Lecture 3, Multirate Signal Processing

Frequency Response

If we have coefficients of an Finite Impulse Response (FIR) filter h, or in general the impulse response, its frequency response becomes (using the Discrete Time Fourier Transform):

$$H(\omega) = \sum_{n=-\infty}^{\infty} h(n) e^{-jn\omega}$$

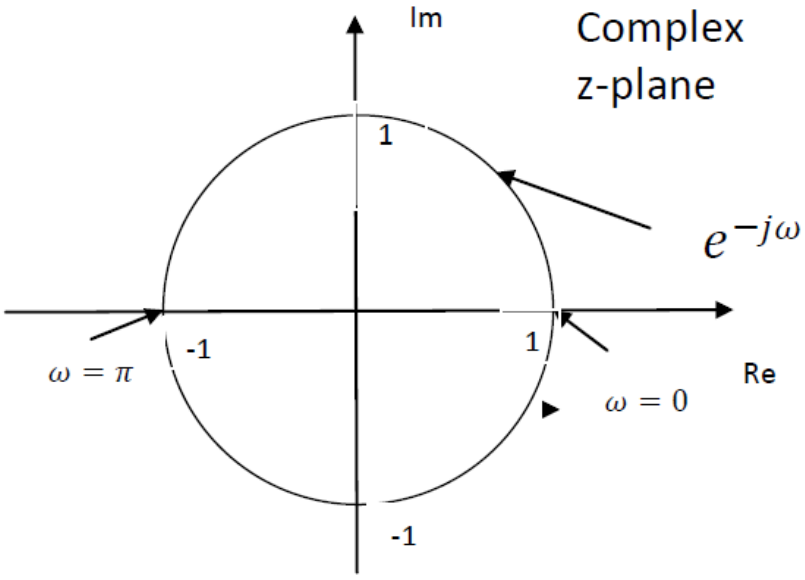
So here we already have 2 different ways to describe a system or a signal, by its **impulse response** in the time or space domain (depending on if it is audio or images), and by its **frequency response**. They are related by this Discrete Time Fourier Transform.

Example: If we have an amplifier which is described by its **frequency response** (including the phase), then we can obtain the time domain version, its **impulse response** by taking the inverse Fourier Transform of its frequency response. This can be seen as the output of our amplifier if the input is a single impulse (because an impulse has a flat spectrum, and hence the name "impulse response"). The output of the amplifier is then the convolution of the input signal (in time-domain) with the impulse response.

In practice usually an impulse is not used (not enough energy or it could destroy the system) but often white noise is input into our amplifier, and then the output is Fourier transformed to obtain its frequency response. White noise has the advantage that it has more energy without destroying the system, and it has an approximately constant spectrum (like the impulse). Another often used method is a sweeping sinusoid, with increasing frequency and constant amplitude. This often gives more precise results but takes longer to measure. The resulting measured frequency response is then inverse Fourier transformed to obtain its impulse response.

If we have the **z-Transform**, we obtain the **frequency response** of our system or signal if we replace $z = e^{j\omega}$.

In this case the z-Transform becomes the DTFT (the transform formulas become identical). This is like "reading out" the z-Transform along the unit circle in the complex domain of the z-Transform:



The periodicity of our frequency domain can also be seen easily here.

Here we can see that in general we obtain a 2π periodicity in the frequency domain. For **real valued** signals, the upper half of the circle has a frequency response which is symmetric to the lower half (**conjugate complex**), and hence strictly speaking we get some sort of pi periodicity for real valued signals (hence Nyquist's Theorem which states that we only can reconstruct frequencies below half the sampling frequency for real valued signals).
Hence for real valued signals only one half of the unit circle is unique. For **complex** signals, **both** halves of the unit circle can contain different frequency responses, and hence there we indeed have 2π periodicity, and we could reconstruct frequencies up to 2π .

We can **imagine complex signals** or frequencies as consisting of 2 signals, one for the real part and one for the imaginary part. For instance in stereo audio signals, the left channel could be seen as a real part and the right channel as an imaginary part of a complex audio signal. The same for negative frequencies: $e^{-j\cdot\omega\cdot n}$ They consist of a cosine real part and a -sine imaginary part.

This unit circle, if it describes an input signal X(z) (for instance a certain frequency), is the input to the z-

Transform H(z), by replacing the z by $e^{j\omega}$, to obtain the frequency response $H(e^{j\omega})$. You can imagine for instance the magnitude of the resulting frequency response as an additional dimension on top of this circle.

If you plot the frequency response for real valued signals, for instance in Python, what is usually done is to plot the upper half of this unit circle. For real valued signals the lower half is symmetric to it.

Example: Low Pass filter as Moving Average

If our input is the signal x(n) and the output of our low pass filter is y(n), then it can be computed as

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

What is the corresponding impulse response of this filter? It is the response of this system to a unit impulse (x(0)=1, x(n)=0 elsewhere):

$h(0)=y(0)=\frac{1}{2}$, $h(1)=y(1)=\frac{1}{2}$. This can also be written as a vector:

$$h=\left[\frac{1}{2},\frac{1}{2}\right]$$

Hence the output is the **convolution** with this vector, $y=x*h$.

Observe: The convolution or filtering operation can also be written as a multiplication of the signal vector and a so-called **Sylvester Matrix:**

$$\begin{bmatrix} \vdots \\ y(0) \\ y(1) \\ y(2) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \cdots & . & . \\ 0.5 & 0.5 & 0 & . \\ 0 & 0.5 & 0.5 & \cdots \\ 0 & 0 & 0.5 & 0.5 \\ . & \vdots & \cdots & . \end{bmatrix} \cdot \begin{bmatrix} x(-1) \\ x(0) \\ x(1) \\ \vdots \end{bmatrix}$$

We also use the following matrix formulation for convolution,

$$\begin{bmatrix} \vdots \\ y(0) \\ y(1) \\ y(2) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \cdots \\ x(-1) & x(0) \\ x(0) & x(1) \\ x(1) & x(2) \\ \vdots & \cdots \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

What is the frequency response of this moving average low pass filter? The z-transform of this filter is

$$H(z) = \sum_{n=0}^{\infty} h(n) z^{-n} = 0.5 \cdot z^{-0} + 0.5 \cdot z^{-1}$$

Replacing z by $e^{j\omega}$ gives us the frequency response of this low pass filter,

$$H(\omega) = \sum_{n=-\infty}^{\infty} h(n) e^{-jn\omega} = 0.5 + 0.5 \ \ e^{-j\omega}$$

This is the complex frequency response. Often we are interested in the magnitude (or absolute value) of this frequency response, because we would like to see how much a signal at a certain frequency is attenuated. An example might be an equalizer filter, which has the goal to obtain an overall frequency response which has a flat magnitude, for instance to equalize an audio amplifier. If the amplifier has e.g. an attenuation of 0.5 at a certain frequency, we need the equalizer filter to have an amplification of magnitude 2 at that frequency. In this way we obtain a constant magnitude of amplification at all frequencies. But observe that the phase of our complex frequency response can still be different for all frequencies.
So our magnitude frequency response is,

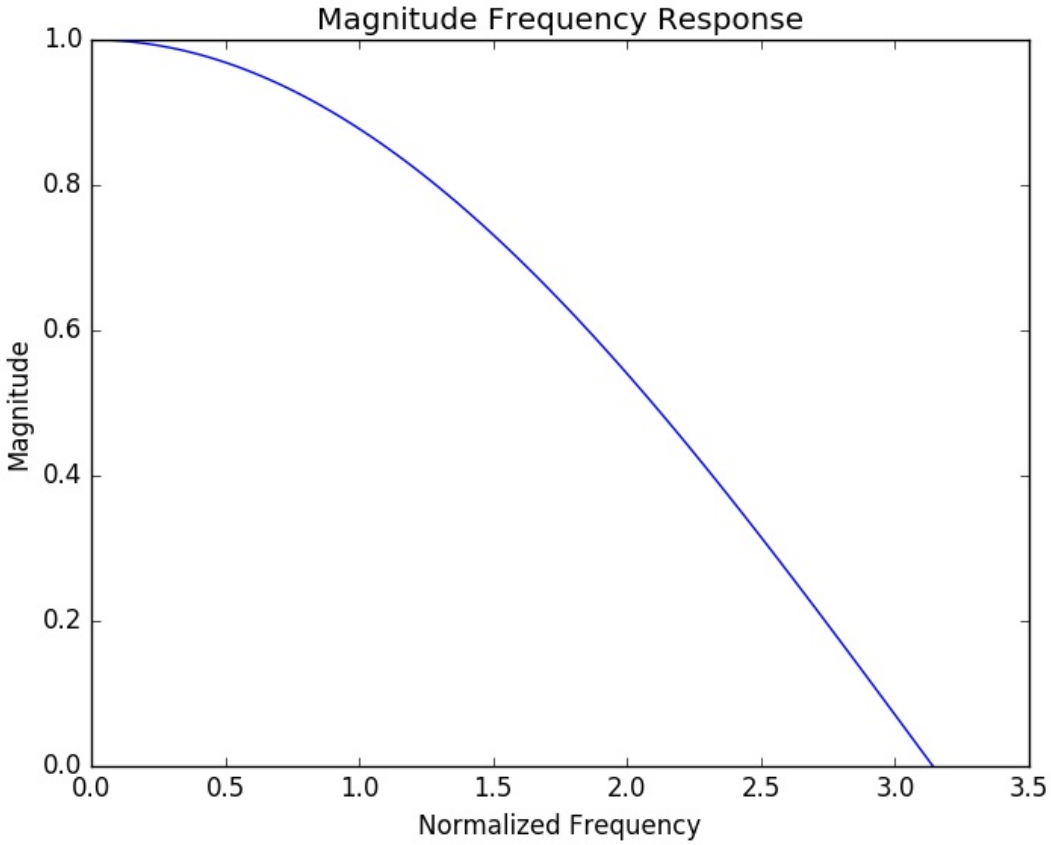
$$|H(\omega)|=|0.5+0.5\ \ e^{-j\omega}|$$

We can compute the frequency response using Python, using the z-Transform:

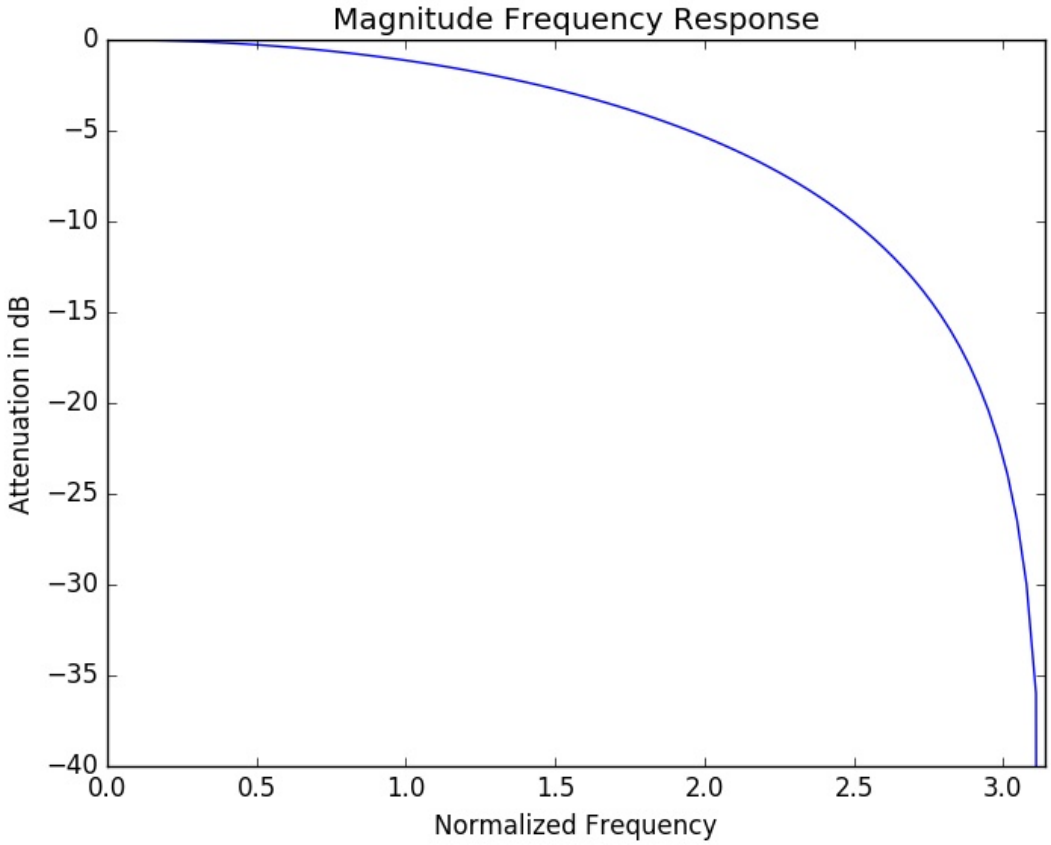
```
ipython --pylab
#Impulse response:
h=[0.5,0.5]

import sympy
z=sympy.symbols('z')
#z-Transform:
Hz=sympy.Poly(np.flipud(h), z**(-1))
```

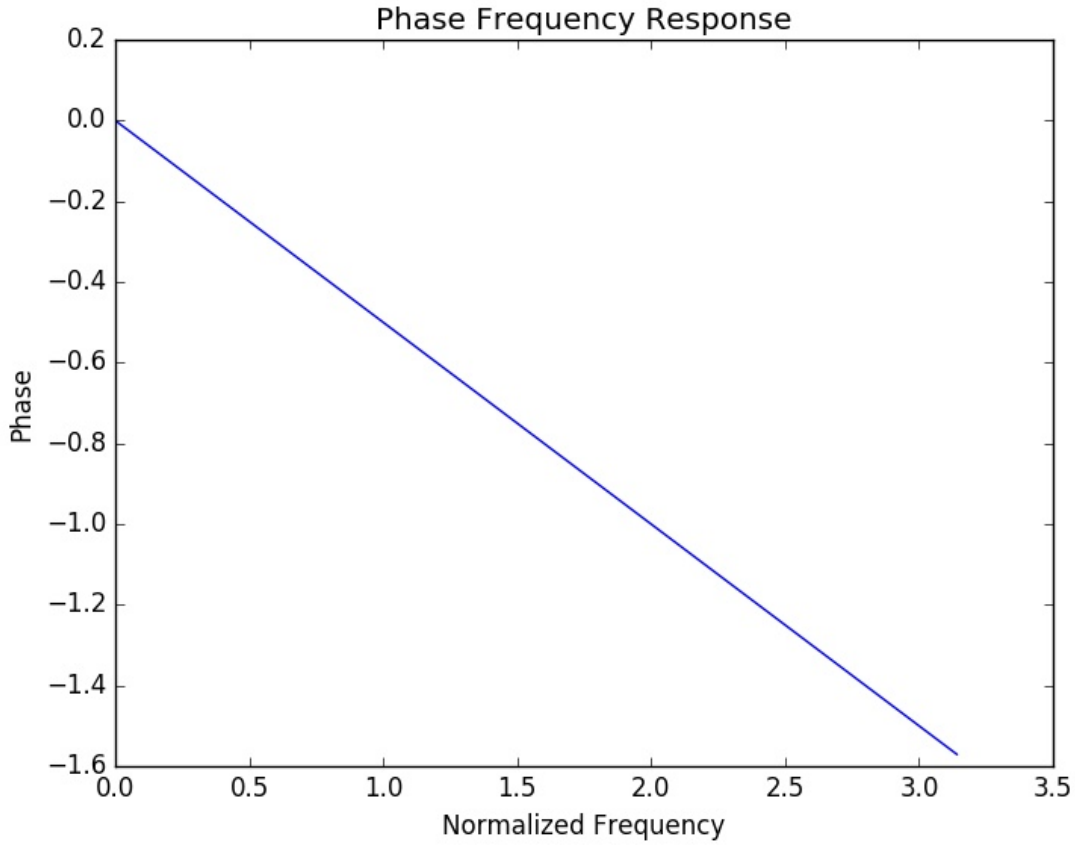
```
#The desired omegas,
#100 sample points of angle between 0 and pi:
omega=linspace(0,pi, 100)
#initialize freq. Resp. with complex
#zeros:
H=zeros(len(omega))*1j
#replace z by exp(-1j*omega) (we replace z^-1 in the argument):
for k in range(len(omega)):
   H[k]=Hz(exp(-1j*omega[k]))
#magnitude response:
plot(omega,abs(H))
xlabel('Normalized Frequency')
ylabel('Magnitude')
title('Magnitude Frequency Response')
```




```
#The dB scale for the magnitude shows
#more detail in the stop band:
plot(omega,20*log10(abs(H)))
axis([0,3.14,-40,0])
xlabel('Normalized Frequency')
ylabel('Attenuation in dB')
title('Magnitude Frequency Response')
```




```
#phase response:
plot(omega,angle(H))
xlabel('Normalized Frequency')
ylabel('Phase')
title('Phase Frequency Response')
```




```
#Or, using the freqz function:
import scipy.signal as signal
omega, H =signal.freqz(h)
```

where `H` is an array of the complex values of the frequency response, and `omega` is an array containing the normalized frequencies at which the frequency response was evaluated.

We see that at frequency 0 the magnitude response has the magnitude 1, hence no attenuation (important for a low pass). At frequency $\omega = \pi$ it has magnitude 0, which is also suitable for a **low pass**. In between it behaves like a cosine does, it slowly reduces, so not really like a perfect low pass, which should have frequency response which looks more like brick shaped (1 at the entire passband, 0 at the entire stop band, ideally). So it is **far from** being a **perfect** low pass filter.

We also have the phase of the frequency response. We see that it is linear function, which is particularly important for image processing. The phase is the angle of $H(\omega)$.

The vertical axis on a dB scale let us see the **small values of the stop band** better, which works for both power and voltage/current.
Observe: The dB numbers are using the input of the filter as a reference. Hence the dB number results from 20*log10 of output “voltage” over input “voltage”.

Remark: You need to be familiar with **dB computations**. You need to be able to answer e.g. following questions:

- What is the definition of dB for voltage and for power?
- What is the („voltage“) ratio of 0.1 in dB?
- What is the (voltage) ratio of -20 dB?
- What is the (voltage) ratio of 6 dB?

- You have 2 systems in a row, the first with -6 dB attenuation, the next with -10 dB attenuation. What is the total atennuation?

If you have trouble with one of the questions, look it up at Wikipedia or in literature.

Now also observe the **phase** plot. We see the phase response is $-\omega/2$, which is linear in ω with a slope of ½, which is why we have a **linear phase** in this case. Hence we also call our filter a **linear phase filter**.

High Pass Example

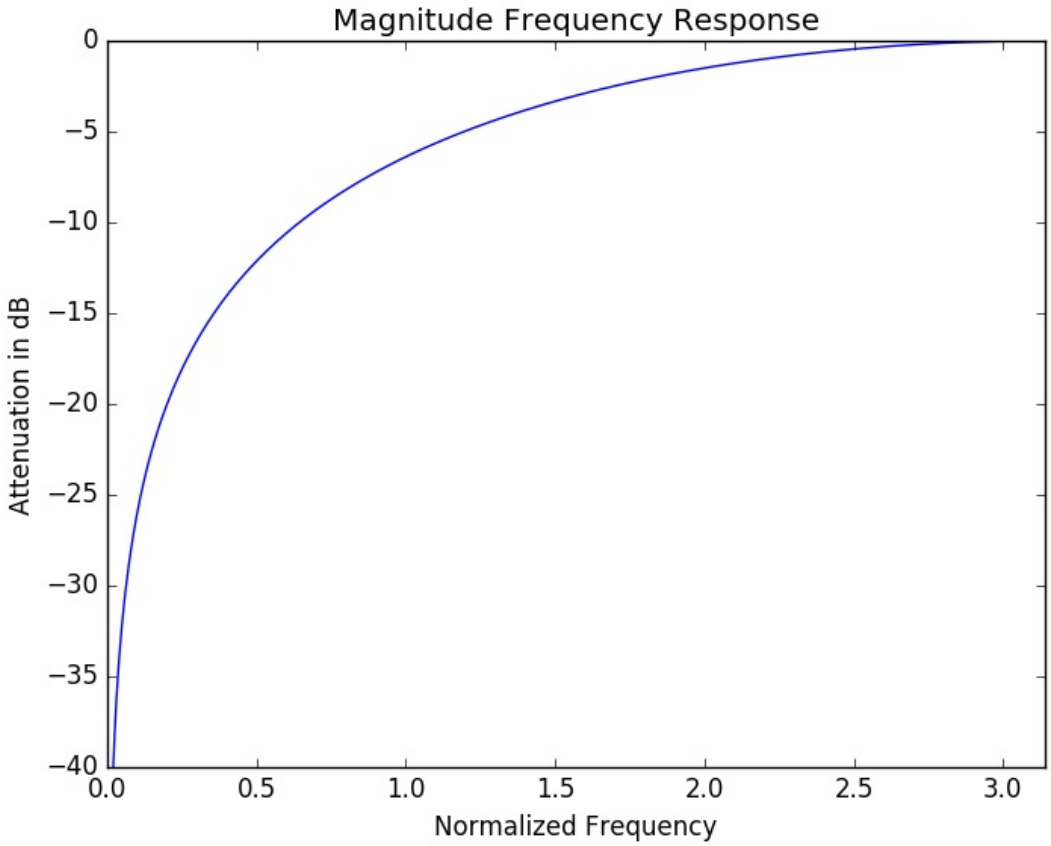
We can do the same analysis for the high pass filter with the impulse response

$$h=[0.5,\ -0.5]$$

In the same way as for the low pass, we obtain its magnitude of the frequency response in Python as

```
ipython --pylab
import scipy.signal as signal
h=[0.5, -0.5]
omega, H=signal.freqz(h)
plot(omega,20*log10(abs(H)))
axis([0,3.14,-40,0])
xlabel('Normalized Frequency')
```

```
ylabel('Attenuation in dB')
title('Magnitude Frequency Response')
```



By looking at frequency $\omega = 0$ we obtain a magnitude of 0, and at $\omega = \pi$ we obtain a magnitude of 1, as we would expect from a high pass. But again, between those points it behaves like a sinusoid, far from a perfect high pass.

This looks like the (frequency) flipped version of our low pass filter.