COM-202: Signal Processing

Chapter 6.b: z-transform, filter structures and filter design

Overview

- realizable filters
- the z-transform and rational transfer functions
- BIBO stability
- pole-zero plots and block diagrams
- filter design: intuitive, from specs, IIR, FIR

the z-transform

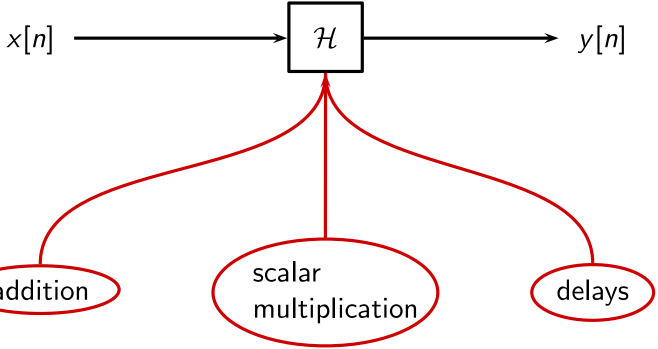
Overview:

- Constant-Coefficient Difference Equations
- The z-transform
- The transfer function
- Region of convergence

Realizable LTI systems

- ideal filters cannot be implemented
- what is the most general, realizable LTI system?
  - linearity: we can only use sums and multiplications
  - time-invariance: we can only multiply by constants
  - realizability: we can only use a finite amount of resources:
    - finite number of operations per output sample
    - finite amount of memory (i.e. we can only remember a finite number of past samples)
- causality required for real-time applications

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
- completely specified by $M+N+1$ scalar coefficients
- $a_0 = 1$ (otherwise renormalize)

Constant-Coefficient Difference Equation

Causal formulation:

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

- we can always make a CCDE causal
- CCDE is an *algorithm* to compute each output value

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

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

- what is the frequency response? The DTFT of the impulse response!
- but how do we compute the impulse response from the CCDE?

Apparently unrelated topic: Polynomial Multiplication

$$p(t) = 1 + 3t + 2t^2$$

$$q(t) = 2 + t - t^2 + 4t^3$$

$$\begin{aligned}(1 + 3t + 2t^2)(2 + t - t^2 + 4t^3) &= 2 + \quad t - \quad t^2 + 4t^3 \\ &\quad + 6t + 3t^2 - 3t^3 + 12t^4 \\ &\quad + 4t^2 + 2t^3 - 2t^4 \quad + 8t^5 \\ &= 2 + 7t + 6t^2 + 3t^3 + 10t^4 + 8t^5\end{aligned}$$

Apparently unrelated topic: Polynomial Multiplication

$$(1 + 3t + 2t^2)(2 + t - t^2 + 4t^3) = 2 + 7t + 6t^2 + 3t^3 + 10t^4 + 8t^5$$

define two sequences using the polynomial coefficients and convolve:

$$x_p[n] = \delta[n] + 3\delta[n-1] + 2\delta[n-2] = \ldots, 0, 0, 1, 3, 2, 0, 0, \ldots$$

$$x_q[n] = 2\delta[n] + \delta[n-1] - \delta[n-2] + 4\delta[n-3] = \ldots, 0, 0, 2, 1, -1, 4, 0, 0, \ldots$$

$$(x_p * x_q)[n] = 2\delta[n] + 7\delta[n-1] + 6\delta[n-2] + 3\delta[n-3] + 10\delta[n-4] + 8\delta[n-5]$$

Polynomial multiplication

$$p(t) = p_0 + p_1 t + \ldots + p_M t^M$$

$$q(t) = q_0 + q_1 t + \ldots + q_N t^N$$

$$r(t) = p(t) \cdot q(t) = \sum_{n=0}^{M+N} r_n t^n$$

$$r_n = \sum_{k=\max\{0,n-N\}}^{\min\{n,M\}} p_k q_{n-k}, \quad 0 \leq n \leq M + N$$

Polynomial multiplication and convolution are the same

if we assume $p_n = 0$ for $n \notin [0, P]$ and $q_n = 0$ for $n \notin [0, Q]$ the formula for the n-th coefficient of the product becomes

$$r_n = \sum_{k=-\infty}^{\infty} p_k q_{n-k}$$

which is identical to the convolution of two sequences:

$$x_r[n] = \sum_{k=-\infty}^{\infty} x_p[k] x_q[n-k]$$

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)

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:

$$\mathcal{Z}\{h[n] * x[n]\} = H(z)X(z)$$

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

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?

it turns out we don't need to!

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

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

- 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

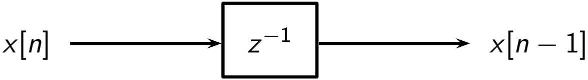
$$H(z) = (1-\lambda)\sum_{n=0}^{\infty} \lambda^n z^{-n} = \frac{(1-\lambda)}{1-\lambda z^{-1}}$$

- transfer function from CCDE:

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

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

Remember the delay block?



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

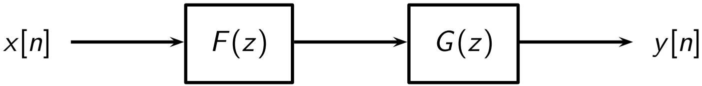
now the notation should make more sense!

The powerful formalism of transfer functions

Manipulating filters using transfer functions

- transfer functions are ratios of polynomials
- cascaded subsystems: product of transfer functions
- parallel subsystems: sum of transfer functions
- complex systems can be analyzed using simple algebra

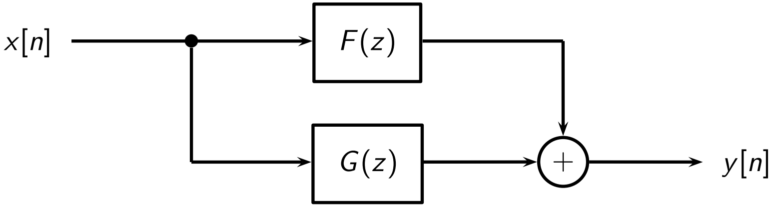
Cascade of filters



transfer function for the cascade:

$$H(z) = F(z)G(z)$$

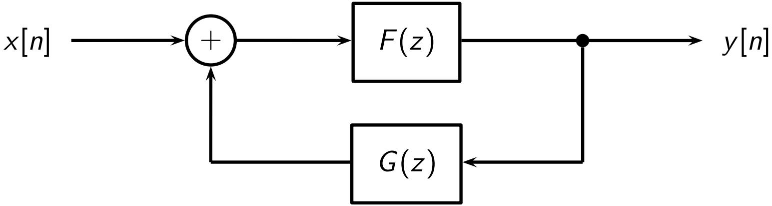
Filters in parallel



transfer function for the cascade:

$$H(z) = F(z) + G(z)$$

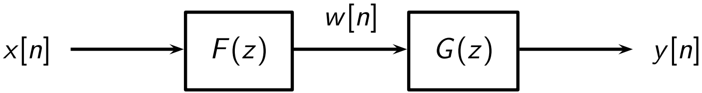
Filters in feedback configuration



transfer function for the cascade:

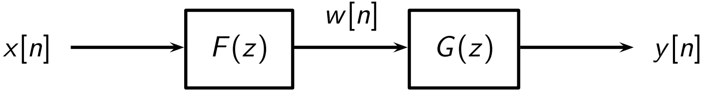
$$H(z) = \frac{F(z)}{1 - F(z)G(z)}$$

Example: CCDE of cascade



- CCDE for $\mathcal{F}$: $w[n] = aw[n-1] + x[n]$
- CCDE for $\mathcal{G}$: $y[n] = by[n-1] + cw[n] + dw[n-1]$
- CCDE for the cascade?

Example: CCDE of cascade



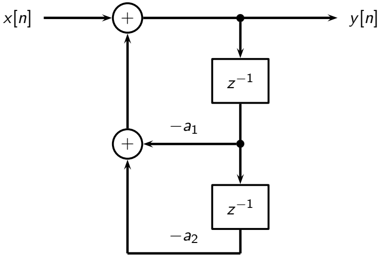
- $F(z) = 1/(1 - az^{-1})$
- $G(z) = (c + dz^{-1})/(1 - bz^{-1})$
- $H(z) = F(z)G(z) = \dfrac{c + dz^{-1}}{1 - (a+b)z^{-1} + abz^{-2}}$
- CCDE for the cascade:

$$y[n] = (a+b)y[n-1] - ab\,y[n-2] + c\,x[n] + d\,x[n-1]$$

Example: impulse response of second-order IIR

$$y[n] = a_1 y[n-1] + a_2 y[n-2] + x[n]$$

$$Y(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}}$$



Example: impulse response of second-order IIR

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

- we can factor the denominator:

$$H(z) = \frac{1}{(1 - p_0 z^{-1})(1 - p_1 z^{-1})}$$

- and then use partial fraction decomposition:

$$H(z) = \frac{c_0}{1 - p_0 z^{-1}} + \frac{c_1}{1 - p_1 z^{-1}}$$

$$c_i = \frac{p_i}{p_0 - p_1}, \quad i = 0, 1$$

We know the impulse response of a first-order IIR

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

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

$$h_\lambda[n] = \lambda^n \, u[n]$$

Example: impulse response of second-order IIR

- second order as a cascade of first-order filters

$$H(z) = \frac{1}{1 - p_0 z^{-1}} \, \frac{1}{1 - p_1 z^{-1}} = H_{p_0}(z) H_{p_1}(z)$$

- as per the convolution theorem, $\mathbf{h} = \mathbf{h}_{p_0} * \mathbf{h}_{p_1}$

$$\begin{aligned} h[n] &= \sum_{k=-\infty}^{\infty} p_0^k u[k] p_1^{n-k} u[n-k] \\ &= \sum_{k=0}^{n} p_0^k \, p_1^{n-k} \\ &= p_1^n \sum_{k=0}^{n} (p_0/p_1)^k = \begin{cases} \dfrac{p_0^{n+1} - p_1^{n+1}}{p_0 - p_1} & p_0 \neq p_1 \\ (n+1) p_0^n & p_0 = p_1 \end{cases} \end{aligned}$$

Example: impulse response of second-order IIR

- if $p_0 \neq p_1$ we can use partial fraction decomposition
- second order as a parallel structure:

$$H(z) = \frac{1}{p_0 - p_1} \left[\frac{p_0}{1 - p_0 z^{-1}} + \frac{p_1}{1 - p_1 z^{-1}} \right]$$

- each subsystem is independent

$$\mathbf{h} = (p_0\,\mathbf{h}_{p_0} + p_1\,\mathbf{h}_{p_1})/(p_0 - p_1)$$

$$h[n] = \frac{1}{p_0 - p_1} \left[p_0\, p_0^k u[k] + p_1\, p_1^k u[k] \right]$$

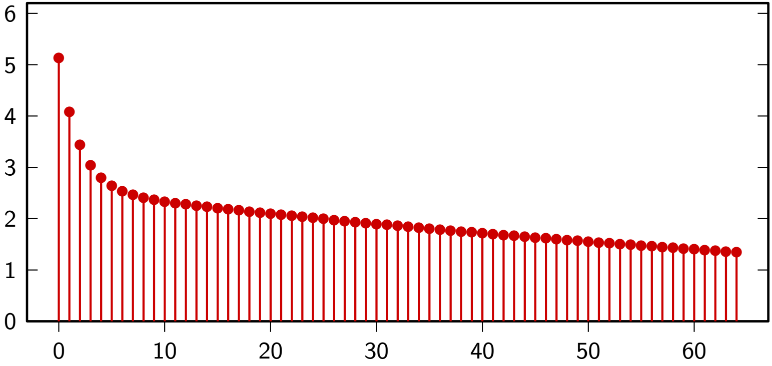
$$= \frac{p_0^{n+1} - p_1^{n+1}}{p_0 - p_1}$$

Impulse response of second-order IIR

1. $p_{0,1} = \lambda_{0,1} \in \mathbb{R}, \lambda_0 \neq \lambda_1$
2. $p_{0,1} = \lambda \in \mathbb{R}$
3. $p_0 = \rho\, e^{j\varphi} \in \mathbb{C}, p_1 = p_0^* = \rho\, e^{-j\varphi}$

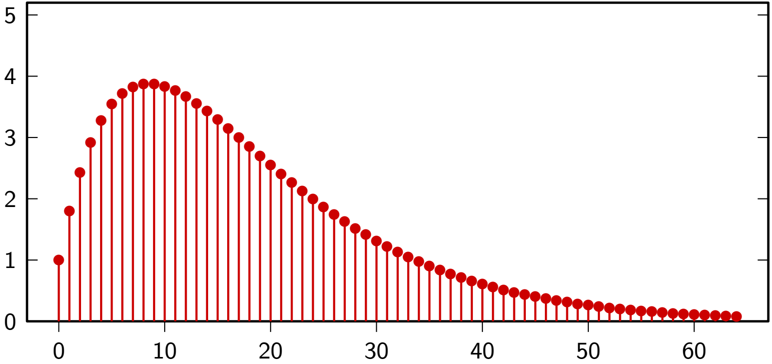
Second-order IIR, distinct real-valued roots

$$H(z) = [1 - 1.59z^{-1} + 0.594z^{-2}]^{-1}, \quad h[n] = \frac{\lambda_0^{n+1} - \lambda_1^{n+1}}{\lambda_0 - \lambda_1}, \quad \lambda_0 = 0.99, \lambda_1 = 0.6$$



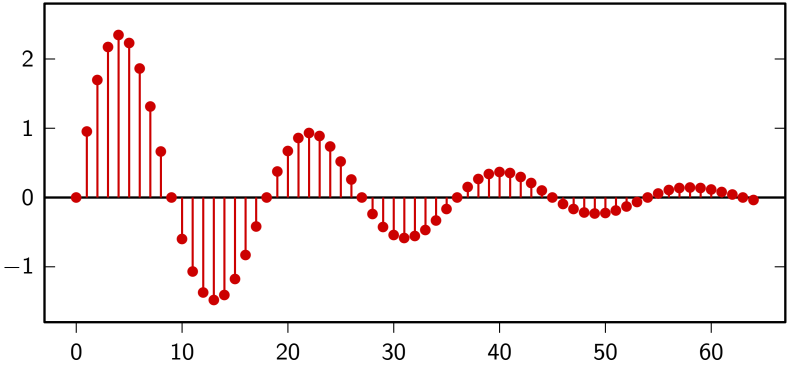
Second-order IIR, double real-valued root

$$H(z) = [1 - 1.8z^{-1} + 0.81z^{-2}]^{-1}, \quad h[n] = (n+1)\lambda^n, \quad \lambda = 0.9$$



Second-order IIR, complex-conjugate roots

$$H(z) = [1 - 0.8927z^{-1} + 0.9025z^{-2}]^{-1}, \quad h[n] = \frac{\rho^n}{\sin\varphi}\sin((n+1)\varphi), \quad \rho = 0.95, \varphi = \pi/9$$



region of convergence

The region of convergence

- the z-transform of a sequence is a power series
- the series may not converge for all values of z
- we can only use the z-transform when the series converge
- we need to find the Region of Convergence (ROC)

Finding the region of convergence

The ROC is defined by the absolute convergence of the power series:

$$z \in \mathrm{ROC}\{X(z)\} \iff \sum_{n=-\infty}^{\infty} \left| x[n] z^{-n} \right| < \infty$$

How can we determine the ROC?

- ROC depends on the values of **x**
- 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 $z = ae^{j\theta}$:

$$\sum_{n=-\infty}^{\infty} \left|x[n]z^{-n}\right| < \infty \iff \sum_{n=-\infty}^{\infty} |x[n]|\,|a^{-n}| < \infty$$

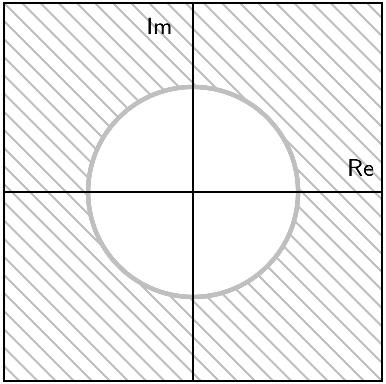
Region of convergence (ROC)

observation #3:

for causal sequences, the ROC extends from a circle to infinity:
assume $z_0 \in$ ROC and $|z_1| > |z_0|$:

$$\sum_{n=0}^{\infty} \left| x[n]\, z_1^{-n} \right| = \sum_{n=0}^{\infty} \frac{|x[n]|}{|z_1^n|} \leq \sum_{n=0}^{\infty} \frac{|x[n]|}{|z_0^n|} \leq \infty$$

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!

ROC for causal systems

Consider the transfer function for an LTI system:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_M z^{-M}}{1 + a_1 z^{-1} + \ldots + a_N z^{-N}}$$

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

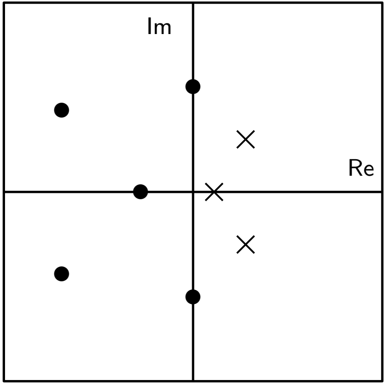
ROC for causal systems

We know:

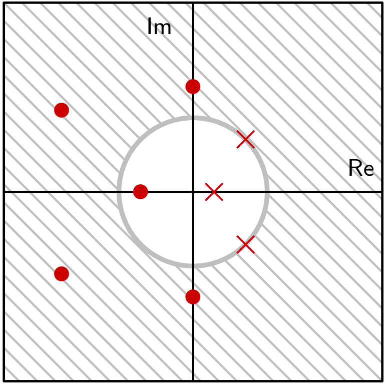
- ROC extends outwards
- ROC cannot include poles

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)

- $G(z) = \dfrac{B(z)}{A(z)}$ with $A(z)$ and $B(z)$ coprime
- since $B(z)$ has no poles, ROC of $G(z)$ same as ROC of $H(z) = 1/A(z)$
- 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:

$$\frac{1}{1-5z^{-1}+6z^{-2}} = \frac{1}{(1-2z^{-1})(1-3z^{-1})}$$
$$= \frac{c_0}{1-2z^{-1}} + \frac{c_1}{1-3z^{-1}}$$

$$c_0 + c_1 = 1$$
$$2c_0 + 3c_1 = 0$$

$$\frac{1}{1-5z^{-1}+6z^{-2}} = \frac{-2}{1-2z^{-1}} + \frac{3}{1-3z^{-1}}$$

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)

- same for multiple poles, just more tedious

$$\mathcal{Z}\{np^n\, u[n]\} = \frac{pz^{-1}}{(1 - pz^{-1})^2}, \qquad \text{ROC: } |z| > |p|$$

- all LTI impulse responses are linear combinations of weighed exponential sequences
- we could use an inverse z-transform to obtain $h[n]$

system stability

Overview:

- BIBO stability
- Stability criteria

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 ($\Rightarrow$)

Hypotheses: bounded input and absolutely summable impulse response

- $|x[n]| < M$
- $\sum_n |h[n]| = L < \infty$

Thesis: output is bounded

- $|y[n]| < \infty$

Proof:

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

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

$$\leq M \sum_{k=-\infty}^{\infty} |h[k]|$$

$$\leq ML$$

Proof ($\Leftarrow$)

Hypothesis: output is bounded for any bounded input

- $|x[n]| < \infty \Rightarrow |(\mathbf{x} * \mathbf{h})[n]| < \infty$

Thesis: impulse response is absolutely summable

- $\sum_n |h[n]| < \infty$

Proof (by contradiction):

- assume hypothesis fulfilled, yet $\sum_n |h[n]| = \infty$
- 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

$$(x * h)[0] = \sum_{k=-\infty}^{\infty} h[k]x[-k] = \sum_{k=-\infty}^{\infty} |h[k]| = \infty$$

The good news

FIR filters are always stable

Checking the stability of IIRs – Example

Let's check the Leaky Integrator:

$$\sum_{n=-\infty}^{\infty} |h[n]| = |1-\lambda| \sum_{n=0}^{\infty} |\lambda|^n$$

$$= \lim_{n\to\infty} |1-\lambda| \frac{1-|\lambda|^{n+1}}{1-|\lambda|}$$

$$< \infty \quad \text{for } |\lambda| < 1$$

stability is guaranteed for $|\lambda| < 1$

Checking the stability of IIRs – General case

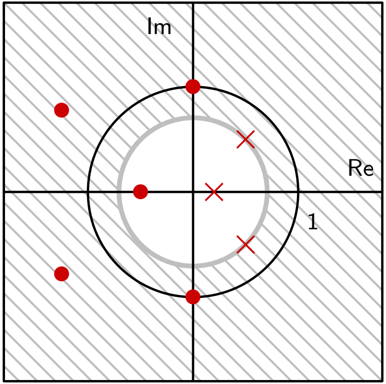
stability of a filter with impulse response $h[n]$ and transfer function $H(z)$:

$$1 \in \text{ROC} \iff H(z) \text{ converges absolutely in } z = 1 \iff \sum_{n=-\infty}^{\infty} |h[n]| < \infty$$

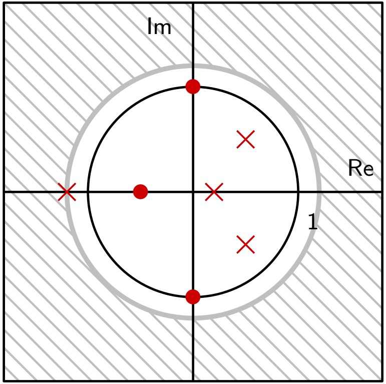
an LTI system is stable if and only if the ROC includes the unit circle

a *causal* system is stable if and only if all poles are inside the unit circle

Stable causal system



Unstable causal system



Common confusion...

$$y[n] = 2y[n-1] + x[n] \qquad \text{(obviously unstable)}$$

apply z-transform as a formal operator:

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

$H(1) = -1 < \infty$, so is the system 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

$$\sum_{n=0}^{\infty} a^n z^{-n} = \lim_{N \to \infty} \frac{1 - (a/z)^N}{1 - (a/z)} = \begin{cases} \dfrac{1}{1 - az^{-1}} & \text{if } |z| > |a| \\ \infty & \text{otherwise} \end{cases}$$

In other words...

the function $\dfrac{1}{1 - az^{-1}}$ 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

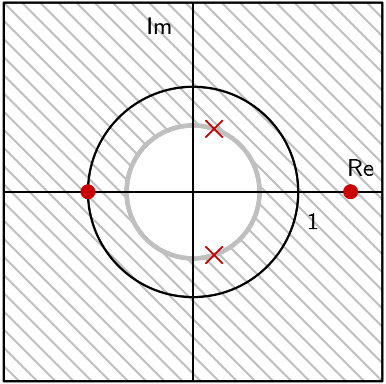
Understanding a pole-zero plot

The effects of poles and zeros

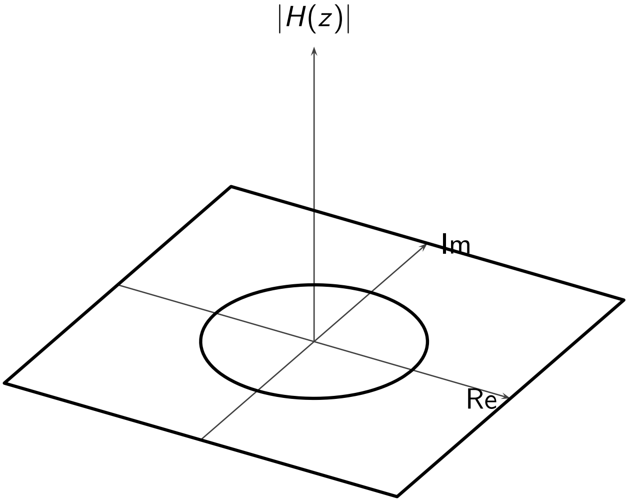
Looking at the magnitude of the transfer function with the "circus tent" method:

- z-transform magnitude 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 height around the unit circle

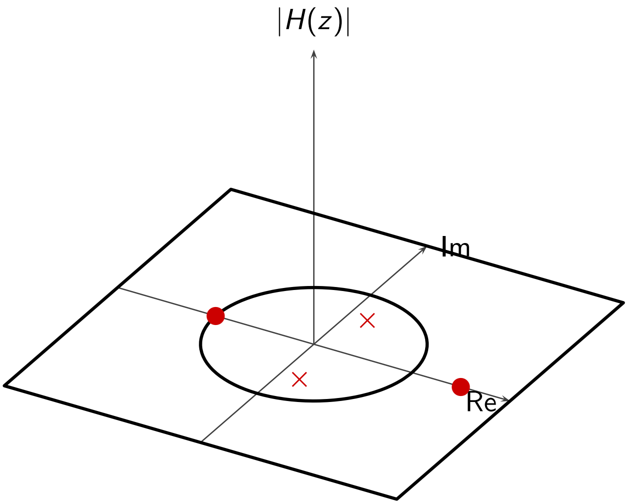
Example: pole-zero plot



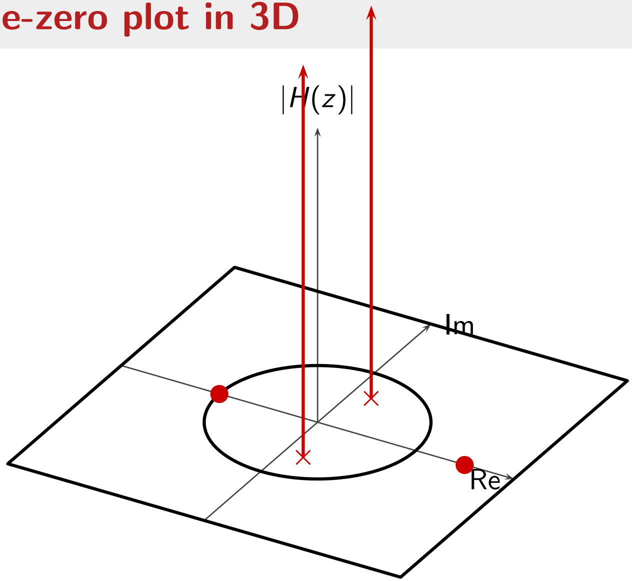
Example: pole-zero plot in 3D



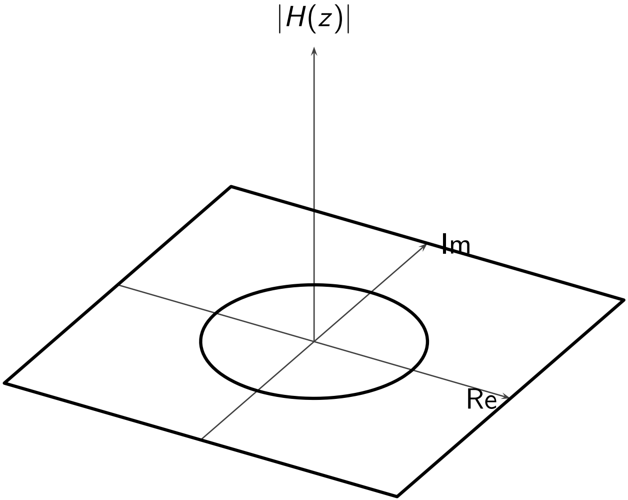
Example: pole-zero plot in 3D



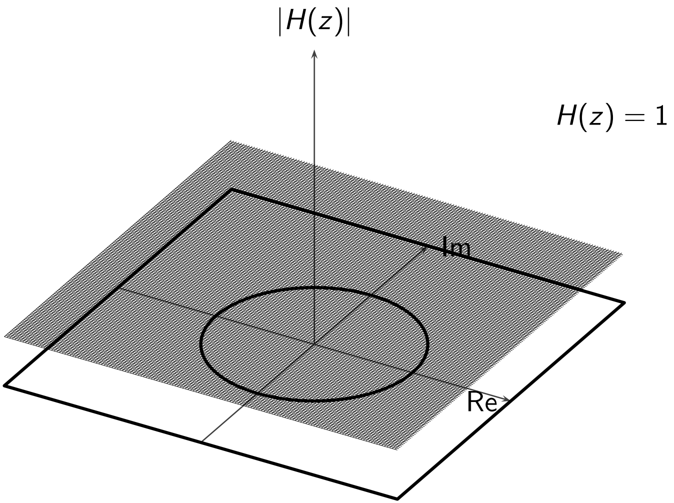
Example: pole-zero plot in 3D



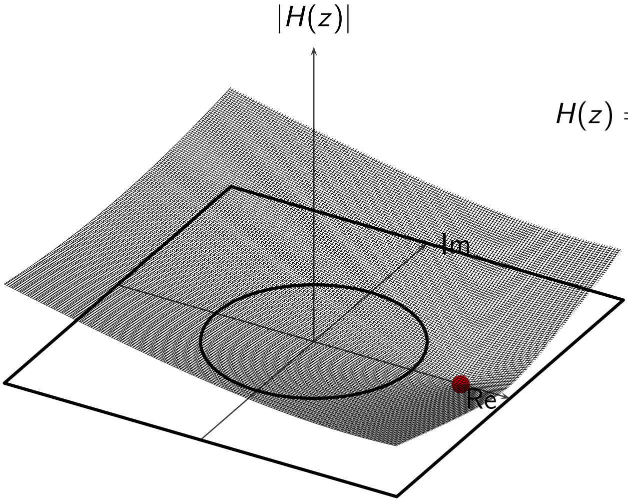
Example: sketching $|H(z)|$



Example: sketching $|H(z)|$

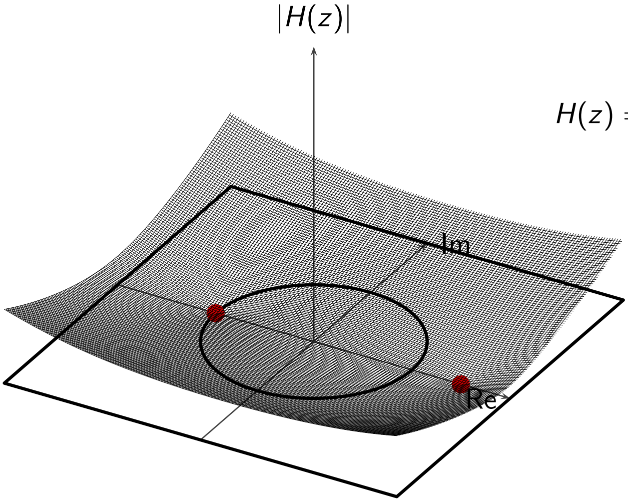


Example: sketching $|H(z)|$



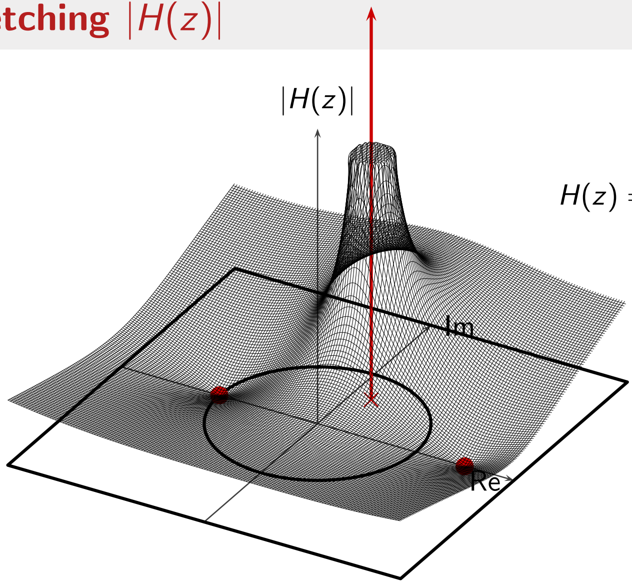
$$H(z) = (1 - z_0\, z^{-1})$$

Example: sketching $|H(z)|$



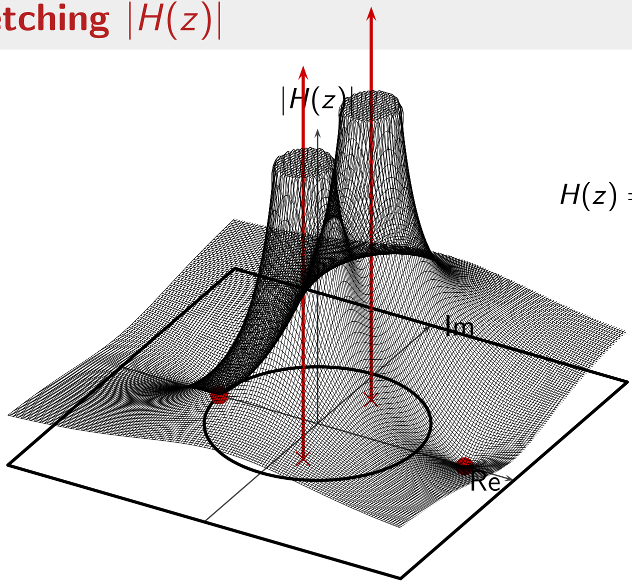
$$H(z) = (1 - z_0\, z^{-1})(1 + z_1\, z^{-1})$$

Example: sketching $|H(z)|$



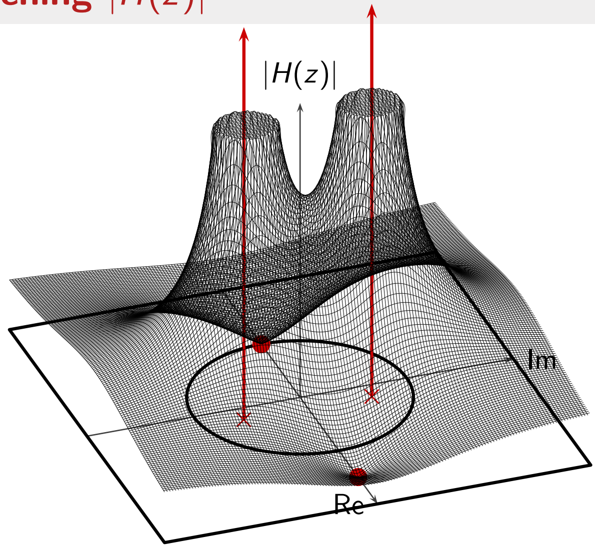
$$H(z) = \frac{(1 - z_0\, z^{-1})(1 + z_1\, z^{-1})}{(1 - p_0\, z^{-1})}$$

Example: sketching $|H(z)|$

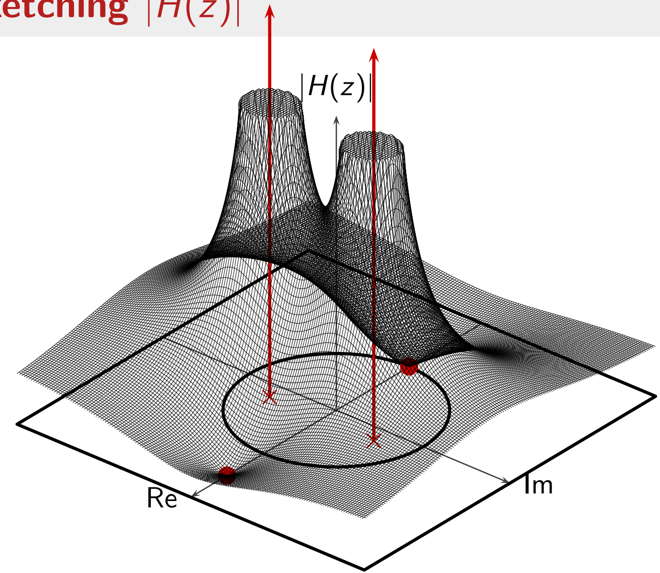


$$H(z) = \frac{(1 - z_0\, z^{-1})(1 + z_1\, z^{-1})}{(1 - p_0\, z^{-1})(1 - p_0^*\, z^{-1})}$$

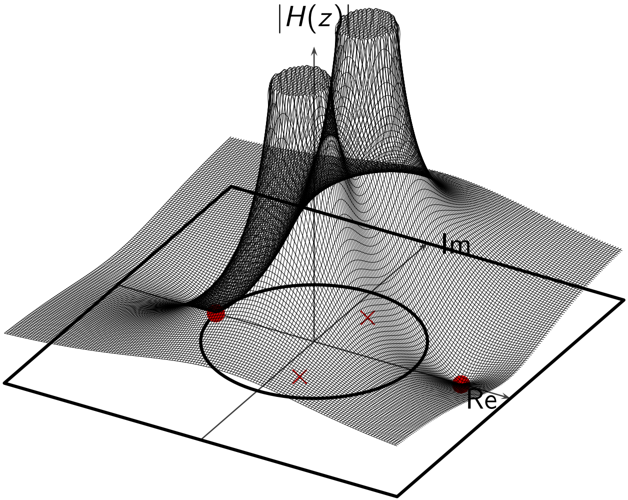
Example: sketching $|H(z)|$



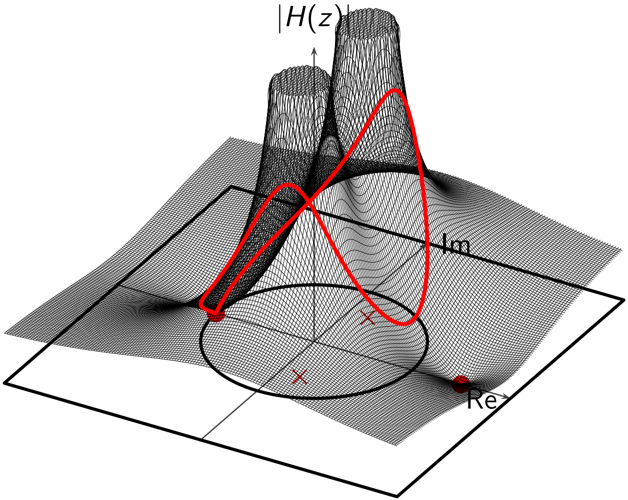
Example: sketching $|H(z)|$



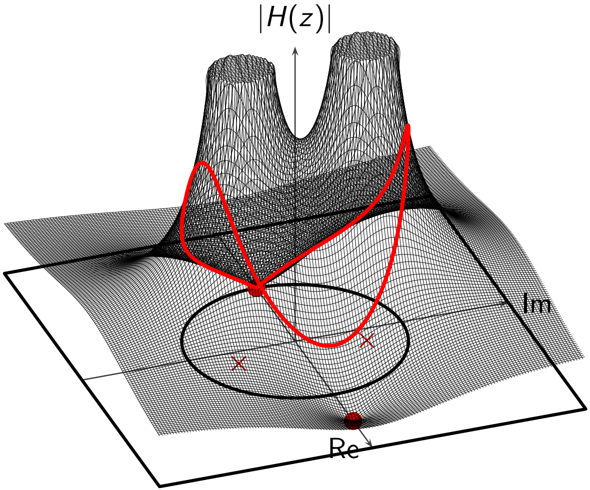
Example: sketching $|H(z)|$



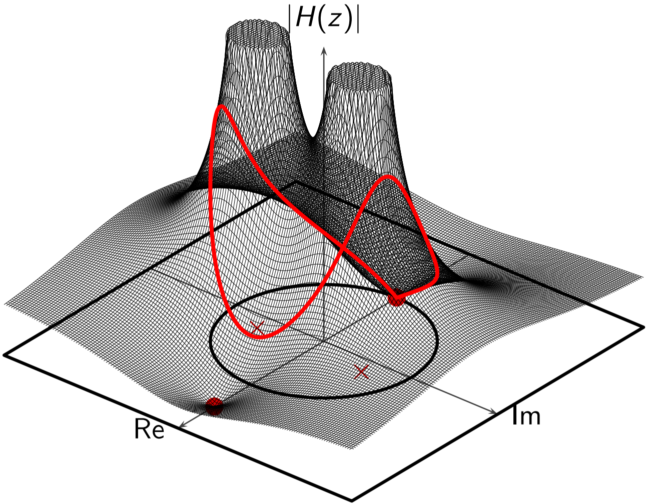
Example: sketching $|H(z)|$



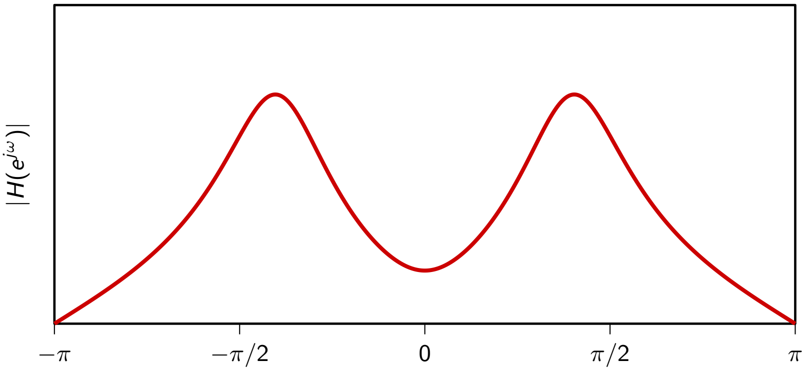
Example: sketching $|H(z)|$



Example: sketching $|H(z)|$



Magnitude of the frequency response



block diagrams

Overview:

- Algorithms for CCDE's
- Block diagram
- Real-time processing

An old friend

```
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
```

Testing the code

```
>>> from leaky import LI
>>> li = LI(0.95)
>>> for x in [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]:
>>>     print(li.filt(x), end=' ')

 0.0, 0.0, 0.0, 0.0, 0.050000000000000, 0.047500000000000,
 0.045125000000000, 0.042868750000000, 0.040725312500000,
 0.038689046875000, 0.036754594531250 0
>>>
```

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

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

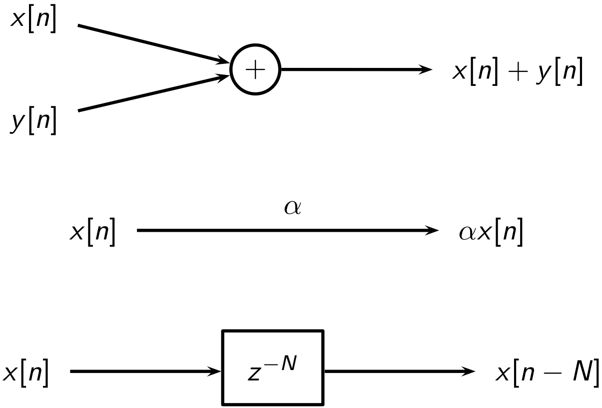
Another old friend

```
class MA:
    def __init__(self, M):
        self.buf = np.zeros(M-1)
        self.norm = 1.0 / M

    def filt(self, x):
        y = (x + np.sum(self.buf)) * self.norm
        self.buf = np.r_[x, self.buf[:-1]]
        return y
```

Key points

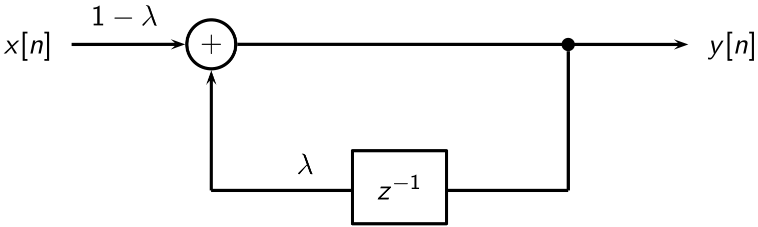
- we now need $M-1$ memory cells to store previous input values
- we need to initialize the storage before first use
- we need 1 multiplication and $M-1$ additions per output sample

We can abstract from the implementation



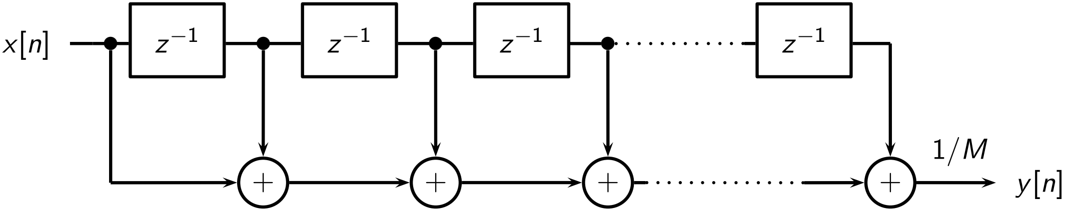
Leaky Integrator

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



Moving Average

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



The second-order section (aka "biquad")

$$y[n] + a_1 y[n-1] + a_2 y[n-2] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2]$$

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

Why are biquads important?

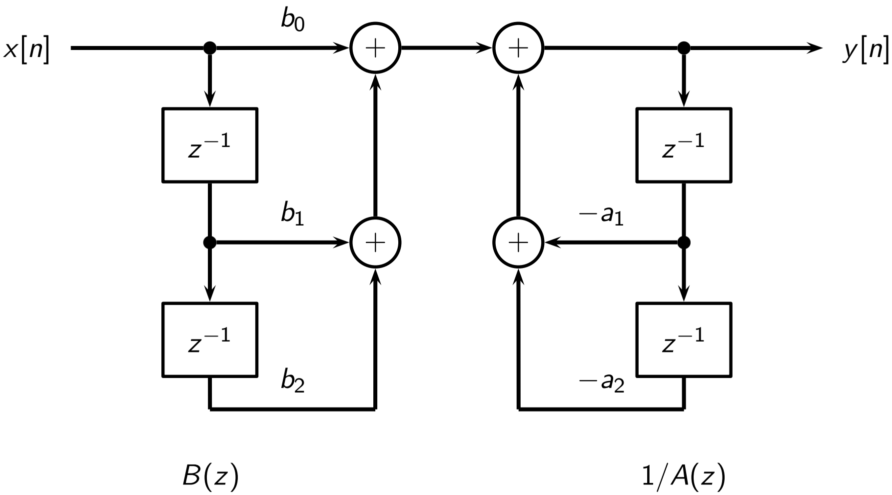
We can always factor a rational transfer function into a cascade of second-order sections:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_M z^{-M}}{1 + a_1 z^{-1} + \ldots + a_N z^{-N}}$$

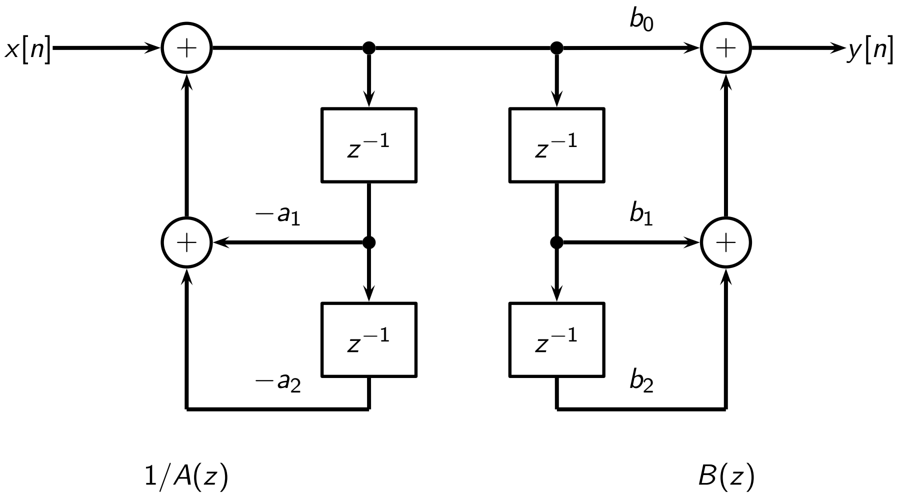
$$= \prod_{k=1}^{\lceil \max\{M,N\}/2 \rceil} \frac{b_{0,k} + b_{1,k} z^{-1} + b_{2,k} z^{-2}}{1 + a_{1,k} z^{-1} + a_{2,k} z^{-2}}$$

- if $H(z)$ has real-valued coefficients, so will all the biquad sections
- cascade implementation is numerically more robust
- a lot of useful filters can be implemented with a single biquad

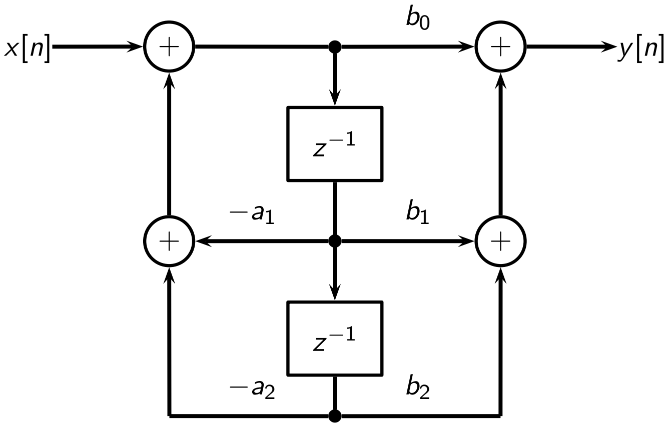
Second-order section, direct form I



Second-order section, direct form I, inverted order



Second-order section, direct form II



intuitive filter design

Simple, useful filters

- many signal processing problems can be solved using simple filters
- e.g. we have already derived simple lowpass filters "intuitively" (Moving Average, Leaky Integrator)
- with a low-order transfer function we can try to design filters by placing poles and zeros "by hand"

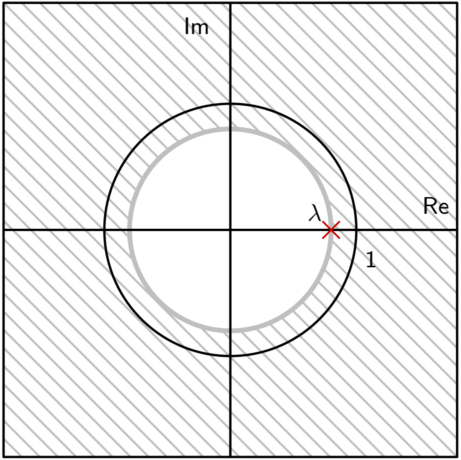
Simple lowpass

- 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

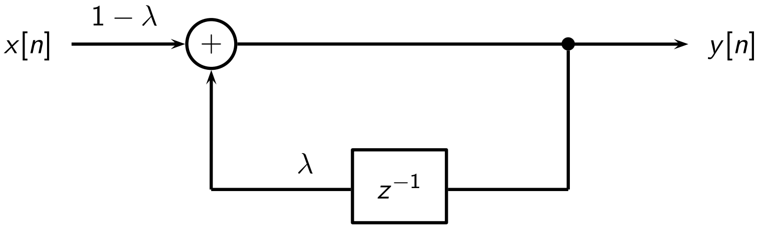
Leaky Integrator

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

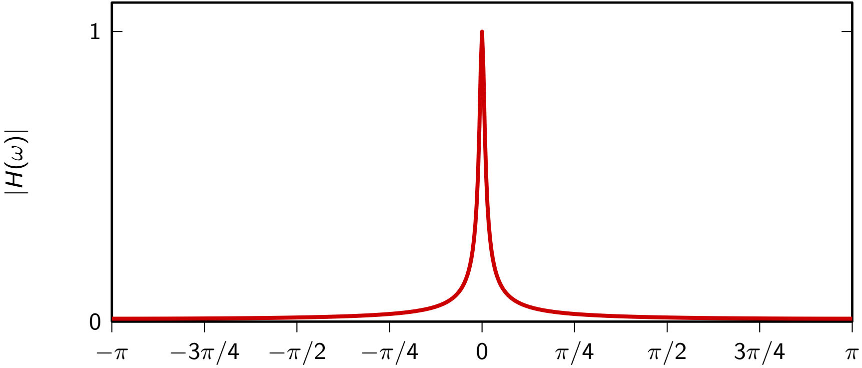
$$y[n] = (1-\lambda)x[n] + \lambda y[n-1]$$



Leaky Integrator, filter structure



Leaky Integrator, $\lambda = 0.98$



DC removal

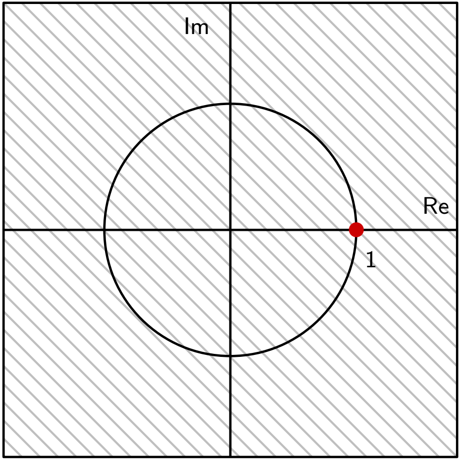
- a DC-balanced signal has zero mean: $\lim_{N\to\infty} \sum_{n=-N}^{N} x[n] = 0$

  i.e. there is no Direct Current component
- its DTFT value at zero is zero
- to remove the DC bias from a non zero-centered signal...
- ... we just need to kill the frequency component at $\omega = 0$

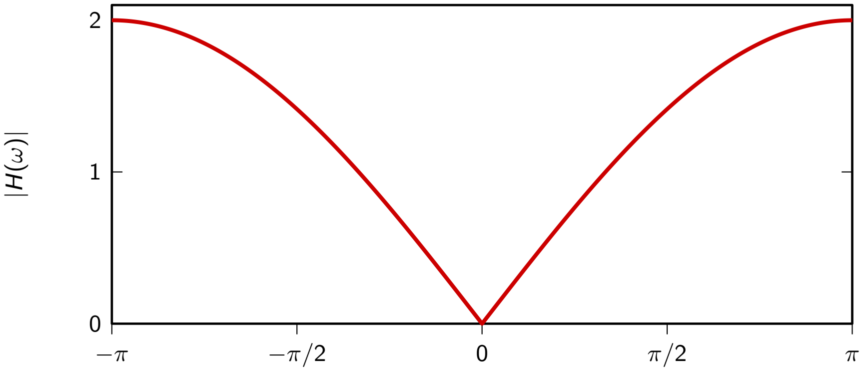
DC removal

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

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



DC notch



Problems with the simple DC notch

we only want to eliminate the DC component but

- too much attenuation around zero, we'd like the magnitude response to climb back up quickly around zero
- magnitude response at $\omega = \pm\pi$ is greater than one: amplification of high frequencies

solutions:

- add a pole to "push up" $H(z)$ (remember the circus tent)
- add a gain factor to make sure gain is at most one

DC removal, improved

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

- gain in $z = -1$ (i.e. $\omega = \pm\pi$):

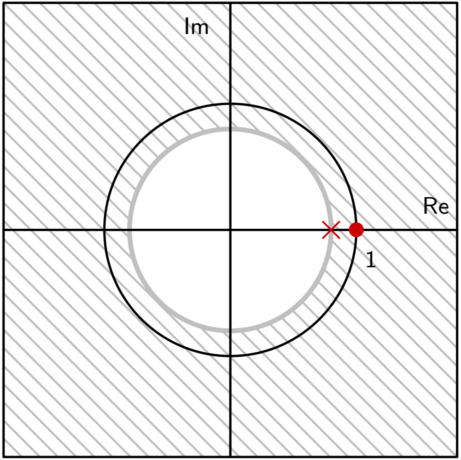
$$H(-1) = G\,\frac{2}{1 + \lambda}$$

- normalization factor: $G = \frac{1+\lambda}{2}$

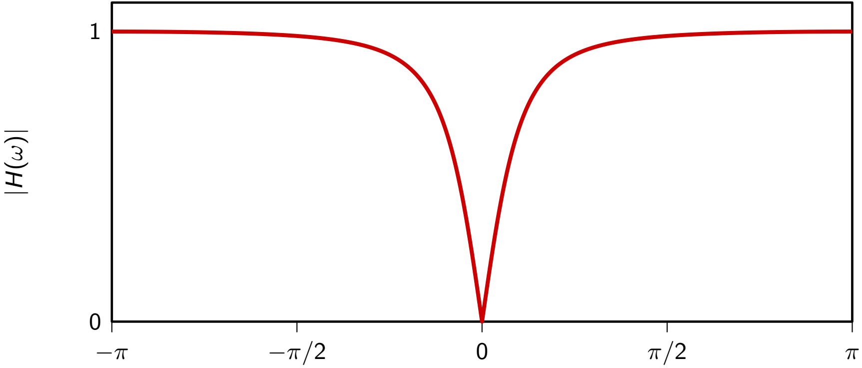
DC removal, improved

$$H(z) = \frac{1+\lambda}{2} \frac{1 - z^{-1}}{1 - \lambda z^{-1}}$$

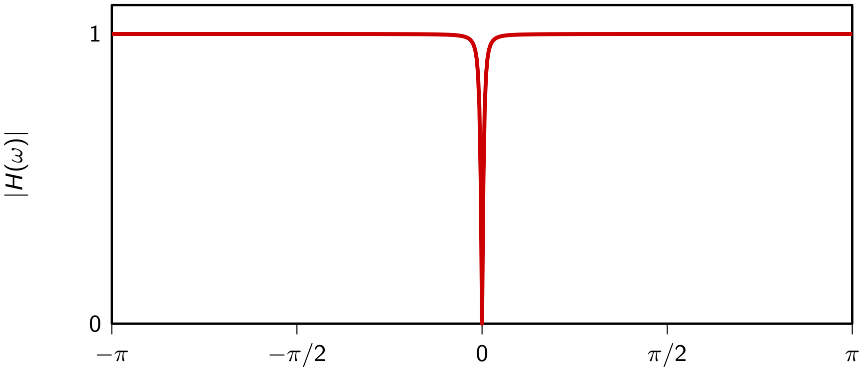
$$y[n] = \lambda y[n-1] + \frac{1+\lambda}{2}(x[n] - x[n-1])$$



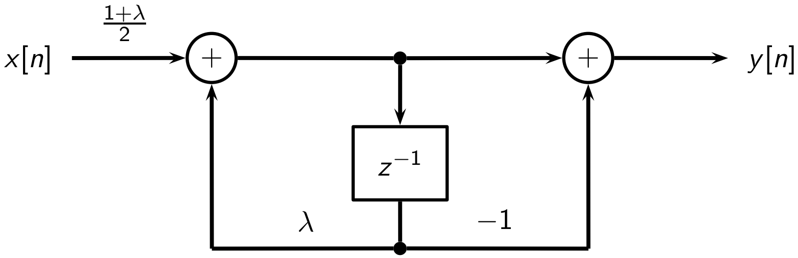
DC notch, $\lambda = 0.7$



DC notch, $\lambda = 0.98$



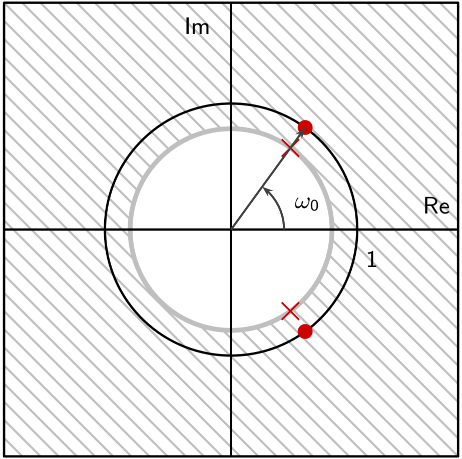
DC notch, filter structure



Hum removal

- similar to DC removal but we want to remove a specific frequency $\omega_0 > 0$
- very useful for audio equipment since amplifiers tend to pick up the hum from the AC power supply (50Hz in Europe and 60Hz in North America)
- idea: shift the pole-zero pair of the DC notch to ω_0
- but, to keep the filter real-valued, we need to add a conjugate pole-zero pair

Hum removal



Hum removal

$$H(z) = G\frac{(1 - e^{j\omega_0}z^{-1})(1 - e^{-j\omega_0}z^{-1})}{(1 - \lambda e^{j\omega_0}z^{-1})(1 - \lambda e^{-j\omega_0}z^{-1})}$$

$$G = \frac{(1+\lambda)^2}{4}$$

Hum removal: finding the gain

we want the gain at $\pm\pi$ to be unitary; the transfer function in -1 before normalization is

$$H_p(-1) = \frac{1+e^{j\omega_0}}{1+\lambda e^{j\omega_0}} \cdot \frac{1+e^{-j\omega_0}}{1+\lambda e^{-j\omega_0}} = r \cdot s$$

$$\begin{aligned}
r &= \frac{1+e^{j\omega_0}}{1+\lambda e^{j\omega_0}} = \frac{e^{j\omega_0/2}(e^{-j\omega_0/2}+e^{j\omega_0/2})}{e^{j\omega_0/2}(e^{-j\omega_0/2}+\lambda e^{j\omega_0/2})} \\
&= \frac{2\cos(\omega_0/2)}{\cos(\omega_0/2) - j\sin(\omega_0/2) + \lambda\cos(\omega_0/2) + j\lambda\sin(\omega_0/2)} \\
&= \frac{2}{(1+\lambda) - j(1-\lambda)\tan(\omega_0/2)}
\end{aligned}$$

Hum removal: finding the gain

- for $\lambda \approx 1$, $r \approx 2/(1+\lambda)$
- similarly,

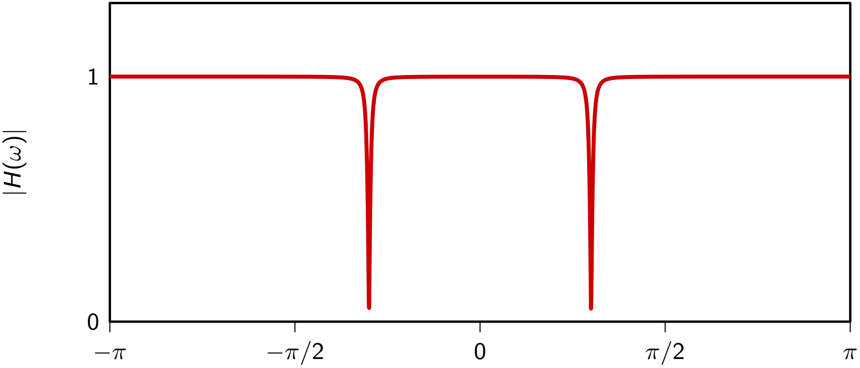
$$s = \frac{1 + e^{-j\omega_0}}{1 + \lambda e^{-j\omega_0}} = \frac{2}{(1+\lambda) + j(1-\lambda)\tan(\omega_0/2)}$$

- for $\lambda \approx 1$, $s \approx 2/(1+\lambda)$

$$|H_p(-1)| = |r||s| \approx \frac{4}{(1+\lambda)^2}$$

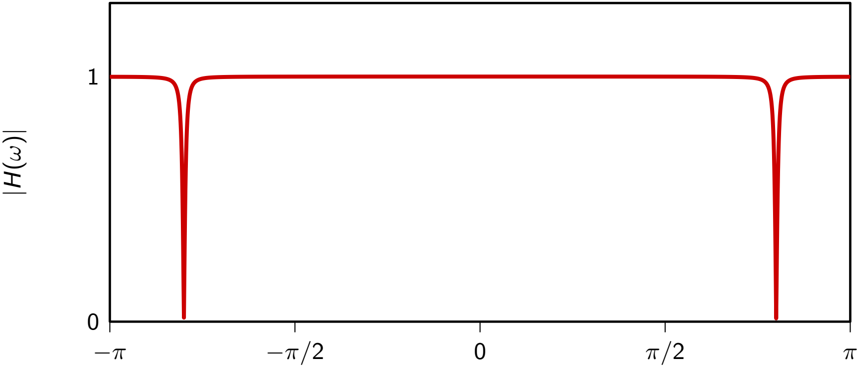
Hum removal

$$\omega_0 = 0.3\pi, \lambda = 0.95$$

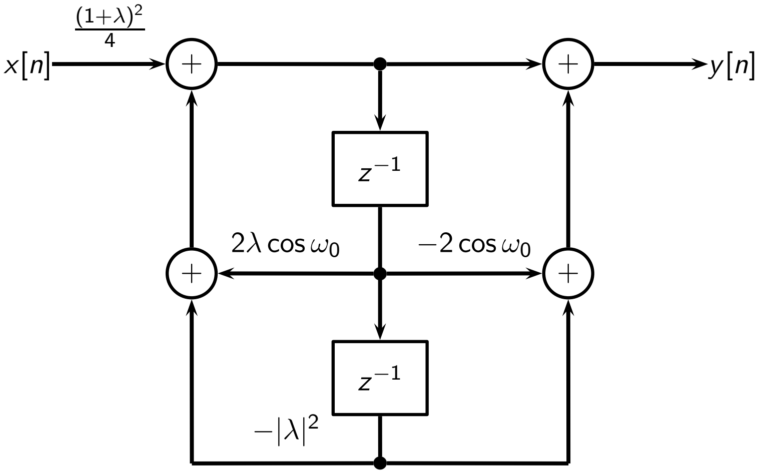


Hum removal

$$\omega_0 = 0.8\pi, \lambda = 0.99$$



Hum removal, filter structure



Tunable Resonator

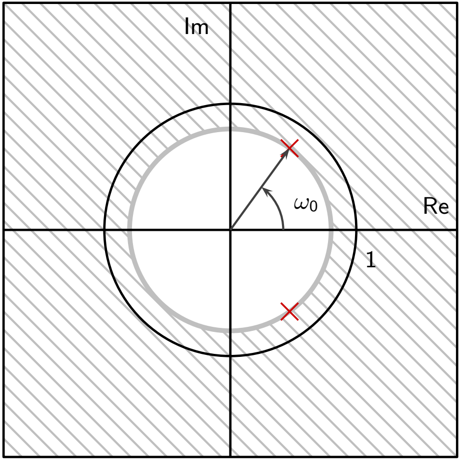
- a resonator is a narrow bandpass filter
- used to detect the presence of a sinusoidal component of a given frequency
- useful in communication systems and telephony (DTMF)
- idea: shift the passband of the Leaky Integrator
- again, to keep the filter real-valued, we need to use a pair of conjugate poles

Simple resonator

$$H_s(z) = \frac{1}{(1 - pz^{-1})(1 - p^* z^{-1})}$$

$$p = \lambda e^{j\omega_0}$$

$$y[n] = x[n] - a_1 y[n-1] - a_2 y[n-2]$$



Simple resonator

$$H_s(z) = \frac{1}{(1 - pz^{-1})(1 - p^* z^{-1})}, \qquad p = \lambda e^{j\omega_0}$$

$$= \frac{1}{1 - 2\Re\{p\}\, z^{-1} + |p|^2\, z^{-2}}$$

$$= \frac{1}{1 - 2\lambda \cos\omega_0\, z^{-1} + |\lambda|^2\, z^{-2}}$$

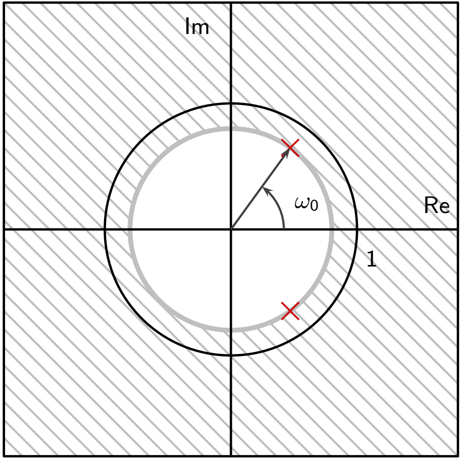
$$a_1 = -2\lambda \cos\omega_0$$

$$a_2 = |\lambda|^2$$

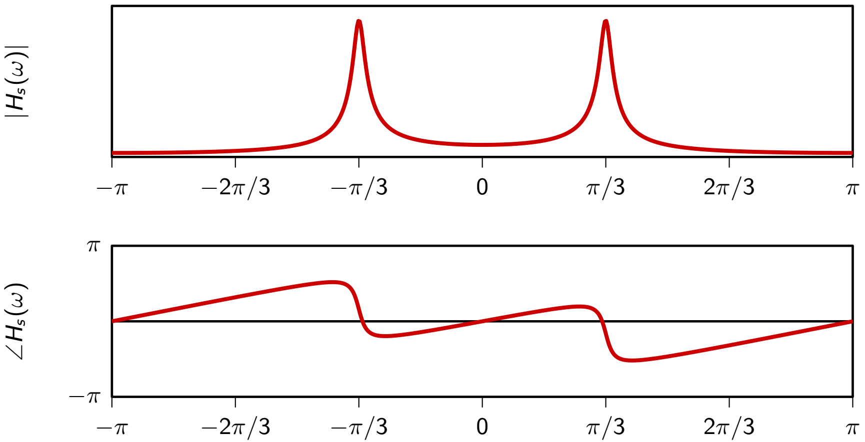
Simple resonator

$$H_s(z) = \frac{1}{1 - 2\lambda \cos \omega_0 \, z^{-1} + |\lambda|^2 \, z^{-2}}$$

$$y[n] = x[n] + 2\lambda \cos \omega_0 y[n-1] - |\lambda|^2 y[n-2]$$



Simple resonator, $\lambda = 0.95, \omega_0 = \pi/3$



Problems with the simple resonator

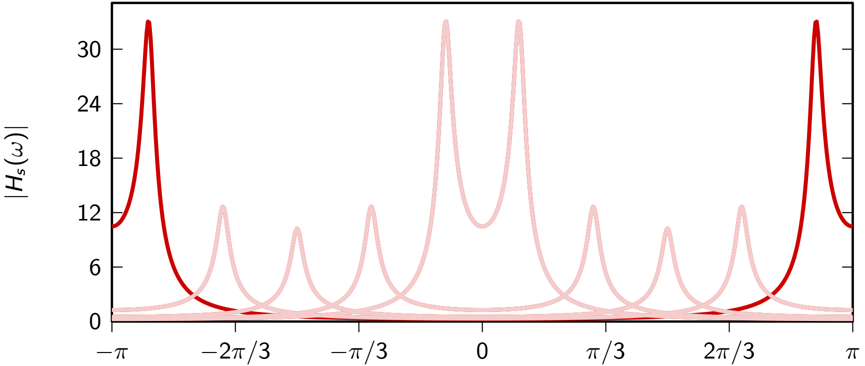
- the gain at the resonating frequency depends on ω_0:

$$|H_s(\omega_0)| = \left[|1 - \lambda|\,|1 - \lambda e^{-j2\omega_0}|\right]^{-1}$$

- we would like to have the same peak gain for all choices of ω_0
- also, we would like the gain to be zero for $\omega = 0, \pm\pi$ (bandpass)

Simple Resonator: varying peak gain

$$|H_s(\omega_0)| = \left[|1-\lambda|\,|1-\lambda e^{-j2\omega_0}|\right]^{-1}$$



Improved resonator

Idea: add a double zero in $\omega = 0$: $H_r(z) = (1 - z^{-2})H_s(z)$

- $(1 - z^{-2})$ makes the frequency response zero at $\omega = 0$ and $\omega = \pi$
- peak gain now:

$$|H_r(\omega_0)| = \frac{1}{|1-\lambda|}\,\frac{|1 - e^{-j2\omega_0}|}{|1 - \lambda e^{-j2\omega_0}|}$$

- with some algebra (like we did for the notch):

$$\frac{|1 - e^{-j2\omega_0}|}{|1 - \lambda e^{-j2\omega_0}|} = \frac{2}{|1 + \lambda - j(1-\lambda)\cot\omega_0|}$$

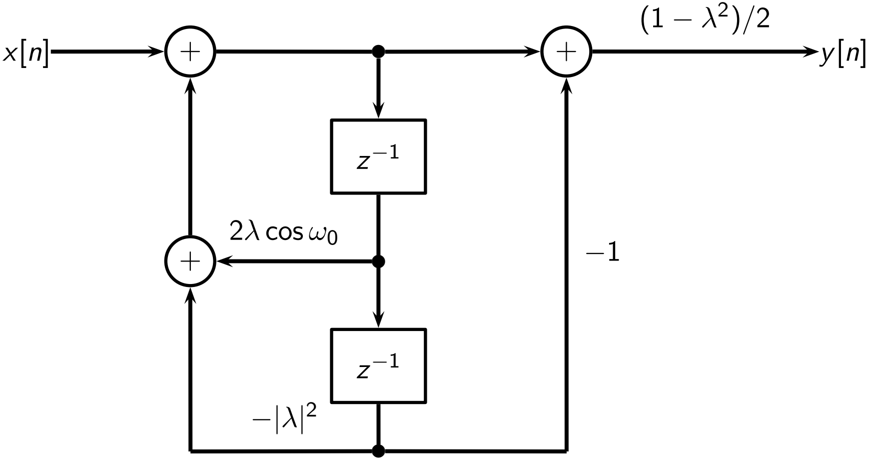
- for λ close to one, $|H_r(\omega_0)| \approx 2/(1-\lambda^2)$

Constant peak gain resonator

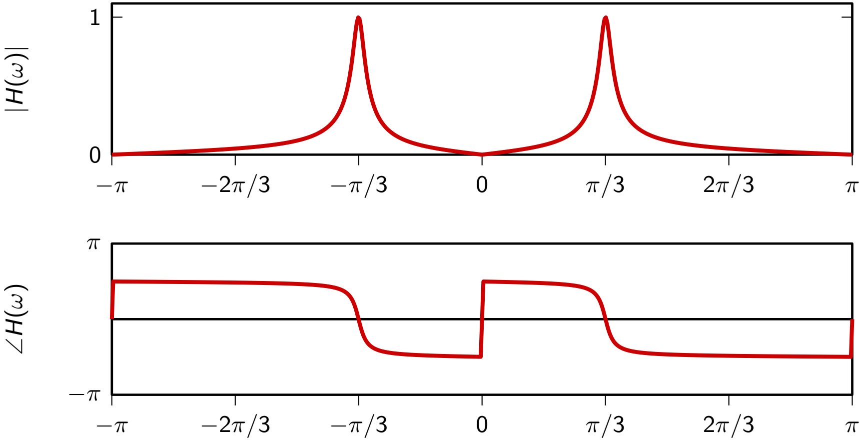
$$H(z) = \left(\frac{1-\lambda^2}{2}\right) \frac{1 - z^{-2}}{1 - 2\lambda \cos\omega_0\, z^{-1} + |\lambda|^2\, z^{-2}}$$

- negligible extra cost
- unit gain at peak
- DC rejection

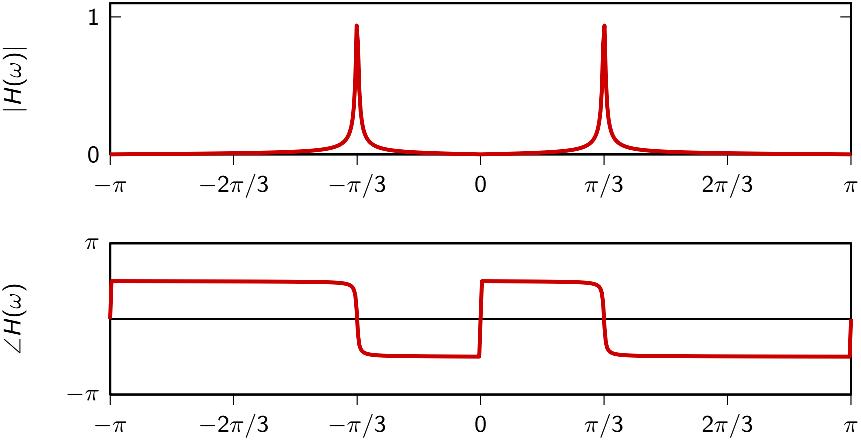
Resonator, filter structure



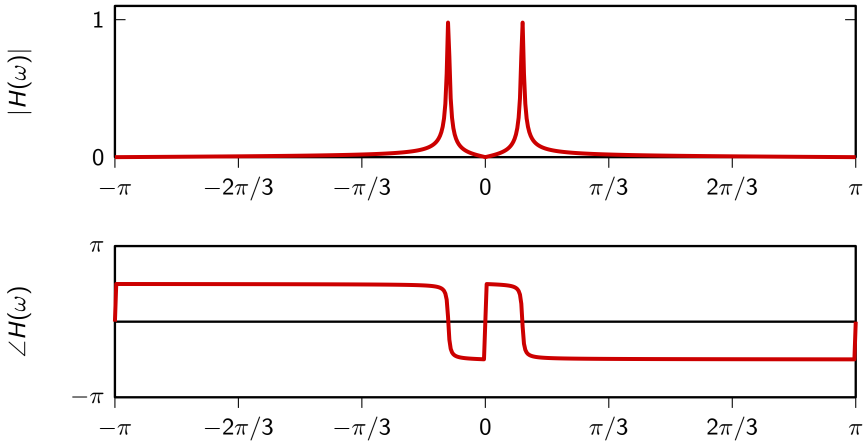
Resonator, $\lambda = 0.95, \omega_0 = \pi/3$



Resonator, $\lambda = 0.99, \omega_0 = \pi/3$



Resonator, $\lambda = 0.99, \omega_0 = \pi/10$



We need more systematic methods for filter design

- “intuitive” filter design can only take us so far
- we need more general and more quantitative design methods
- many different “recipes” exist
- goal is to fulfill a set of requirements while minimizing some error metric

filter design: the setup

The filter design problem

You are given a set of requirements:

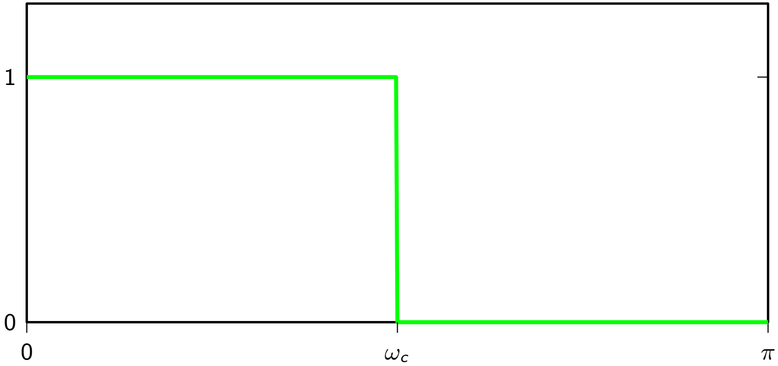
- frequency response: passband(s) and stopband(s)
- phase: overall delay, linearity
- some limit on computational resources and/or numerical precision

You must determine N, M, a_k's and b_k's in

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_{M-1} z^{-M}}{1 + a_1 z^{-1} + \ldots + a_{N-1} z^{-N}}$$

in order to best fulfill the requirements

Example: lowpass specs



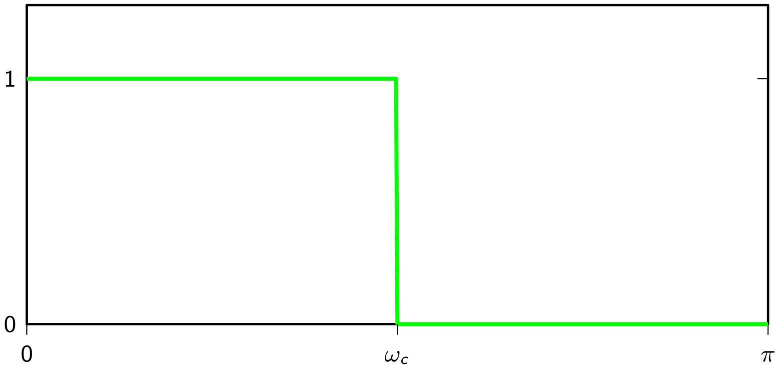
Practical limitations

- passband/stopband transitions cannot be infinitely sharp
  $\Rightarrow$ use *transition bands*

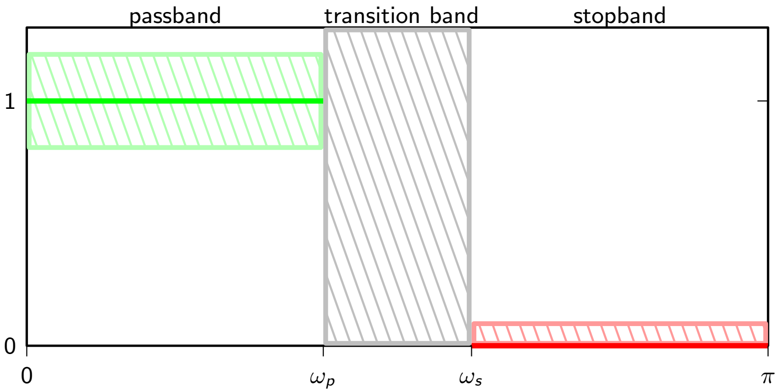
- magnitude response cannot be constant over an interval
  $\Rightarrow$ specify magnitude *tolerances* over bands

- in general:
  - smaller transition bands $\Rightarrow$ higher filter order
  - smaller error tolerances $\Rightarrow$ higher filter order
  - higher filter order $\Rightarrow$ more expensive, larger delay

Example: lowpass specs



Realistic specs



Why we can't have a "vertical" transition

$$H(z) = \frac{B(z)}{A(z)} \quad \text{is a rational function with } A, B \in C^{\infty}$$

polynomial rational functions cannot have jump discontinuities

Why we can't have a flat response

$$H(z) = \frac{B(z)}{A(z)}, \qquad \text{with } A \text{ and } B \text{ polynomials}$$

$$\begin{aligned}
H(e^{j\omega}) = c \text{ over an interval} \quad &\Rightarrow \quad B(z) - cA(z) = 0 \text{ over an interval} \\
&\Rightarrow \quad B(z) - cA(z) \text{ has an infinite number of roots} \\
&\Rightarrow \quad B(z) - cA(z) = 0 \text{ for all values of } z \\
&\Rightarrow \quad H(e^{j\omega}) = c \text{ over the entire } [-\pi, \pi] \text{ interval.}
\end{aligned}$$

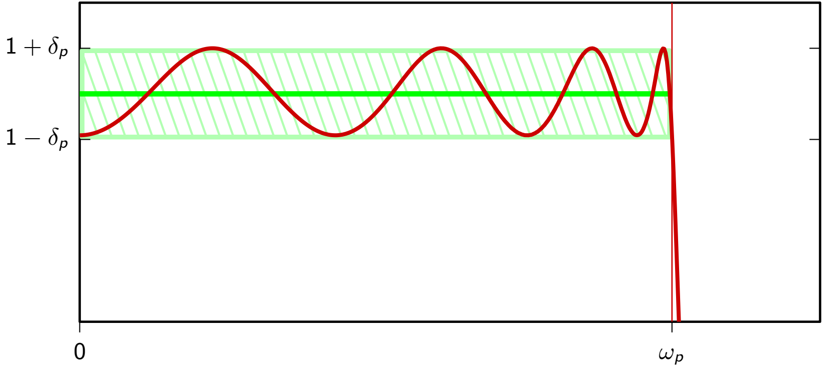
Deviation from the target response

frequency response cannot be constant so there will be an approximation error:

- it's important to be able to control the max error
- error can change monotonically
- error can oscillate around zero

Important case: equiripple error

equiripple: max and min error values alternate with equal magnitude



The big questions

- IIR or FIR?
- how to determine the coefficients?
- how to evaluate the performance?

IIRs: pros and cons

Pros:

- computationally efficient
- can achieve strong attenuations easily
- "natural sounding" in audio applications

Cons:

- stability and numerical precision issues
- difficult to design for arbitrary frequency responses
- phase response is always nonlinear

FIRs: pros and cons

Pros:

- always stable
- numerically robust
- optimal design techniques exist for arbitrary responses
- can have linear phase

Cons:

- computationally more expensive than similar IIRs
- large processing delay (not suitable for “live” applications)

The design methods

- finding N, M, a_k's and b_k's from specs is a difficult nonlinear problem
- established methods:
  - IIR: ready-made cookbooks (based on old analog designs)
  - FIR: optimal design algorithm (Parks-McClellan)

IIR filter design methods

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

Three classic filter families to be aware of

- Butterworth (smooth monotonic frequency response)
- Chebyshev (monotonic/equiripple)
- Elliptic (equiripple)

Butterworth lowpass

Magnitude response:

- maximally flat
- monotonic over $[0, \pi]$

Design parameters:

- order N (N poles and N zeros)
- cutoff frequency

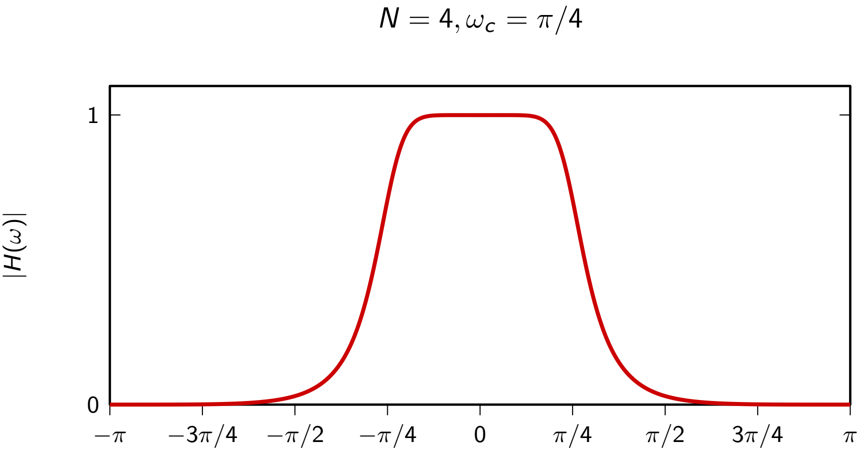
Design test criterion:

- width of transition band
- passband error

Butterworth lowpass design with SciPy

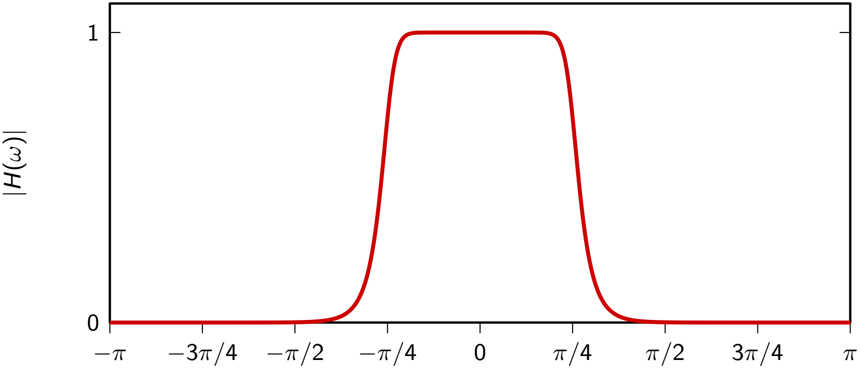
```
import scipy.signal as sp

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



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:

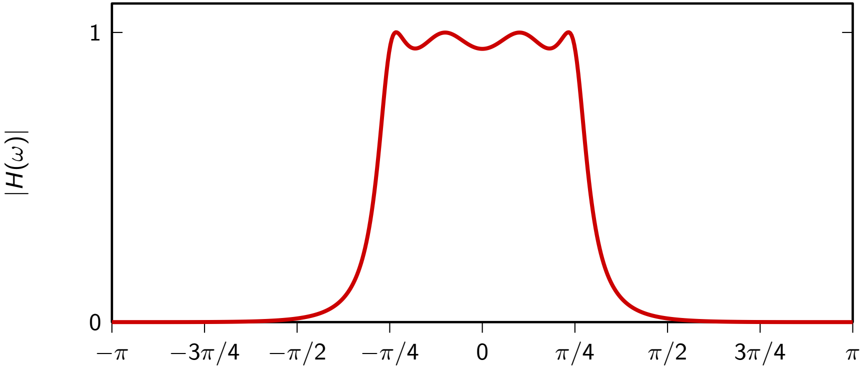
- width of transition band
- stopband error

Chebyshev lowpass design with SciPy

```
b, a = sp.cheby1(4, .12, 0.25)
```

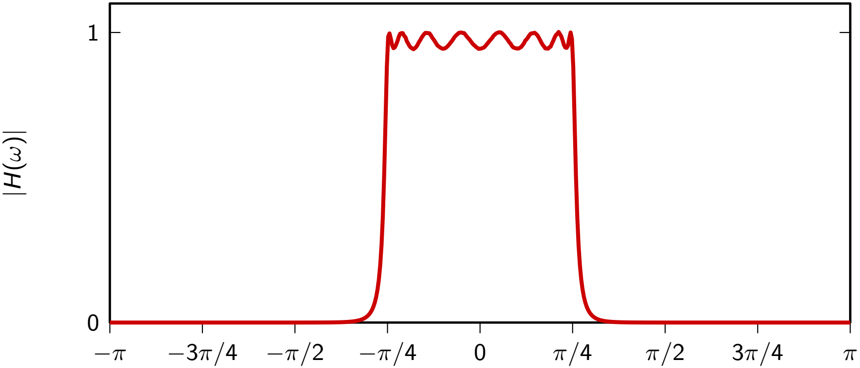
Chebyshev lowpass example

$$N = 4, \omega_c = \pi/4, e_{\max} = 12\%$$



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:

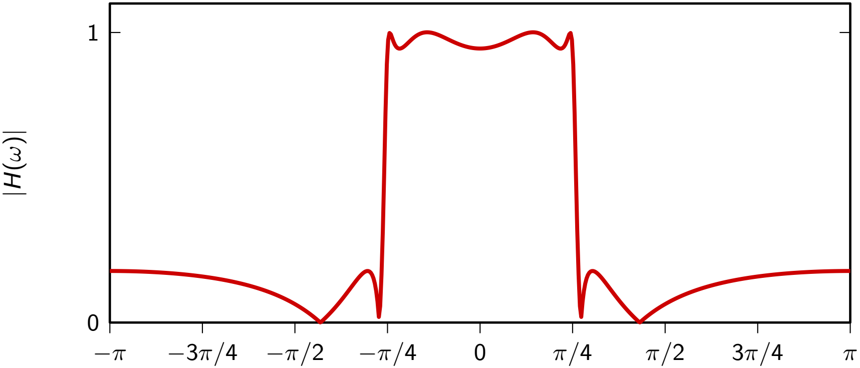
- width of transition band

Elliptic lowpass design with SciPy

```
b, a = sp.ellip(4, .1, 50, 0.25)
```

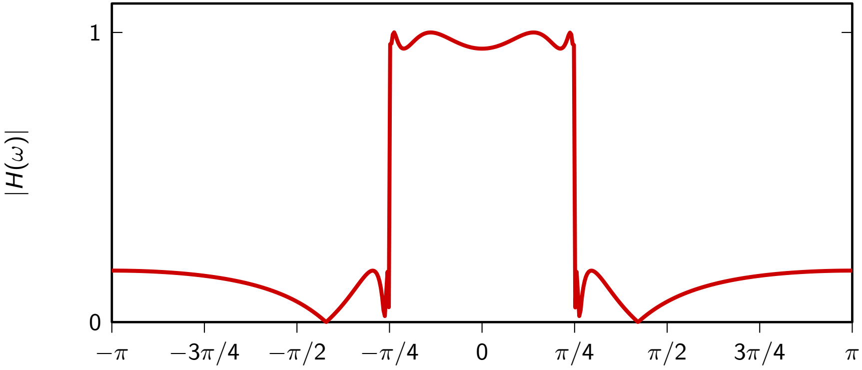
Elliptic lowpass example

$$N = 4, \omega_c = \pi/4, e_{\text{max}} = 12\%, \text{att}_{\text{min}} = 0.03$$

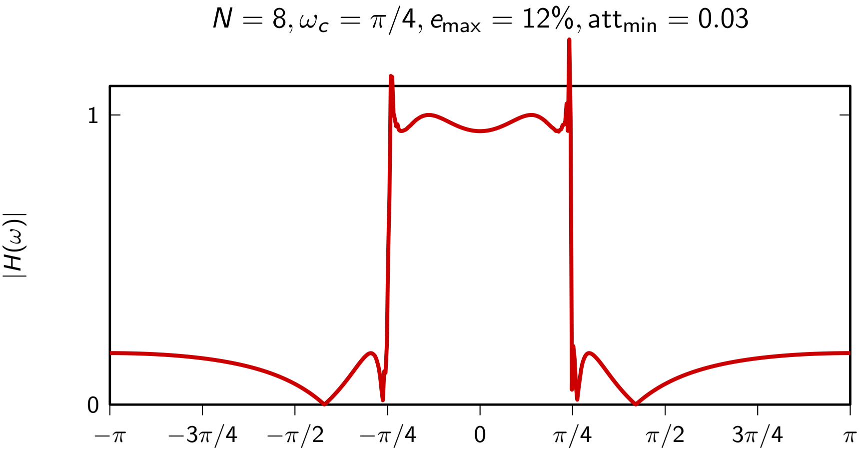


Elliptic lowpass example

$$N = 6, \omega_c = \pi/4, e_{\text{max}} = 12\%, \text{att}_{\text{min}} = 0.03$$



Elliptic lowpass example: numerical errors for high-order



Let’s compare

- compare magnitude response of 4th-order lowpass filters
- same cutoff frequency and transition band width
- plot the magnitude response in dB

The decibel for amplitude ratios

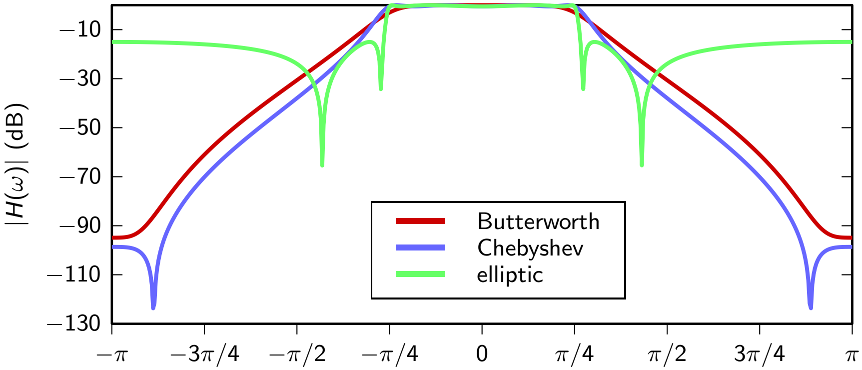
Relative measure of amplitude in log scale:

$$|H(\omega)|_{\mathrm{dB}} = 20 \log_{10} \frac{|H(\omega)|}{H_{\mathrm{ref}}}$$

Here we choose $H_{\mathrm{ref}} = 1$, target value in passband.

- -6 dB = half the amplitude
- -20 dB = one tenth of the amplitude

4-th order IIR lowpass comparison



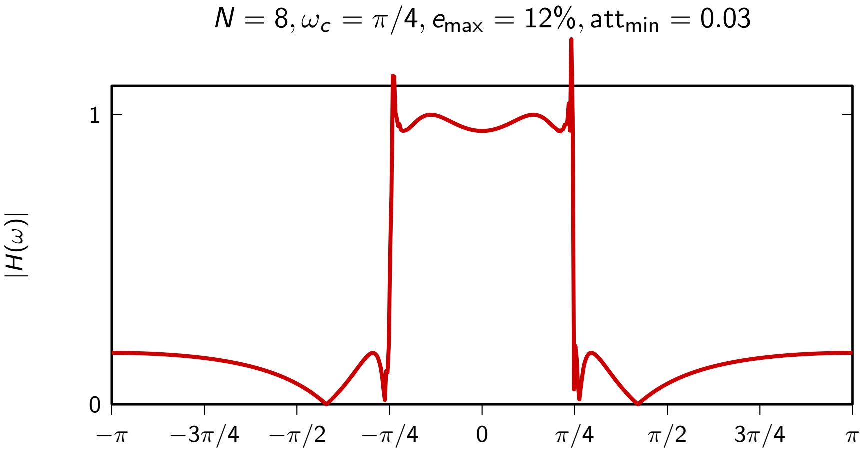
all filters require 9 multiplications per output sample

Qualitative comparison

For a given order N

- sharpness of transition band: Elliptic $>$ Chebyshev $>$ Butterworth
- phase distortion: Butterworth $<$ Chebyshev $<$ Elliptic
- passband ripples Butterworth $<$ Chebyshev $<$ Elliptic
- stopband attenuation: Elliptic $>$ Chebyshev $>$ Butterworth

Elliptic lowpass example: numerical errors for high-order



Numerical precision issues

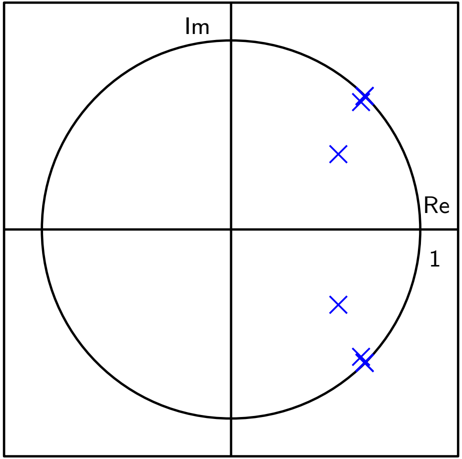
- all digital devices represent numbers using finite precision
- poles are the roots of the denominator of the transfer function
- filter algorithms store the value of the coefficients, not of the poles
- the value of a pole is a nonlinear function of the filter coefficients
- insufficient numerical precision may cause poles to drift out of unit circle

Pole drifting: example

- nominal pole: $p = \rho e^{j\theta}$, magnitude $|p| = \rho$
- second-order transfer function: $P(z) = (1 - pz^{-1})(1 - p^* z^{-1})$
- $P(z) = 1 + a_1 z^{-1} + a_2 z^{-2}$, with $a_1 = -2\rho \cos\theta$ and $a_2 = \rho^2$
- coefficients $a_{1,2}$ are stored with finite precision
- actual pole magnitude $|\hat{p}| = \frac{2}{|\sqrt{a_1^2 - 4a_2} - a_1|}$

# decimal digits for $a_{1,2}$	$\rho - |\hat{p}|$
8	$2.22 \cdot 10^{-16}$
7	$5.00 \cdot 10^{-9}$
4	$5.00 \cdot 10^{-9}$
3	$4.00 \cdot 10^{-4}$
2	$4.91 \cdot 10^{-3}$

Poles of the 8th order elliptic lowpass



Pole magnitude

Magnitude of poles as a function of the number of digits used to store coefficients

# digits				
9	0.99969893	0.99641971	0.96231223	0.6929287
8	0.99970234	0.99641583	0.96231266	0.69292873
7	0.99987231	0.99622669	0.96233196	0.69292855
6	1.0027213	0.99267273	0.96304264	0.69292212
5	1.00418091	0.99647046	0.95797945	0.69292331

Numerical precision: how to mitigate

- design filter in factored form
- use a cascade of second-order sections
- in Python: `b, a = sp.ellip(4, .1, 50, 0.25, output='sos')`

FIR filter design methods

IIRs: pros and cons (recap)

Pros:

- computationally efficient
- can achieve strong attenuations easily
- "natural sounding" in audio applications

Cons:

- stability and numerical precision issues
- difficult to design for arbitrary frequency responses
- phase response is always nonlinear

FIRs: pros and cons (recap)

Pros:

- always stable
- numerically robust
- optimal design techniques exist for arbitrary responses
- can have linear phase

Cons:

- computationally more expensive than similar IIRs
- large processing delay (not suitable for “live” applications)

FIR design methods

FIR filters exist only in discrete time (there are no analog FIRs)

Three important design methods:

- impulse truncation, window method
- frequency sampling
- Parks-McClellan algorithm

Quick-and-dirty design methods (recap)

- impulse truncation
- frequency sampling

Advantages:

- simple and intuitive
- can be applied to arbitrary frequency responses

Drawbacks:

- cannot control the approximation error
- longer than optimally-desinged FIRs

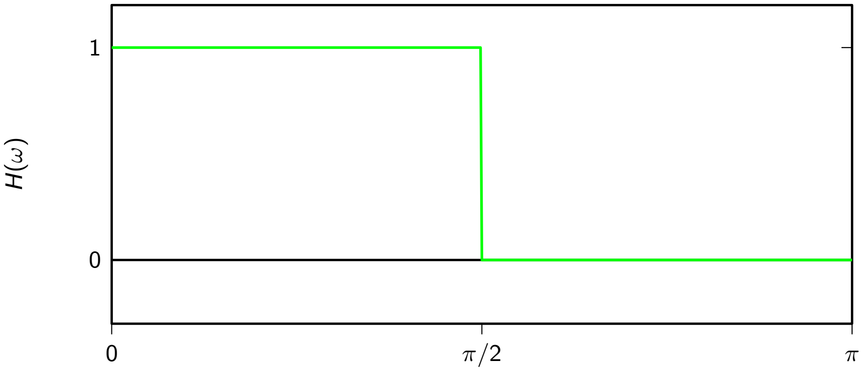
Impulse truncation (recap)

- start with a zero-phase ideal filter (or combination thereof)
- derive the closed-form expression of the impulse response $h[n]$
- keep $M = 2N + 1$ samples around $n = 0$:

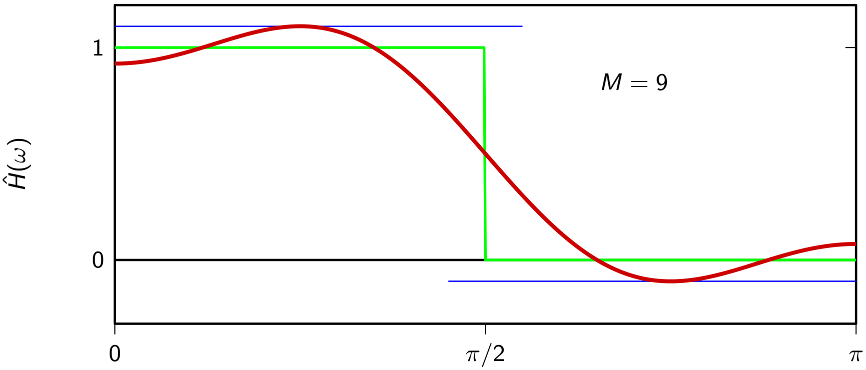
$$\hat{h}[n] = \begin{cases} h[n] & |n| \leq N \\ 0 & \text{otherwise} \end{cases}$$

- we may use a tapering window to reduce ripples

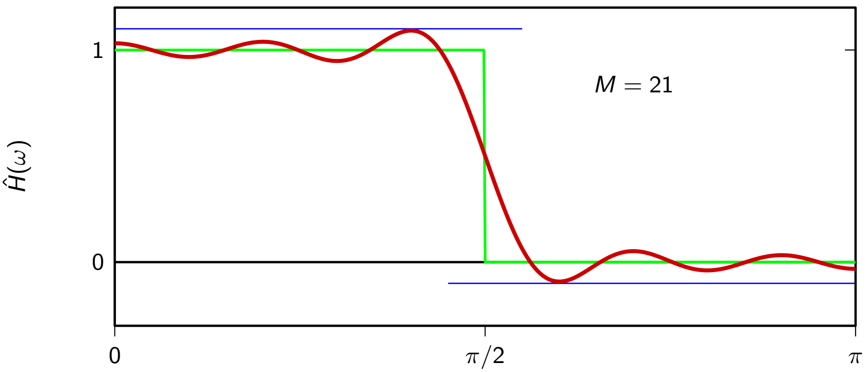
The Gibbs phenomenon (recap)



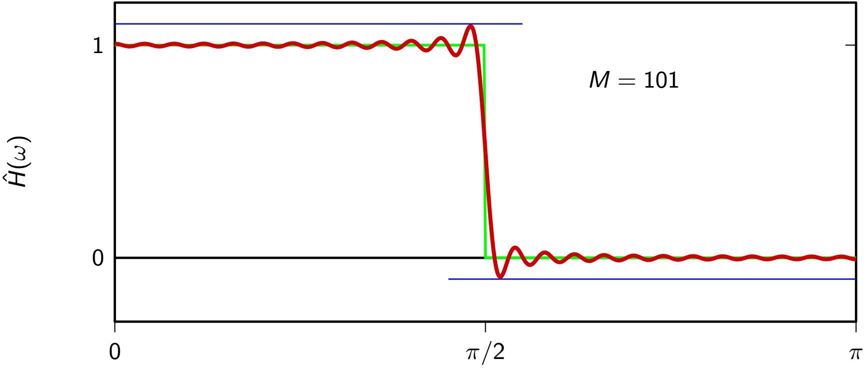
The Gibbs phenomenon (recap)



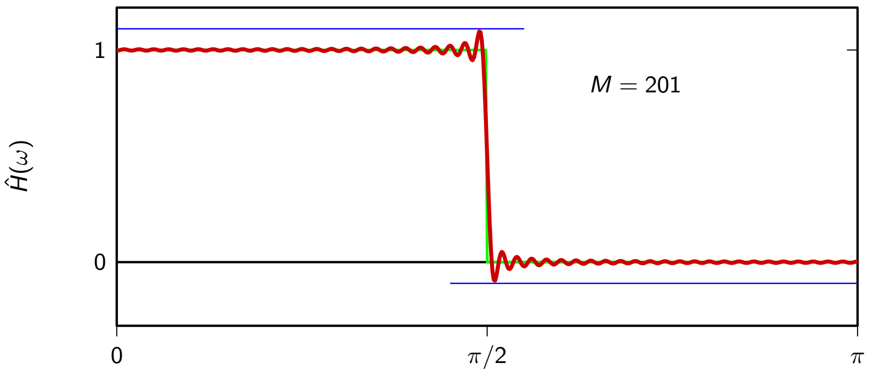
The Gibbs phenomenon (recap)



The Gibbs phenomenon (recap)



The Gibbs phenomenon (recap)



Frequency sampling (recap)

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

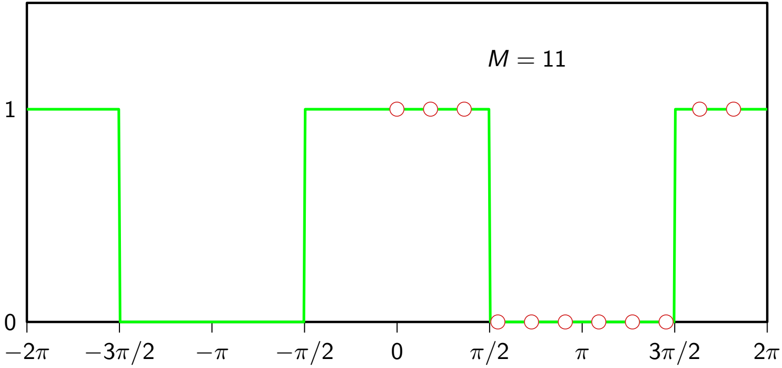
$$H_M[k] = H((2\pi/M)k), \quad k = 0, 1, \ldots, M-1$$

- compute the inverse DFT: $h_M[n] = \text{IDFT}\{H_M[k]\}$
- use the impulse response

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

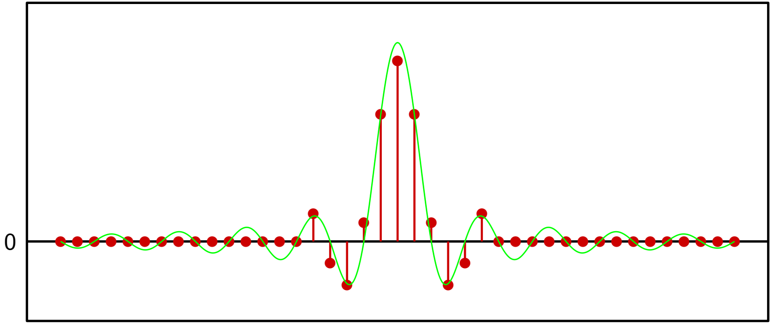
Example: ideal lowpass with cutoff $\pi/2$

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

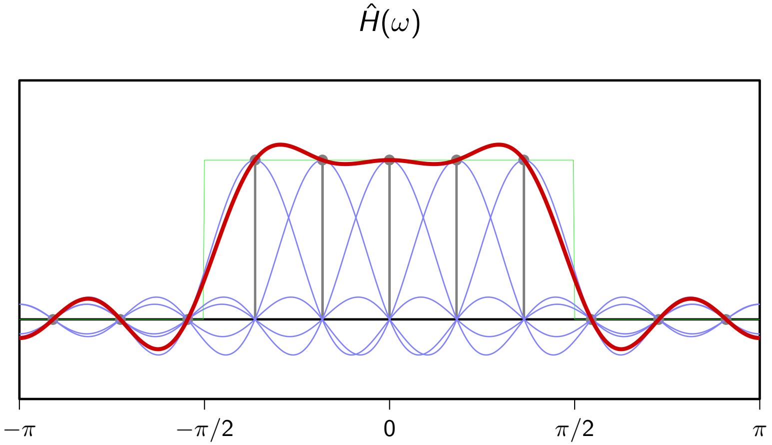


Frequency sampling: impulse response from IDFT

$h_M[n]$ $\hat{h}[n]$



Frequency sampling: frequency response



still no control over max error

linear-phase FIRs

Optimal linear-phase filter design

In the 1970s Parks and McClellan developed an algorithm to design optimal FIR filters with

- (generalized) linear phase
- equiripple error in passband and stopband

Linear phase responses

- zero phase: $H(\omega) \in \mathbb{R}$
  - no processing delay
  - examples: $h[n] = \delta[n]$, $h[n] = \text{sinc}(an)$
- linear phase: $H(\omega) = A(\omega)e^{-j\omega d}, \quad A(\omega) \in \mathbb{R}$
  - processing delay of d samples
  - examples: $h[n] = \delta[n-d]$, moving average filter
- generalized linear phase: $H(\omega) = A(\omega)e^{-j(\omega d - \beta)}, \quad A(\omega) \in \mathbb{R}$
  - if $h[n] \in \mathbb{R}$ then $\beta = \pm\pi$ or $\beta = \pm\pi/2$
  - examples: $h[n] = \delta[n] - \delta[n-2]$ $(d = 1, \beta = -\pi/2)$,
    $h[n] = \delta[n] + j\delta[n-2]$ $(d = 1, \beta = \pi/4)$

Impulse responses with generalized linear phase

$$H(\omega) = A(\omega)e^{-j(\omega d - \beta)}, \quad A(\omega) \in \mathbb{R}$$

- impulse response

$$e^{j\omega d} H(\omega) = A(\omega)e^{j\beta} \Rightarrow h[n+d] = e^{j\beta}\, a[n]$$

- condition on $a[n]$:

$$A(\omega) \in \mathbb{R} \Leftrightarrow a[n] = a^*[-n]$$

- condition on $h[n]$:

$$h[d+n] = e^{2j\beta} h^*[d-n]$$

Impulse responses for generalized linear phase

$$h[d+n] = e^{2j\beta} h^*[d-n]$$

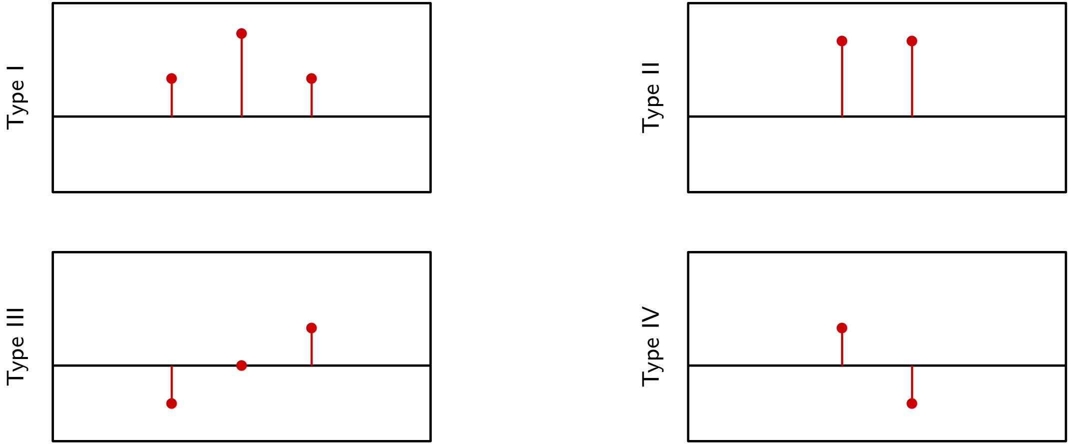
- if $h[n] \in \mathbb{R}$:
  - symmetric impulse response ($\beta = \pm\pi$): $h[d+n] = h[d-n]$
  - antisymmetric impulse response ($\beta = \pm\pi/2$): $h[d+n] = -h[d-n]$
- if $h[n] \in \mathbb{C}$:
  - hermitian-symmetric impulse response ($\beta = \pm\pi$): $h[d+n] = h^*[d-n]$
  - hermitian-antisymmetric impulse response ($\beta = \pm\pi/2$): $h[d+n] = -h^*[d-n]$
  - otherwise: $(e^{-j\beta} h[d+n]) = (e^{-j\beta} h[d-n])^*$

Impulse responses for generalized linear phase

- realizable IIRs cannot have linear phase:
  - if IIR, $h[n]$ extends at least to $+\infty$ or $-\infty$
  - $h[d+n] = e^{2j\beta} h^*[d-n]$ implies that $h[n]$ is infinite, two sided
  - only ideal (non-realizable) IIRs like the sinc can have linear phase
- we will consider only real-valued, FIR filters
- there are four types of real-valued, linear-phase FIRs

Linear-phase FIRs

symmetric or antisymmetric impulse responses guarantee linear phase



Linear phase (Type I)

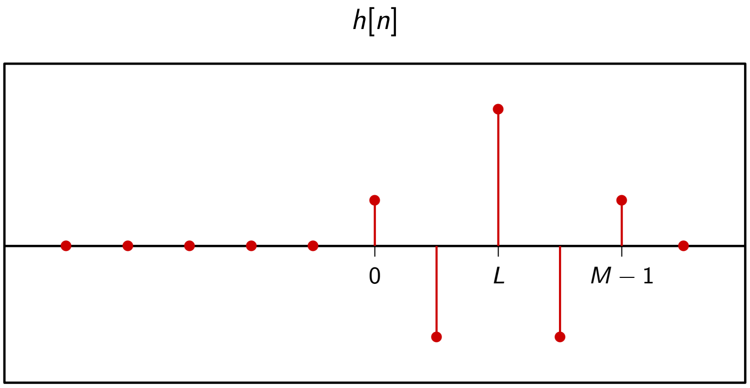
filter length is **odd**: $M = 2L + 1$

$$h[L + n] = h[L - n]$$

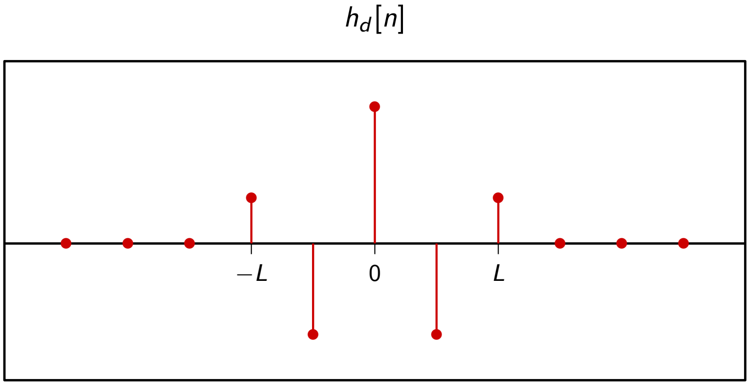
zero-centered filter:

$$h_d[n] = h[n + L]$$
$$h_d[n] = h_d[-n]$$

Causal linear phase (Type I)



Noncausal zero phase (Type I)



Linear phase (Type I)

$$H_d(z) = \sum_{n=-L}^{L} h_d[n] z^{-n}$$

$$= h_d[0] + \sum_{n=1}^{L} h_d[n](z^n + z^{-n})$$

$$H_d(\omega) = h_d[0] + \sum_{n=1}^{L} h_d[n](e^{j\omega n} + e^{-j\omega n})$$

$$= h_d[0] + 2\sum_{n=1}^{L} h_d[n] \cos \omega n \qquad \in \mathbb{R}$$

Linear phase (Type I)

$$H(z) = z^{-L} H_d(z)$$

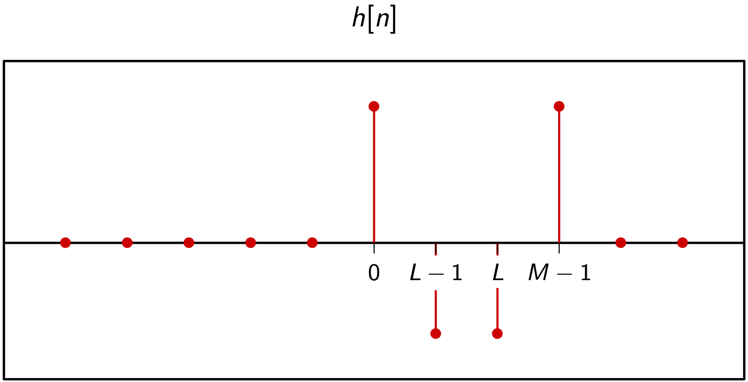
$$H(\omega) = \left[h[L] + 2 \sum_{n=1}^{L} h[n+L] \cos n\omega \right] e^{-j\omega L}$$

Linear phase (Type II)

filter length is **even**: $M = 2L$

$$h[n] = h[2L - 1 - n]$$

Linear phase (Type II)



Linear phase (Type II)

$$
\begin{aligned}
H(z) =& h[0] && + h[1]z^{-1} && + \dots && + h[L-1]z^{-L+1} + \\
& h[2L-1]z^{-2L+1} && + h[2L-2]z^{-2L+2} && + \dots && + h[L]z^{-L} \\
=& h[0] && + h[1]z^{-1} && + \dots && + h[L-1]z^{-L+1} + \\
& h[0]z^{-2L+1} && + h[1]z^{-2L+2} && + \dots && + h[L-1]z^{-L} \\
=& \sum_{n=0}^{L-1} h[n](z^{-n} + z^{-2L+1+n})
\end{aligned}
$$

Linear phase (Type II)

$$C = (M-1)/2 = (2L-1)/2 = L - 1/2 \quad \text{(non-integer!)}$$

$$\begin{aligned} H(z) &= \sum_{n=0}^{L-1} h[n](z^{-n} + z^{-2C+n}) \\ &= z^{-C} \sum_{n=0}^{L-1} h[n](z^{(C-n)} + z^{-(C-n)}) \end{aligned}$$

Linear phase (Type II)

$$H(\omega) = \left[2 \sum_{n=0}^{L-1} h[n] \cos(\omega(C - n)) \right] e^{-j\omega C}$$

$$C = L - \frac{1}{2}$$

Linear-phase FIRs

- frequency response is of the form

$$H(\omega) = A(e^{j\omega})e^{-jC\omega}, \qquad A(e^{j\omega}) \in \mathbb{R}$$

- processing delay is $C = (M-1)/2$ samples
- delay is non-integer for even-length filters!

Zero locations

this applies to all FIRs, linear-phase or not:

- FIRs have only zeros
- transfer function is a finite-degree polynomial: $H(z) = \sum_{k=0}^{M-1} h[k]z^{-k}$
- if $h[n] \in \mathbb{R}$ and $H(z_0) = 0$ then $H(z_0^*) = 0$

the zeros of linear-phase FIRs have additional properties

Zero locations for Type I

$$H(z) = z^{-L}\left[h[0] + \sum_{n=1}^{L} h[n](z^n + z^{-n})\right]$$

$$H(z^{-1}) = z^{L}\left[h[0] + \sum_{n=1}^{L} h[n](z^n + z^{-n})\right]$$

$$H(z^{-1}) = z^{2L} H(z)$$

$$\text{if } H(z_0) = 0 \text{ then } H(1/z_0) = 0$$

Property of all linear-phase FIRs

if z_0 is a zero, $1/z_0$ is also a zero

(easy to prove for all linear-phase FIRs)

Zero locations for linear-phase FIRs

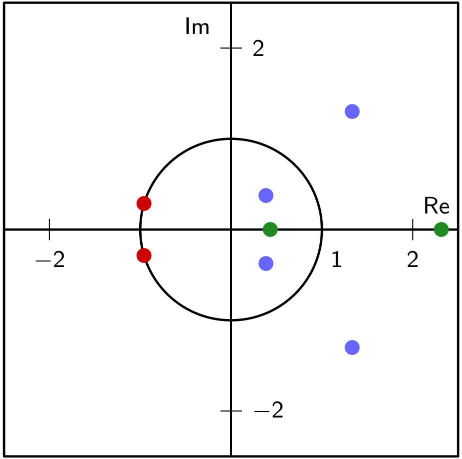
If $H(z_0) = 0$:

- $H(z_0^*) = 0$
- $H(1/z_0) = 0$

If $z_0 = \rho e^{j\theta}$ is a zero, these are zeros too:

- $\rho e^{-j\theta}$
- $(1/\rho) e^{j\theta}$
- $(1/\rho) e^{-j\theta}$

Typical zero plot for linear-phase FIRs



Forced zeros in linear-phase FIRs

because of the symmetries in the impulse response,
linear-phase FIRs (xcept Type I) have "automatic" zeros

type	**forced zero locations**
Type I	none
Type II	zero at $\omega = \pi$
Type III	zeros at $\omega = 0$ and $\omega = \pm\pi$
Type IV	zero at $\omega = 0$

Example: forced zeros in Type III

$$H(z) = z^{-L}\left[\sum_n h_d[n](z^n - z^{-n})\right]$$

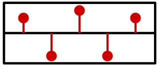
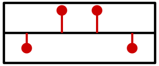
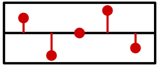
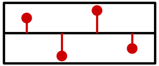
$$H(z^{-1}) = -z^{2L}H(z)$$

$$H(1) = -H(1) \implies H(1) = 0$$

$$H(-1) = -H(-1) \implies H(-1) = 0$$

- Type III FIRs not good for low- or high-pass
- good for bandpass though!

Properties of linear-phase FIRs

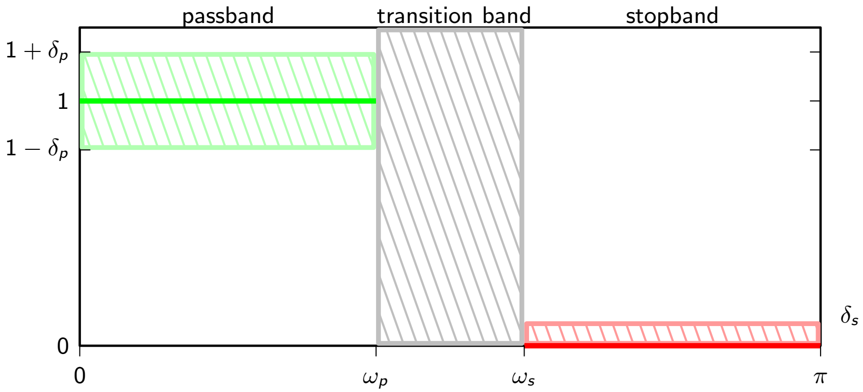
type	length	sym.	delay	zeros	
I	odd	S	integer		
II	even	S	non-int.	$\pm\pi$	
III	odd	A	integer	$0, \pm\pi$	
IV	even	A	non-int.	0	

The Parks-McClellan algorithm

The Parks-McClellan algorithm (also known as minimax optimization)

- can design all types of linear-phase FIRs
- minimizes the maximum error in passband and stopband
- the error is equiripple in passband and stopband
- can be used for “nonstandard” FIR design (Hilbert filter, differentiator, etc.)

Typical lowpass design specs



The Parks-McClellan algorithm: key ideas

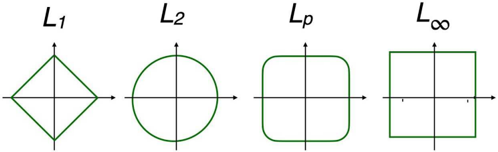
Using a zero-centered Type I (symmetric, odd-length):

- frequency response is real: $H_d(\omega) = h_d[0] + 2\sum_{n=1}^{C} h_d[n]\cos\omega n$
- use Chebyshev polynomials to write response as $P(x) = \sum_{k=0}^{C} a_k x^k$, with $x = \cos\omega$
- fit $P(x)$ to the specifications using the L_∞ norm (**mini**mizing the **max**imum error)
- solve the fitting problem with an efficient numerical algorithm (the *Remez exchange algorithm*)

Short aside: error norms (aka "loss functions")

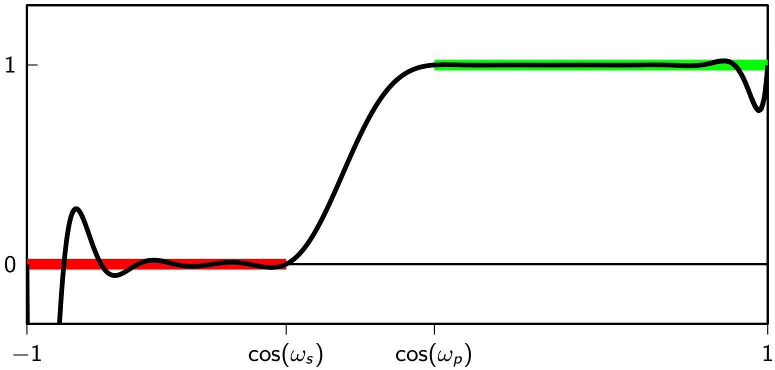
$$\|\mathbf{x}\|_p = \left(\sum_n |x_n|^p\right)^{\frac{1}{p}}$$

- L_2 norm: minimize the Mean Square Error (global minimization)
- L_1 norm: minimize the sum of the magnitudes
- L_∞ norm: minimize the maximum absolute value

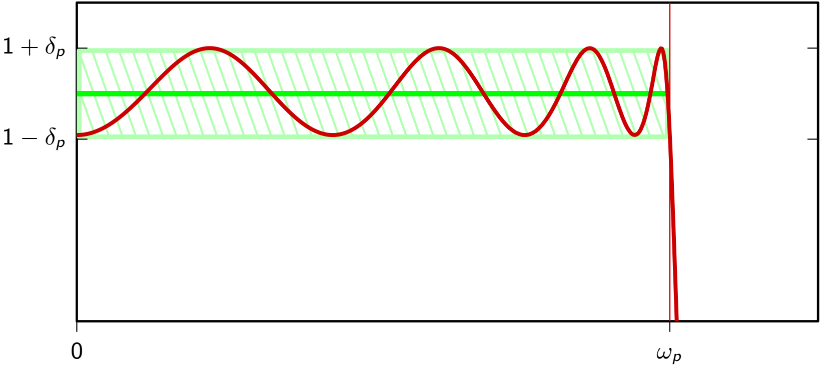


L_2 polynomial fitting doesn't work

polynomial will fit well in places and go crazy at edges...



L_∞ fitting will lead to equiripple error



The Parks-McClellan recipe for a Type I lowpass

User data:

- filter length $M = 2L + 1$
- ω_p and ω_s
- stopband -to-passband tolerance *ratio* δ_s/δ_p

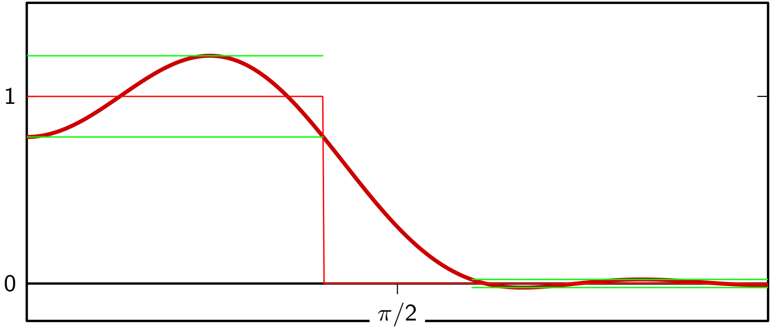
Run the Parks-McClellan algorithm to obtain:

- M filter coefficients
- stopband and passband tolerances δ_s and δ_p
- if error too big in either band, increase M and retry.

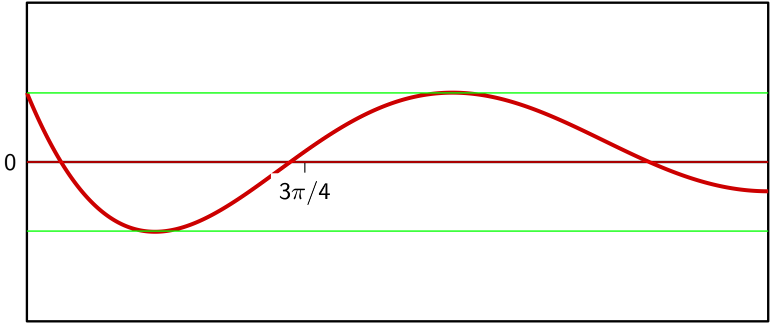
Example

- $M = 9$ ($L = 4$)
- $\omega_p = 0.4\pi$
- $\omega_s = 0.6\pi$
- $\delta_s / \delta_p = 1/10$

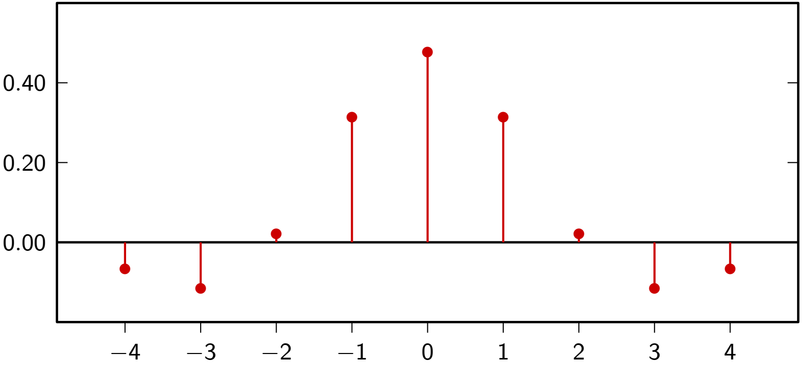
Final Result



Final Result (stopband)



Final Result (Impulse Response)



Optimal 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

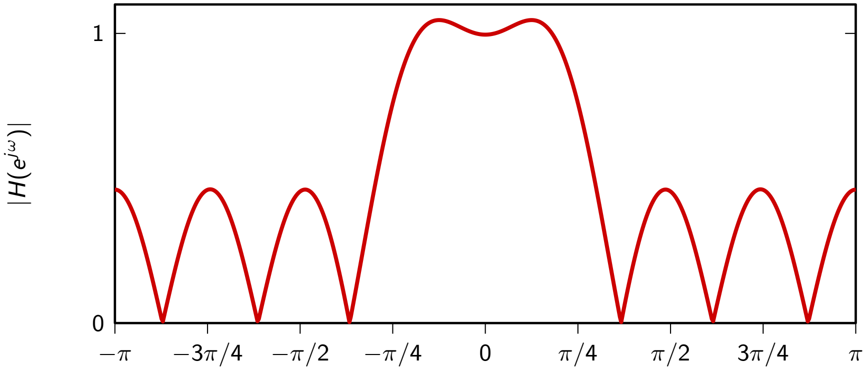
Butterworth lowpass design with SciPy

Let $p = \omega_p/(2\pi)$ and $s = \omega_s/(2\pi)$:

```
import scipy.signal as sp

M, p, s = 9, 0.1, 0.15
delta_p, delta_s = 10, 1

h = sp.remez(M, [0, p, s, 0.5], [1, 0], [delta_p, delta_s])

wb, Hb = sp.freqz(h, 1, 1024);
plt.plot(wb/np.pi, np.abs(Hb));
```

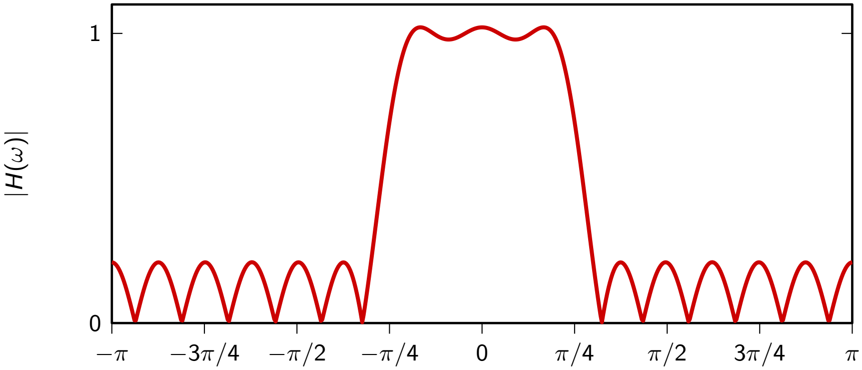
Optimal lowpass example

$$N = 9, \omega_p = 0.2\pi, \omega_s = 0.3\pi, \delta_p/\delta_s = 10$$



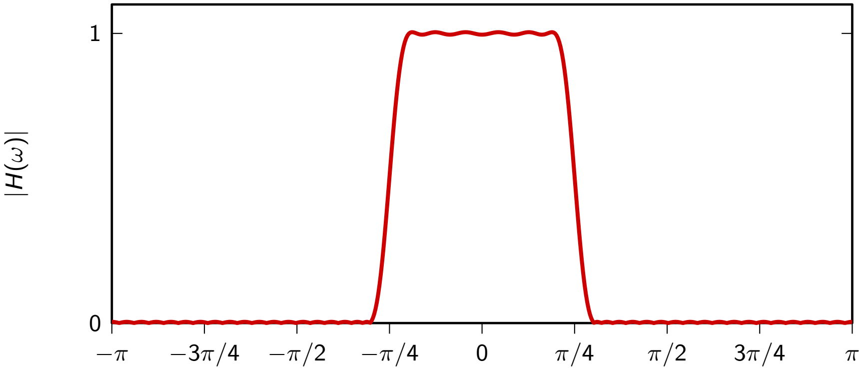
Optimal lowpass example

$$N = 19, \omega_p = 0.2\pi, \omega_s = 0.3\pi, \delta_p/\delta_s = 10$$

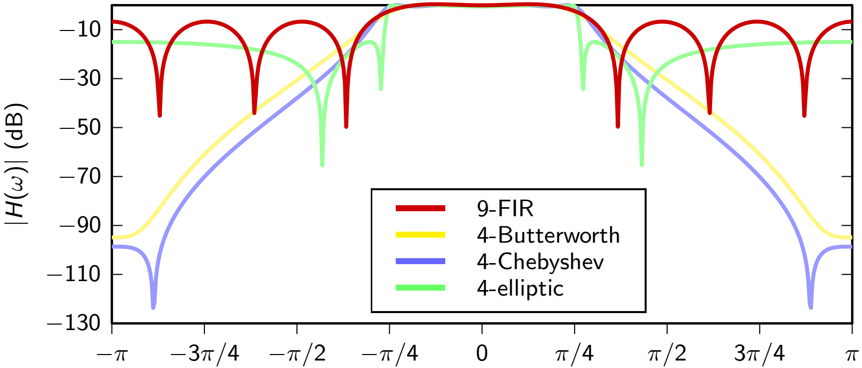


Optimal lowpass example

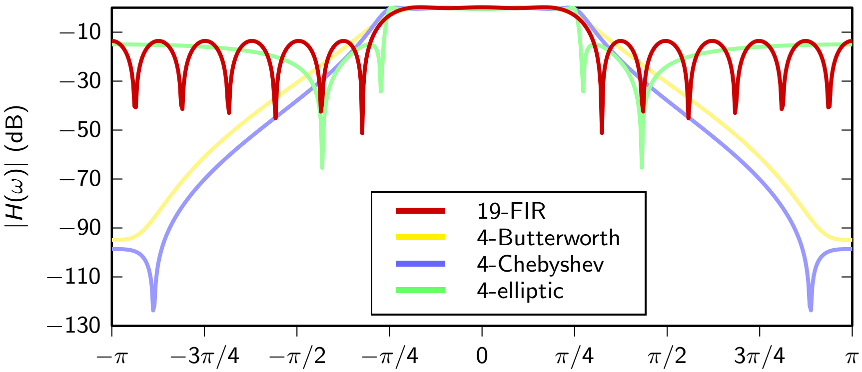
$$N = 51, \omega_p = 0.2\pi, \omega_s = 0.3\pi, \delta_p/\delta_s = 1$$



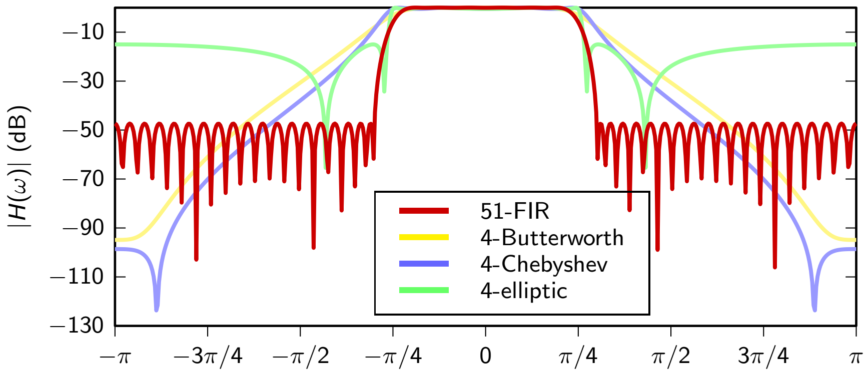
Lowpass comparison, $\omega_c = \pi/4$



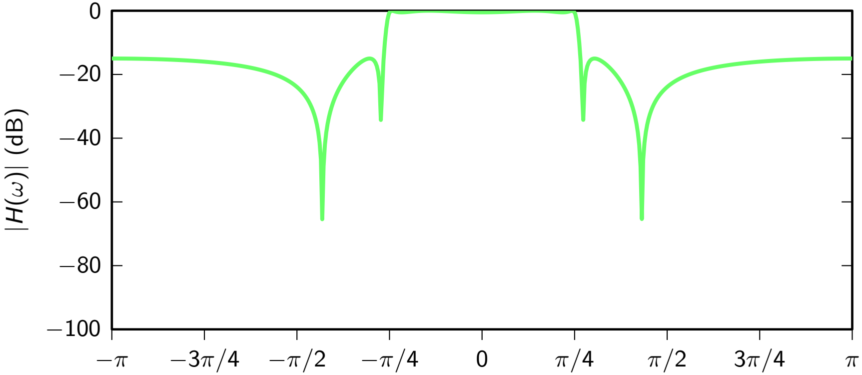
Lowpass comparison, $\omega_c = \pi/4$



Lowpass comparison, $\omega_c = \pi/4$

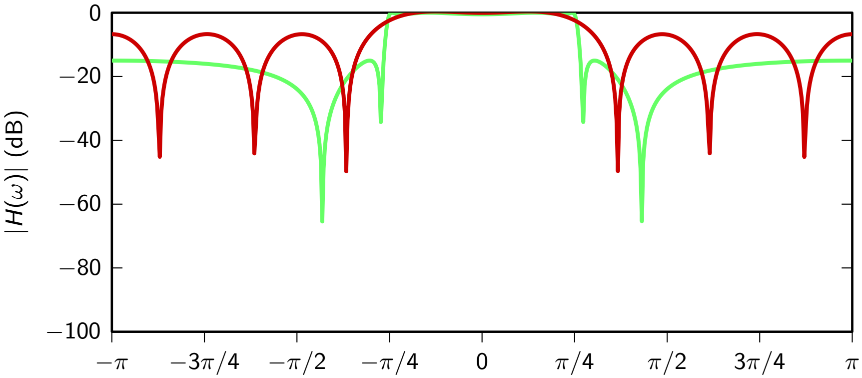


4th-order elliptic lowpass, $\omega_c = \pi/4$

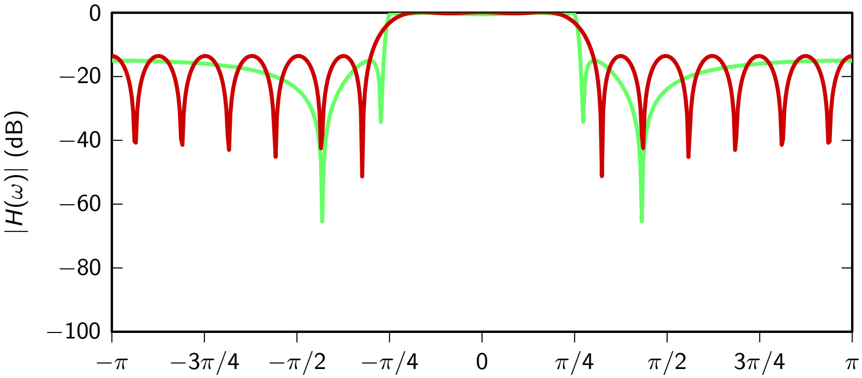


9 multiplications per output sample

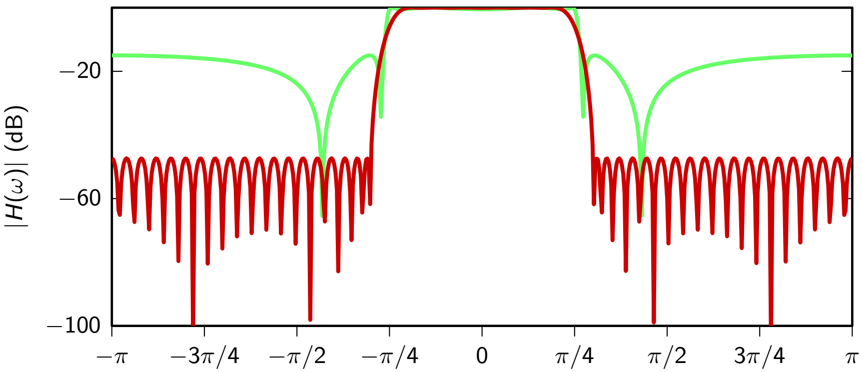
9-tap optimal FIR lowpass, $\omega_c = \pi/4$



19-tap optimal FIR lowpass, $\omega_c = \pi/4$



51-tap optimal FIR lowpass, $\omega_c = \pi/4$



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!