Contents

1 Week 5 Module 4: Part 1 Introduction to Filtering

1.1 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 .

1.1.1 Linearity

Linearity is expressed by the equivalence

$$\mathfrak{H}\big\{\alpha \, x_1[n] + \beta \, x_2[n]\big\} = \alpha \, \mathfrak{H}\big\{x_1[n]\big\} + \beta \, \mathfrak{H}\big\{x_2[n]\big\} \tag{1}$$

- Fuzz-Box, example for a none linear device

TODO Add calculation examples

1.1.2 Time invariance

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

$$y[n] = \mathfrak{H}\big\{x[n]\big\} \Leftrightarrow \mathfrak{H}\big\{x[n - n_o]\big\} = y[n - n_o] \tag{2}$$

- Wah-Pedal, example of a time variant device

TODO Add calculation examples

1.1.3 Convolution

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

$$h[n] = \mathfrak{H}\big\{\delta[n]\big\} \tag{3}$$

> 📖 **Impulse Response**
>
> | Impulse response fully characterize the LTI system!

We can always write

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \tag{4}$$

by linearity and time invariance

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]\,h[n-k] \tag{5}$$

$$= x[n] * h[n] \tag{6}$$

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.

$$(x * y)[n] = \langle x^*[k], y[n-k] \rangle$$

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

1.2 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

3

1.2.1 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
$$(x[n] * y[n]) * w[n] = x[n] * (y[n] * w[n])$$

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

1.2.3 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 sequences. **FIR** Filter are always stable, since only in the convolution sum only a finite number of terms are involved.

1.2.4 Filtering by Example

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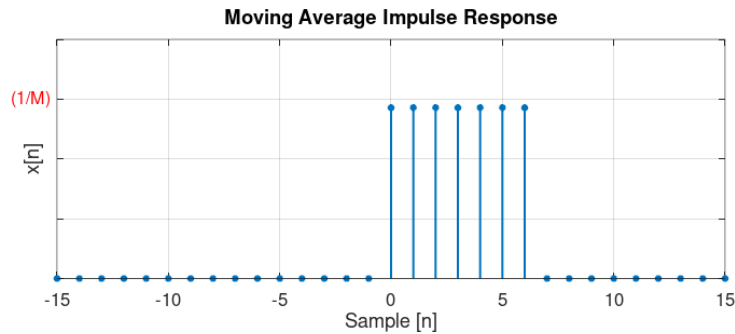
$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$
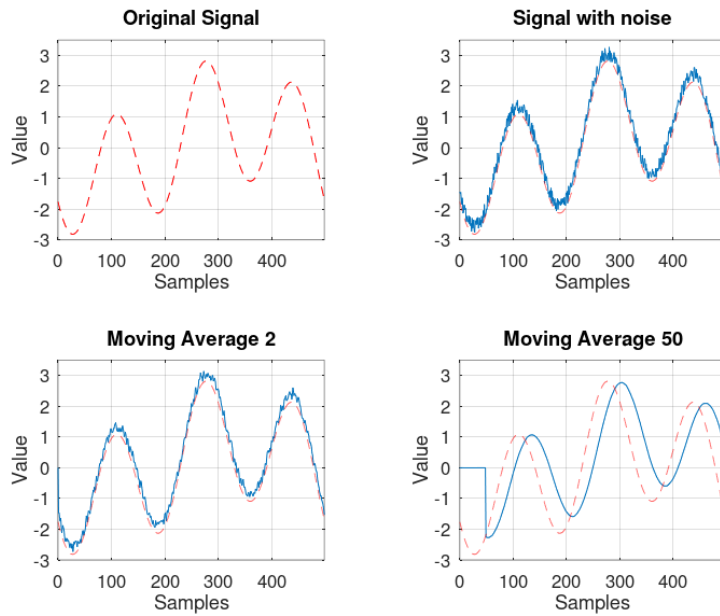
Impulse Response

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

5

```matlab
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
```


Moving Average Impulse Response

MA Analysis

- soomthin effect is proportional to M

- number of operations and storage also proportional to M

From the MA to first-order recursion

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

$$M_{y_{M[n]}} = x[n] + (M-1)y_{M-1}[n-1]$$

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

$$y_M[n] = \lambda y_{M-1}[n-1] + (1-\lambda)x[n], \ \lambda = \frac{M-1}{M}$$
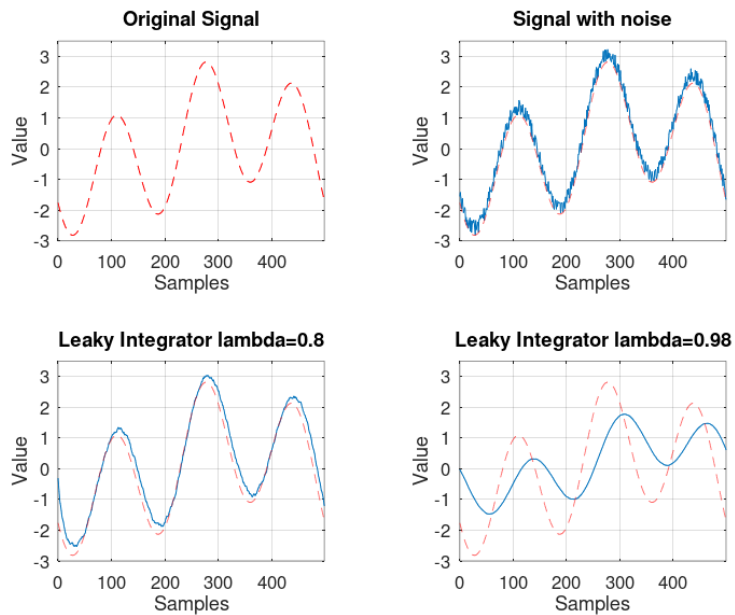
IIR Filter: The Leaky Integrator

- when M is large, $y_{M-1}[n] \approx y_M[n]$ and $(\lambda \approx 1)$

- the filter becomes: $y[n] = \lambda y[n-1] + (1-\lambda)x[n]$

6

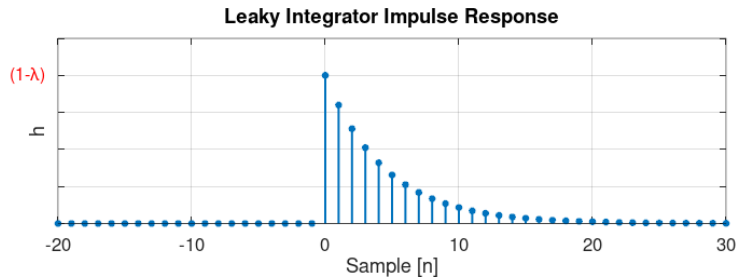- the filter is now recursive, since it uses its previous output value

```matlab
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
```


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

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

7

Leaky Integrator Impulse Response

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

The leaky integrator why the name

- Discrete Time integrator is a boundless accumulator

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

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

To prevent "explosing" we scale the accumulator with λ:

$\lambda y[n-1]$	keep only a fraction λ 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

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

1.3 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

8

- 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)

1.4   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.

1.5   Frequency Response

1.5.1   References

- Signal and System for Dummies: Frequency Response

1.5.2   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.

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

- 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.

1.5.3 Magnitude and phase

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

amplitude	A	phase shift	θ
amplification	>1	delay	<0
attenuation	$0 \le A < 1$	advancment	>0

1.5.4 The convolution theorem

The convolution theorem summerizes this result in

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

1.5.5 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

1.5.6 Example of Frequency Response: Moving Average Filter

The difference equation from M-point averager is
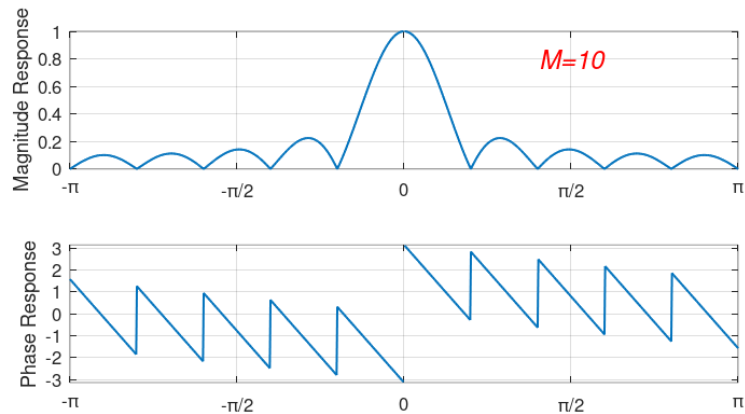
$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$

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 $\frac{sin(\frac{\omega}{2}M)}{sin(\frac{\omega}{2}M)}$
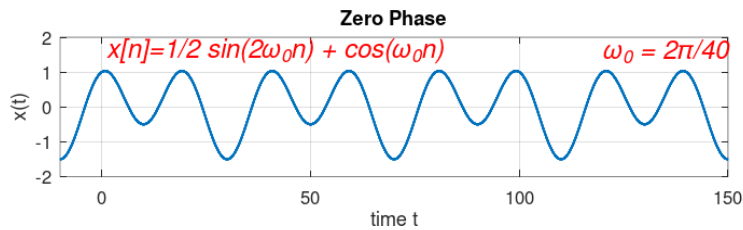
The Magnetute response of the moving average filter

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

1.5.7   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.
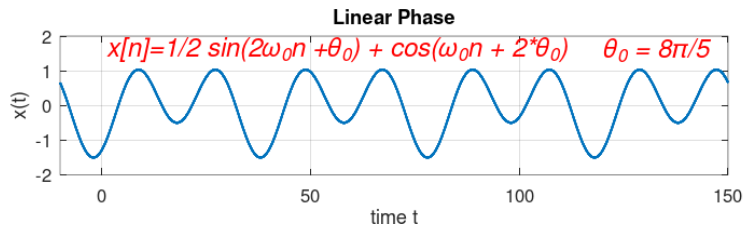
- non linear phase: which covers all the other properties now the shape of the signal in the time domain changes.
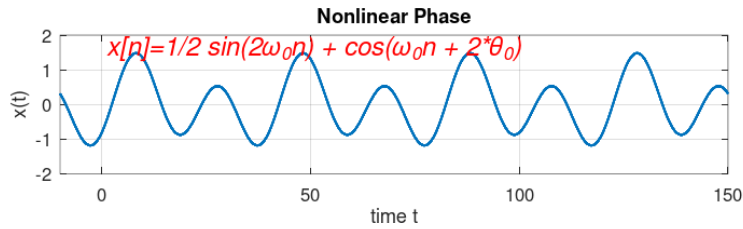
Spectrum

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

1.5.8 Linear Phase

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

1.5.9 Moving Average is linear Phase

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

1.5.10 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

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

13

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

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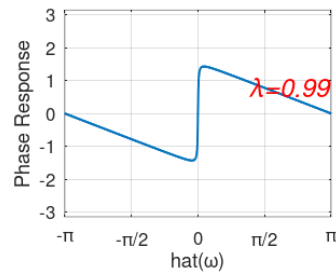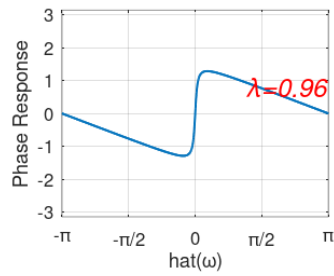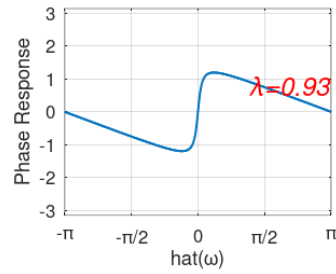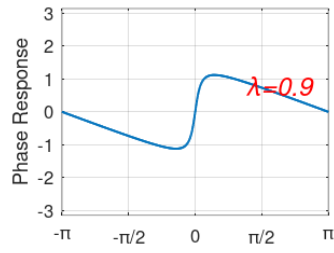
$$\angle x = tan^{-1}\left[-\frac{a}{b}\right]$$

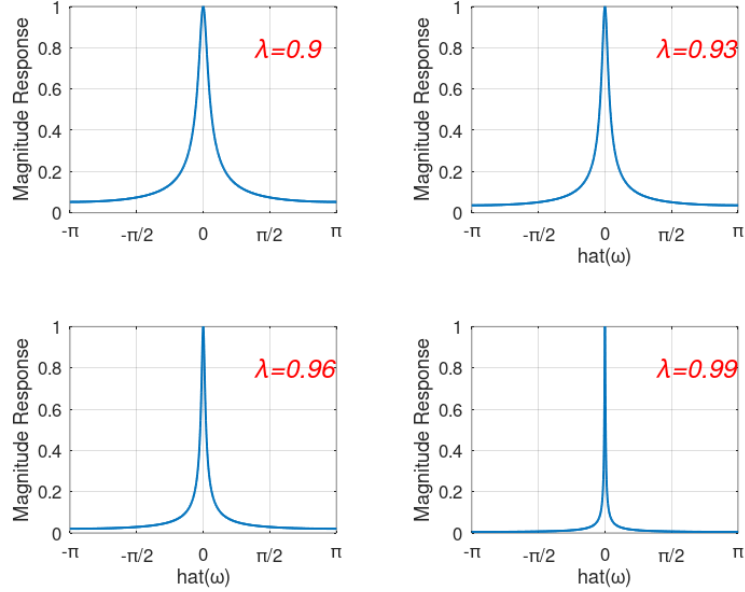$$H(e^{j\omega}) = \frac{1 - \lambda}{(1 - \lambda cos\omega) - jsin\omega}$$

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

14

1.5.11 **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.
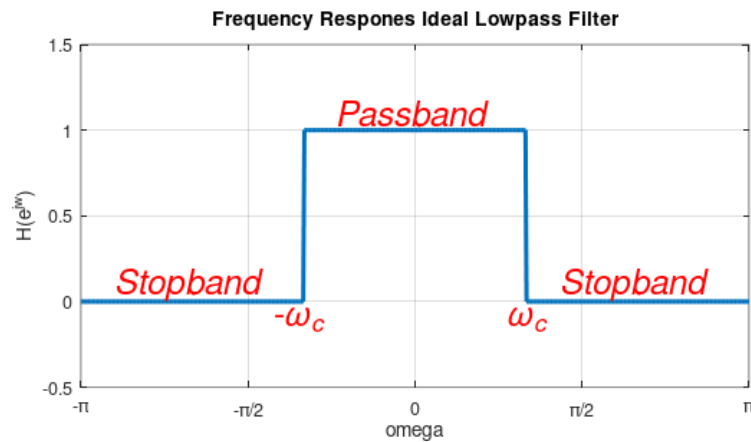
With Sawtooth Wave

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

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

1.6 Ideal Filters

1.6.1 The ideal lowpass filter frequency response

1.6.2 Ideal lowpass filter impulse response

- Lets low frequencies go through

- Attenuates i.e. kills high frequencies

Cut off Frequency ω_c - the frequency response transitions from 1 to zero

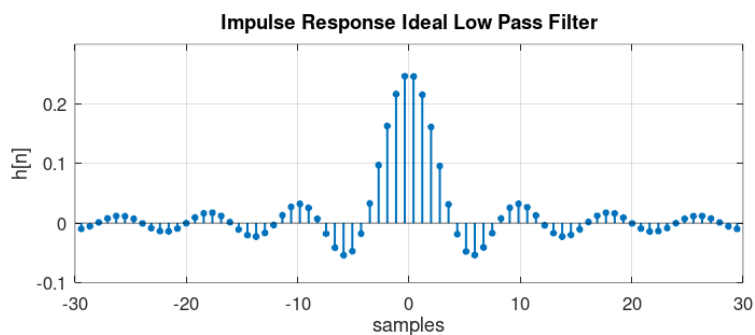**Passband** $\omega_b = 2\omega_c$

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

- 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

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

Impulse Response Ideal Low Pass Filter

- 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 $\left(\dfrac{1}{n}\right)$, we need a lot of samples for a good approximation.

Impulse Response: From normlized Algorithm to Octave Implementation

$$\frac{sin(\omega_c\, n)}{\pi\, n} = \frac{\omega_c}{\pi} \cdot sinc(n\frac{\omega_c}{\pi});$$

$$= \frac{\frac{\pi}{c}}{\pi} \cdot sinc(n\frac{\frac{\pi}{c}}{\pi});$$

$$= \frac{1}{c} \cdot sinc(n\frac{1}{c});$$

$$= \frac{1}{c} \cdot sinc(\frac{n}{c});$$

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 $\frac{1}{2}$

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

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

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

1.6.3   Example

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

Frequency Response Ideal Low Pass Filter

Impulse Response Ideal Low Pass Filter
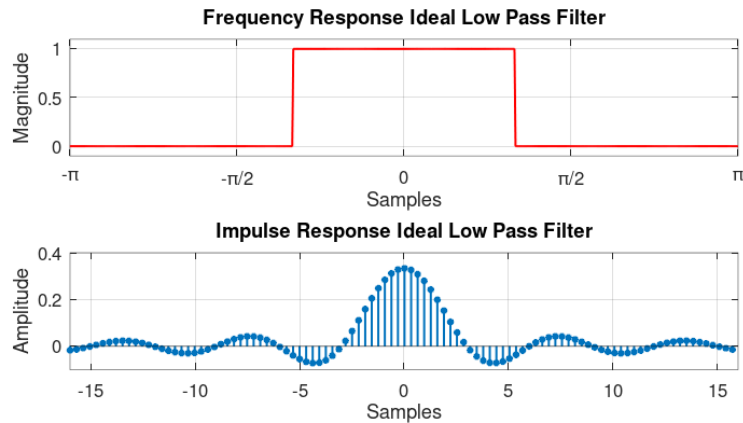
1.6.4 **TODO** Ideal filters derived from the ideal low pass filter

1.6.5 **TODO** Demodulation revisted

1.7 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: $\frac{48000\text{samples}}{second} \frac{16 bits}{samples} = 768 kbits/s$
  - Duration: 60s
  - Mono Raw Data Storage Usage: $60s * 76.8 kbits/s = 46 Mbit = 5.8 MByte$
  - Stereo Raw Data Storage Usage: $2 * 5.8 MBytes = 12 MBytes$

19

– MP3 Compressed Storage Usage: $1.5MBytes$
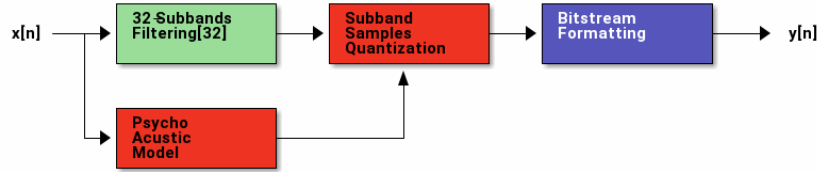
- Clever Quantiziation Scheme: Number of bits allocated to each sub-band is dependent on the perceptual importance of each sub-band with respect to overall quality of the audio wave-form

- Masking Effect of the human auditory system.

1.7.1 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

1.7.2 Subband Filter

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

1.8 Programing Assignment 1

```python
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 non

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

    return Y_db
```

```python
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()
```