# Signal Processing for Communication

Ch. Bollinger

*<2020-07-20 Mo>*

## Contents

# 1 Week 1 Module 1: Basics of Digital Signal Processing

## 1.1 Introduction to digital signal processing

### 1.1.1 Signal

- Description of the evoultion of a physical phenomenon

| | |
|---|---|
| phenomenon | signal |
| weather | temperature |
| sound | pressure |
| sound | magnetic deviation |
| light intensity | gray level on paper |

### 1.1.2 Processing

- **Analysis:** Understanding the information carried by the signal
- **Synthesis:** Creating a signal to contain the given information

### 1.1.3 Digital

- Discrete Time
  - Splice up time into a series of descrete instance without loosing information
  - Harry Nyquist and Claude Shannon state with the Sampling Theorem that continous time representation and discrete time representation are equivalent.
  - The Sampling Theorem: Under appropriate "slowness" conditions for x(t) we have

$$x(t) = \sum_{n=-\infty}^{\infty} x[n]\ sinc(\frac{t - nT_s}{T_s}) \tag{1}$$

  - The conditiion under which the Sampling Theorem holds was given by Fourier and it's Fourier Analysis.
  - The fouriere transform will give us a quantitive measure how fast a signal moves
- Discrete Amplitude
  - Through discretisation of ampltitudes only a set of predefined values are possible.
  - The set of levels is countable i.e. we can always map the level of a sample to an integer. If our data is represented by integer it becomes complete abstract and general which has very importand consequences in the following three domains:
    - **Storage** special devices for recoding needed
    - **Processing** General purpose microprocessor is sufficient
    - **Transmission** Reproduction of the original signal and therefore eliminating nois is easy

### 1.1.4 From Analog to Digital Signal Processing

- Analog asks for $f_{(t)}$ =?
- Digital represents data as a sequence of numbers (scaled with a factor of 1000)

-12 -12 -12 -11 -11 -12 -12 -11 -11 -10

-10 -10 -9 -10 -10 -9 -9 -9 -9 -9

-8 -8 -7 -7 -8 -8 -8 -7 -7 -7

## 1.2 Discrete-Time Signals

### 1.2.1 Basic Definitions

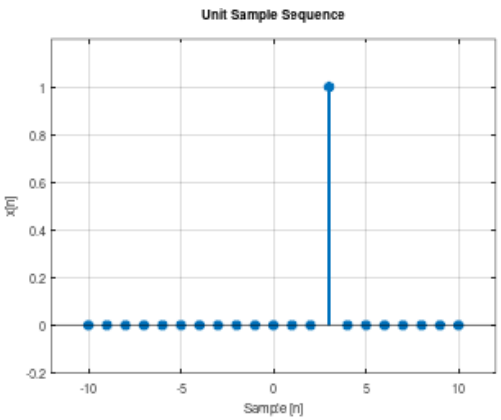
- Sequence: defined as complex-valued function
- Discrete-Time Signal: a sequnece of complex numbers
  - one dimension (for now)
  - notation: x[n]
  - two-sides sequencies: x: $\mathbb{Z} \to \mathbb{C}$
  - n is *a-dimensional* "time", sets an order on the sequence of samples
  - analysis: periodic measurement
  - synthesis: stream of generated samples, reproduce a physical phenomenon

### 1.2.2 Octave Algorithm for some basic Signals

#### Unit Impulse

```
function [x,n] = impseq(n0,n1,n2)
% Generates x(n) = delta(n-n0); n1 <= n0 <= n2
% ---------------------------------------------
% [x,n] = impseq(n0,n1,n2)
%
  n = [n1:n2]; x = [(n-n0) == 0];
end
```

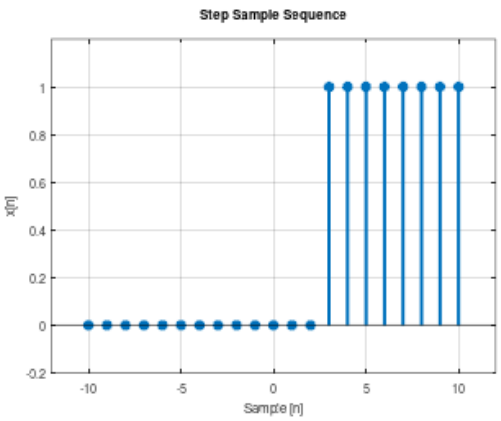
$$x[n] = \delta[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$




#### Unit Step

```
function [x,n] = stepseq(n0,n1,n2)
% Generates x(n) = delta(n-n0); n1 <= n0 <= n2
% ---------------------------------------------
% [x,n] = stepseq(n0,n1,n2)
%
  n = [n1:n2]; x = [(n-n0) >= 0];
end
```

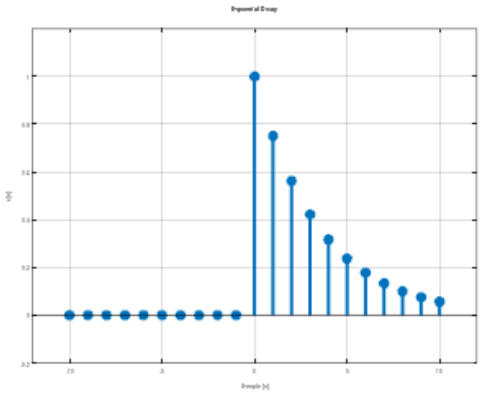
$$x[n] = u[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$




### Real-valued exponential Sequence

```
function [x,n] = expseq(n1,n2,a)
% Generates x(n) = a^n
% ------------------------------------------
% [x,n] = expseq(n1,n2,A,omega,phi)
%
  n = [n1:n2];
  for (i = 1 : length(n))
    if (n(i) >= 0)
      x(i) = (a).^n(i);
    else
      x(i) = 0;
    end
  end
end
```

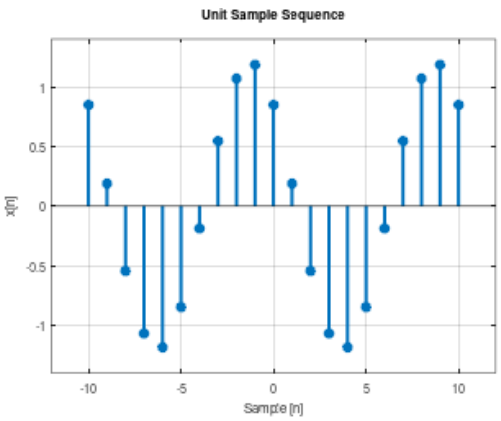
$$x[n] = a^n, \forall n\ a \in \mathbb{R}$$



### Sinusoidal Sequence

```
function [x,n] = cosseq(n1,n2,A, omega, phi)
% Generates x(n) = A*cos(2*pi*omega*n + phi); n1 <= n2
% ------------------------------------------
% [x,n] = cosseq(n1,n2,A,omega,phi)
%
  n = [n1:n2]; x = A*cos(2*pi*omega*n + phi);
end
```

$$x[n] = A\,cos(\omega_0 n + \Phi)$$




### 1.2.3 Classes of Discrete-Time signals

#### 1.2.3.1 Finite-Length

- indicate notation: $x[n]$, $n = 0.1.2.....N-1$
- vector notation: $x = [x_0, x_1, ...x_{N-1}]^T$
- practical entities, good for numerical packages (e.g. numpy)

#### 1.2.3.2 Infinte-Length

- sequence notation: $x[n]$, n $\in \mathbb{Z}$
- abstraction, good for theorems

#### 1.2.3.3 Periodic

- N-periodic sequence: $\tilde{x}[n] = \tilde{x}[n + kN]$, n,k,N $\in \mathbb{Z}$
- same information as in finite-length of length N
- natural bridge between finite and infinite length

#### 1.2.3.4 Finite-Support Finite-support sequence

$$\overline{x}[n] = \begin{cases} x[n] & if 0 \leq n < N, n \in \mathbb{Z} \\ 0 & otherwise \end{cases} \tag{2}$$

- same information as in finite-length of length N
- another bridge between finite and infinite lengths

#### 1.2.3.5 Elementary Operations

**Scaling**

$$y[n] = ax[n] \rightarrow \begin{cases} a > 0 & amplification \\ a < 0 & attenuation \end{cases} \tag{3}$$

**Sum**

$$y[n] = x[n] + z[n] \tag{4}$$

**Product**

$$y[n] = x[n] * z[n] \tag{5}$$

**Shift**

$$y[n] = x[n-k] \rightarrow \begin{cases} k > 0 & deleay \\ k < 0 & anticipate \end{cases} \tag{6}$$

**Integration**

$$y[n] = \sum_{k=-\infty}^{n} x[k] \tag{7}$$

**Differentation**

$$y[n] = x[n] - x[n-1] \tag{8}$$

**Relation Operator and Signals**

- The unit step can be optained by applying the integration operator to the discrete time pulse.
- The unit impulse can be optained by applying the differentation operator to the unit step.

### 1.2.4 Energy and Power

**Energy** Many sequencies have an infinity amount of energy e.g. the unit step u[n],

$$E_x = ||x||_2^2 = \sum_{k=-\infty}^{\infty} |x[n]|^2 \tag{9}$$

**Power** To describe the energetic properties of the sequencies we use the concept of power

$$P_x = ||x||_2^2 = \frac{1}{N} \sum_{n=0}^{N-1} |x[n]|^2 \tag{10}$$

Many signals have infi

## 1.3 Basic signal processing

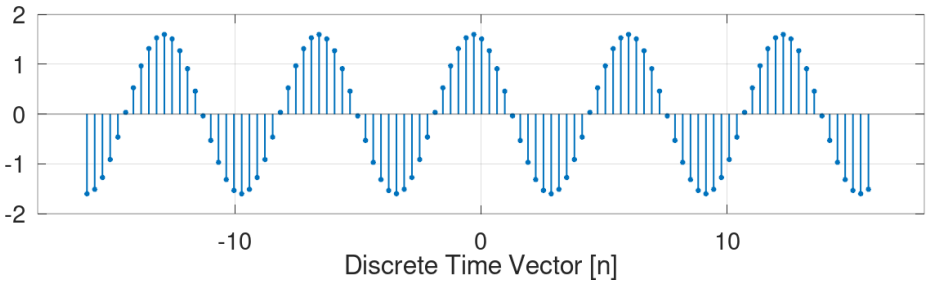
### 1.3.1 How a PC plays discrete-time sounds

#### 1.3.1.1 The discrete-time sinusoid

$$x[n] = sin(\omega_0\, t + \Theta)$$

```
N=33                        # Vector lenght
n=-(N-1)/2:pi/10:(N-1)/2; # Discrete Time Vector
omega0 = pi/10;
theta = pi/2

f = 1.6*sin(omega0+n + theta); # The sinusoid

# Do not open the graphic window in org
figure( 1, "visible", "off");

stem(n,f, "filled", "linewidth", 2, "markersize", 6);
axis([-(N-1+4)/2 (N-1+4)/2 -2 2])
set(gca, "fontsize", 24);
grid on ;
xlabel("Discrete Time Vector [n]");
```

```
print -dpng "-S1400,350" ./image/sin.png;
# Org-Mode specific output
ans = "./image/sin.png";
```




#### 1.3.1.2 Digital vs physical frequency

- Discrete Time:
  - Periodicity: how many samples before the pattern repeats (M)
  - n: no physical dimension
- Physical World:
  - Periodicity: hoq many seconds before the pattern repeats
  - frequency measured in Hz
- Soundcard $T_s$ System Clock
  - A sound card takes ever $T_s$ an new sample from the discrete-time sequence.
  - periodicity of M samples $\rightarrow$ periodicity of $M\ T_s$ seconds
  - real world frequency

$$f = \frac{1}{M\ T_s} Hz \tag{11}$$

- Example
  - usually we choose $F_s$ the number of samples per seconds
  - $T_s = 1/F_s$

$$F_s = 48000\text{e.g. a typical value}$$
$$T_s = 20.8\mu\ s$$
$$f = 440Hz\ \text{, with M} = 110$$

### 1.3.2 The Karplus Strong Algorithm

#### 1.3.2.1 The Moving Average

- simple average (2 point average)

$$m = \frac{a+b}{2} \tag{12}$$

- moving average: take a "local" average

$$y[n] = \frac{x[n] + x[n-1]}{2} \tag{13}$$

- Average a sinusoid

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

**Linear Transformation**

Applying a linear transformation to a sinusoidal input results in a sinusoidal output of the same frequency with a phase shift.

#### 1.3.2.2 Reversing the loop

$$y[n] = x[n] + \alpha\, y[n-1] \rightarrow \text{ The Karplus Strong Algorithm} \tag{14}$$

- **Zero Initial Conditions:**
  - set a start time (usually $n_0 = 0$)
  - assume input and output are zero for all time before $N_0$

## 1.4 Digital Frequency

**Digital Frequency**

$$\begin{aligned} \sin\Big(n\big(\omega + 2k\pi\big)\Big) &= \sin\big(n\omega + \phi\big),\ \text{k in } \mathbb{Z} \\ &= e^{i(\phi + n*2\pi\omega)} \end{aligned} \tag{15}$$

**Complex Exponential**

$$\omega = \frac{M}{N} \times 2 \times \pi \tag{16}$$

## 1.5 The Reproduction Formula

**Reproduction Formula**

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \tag{17}$$

Any signal can be expressed as a linear combination of wighted and shifted pulses.

# 2 Week 2 Module 2: Vector Spaces

## Vector Space

Vector spaces build among others a common framework to work with the four classes of signals:

- Finite Length Signal
- Infinte Length Signal
- Periodic Signal
- Finite Support Signal

Finite length and periodic signal, i.e. the "practical signal processing" live in the $\mathbb{C}^N$ Space. To represent infinite length signals we need something more. We require sequneces to be square-summabe $\sum_{n=-\infty}^{\infty} |x[n]|^2$

| | |
|---|---|
| $\mathbb{R}^2$, $\mathbb{R}^3$ | Euclidean space, geomtery |
| $\mathbb{R}^n$, $\mathbb{C}^n$ | Linear algebra |
| $\ell_2(\mathbb{Z})$ | Square-Summable infinite sequences |
| $L_2([a,b])$ | Square-integrable functions over an interval |

### 2.0.1 Operationl Definitions

## Inner Product

Measure of similarity between vectors

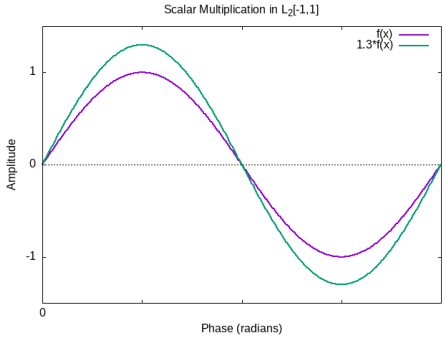
| | |
|---|---|
| **Inner Product** | $\langle \mathbf{x}, \mathbf{y} \rangle := \sum_{n=0}^{N-1} x_n \, y_n$<br>A vector space with an inner product is called an **inner product space** |
| **Inner Product in $\mathbb{R}^2$** | $\langle \mathbf{x}, \mathbf{y} \rangle = x_0 y_0 + x_1 y_1 = \mathbf{x} + \mathbf{y} cos(\alpha)$ |
| **Inner Product in $\mathbb{L}_{[-1,1]}$** | $\langle \mathbf{x}, \mathbf{y} \rangle = \int_{-1}^{1} x(t) y(t) dt$ |
| **Norm of a Vector** | $\mathbf{v} := \langle \mathbf{v}, \mathbf{v} \rangle = \|\|\mathbf{v}\|\|^2$<br>self inner product |
| **Orthogonal** | $\langle \mathbf{p}, \mathbf{q} \rangle = 0$<br>maximal different vectors<br>inner product = 0 |
| **Distance** | $d(x, y) = \mathbf{x} - \mathbf{y}_2$ |

### 2.0.2 Some Examples

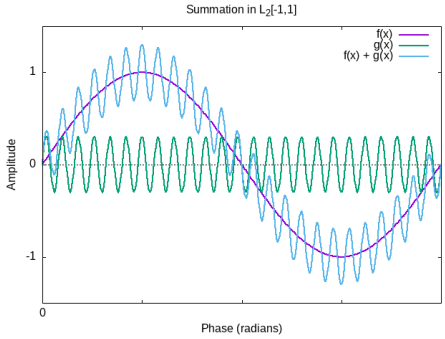
Not all vector spaces have got a graphical representation. The following table shows the graphical representation of vector spaces

| graphical representation | no graphical respresentation |
|---|---|
| $\mathbb{R}^2$ | $\mathbb{C}^N$ for N>1 |
| $\mathbb{R}^3$ | $\mathbb{R}^N$ for N>3 |
| $\mathbb{L}_{[-1,1]}$ | |

Scalar Multiplication in $\mathbb{L}_2[-1,1]$

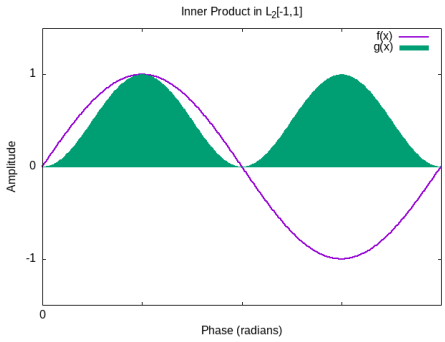



Summation of two Vectors in $\mathbb{L}_2[-1,1]$




Inner Product in $\mathbb{L}_2[-1,1]$ - The Norm:
with x = sin(π\,t)

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




## 2.1 Hilbert Space

A hilbert space is an **inner product space** which fulfills completeness.

## 2.2 Signal Spaces

Finite length signal live in $\mathbb{C}^N$

- all operations well defined and intuitive
- space of N-periodic signals sometimes indicated by $\tilde{\mathbb{C}}^N$

## 2.3 TODO Vecotor Bases

## 2.4 TODO Subspace Approximations

# 3 Week 3 Module 3: Part 1 - Basics of Fourier Analysis

## 3.1 Introduction to Fourier Analysis

### 3.1.1 Sustainable dynamic systems exhibit oscillatory behavior

- A train has got an engine which makes the wheels turn in circular motion
- Waves, ebb and flow can be modeled as sinusoidal fashion
- Musical instruments generates sound by vibrating at a certain fundamental frequency
- Intuitivly: things that don't move in circles can't last
  - bombs
  - rockets
  - human beeings

### 3.1.2 Descriptin of the oscillations in the plane

**Period** $P$

**Frequency** $f = \frac{1}{P}$

**Ordinate** $\sin(ft)$

**Abscissa** $\cos(ft)$

### 3.1.3 Example Sinusoidal Detectors in our Body:

**cochlea** In the inner ear that detects air pressure sinusoids at frequenies from 20 to 20kHz

**retina** In the eye to detect electromagnetic sinusoids with frequency 430THz to 790THz. This is the frequency of lights in the visible spectrum

Humans anlayze complex signals (audio, images) in terms of their sinusoidal components

Frequency Domain semms to be as good a the time domain

### 3.1.4 Fundamental Questions: Can we decompse any signal into sinusoidal elements?

- Yes, Fourier showed us how to do it exactely
- Analysis
  - From time domain to frequency domain
  - Find the contribution of different frequencies
  - Discover "hidden" signal properties
- Synthesis
  - From frequency domain to time domain
  - Create signals with known frequency content
  - Fit signals to specific frequency regions

## 3.2 The Discrete Fourier Transform (DFT)

### 3.2.1 The DFT as a change of basis

### 3.2.2 The Fourier Basis for $\mathbb{C}^N$ in "Signal" Notation

$$w_k[n] = e^{j\frac{2\pi}{N}nk} \text{with } n,k = 0,1,...,N-1 \tag{18}$$

### 3.2.3 The Fourier Basis in Vector Notation

$$\{\mathbf{w}^{(k)}\}_{k=0,1...N-1} \text{with } w_n^{(k)} = e^{j\frac{2\pi}{N}nk},\ n = 0, 1, ...N-1 \tag{19}$$

**N** N Dimension of vector space

**k** Index for different vectors and goes from 0..N-1

**n** Index of element in each vector goes from 0...N-1

### 3.2.4 Definition of the DFT

#### 3.2.4.1 Basis Expansion Vector Notation

##### 3.2.4.1.1 Analysis Formula

$$X_k = \langle \mathbf{w}^{(k)}, \mathbf{x} \rangle \text{ k = 0,...N-1} \tag{20}$$

$X_k$ Coefficent for the new basis. Inner Product of $\mathbf{x}$ with each vector $\mathbf{w}^{(k)}$

$\mathbf{x}$ An arbitrary vector of $\mathbb{C}^N$

$\mathbf{w}^{(k)}$ New basis

##### 3.2.4.1.2 Synthesis Formula

$$\mathbf{x} = \frac{1}{N}\sum_{k=0}^{N-1} X_k \mathbf{w}^{(k)} \text{ k = 0,...N-1} \tag{21}$$

#### 3.2.4.2 TODO Basis Expansion Matrix Form

#### 3.2.4.3 Basis Expansion Signal Notation

- Consider explicitely the operations involved in the transformation
- This notion is particulary useful if you want to consider the algorithmic nature of the transform

##### 3.2.4.3.1 Analysis Formula N-point signal in the frequency domain

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk},\ k = 0, 1, ..N-1$$

$X[k]$ Signal vector in the frequency domain

$x[n]$ Signal vector in the (discrete) time domain

**Reminder** This is the inner Product in explicite form

#### 3.2.4.3.2 Synthesis Formula N-point signal in the time domain

$$x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] e^{j\frac{2\pi}{N}nk},\ k = 0, 1, ..N-1$$

$X[k]$ Signal vector in the frequency domain

$\frac{1}{N}$ Normalisation coeficent

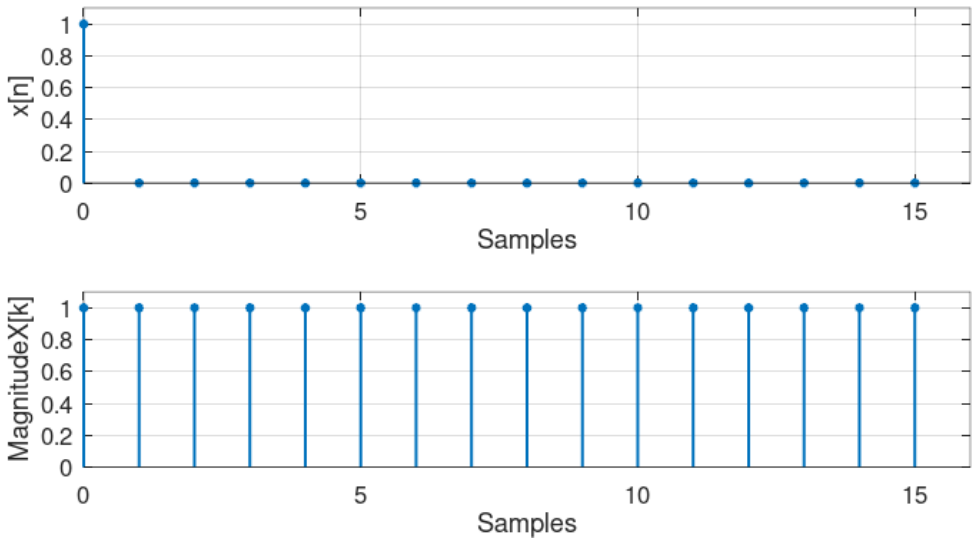
**Reminder** This is the inner Product in explicite fashion

### 3.2.5 Examples

#### 3.2.5.1 DFT of the impulse function

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

$$X[k] = \sum_{n=0}^{N-1} \delta[n] e^{-j\frac{2\pi}{N}nk} = 1$$

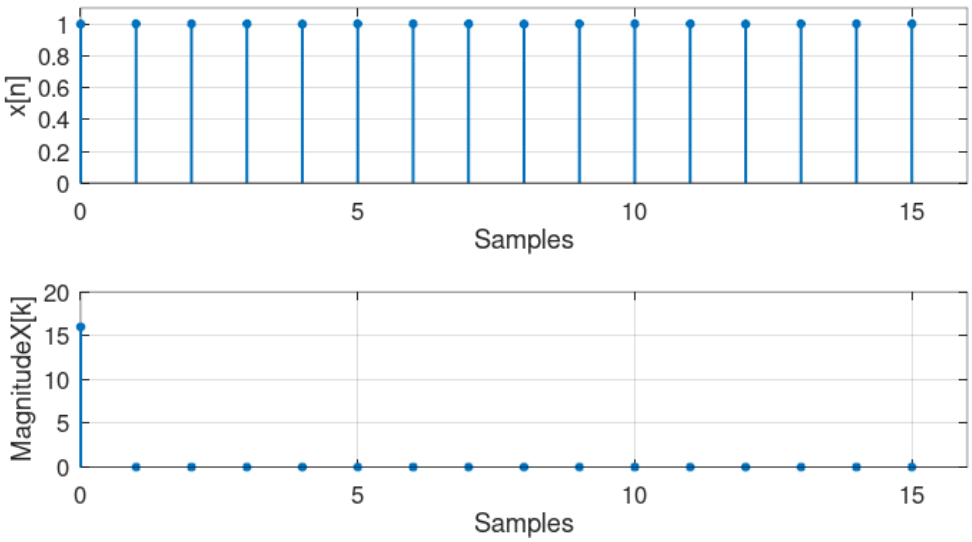



- The delata contains all frequencies over the range of all possible frequencies

#### 3.2.5.2 DFT of the unit step

$$x[n] = 1$$

$$X[k] = \sum_{n=0}^{N-1} e^{-j\frac{2\pi}{N}nk} = N\delta[k]$$




#### 3.2.5.3 DFT Cosine Calculation Problem 1

$$x[n] = 3\cos(2\pi/16 \times n),\ x[n] = \mathbb{C}^{64}$$

1. Determine dimension and fundamental frequency of the signal
   - Dimension of space N = 64
   - Fundamental frequency $\omega = \frac{2\pi}{N} = \frac{2\pi}{64}$

     All frequencies in the fourier basis will be a multiple of the fundamental frequency $\omega$. With this in mind we can start by expressing our sinuoid as a muiltiple of the fundamental frequenncy in space $\mathbb{C}^{64}$.

2. Express the signal as a multiple of the fundamental frequency in space.

$$\begin{aligned} X[n] &= 3\cos(\frac{2\pi}{16}n) \\ &= 3\cos(\frac{2\pi}{64}4n) \\ &= \frac{3}{2}\left[e^{j\frac{2\pi}{64}4n} + e^{-j\frac{2\pi}{64}4n}\right], \text{ with Euler: } cos(\omega) = \frac{e^{j\omega} + e^{-j\omega}}{2} \\ &= \frac{3}{2}\left[e^{j\frac{2\pi}{64}4n} + e^{j\frac{2\pi}{64}60n}\right], \text{ with: } j\frac{2\pi}{64}60n = -j\frac{2\pi}{64}4n + j2\pi n \\ &= \frac{3}{2}\langle w_4[n] + w_{60}[n]\rangle \end{aligned}$$

   - $w_4[n]$ Basis vector number 4
   - $w_{60}[n]$ Basis vector number 60

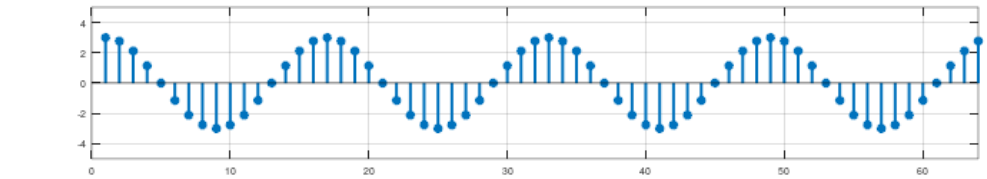
     Now we don't like this minus. So what we're going to do is exploit the fact that we can always add an integer multiple of 2pi to the exponent of the complex exponential. And the point will not change on the complex plane.

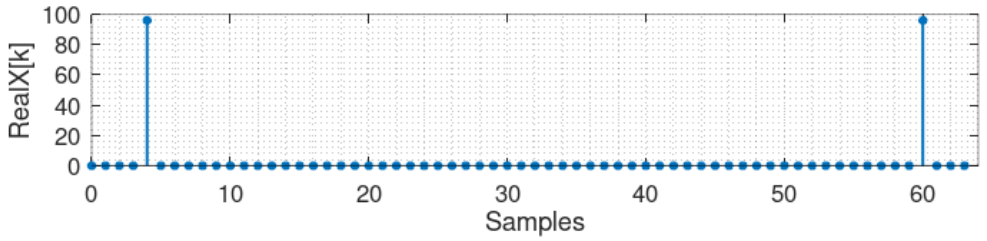
   - **The original signal is now expressed as the sum of two fourier basis vectors**

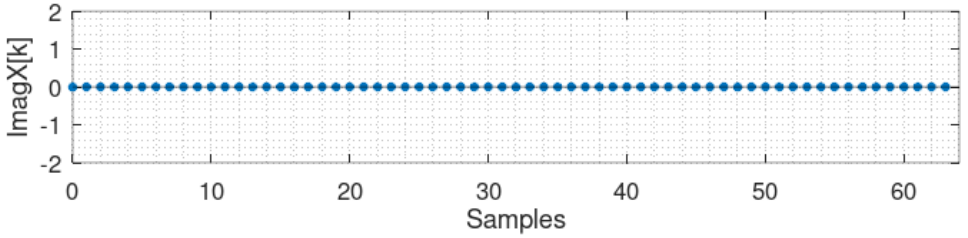
3. Calculate the DFT with the analysis forumla

$$X[k] = \langle w_k[n], x[n] \rangle, \text{ with: } k = 0, 1, ..N-1$$
$$= \begin{cases} 96 & \text{for } k = 4, 60 \\ 0 & \text{otherwise} \end{cases}$$

- $w_k[n]$ Canonical basis vector number k

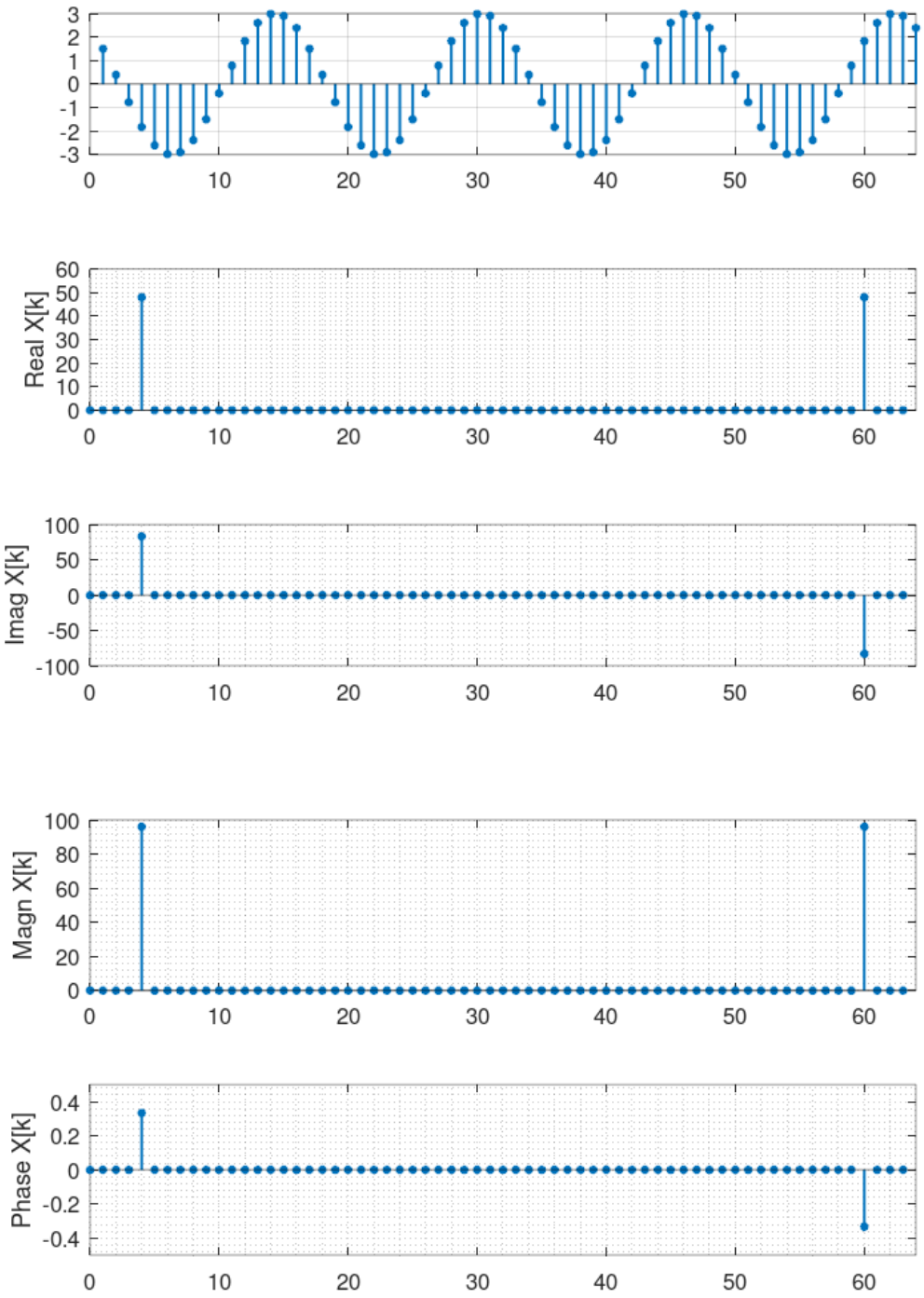





#### 3.2.5.4 DFT Cosine Calculation Problem 2

$$x[n] = 3\ cos(2\ pi/16\ n + pi/3),\ x[n] \in \mathbb{C}^{64}$$
$$X[k] = \begin{cases} 96e^{j\frac{\pi}{3}} & \text{for } k = 4 \\ 96e^{-j\frac{\pi}{3}} & \text{for } k = 96 \\ 0 & \text{otherwise} \end{cases}$$

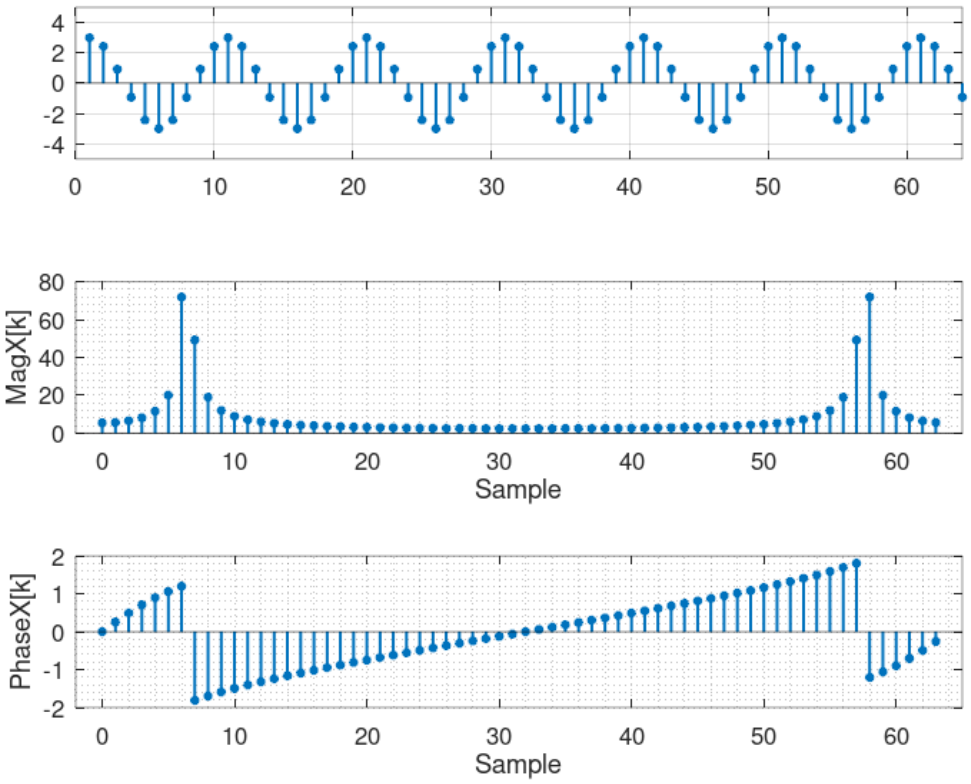



The calcution of the phase just does not work out of the box with octave.

#### 3.2.5.5 DFT Cosine Calculation Problem 3

$$x[n] = 3\ cos(2\ pi/10\ n),\ x[n] \in \mathbb{C}^{64}$$

$$X[k] = \begin{cases} 96e^{j\frac{\pi}{3}} & \text{for } k = 4 \\ 96e^{-j\frac{\pi}{3}} & \text{for } k = 96 \\ 0 & \text{otherwise} \end{cases}$$




### 3.2.6 Properties of the DFT

**Linearity** $DFT\alpha x[n] + \beta y[n] = DFT\alpha x[n] + DFT\beta y[n]$

### 3.2.7 Interpreting a DFT Plot

- Frequency coefficence $< \pi[0...N/2]$ are interpreted as counter clock wise rotation in the plane
- Frequency coefficence $> \pi[N/2...N-1]$ are interpreted as clock wise rotation in the plane
- The fastest frequency of the signal in the vector space is at N/2

**Energy of a Signal**

The square magnitude of the k-th DFT coefficent is proportional to the signal's energy at frequency $\omega = (\frac{2\pi}{N})k$

- Energy concentrated on single frequency (counterclockwise and clockwise combine to give real signal)

$$x1[n] = 3\ cos(2\ pi/16\ n),\ x[n] \in \mathbb{C}^{64}$$
$$x1[n] = u[n] - u[n-4]$$

- For real signals the DFT is symmetric in magnitude
  - $|X[k]| = |X[N-k]|$, for $k = 1, 2, ...[N/2]$
  - For real signals, magnitude plots need only $[N/2] + 1$ points

### 3.2.8 DFT Analysis

#### 3.2.8.1 Daily Temeperature (2920 days)

- The recorded signal

#### 3.2.8.2 TODO Add dayly temp image

- average value (0-th DFT coefficient: 12.3°
- DFT main peak for $k = 8$, value 6.4°C
- 8 cycles over 29920 days
- $period = \frac{2920}{8} = 365 days$
- temperature exursion: 12.3° +/- 12.8°C

The fastest positive frequency of a singnal is at $fracN2$ samples. Since a full revolution of 2 $\pi$ requires N samples, the discrete frequency corresponding with $\frac{N}{2}$) is $\pi$.

#### 3.2.8.3 Labeling Frequency Band Axis

- If "clock" of a System is $T_s$
  - fastest (positive) frequency is $\omega = \pi$
  - sinosoid at $\omega = \pi$ needs two samples to do a full revolution
  - time between samples: $T_s = \frac{1}{F_s}$ seconds
  - real world period for fastest sinosoid: $2T_s$ seconds
  - reald world frequency for fastest sinosoid: $F_s/2$ Hz
- The discrete frequency x of a sinusoid compenent at peak k can be determined as follows:

$$\frac{x}{k} = \frac{N}{2\pi}, \text{ with k=0...N-1} \tag{22}$$

- The real world frequency of a siusoid compenent at peak k can be determined as follows:

$$\begin{aligned} \frac{x}{k} &= \frac{2\pi}{N}, \text{ with k=0...N-1} \\ \frac{f_s}{2} &=\rightarrow \pi, \ f_s \text{ sampling frequency} \\ \frac{x}{k} &= \frac{f_s}{N} \\ x &= \frac{k f_s}{N} \end{aligned}$$

**3.2.8.3.1 Example 1** A DFT analysis of a signal with length $N = 4000$ samples at a frequency $fs = 44.1kHz$ shows a peak at $k = 500$. What is the corresponding frequency in Hz of this digital frequency in Hz.

- Solution

$$\begin{aligned} \frac{x}{k} &= \frac{2\,\pi}{N} \\ x &\rightarrow \frac{2\,\pi\,k}{N} \\ \frac{f_s}{2} &\rightarrow \pi \\ x &= \frac{k}{N} f_s \qquad\qquad = 55125.5 \end{aligned}$$

**3.2.8.3.2 Example 2** Calculation of the corresponding frequency vector for a signal for which its spectrum is analysed with the fourier transform

- Sampling Period: $T_s = 1/1000s$
- Sampling Frequency: $f_s = 1/T = 1000Hz$
- Vector Length $N = 2^10 = 1024$

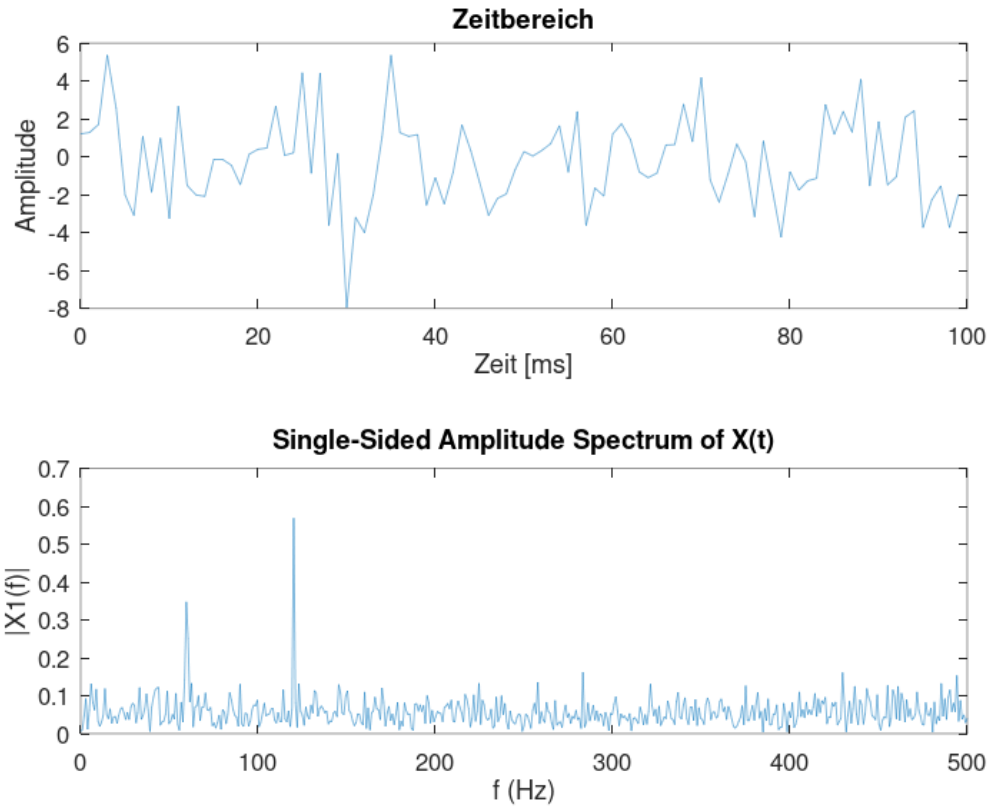
$$\frac{X}{k} = \sum_{n=1}^{N} x[n] e^{-j2\pi(k-1)(\frac{n-1}{N})}$$

$$f(k) = \frac{k-1}{NT}, \text{ corresponding Frequency in Hz}$$

- StackOverflow

```
clear all;
close all;
N  = 1024;   # vector length
Fs = 1000;   # Sample Frequency Fs = 1000Hz
Ts = 1/Fs;   # Sampling Period  Ts = 0.001s
f1 = 60;     # 50Hz
f2 = 120;    # 120Hz

n  = 0:Ts:(N-1)*Ts;                    # time vector
x  = sin(2*pi*f1*n) + sin(2*pi*f2*n);  # a sinusoid signal
xr = x + 2*randn(size(n));             # a noisy signal

X  =fft(xr);                           # FFT
X2 = 1/N*abs(X);                       # FFT magnitude full buffer length
F2 = Fs*(0:(N-1))/N;                   # Frequency vector full buffer length

X1 = X2(1:N/2+1)/2;                    # FFT magniture half buffer lenght
X1(2:end-1) = 2*X1(2:end-1);           # Arranged values
F1 = Fs*(0:(N/2))/N;                   # Frequency vector half buffer length

figure( 1, "visible", "off" )

subplot(2,1,1)
plot(Fs*n(1:100),xr(1:100));
title('Zeitbereich')
ylabel('Amplitude');
xlabel('Zeit [ms]')
set(gca, "fontsize", 24);

subplot(2,1,2)
plot(F1,X1)
title('Single-Sided Amplitude Spectrum of X(t)')
xlabel('f (Hz)')
ylabel('|X1(f)|')
set(gca, "fontsize", 24);


## subplot(2,1,3);
## plot(F2,X2);
## title('Two-Sided Amplitude Spectrum of X(t)')
```

```
## ylabel('|X2(f)|')
## xlabel('Frequenz [Hertz]')
## set(gca, "fontsize", 24);

                                        # Org-Mode specific setting
print -dpng "-S800,600" ./image/eth-example.png;
ans = "./image/eth-example.png";
```




#### 3.2.8.4 DFT Example - Analysis of Musical Instruments

- The fundamental note is the first peak in the spectrum
- The relative size of the harmonics gives the timber or the charachter of an instrument

### 3.2.9 TODO DFT Synthesis

### 3.2.10 DFT Examples

#### 3.2.10.1 Tide Prediction in Venice

#### 3.2.10.2 MP3 Compression

- MP3 compression approx. factor 20 or more
- Compression introduces nois from approximation error
- Noise Shaping : Error shaped as the song in the Fourier domain.
- Perceptual Compression inclueds the human hearing system properties intto compression algorithm

#### 3.2.10.3 Video Signal of the Day: The first man-made signal from outer space

$$f = \frac{\omega f_s}{2\pi}$$

- A multiplication in time domain corresponds to a convolution in frequency domain

## 3.3 The Short-Time Fourier Transform STFT

- STFT is a clever way of using DFT
- Spectrogram, is a graphical way to represent the STFT data

#### 3.3.0.1 The short-time Fourier transform

- DTMF Dual-Tone Multi Frequency dialing
- Time representation obfuscates frequency
- Frequency representation obfuscates time

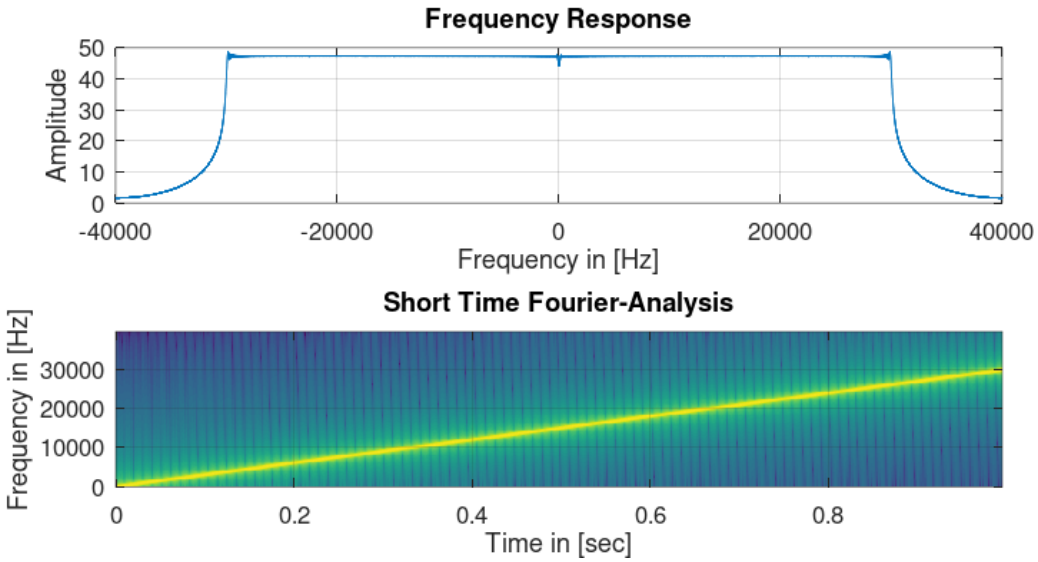
$$x[m;k] = \sum_{n=0}^{L-1} x[m+n]e^{-j\frac{2\pi}{L}nk}$$

  - **m** Starting point of the localiced DFT
  - **k** Is the DFT index

#### 3.3.0.2 ONGOING The spectrogram

- color-code the magnitued: dark is small, white is large
- use $10log_{10}(|X[m,k]|$ to see better (powr in dBs)
- plot spectral slices one after another

### 3.3.1 STFT Example



# 4 Week 4 Module 3: Part 2 - Advanced Fourier Analysise

## 4.1 Discrete Fourier Series DFS

**Discrete Fourier Series**

DFS = DFT with periodicity explicit $\tilde{X}[k] = DFS\{x[n]\}$

- The DFS maps an N-Periodic signal onto an N-Periodic sequence of Fourier coeffients
- The inverse DFS maps $n_{periodic}$ sequence of Fourier coeffiencts a set onto an N-periodic signal
- DFS is an extension of the DFT for periodic sequencies
- A circular time-shift is an natural extension of a shift fo finite length signals.

### 4.1.1 Finite-length time shifts revisted

- The DFS helps us understand how to define time shifts for finite-lenght signals.

test

**For an N-periodic sequence** $\tilde{x}[n]$

$$\begin{aligned}
&\tilde{x}[n-M] \text{ is well-defined for all } M \in \mathbb{N} \\
&DFS\{\tilde{x}[n-M]\} = e^{-j\frac{2\pi}{N}Mk}\tilde{X}[k] \text{ delay factor} \\
&IDFS\left\{ e^{-j\frac{2\pi}{N}Mk}\tilde{X}[k] \right\} = \tilde{x}[n-M] \text{ delay factor}
\end{aligned}$$

**For an N-length signal** $x[n]$

$$\begin{aligned}
&\tilde{x}[n-M] \text{ not well-defined for all } M \in \mathbb{N} \\
&build\ \tilde{x}[n] = x[n\ mod\ N] \Rightarrow\ \tilde{X}[k] = X[k] \\
&IDFT\left\{ e^{-j\frac{2\pi}{N}Mk}X[k] \right\} = IDFS\left\{ e^{-j\frac{2\pi}{N}Mk}\tilde{X}[k] \right\} = \tilde{x}[n-M] = x[(n-M)\ mod\ N]
\end{aligned}$$

**Periodicity**

Shifts for finite-length signals are "naturally" circular

#### 4.1.1.1 Analysis Formula for a N-Periodic Signal in the frequency domain

$$\tilde{X}[k] = \sum_{n=0}^{N-1} \tilde{x}[n]e^{-j\frac{2\pi}{N}nk},\ k \in \mathbb{Z} \tag{23}$$

$X[k]$ Signal vector in the frequency domain

$x[n]$ Signal vector in the (discrete) time domain

**Reminder** This is the inner Product in explicite form

#### 4.1.1.2 Synthesis Formula for a N-Periodic Signal in the time domain

$$\tilde{x}[n] = \frac{1}{N}\sum_{n=0}^{N-1} \tilde{X}[k] e^{j\frac{2\pi}{N}nk},\ k \in \mathbb{Z} \tag{24}$$

$x[n]$ Signal vector in the (discrete) time domain

$X[k]$ Signal vector in the frequency domain

$\frac{1}{N}$ Normalisation coeficent

**Reminder** This is the inner Product in explicite fashion

## 4.2 The Discret-Time Fourier Transform (DTFT)

### 4.2.1 Overview Fourier Transform

- N-Point finite-length siganls: DFT
- N-Point periodic signals: DFS
- Infinite length (non periodic) signals: DTFT

### 4.2.2 Karplus Strong revisted and the DTFT

#### 4.2.2.1 Plotting the DTFT

- Frequencies go from $-\pi$ to $\pi$
- Positive frequencies are on the right hand side of the x-axis
- Negative frequencies are on the left hand side of the x-axis
- Low frequencies are centered around 0
- High frequnecies will be on the extreme of the bound

### 4.2.3 Formal Definition of the DTFT

- $x[n] \in \ell_2(\mathbb{Z})$, the space of square summable infinity sequneces
- define the function of $\omega \in \mathbb{R}$

$$F(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}, \text{ with } \omega = \frac{2\pi}{N} \text{ and } N \to \infty$$

- inversion (when $F(\omega)$ exists):

$$x[n] = \frac{1}{2\,\pi}\int_{-\pi}^{\pi} F(e^{j\omega}); e^{j\ \omega\ n} \mathrm{d}\omega, \text{ with } n \in \mathbb{Z}$$

### 4.2.4 Properties of the DTFT

| | |
|---|---|
| **linearity** | $DTFT\{\alpha x[n] + \beta y[n]\} = \alpha X(e^{j\omega}) + \beta Y(e^{j\omega})$ |
| **timeshift** | $DTFT\{x[n-M]\} = e^{-j\omega M}\ X(e^{j\omega})$ |
| **modulation** | $DTFT\{e^{-j\omega_0 M}\ x[n]\} = X(e^{j\ (\omega-\omega_0)})$ |
| **time reversal** | $DTFT\{x[-n]\} = X(e^{-jw})$ |
| **conjugation** | $DTFT\{x^*[n]\} = X^* X(e^{-j\ \omega})$ |

### 4.2.5 Some particular cases

- if $x[n]$ is symmetric, the DTFT is symmetric: $x[n] = x[-n] \iff X(e^{j\omega}) = X(e^{-j\omega})$
- if $x[n]$ is real, the DTFT is Hemitian-symmetric: $x[n] = x^*[n] \iff X(e^{j\omega}) = X^*(e^{-j\omega})$
- if $x[n]$ is real, the magnitude of th eDTFT is symmetric $x[n] \in \mathbb{R} \implies |X(e^{j\omega})| = |X(e^{-j\omega})|$
- if $x[n]$ is real and symmetric, $X(e^{j\omega})$ is also real and symmetric

## 4.3 TODO Sinusoidal Modulation

# 5 Week 5 Module 4: Part 1 Introduction to Filtering

## 5.1 Linear Time-Invariant Systems

**LTI System**

Linearity and Time Invariance taken together: A Linear Time Invariant System is completely charachterized by its response to the input in particular by its the Impulse Response .

### 5.1.1 Linearity

Linearity is expressed by the equivalence

$$\mathfrak{H}\{\alpha\, x_1[n] + \beta\, x_2[n]\} = \alpha\, \mathfrak{H}\{x_1[n]\} + \beta\, \mathfrak{H}\{x_2[n]\} \tag{25}$$

- Fuzz-Box, example for a none linear device

#### 5.1.1.1 TODO Add calculation examples

### 5.1.2 Time invariance

- The system behaves the same way independently of when a it's switched on

$$y[n] = \mathfrak{H}\{x[n]\} \Leftrightarrow \mathfrak{H}\{x[n - n_o]\} = y[n - n_o] \tag{26}$$

- Wah-Pedal, example of a time variant device

#### 5.1.2.1 TODO Add calculation examples

### 5.1.3 Convolution

The impulse response is the output of a filter when the input is the delta function.

$$h[n] = \mathfrak{H}\{\delta[n]\} \tag{27}$$

**Impulse Response**

Impulse response fully characterize the LTI system!

We can always write

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \tag{28}$$

by linearity and time invariance

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k] \tag{29}$$

$$= x[n] * h[n] \tag{30}$$

Performing the convolution algorithmically

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

**Ingredients**
- a sequence x[m]
- a second sequence h[m]

**The Recipe**
- time-reverse h[m]
- at each step n (from $-\infty \, to \, \infty$):
  - center the time-reversed h[m] in n (i.e. by shift -n)
  - compute the inner product

Furthermore, the convolution can be defined in terms of the inner product between two sequencies.

$$\begin{aligned}(x * y)[n] &= \langle x^*[k], y[n-k] \rangle \\ &= \sum_{n=-\infty}^{\infty} x[k]y[n-k]\end{aligned}$$

## 5.2 Filtering in the Time Domain

For the convolution of two sequencies to exist, the convolution sum must be finite i.e. the both sequencies must be absolutely summable

### 5.2.1 The convolution operator

**Linearity**
$$x[n] * (\alpha \cdot y[n] + \beta \cdot w[n]) = *\alpha \cdot x[n] * y[n] + \beta \cdot x[n] * w[n])$$
$$w[n] = x[n] * y[n] \iff x[n] * y[n-k] = w[n-k]$$

**Commutative**
$$x[n] * y[n] = y[n] * x[n]$$

**Associative**
$$(x[n] * y[n]) * w[n] = x[n] * (y[n] * w[n])$$

### 5.2.2 Convolution and inner Product

$$x[n] * h[n] = \langle \, h^*[n-k], x[k] \rangle$$

Filtering measures the time-localized similarity between the input sequence and a prototype sequence - the time reversed impulse response.
In general the convolution operator for a signal is defined with repsect to the inner product of its underlaying Hilbert space:

**Square Summable Sequence** $\ell_2(\mathbb{Z})$ $x[n] * h[n] = \langle \, h^*[n-k], x[k] \rangle$

**N-Periodic Sequence** $\tilde{x}[n] * \tilde{y}[n]) \sum_{k=0}^{N-1} \tilde{x}[n-k]\tilde{y}[k]$

**Square Integrable Function** $L_2([-\pi, \pi])$ $X(e^{j\omega}) * Y(e^{\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\sigma}) \cdot Y(e^{j(\omega-\sigma)})$

### 5.2.3 Properties of the Impulse Response

**Causality** A system is called causal if its output does not depend on futre values of the input. In practice a causual system is the only type of "real-time" syste we can actually implement.

**Stability** A system is called bounded-input bounded-output stabel (BIBO stable) if its output is bounded for all bounded input sequencies. **FIR** Filter are always stable, since only in the convolution sum only a finite number of terms are involved.

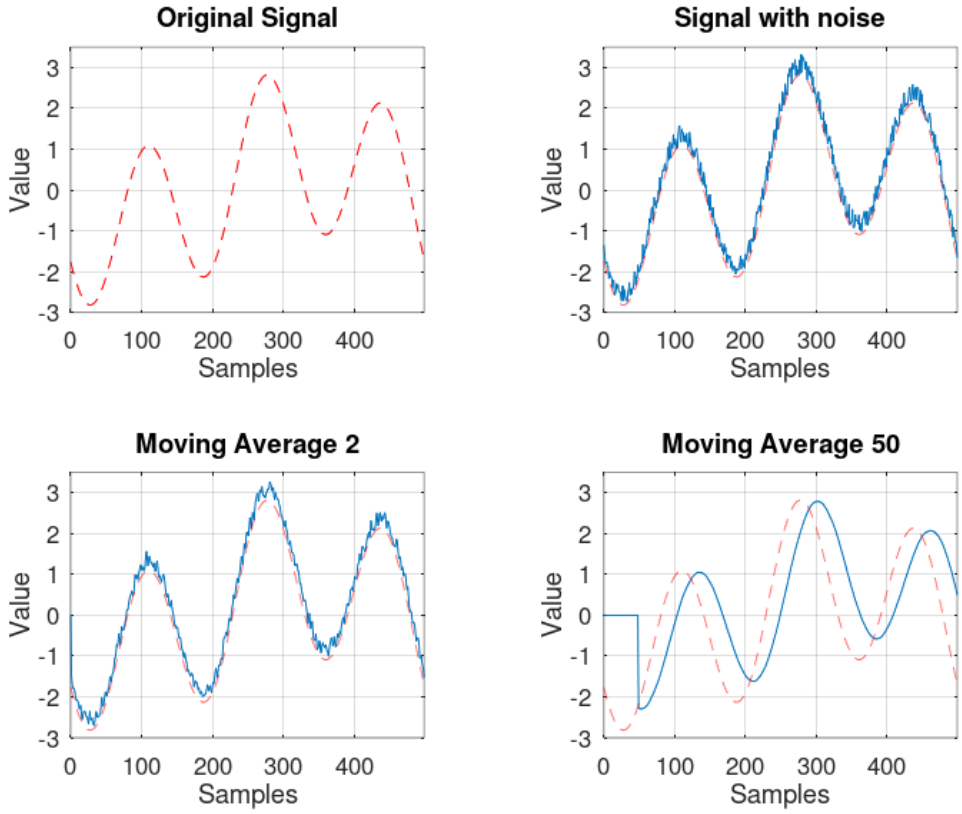
### 5.2.4 Filtering by Example

#### 5.2.4.1 FIR Filter: Moving Average

Typicale filtering scenario: denoising

- idea: replace each sample by the local average. Average are useually good to eliminate random variation from which you don't know mutch about it.
- for instance: $y[n] = (x[n] + x[n-1])/2$
- more generally:

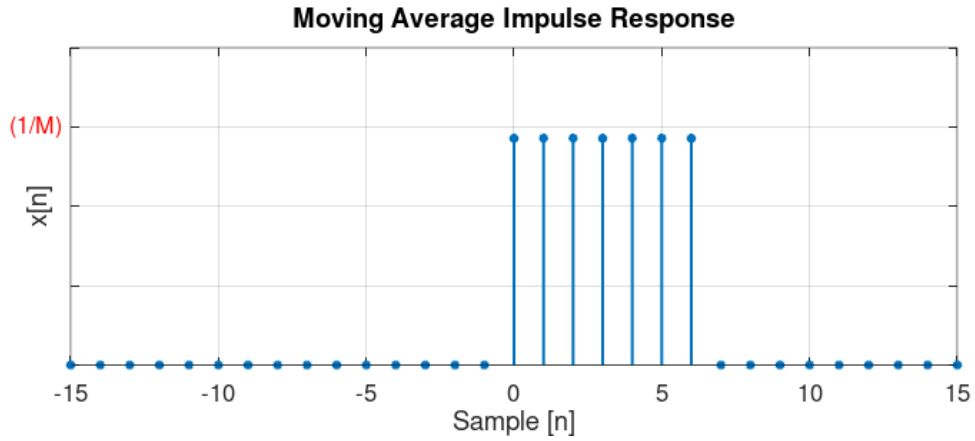
$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$




##### 5.2.4.1.1 Impulse Response

$$h[n] = \frac{1}{M} \sum_{k=0}^{M-1} \delta[n-k] \begin{cases} \frac{1}{M} & \text{for } 0 \le n < M \\ 0 & \text{otherwise} \end{cases}$$

```
function [x,n] = ma_impresp(M,n1,n2)
% Generates x(n) = delta(n); 0 <= M
% -----------------------------------------
% [x,n] = stepseq(n0,n1,n2)
%
  n = [n1:n2]; x = [ (n >= 0) & !((n-M) >= 0) ]./M;
end
```




#### 5.2.4.1.2 MA Analysis

- soomthin effect is proportional to M
- number of operations and storage also proportional to M

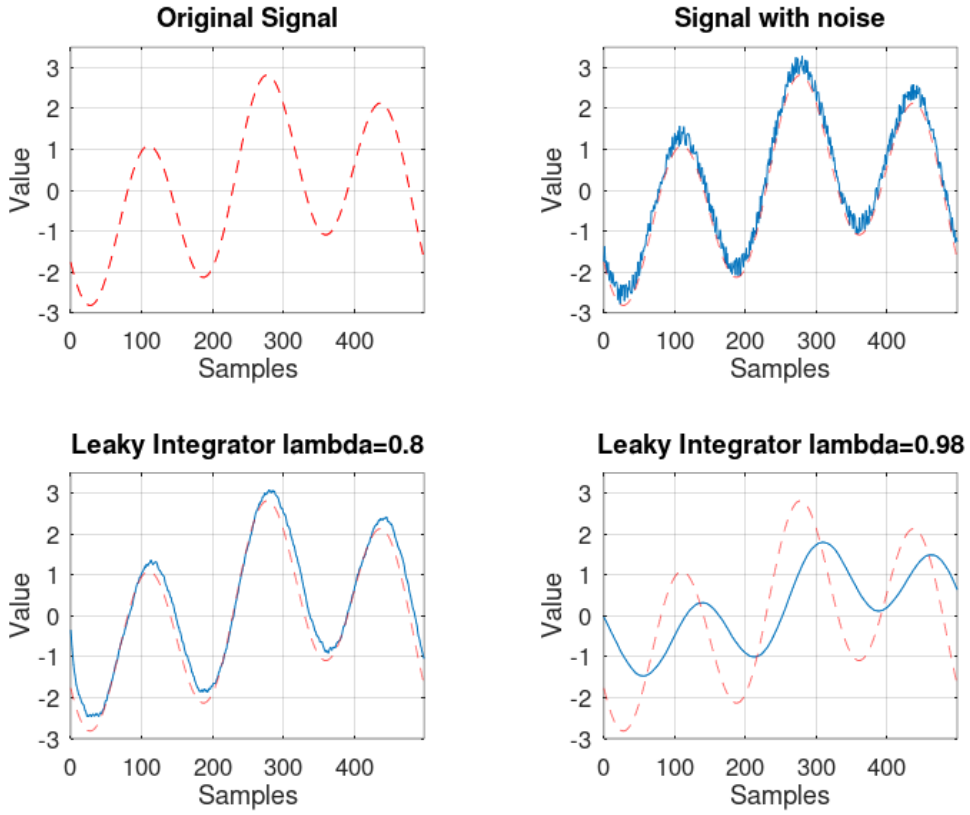
#### 5.2.4.1.3 From the MA to first-order recursion

$$y_{M[n]} = \sum_{k=0}^{M-1} x[n-k] = x[n]X \sum_{k=1}^{M-1} x[n-k]$$
$$M_{y_{M[n]}} = x[n] + (M-1)y_{M-1}[n-1]$$
$$y_M[n] = \frac{M-1}{M} y_{M-1}[n-1] + \frac{1}{M}x[n]$$
$$y_M[n] = \lambda y_{M-1}[n-1] + (1-\lambda)x[n],\ \lambda = \frac{M-1}{M}$$

### 5.2.4.2 IIR Filter: The Leaky Integrator

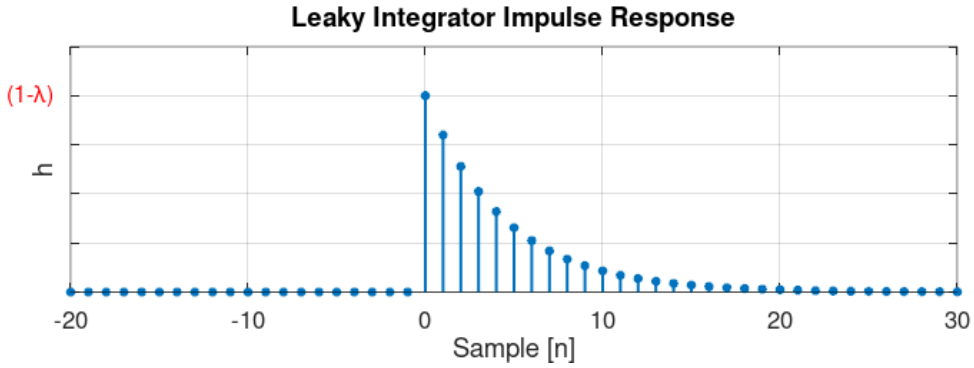
- when M is large, $y_{M-1}[n] \approx y_M[n]$and $(\lambda \approx 1)$
- the filter becomes: $y[n] = \lambda y[n-1] + (1-\lambda)x[n]$
- the filter is now recursive, since it uses its previous output value

```
function y = lky_impresp(a,b,lambda,x)
% Generates x(n) = a^n
% ------------------------------------------
% [x,n] = lky_impresp(a,b, lambda, x)
% y[n] -lambda y[n-1] = (1-lambda) x[n]
% a = [1, -lambda];
% b = [(1-lambda)];

  b = [1-lambda];
  a = [1, -lambda];
  y = filter(b,a,x);
end
```




**5.2.4.2.1 Impulse Response** For the impulse we just need to plug the delta function

$$\begin{aligned} h[n] &= (\lambda y[n-1] + (1-\lambda))\delta[n] \\ &= (1-\lambda)\lambda^n u[n] \end{aligned}$$




The peak at n=0 is $1-\lambda$.

**5.2.4.2.2 The leaky integrator why the name**

- Discrete Time integrator is a boundless accumulator

$$\begin{aligned} y[n] &= \sum_{k=-\infty}^{n} x[k] \\ &= y[n-1] + x[n] \Rightarrow \text{ almost leaky integrator} \end{aligned}$$

To prevent "explosing" we scale the accumulator with $\lambda$:

| | |
|---|---|
| $\lambda y[n-1]$ | keep only a fraction $\lambda$ of the accumulated value so far and forget ("leak") a fraction $\lambda - 1$ |
| $(1-\lambda)x[n]$ | add only a fraction $1-\lambda$ of the current value to the accumulator. |

So we get the leaky integrator from the accumulator

$$y[n] = \lambda \cdot y[n-1] + (1-\lambda) \cdot x[n] \Rightarrow \text{ almost leaky integrator}$$

## 5.3 Classification of Filters

**FIR** Finite Impulse Response Filter

- Impulse response has finite support
- only a finite number of samples are involved in the computation of each output
- Example: Moving Average Filter

**IIR** Infinite Impulse Response Filter

- Impulse response has inifinte 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
- Example: The Leaky Integrator

**Casual**

- impulse response is zero for n $< 0$
- only past samples are involved in the computation of each output sample
- causul filters can work "on line" since they only need the past

**Noncasual**

- impulse response in nonzero for some (or all) n $< 0$
- can still be implemented in a offline fashing (e.g. image processing)

## 5.4 Filter Stability

**FIR Filter**

FIR filters are always stable

because their impuls response only contains a finite number of non-zero values, and therefore the sum of their absolute values will always be finite.

## 5.5 Frequency Response

### 5.5.1 References

- Signal and System for Dummies: Frequency Response

### 5.5.2 Eigensequence

If a complex exponential is applied to a LTI filter its response is the DTFT of the impulse response of the LTI filter times the complex exponential.

$$
\begin{aligned}
x[n] &= e^{j\omega_0 n} \\
y[n] &= \mathfrak{H}\{x[n]\} \\
y[n] &= x[n] * h[n] \\
y[n] &= e^{j\omega_0 n} * h[n] \\
y[n] &= H(e^{j\omega_0})e^{j\omega_0 n}
\end{aligned}
$$

- DTFT of impulse response determinse the frequency characteristic of a filter
- Complex exponential are eignesequences of LTI systems, i.e. linear filters cannot change the frequency of a sinusoid.

### 5.5.3 Magnitude and phase

$$
\begin{aligned}
&\text{if } H(j^{j\omega_0}) = Ae^{j\theta}, \text{ then} \\
&\mathfrak{H}\{e^{j\omega_0 n}\} = Ae^{j(\omega_0 n+\theta)}
\end{aligned}
$$

| **amplitude** | **A** | **phase shift** | $\theta$ |
|---|---|---|---|
| amplification | $>1$ | delay | $< 0$ |
| attenuation | $0 \leq A < 1$ | advancment | $> 0$ |

### 5.5.4 The convolution theorem

The convolution theorem summerizes this result in

$$DTFT\{x[n] * h[n]\} = X(e^{j\omega})H(e^{j\omega})$$

### 5.5.5 Frequency response

The DTFT of the impulse response is called the frequency response

$$H(e^{j\omega}) = DTFT\{h[n]\}$$

| **magnitude** | $\mid H(e^{j\omega}) \mid$ | **phase** |
|---|---|---|
| amplification | $> 1$ | overall shape and |
| attenuation | $< 1$ | phase changes |

### 5.5.6 Example of Frequency Response: Moving Average Filter

The difference equation from M-point averager is

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

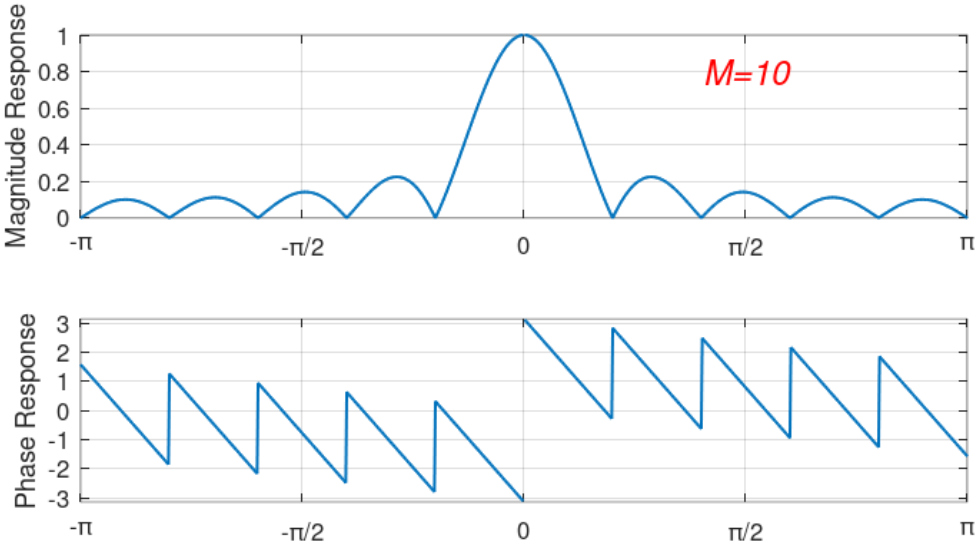
The Frequency response of the moving average filter

$$H(e^{j\omega}) = \frac{1}{M}\sum_{k=0}^{M-1} e^{-j\omega k} = \frac{1}{M}\sum_{k=0}^{M-1} \left(e^{-j\omega}\right)^k$$
$$= \frac{1}{M}\frac{(1-e^{-j\omega M})}{(1-e^{j\omega})}$$

- The frequency response is composed of a linear term $e^{-j\omega\frac{M-1}{2}}$ and $\pm\pi$ due to the sign changes of $\frac{sin(\frac{\omega}{2}M)}{sin(\frac{\omega}{2}M)}$

The Magnetute response of the moving average filter

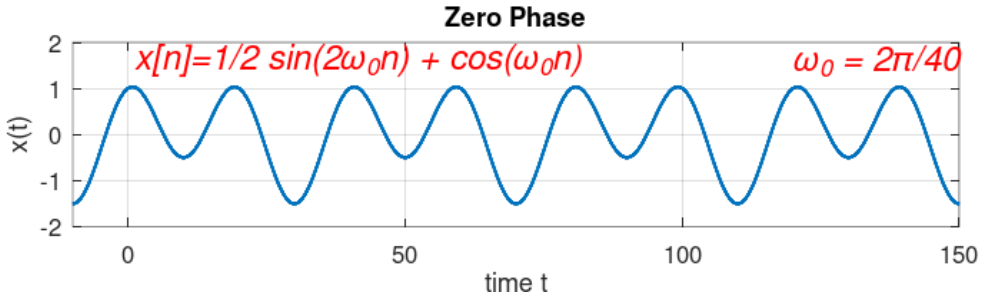
$$H(e^{j\omega}) = \frac{1}{M}\left|\frac{sin(\frac{\omega}{2}M)}{sin(\frac{\omega}{2})}\right|$$



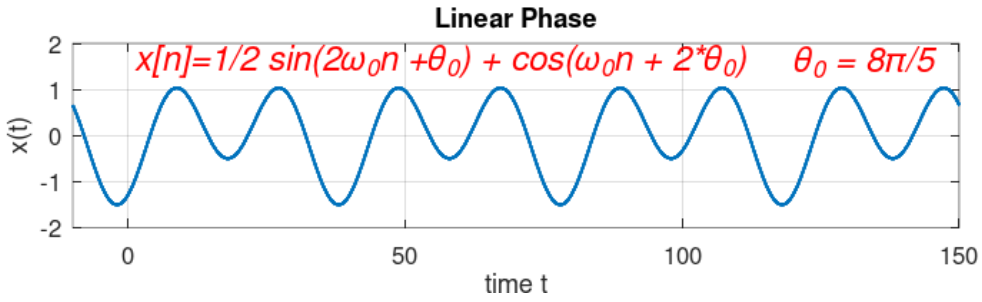

### 5.5.7 Phase and signal shape

To understand the effects of the phase on a signal is to distinguihs three different cases

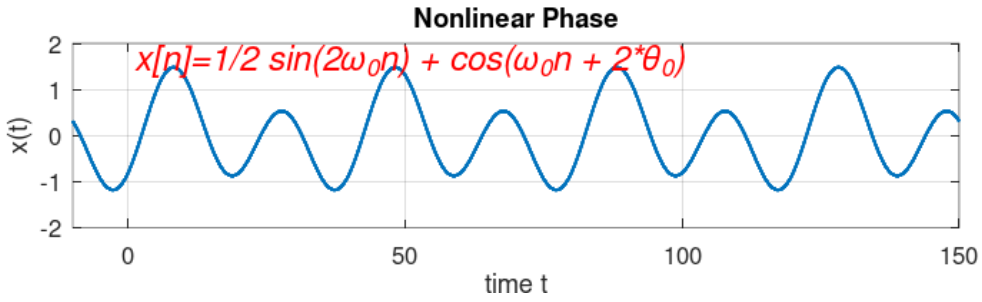
- zero phase: the spectrum is real: $\angle H(e^{jw}) = 0$




- linear phase: the phase is proportional to the frequency via a real factor, d: $\angle H(e^{jw}) = d\omega$ the phase is proportional to the frequency of the sinusoid. The net effect is a shift of the signal if the phase component is porportional to the frequency.




- non linear phase: which covers all the other properties now the shape of the signal in the time domain changes.




**Spectrum**

The spectrum of all three signals x[n] remains exactely the same in magnitude.

### 5.5.8 Linear Phase

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

### 5.5.9 Moving Average is linear Phase

$$\begin{aligned} H(e^{j\omega}) &= A(e^{j\omega})e^{-j\omega d} \\ &\Rightarrow A(e^{j\omega}) \text{: pure real term} \\ &\Rightarrow e^{-j\omega d} \text{: pure phase term} \\ &= \frac{1}{M}\frac{sin(\frac{\omega}{2}M)}{sin(\frac{\omega}{2}M)} e^{-j\omega\frac{M-1}{2}} \Rightarrow \frac{M-1}{2} = d \end{aligned}$$

### 5.5.10 Example of Frequency Response: Leaky Integrator

The Frequency response of the leaky integrator

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

Finding the magnitude and phaser requires a little algebra
From Complex Algebra

$$\frac{1}{a+jb} = \frac{1-jb}{a^2+b^2}$$

So that if $x = \frac{1}{a+jb}$

$$|x|^2 = \frac{1}{a^2+b^2}$$
$$\angle x = tan^{-1}\left[-\frac{a}{b}\right]$$

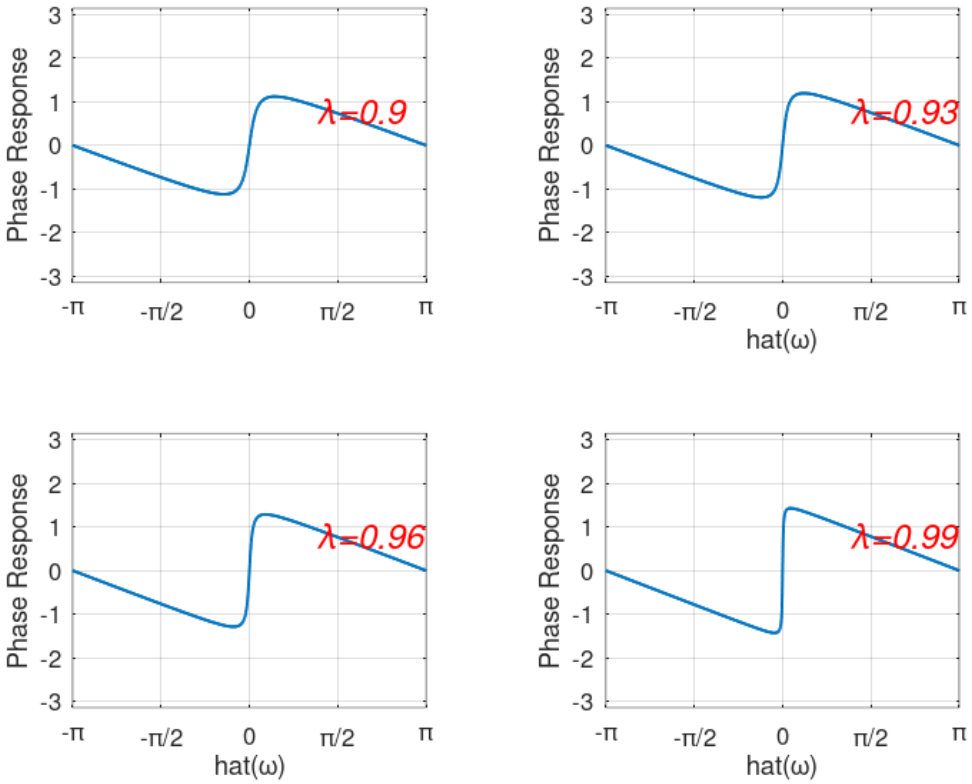
$$H(e^{j\omega}) = \frac{1-\lambda}{(1-\lambda cos\omega) - jsin\omega}$$

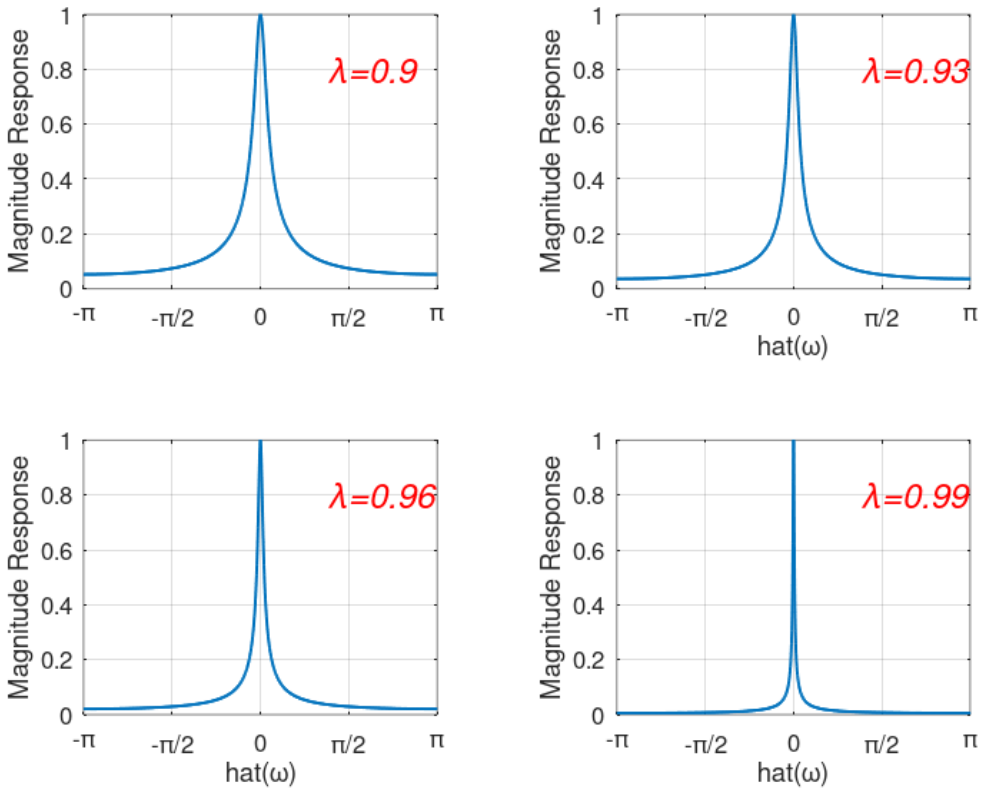
so that:

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

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

The phase is nonlinear in this case






### 5.5.11 TODO Example of Frequency Response: Karplus Strong Algorithm

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

The Karplus-Strong algorithm is initialized with a finite support signal x of support M. And then we use a feedback loop with a delay of M taps. To qproduce multiple copies of the original finite support signal, scaled by an exponentially decaying factor alpha.

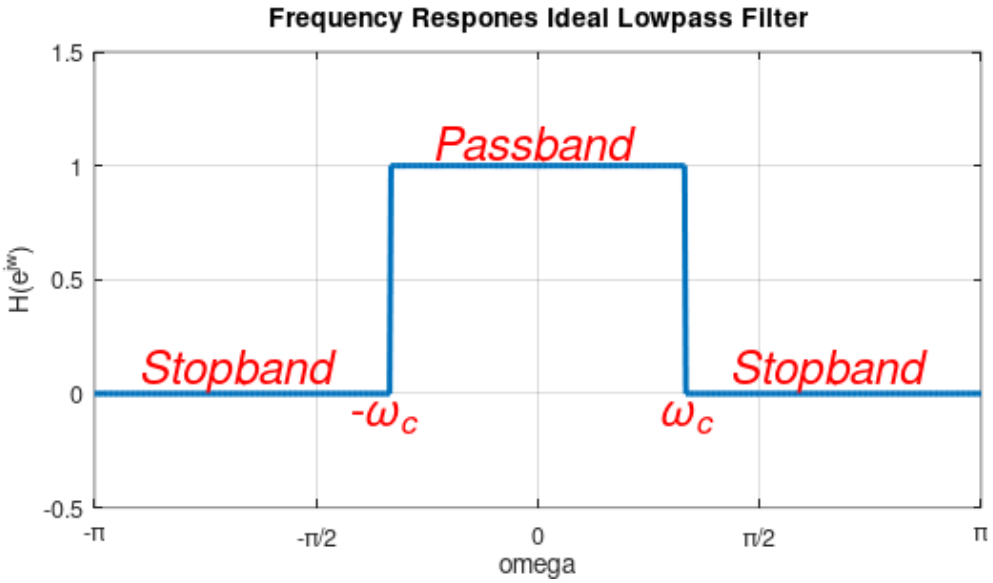
#### 5.5.11.1 With Sawtooth Wave

$$\tilde{X}(j\omega)W(j\omega) = e^{-j\omega}\left(\frac{M+1}{M-1}\right)\frac{1 - e^{-j(M-1)\omega}}{\left(1 - e^{j\omega}\right)^2} - \frac{1 - e^{-j(M+1)\omega}}{\left(1 - e^{j\omega}\right)^2}$$

$$X(j\omega)W(j\omega) = \frac{1}{1 - \alpha e^{-j\omega M}}$$

## 5.6 Ideal Filters

### 5.6.1 The ideal lowpass filter frequency response




### 5.6.2 Ideal lowpass filter impulse response

- Lets low frequencies go through
- Attenuates i.e. kills high frequencies

**Cut off Frequency** $\omega_c$ - the frequency response transitions from 1 to zero

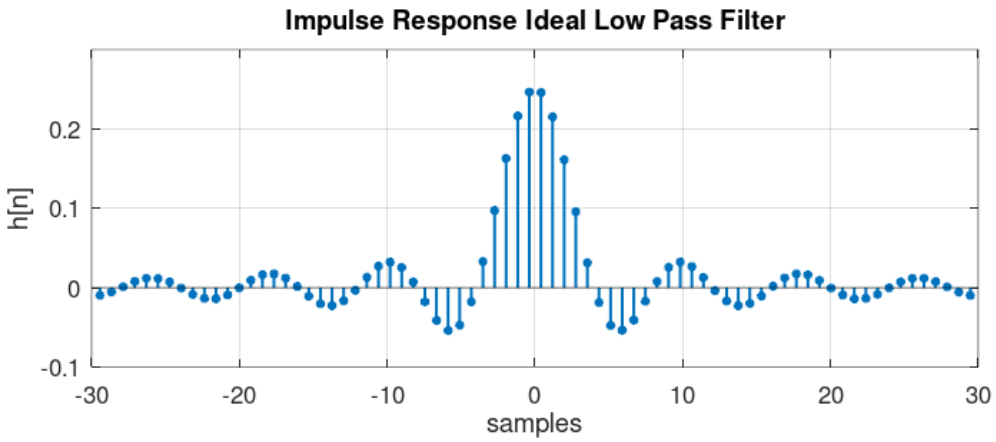
**Passband** $\omega_b = 2\omega_c$

$$H(e^{j\omega}) = \begin{cases} 1 & \text{for } |\omega| \leq \omega_c \\ 0 & \text{otherwise} \end{cases}$$

- perfectly flat passband
- infinite attenuation in stopband
- zero-phase (no delay)

Calculation of the impulse response from the frequency response of an ideal low pass filter. Impulse Respones

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




- from Mathworks
- The impulse response has infinite support to the right and to the left
- Independant of how the convolution is computed, it will always take an inifintie number of operations.
- The impulse response decays slowly in time $\left(\frac{1}{n}\right)$, we need a lot of samples for a good approximation.

#### 5.6.2.1 Impulse Response: From normlized Algorithm to Octave Implementation

$$\begin{aligned}\frac{sin(\omega_c\, n)}{\pi\, n} &= \frac{\omega_c}{\pi}.\cdot sinc(n\frac{\omega_c}{\pi});\\ &= \frac{\frac{\pi}{c}}{\pi}.\cdot sinc(n\frac{\frac{\pi}{c}}{\pi});\\ &= \frac{1}{c}.\cdot sinc(n\frac{1}{c});\\ &= \frac{1}{c}.\cdot sinc(\frac{n}{c});\end{aligned}$$

#### 5.6.2.2 The sinc-rect pair:

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

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

- rect is the indicator function from $-\frac{1}{2}$ to $\frac{1}{2}$

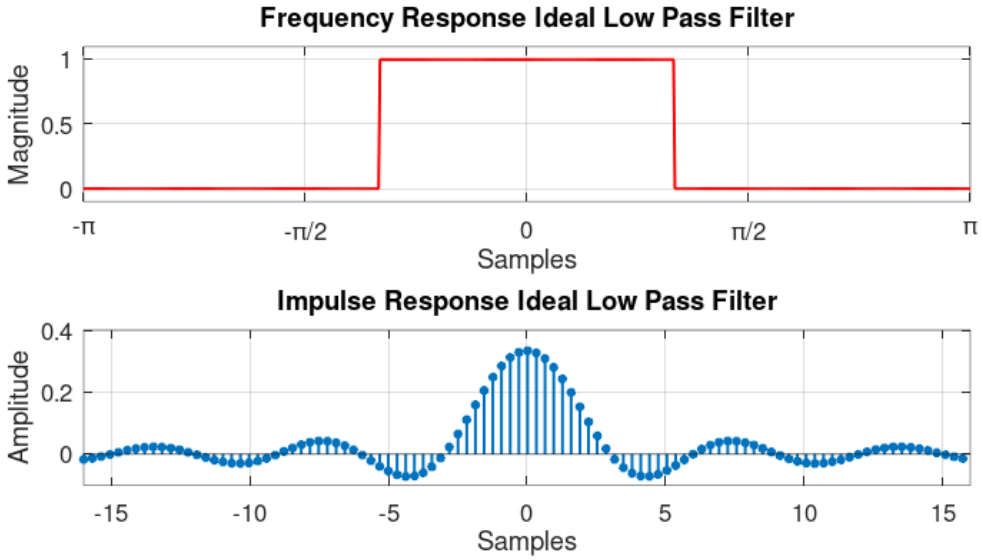
**5.6.2.3 Canonical form of the ideal low pass filter** The sinct-rect pair can be written in canonical form as follow: $~$

$$H(e^{j\,\omega}) = rect\left(\frac{\omega}{2\,\omega_c}\right) \quad \overleftrightarrow{DTFT} \quad \frac{\omega_c}{\pi}\, sinc\left(\frac{\omega_c}{\pi}\, n\right) = h[n]$$

- The Impulse response is normalized by $\frac{\omega_c}{\pi}$

### 5.6.3 Example

**Calculation of the impulse- and frequency response for an ideal low pass filter with $\omega_c \frac{\pi}{3}$**

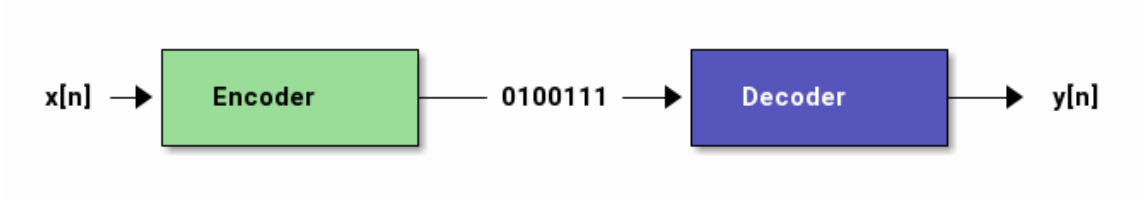



### 5.6.4 TODO Ideal filters derived from the ideal low pass filter

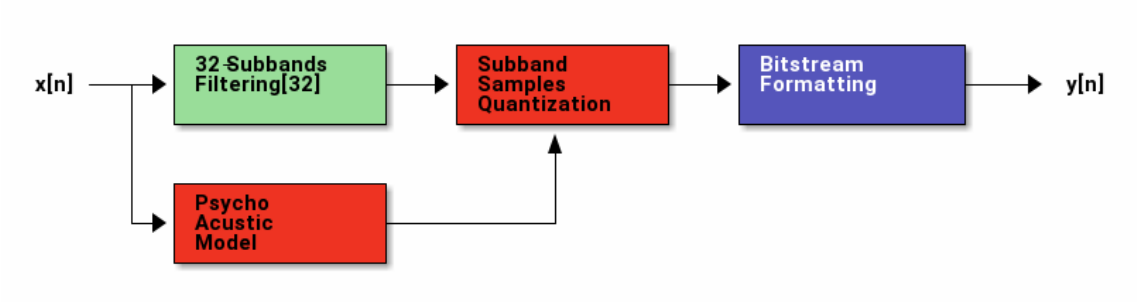
### 5.6.5 TODO Demodulation revisted

## 5.7 MP3 Encoder

- **Goal:** Reduce number of bits to represent original signal x[n]
- MP3: Motion Picture Expert G3roup




- **Lossy Compression:** $x[n] \neq y[n]$
- Put noise where not perceptible by human ear
- *Example:*Raw Storage Consumption DVD
  - Sample Rate: 48kHz
  - Bits per Sample: 16
  - Bit Rate: $\frac{48000\text{samples}}{second} \frac{16bits}{samples} = 768kbits/s$
  - Duration: 60s
  - Mono Raw Data Storage Usage: $60s * 76.8kbits/s = 46Mbit = 5.8MByte$
  - Stereo Raw Data Storage Usage: $2 * 5.8MBytes = 12MBytes$
  - MP3 Compressed Storage Usage: $1.5MBytes$




- Clever Quantiziation Scheme: Number of bits allocated to each subband is dependent on the perceptual importance of each sub-band with respect to overall quality of the audio wave-form
- Masking Effect of the human auditory system.

### 5.7.1 Psycho Acoustic Model, How it Works

- The psycho acoustic model is not part of the mp3 standard
- calculate the minimum number of bits that we need to quantize each of the 32 subband filter outputs, so that the perceptual distortion is as little as possible

step 1 Use FFT to estimate the energy of the signal in each subband

step 2 Distinguish beween tonal (sinusoid like) and non-tonal (nois-like) compnent

step 3 Determine indicidual masking effect of tonal and non-tonal component in each critical band

step 4 Determine the total masking effect by summing the individual contirbution

step 5 Map this total effect to the 32 subbands

step 6 Determine bit alloction by allocating priority bits to subbands with lowest singal-to-mask ratio

### 5.7.2 Subband Filter

$$h_i[n] = h[n]cos(\frac{pi}{64}(2i+1)(n-16))$$

## 5.8 Programing Assignment 1

```
import matplotlib
import numpy as np
matplotlib.use('Agg')
import matplotlib.pyplot as plt


def scaled_fft_db(x):
    """ ASSIGNMENT 1:
        Module 4 Part 1:
        Apply a hanning window to len(x[n]) = 512
    """

    N = len(x)              # number of samples
    n = np.arange(N)        # time vector
    # a) Compute a 512-point Hann window and use it to weigh the input data.
    sine_sqr = np.sin((np.pi*n)/(N-1))**2     # sin(x)^2 = 1/2*(1 - cos(2x))
    c = np.sqrt(511/np.sum(sine_sqr))
    w = c/2 * (1 - np.cos((2 * np.pi * n)/(N - 1)))
```

```
    # b) Compute the DFT of the weighed input, take the magnitude in dBs and
    #    normalize so that the maximum value is 96dB.
    y = w * x
    Y = np.fft.fft(y) / N
    # c) Return the first 257 values of the normalized spectrum
    Y = Y[0: np.int(N/2+1)]
    # Take the magnitude of X
    Y_mag = np.abs(Y)
    nonzero_magY = np.where(Y_mag != 0)[0]

    # Convert the magnitudes to dB
    Y_db = -100 * np.ones_like(Y_mag)    # Set the default dB to -100
    Y_db[nonzero_magY] = 20*np.log10(Y_mag[nonzero_magY])  # Compute the dB for nonzero magnitude ind

    # Rescale to amx of 96 dB
    max_db = np.amax(Y_db)
    Y_db = 96 - max_db + Y_db

    return Y_db

def test():
    N = 512
    n = np.arange(N)
    x = np.cos(2*np.pi*n/10)

    # Y = scaled_fft_db(x)
    Y = scaled_fft_db(x)

    fig=plt.figure(figsize=(6,3))
    plt.semilogy(abs(Y))
    plt.grid(True)

    fig.tight_layout()
    plt.savefig('image/python-matplot-fig-04.png')
    return 'image/python-matplot-fig-04.png' # return filename to org-mode

return test()
```

# 6 Week 6 Module 4 Part 2: Introduction to Filtering

- First strategy of filter design: Imitation
  - uuImpulse truncation
  - Window Method
  - Frequency Sampling

  Trying to replicate the structure of either the impulse response or the frequency response of ideal filters.

## 6.1 Filter Design Part 1 (FIR Filter)

- An ideal filter is not realizable in practice because the impulse response is a two-sided infinite support sequence.

### 6.1.1 Reference

- The Scientist and Engineers Guide to DSP: Recurscive Filter

### 6.1.2 Impulse truncation

**Impulse Truncation**

1. Pick $\omega_c$
2. Compute ideal impulse response h[n] (analytically)
3. truncate h[n] to a finite-support $\hat{h}[n]$
4. $\hat{h}[n]$ defines an FIR filter

FIR approximation of lenght M = 2N+1

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

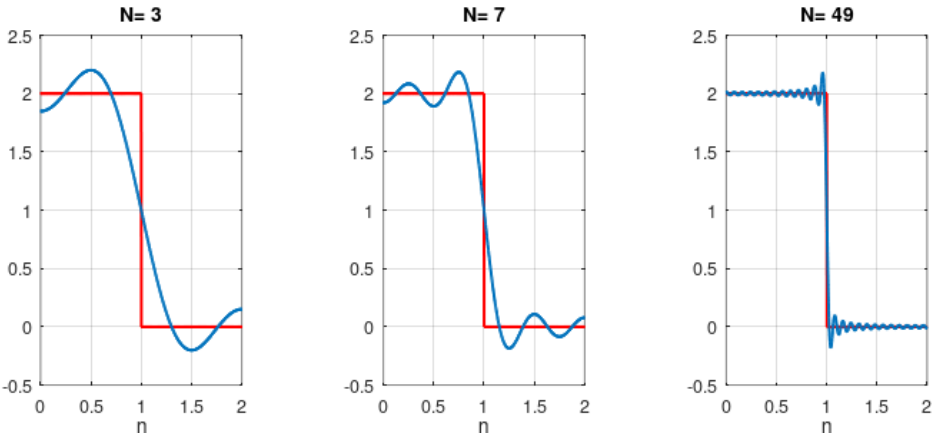
- **Why approximation by truncation could be a good idea** A justification of this method is the computation of the mean square error:

$$\begin{aligned} MSE &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega}) - \hat{H}(e^{j\omega})|^2 d\omega \\ &= ||H(e^{j\omega}) - \hat{H}(e^{j\omega})||^2 \\ &= ||h[n] - \hat{h}[n]||^2 \\ &= \sum_{n=-\infty}^{\infty} |h[n] - \hat{h}[n]|^2 \end{aligned}$$

The means square error MSE is minimized by symmetric impulse truncation around zero

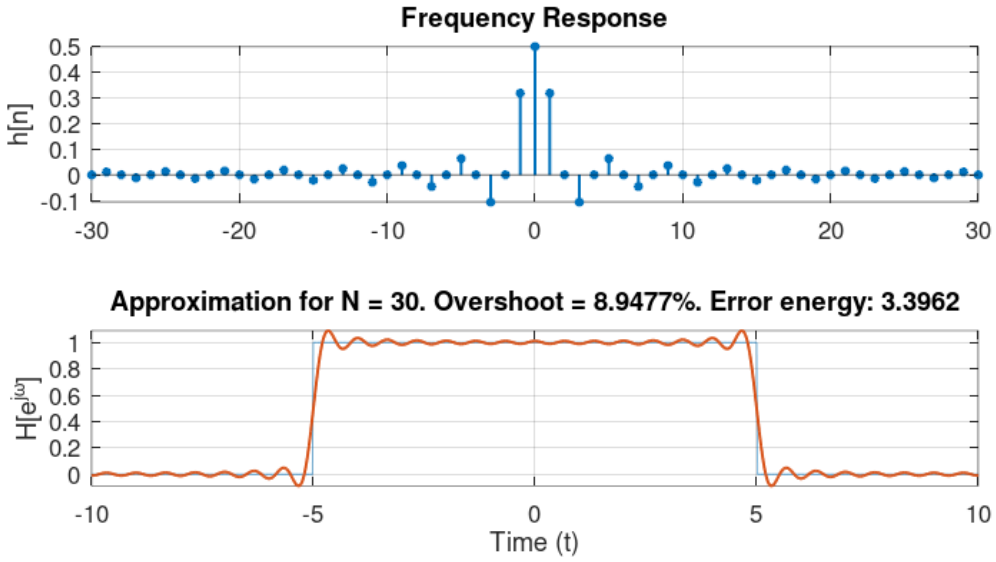
- **Why approximation by truncation is not such a good idea** The maximum error around the cutoff frequency is around 9% of the height of the jump regardless of N. This is known as the Gibbs Phenomenon.

#### 6.1.2.1 The Gibbs Phenomenon




References:

- Matlab Answers




### 6.1.3 Window method

The impulse truncation can be interpreted as the product of the ideal filter response and a rectangular window of N points.
From the modulation theorem, the DTFT of the product f two signals is equivalent to the convolution of their DTFTs. Hence, the choice of window influences the quality of the approximation results.

**Window Method**

The window method is just a generalization of the impulse truncation method where we use a different window shape.

For example, by using a triangular window, we reduce the Gibbs error at the price of a longer transition.

**6.1.3.1 The modulation theorem revisited.** We can consider the approximated filter as

$$\hat{h}[n] = h[n]\, w[n]$$

with the indicator function w[n]

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

*The question is how can we express the Fourier Transform* $\hat{H}(e^{j\omega}) =?$ *of the filter as the product of two sequences?* For that, we have to study the modulation theorem.

- Convolution Theorem states that the Fourier Transform of the convolution of two sequences is the product in the frequency domain of the Fourier Transforms.

$$DTFT\{(x * y)[n]\} = X(e^{j\omega})\, Y(e^{j\omega})$$

- Modulation Theorem The modulation theorem states that the Fourier Transform of the product of two sequences is the convolution in the frequency domain of the Fourier Transform

$$DTFT\{(x[n]\ y)[n]\} = (X * Y)(e^{j\omega})$$

- Convolution in the Frequency Domain

in $\mathbb{C}^{\infty}$ the space of infinite support signals, the convolution can be defined in terms of the inner product of the two sequences.

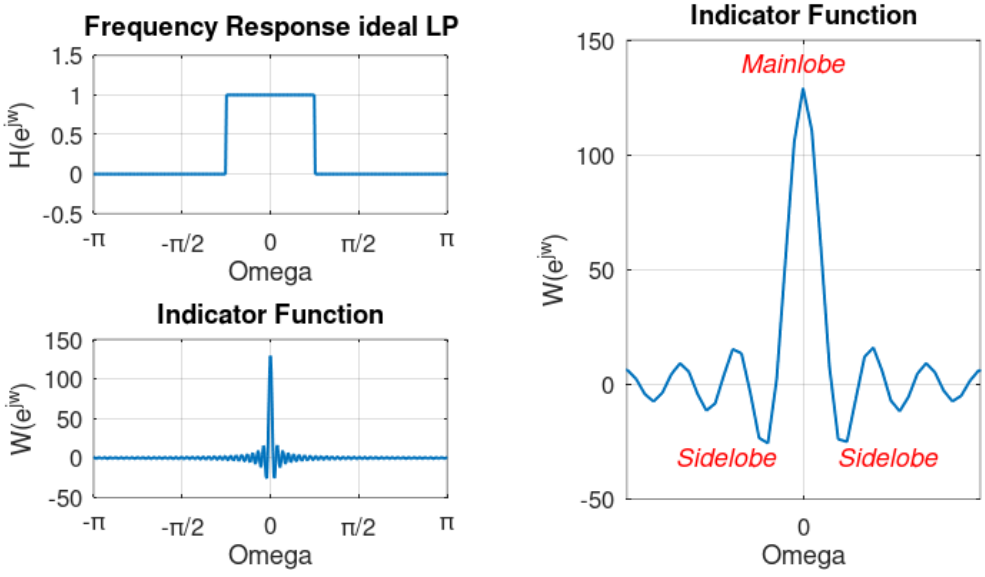
$$\begin{aligned}(x * y)[n] &= \langle x^*[k], y[n-k] \rangle \\ &= \sum_{n=-\infty}^{\infty} x[k]y[n-k]\end{aligned}$$

We can adapt the same strategie in $\mathbb{L}\Big(\big[-\pi, \pi\big]\Big)$, which is the space where the DTFT life's. So we find the convolution of two Fourier Transforms as the inner product of the first Fourier Transform conjugated and the second Fourier Transform frequence reversed and delayed by $\omega$

$$\begin{aligned}(X * Y)(e^{j\,\omega}) &= \langle X^*(e^{j\,\sigma}), Y(e^{j\,\omega-\sigma}) \rangle \\ &= \frac{1}{2\pi} \int_{-\pi}^{pi} X^*(e^{j\,\sigma})\, Y(e^{j\,\omega-\sigma})\, d\sigma\end{aligned}$$

If we apply the definition of the inner product for $L2([-\pi, \pi])$ we get that the convolution between two Fourier Transforms.

### 6.1.3.2 Mainlobe and Sidelobes



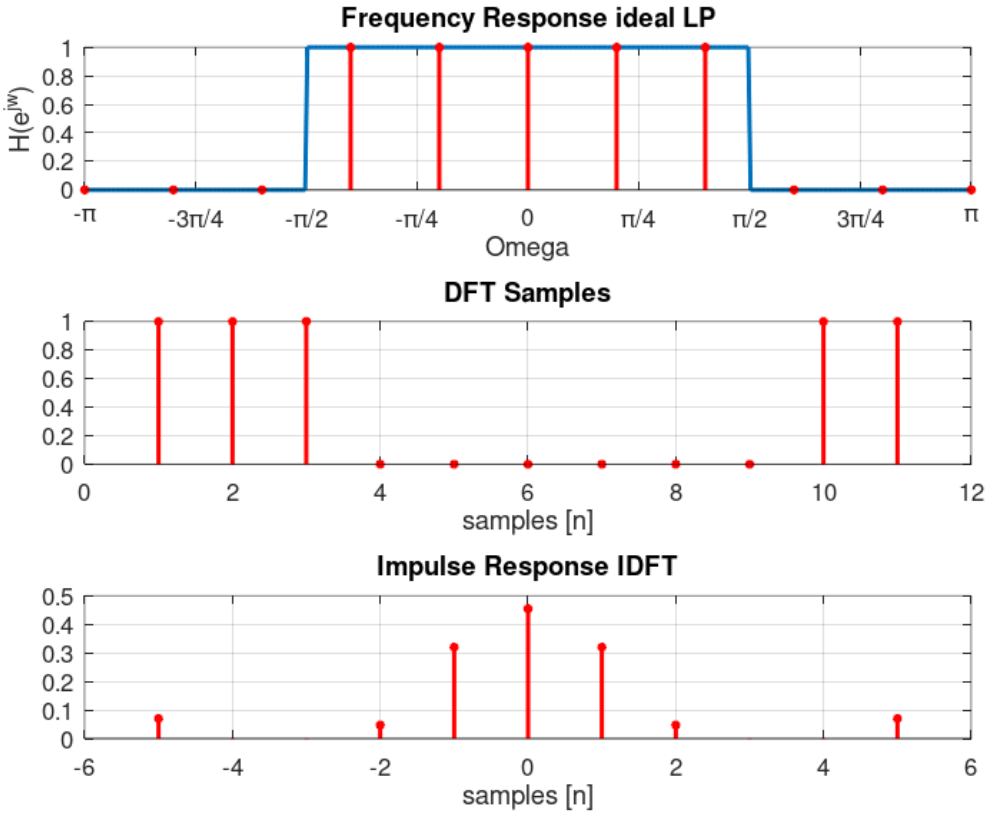
**We want:**

- narrow mainlobe ⇒ to have sharp transition
- small sidelobe ⇒ gibbs error is small
- short window ⇒ FIR is efficient

### 6.1.4 Frequency sampling

**Frequency Sampling**

1. Draw desired frequency response $H(e^{j\omega})$
2. take M values at $\omega_k = \frac{2\pi}{M} \cdot k$
3. compute IDFT of values
4. use result as M-tap impulse response $\hat{h}[n]$




- **Why Frequency Sampling is not such a good idea:**
  - frequency response is DTFT of finite-support, whose DFT we know
  - frequency response is interpolation of frequency samples
  - interpolator is transform N-tap rectangular window (no escape from the indicator function)
  - again no control over main- and sidelobe

**Summery Imitation**

These methods to approximate ideal filters are certainly very useful when we want to derive a quick and dirty prototype, and we don't have time to use more sophisticated filter design methods

## 6.2 Signal of the Day: Camera Resolution and space exploration

### 6.2.1 Rosettta Mission: Spacecraft

- Reaching Comet 67P. 10 years to get momentum to get its orbit.
- Resolution of taken pictures:

| Resolution | at Distance | Year | |
|---|---|---|---|
| 1km/pixel | 86'000km | 28. June 2014 | |
| | 12'000km | 14. July 2014 | |
| 100m/pixel | 5'500km | 20. July 2014 | |
| 5.3m/pixel | 285km | 3. August 2014 | |
| 11cm/pixel | 6km | 14. February 2015 | most detailed pictures of a planet |

Is it necessary to send a probe for 10years into space to get high resolution pictures?

### 6.2.2 Image Formation

$$\begin{aligned} i(x,y) &= s(x,y) * h(x,y), \text{ i: image that is formed,} \\ &= s(x,y) * t(x,y) * p(x,y) \end{aligned}$$

- i: image that is formed on the retina or camera
- s: light sources (source image)
- h: transfer function of the light
- t: medium through the light is traveling
- p: point spread function (PSF), lenses and focal distance

The major enemy to image quality of telescope on earth are the atmospheric disturbances.

**The pinhole camera** A certain pixel density is required to distinguish light sources on the image plane. We might be tempted to say the maximum achievable resolution is only depend on the **resolution** of the sensor at the back of the camera. In reality the resolution is limited by pixel density resolution is limited by diffraction.

**Diffraction** (Beugung) The image of an original point light source will appear as a diffraction pattern. The diffraction pattern through a small circular aperture is called **Airy disk**.

**Rayleigh's criterion** Minimum angle $\theta$ between light point sources that guarantees resolution

$$\theta = 1.22\frac{\lambda}{D}$$

- $\lambda$ : wave length of the light that hits the camera
- D : Diameter of the aperture

### 6.2.3 Seeing the Lunar Excursion Module (LEM)

- size of LEM $\approx$ 5m
- distance to the Moon $\approx$
- $Rightarrow \theta$ subtended by the LEM is $\approx 0.003 arcsec$
- Hubble's aperture: 2.4m
- visible spectrum $\lambda \approx 550nm$
- Rayleigh's criterion: $\theta \approx 0.1 arcsec$
  $\Rightarrow$ to see the LEM, Hubble should have an aperture of 80m!!!!

### 6.2.4 Rayleigh's criterion, Spatial Resolution

$$\delta x = 1.22\ f\ \frac{f}{D} = \theta \cdot f$$

If the pixel separation on the camera sensor is not less than $\delta x$ our camera will be resolution limited rather than diffraction limited.

- f: foco length
- f/D: f-number

pixel density takes into account the size of the sensor.

### 6.2.5 What about mega pixels?

How many mega pixels one need on an commercial camera. This actually depends on the size of the sensor and on the optics:

- f-number of all trades: f/8
- spatial Rayleigh's criterion: $\delta x \approx 4\mu m$
- max pixel area $16 \cdot 10^{-5}$
  $\Rightarrow$ to opperate at the diffraction limit we need $62'500 pixels/mm^2$
  Highend camera usually have one of the following sensors:
- APS-C sensor ($329mm^2$): 20 MP $\Rightarrow$ the camera is operating at the defraction limit
- 35-mm sensor ($864mm^2$): 54 MP $\Rightarrow$ the camera is operating at the defraction limit

## 6.3 Realizable Filters

### 6.3.1 The Z-Transform

#### 6.3.1.1 References

. Signals and Systems for Dummies: Z-Transform

#### 6.3.1.2 Z-Transform

maps a discrete-time sequence x[n] onto a function of $\sum_{n=-\infty}^{\infty} x[n]\ z^{-n}$.

$$x[n] = \sum_{n=-\infty}^{\infty} x[n]\ z^{-n} \tag{31}$$

The z-Transform is an extension of the DTFT to the whole complex plane and is equal to the DTFT for $z = e^{j\omega}$.

$$X(z)|_{z=e^{j\omega}} = DTFT\{x[n]\} \tag{32}$$

Key properties of the z-Transform are:

- linearity: $\mathcal{Z}\{\alpha x[n] + \beta y[n]\} = \alpha X(z) + \beta Y(z)$
- time shift: $\mathcal{Z}\{x[n-N]\} = z^{-N}X(z)$

Applying the z-transform to CCDE's

$$\sum_{k=0}^{N-1} a_k y[n-k] = \sum_{k=0}^{M-1} b_k x[n-k]$$
$$Y(z)\sum_{k=0}^{N-1} a_k z^{-k} = X(z)\sum_{k=0}^{M-1} b_k z^{-k}$$
$$Y(z) = H(z)X(z)$$

- **M input values**
- **N output values**

**6.3.1.3 Constant Difference Equation** A constant coefficent difference equation (CCDE) expresses the input-, output relationship of an LTI system as a linear combination of output samples equal to a linear combination of input samples

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

In the z-domain, a Constant Coefficent Difference Equation CCDE is represented as a ration $H(z)$ of two polynomials of $z^{-1}$.

$$H(z) = \frac{\sum_{k=0}^{M-1} b_k z^{-k}}{\sum_{k=0}^{N-1} a_k z^{-k}} \tag{33}$$

**6.3.1.4 Frequency Response** The frequency response of a filter is equal to this transfer function evaluated at $z = ^{j\omega}$.

$$H(j\omega) = H(z)|_{Z=e^{j\omega}} = \frac{\sum_{k=0}^{M-1} b_k z^{-k}}{\sum_{k=0}^{N-1} a_k z^{-k}} \tag{34}$$

### 6.3.2 Z-Transform of the leaky integrator

$$
\begin{aligned}
y[n] &= (1-\lambda)x[n] + \lambda y[n-1] \\
Y(z) &= (1-\lambda)X(z) + \lambda z^{-1}Y(z) \\
Y(z) - \lambda z^{-1}Y(z) &= (1-\lambda)X(z) \\
Y(z)\left(1-\lambda z^{-1}\right) &= (1-\lambda)X(z) \\
Y(z) &= H(z)X(z) \\
H(z) &= \frac{Y(z)}{X(z)} = \frac{1-\lambda}{1-\lambda z^{-1}} \\
H(e^{j\omega}) &= \frac{1-\lambda}{1-\lambda e^{-j\omega}}
\end{aligned}
$$

**6.3.2.1 LTI Systems** An LTI system can be represented as the convolution $y[n] = x[n] * h[n]$. From the convolution property of the Z-transform, it follows that the z-transform of y[n] is:

$$Y(z) = H(z)\, X(z) \tag{35}$$

### 6.3.3 Region of convergence

Conditions for convergences

- The zeros/poles are the roots of the numerator/denominator of the rational transfer function
- the region of convergence is only determined by the magnitude of the poles
- the z-transform of a causal LTI system extends outwards from the largest magnitude pole

**BIBO-Stable**

An LTI system is stable if its region of convergence includes the unit circle

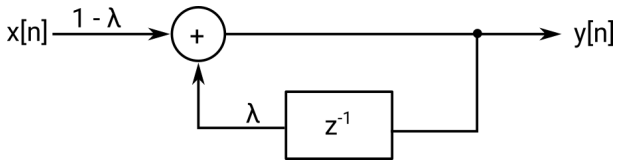
## 6.4 Filter Design Part 2

- many signal processing problems can be solved using simple filters
- we have seen simple lowpass filters already (Moving Average, Leaky Integrator)
- simplel (low order) transfer functions allow for intuitive design and tuning

### 6.4.1 Intuitive IIR Designs

#### 6.4.1.1 Leaky Integrator

##### 6.4.1.1.1 Filter Structure



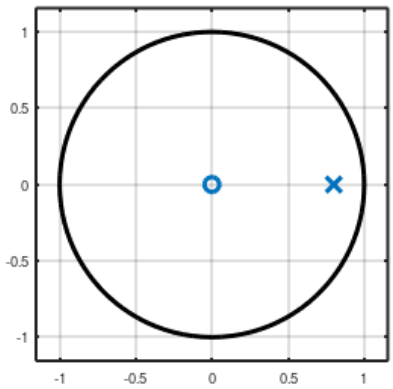

##### 6.4.1.1.2 Transfer Function

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

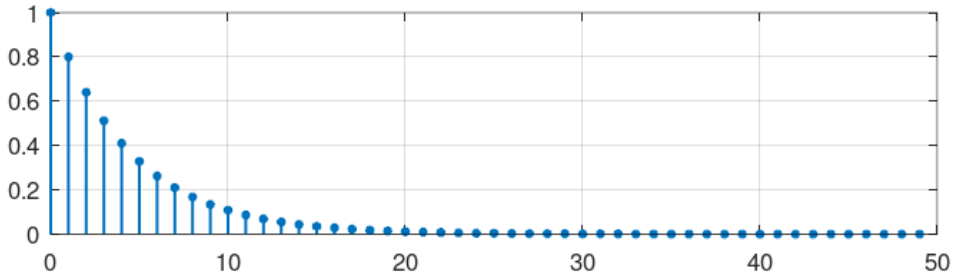
#### 6.4.1.1.3 CCDE

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

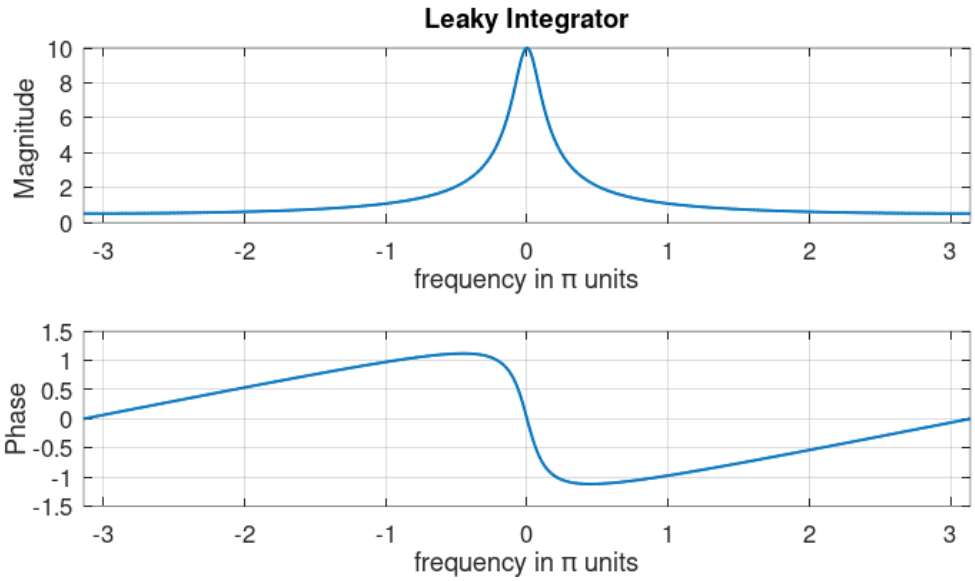
#### 6.4.1.1.4 Pole-Zero Plot




#### 6.4.1.1.5 Impulse response




#### 6.4.1.1.6 Frequency Response




### 6.4.1.2 Resonator

- a resonator is a narrow bandbass filter

- used to detect presence of a given frequency
- useful in communication systems and telephone (DTMF)
- Idea: shift passband of the Leaky Integrator

#### 6.4.1.2.1 Transfer Function

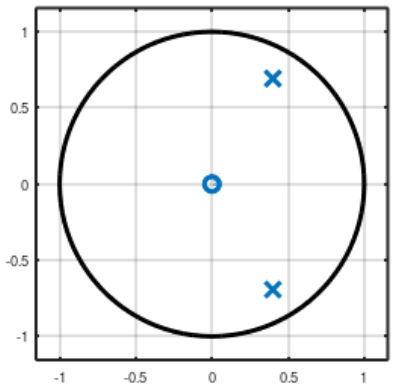
$$H(z) = \frac{G_0}{(1 - pz^{-1})(1 - p^* z^{-1})}$$
$$p = \lambda e^{j\omega_0}$$
$$H(z) = \frac{G_0}{1 - 2\mathcal{R}pz^{-1} + |p|^2 z^{-2}}$$
$$H(z) = \frac{G_0}{1 - 2\lambda\omega_0 z^{-1} + |\lambda|^2 z^{-2}}$$

The coeffience to be used in the CCDE

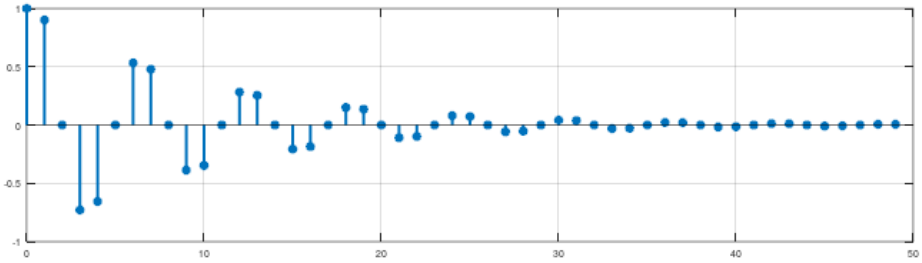
$$a_1 = 2\lambda cos\omega_0$$
$$a_2 = -|\lambda|^2$$

#### 6.4.1.2.2 Pole-Zero Plot

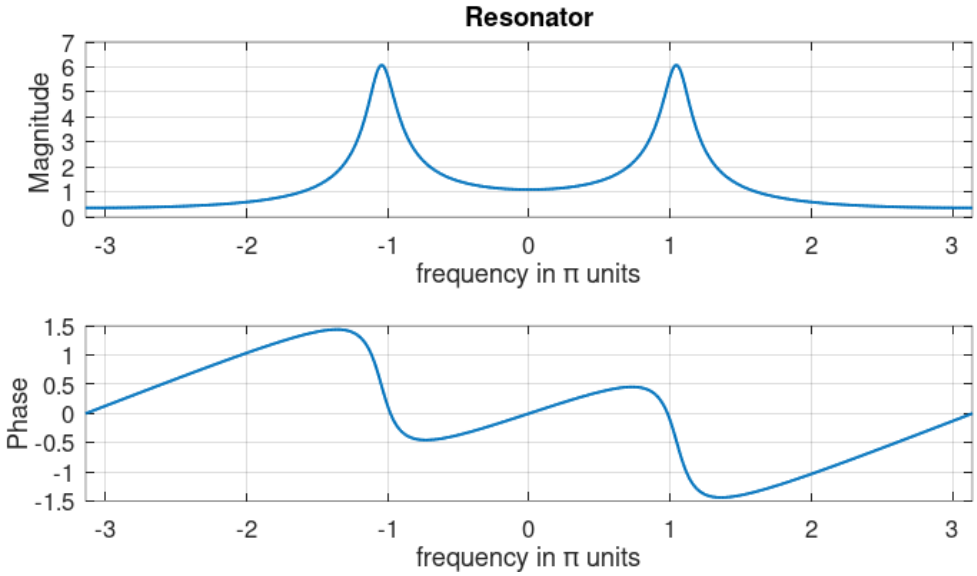
- Move the pole of the leaky integrator radially around the circle of radius lambda to shift the passband at the frequency that we are interested in, i.e. $\omega_0$. interested in selecting. Since we want a real filter, we also have to create a complex conjugate pole at an angle that is $-\omega_0$.



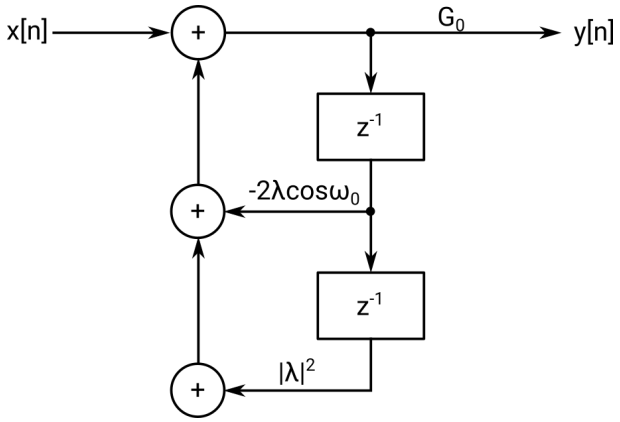

#### 6.4.1.2.3 Impulse response



#### 6.4.1.2.4 Frequency Response




#### 6.4.1.2.5 Filter Structure




### 6.4.1.3 DC Removal

- a DC-balances signal has zero sum: $\lim_{N\to\infty} \sum_{n=-N}^{N} x[n] = 0$ i.e. there is no Direct Current component
- its DTFT value at zero is zero for an $\omega = 0$
- we want to remove the DC bias from a non zero-centered signal
- we want to kill the frequency component at $\omega = 0$

#### 6.4.1.3.1 Transfer Function

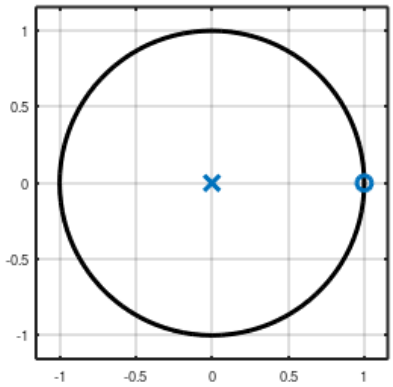
$$H(z) = 1 - Z^{-1}$$

#### 6.4.1.3.2 CCD

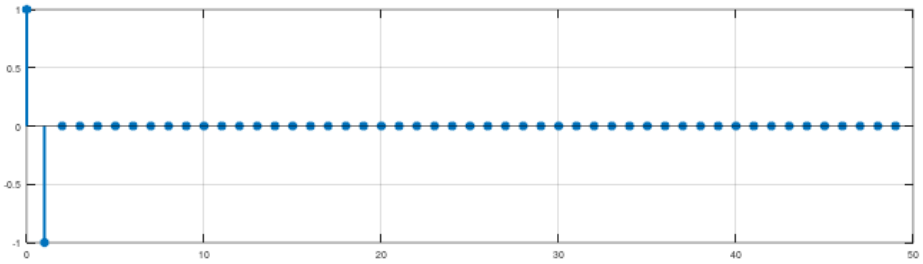
$$y[n] = x[n] - x[n-1]$$

#### 6.4.1.3.3 Pole-Zero Plot

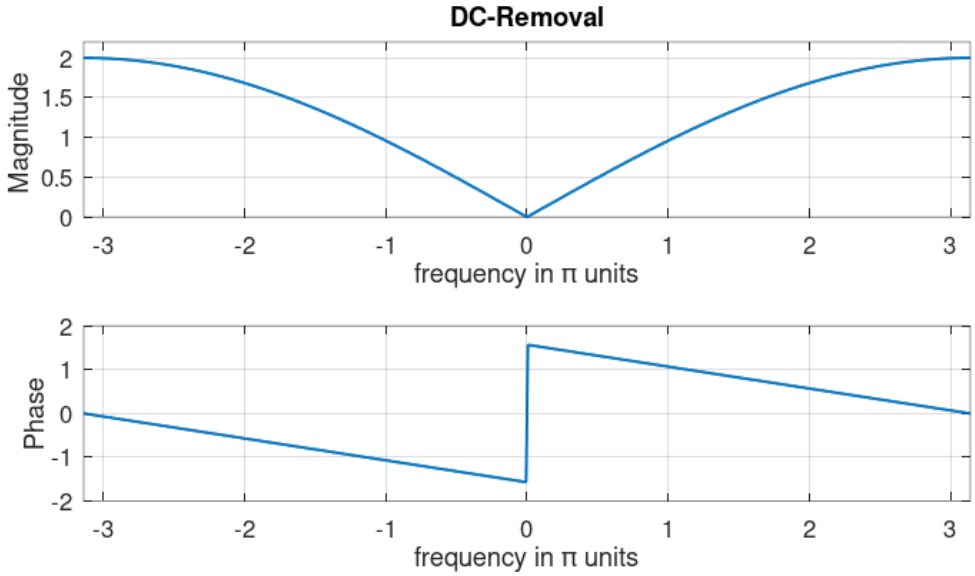
- Simply place a zero at $ z = 1$




#### 6.4.1.3.4 Impulse response




#### 6.4.1.3.5 Frequency response




This is not an acceptable characteristic because it introduces a very big attenuation over almost the entety of the frequency support.

### 6.4.1.4 DC Removal Improved - DC-Notch Filter

#### 6.4.1.4.1 Transfer Function

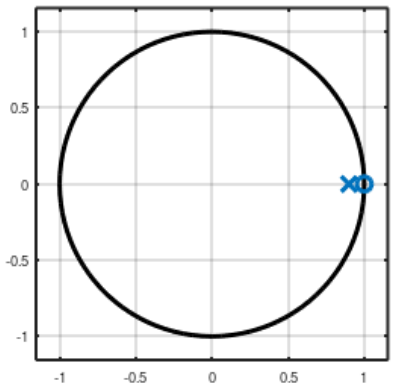
$$H(z) = \frac{1 - z^{-1}}{1 - \lambda z^{-1}}$$

#### 6.4.1.4.2 CCDE

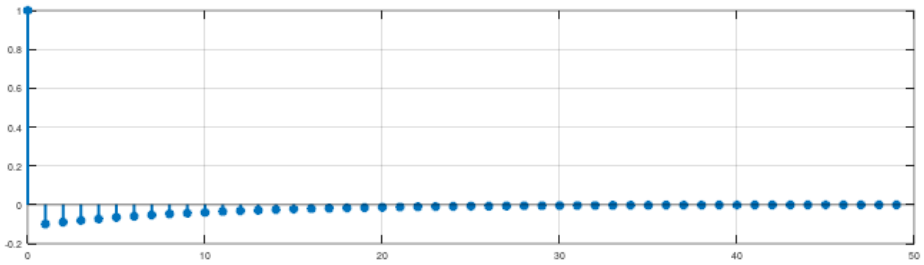
$$y[n] = \lambda y[n-1] + x[n] - x[n-1]$$

#### 6.4.1.4.3 Pole-Zero Plot

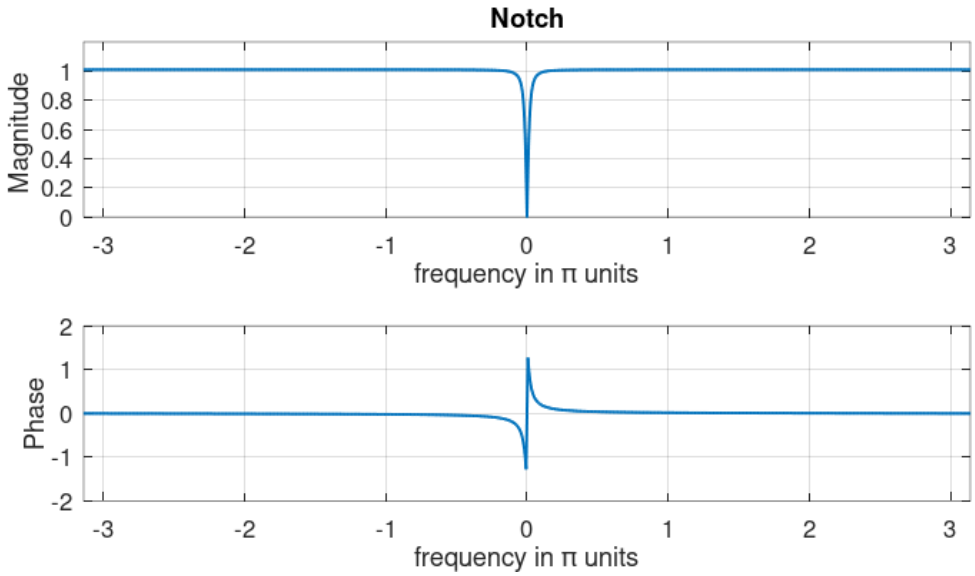
- and if we remember the circus tent method, we know that we can push up the z-transform by putting a pole in the vicinity of the 0. So we try and do that and we combine therefore, the effect of a 0 and 1 with the effect of a pole close to one, and inside the unit circle, for obvious reasons of stability.



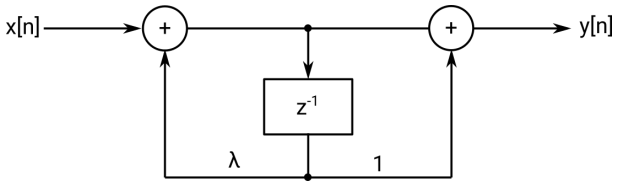

#### 6.4.1.4.4 Impulse response




#### 6.4.1.4.5 Frequency Response




#### 6.4.1.4.6 Filter Structure




### 6.4.1.5 Hum Removal

- The hum removal filter is to the dc notch what the resonator is to the leaky integrator
- similar to DC removal but want to remove a specific nonzero frequency
- very usful for musicaians amplifiers for electronic guitars pick up the hum from the electronic mains (50Hz in Europe and 60Hz in North America)
- we need to tune the hum removal according the country

#### 6.4.1.5.1 Transfer Function

$$
\begin{aligned}
H(z) &= \frac{(1-e^{j\omega_0}z^{-1})(1-e^{-j\omega_0}z^{-1})}{(1-\lambda e^{j\omega_0}z^{-1})(1-\lambda e^{-j\omega_0}z^{-1})} \\
p &= e^{j\omega_0} \\
q &= \lambda e^{j\omega_0} \\
&= \frac{(1-pz^{-1})(1-p*z^{-1})}{(1-qz^{-1})(1-q*z^{-1})} \\
H(z) &= \frac{1-2\mathcal{R}pz^{-1}+|p|^2z^{-2}}{1-2\mathcal{R}qz^{-1}+|q|^2z^{-2}} \\
&= \frac{1-2\omega_0 z^{-1}+z^{-2}}{1-2\lambda\omega_0 z^{-1}+|\lambda|^2z^{-2}}
\end{aligned}
$$

The coeffience to be used in the CCDE

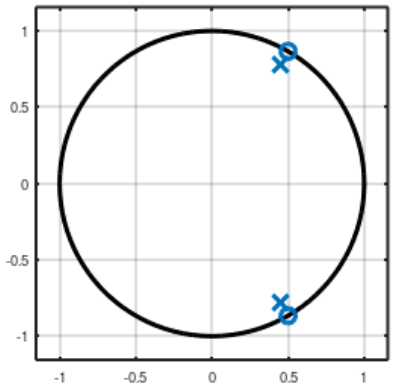
$$
\begin{aligned}
a_1 &= -2\lambda cos\omega_0 \\
a_2 &= |\lambda|^2 \\
b_1 &= -2\omega_0 \\
b_2 &= 1
\end{aligned}
$$

#### 6.4.1.5.2 CCDE

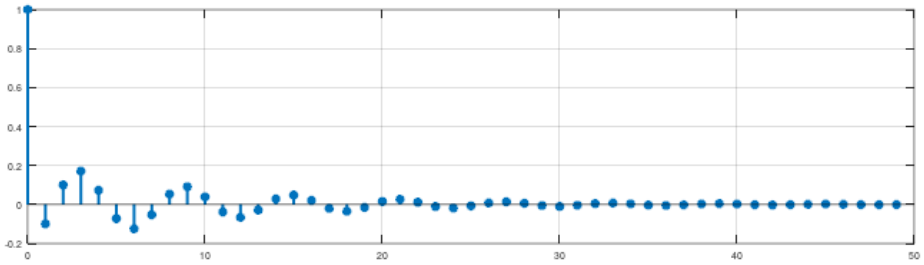
$$y[n] = 2\lambda\ cos\omega_0\ y[n-1] + |\lambda|^2\ y[n-2] + x[n] - 2\ cos\omega_0\ x[n-1] + \ x[n-2]$$

#### 6.4.1.5.3 Pole-Zero Plot

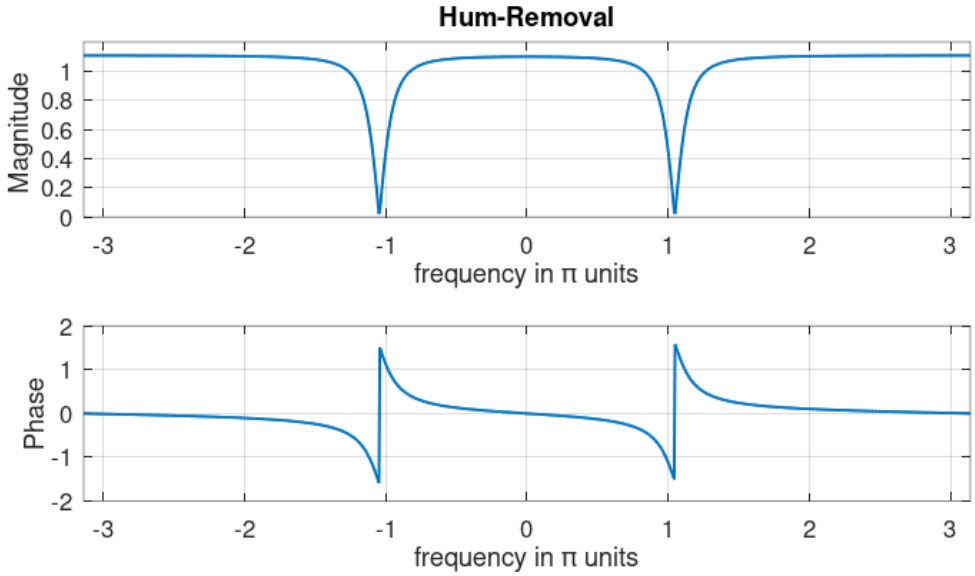
- and if we remember the circus tent method, we know that we can push up the z-transform by putting a pole in the vicinity of the 0. So we try and do that and we combine therefore, the effect of a 0 and 1 with the effect of a pole close to one, and inside the unit circle, for obvious reasons of stability.



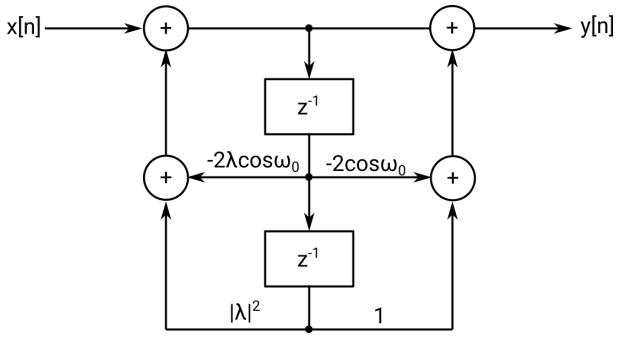

#### 6.4.1.5.4 Impulse response




#### 6.4.1.5.5 Frequency Response



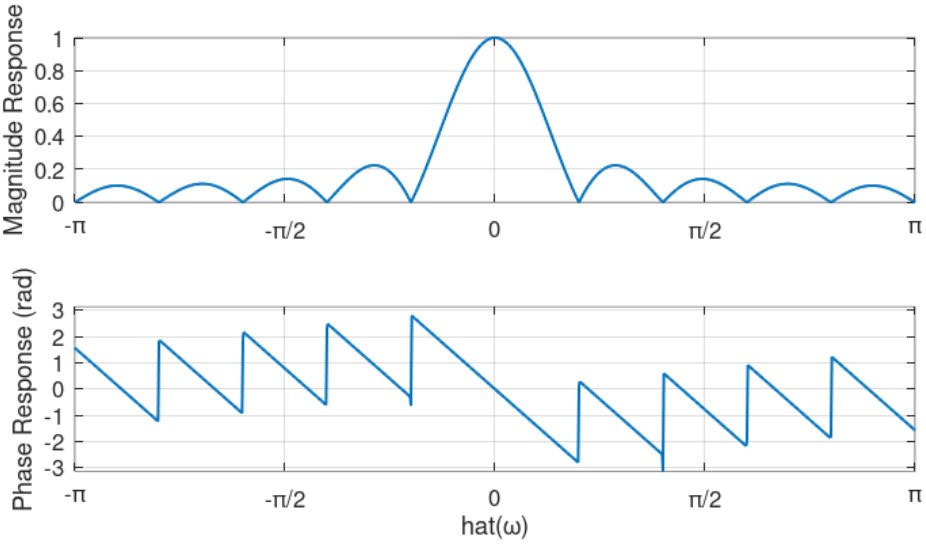

#### 6.4.1.5.6 Filter Structure




### 6.4.2 Matlab

**Dirichlet** The Dirichlet or periodic sync function can be used to analyze Moving Average Filters $D_M(j\omega) = diric(\omega, M) = \frac{sin(\frac{\omega}{2}M)}{sin(\frac{\omega}{2}M)}$

**Freqz** The frequency response can be plotted most easily using freqz() function.



## 6.5 Filter Design Part 3

### 6.5.1 Filter Specification

### 6.5.2 IIR Design

Filterdesign was established art long before digital processing appeared

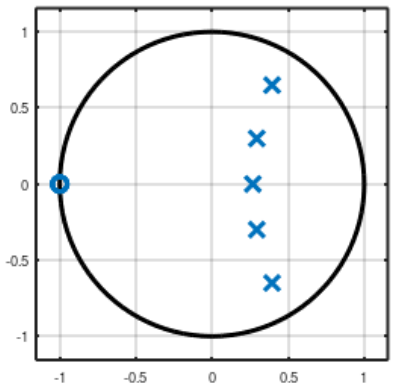
- AFD: Analog Filter Design
- lots of nice analog filters exist
- methods exist to "translate" the analog design into a rational transfer function
  - **impulse invariance transformation**, preserves the shape of the impulse response
  - finite difference approximation, converts a differential equation into a ccde
  - step invariance, preserves the shape of the step response
  - matched-z transformation, matches the pole-zero representation
  - **bilinear transformation**, preserves the system function representation
- most numerical packages (Matlab, etc.) provide ready-made routines
- design involves specifying some parameters and testing that the specs are fulfilled

Table 1: Butterworth lowpass

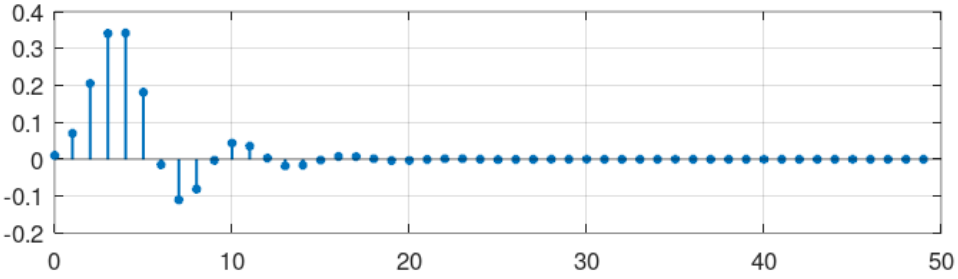
| Magnitude response | Design Parameters | Test values |
|---|---|---|
| maximally flat | order N | width of transition band |
| monotonic over $[0, \pi]$ | cutoff frequency | passband error |

#### 6.5.2.1 Butterworth lowpass

##### 6.5.2.1.1 Pole-Zero Plot




##### 6.5.2.1.2 Impulse Response



#### 6.5.2.1.3 Frequency Response

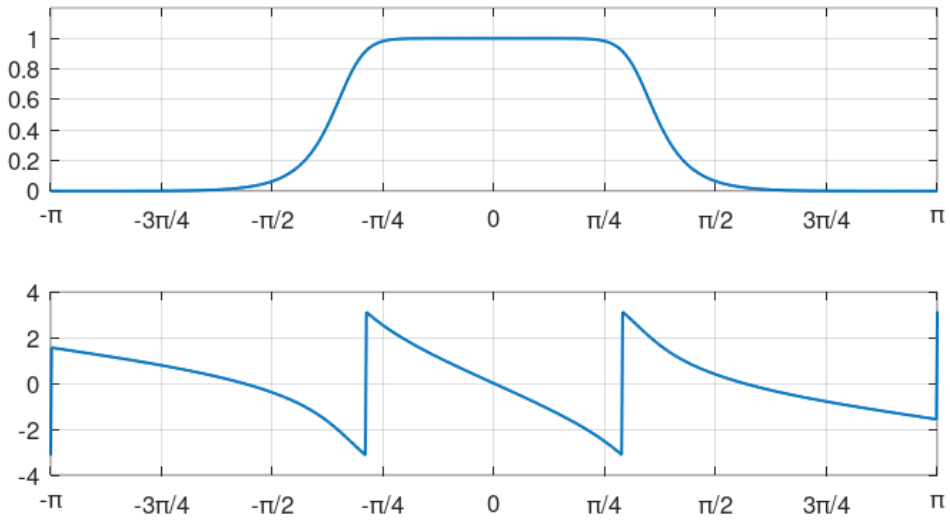
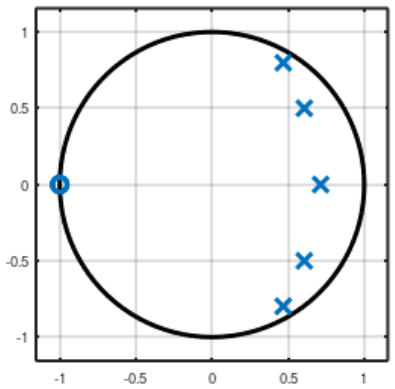



Table 2: Chebyshev lowpass

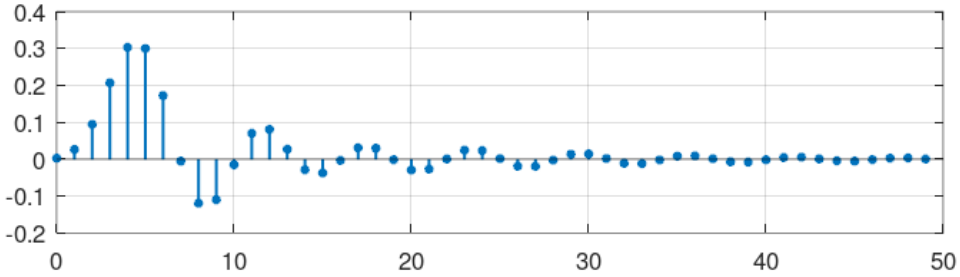
| Magnitude response | Design Parameters | Test values |
|---|---|---|
| equiripple in passband | order N | width of transition band |
| monotonic in stopband | passband max error | stopband error |
| | cutoff frequency | |

### 6.5.2.2 Chebyshev lowpass

#### 6.5.2.2.1 Pole-Zero Plot




#### 6.5.2.2.2 Impulse Response




#### 6.5.2.2.3 Frequency Response

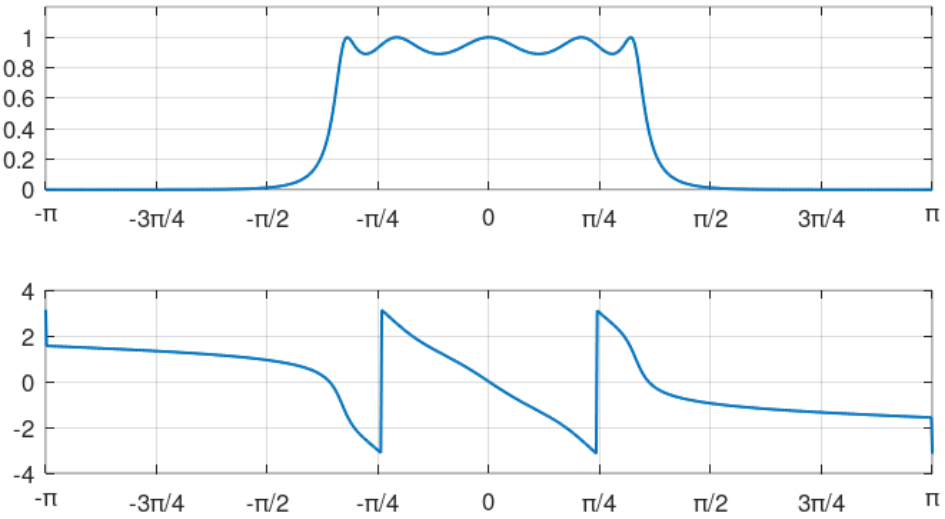
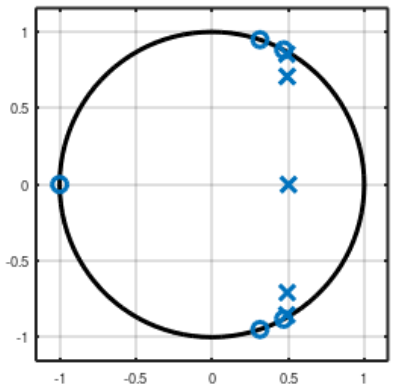



Table 3: Elliptic lowpass

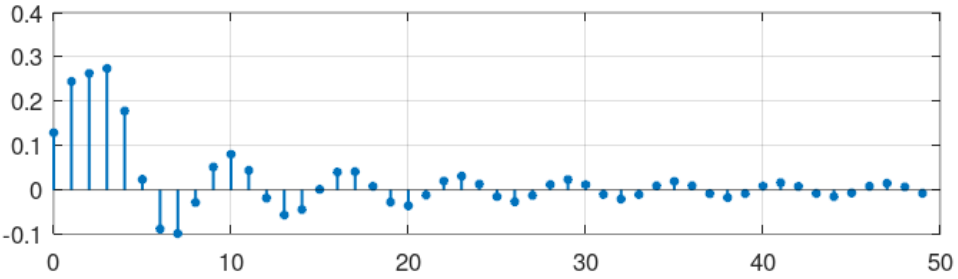
| Magnitude response | Design Parameters | Test values |
|---|---|---|
| equiripple in passband | order N | width of transition band |
| equiripple in stopband | cutoff frequnecy | |
| | passband max error | |
| | stopband min attenuation | |

### 6.5.2.3 Elliptic lowpass

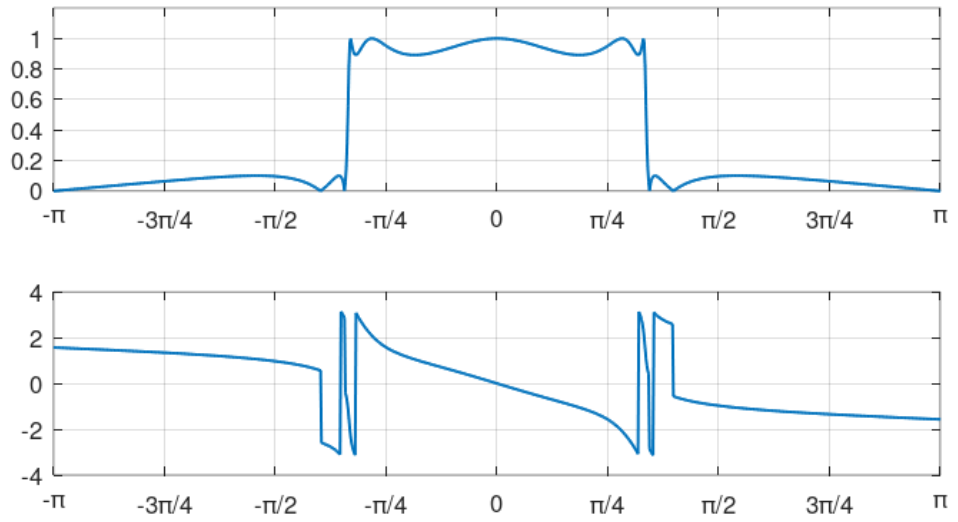
#### 6.5.2.3.1 Pole-Zero Plot




#### 6.5.2.3.2 Impulse Response




#### 6.5.2.3.3 Frequency Response



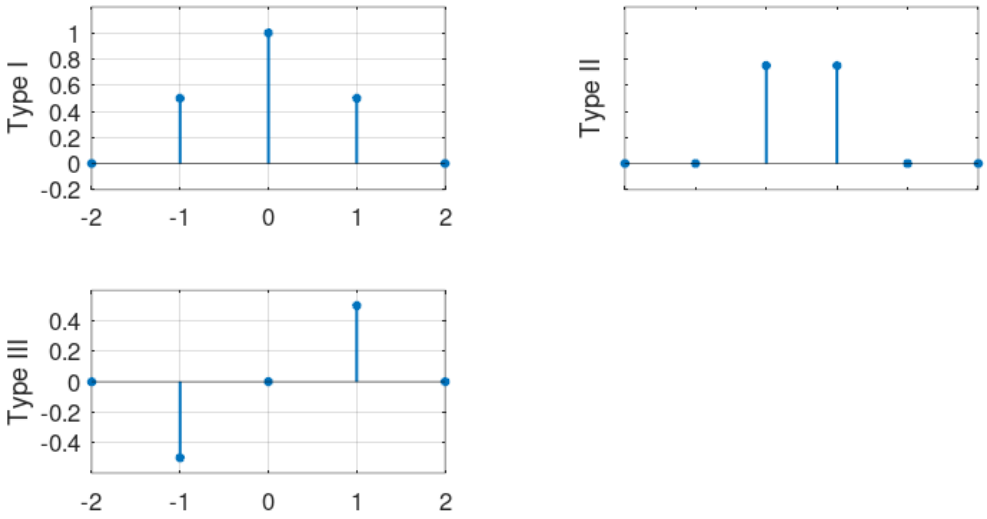

### 6.5.3 FIR Design

**6.5.3.1 Optimal minmax design** FIR filters are `digital` signal processing "exclusivity". In the 70s Parks and McClellan developed an algorithm to design optimal FIR filters:

- linear phase
- equiripple error in passband and stopband

algorithm proceeds by **minimizing** the maximum error in passpand and stopband

**6.5.3.1.1 Linear Phase** Linear phase derives from a symmetric or antisymmetric impulse respones




**Type I-Filters** Odd length impulse response, and are symmetric

**Type II-Filters** Even length impulse response, and are symmetric

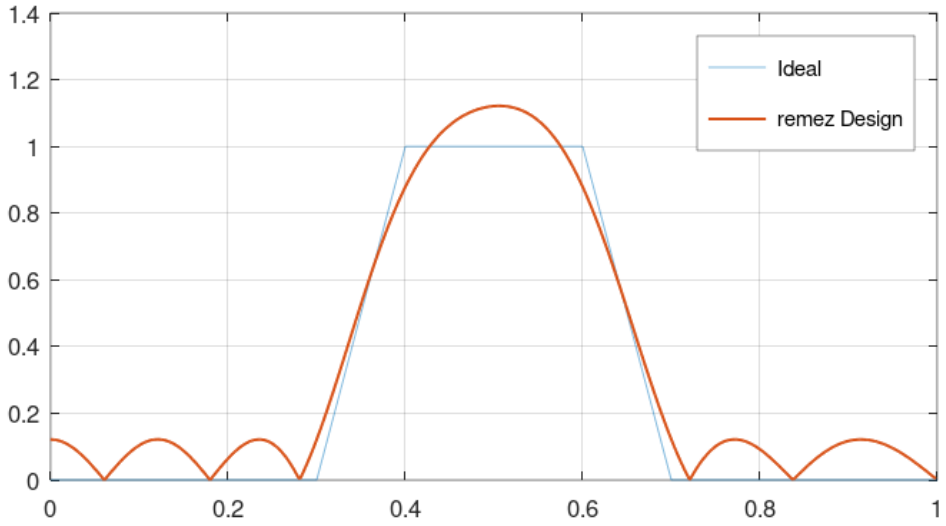
**Type III-Filters** Odd length impulse response, and are antisymmetric

**Type IV-Filters** Even length impulse response, and are antisymmetric

Type-II and Type-IV Filters are symmetric and antisymmetric filters, respectively, both of which have an even number of taps. That means that the center symmetry of these filters fall in between samples. And so they both introduce a non integer linear phase factor, of one half sample.

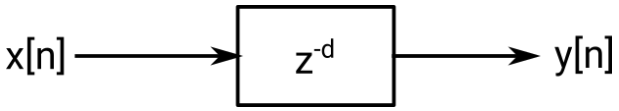
### 6.5.4 The Park McMellon Design Algorithm

| Magnitude response | Design Parameters | Test values |
|---|---|---|
| equiripple in passband and stopband | order N | passband max error |
| | passband edge $\omega_p$ | stopband max error |
| | stopband edge $\omega_s$ | |
| | ratio of passband to stopband error $\frac{\delta_p}{delta_s}$ | |




## 6.6 ONGOING Notes and Supplementary Materials

### 6.6.1 The Fractional Delay Filter (FDF)



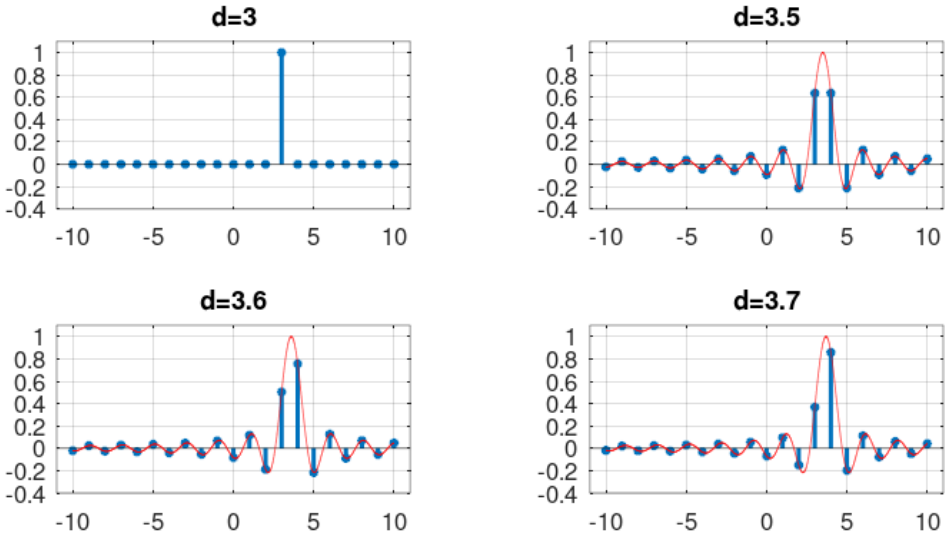

The transfer function of a simple delay $z^{-d}$ is:

$$H(e^{j\omega}) = e^{-j\omega d},\ d \in \mathbb{Z}$$

what happens if, in $H(e^{j\omega}$ we use a non-integer $d \in \mathbb{R}$?

#### 6.6.1.1 Impulse Response

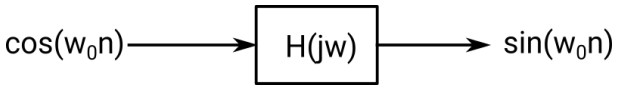
$$\begin{aligned} h[n] &= IDFT\left\{e^{j\omega d}\right\} \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{j\omega d}e^{j\omega n}d\omega \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{j\omega(n-d)}d\omega \\ &= \frac{1}{\pi(n-d)}\frac{e^{j\pi(n-d)} - e^{-j\pi(n-d)}}{2j} \\ &= \frac{sin\pi(n-d)}{\pi(n-d)} \\ &= sinc(n-d)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




For now suffice it to say that we can actually interpolate in discrete time and find intermediate values of a discrete time sequence using just discrete times filters like the fractional delay

### 6.6.2 ONGOING The Hilbert Filter

- Demodulator




can we build such a thing?

### 6.6.3 Implementing of Digital Filters

```c
#include <stdio.h>
double leaky(double x) {
    static const double lambda = 0.9;
    static double y = 0;   // 1x memory cell
    // plus initialization

    // algorithm: 2x multiplication, 1x addition
    y = lambda * y + (1-lambda) *x;
    return y;
}

int main() {
    int n;
    for(n = 0; n <20; n++)
    {
        //call with delta signl
        printf("%.4f ", leaky(n==0 ? 1.0 : 0.0));
        if(!((n+1)%10)) printf("\n");
    }
}
```

#### 6.6.3.1 Leaky Integrator in C

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0.1 | 0.09 | 0.081 | 0.0729 | 0.0656 | 0.059 | 0.0531 | 0.0478 | 0.043 | 0.0387 |
| 0.0349 | 0.0314 | 0.0282 | 0.0254 | 0.0229 | 0.0206 | 0.0185 | 0.0167 | 0.015 | 0.0135 |

- we need a ”memory cell” to store previous state
- we need to initialize the storage before first use
- we need 2 multiplications and one addition per output sampel

```c
#include <stdio.h>
double ma(double x) {
    static const int M = 5;
    static double z[M]; // Mx memory cells
    static int ix = -1;

    int n;
    double avg = 0;

    if(ix == -1) {      // initalize storage
        for(n=0; n<M; n++)
            z[n] = 0;
        ix = 0;
    }

    z[ix] = x;
    ix = (ix + 1) % M;  // circular buffer

    for(n=0; n<M; n++)  // Mx additions
        avg += z[n];

    return avg / M;     // 1x division
}

int main() {
    int n;
    for (n = 0; n<20; n++)
    {
       // call with delta signl
        printf("%.4f ", ma(n==0 ? 1.0 : 0.0));
        if(!((n+1)%10)) printf("\n");
    }
}
```

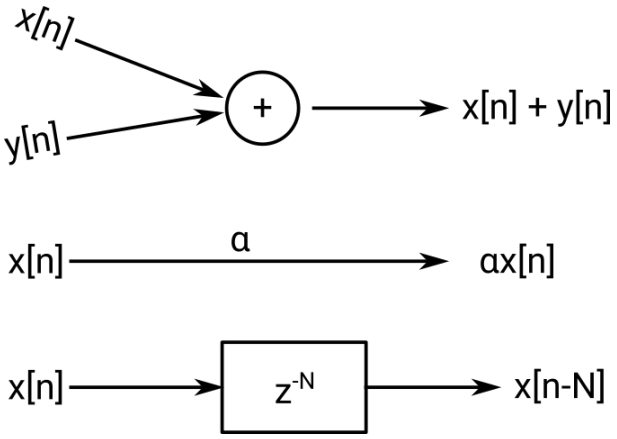
#### 6.6.3.2 Moving Average in C

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

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

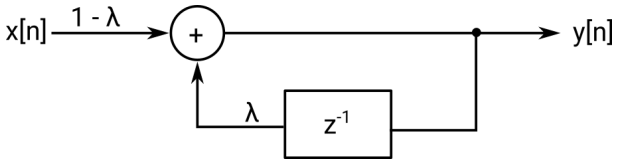
**6.6.3.3 Programming Abstraction** With this three building blocks we can describe and Constant Coefficient Equation.

#### 6.6.3.3.1 Building Blocks



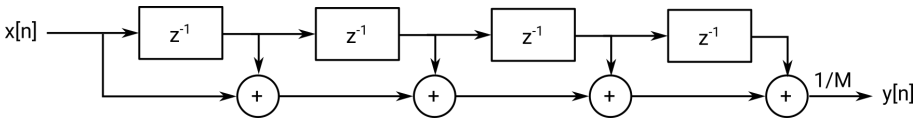

#### 6.6.3.3.2 Leaky Integrator

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




#### 6.6.3.3.3 Moving Average

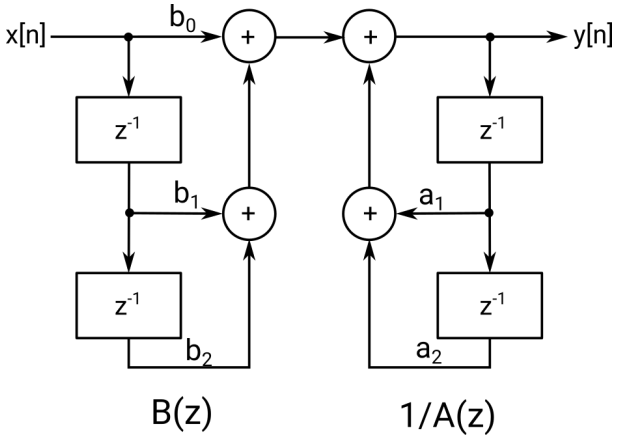
$$y[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-k]$$



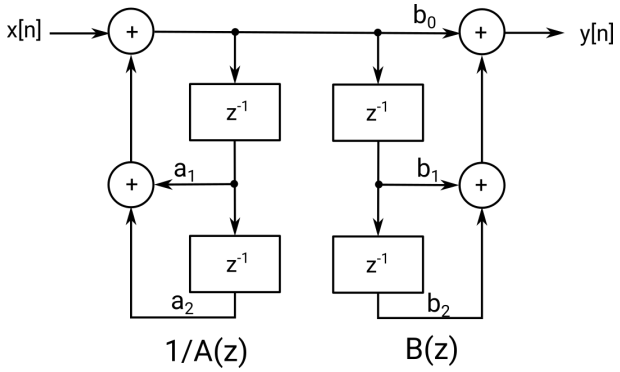

#### 6.6.3.3.4 The second-order section

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

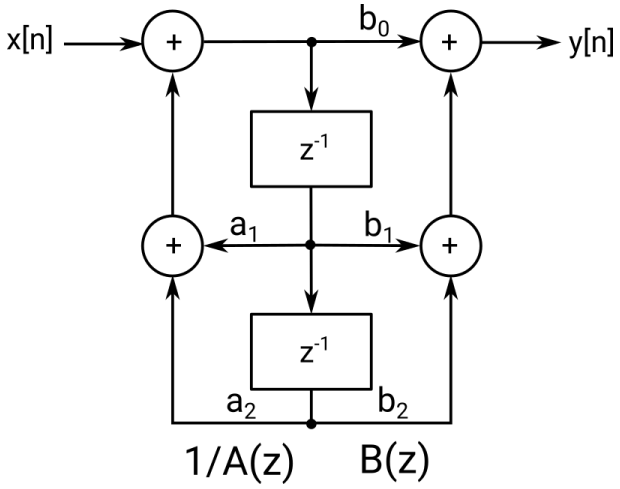
#### 6.6.3.3.5 Second-order section, direct form I



**6.6.3.3.6 Second-order section, inverted direct form I** Because the convolution is commutative, numerator and denominator may be swapped.




**6.6.3.3.7 Second-order section, direct form II** Since the content of the delay cells are exactly the same for all time, so we can lumb the delay cells together.




### 6.6.4 TODO Real-Time Processing

#### 6.6.4.1 I/O and DMA Everything works in synch with a system clock of period $T_s$

- record a value $x_i[n]$
- process the value in a casual filter
- play the output $x_o[n]$

Everything needs to happen in at most $T_s$ seconds!

Buffering:

- interrupt for each sample would be too much overhead
- soundcard consumes samples in buffers
- soundcard notifies when buffer used up
- CPU can fill a buffer in less time than soundcard can empty it

Double Buffering

- Delay $d = T_s \times \frac{L}{2}$ L: Lenght of the Buffer
- If CPU doesn't fill the buffer fast enough: **underflow**

Multiple I/O Processing

- Delay: $d = T_s \times L$
- usually start out process first

#### 6.6.4.2 Implementation Framework

Low Level

- study soundcard data sheet
- write code to program soundcard via writes to IO Ports
- write an interrupt handler
- write the code to handle the data

High Level

- choose a good API
- write a callback function to handle the data

```
int Callback( const void *input,     // pointer to the input buffer
              void *output,          // pointer to the output buffer
              unsigned long samples, // length of samples
    );
{
    float* pIn = (float*)input;      // Convert the generic buffers
    float* pOut = (float*)output;    // to the right data type
    for (int n=0; n < samples; n++)  // Calls process for each sample in input the buffer
        *pOut++ = Process(*pIn++);   // and store the result into the output buffer.
}
```

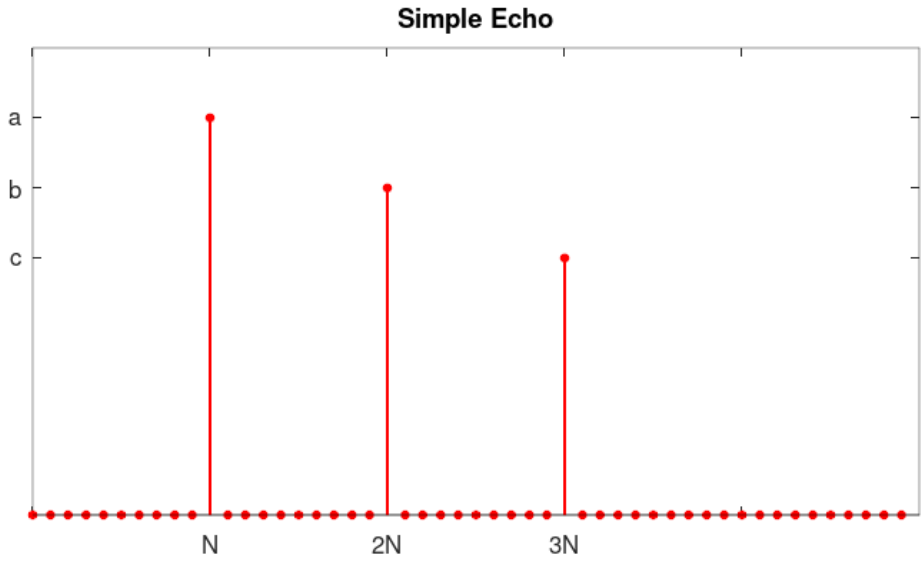
#### 6.6.4.3 Callback Prototype

```
                                 // 10 sec @ 24kHz
enum {BUF_LEN = 0x10000};        // Buffer length, power of 2
enum {BUF_MASK = BUF_LEN -1};    // Circular buffer mask
float m_pY[BUF_LEN];             // 2 Buffers
float m_pX[BUF_LEN];
int m_Ix;                        // 2 indixes into the buffers
int m_Iy;
float Process(float Sample)
{
    m_PX[m_Ix] = Sample;         // store the sample into input buffer
    float y = Effect();          // call Effect()
    m_pY[m_Iy] = y;              // store the output into output buffer
    m_Ix = (m_Ix + 1) & BUF_MAS; // Update indices with the ciruclar strategy
    m_IY = (m_Iy + 1) & BUF_MAS;
    return y;                    // return current output sample
}
```

#### 6.6.4.4 Processing Gateway

**6.6.4.5 Effect** Implementing the echo effect as a reflection of the original signal, scaled with a factor at subsequent points in time:

$$y[n] = \frac{ax[n] + bx[n-N] + cx[n-2N]}{a+b+c}$$




```
float Echo() {
    static float a = 0.85;              // the three scaling factors
    static float b = 0.6f;
    static float c = 0.45f;
    static float norm = 1.0f/(a+b+c);   // the normalisation factor
    static int N = (int)(0.3 * m_SR);   // delay between reflextion

    return norm * ( a * m_pX[m_Ix]
                  + b * m_pX[(m_Ix + BUF_LEN -N)   & BUF_MASK]
                  + c * m_pX[(m_Ix + BUF_LEN -2*N) & BUF_MASK]); }
```

### 6.6.5 TODO Derevereration and echo cancellation

# 7 Week 7 Module 5: Sampling and Quantization

Interpolation describes the process of building a continuous-time signal $\mathbf{x(t)}$ from a sequence of samples $\mathbf{x[n]}$. In other words, interpolation allows moving from the discrete-time world to the continuous-time world. Interpolation raises two interesting questions:
The first one is how to interpolate between samples?

- In the case, of two samples, this is simple enough and there is there is a straight line that goes between these two samples.
- In the case of three samples, similarly, you have a parabola that goes through these 3 samples.
- If you have many samples, you can try to do the same and go through all samples but you see this is a trickier issue compared to what we have done with two or three samples.

The second question is:

- is there a minimum set of values you need to measure the function at so that you can perfectly reconstruct it.

Later on in the module, we are going to study sampling, i.e. the process of moving from a continuous-time signal to a sequence of samples. In other words, sampling allows moving from the continuous-time world to the discrete-time world. Suppose we take equally-spaced samples of a function $\mathbf{x(t)}$. The question is when is there a one-to-one relationship between the continuous-time function and its samples, i.e. when do the samples form a unique representation of the continuous-time function? To answer this question, we are going to use all the tools in the toolbox that we have looked at so far:

- Hilbert spaces
- projections
- filtering
- sinc functions
- and so on.

Everything comes together in this module to develop a profound and very useful result, the **sampling theorem**.
Before moving to the heart of the topic, let us briefly review its history. The Shannon sampling theorem has a very interesting history which goes back well before Shannon. Numerical analysts were concerned about interpolating tables of functions and the first one to proove a version of the sampling theorem was Whittaker in England in 1915. Harry Nyquist at Bell Labs came up with the Nyqvist criterion, namely that a function that has a maximum frequency $\mathrm{F0F_{0F0}}$ could be sampled at $\mathrm{2F02F_{02F0}}$. In the Soviet Union, Kotelnikov proved a sampling theorem. The son of the first Whittaker further proved results on the sampling theorem. Then Herbert Raabe in Berlin wrote his PhD thesis about a sampling theorem that, wrong time wrong city, he got zero credit for. Denis Gabor worked on a version of the sampling theorem in the mid 1940s. Then Claude Shannon, the inventor of information theory, wrote a beautiful paper that is in the further reading for this class where the Shannon sampling theorem appears in the form that we use today. Last but not least, in 1949 Someya in Japan also proved the sampling theorem. You can see that it's a very varied history, it's a fundamental result where many people independently came up with this result.

## 7.1 The Continous-Time World

### 7.1.1 Introduction

The continuous-time world is the world we live in, the physical reality of the world, in contrast with the discrete-time world, the world inside a computer. We are first going to look at models of the world and compare digital with analog views of the world. Then we are going to study continuous-time signal processing in greater details. Furthermore, we will introduce the last form of Fourier transform we have not yet encountered in this class, the continuous-time Fourier transform.

### 7.1.2 The continous-time paradigm

Table 4: Two views of the world

| **Digital World** | **Analog World** |
|---|---|
| countable integer index n | real-valued time t [sec] |
| sequences $x[n] \in \ell_2(\mathbb{Z})$ | function $x(t) \in L_2(\mathbb{R})$ |
| frequency $\omega \in [-\pi, \pi]$ | frequency $\Omega \in \mathbb{R}(rad/sec)$ |
| DTFT: $\ell_2(\mathbb{Z}) \rightarrow L_2[-\pi, \pi]$ | FT: $L_2(\mathbb{R}) \rightarrow L_2(\mathbb{R})$ |

Emacs 24.3.1 (Org mode 8.2.5h)