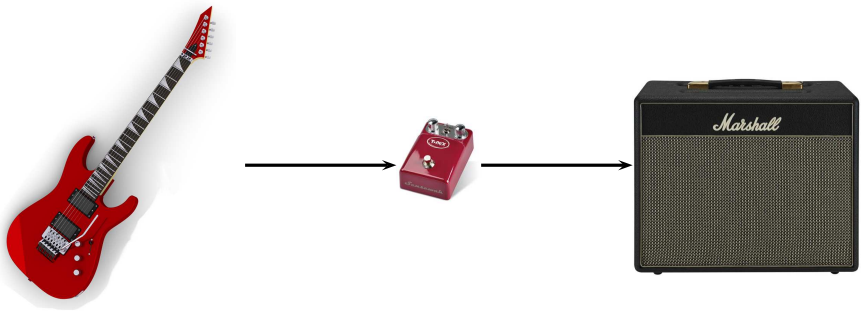
$$y_k[n] = y[n-k] \quad \text{with} \quad \begin{cases} \mathbf{x}_k = \mathcal{S}^{-k}\mathbf{x} & (x_k[n] = x[n-k]) \\ \mathbf{y} = \mathcal{H}\mathbf{x} \\ \mathbf{y}_k = \mathcal{H}\mathbf{x}_k \end{cases}$$

- realistic requirement: device should work the same way today and tomorrow
- analog systems exhibit "aging", digital systems don't

Time invariance

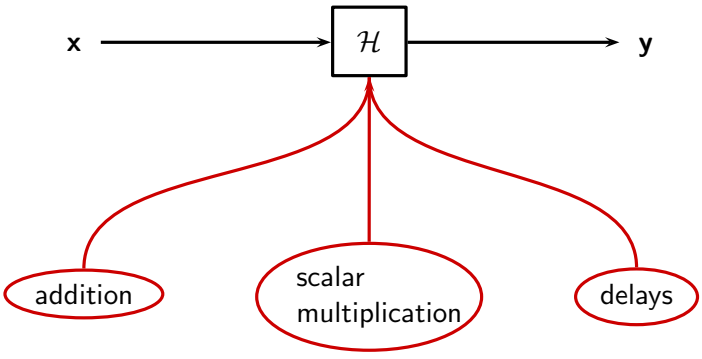


Time variance



Play

Linear, time-invariant systems



Linear, time-invariant systems

$$y[n] = H(x[n], x[n-1], x[n-2], \ldots, y[n-1], y[n-2], \ldots)$$

with $H(\cdot)$ a linear function of its arguments

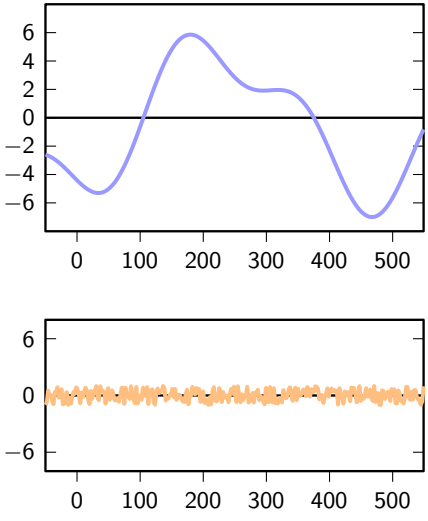
filtering by example

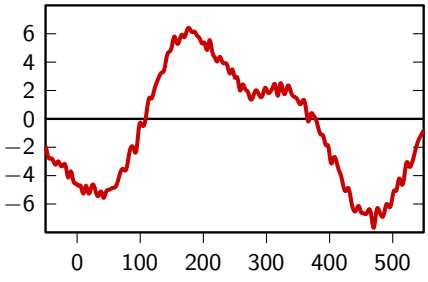
Filtering by example

Two fundamental filters:

- Moving average
- Leaky integrator

Typical filtering scenario: denoising





Denoising by averaging

Hypotheses:

- noisy signal samples are true values plus random noise value: $\mathbf{x} = \mathbf{x}_c + \boldsymbol{\eta}$
- noise values are random and with zero mean:

$$\frac{1}{M} \sum_{m=0}^{M-1} \eta[n-m] \approx 0 \quad \text{for } M \text{ large enough}$$

- the signal is varying slowly (current value is very similar to previous values):

$$x[n-m] \approx x[n] \quad \text{for } m \text{ reasonably small}$$

Denoising by averaging

Idea: to remove the noise, replace each sample by the average of M consecutive samples:

$$\begin{aligned}
\frac{1}{M}\sum_{m=0}^{M-1}\hat{x}[n-m] &= \frac{1}{M}\sum_{m=0}^{M-1}(x[n-m]+\eta[n-m]) \\
&= \frac{1}{M}\sum_{m=0}^{M-1}x[n-m] + \frac{1}{M}\sum_{m=0}^{M-1}\eta[n-m] \\
&\approx \frac{1}{M}\sum_{m=0}^{M-1}x[n] + 0 \\
&= x[n]
\end{aligned}$$

The Moving Average filter

$$y[n] = \frac{1}{M} \sum_{m=0}^{M-1} x[n-m]$$

- each output value averages current and previous $M-1$ input values
- average is recomputed at every step (hence "moving")
- computational requirements:
  - $M-1$ additions
  - one multiplication (by $1/M$)
  - $M-1$ memory cells (to remember the previous input values)

A simple implementation (plain Python)

```
class MA:
    def __init__(self, M):
        # pre-allocate storage for past input values
        self.buf = [0.0] * (M-1)
        self.norm = 1.0 / M

    def filt(self, x):
        # compute the local average
        y = x
        for v in self.buf:
            y += v
        y *= self.norm
        # update the buffer, eg: [a, b, c, d, e] -> [x, a, b, c, d]
        self.buf[1:] = self.buf[:-1]
        self.buf[0] = x
        return y
```

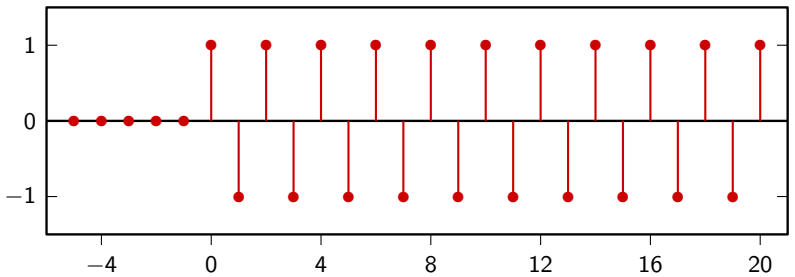
A better implementation using NumPy

```
import numpy as np

class MA:
    def __init__(self, M):
        self.buf = np.zeros(M)

    def filt(self, x):
        self.buf = np.roll(self.buf, 1)
        self.buf[0] = x
        return np.mean(self.buf)
```

Testing time!

let's test the algorithm on the sequence

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



Testing the implementation

```
> ma = MA(4)
> for n in range(20):
    print(ma.filt((-1) ** n), end=', ')

> 0.25, 0.0, 0.25, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
```

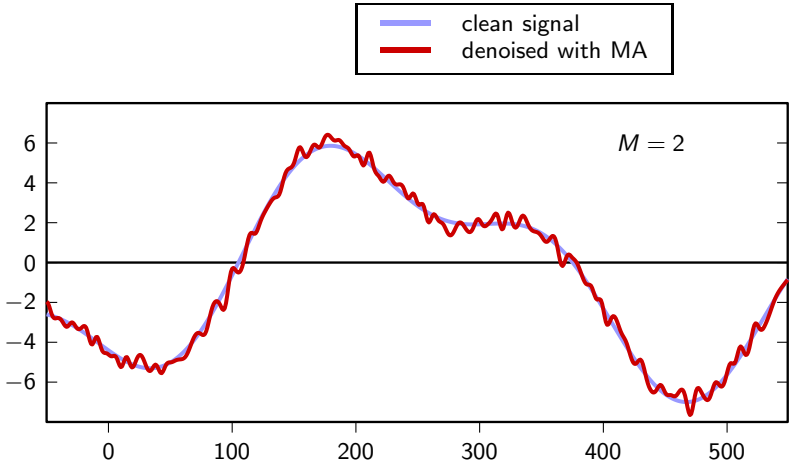
Question

can you guess what the output is going to be for $M = 5$?

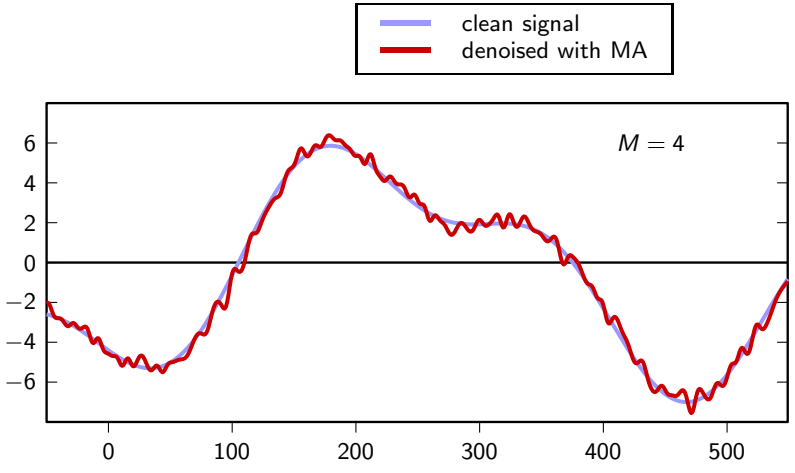
```
> ma = MA(5)
> for n in range(20):
    print(ma.filt((-1) ** n), end=', ')
```

```
> 0.2, 0.0, 0.2, 0.0, 0.2, -0.2, 0.2, -0.2, 0.2, -0.2, 0.2, -0.2, 0.2,
-0.2, 0.2, -0.2, 0.2, -0.2, 0.2, -0.2,
```

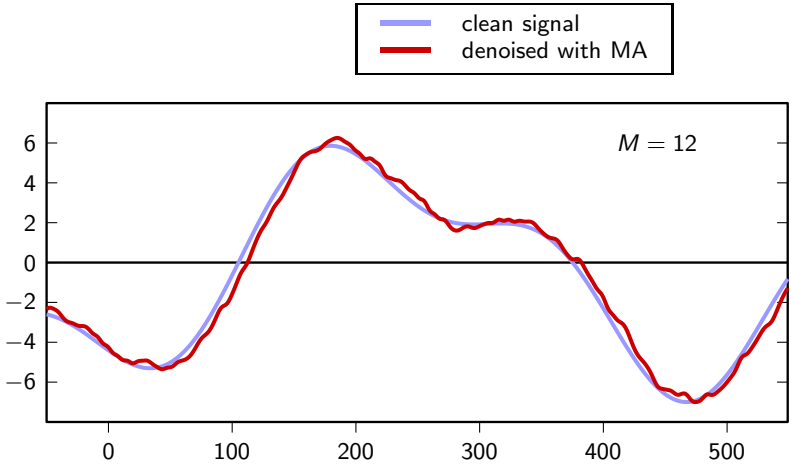
Does it work for denoising? Let's try it out!



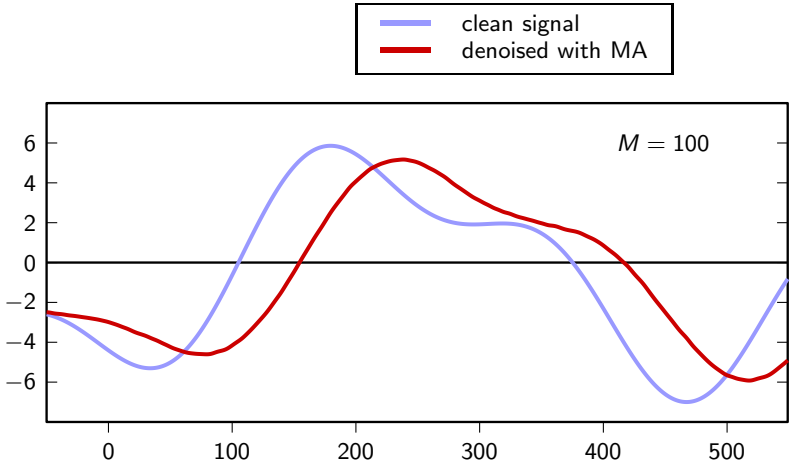
Does it work for denoising? Let's try it out!



Does it work for denoising? Let's try it out!



Does it work for denoising? Let's try it out!



Performance analysis

- smoothing effect proportional to M
- number of operations and storage also proportional to M
- there appears to be a “delay” between input and output

Idea: updating the average

Suppose we know the local average at time $n-1$;
can we compute the average at time n as an update of the previous value?

$$y[n] = f(y[n-1], x[n])$$

Updating the average

$$y[n] = \frac{1}{M} \sum_{m=0}^{M-1} x[n-m]$$

$$y[n] = \frac{1}{M} \Big[x[n] + x[n-1] + \ldots + x[n-(M-2)] + x[n-(M-1)] \Big]$$

$$y[n-1] = \frac{1}{M} \Big[x[n-1] + x[n-2] + \ldots + x[n-1-(M-2)] + x[n-1-(M-1)] \Big]$$

Updating the average

$$y[n-1] = \frac{1}{M}\Big[\qquad x[n-1] + x[n-2] + \ldots + x[n-M+1] + x[n-M]\Big]$$

$$y[n] = \frac{1}{M}\Big[x[n] + x[n-1] + x[n-2] + \ldots + x[n-M+1]\qquad\Big]$$

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

but this still requires M memory cells...

Updating the average

Idea: use the approximation $x[n-M] \approx y[n-1]$

- reasonable for "slow" signals, $x[n-m] \approx x[n]$ for m small
- equivalent to "forgetting" $(1/M)$-th of the previous local average

$$y[n-1] - (1/M)x[n-M] \approx (1-1/M)y[n-1]$$

- "forgetting factor" $\lambda = 1 - 1/M$, close to one for M large

$$\begin{aligned} y[n] &= y[n-1] - \frac{1}{M}x[n-M] + \frac{1}{M}x[n] \\ &\approx \lambda y[n-1] + (1-\lambda)x[n] \end{aligned}$$

The Leaky Integrator

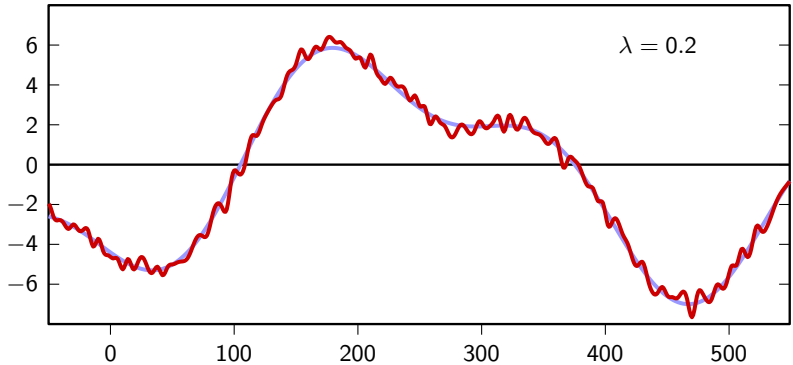
$$y[n] = \lambda y[n-1] + (1-\lambda)x[n]$$

- each output value is an update of the previous output
- the algorithm is *recursive*
- computational requirements:
  - one addition
  - two multiplications
  - one memory cell to remember the previous *output*

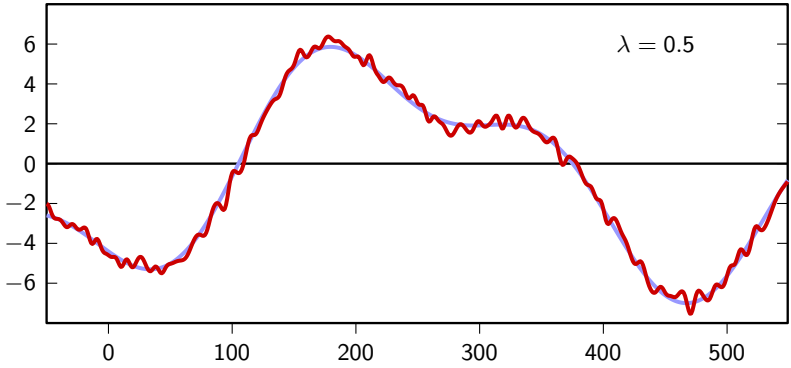
A simple implementation

```
class LI:
    def __init__(self, lam):
        self.buf = 0
        self.lam = lam

    def filt(self, x):
        self.buf = self.lam * self.buf + (1 - self.lam) * x
        return self.buf
```

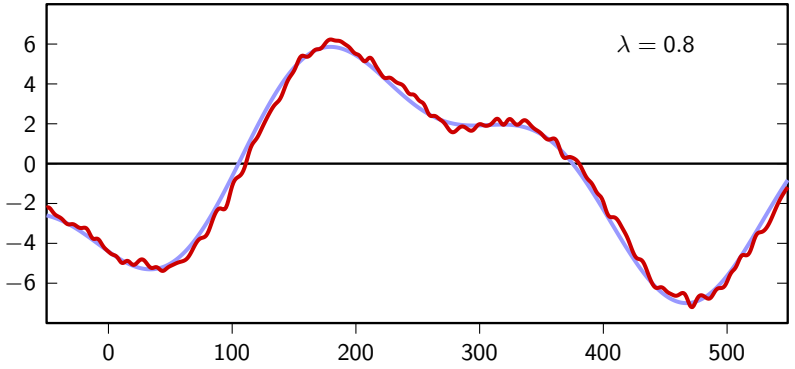
Does it work? Let's try it out!



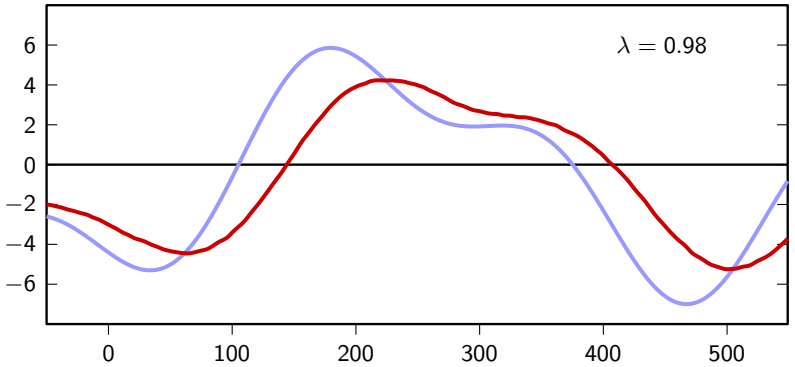
Does it work? Let's try it out!



Does it work? Let's try it out!



Does it work? Let's try it out!



Performance analysis

- smoothing effect dependent on λ
- number of operations and storage *independent of* λ
- there also appears to be a processing "delay"

Leaky Integrator: why the name

A discrete-time "integrator" just accumulates input values:

$$y[n] = \sum_{k=-\infty}^{n} x[k]$$

We can rewrite the integrator recursively as

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

Leaky Integrator: why the name

To prevent "explosions" pick $\lambda < 1$

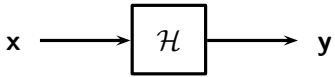
$$y[n] = \lambda y[n-1] + (1-\lambda)x[n]$$

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

the impulse response

Linear, time-invariant systems



$$\mathcal{H}(\alpha\,\mathbf{x}_1 + \beta\,\mathbf{x}_2) = \alpha\,\mathcal{H}\mathbf{x}_1 + \beta\,\mathcal{H}\mathbf{x}_2$$

$$\mathcal{H}\mathcal{S}^k\mathbf{x} = \mathcal{S}^k\mathcal{H}\mathbf{x}$$

The situation so far

- we "designed" two filters *algorithmically*
- both algorithms are of the form

$$y[n] = \sum_{k=1}^{N-1} a_k y[n-k] + \sum_{k=0}^{M-1} b_k x[n-k]$$

  - moving average: $N = 1,\ b_k = 1/M$
  - leaky integrator: $M = 1, N = 2,\ a_1 = \lambda,\ b_0 = 1 - \lambda$

- we were able to find the coefficients intuitively...
- ... but we can't use intuition for more complicated filters

The path to filter design

- LTI systems are fully described by their *impulse response*
- once we know the impulse response we can implement the filter via *convolution*
- the DTFT of the impulse response describes how a filter works in the frequency domain
- filter design starts from the frequency domain and finds the coefficients of the algorithm

Impulse response

$$\mathbf{h} = \mathcal{H}\boldsymbol{\delta}$$

Fundamental result: impulse response fully characterizes the LTI system!

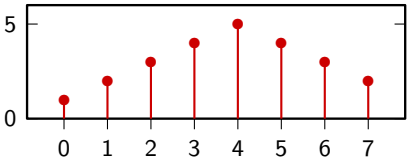
Every signal is a linear combination of atomic time elements

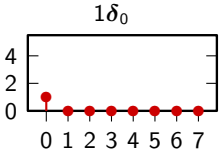
$$\mathbf{x} = \sum_{k=-\infty}^{\infty} x[k]\,\boldsymbol{\delta}_k,$$

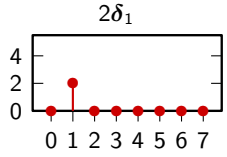
$$\boldsymbol{\delta}_k = \mathcal{S}^{-k}\boldsymbol{\delta}$$

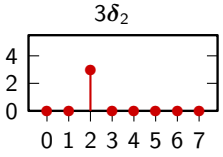
$$\delta_k[n] = \delta[n-k] = \begin{cases} 1 & n = k \\ 0 & n \neq k. \end{cases}$$

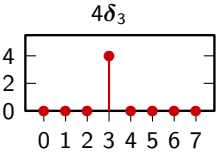
Time domain: sum of time pulses

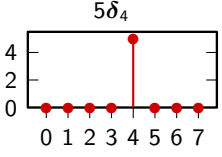


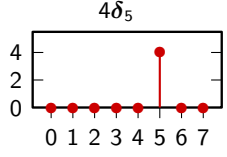


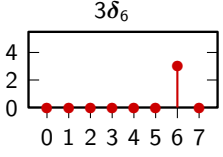


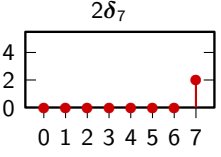












Filter's output from impulse response

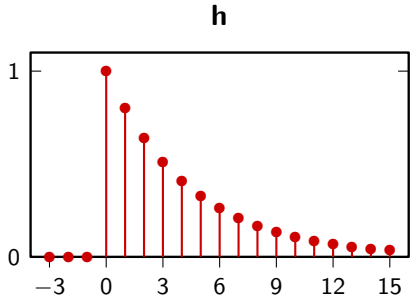
$$\mathcal{H}\mathbf{x} = \mathcal{H}\left(\sum_{k=-\infty}^{\infty} x[k]\,\mathcal{S}^{-k}\boldsymbol{\delta}\right) \qquad \textit{using linearity...}$$

$$= \sum_{k=-\infty}^{\infty} \mathcal{H}(x[k]\,\mathcal{S}^{-k}\boldsymbol{\delta}) \qquad \textit{using linearity...}$$

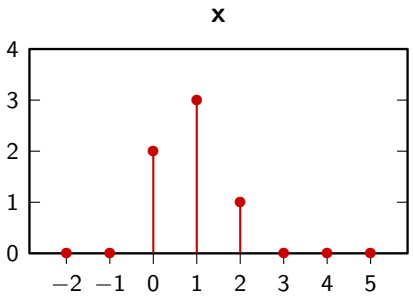
$$= \sum_{k=-\infty}^{\infty} x[k]\,\mathcal{S}^{-k}\mathcal{H}\boldsymbol{\delta} \qquad \textit{using time invariance...}$$

$$= \sum_{k=-\infty}^{\infty} x[k]\,\mathcal{S}^{-k}\mathbf{h}$$

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

- $\mathbf{x} = 2\boldsymbol{\delta} + 3\boldsymbol{\delta}_1 + \boldsymbol{\delta}_2$
- we know the impulse response $\mathbf{h} = \mathcal{H}\boldsymbol{\delta}$;
- compute $\mathbf{y} = \mathcal{H}\mathbf{x}$ exploiting linearity and time-invariance

Example

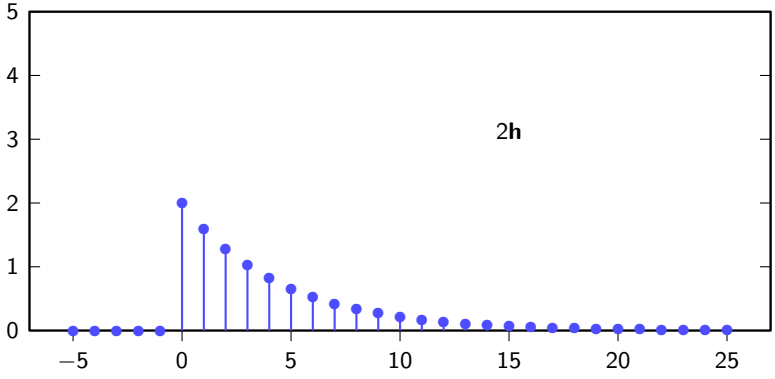
$$\mathbf{x} = \mathcal{H}\{2\boldsymbol{\delta} + 3\mathcal{S}^{-1}\boldsymbol{\delta} + \mathcal{S}^{-2}\boldsymbol{\delta}\}$$

$$= 2\mathcal{H}\boldsymbol{\delta} + 3\mathcal{S}^{-1}\mathcal{H}\boldsymbol{\delta} + \mathcal{S}^{-2}\mathcal{H}\boldsymbol{\delta}$$

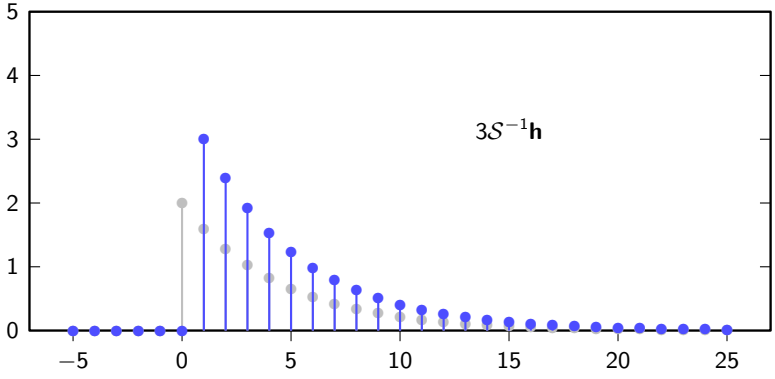
$$= 2\mathbf{h} + 3\mathcal{S}^{-1}\mathbf{h} + \mathcal{S}^{-2}\mathbf{h}$$

$$x[n] = 2h[n] + 3h[n-1] + h[n-2]$$

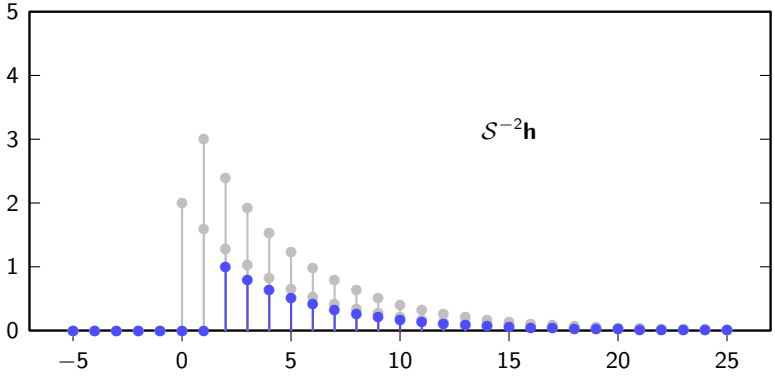
Example



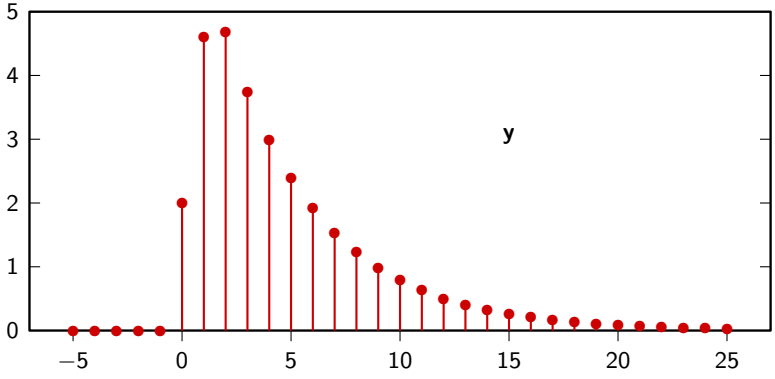
Example



Example



Example



Convolution

$$\begin{aligned}\mathbf{y} &= \mathcal{H}\mathbf{x} \\ &= \sum_{k=-\infty}^{\infty} x[k]\,\mathcal{S}^{-k}\mathbf{h} \\ &= \mathbf{x} * \mathbf{h}\end{aligned}$$

algorithmic expression for the individual output sample

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

Convolution algorithm

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

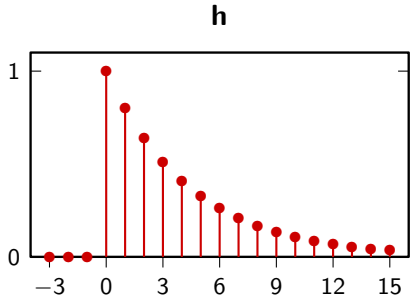
Ingredients:

- a sequence **x**
- a second sequence **h**

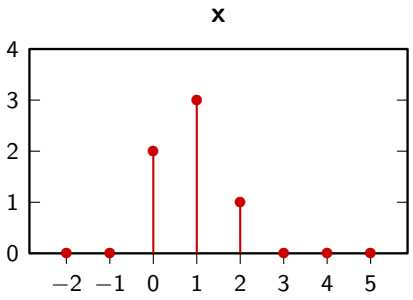
The recipe:

- time-reverse **h**
- at each step n (from $-\infty$ to ∞):
  - center the time-reversed **h** in n (i.e. delay by n)
  - multiply it by **x** element-wise
  - sum all the products

Same example, different perspective

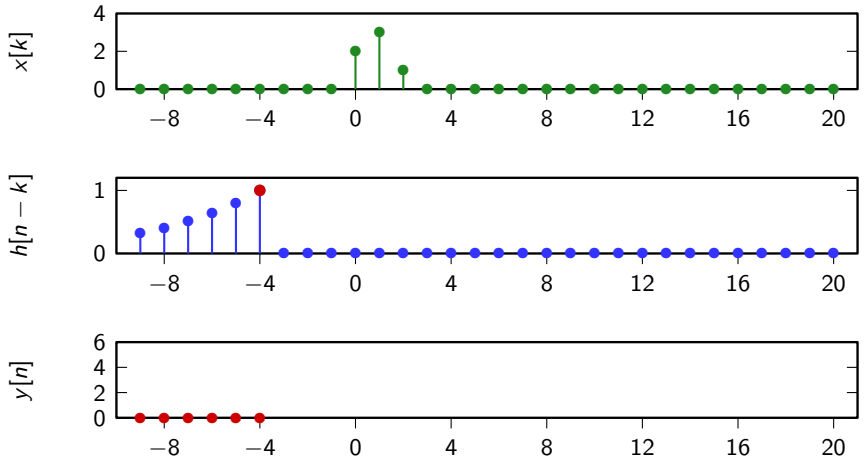


$$h[n] = \alpha^n \, u[n]$$

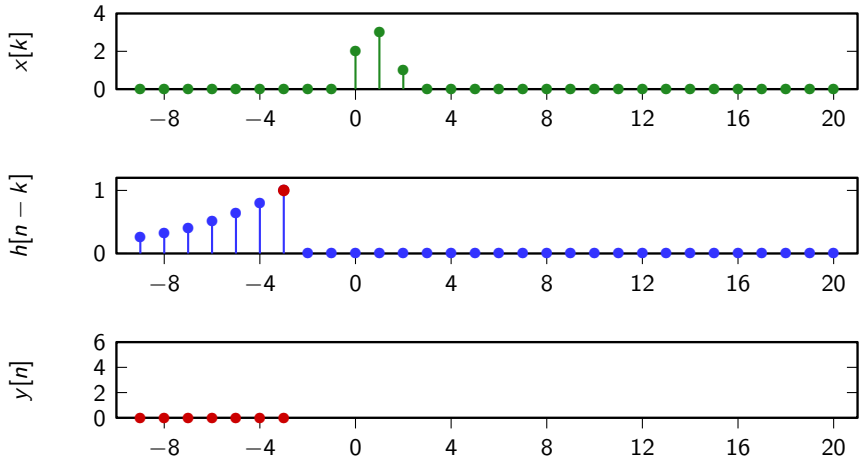


$$x[n] = \begin{cases} 2 & n = 0 \\ 3 & n = 1 \\ 1 & n = 2 \\ 0 & \text{otherwise} \end{cases}$$

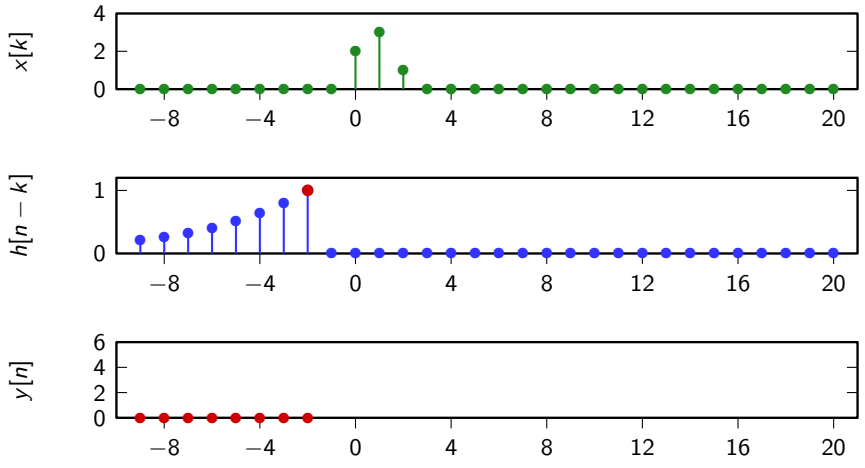
Convolution example



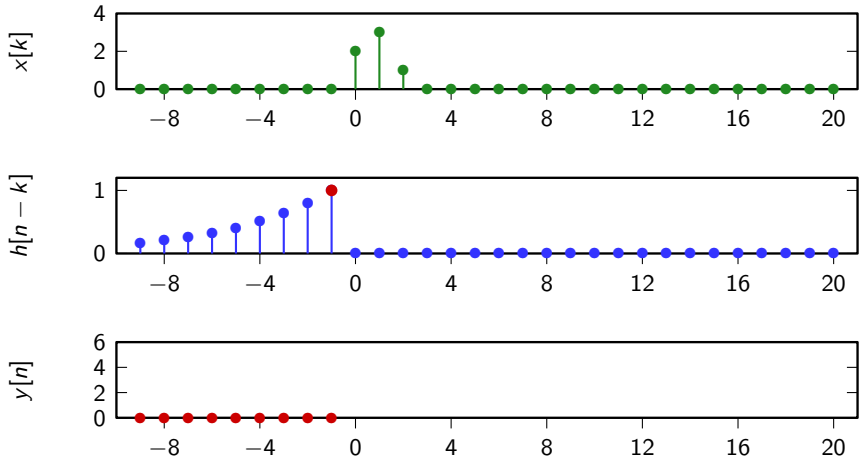
Convolution example



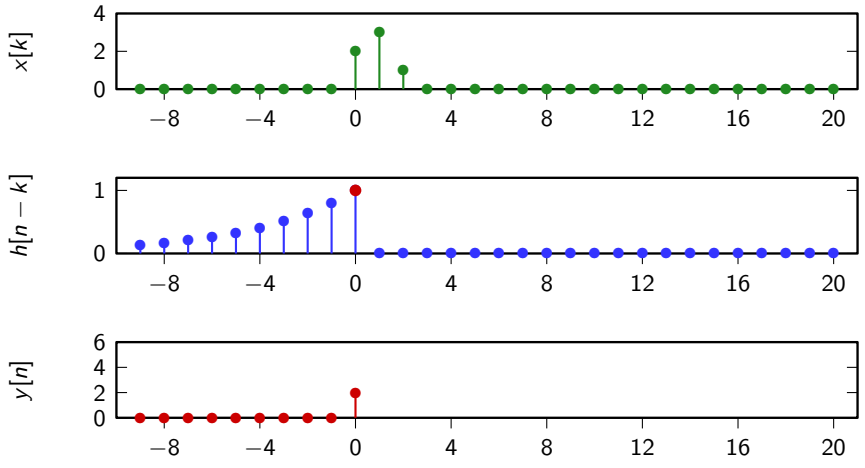
Convolution example



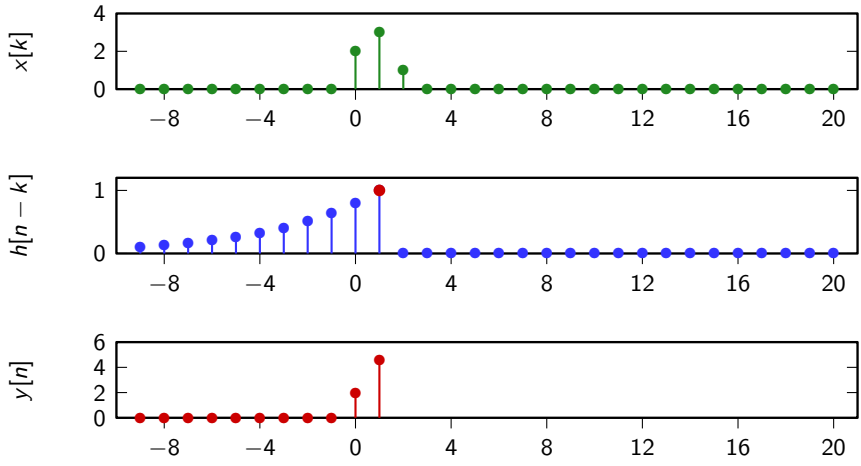
Convolution example



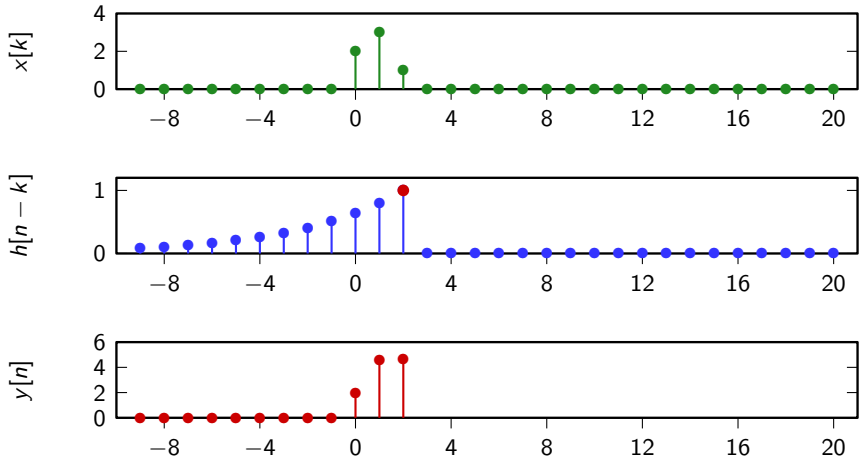
Convolution example



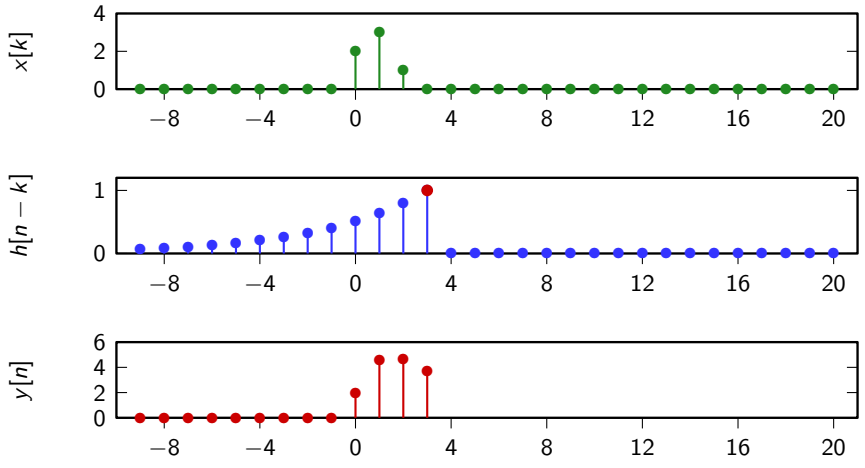
Convolution example



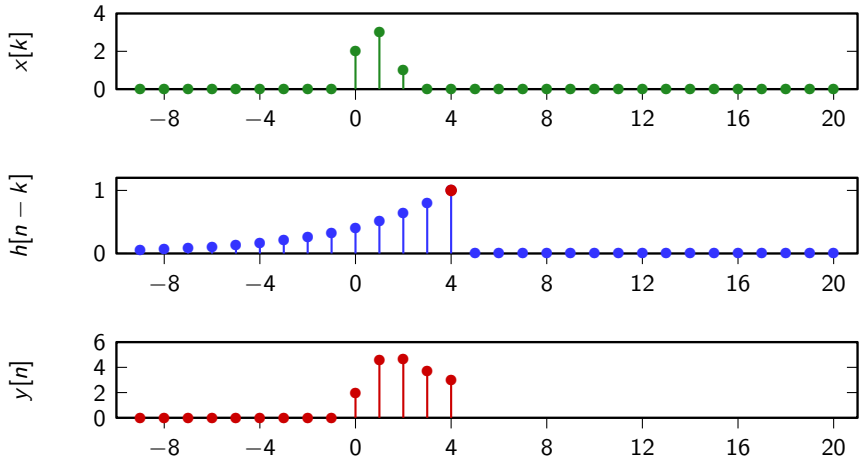
Convolution example



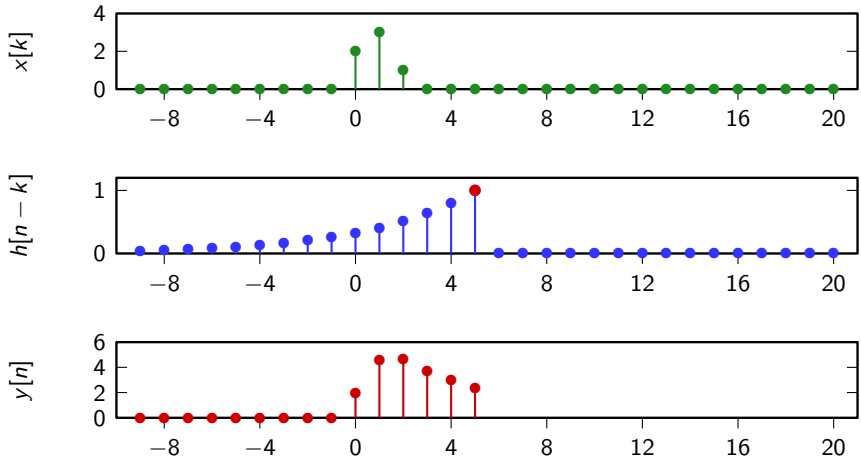
Convolution example



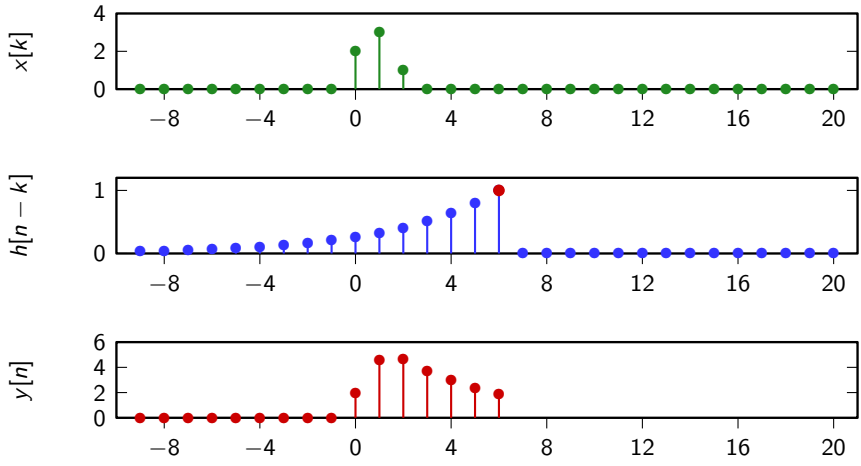
Convolution example



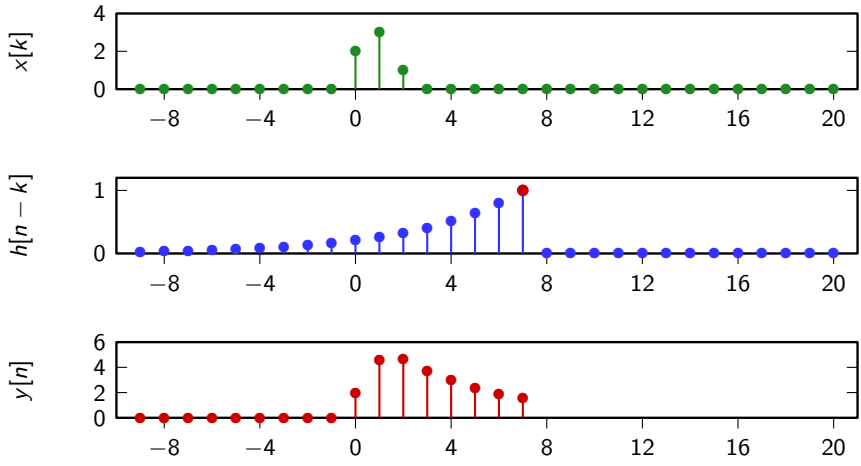
Convolution example



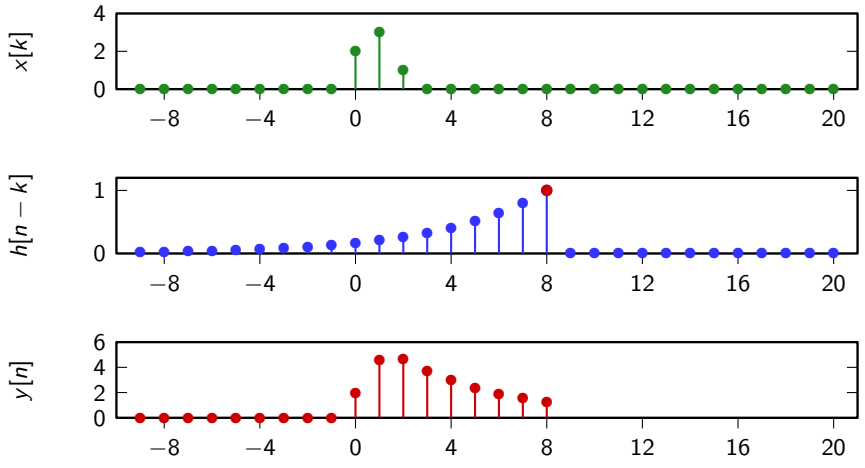
Convolution example



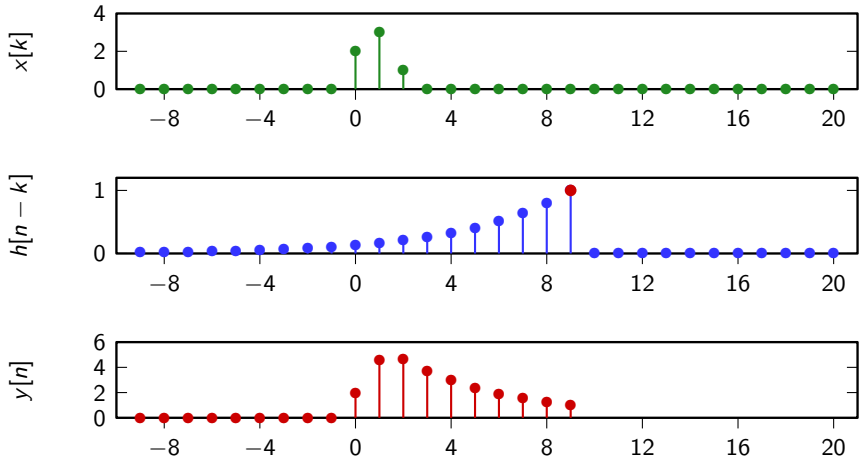
Convolution example



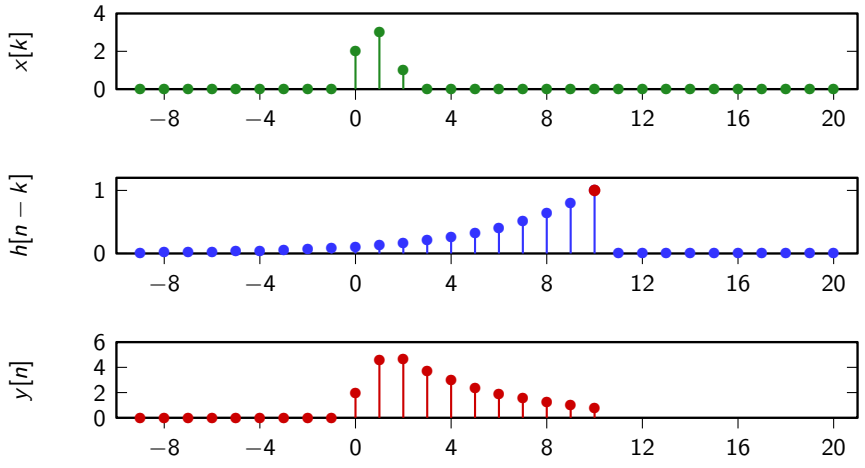
Convolution example



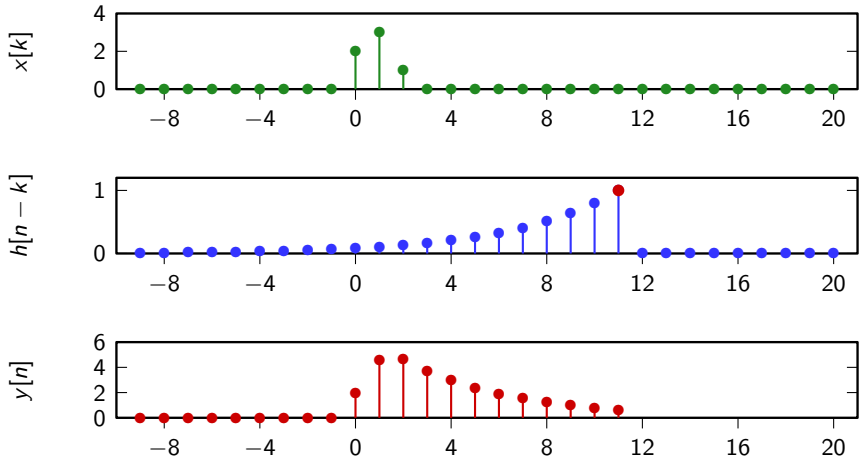
Convolution example



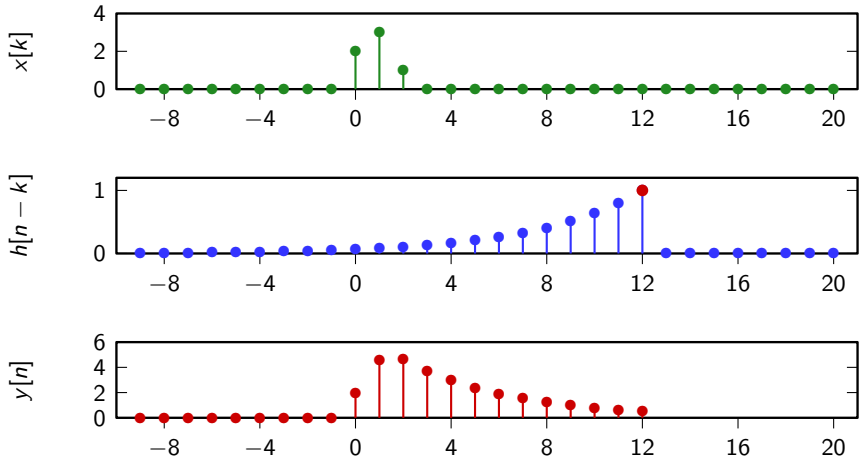
Convolution example



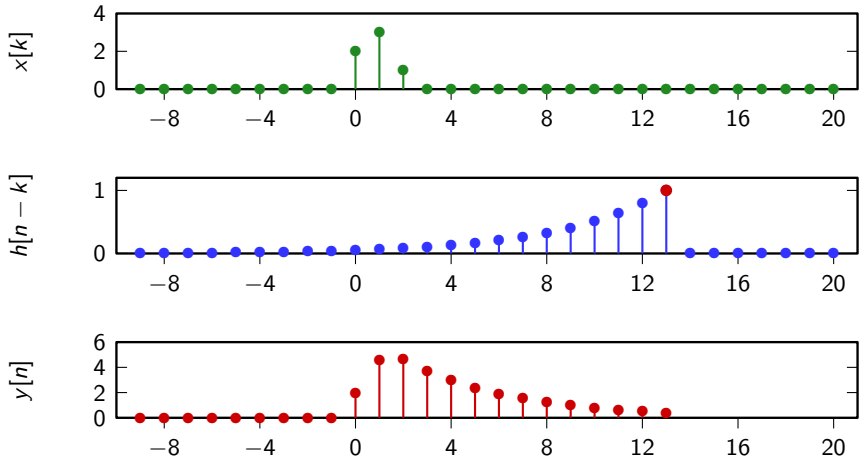
Convolution example



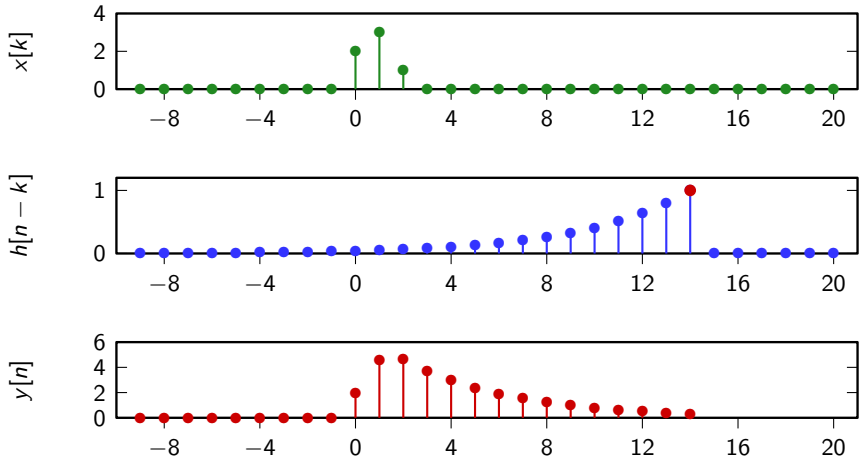
Convolution example



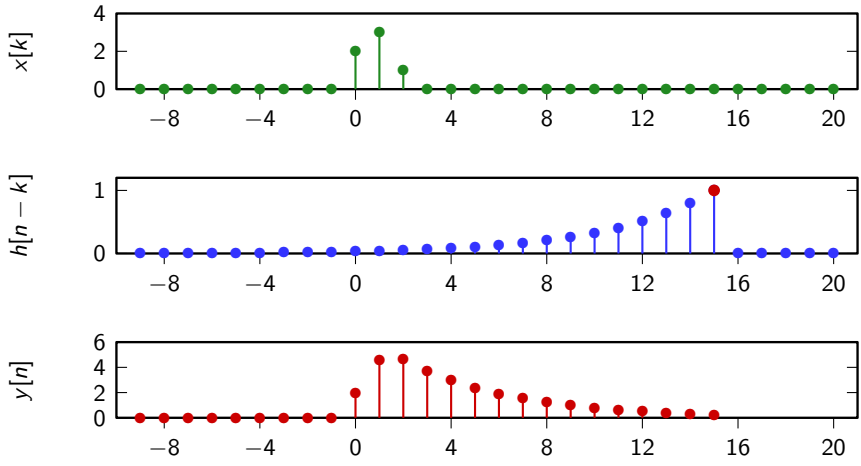
Convolution example



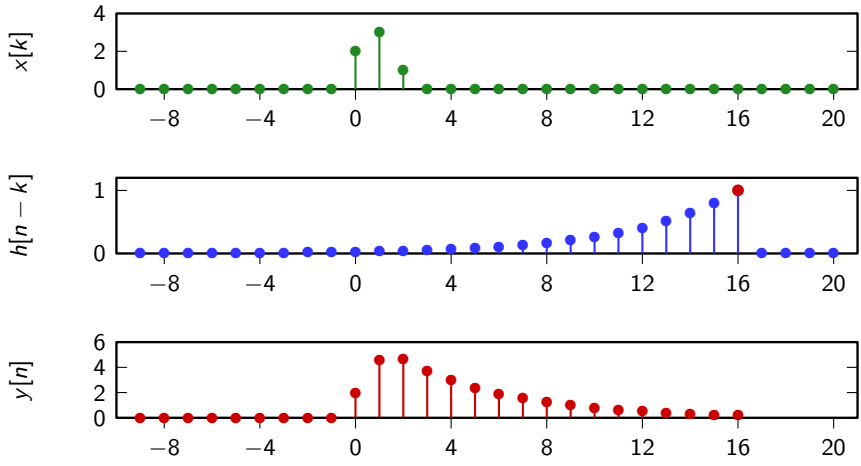
Convolution example



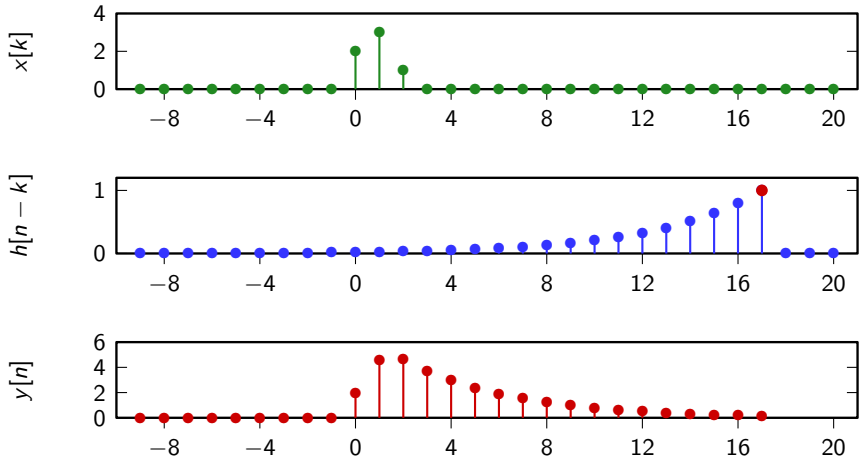
Convolution example



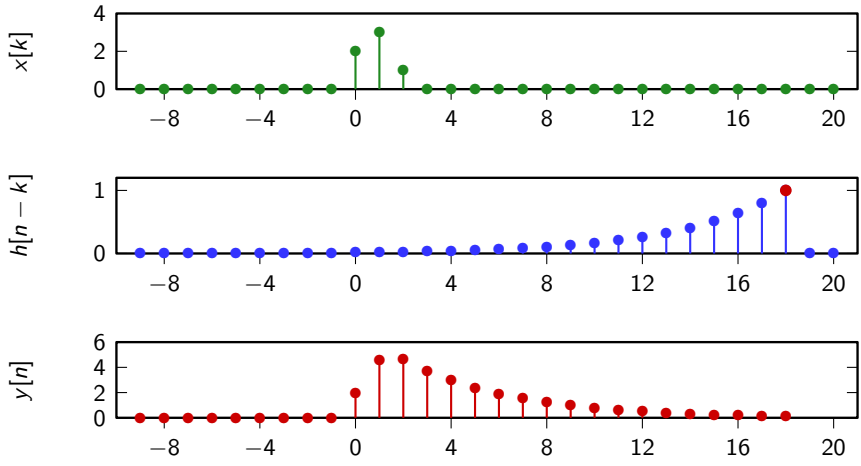
Convolution example



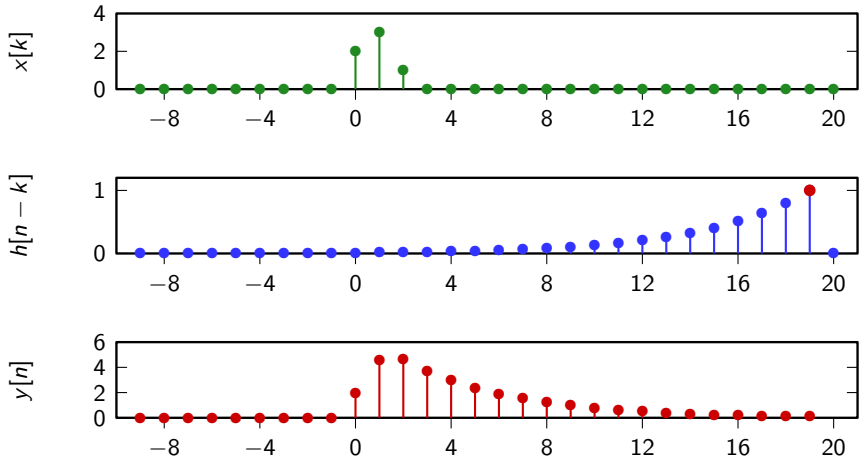
Convolution example



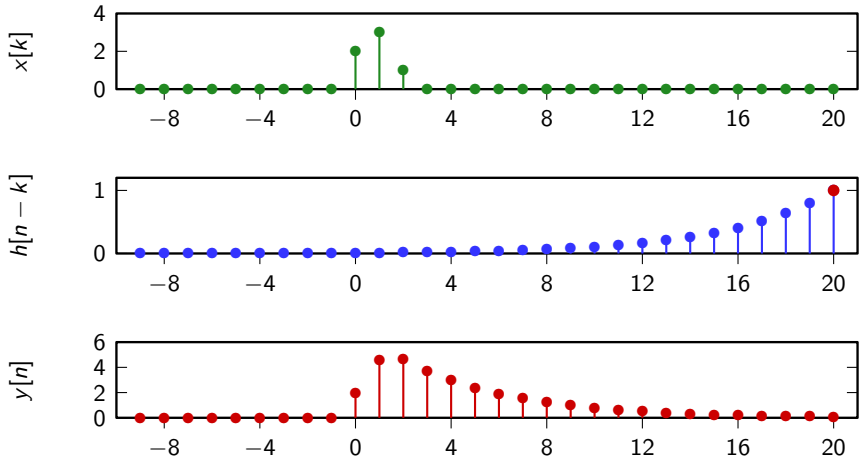
Convolution example



Convolution example

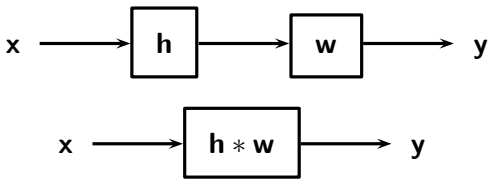


Convolution example



Convolution properties

- linearity and time invariance (by definition)
- commutativity: $\mathbf{x} * \mathbf{h} = \mathbf{h} * \mathbf{x}$
- associativity for absolutely- and square-summable sequences: $(\mathbf{x} * \mathbf{h}) * \mathbf{w} = \mathbf{x} * (\mathbf{h} * \mathbf{w})$



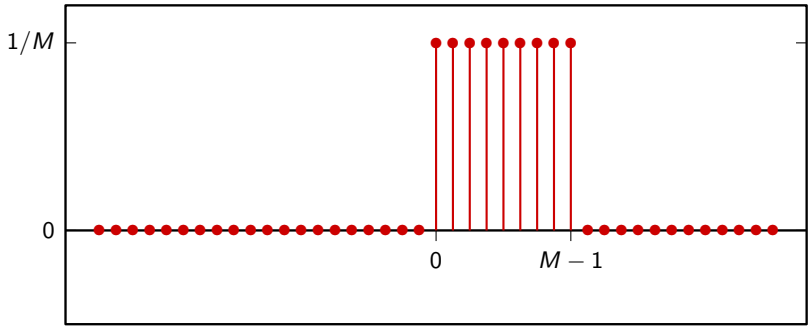
Moving Average: impulse response

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$

$$h[n] = \frac{1}{M} \sum_{k=0}^{M-1} \delta[n-k] \qquad x[n] \leftarrow \delta[n]$$

$$= \begin{cases} 1/M & \text{for } 0 \leq n < M \\ 0 & \text{otherwise} \end{cases}$$

Moving Average: impulse response



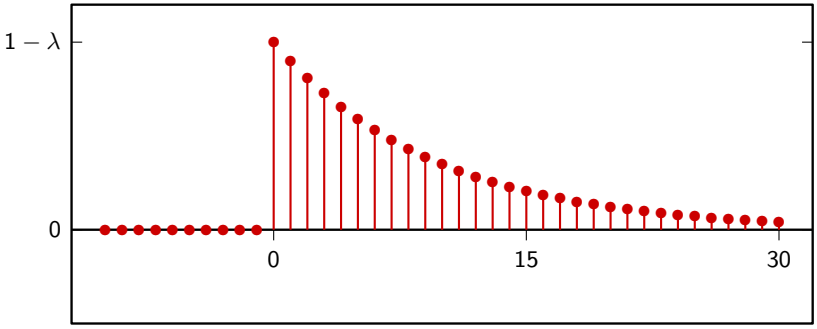
Leaky integrator: impulse response

$$y[n] = \lambda y[n-1] + (1-\lambda)\delta[n]$$

- $y[n] = 0$ for all $n < 0$
- $y[0] = \lambda y[-1] + (1-\lambda)\delta[0] = (1-\lambda)$
- $y[1] = \lambda y[0] + (1-\lambda)\delta[1] = \lambda(1-\lambda)$
- $y[2] = \lambda y[1] + (1-\lambda)\delta[2] = \lambda^2(1-\lambda)$
- $y[3] = \lambda y[2] + (1-\lambda)\delta[3] = \lambda^3(1-\lambda)$
- ...

Leaky integrator: impulse response

$$h[n] = (1 - \lambda)\lambda^n \, u[n]$$



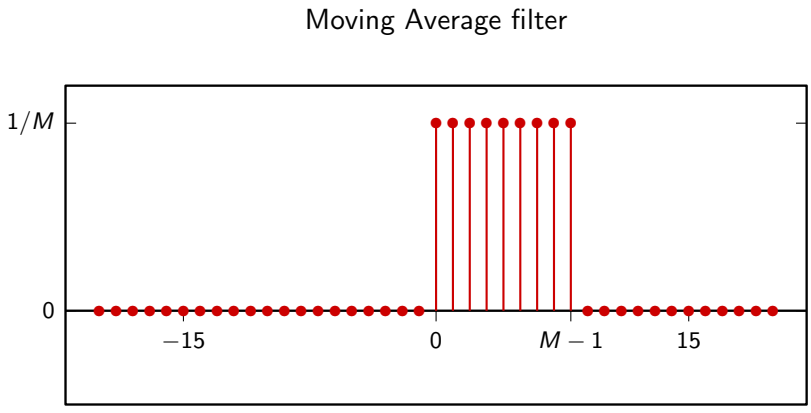
Filter types according to impulse response

- 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

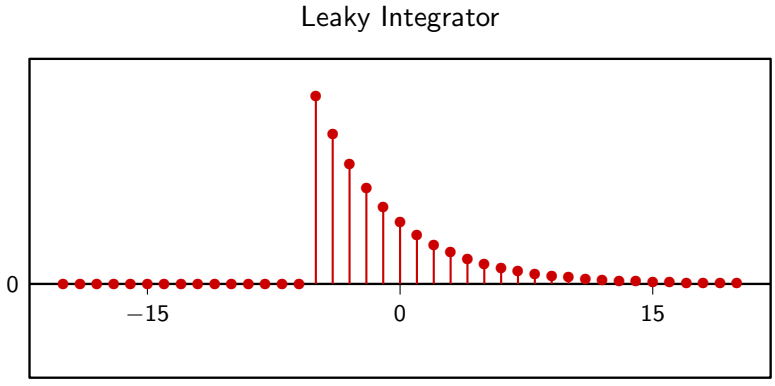
FIR (example)



IIR

- 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)

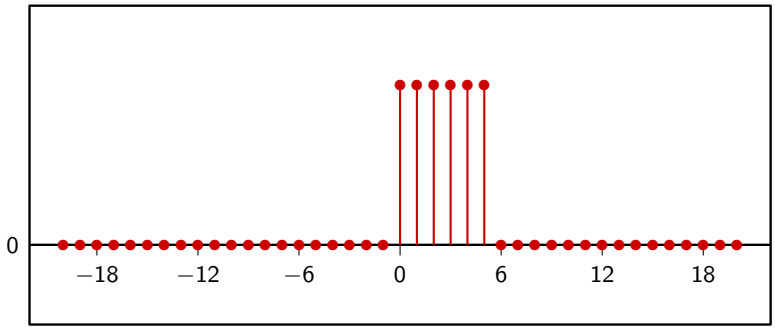


Causal vs Noncausal

- 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 a offline fashion (when all input data is available on storage, e.g. in Image Processing)

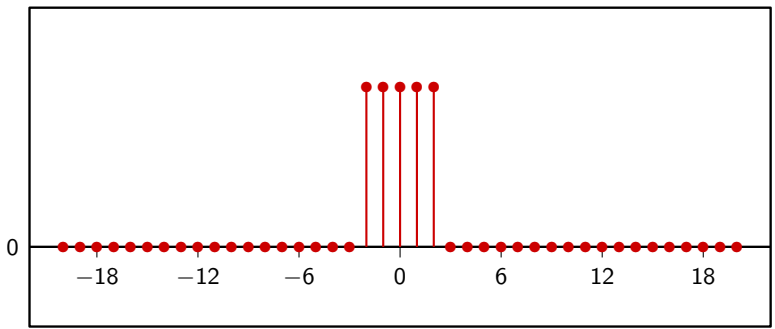
Causal example

Moving Average filter

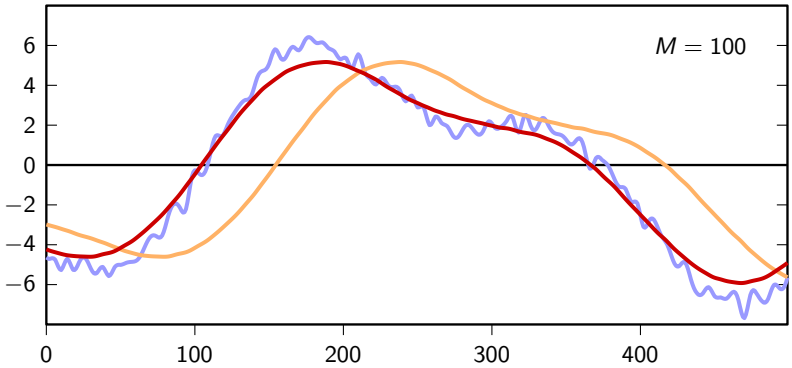


Noncausal example

Zero-centered Moving Average filter



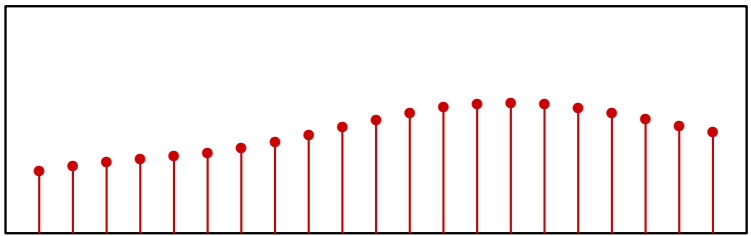
Causal and Noncausal Moving Average



Processing delay: intuition

Moving Average:

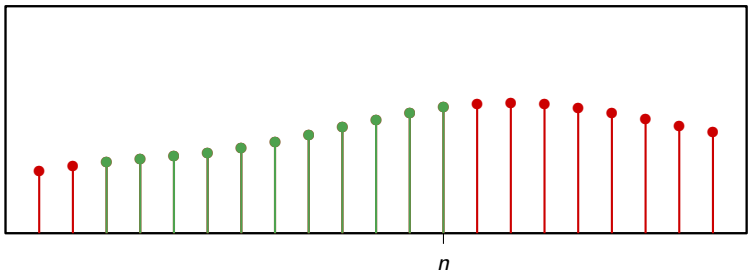
$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$



Processing delay: intuition

Moving Average:

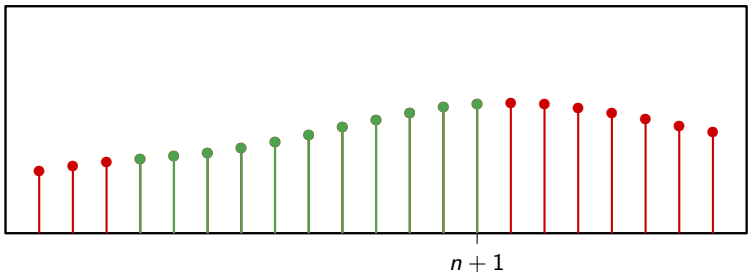
$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$



Processing delay: intuition

Moving Average:

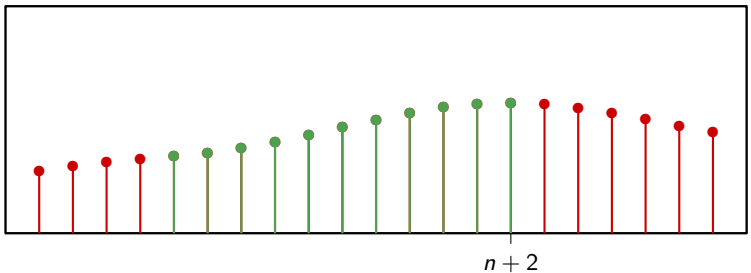
$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$



Processing delay: intuition

Moving Average:

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$



Processing delay: intuition

We assumed the signal is smooth; if it is linear, i.e. $x[n] = an$,

$$
\begin{aligned}
y[n] &= \frac{1}{M} \sum_{k=0}^{M-1} x[n-k] \\
&= \frac{1}{M} \sum_{k=0}^{M-1} a(n-k) \\
&= \frac{a}{M} \left[\sum_{k=0}^{M-1} n - \sum_{k=0}^{M-1} k \right] \\
&= \frac{a}{M} \left[Mn - M(M-1)/2 \right] \\
&= a(n - (M-1)/2) \\
&= x[n - (M-1)/2] \qquad \leftarrow \textit{delay of } (M-1)/2 \textit{ samples (assume } M \textit{ odd)}
\end{aligned}
$$

Processing delay: intuition

What about a slow sinusoid of the form $x[n] = \cos(\omega_0 n)$?

$$
\begin{aligned}
y[n] &= \frac{1}{M} \sum_{k=0}^{M-1} \cos(\omega_0 (n-k)) \\
&= \frac{1}{M} \operatorname{Re}\left\{ \sum_{k=0}^{M-1} e^{j\omega_0 (n-k)} \right\} = \frac{1}{M} \operatorname{Re}\left\{ e^{j\omega_0 n} \sum_{k=0}^{M-1} e^{-j\omega_0 k} \right\} \\
&= \frac{1}{M} \operatorname{Re}\left\{ e^{j\omega_0 n} \frac{1 - e^{-j\omega_0 M}}{1 - e^{-j\omega_0}} \right\} = \frac{1}{M} \operatorname{Re}\left\{ e^{j\omega_0 n} \frac{e^{-j\omega_0 M/2} \left[e^{j\omega_0 M/2} - e^{-j\omega_0 M/2} \right]}{e^{-j\omega_0/2} \left[e^{j\omega_0/2} - e^{-j\omega_0/2} \right]} \right\} \\
&= \frac{\sin\left(\frac{\omega_0}{2} M\right)}{M \sin\left(\frac{\omega_0}{2}\right)} \operatorname{Re}\left\{ e^{j\omega_0 n} e^{-j\omega_0 (M-1)/2} \right\} \\
&= c(\omega_0, N) \cos\left(\omega_0 \left(n - \frac{M-1}{2} \right) \right) \propto x[n - (M-1)/2]
\end{aligned}
$$

Processing delay

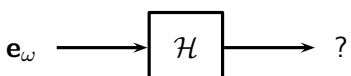
- all causal filters introuduce a processing delay
- processing delay is best understood in the frequency domain

the frequency response

Filtering a complex-valued oscillation

$\mathbf{e}_\omega$: complex exponential sequence at frequency ω

$$e_\omega[n] = e^{j\omega n}$$



A remarkable result

$$\mathbf{e}_\omega * \mathbf{h} = \mathbf{h} * \mathbf{e}_\omega$$

convolution is commutative

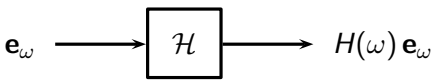
$$= \sum_{k=-\infty}^{\infty} h[k]\mathcal{S}^{-k}\mathbf{e}_\omega$$

$\mathcal{S}^{-k}\mathbf{e}_\omega = e^{-j\omega k}\mathbf{e}_\omega$

$$= \mathbf{e}_\omega \sum_{k=-\infty}^{\infty} h[k]e^{-j\omega k}$$

$$= H(\omega)\,\mathbf{e}_\omega$$

A remarkable result



- complex exponentials are *eigensequences* of LTI systems: $\mathcal{H}\mathbf{e}_\omega = C\mathbf{e}_\omega$
- scalar factor C is DTFT of impulse response at input frequency
- LTI systems cannot change the frequency of sinusoidal inputs

Magnitude and phase

If $H(\omega) = Ae^{j\theta}$, then

$$(\mathcal{H}\mathbf{e}_\omega)[n] = Ae^{j(\omega n+\theta)}$$

amplitude:
- amplification ($A > 1$)
- attenuation ($0 \leq A < 1$)

phase shift:
- delay ($\theta < 0$)
- advancement ($\theta > 0$)

Filters in the frequency domain

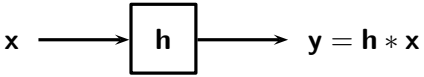
how do filters change the spectrum of an input signal?

$$\text{DTFT}\{\mathbf{x} * \mathbf{h}\} = ?$$

Intuition:

- we know how filters modify a sinusoid
- signals can be expressed as a linear combination of sinusoids
- by linearity...

The convolution theorem



$$\mathbf{x} \overset{\text{DTFT}}{\longleftrightarrow} \mathbf{X}$$

$$\mathbf{h} \overset{\text{DTFT}}{\longleftrightarrow} \mathbf{H}$$

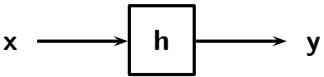
$$\mathbf{y} \overset{\text{DTFT}}{\longleftrightarrow} \mathbf{Y}$$

The convolution theorem

$$
\begin{aligned}
Y(\omega) &= \sum_{n=-\infty}^{\infty} y[n] e^{-j\omega n} \\
&= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k] h[n-k] e^{-j\omega n} \\
&= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k] h[n-k] e^{-j\omega(n-k)} e^{-j\omega k} \\
&= \sum_{k=-\infty}^{\infty} x[k] e^{-j\omega k} \sum_{n=-\infty}^{\infty} h[n-k] e^{-j\omega(n-k)} \\
&= H(\omega) X(\omega)
\end{aligned}
$$

$$
y[n] = \sum_{k=-\infty}^{\infty} x[k] h[n-k]
$$

The convolution theorem



$$\mathbf{y} = \mathbf{h} * \mathbf{x}$$

$$\mathbf{Y} = \mathbf{H}\mathbf{X}$$

Frequency response

$$\mathbf{y} = \mathbf{h} * \mathbf{x}$$

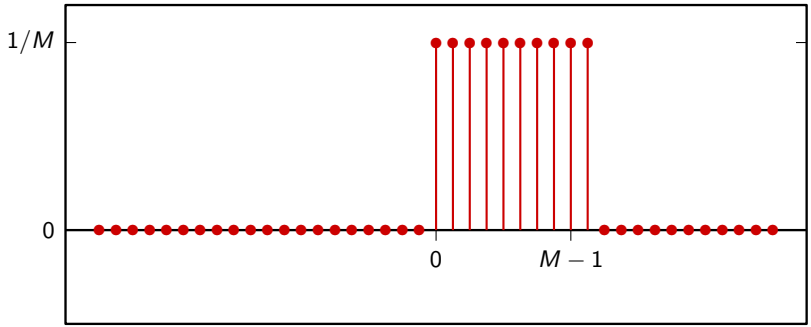
$$\mathbf{Y} = \mathbf{H}\,\mathbf{X}$$

Two effects on the spectrum of the input signal:

- magnitude: amplification ($|H(\omega)| > 1$) or attenuation ($|H(\omega)| < 1$)
- phase: overall delay and "shape" change

Moving Average: frequency response

$$h[n] = (u[n] - u[n - M])/M$$

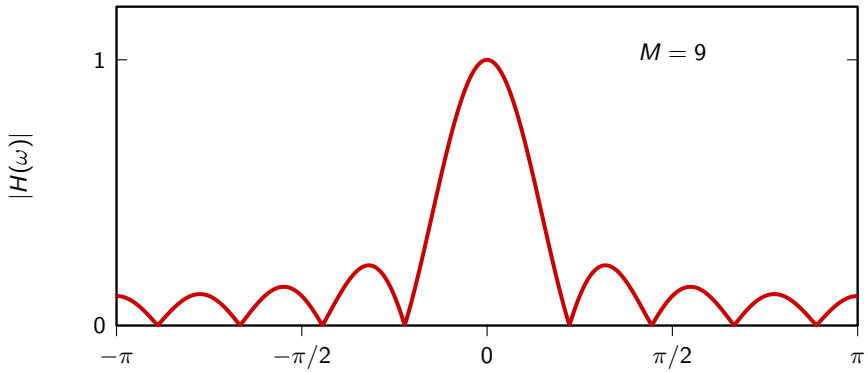


Moving Average: frequency response

$$
\begin{aligned}
H(\omega) &= \sum_{n=0}^{M-1} \frac{1}{M} e^{-j\omega n} \\
&= \frac{1}{M} \frac{1 - e^{-j\omega M}}{1 - e^{-j\omega}} \\
&= \frac{1}{M} \frac{e^{-j\frac{\omega M}{2}} \left[e^{j\frac{\omega M}{2}} - e^{-j\frac{\omega M}{2}} \right]}{e^{-j\frac{\omega}{2}} \left[e^{j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}} \right]} \\
&= \frac{1}{M} \frac{\sin\left(\frac{\omega}{2} M\right)}{\sin\left(\frac{\omega}{2}\right)} e^{-j\frac{\omega}{2}(M-1)}
\end{aligned}
$$

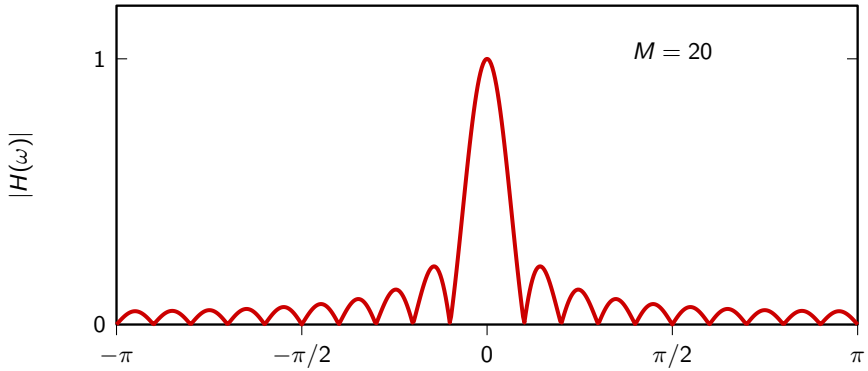
Moving Average: magnitude response

$$|H(\omega)| = \frac{1}{M} \left| \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} \right|$$



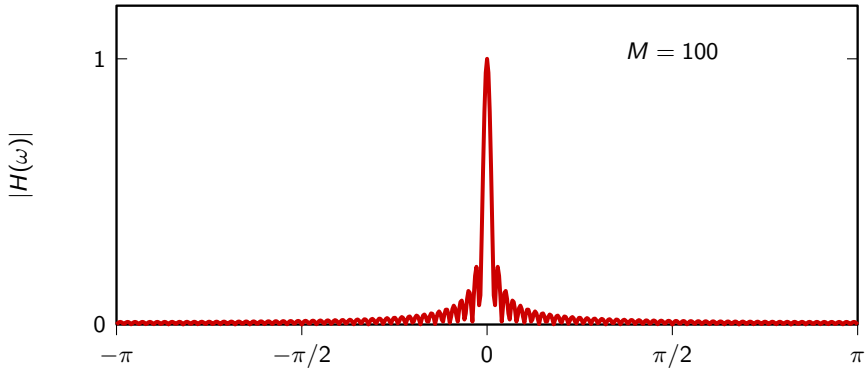
Moving Average: magnitude response

$$|H(\omega)| = \frac{1}{M} \left| \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} \right|$$

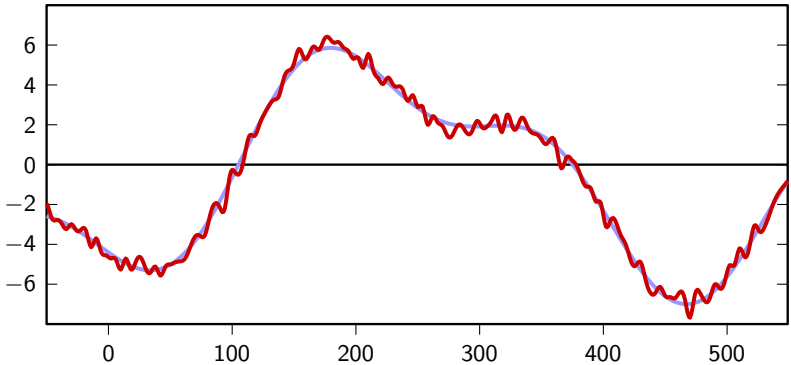


Moving Average: magnitude response

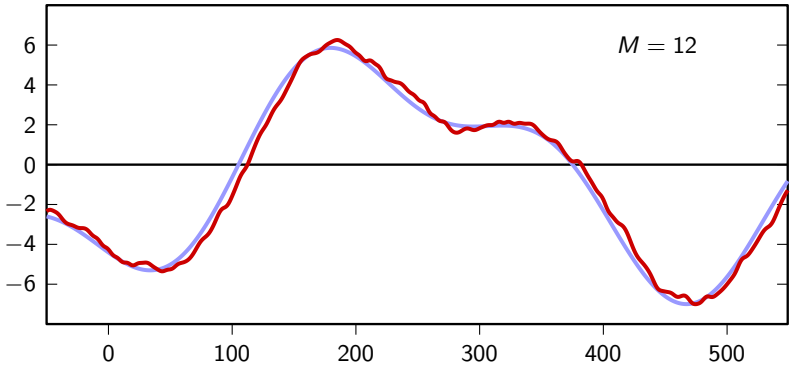
$$|H(\omega)| = \frac{1}{M} \left| \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} \right|$$



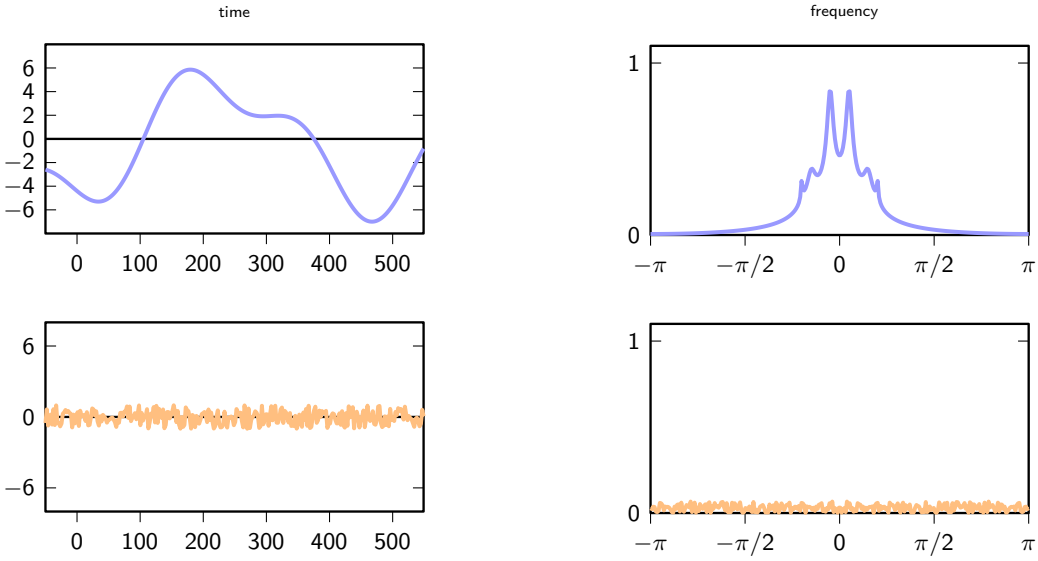
Denoising revisited



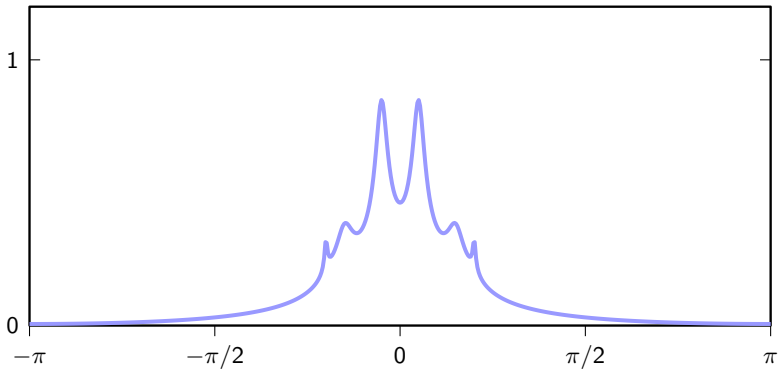
Denoising revisited



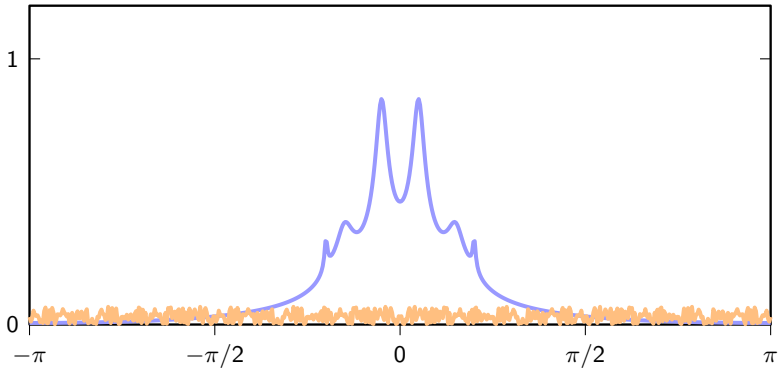
Denoising in the frequency domain



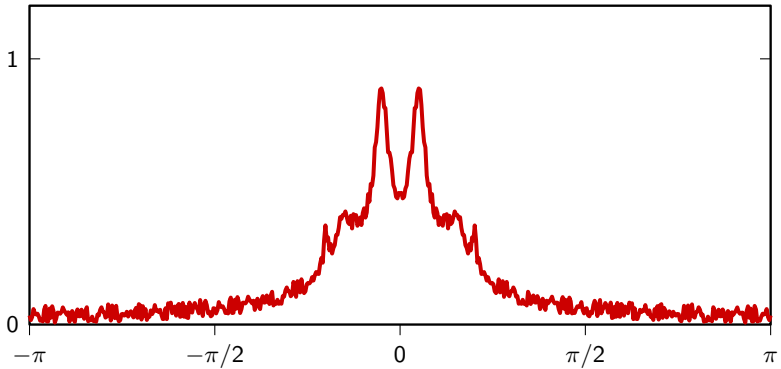
Denoising in the frequency domain



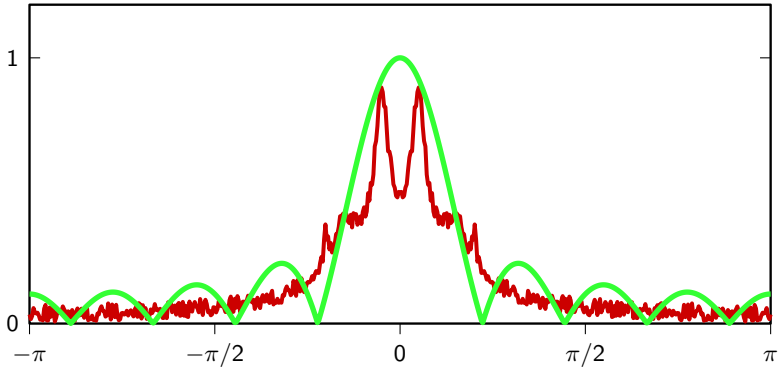
Denoising in the frequency domain



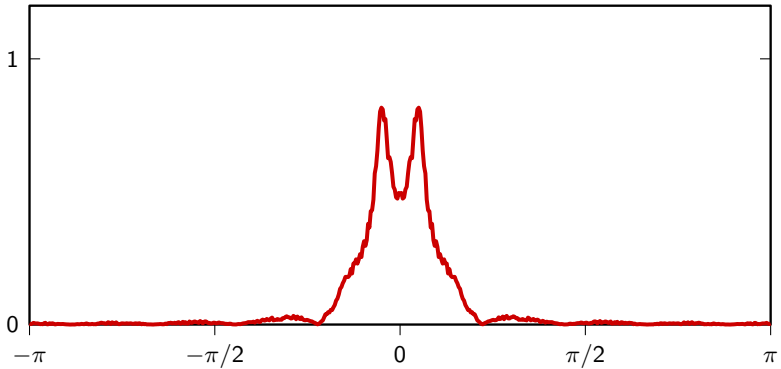
Denoising in the frequency domain



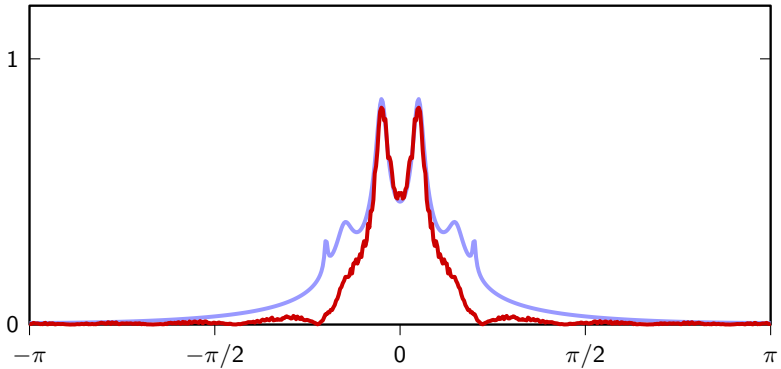
Denoising in the frequency domain



Denoising in the frequency domain



Denoising in the frequency domain



What about the phase?

Assume $|H(\omega)| = 1$

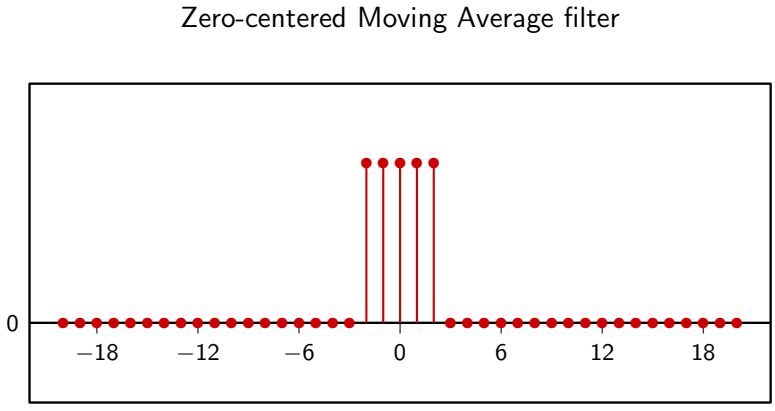
- zero phase: $\angle H(\omega) = 0$
- linear phase: $\angle H(\omega) = d\omega$, $d \in \mathbb{R}$
- nonlinear phase

Zero phase

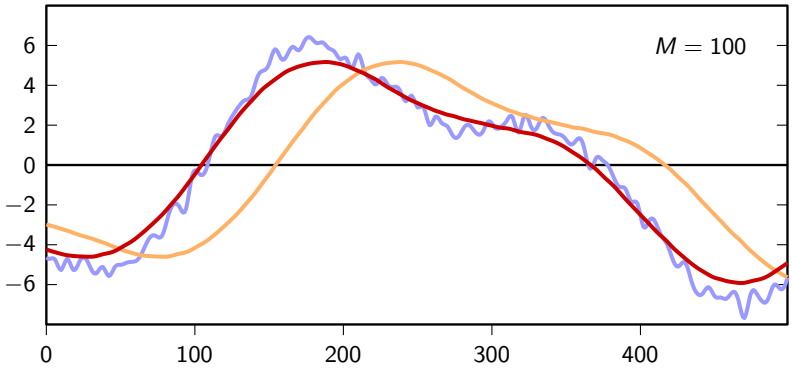
If $\angle H(\omega) = 0$:

- $H(\omega) \in \mathbb{R}$
- impulse response must be real and symmetric
- no causal filter can have a zero phase response
- a zero-phase filter has no processing delay

Noncausal moving average

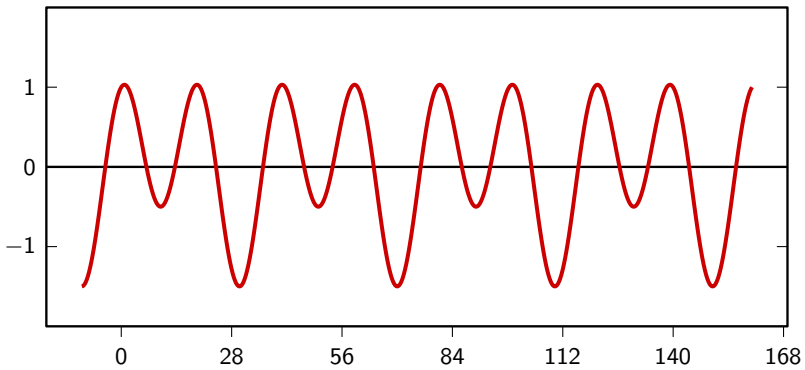


Causal vs Noncausal Moving Average



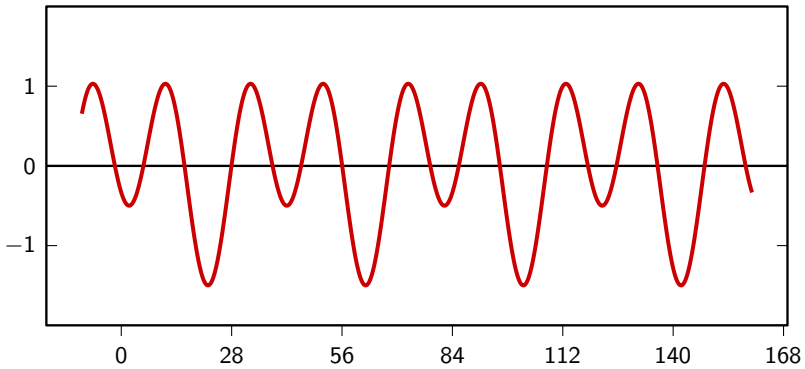
Phase and signal shape

$$x[n] = \frac{1}{2}\sin(\omega_0 n) + \cos(2\omega_0 n) \qquad \omega_0 = \frac{2\pi}{40}$$



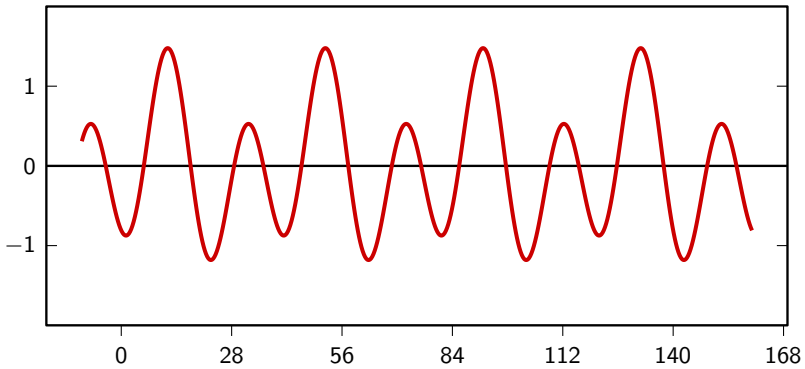
Phase and signal shape: linear phase offset

$$x[n] = \frac{1}{2}\sin(\omega_0 n + \theta_0) + \cos(2\omega_0 n + 2\theta_0) \qquad \theta_0 = \frac{8\pi}{5}$$

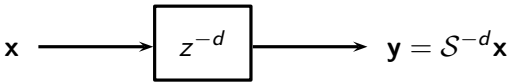


Phase and signal shape: nonlinear phase offset

$$x[n] = \frac{1}{2}\sin(\omega_0 n) + \cos(2\omega_0 n + 2\theta_0)$$



Linear phase



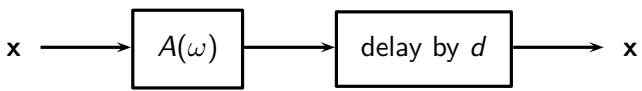
simple delay

- $y[n] = x[n-d]$
- $Y(\omega) = e^{-j\omega d}\, X(\omega)$
- $H(\omega) = e^{-j\omega d}$
- linear phase response

Linear phase systems

A linear phase system can be split as the cascade of

- a zero-phase system with frequency response $A(\omega) \in \mathbb{R}$
- a delay
- we will see later how this works if d is not an integer



Moving Average is linear phase

$$H(\omega) = \frac{1}{M} \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} e^{-j\frac{M-1}{2}\omega}$$

The processing delay introduced by a causal Moving Average is $(M-1)/2$ samples

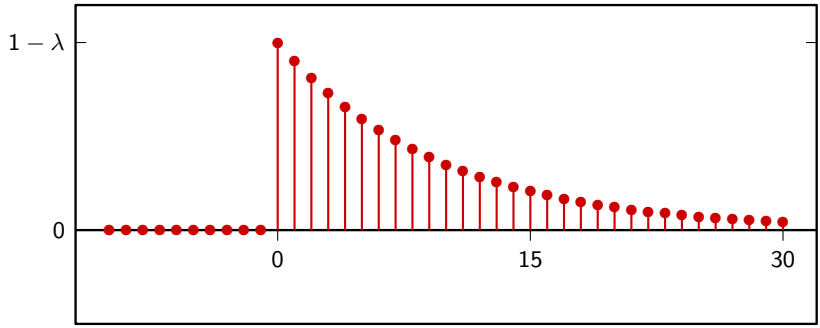
Noncausal moving average

$$h_c[n] = h[n + (M-1)/2]$$

$$\begin{aligned} H_c(\omega) &= e^{j\frac{M-1}{2}\omega} H(\omega) \\ &= \frac{1}{M} \frac{\sin\left(\frac{\omega}{2}M\right)}{\sin\left(\frac{\omega}{2}\right)} \in \mathbb{R} \end{aligned}$$

Leaky integrator: frequency response

$$h[n] = (1 - \lambda)\lambda^n \, u[n]$$



Leaky integrator: frequency response

$$H(\omega) = \frac{1-\lambda}{1-\lambda e^{-j\omega}}$$

Finding magnitude and phase require a little algebra...

Leaky integrator: frequency response

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: frequency response

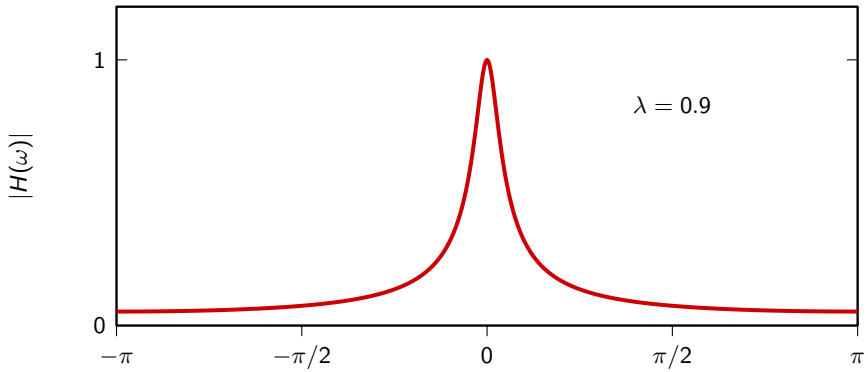
$$H(\omega) = \frac{1-\lambda}{(1-\lambda\cos\omega) - j\lambda\sin\omega}$$

so that:

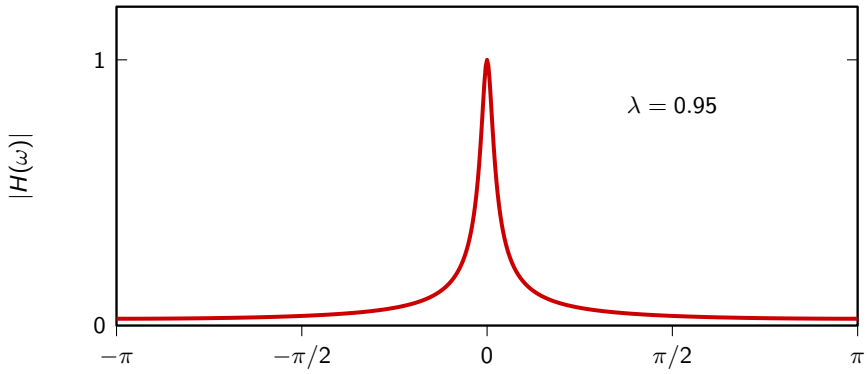
$$|H(\omega)|^2 = \frac{(1-\lambda)^2}{1-2\lambda\cos\omega+\lambda^2}$$

$$\angle H(\omega) = \tan^{-1}\left[\frac{\lambda\sin\omega}{1-\lambda\cos\omega}\right]$$

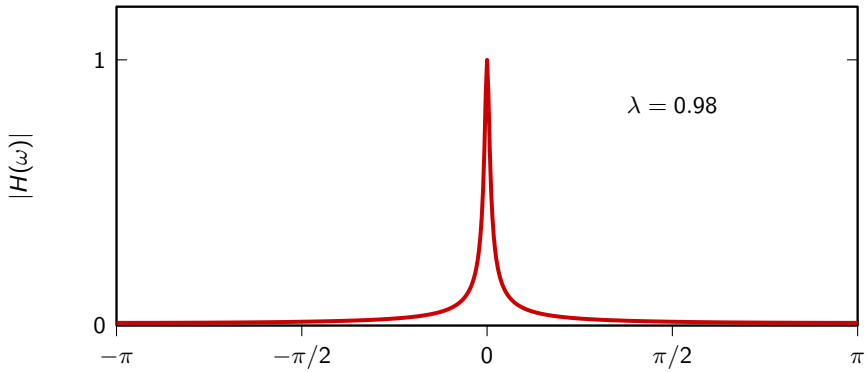
Leaky integrator: magnitude response



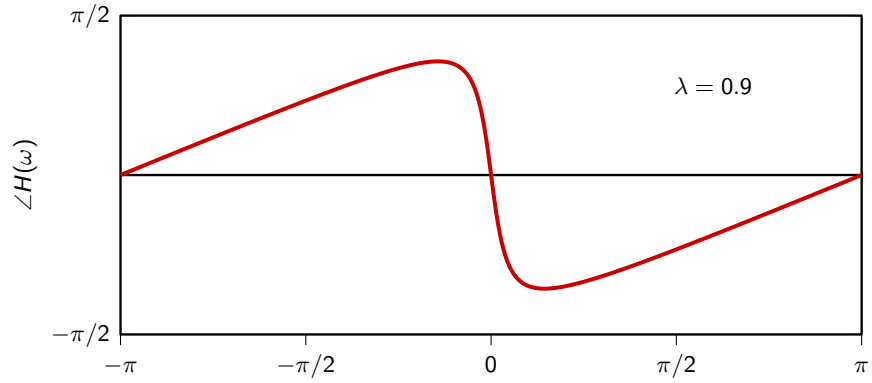
Leaky integrator: magnitude response



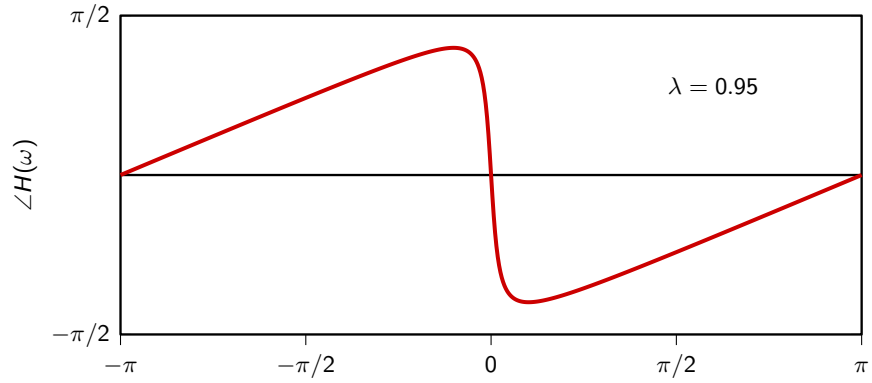
Leaky integrator: magnitude response



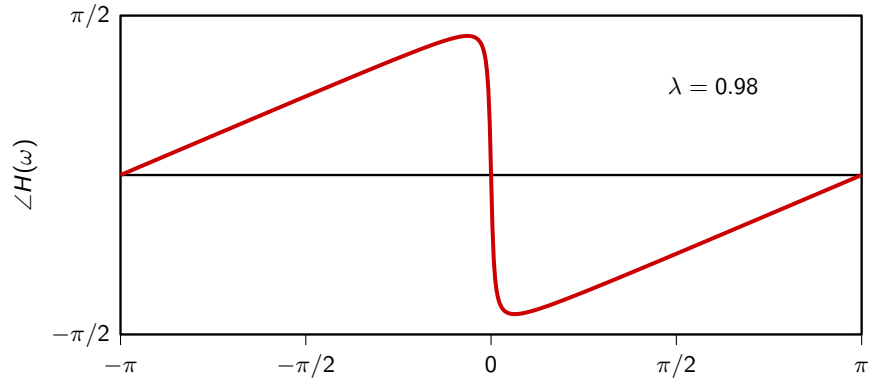
Leaky integrator: phase response



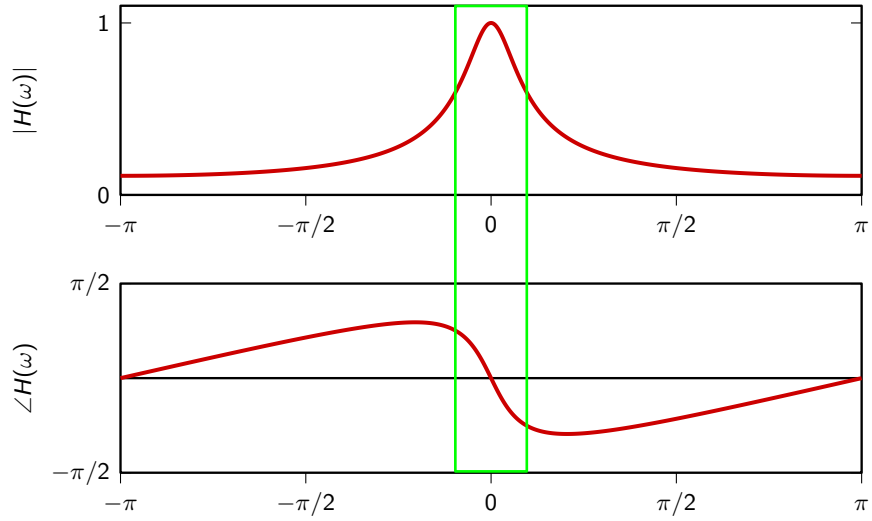
Leaky integrator: phase response



Leaky integrator: phase response



Phase is sufficiently linear where it matters



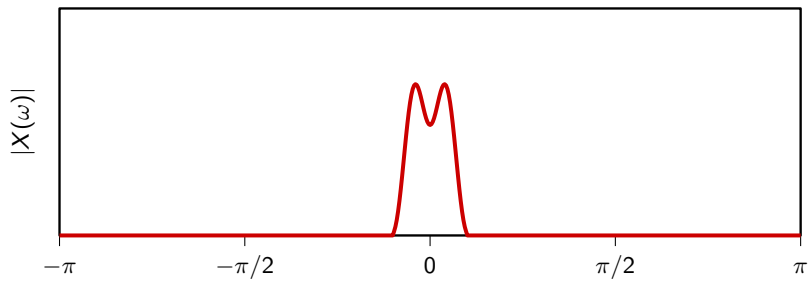
Modulation and demodulation

Classifying signals in frequency

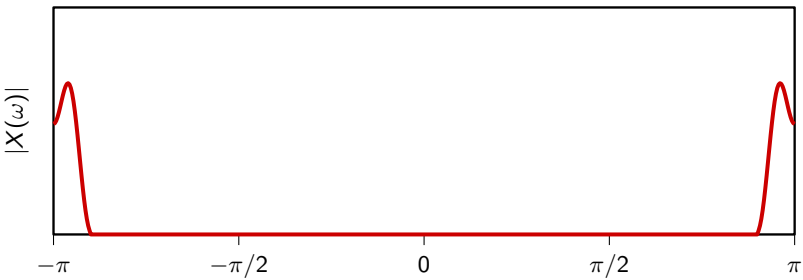
Three broad categories according to where most of the spectral energy resides:

- lowpass signals (also known as “baseband” signals)
- highpass signals
- bandpass signals

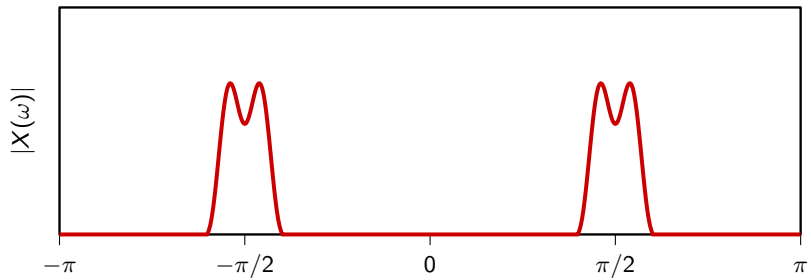
Lowpass example



Highpass example



Bandpass example

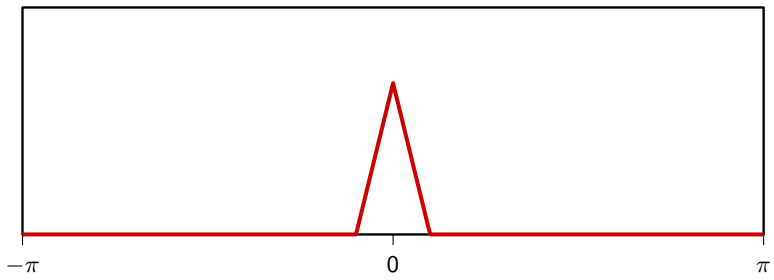


Sinusoidal modulation

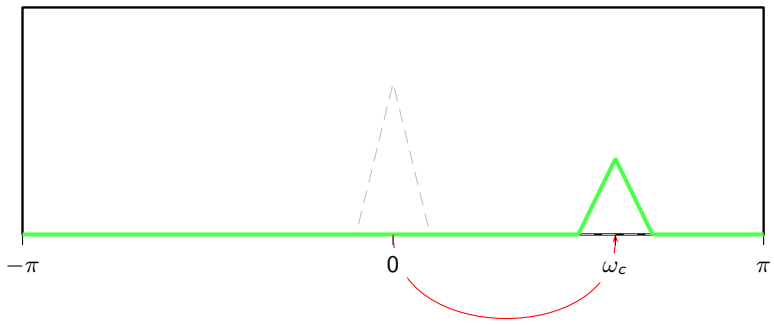
$$\begin{aligned}\text{DTFT}\left\{x[n]\cos(\omega_c n)\right\} &= \text{DTFT}\left\{\frac{1}{2}e^{j\omega_c n}x[n] + \frac{1}{2}e^{-j\omega_c n}x[n]\right\} \\ &= \frac{1}{2}\left[X(\omega-\omega_c) + X(\omega+\omega_c)\right]\end{aligned}$$

- usually $x[n]$ baseband
- ω_c is the *carrier* frequency

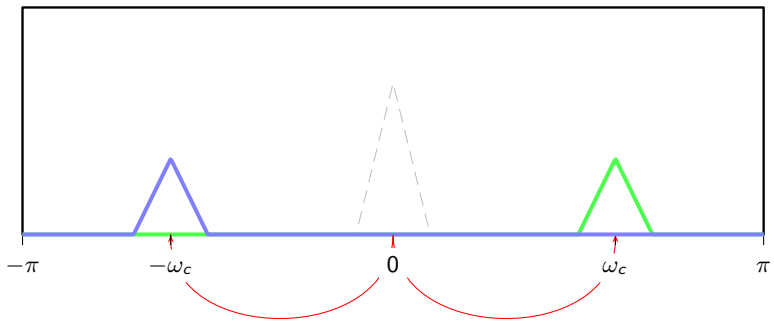
Example



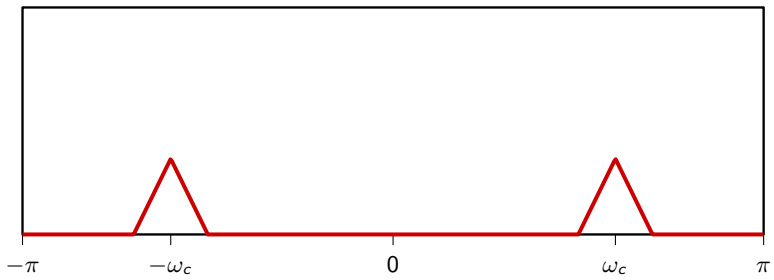
Example



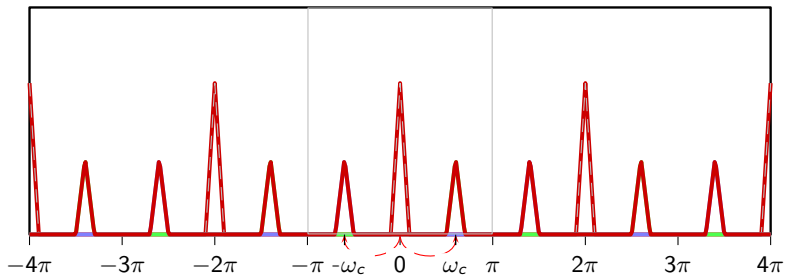
Example



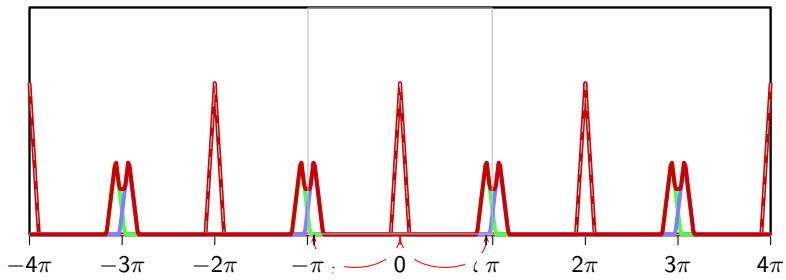
Example



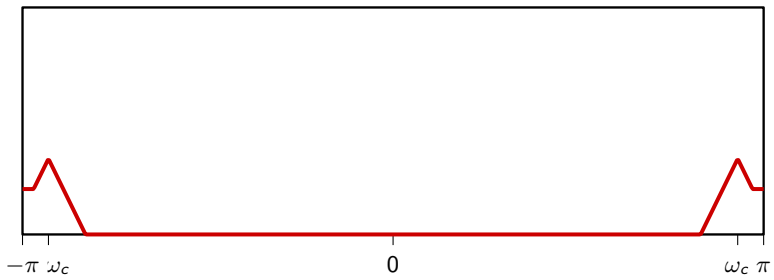
Again, explicitly showing the periodicity of the spectrum



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

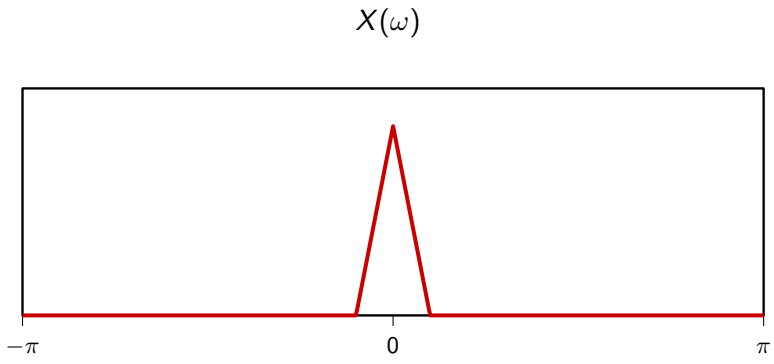
Demodulation

$$y[n] = x[n]\cos(\omega_c n) \qquad Y(\omega) = \frac{1}{2}\left[X(\omega - \omega_c) + X(\omega + \omega_c)\right]$$

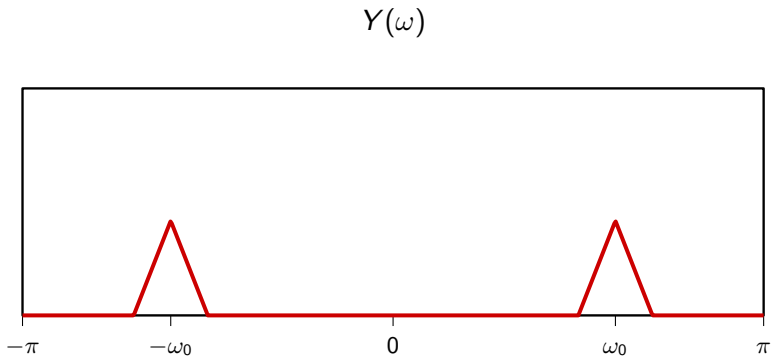
$$x'[n] = 2y[n]\cos(\omega_c n) \qquad X'(\omega) = Y(\omega - \omega_c) + Y(\omega + \omega_c)$$

$$= X(\omega) + \frac{1}{2}\left[X(\omega - 2\omega_c) + X(\omega + 2\omega_c)\right]$$

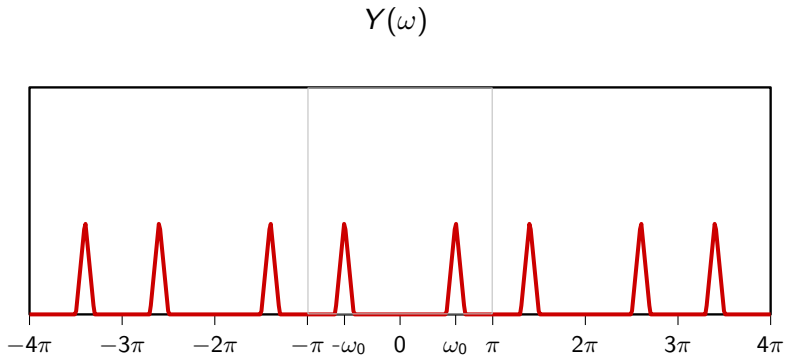
Demodulation



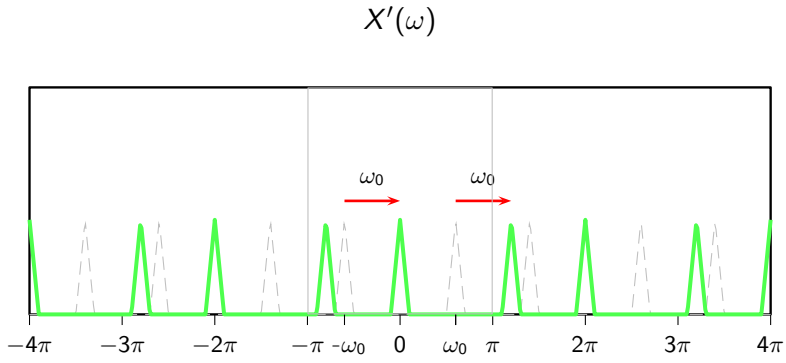
Demodulation



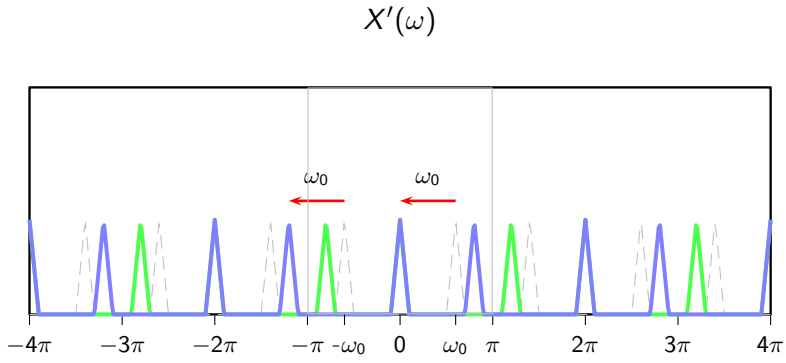
Demodulation



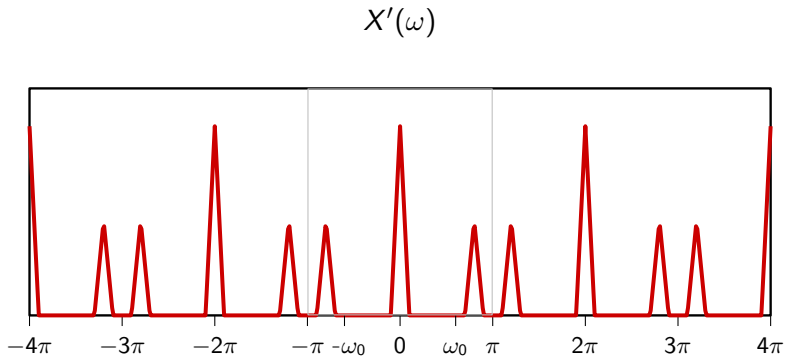
Demodulation



Demodulation



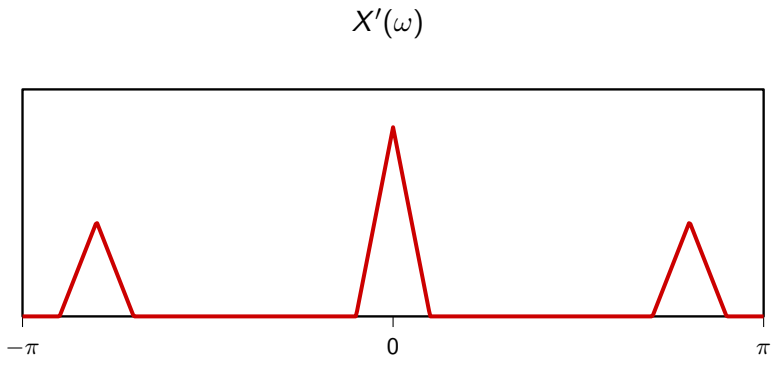
Demodulation



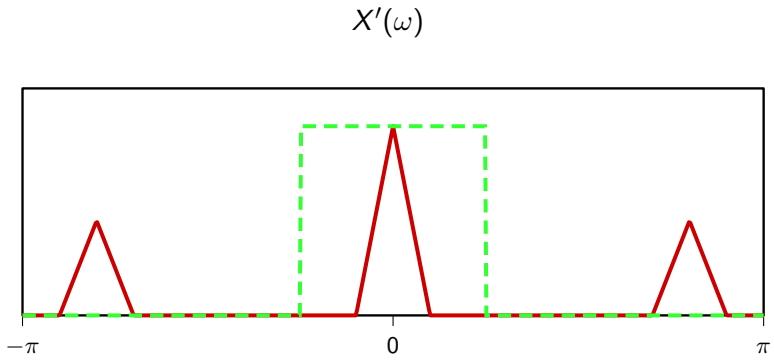
Demodulation

- we recovered the baseband signal exactly...
- but we have some spurious high-frequency components
- how do we get rid of those?

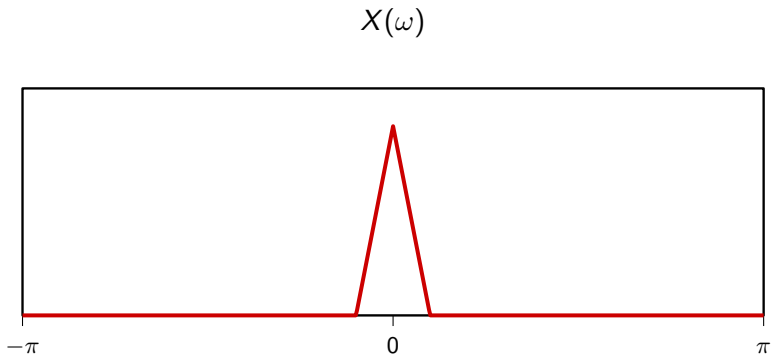
Demodulation with lowpass filtering



Demodulation with lowpass filtering



Demodulation with lowpass filtering



The modulation theorem

The convolution and modulation theorems

another example of time-frequency duality:

$$\mathbf{x} * \mathbf{y} \overset{\text{DTFT}}{\longleftrightarrow} \mathbf{XY}$$

$$\mathbf{xy} \overset{\text{DTFT}}{\longleftrightarrow} \mathbf{X} * \mathbf{Y}$$

What is the convolution of DTFTs?

in $\ell_2(\mathbb{Z})$

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

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

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

$$(\mathbf{x} * \mathbf{y})[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k]$$

in $L_2([-\pi, \pi])$

$$\langle \mathbf{X}, \mathbf{Y} \rangle = \frac{1}{2\pi} \int_{-\pi}^{\pi} X^*(\sigma)Y(\sigma)d\sigma$$

$$(\mathcal{R}\mathbf{Y})(\sigma) = Y(-\sigma)$$

$$(\mathcal{S}^{-\omega}\mathcal{R}\mathbf{Y})(\sigma) = Y(\omega - \sigma)$$

$$(\mathbf{X} * \mathbf{Y})(\omega) = \langle \mathbf{X}^*, \mathcal{S}^{-\omega}\mathcal{R}\mathbf{Y} \rangle$$

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\sigma)Y(\omega - \sigma)d\sigma$$

Modulation theorem

$$\mathbf{w} = \mathbf{x}\,\mathbf{y}$$

$$\mathbf{W} = \mathbf{X} * \mathbf{Y}$$

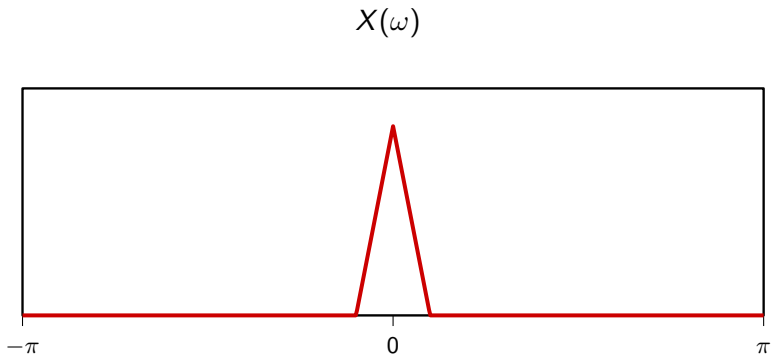
$$W(\omega) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\sigma) Y(\omega - \sigma) d\sigma$$

Modulation theorem: proof

Let's compute the inverse DTFT of $\mathbf{W} = \mathbf{X} * \mathbf{Y}$ at index n:

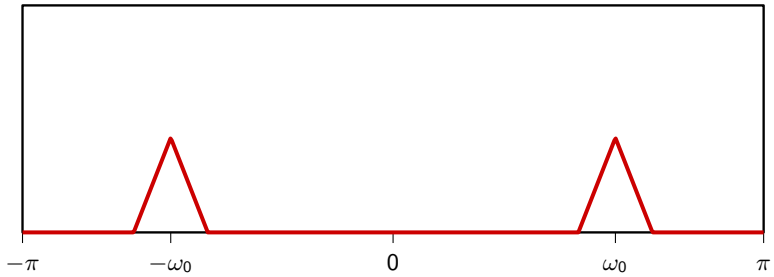
$$\begin{aligned}
\frac{1}{2\pi}\int_{-\pi}^{\pi} W(\omega)e^{j\omega n}d\omega &= \frac{1}{(2\pi)^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} X(\sigma)Y(\omega-\sigma)e^{j\omega n}d\sigma d\omega \\
&= \frac{1}{(2\pi)^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} X(\sigma)Y(\omega-\sigma)e^{j\sigma n}e^{j(\omega-\sigma)n}d\sigma d\omega \\
&= \frac{1}{2\pi}\int_{-\pi}^{\pi} X(\sigma)e^{j\sigma n}d\sigma \,\frac{1}{2\pi}\int_{-\pi}^{\pi} Y(\omega-\sigma)e^{j(\omega-\sigma)n}d\omega \\
&= x[n]\,y[n]
\end{aligned}$$

Sinusoidal modulation and Dirac deltas



Sinusoidal modulation and Dirac deltas

$$Y(\omega) = \text{DTFT}\left\{x[n]\,\cos(\omega_0 n)\right\}$$



Sinusoidal modulation and Dirac deltas

$$\begin{aligned}\text{DTFT}\{x[n]\cos\omega_0 n\} &= X(\omega) * \frac{1}{2}[\tilde{\delta}(\omega-\omega_0)+\tilde{\delta}(\omega+\omega_0)] \\ &= \frac{1}{4\pi}\int_{-\pi}^{\pi} X(\omega-\sigma)\tilde{\delta}(\sigma-\omega_0)d\sigma + \frac{1}{4\pi}\int_{-\pi}^{\pi} X(\omega-\sigma)\tilde{\delta}(\sigma+\omega_0)d\sigma \\ &= \frac{1}{2}[X(\omega-\omega_0)+X(\omega+\omega_0)]\end{aligned}$$

Dirac deltas must always be inside of an integral!

Modulation, frequency beatings, and tuning a guitar

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:

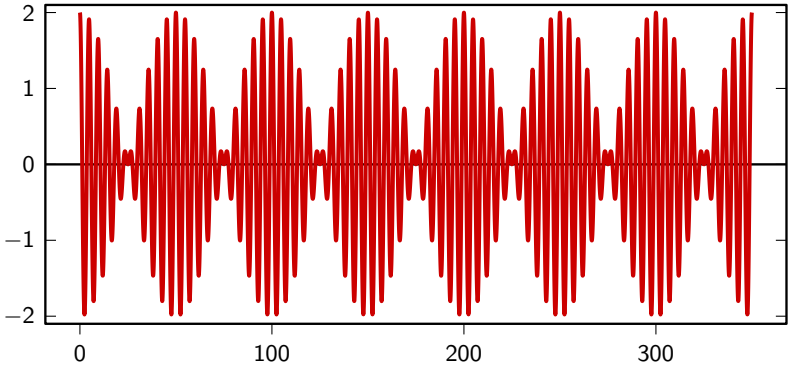
$$\begin{aligned} x[n] &= \cos(\omega_0 n) + \cos(\omega n) \\ &= 2\cos\left(\frac{\omega_0 + \omega}{2} n\right) \cos\left(\frac{\omega_0 - \omega}{2} n\right) \\ &\approx 2\cos(\Delta_\omega n)\cos(\omega_0 n) \end{aligned}$$

What we hear when we tune

$$x[n] \approx 2\cos(\Delta_\omega n)\cos(\omega_0 n) \quad \boxed{2\cos(\Delta_\omega n)} \cdot \boxed{\cos(\omega_0 n)}$$

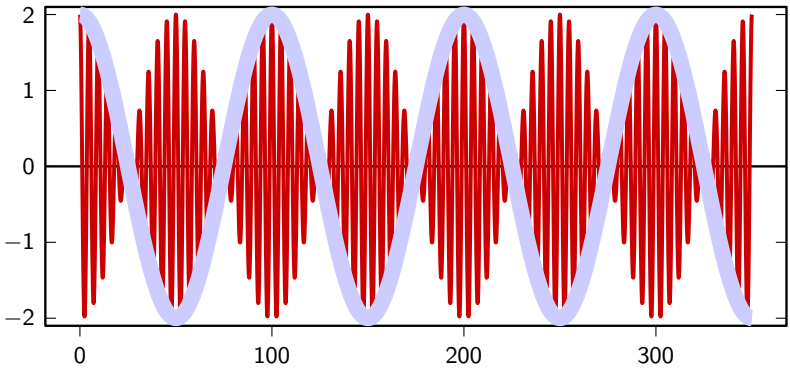
- "error" signal
- modulation at ω_0
- when $\omega \approx \omega_0$, the frequency of the error signal is too low to be heard; modulation brings it up to hearing range and we perceive it as amplitude oscillations of the carrier frequency

In the time domain...



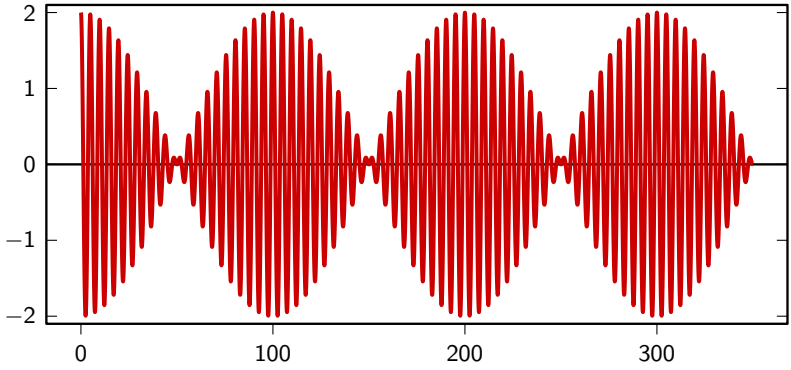
$\omega_0 = 2\pi \cdot 0.2, \quad \omega = 2\pi \cdot 0.22, \quad \Delta_\omega = 2\pi \cdot 0.0100$

In the time domain...



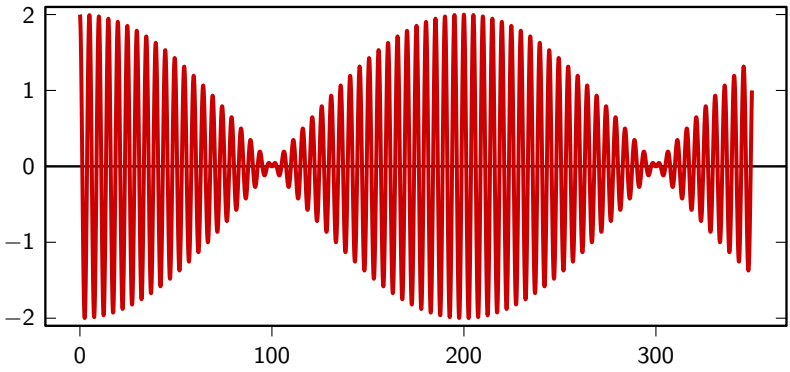
$\omega_0 = 2\pi \cdot 0.2, \quad \omega = 2\pi \cdot 0.22, \quad \Delta_\omega = 2\pi \cdot 0.0100$

In the time domain...



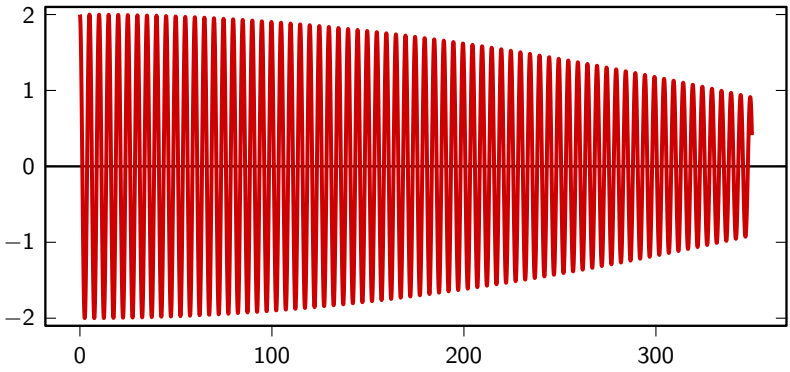
$$\omega_0 = 2\pi \cdot 0.2, \quad \omega = 2\pi \cdot 0.21, \quad \Delta_\omega = 2\pi \cdot 0.0050$$

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...



$$\omega_0 = 2\pi \cdot 0.2, \quad \omega = 2\pi \cdot 0.201, \quad \Delta_\omega = 2\pi \cdot 0.0005$$

Filter classification

Impulse response

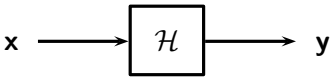
$$\mathbf{h} = \mathcal{H}\boldsymbol{\delta}$$

$$\mathbf{y} = \mathbf{h} * \mathbf{x}$$

Filter classification based on impulse response

- Finite Impulse Response (FIR)
- Infinite Impulse Response (IIR)
- causal/noncausal

Frequency response



$$\mathbf{H} = \mathrm{DTFT}\left\{\mathcal{H}\boldsymbol{\delta}\right\}$$

$$\mathbf{Y} = \mathbf{HX}$$

Magnitude response

Filters act as a multiplicative "mask" in the frequency domain:

- stopband: region where $|H(\omega)| \approx 0$
- passband: region where $|H(\omega)| \geq 1$

filters are classified according to passband location

Filter classification based on frequency response

types of magnitude response:

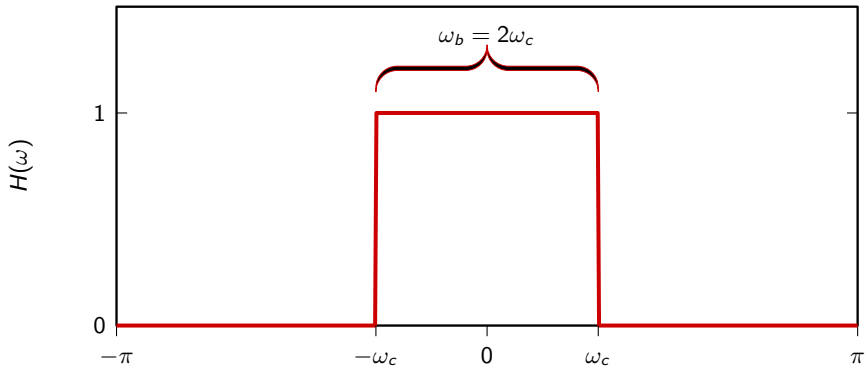
- Lowpass
- Highpass
- Bandpass
- Allpass

types of phase response:

- Linear phase
- Nonlinear phase

Ideal filters

What is the best lowpass we can think of?



Ideal lowpass filter

$$H(\omega) = \begin{cases} 1 & \text{for } |\omega| \leq \omega_c \\ 0 & \text{otherwise} \end{cases} \qquad (2\pi\text{-periodicity implicit})$$

- design parameter: cut-off frequency ω_c
- perfectly flat passband with unit amplitude
- infinite attenuation in stopband
- zero-phase (no delay)

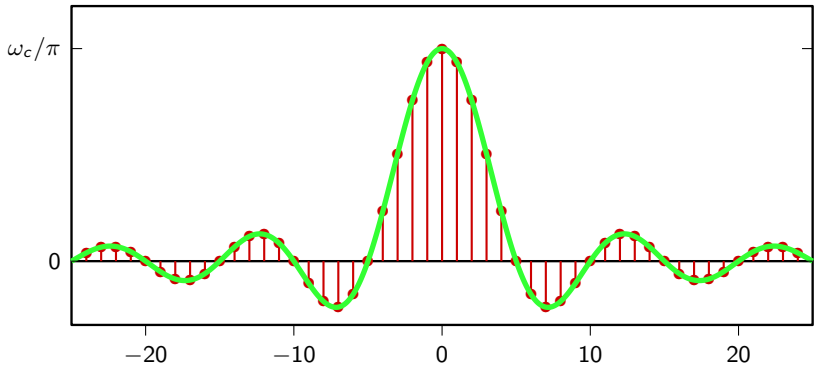
Ideal lowpass filter: impulse response

$$\mathbf{h} = \text{IDTFT}\{\mathbf{H}\}$$

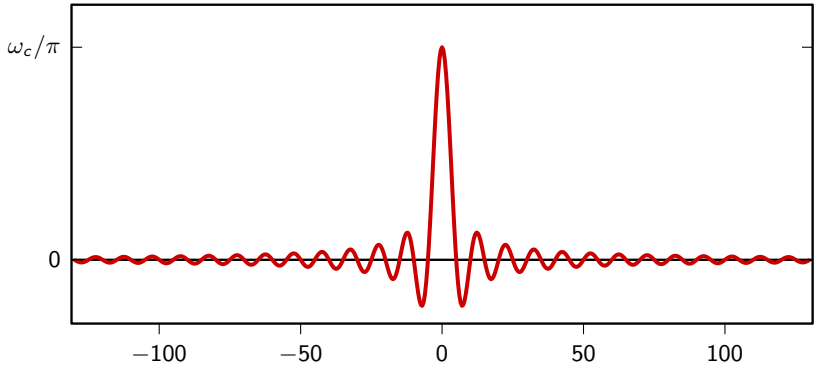
$$\begin{aligned} h[n] &= \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\omega)e^{j\omega n}d\omega \\ &= \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} e^{j\omega n}d\omega \\ &= \frac{1}{\pi n}\frac{e^{j\omega_c n} - e^{-j\omega_c n}}{2j} \\ &= \frac{\sin\omega_c n}{\pi n} \end{aligned}$$

Ideal lowpass filter: impulse response

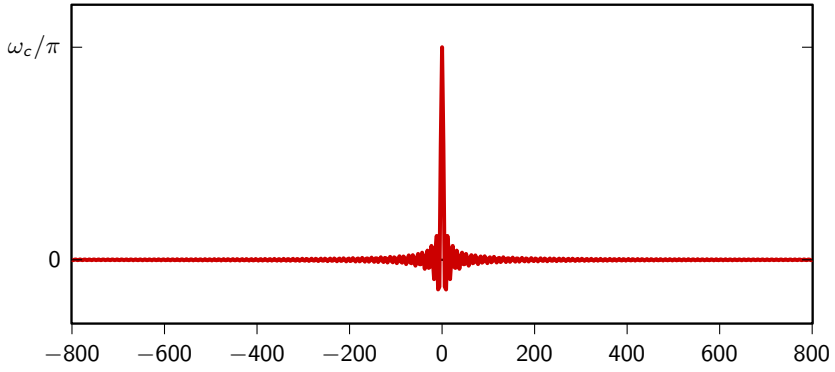
$$\omega_c = \pi/5$$



Ideal lowpass filter: impulse response



Ideal lowpass filter: impulse response



The sinc and rect functions

The sinc-rect pair:

$$\text{rect}(x) = \begin{cases} 1 & |x| \leq 1/2 \\ 0 & |x| > 1/2 \end{cases}$$

$$\text{sinc}(x) = \begin{cases} \dfrac{\sin(\pi x)}{\pi x} & x \neq 0 \\ 1 & x = 0 \end{cases}$$

(note that $\text{sinc}(m) = 0$ for $m \in \mathbb{Z} \setminus 0$)

The ideal lowpass in canonical form

$$\text{rect}\left(\frac{\omega}{2\omega_c}\right) \overset{\text{DTFT}}{\longleftrightarrow} \frac{\omega_c}{\pi}\,\text{sinc}\left(\frac{\omega_c}{\pi}n\right)$$

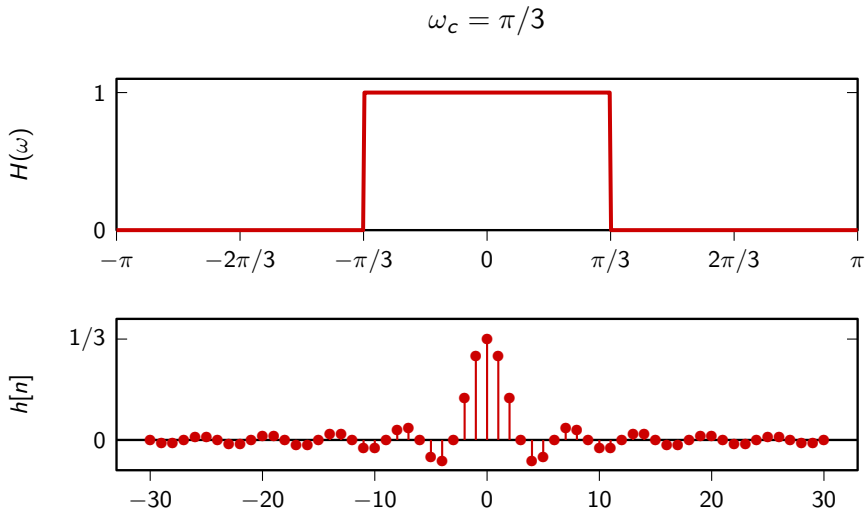
Example

Ideal lowpass filter with cutoff frequency $\omega_c = \pi/3$:

$$H(\omega) = \text{rect}\left(\frac{\omega}{2\pi/3}\right)$$

$$h[n] = \frac{1}{3}\text{sinc}\left(\frac{n}{3}\right)$$

Example


$\omega_c = \pi/3$
$H(\omega)$
1
0
$-\pi$
$-2\pi/3$
$-\pi/3$
0
$\pi/3$
$2\pi/3$
π
$h[n]$
1/3
0
-30
-20
-10
0
10
20
30

Time-frequency duality

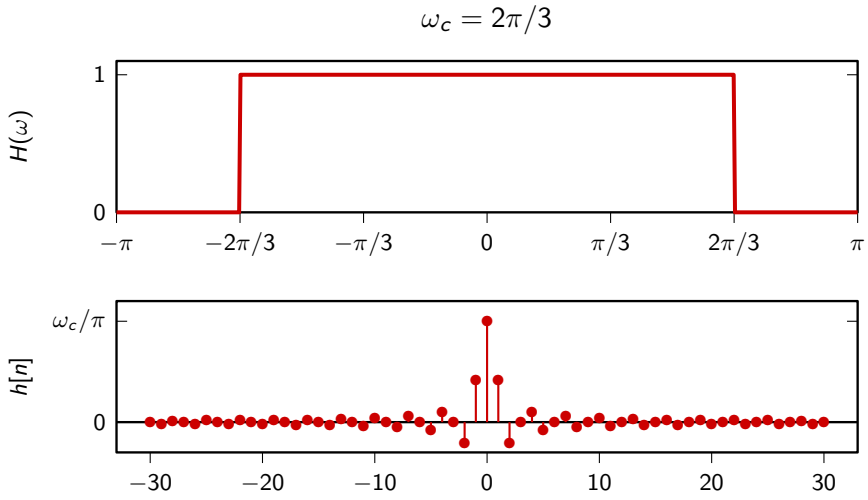
Always remember:

- narrow in frequency $\Rightarrow$ wide in time
- wide in frequency $\Rightarrow$ narrow in time

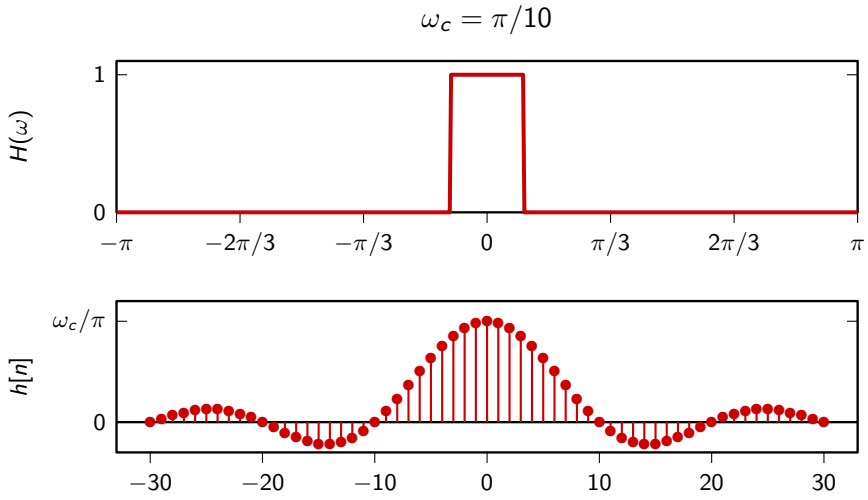
$$\text{rect}\left(\frac{\omega}{2\omega_c}\right) \overset{\text{DTFT}}{\longleftrightarrow} \frac{\omega_c}{\pi}\,\text{sinc}\left(\frac{\omega_c}{\pi}n\right)$$

- width in frequency: $2\omega_c$
- width in time: width of main lobe
- first zero crossing: $n = \pi/\omega_c$ (since $\text{sinc}(1) = 0$)
- width in time: $\propto 1/\omega_c$

Wide bandwidth



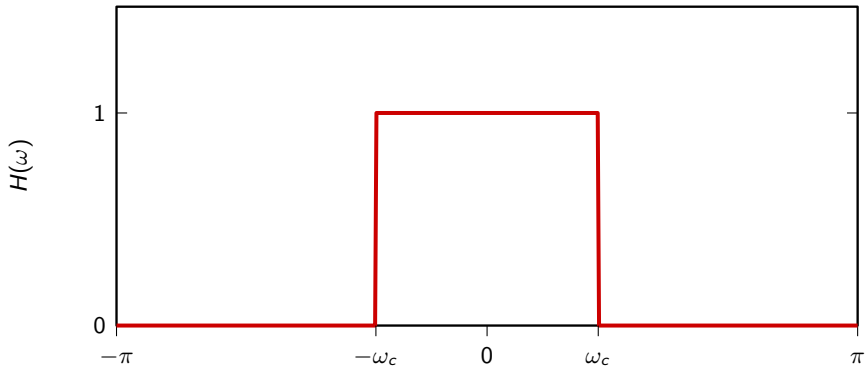
Narrow bandwidth



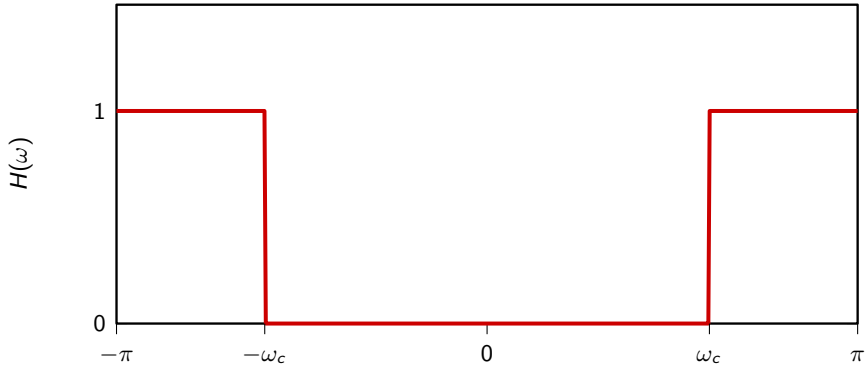
To help your memory

- frequency domain is easy to remember: $H(\omega) = \text{rect}\left(\frac{\omega}{2\omega_c}\right)$
- time domain is a sinc of the form $h[n] = a\,\text{sinc}(an)$
- to find a:
  - remember $\text{sinc}(0) = 1$, so $a = h[0]$
  - use the inverse DTFT formula for $n = 0$: $h[0] = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\omega)d\omega$
  - in this case, $h[0] = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\omega)d\omega = (2\omega_c)/(2\pi)$
  - $a = \omega_c/\pi$

From the ideal lowpass...



... to the ideal highpass



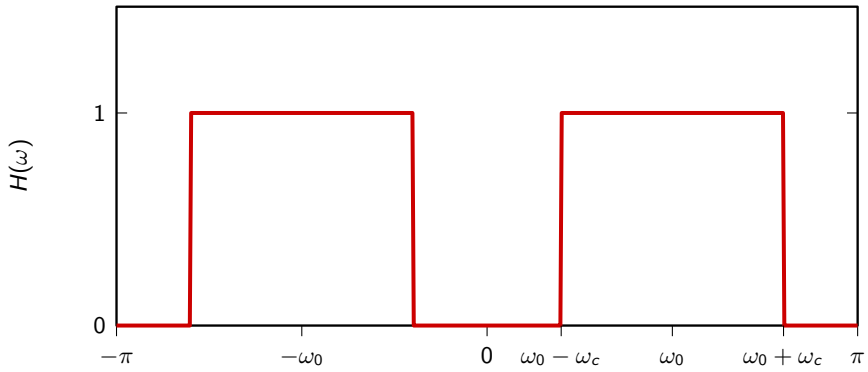
Ideal highpass filter

$$H_{hp}(\omega) = \begin{cases} 1 & \text{for } \pi \geq |\omega| \geq \omega_c \\ 0 & \text{otherwise} \end{cases} \qquad (2\pi\text{-periodicity implicit})$$

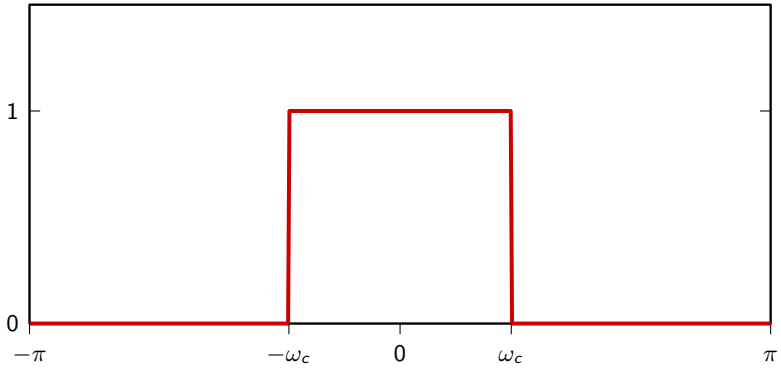
$$H_{hp}(\omega) = 1 - H_{lp}(\omega)$$

$$h_{hp}[n] = \delta[n] - \frac{\omega_c}{\pi}\text{sinc}\left(\frac{\omega_c}{\pi}n\right)$$

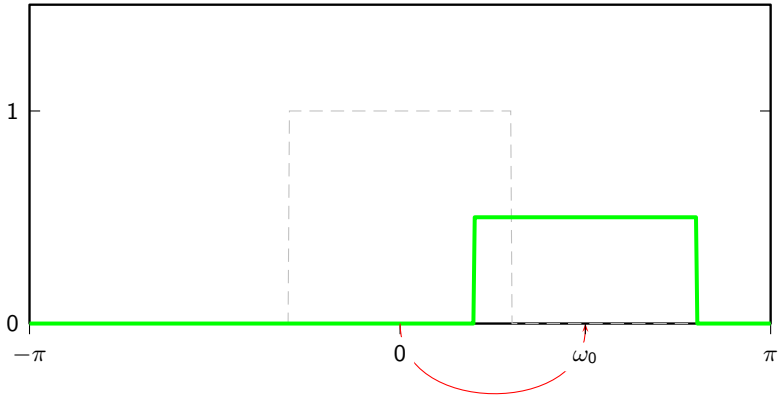
Ideal bandpass filter



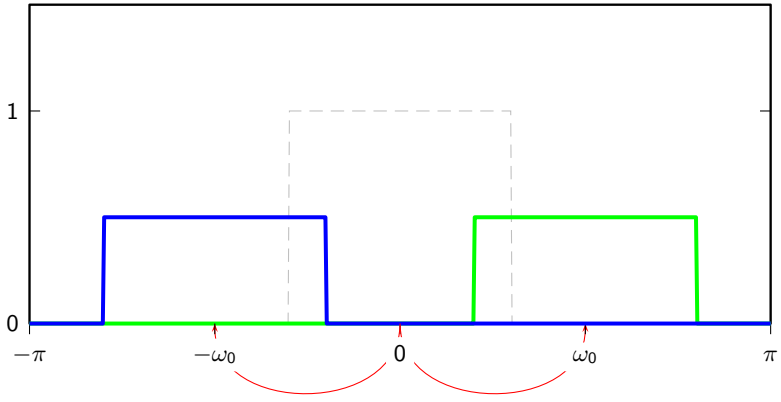
Ideal bandpass filter



Ideal bandpass filter



Ideal bandpass filter



Ideal bandpass filter

$$H_{bp}(\omega) = \begin{cases} 1 & \text{for } |\omega \pm \omega_0| \leq \omega_c \\ 0 & \text{otherwise} \end{cases} \qquad (2\pi\text{-periodicity implicit})$$

$$h_{bp}[n] = 2\cos(\omega_0 n)\frac{\omega_c}{\pi}\,\mathrm{sinc}\left(\frac{\omega_c}{\pi}n\right)$$

The bad news

the ideal lowpass is... *ideal*: it cannot be implemented in practice

Why we can't implement an ideal filter (I)

The impulse response $h[n] = \frac{\omega_c}{\pi} \operatorname{sinc}\left(\frac{\omega_c}{\pi} n\right)$:

- has two-sided infinte support
- cannot be made causal

computing a single output value requires knowing the entire input (i.e. knowing the future)

by contrast, note that the impulse response of computable IIR filters (such as the Leaky Integrator) can always be made causal

Why we can't implement an ideal filter (II)

If $\mathbf{x}$ has infinite support

$$y[n] = (\mathbf{h} * \mathbf{x})[n] = \frac{\omega_c}{\pi} \sum_{k=-\infty}^{\infty} x[k] \operatorname{sinc}\left(\frac{n-k}{\pi/\omega_c}\right)$$

involves an infinite number of terms, i.e. it's not computable in finite time.

Why we can't implement an ideal filter (III)

What if $\mathbf{x}$ has finite support (say from 0 to $N-1$)?

$$y[n] = (\mathbf{h} * \mathbf{x})[n] = \frac{\omega_c}{\pi} \sum_{k=0}^{N-1} x[k] \operatorname{sinc}\left(\frac{n-k}{\pi/\omega_c}\right)$$

can be computed but $y[n]$ will be nonzero for all n starting at $-\infty$;
again, we can't compute this in finite time

Why we can't implement an ideal filter (IV)

But can't we come up with an algorithm like we did for the Leaky Integrator?
After all, the LI has an infinite impulse response!

Unfortunately no computable algorithm will yield a rect-shaped filter.
We will see why in the upcoming lectures about rational transfer functions.

FIR approximations of ideal filters

Overview:

- Impulse truncation
- Window method
- Frequency sampling

How can we approximate an ideal lowpass?

Idea #1:

- pick ω_c
- compute ideal impulse response $\mathbf{h}$
- truncate $\mathbf{h}$ to a finite-support $\bar{\mathbf{h}}$
- $\bar{\mathbf{h}}$ defines an FIR filter, which we can always implement

Approximation by truncation

FIR approximation of length $M = 2N + 1$:

$$\bar{h}[n] = \begin{cases} \frac{\omega_c}{\pi} \operatorname{sinc}\left(\frac{\omega_c}{\pi} n\right) & |n| \leq N \\ 0 & \text{otherwise} \end{cases}$$

Why this looks like a good idea

minimization of MSE (norm in $L_2([-\pi, \pi])$) between ideal filter and its approximation:

$$\text{MSE} = \|\mathbf{H} - \bar{\mathbf{H}}\|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(\omega) - \bar{H}(\omega)|^2 d\omega$$

using Parseval's theorem

$$\begin{aligned} \text{MSE} &= \|\mathbf{H} - \bar{\mathbf{H}}\|^2 \\ &= \|\mathbf{h} - \bar{\mathbf{h}}\|^2 \qquad (\textit{norm in } \ell_2(\mathbb{Z})) \\ &= \sum_{n=-\infty}^{\infty} |h[n] - \bar{h}[n]|^2 \end{aligned}$$

Remember the conservation of energy

norm in $\ell_2(\mathbb{Z})$ *norm in* $L_2([-\pi, \pi])$

$$\|\mathbf{x}\|^2 = \|\mathbf{X}\|^2$$

$$\sum_{n=-\infty}^{\infty} |x[n]|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(\omega)|^2 d\omega$$

$$\left\| \frac{\omega_c}{\pi} \operatorname{sinc}\left(\frac{\omega_c}{\pi} n\right) \right\|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \operatorname{rect}\left(\frac{\omega}{2\omega_c}\right) \right|^2 d\omega = \frac{\omega_c}{\pi}$$

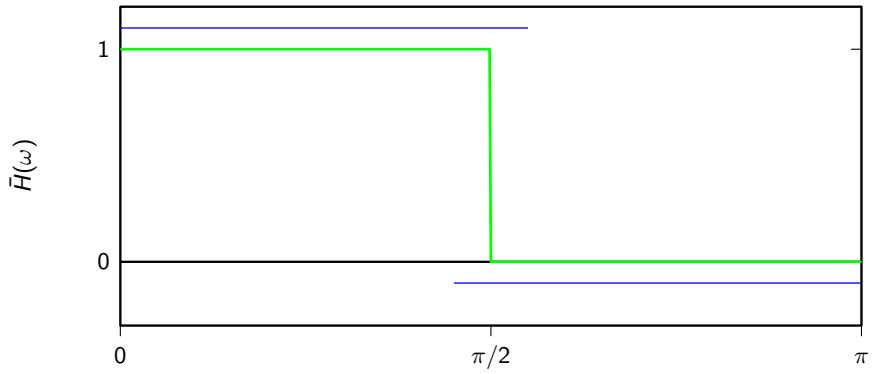
Keep the coefficients around $n = 0$

let $I = \{n_0, n_1, \ldots, n_M\}$ the set of indices of the coefficients we keep:

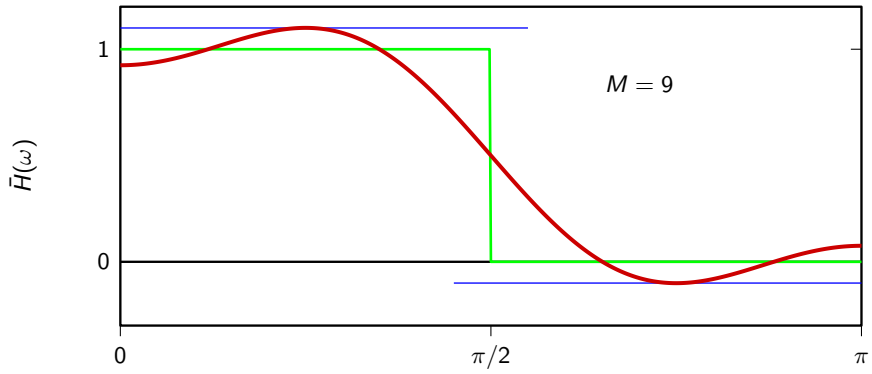
$$\begin{aligned}
\text{MSE} &= \sum_{n=-\infty}^{\infty} |h[n] - \bar{h}[n]|^2 = \sum_{n=-\infty}^{\infty} \left[|h[n]|^2 + |\bar{h}[n]|^2 - 2|h[n]||\bar{h}[n]|\right] \\
&= \sum_{n=-\infty}^{\infty} |h[n]|^2 - \sum_{n \in I} |\bar{h}[n]|^2 \\
&= \omega_c/\pi - \sum_{n \in I} |\bar{h}[n]|^2
\end{aligned}$$

- MSE is minimized by keeping the largest coefficients
- $|\operatorname{sinc}(\alpha n)| \propto 1/n$
- MSE is minimized by symmetric impulse truncation around zero

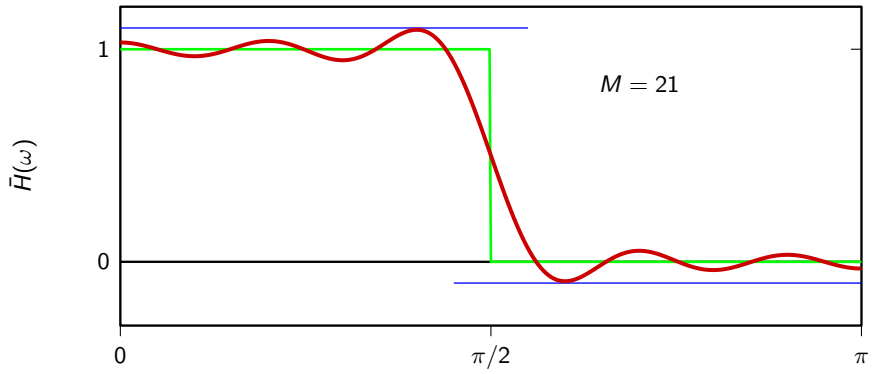
Why it's not such a good idea, actually



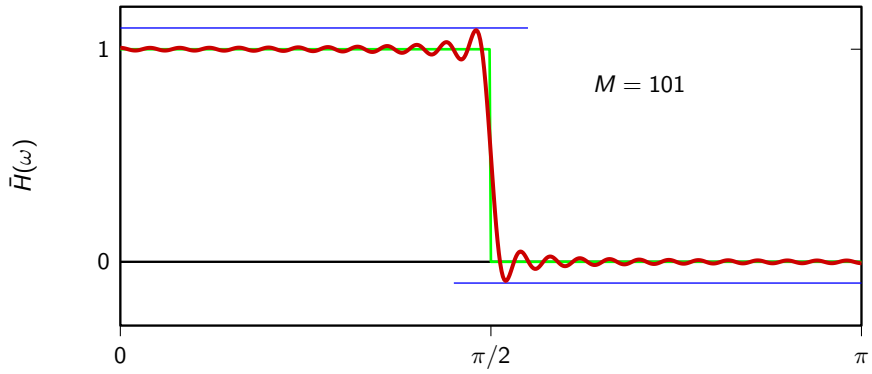
Why it's not such a good idea, actually



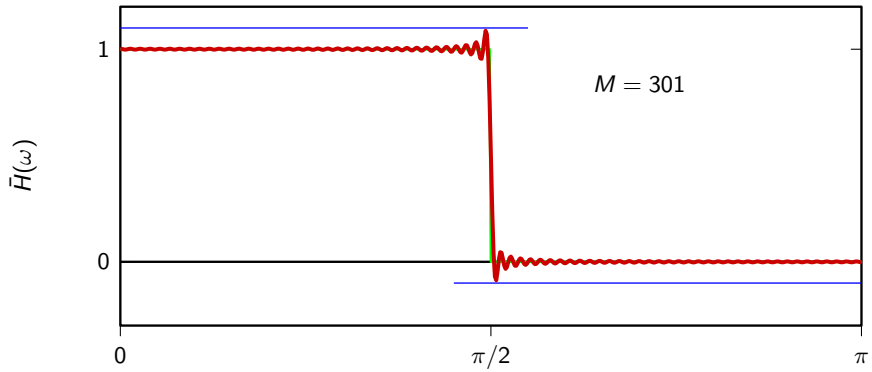
Why it's not such a good idea, actually



Why it's not such a good idea, actually



Why it's not such a good idea, actually



The Gibbs phenomenon

The maximum error around the cutoff frequency
is around 9% of the height of the jump
regardless of M

Understanding the Gibbs phenomenon

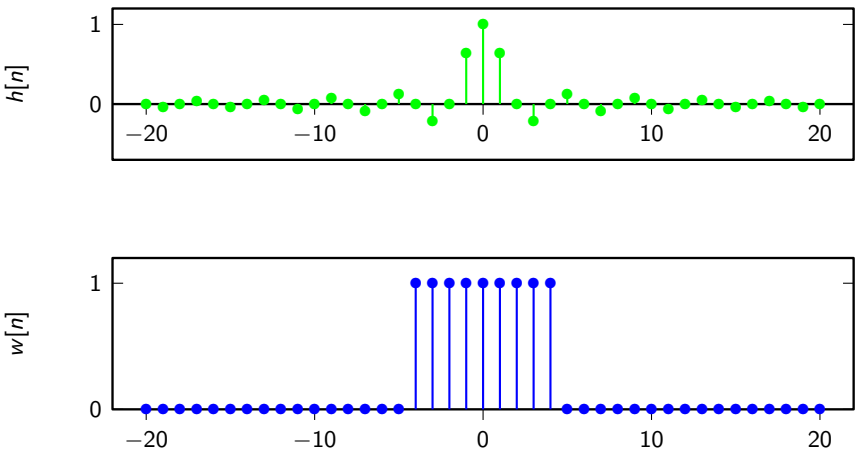
$$\bar{\mathbf{h}} = \mathbf{h}\,\mathbf{w}$$

$$w[n] = \begin{cases} 1 & |n| \leq N \\ 0 & \text{otherwise} \end{cases}$$

$$\bar{\mathbf{H}} = \mathbf{H} * \mathbf{W}$$

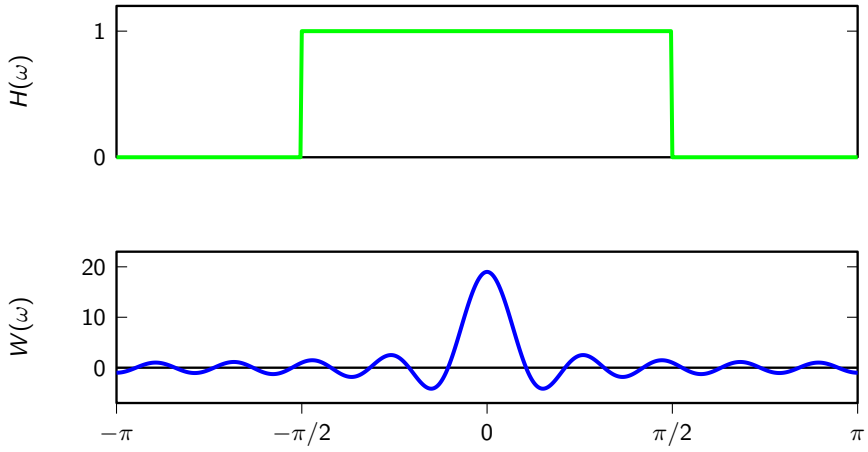
Understanding the Gibbs phenomenon

$$\bar{\mathbf{h}} = \mathbf{h}\,\mathbf{w}$$



Understanding the Gibbs phenomenon

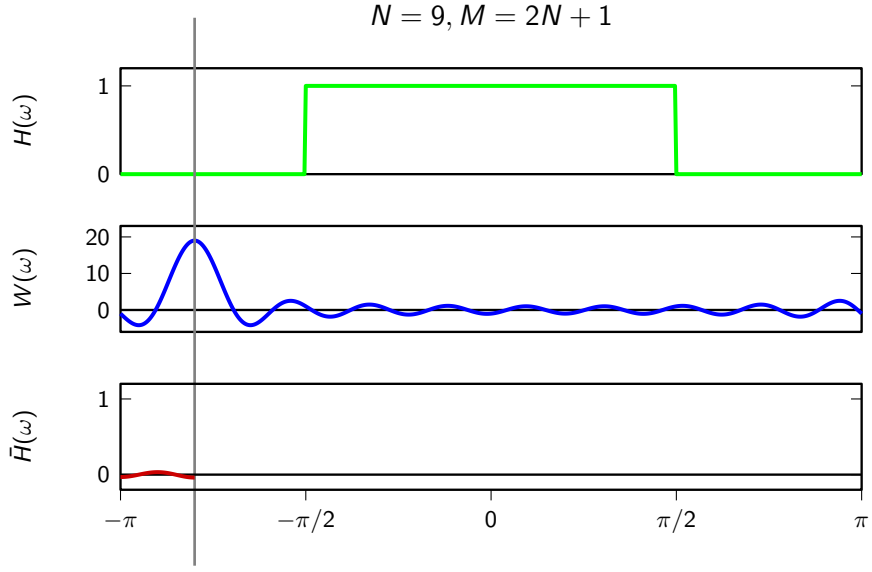
$$\bar{\mathbf{H}} = \mathbf{H} * \mathbf{W}, \qquad W(\omega) = \sin(\omega M/2)/\sin(\omega/2)$$



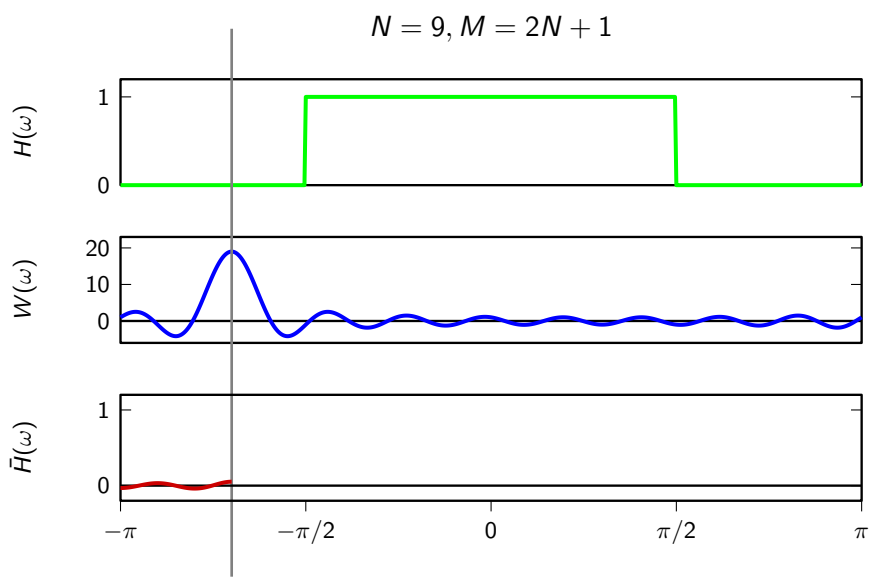
Remember the modulation theorem

$$(\mathbf{X} * \mathbf{Y})(\omega) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\sigma) Y(\omega - \sigma) d\sigma$$

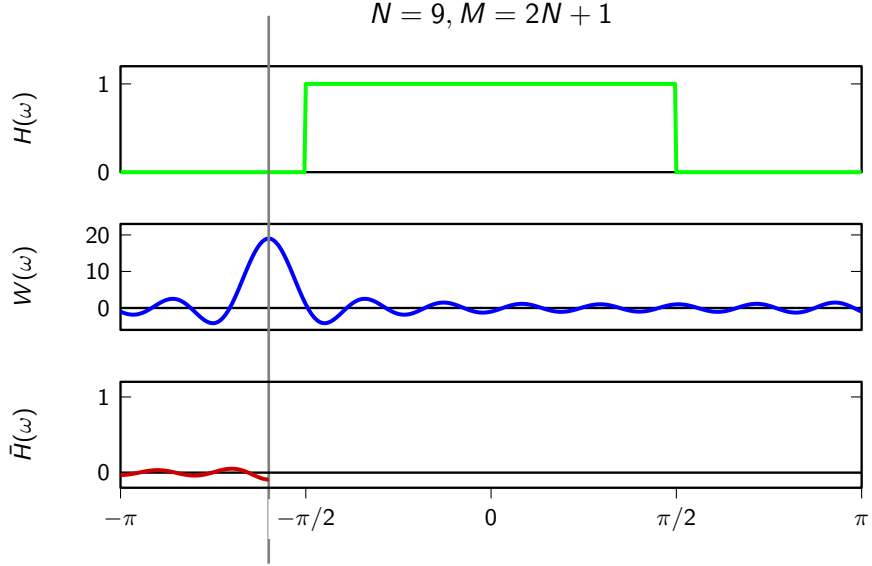
Implicit frequency-domain convolution



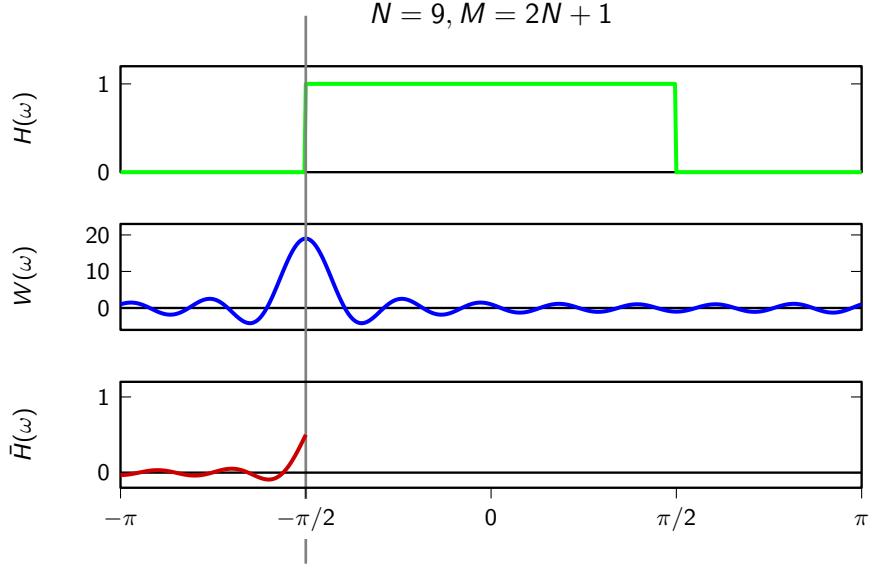
Implicit frequency-domain convolution



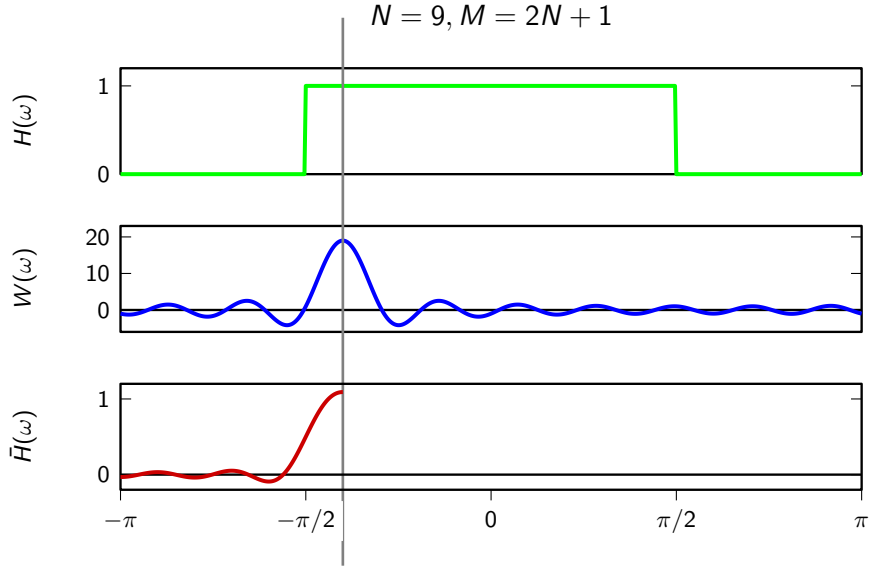
Implicit frequency-domain convolution



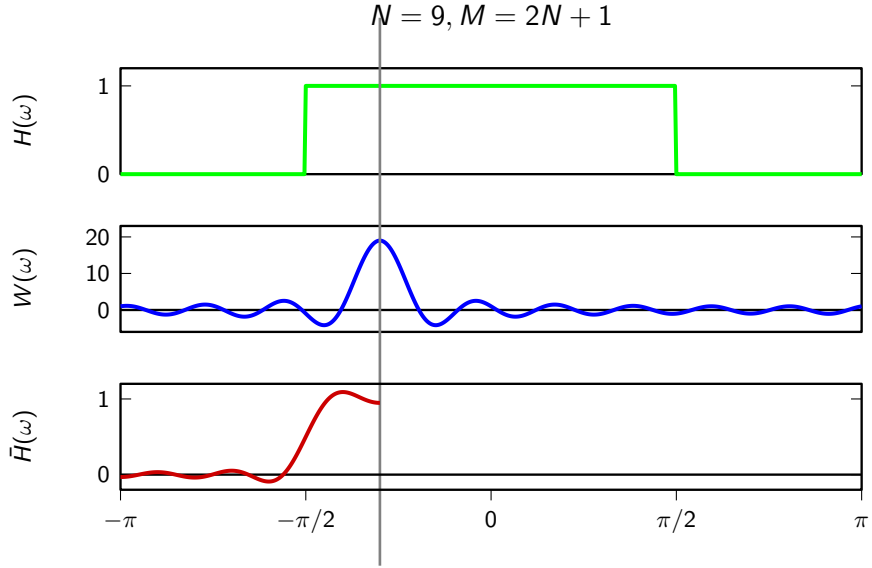
Implicit frequency-domain convolution



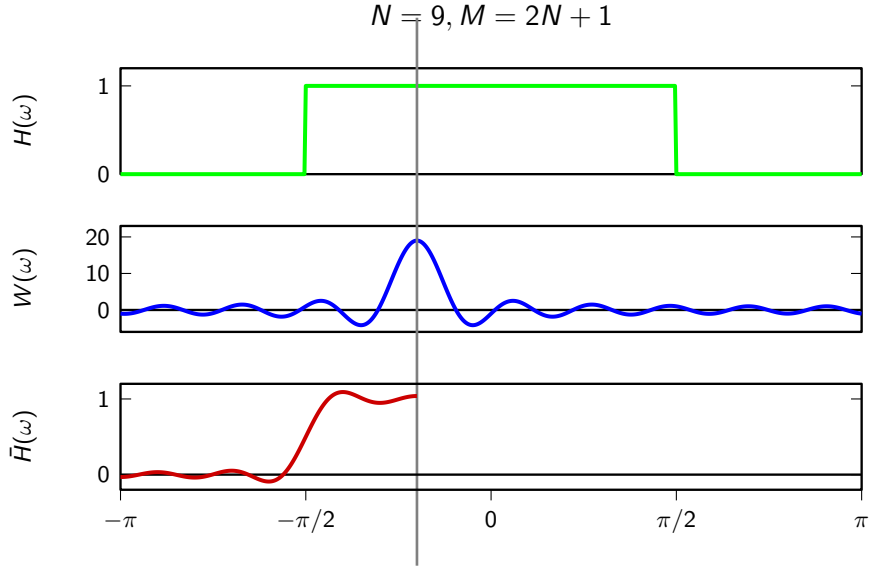
Implicit frequency-domain convolution



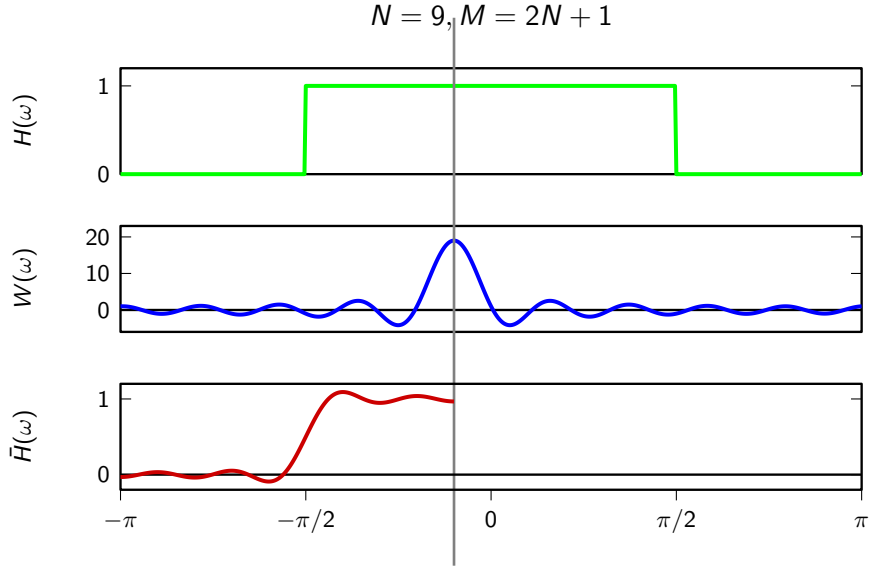
Implicit frequency-domain convolution



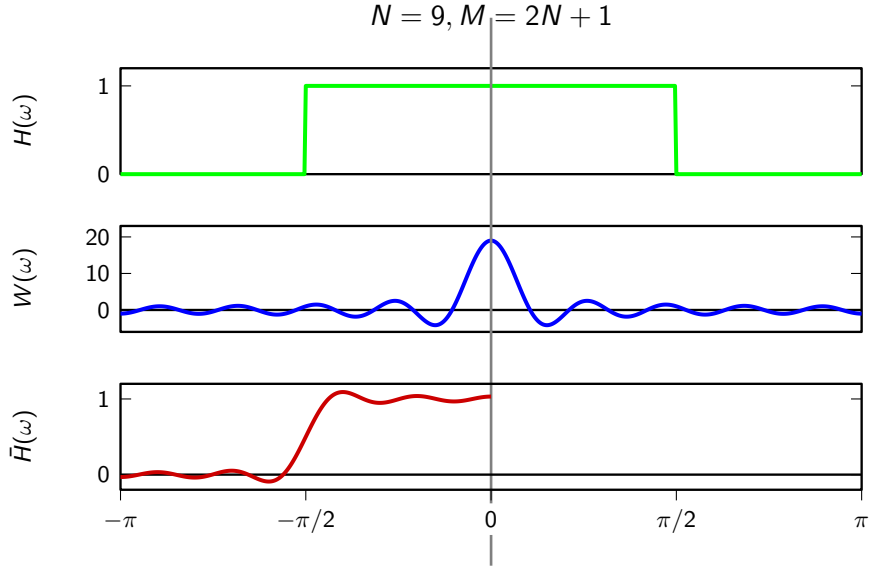
Implicit frequency-domain convolution



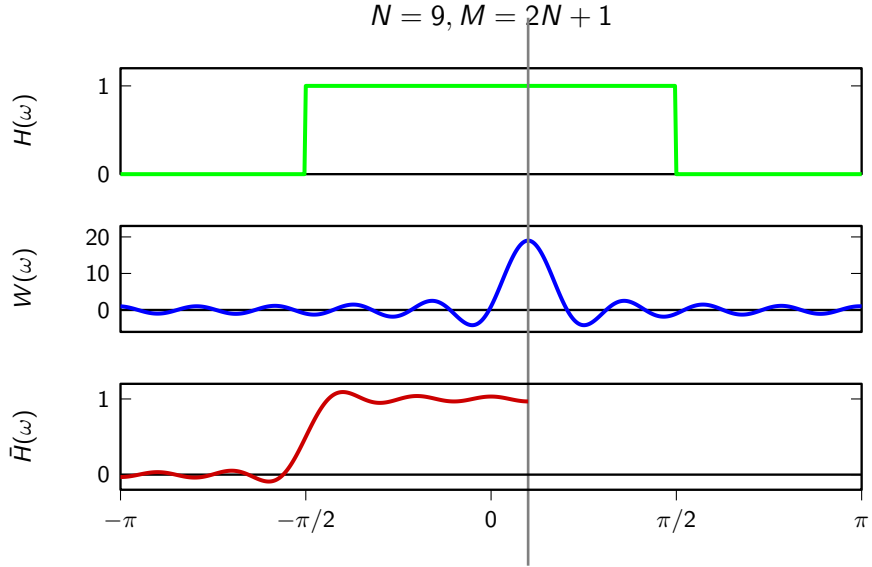
Implicit frequency-domain convolution



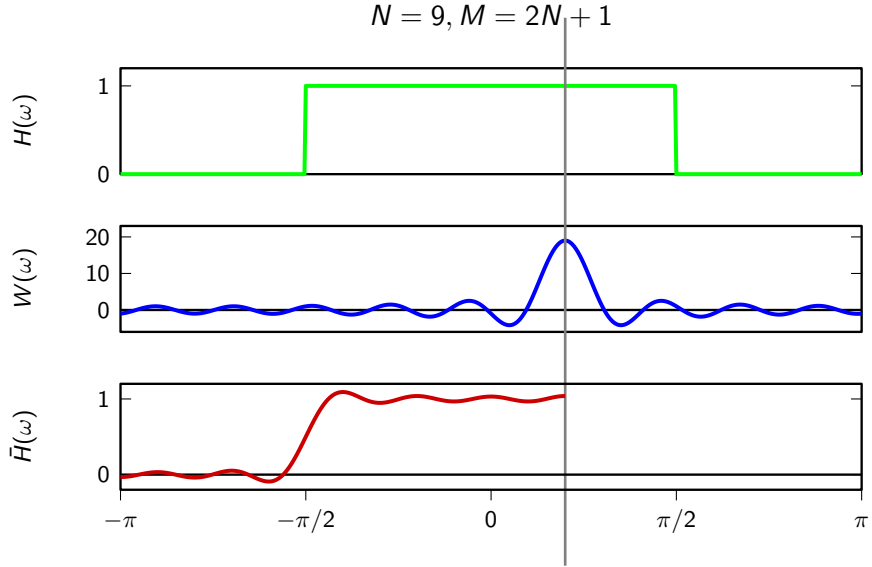
Implicit frequency-domain convolution



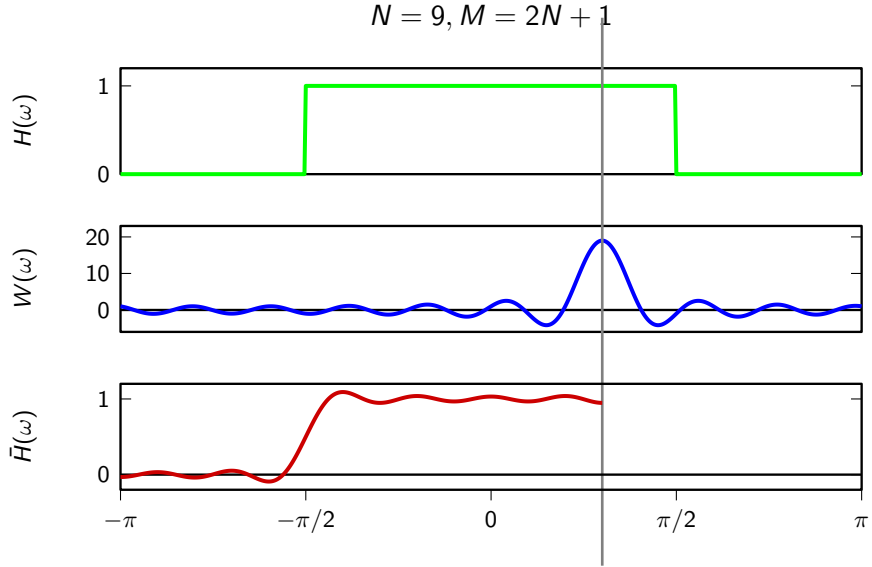
Implicit frequency-domain convolution



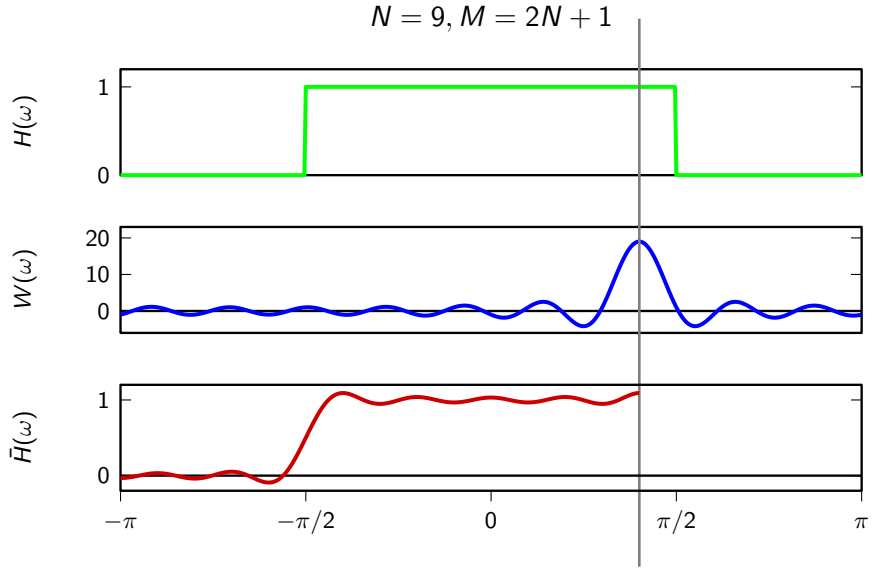
Implicit frequency-domain convolution



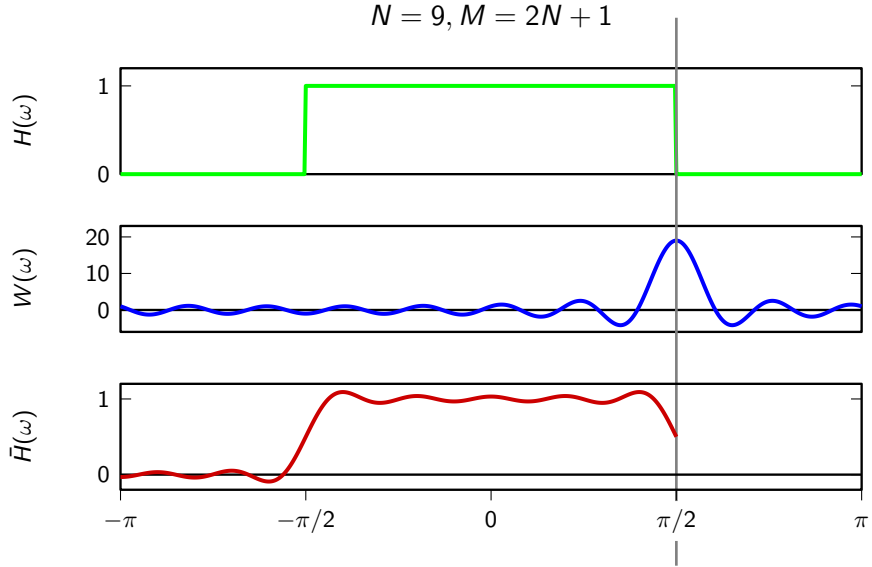
Implicit frequency-domain convolution



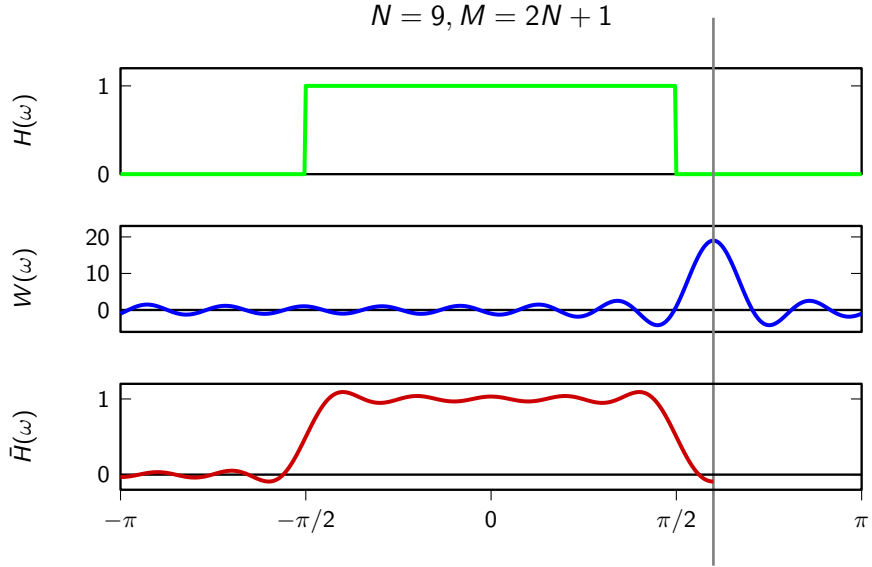
Implicit frequency-domain convolution



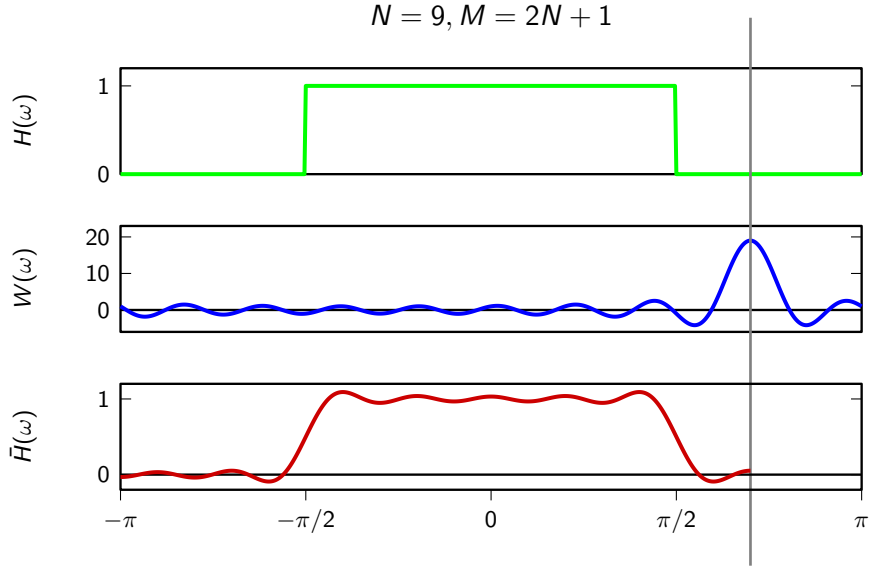
Implicit frequency-domain convolution



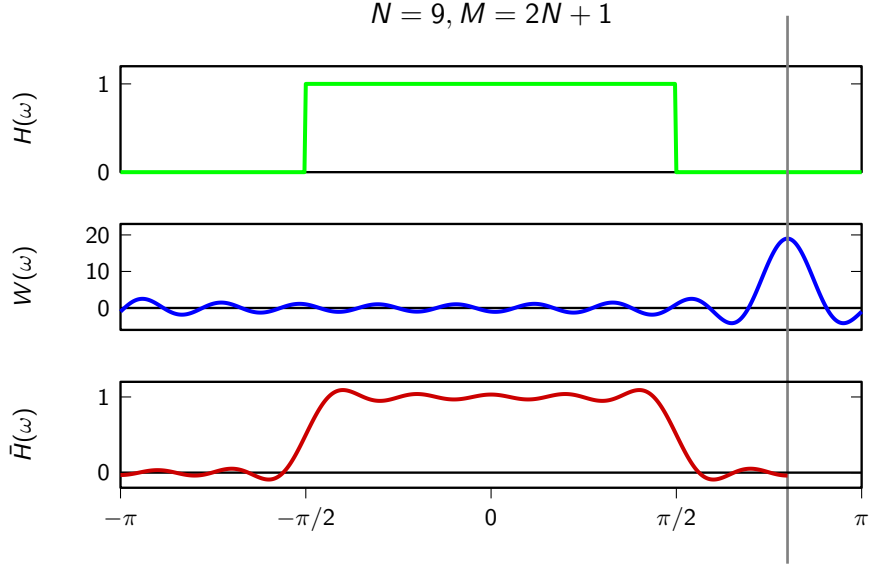
Implicit frequency-domain convolution



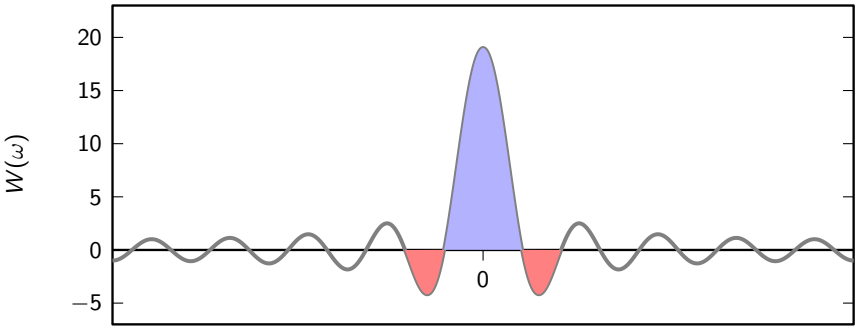
Implicit frequency-domain convolution



Implicit frequency-domain convolution



Mainlobe and sidelobes



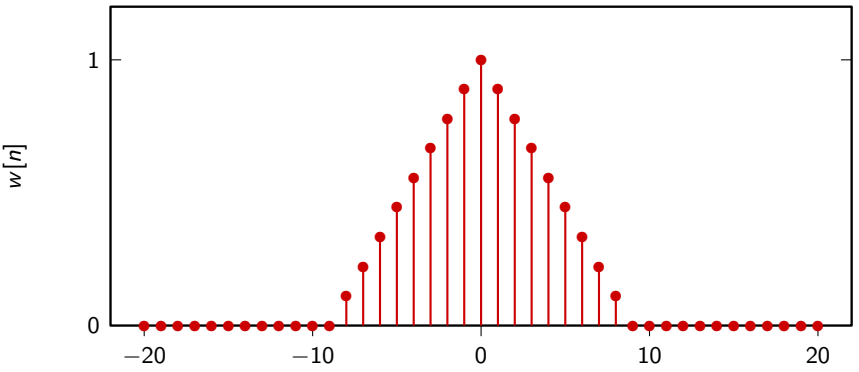
What if we change the window?

We want:

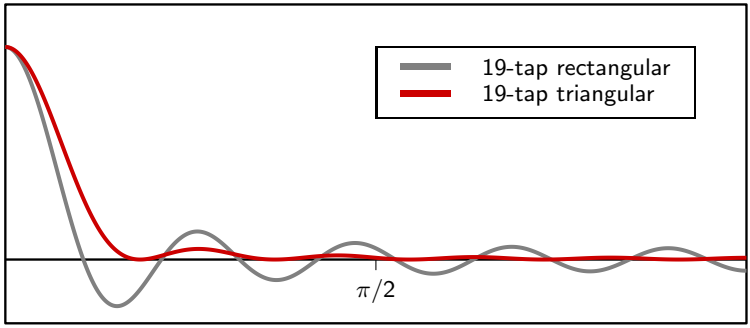
- narrow mainlobe so that transition is sharp
- small sidelobe so Gibbs error is small
- short window so FIR is efficient

very conflicting requirements!

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:

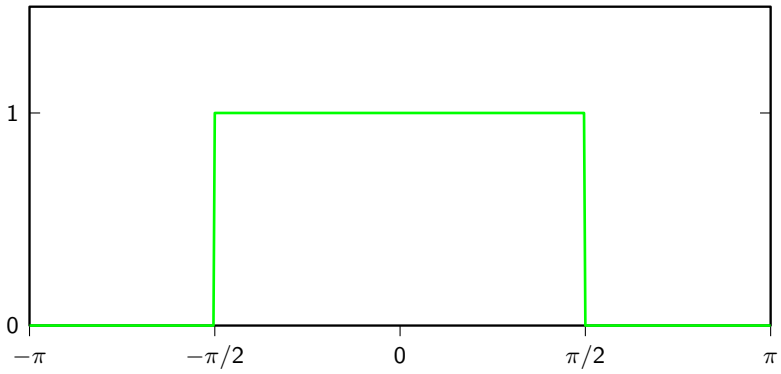
- draw desired frequency response $H(\omega)$
- take M equally-spaced values of the frequency response over the $[0, 2\pi]$ interval:

$$H_M[k] = H(\omega_k), \qquad \omega_k = (2\pi/M)k, \quad k = 0, 1, \ldots, M-1$$

- compute the inverse DFT: $\mathbf{h}_M = \text{IDFT}\{\mathbf{H}_M\}$
- use the impulse response

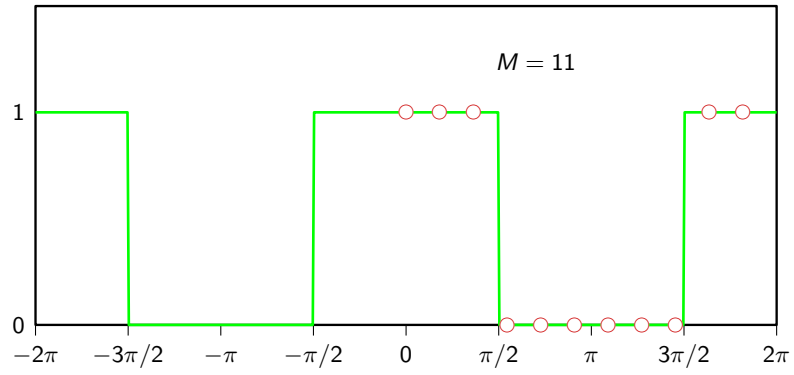
$$\bar{h}[n] = \begin{cases} h_M[n] & 0 \le n < M \\ 0 & \text{otherwise} \end{cases}$$

Frequency sampling: desired response

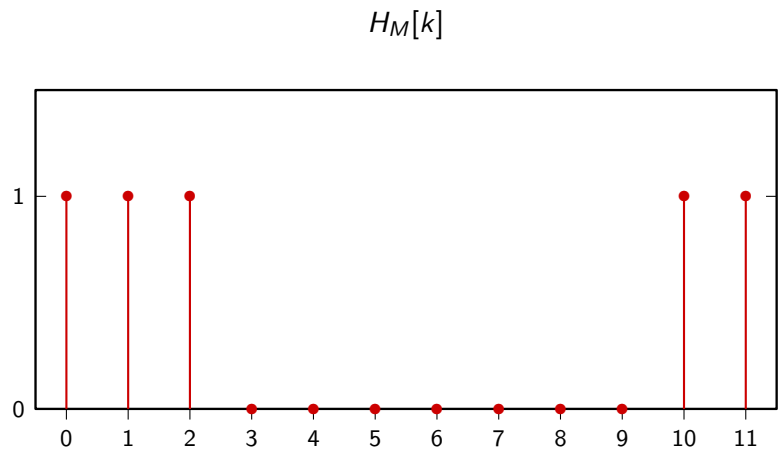


Frequency sampling: from DTFT to DFT

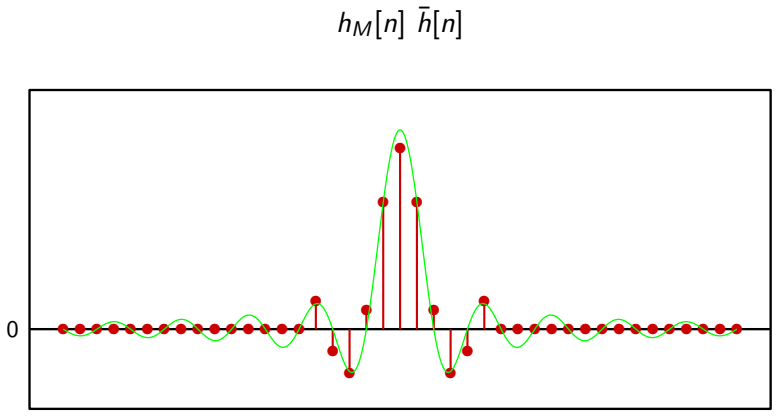
get M samples over the $[0, 2\pi]$ interval, so they are ready for the IDFT



Frequency sampling: DFT samples



Frequency sampling: impulse response from IDFT



Frequency sampling: what happens in the time domain

$$\mathbf{h}_M = \text{IDFT}\{\mathbf{H}_M\}$$

$$H_M[k] = H\left(\frac{2\pi}{M}k\right), \quad k = 0, 1, \ldots, M-1$$

Frequency sampling: what happens in the time domain

$$
\begin{aligned}
h_M[n] &= \frac{1}{M} \sum_{k=0}^{M-1} H_M[k]\, e^{j\frac{2\pi}{M}nk} \\
&= \frac{1}{M} \sum_{k=0}^{M-1} H\left(\frac{2\pi}{M}k\right)\, e^{j\frac{2\pi}{M}nk} \\
&= \frac{1}{M} \sum_{k=0}^{M-1} \left(\sum_{m=-\infty}^{\infty} h[m] e^{-j\frac{2\pi}{M}k} \right)\, e^{j\frac{2\pi}{M}nk} \\
&= \sum_{m=-\infty}^{\infty} h[m]\, \frac{1}{M} \sum_{k=0}^{M-1} e^{-j\frac{2\pi}{M}(m-n)k}
\end{aligned}
$$

a familiar result

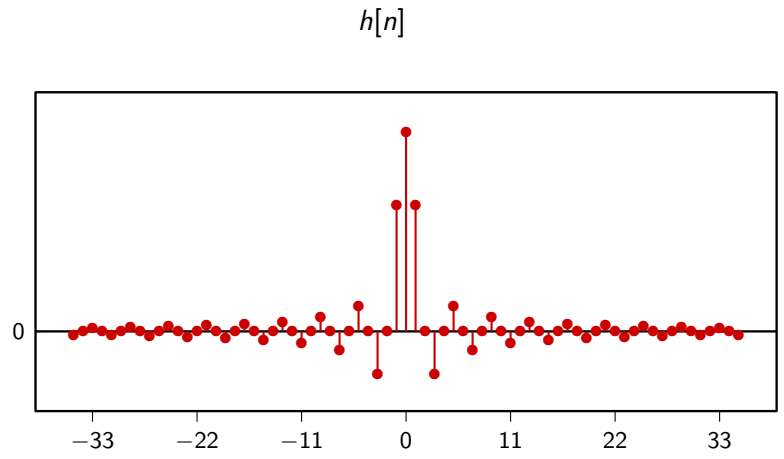
$$\sum_{k=0}^{M-1} e^{-j\frac{2\pi}{M}(m-n)k} = \begin{cases} M & \text{if } m-n \text{ multiple of M} \\ 0 & \text{otherwise} \end{cases}$$

Frequency sampling: what happens in the time domain

$$h_M[n] = \sum_{m=-\infty}^{\infty} h[m]\ \delta[(m-n) \mod M]$$
$$= \sum_{m=-\infty}^{\infty} h[n+mM]$$

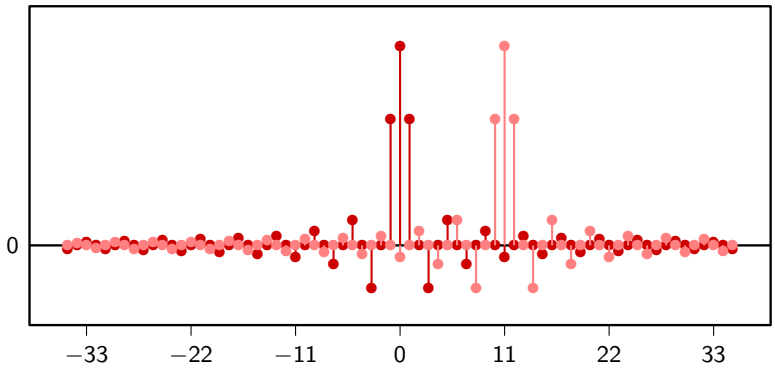
sampling in the frequency domain results in periodization in the time domain

Frequency sampling: impulse response from IDFT



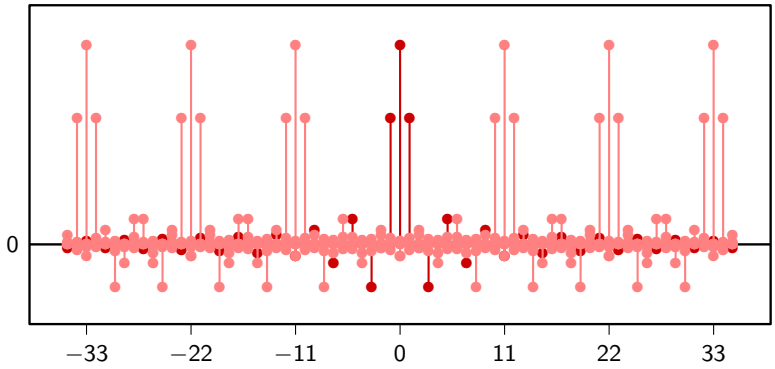
Frequency sampling: impulse response from IDFT

$h[n], h[n - M]$



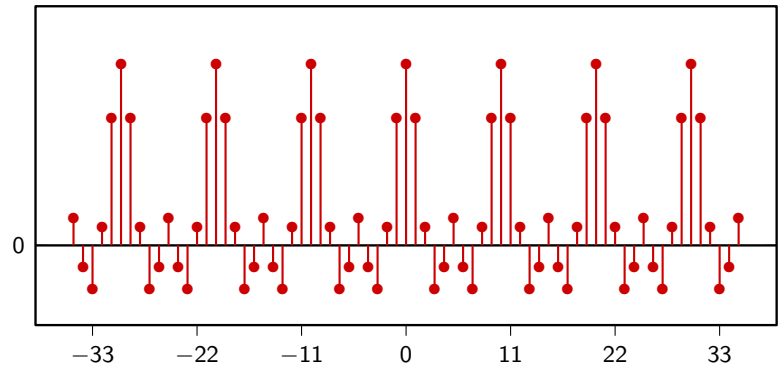
Frequency sampling: impulse response from IDFT

$$\ldots, h[n+2M], h[n+M], h[n], h[n-M], h[n-2M], \ldots$$

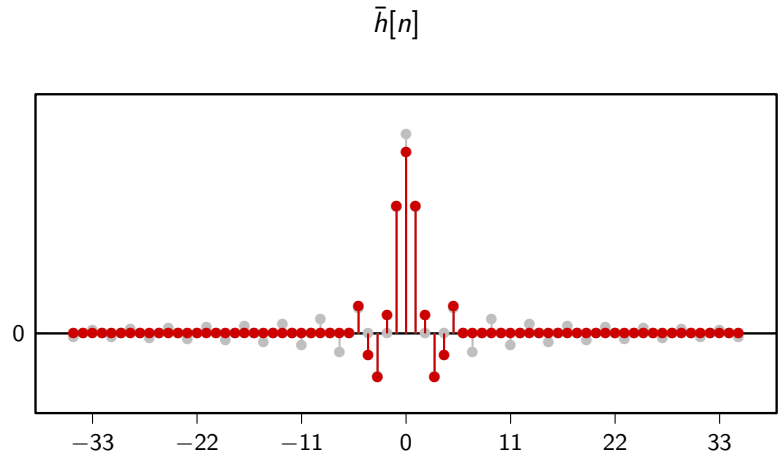


Frequency sampling: impulse response from IDFT

$$h_M[n] = \sum_{m=-\infty}^{\infty} h[n + mM]$$



Frequency sampling: impulse response from IDFT



Frequency sampling: what happens in the frequency domain

what is the frequency response $\bar{\mathbf{H}}$?

- $\mathbf{h}_M$: M-point inverse DFT of frequency samples $\mathbf{H}_M$ with $H_M[k] = H\left(\frac{2\pi}{M}k\right)$
- $\bar{\mathbf{h}}$: finite-support extension of $\mathbf{h}_M$
- DFT coefficients $\mathbf{H}_M$ are known
- use the DFT to DTFT result for finite-support sequences: Lagrangian interpolation

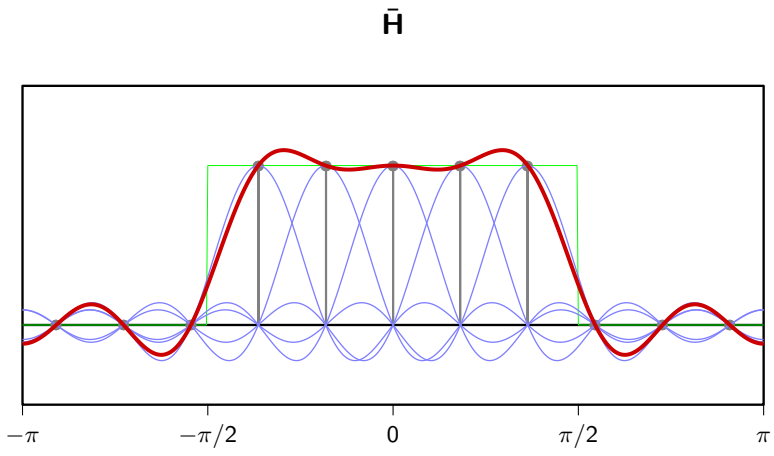
DTFT of finite-support signals

smooth interpolation of DFT values:

$$\bar{H}(\omega) = \sum_{k=0}^{M-1} H_M[k] \Lambda_M \left(\omega - \frac{2\pi}{M} k \right)$$

$$\Lambda_M(\omega) = \frac{1}{M} \frac{\sin\left(\frac{\omega}{2} M\right)}{\sin\left(\frac{\omega}{2}\right)} e^{-j\frac{\omega}{2}(M-1)}$$

Frequency sampling: frequency response



Frequency sampling: pros and cons

Pros:

- simple
- works with arbitrary frequency responses

Cons:

- can't control max error