BME I5100: Biomedical Signal Processing

Linear systems, Fourier and z-Transform

Lucas C. Parra
Biomedical Engineering Department
City College of New York

CCNY

Schedule

Week 1: Introduction
Linear, stationary, normal - the stuff biology is **not** made of.

Week 1-4: Linear systems (mostly discrete time)
Impulse response
Moving Average and Auto Regressive filters
Convolution
Discrete Fourier transform and z-transform

Week 5-7: Random variables and stochastic processes
Random variables
Multivariate distributions
Statistical independence

Week 8: Electrophysiology
Origin and interpretation of Biopotentials

Week 9-14: Examples of biomedical signal processing
Probabilistic estimation
Linear discriminants - **detection** of motor activity from MEG
Harmonic analysis - **estimation** of hart rate in Speech
Auto-regressive model - **estimation** of the spectrum of 'thoughts' in EEG
Independent components analysis - **analysis** of MEG signals

z-Transform

The z-transform is a map of a discrete signal $x[n]$ into complex valued function $X(z)$ with complex valued z

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] z^{-n}$$

For a **finite** sequence the z-transform is defined for all z.

For **infinite** sequence it is only defined for values of z for which the sum converges.

z-transform is important because of the **Convolution Theorem**:

$$h[n] * x[n] \Leftrightarrow H(z) X(z)$$

z-Transform - Convolution theorem

$$h[n]*x[n] \Leftrightarrow H(z)X(z)$$

Because the z-transform of the convolution ...

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

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

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

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

$$= H(z)X(z)$$

Z-Transform - Difference model

We approximated the response of an IIR system by the difference model

$$y[n]=-\sum_{k=1}^{P} a[k]\,y[n-k]+\sum_{k=0}^{Q} b[k]\,x[n-k]$$

The z-transform relates the impulse response $h[k]$ to $a[k]$ and $b[k]$. Define $a[0]=1$, then we can rewrite:

$$\sum_{k=0}^{P} a[k]\,y[n-k]=\sum_{k=0}^{Q} b[k]\,x[n-k]$$

After z-transform on both sides we have

$$A(z)\,Y(z)=B(z)\,X(z)$$

z-Transform -rational impulse response

With the z-transform of $y[n]=h[n]*x[n]$: $Y(z)=H(z)X(z)$

We obtain $B(z)X(z)=A(z)Y(z)=A(z)H(z)X(z)$

$$H(z)=\frac{B(z)}{A(z)}$$

Impulse response is called **rational** if it can be expressed exactly as:

$$H(z)=\frac{B(z)}{A(z)}=\frac{\sum_{k=0}^{Q} b[k]z^{-k}}{1+\sum_{k=1}^{P} a[k]z^{-k}}$$

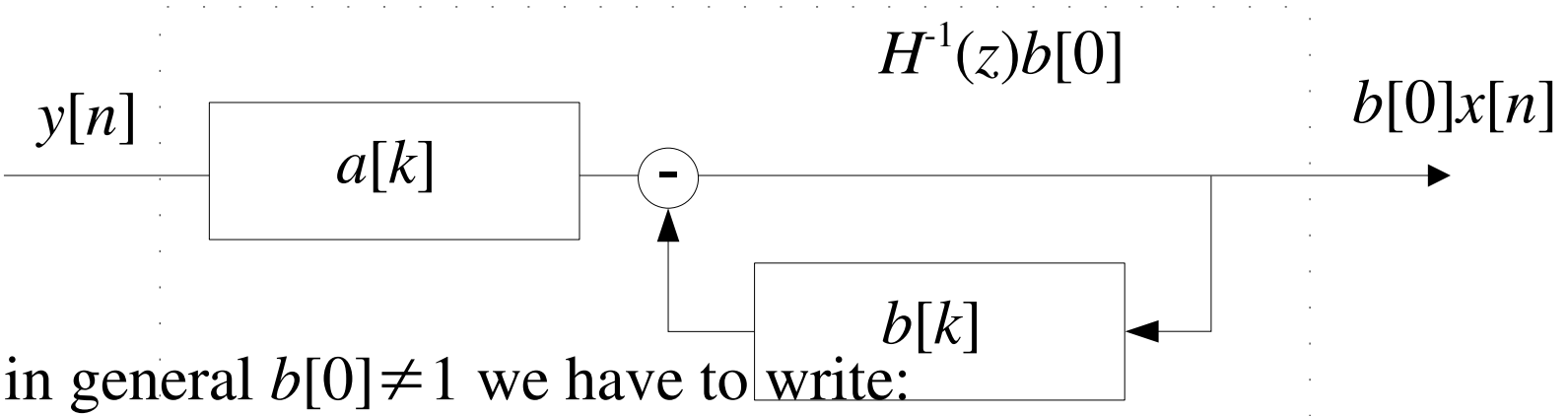
With a sufficient number of terms any $H(z)$ can be approximated by a rational **polynomial**.

z-Transform - Difference model and inverse

The z-transform of an ARMA filter and its inverse are

$$Y(z)=\frac{B(z)}{A(z)}X(z) \qquad X(z)=\frac{A(z)}{B(z)}Y(z)$$

The AR part becomes the MA and the MA becomes AR:



Since in general $b[0]\neq 1$ we have to write:

$$b[0]x[t]=-\sum_{k=1}^{Q} b[k]x[n-k]+\sum_{k=0}^{P} a[k]y[n-k]$$

z-Transform -rational impulse response

Note that any polynomial of order -Q can be decomposed into Q factors of the form, $(1\text{-}z_k z^{-1})$, with roots z_k.

$$\sum_{k=0}^{Q} b[k] z^{-k} = b[0] \prod_{k=1}^{Q} \left(1 - z_k z^{-1}\right)$$

Example:
Factor representation of square polynomial with roots z_1 and z_2

$$(z - z_1)(z - z_2) = z^2 - z(z_1 + z_2) + z_1 z_2$$

$$\left(1 - z_1 z^{-1}\right)\left(1 - z_2 z^{-1}\right) = 1 - (z_1 + z_2) z^{-1} + z_1 z_2 z^{-2}$$

Coefficients of the second order polynomial are then: $b[0] = 1, b[1] = -z_1 - z_2, b[2] = z_1 z_2$

For $b[k] \rightarrow z_k$ use

```
>> z=roots(b/b(1));
```

For $z_k \rightarrow b[k]$ use

```
>> b=poly(z)*b(1);
```

z-Transform -Pole-zero representation of IR

Since $a[k]$ and $b[k]$ define polynomials in z^{-1} this lead to the pole-zero representation of the impulse response:

$$H(z) = \frac{B(z)}{A(z)} = b[0] \frac{\prod_{k=1}^{Q} (1 - z_k z^{-1})}{\prod_{k=1}^{P} (1 - p_k z^{-1})}$$

For $P=0$ the system is called **all zeros**, for $Q=0$ it is called **all poles**.

See Bruce, page 296 for dependence of $h[n]$ on location of poles.

```
>> plot(filter(b(1)*poly(z),poly(p),[1 zeros(1,100)]))
```

A rational $H(z)$ is **stable** if the poles $|p_k| < 1$. Hence stability test:

```
>> abs(roots(a))<1
```

See Bruce for proof on page 450.

z-Transform

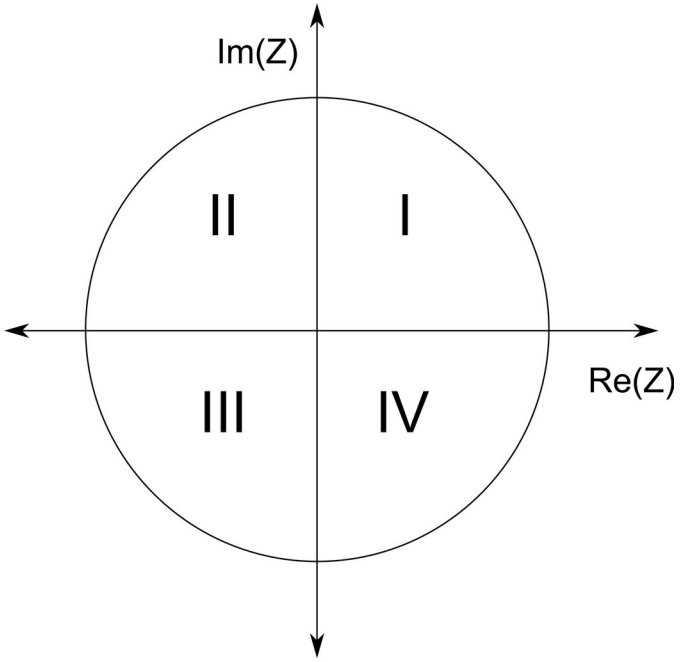
Homework: Poles location and Impulse response

Plot the impulse response for filters with the following **configuration of poles** (see below figure for definition of quadrants)

1. TWO conjugate poles, one in quadrant I and the conjugate in quadrant IV.
2. TWO conjugate poles, one in quadrant II and the conjugate in quadrant III.
3. ONE pole in quadrant I.
4. TWO conjugate poles outside of the unit circle.

The final output of your homework should be 4 figures, each figure should be the impulse response of each pole configuration tried.

For the **zeros**, you can try any combination you want.



Discrete Time Fourier Transform (DTFT)

The **Discrete Time Frourier Transform** (DTFT) is the z-transform on the unit circle $z = e^{j\omega} = \cos(\omega) + j\sin(\omega)$

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-jn\omega}$$

The DTFT is an invertible transformation

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} d\omega \, X(e^{j\omega}) e^{jn\omega}$$

This is simply because $\int_{2\pi} d\omega \, e^{-j\omega n} = 2\pi\delta_n$

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} d\omega \, X(e^{j\omega}) e^{jn\omega} = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} x[k] \int_{-\pi}^{\pi} d\omega \, e^{-j\omega(k-n)} = x[n]$$

Fourier Transform Summary

Fourier transform
(cont. time, cont. freq.)

$$F(\nu)=\int_{-\infty}^{\infty} dt\, f(t)\, e^{-j2\pi\nu t}$$

$$f(t)=\int_{-\infty}^{\infty} d\nu\, F(\nu)\, e^{j2\pi\nu t}$$

Discrete time FT
(disc. time, cont. freq.)

$$X(e^{j\omega})=\sum_{n=-\infty}^{\infty} x[n]\, e^{-jn\omega}$$

$$x[n]=\frac{1}{2\pi}\int_{-\pi}^{\pi} d\omega\, X(e^{j\omega})\, e^{jn\omega}$$

Fourier series
(cont. time, disc. freq.)

$$a_k=\frac{1}{2T}\int_{-T}^{T} dt\, x(t)\, e^{-jk\omega t}$$

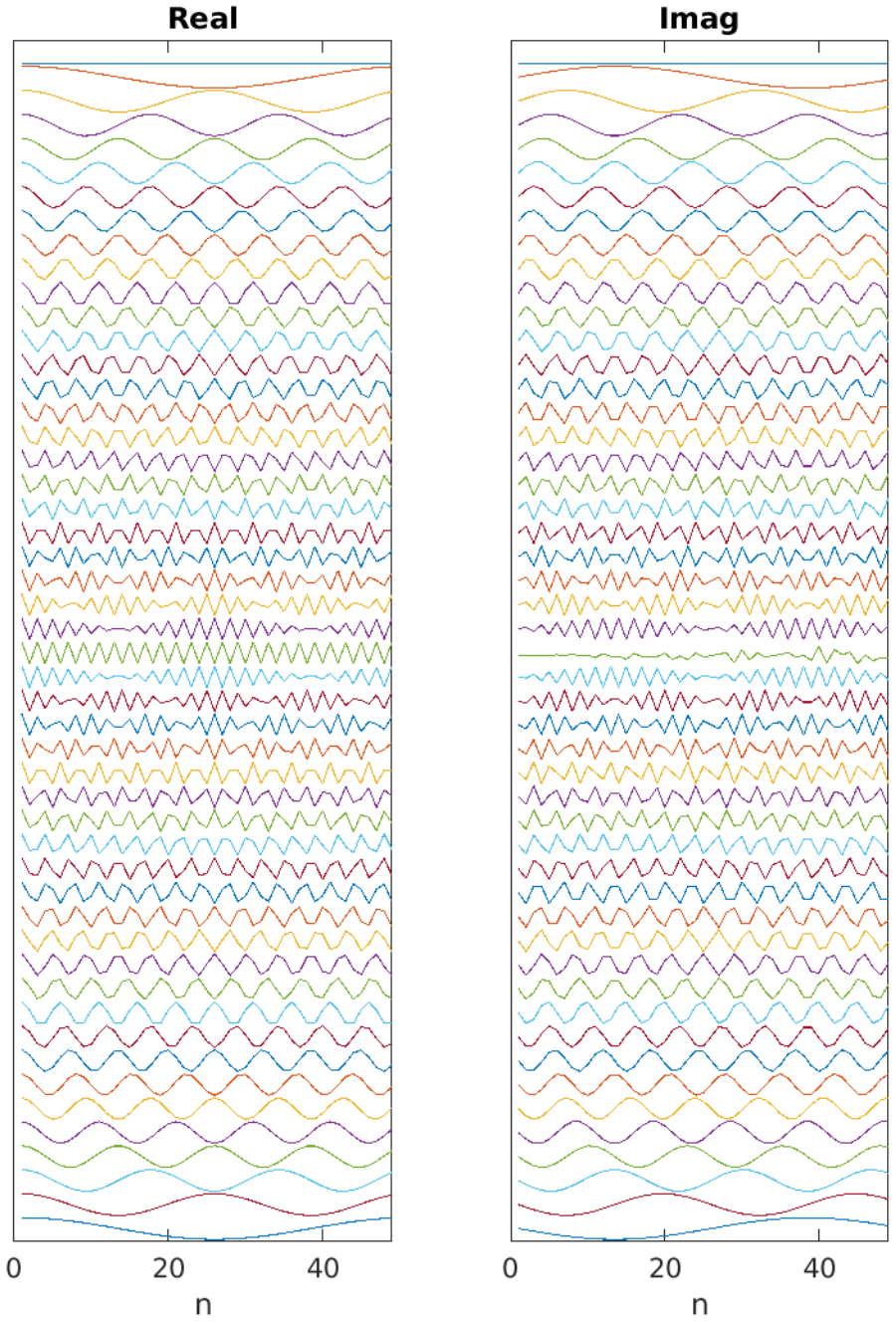
$$x(t)=\sum_{k=-\infty}^{\infty} a_k\, e^{jk\omega t}$$

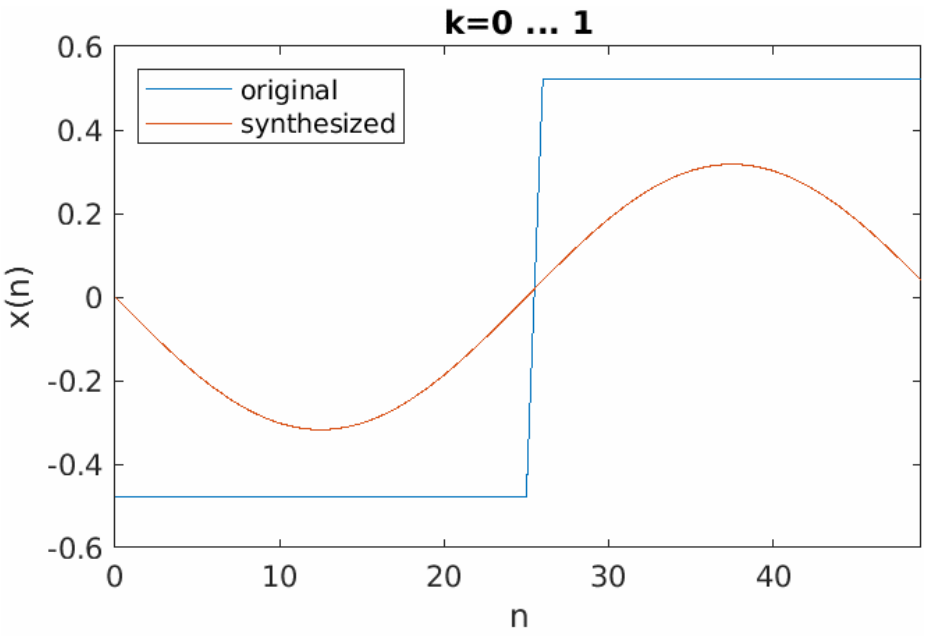
Discrete FT
(disc. time, disc. freq.)

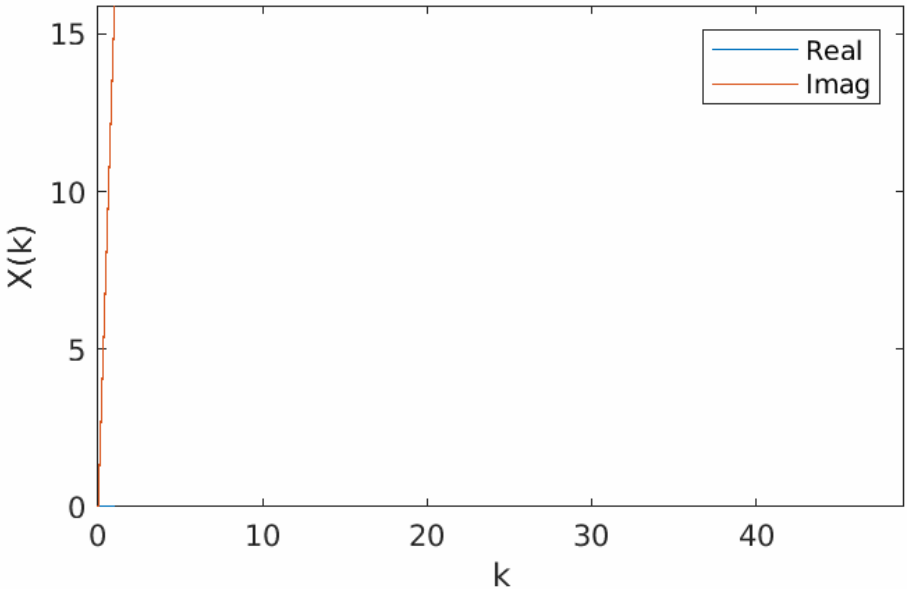
$$X[k]=\sum_{n=0}^{N-1} x[n]\, e^{-j\,2\pi\, kn/N}$$

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

Discrete Fourier Transform







DTFT - Properties

The following properties derive directly from the z-transform

$$x[n] \xrightarrow{\text{DTFT}} X(e^{j\omega})$$

Conjugation $x^*[n]$ $X^*(e^{-j\omega})$

Delay $x[n-n_0]$ $e^{-j\omega n_0} X(e^{j\omega})$

Time reversal $x[-n]$ $X(e^{-j\omega})$

Correlation

$$\sum_{n=-\infty}^{\infty} x[n+k]\,y^*[n] \qquad X(e^{j\omega})\,Y^*(e^{j\omega})$$

Conjugate symmetry for $x[n] \in \mathbb{R}$ $X(e^{j\omega}) = X^*(e^{-j\omega})$

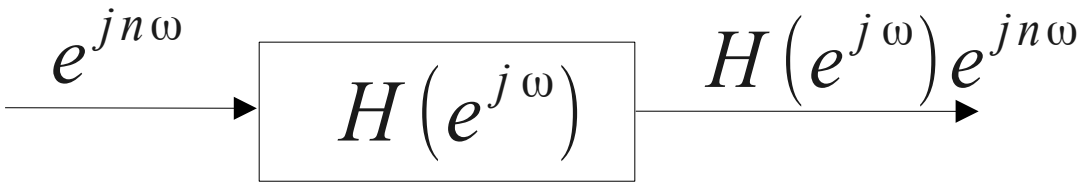
DTFT - System frequency response

Consider stationary oscillatory input to a LSI system $h[k]$:

$$x[n]=e^{jn\omega}$$

$$y[n]=\sum_{k=-\infty}^{\infty} h[k]x[n-k]=\sum_{k=-\infty}^{\infty} h[k]e^{j(n-k)\omega}=H\left(e^{j\omega}\right)e^{jn\omega}$$

The output is the input times the DTFT of the impulse response



The oscillation with frequency ω has been modified in phase Φ and amplitude A

$$A(\omega)=\left|H\left(e^{j\omega}\right)\right| \qquad \Phi(\omega)=arg\left(H\left(e^{j\omega}\right)\right)$$

$$H\left(e^{j\omega}\right)=Ae^{j\Phi}$$

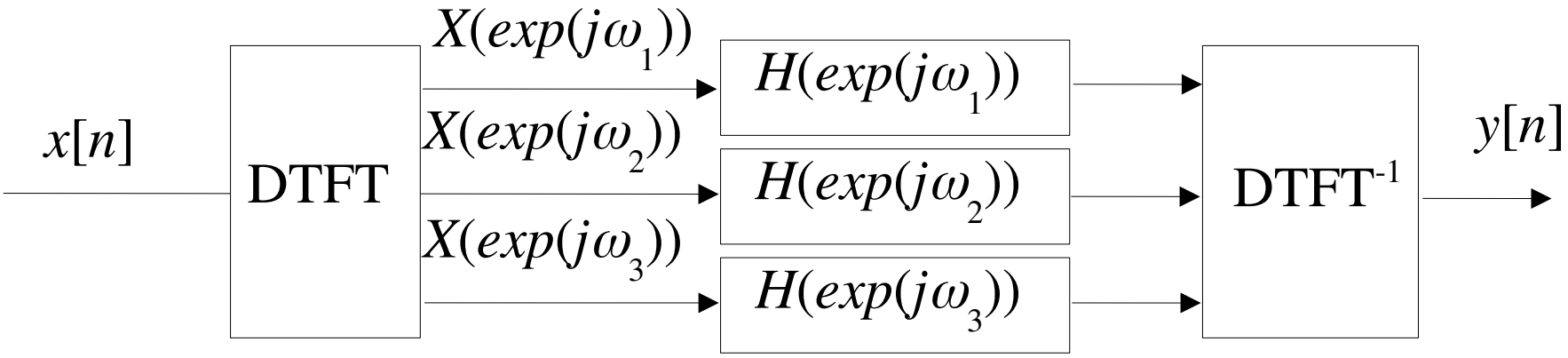
DTFT - System frequency response

DTFT inversion formula tells us that arbitrary input x[n] can be decomposed into sum of oscillations

$$x[n]=\frac{1}{2\pi}\int_{-\pi}^{\pi} d\omega\, X\left(e^{j\omega}\right)e^{jn\omega}$$

The system response to that is given by the convolution theorem

$$Y\left(e^{j\omega}\right)=H\left(e^{j\omega}\right)X\left(e^{j\omega}\right)$$



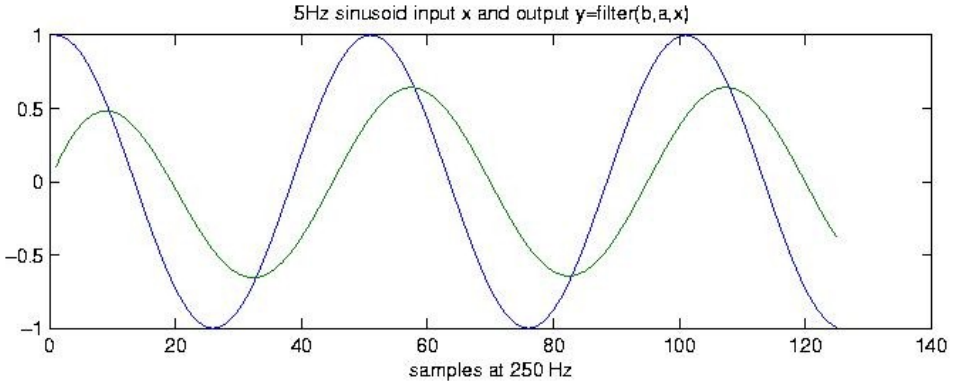
DTFT - System frequency response

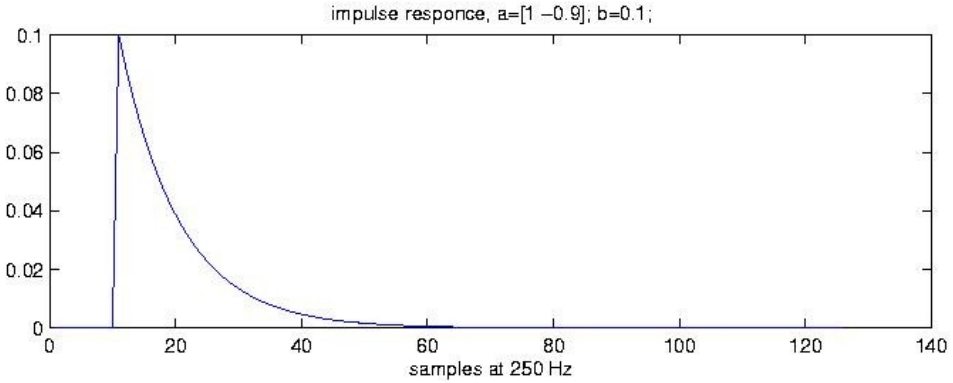
Example: $a[1] = -0.9,\ b[0] = 0.1$:

$$H(z) = \frac{0.1}{1 - 0.9 z^{-1}}$$

Time domain response

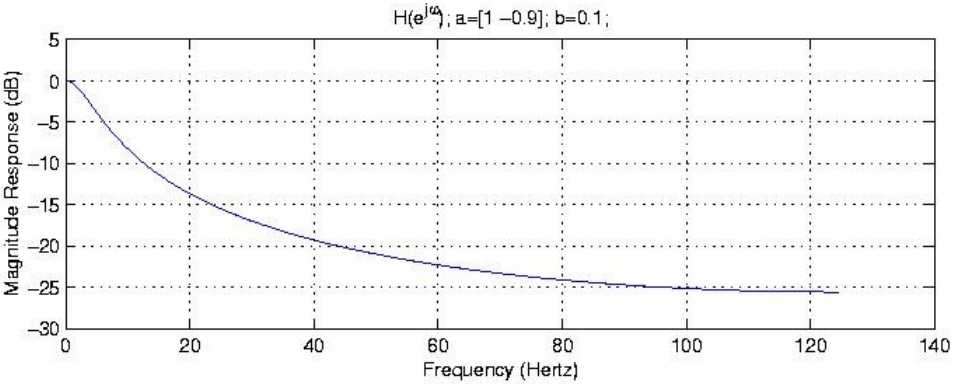
```
>> plot(filter(b,a,u))
```

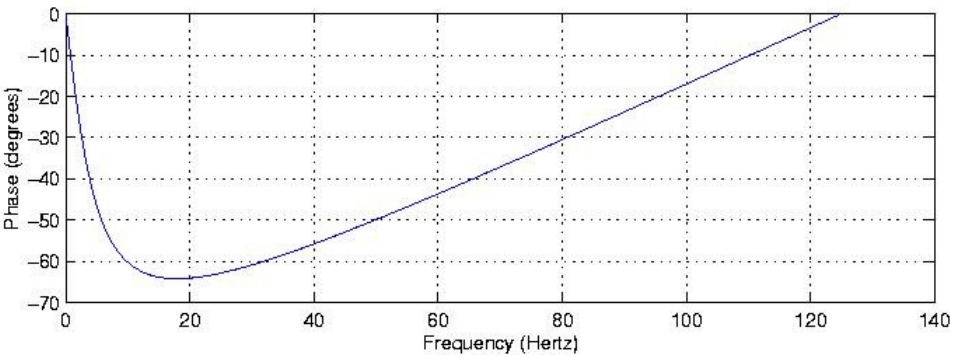




Frequency domain response

```
>> freqz(b,a)
```





DTFT - Filter design example

Goal: Highpass filter the DC drift in EEG with minimal latency and memory.

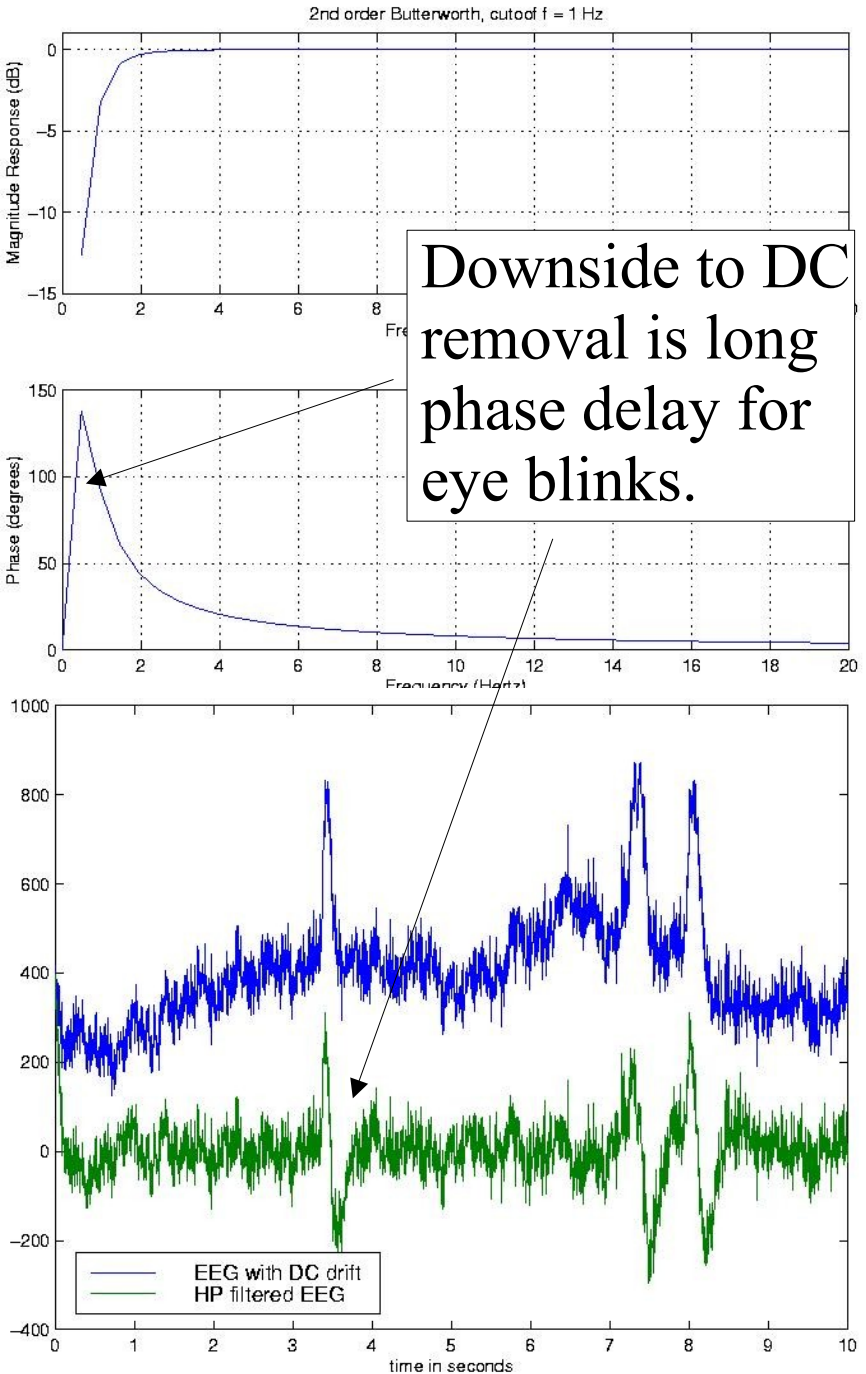
```
>> filterDesigner
```

Solution: Select IIR filter, 2nd order Butterworth with cutoff at 1Hz.

For IIR filter, save filter coefficients as "second order sections" and gain (SOS,G) and use sosfilt() instead of filter().

```
>> y=sosfilt(SOS,x)*prod(G);
```

SOS are 2nd order ARMA filters applied in sequence, which is numerical more robust than single ARMA of large order.



Assignment – design 60Hz notch filter

Design a 60Hz notch filter (all frequencies have unit gain except 60Hz). You can use filterDesigner at first, but then use matlab code to design the filter. Test this with the signal stored in gamma.mat. There is a variable called osc. The first channel has strong 60Hz noise.

signal = osc(:,1);

Apply the notch filter to this signal and show the spectrogram before and after using the 'spectrogram' function. (2 graphs)

Also show the magnitude, phase and impulse response (3 graphs).

Also show the signal itself before and after filtering, in the same graph (1 graph). So in total you will have 6 graphs.

If you are using an IIR filter apply the filter using the 'sosfilt' function. If you are using FIR filter, use the 'filter' function.

DTFT – Group delay

Phase delay $\varphi(\omega)$ corresponds to delay in time, Δt, that is frequency dependent:

Group delay in samples

$$n(\omega) = \frac{\phi(\omega)}{\omega}$$

Group delay in time

$$\Delta t(\omega) = \frac{n(\omega)}{f_s} \qquad f = \frac{\omega}{2\pi} f_s$$

To avoid phase distortions it is desirable to have a constant group delay, i.e. all frequency components are delayed by the same amount in time. This can be achieved with a *linear phase*:

$$n(\omega) = \frac{n_o \omega}{\omega} = n_o$$

DTFT - Zero phase and linear phase

A filter is said to have *zero phase* if it introduces no phase delay

$$H\left(e^{j\omega}\right)=\left|H\left(e^{j\omega}\right)\right|e^{j0}=\left|H\left(e^{j\omega}\right)\right|$$

True for all symmetric FIR filters: $h[-n]=h^{*}[n]$

A filter is said to have *linear phase* if

$$H\left(e^{j\omega}\right)=\left|H\left(e^{j\omega}\right)\right|e^{j\omega n_0}$$

Linear phase corresponds to shift in time. Because delay in time corresponds to a multiplication with a linear phase term:

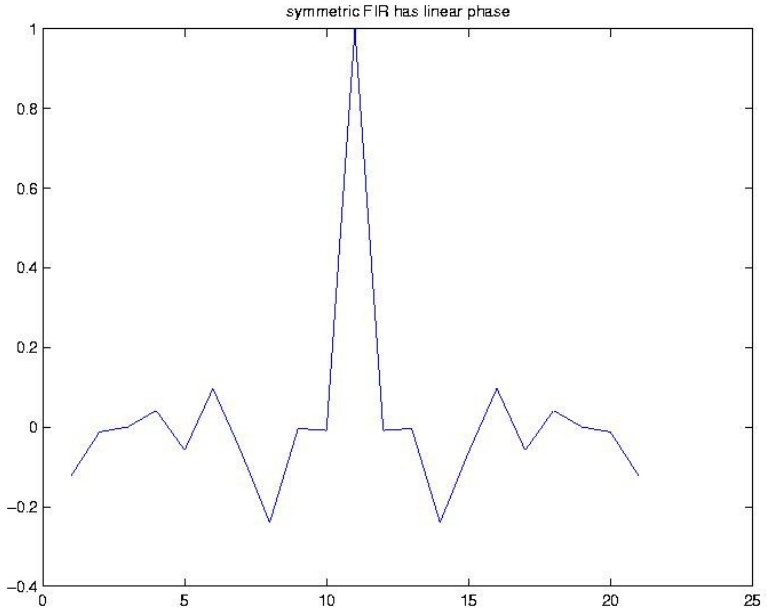
$$x_{n_0}=x[n+n_0] \quad \Rightarrow \quad X_{n_0}\left(e^{j\omega}\right)=e^{j\omega n_0}X\left(e^{j\omega}\right)$$

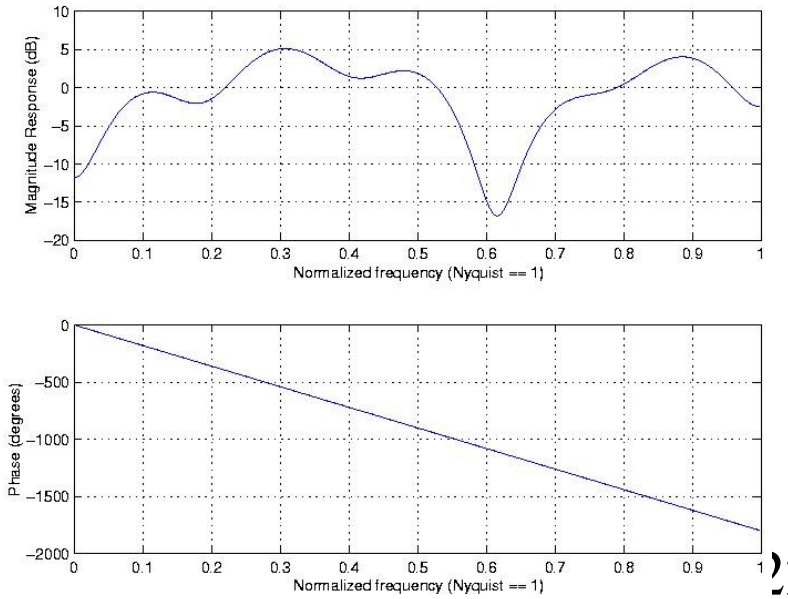
$$Y\left(e^{j\omega}\right)=\left|H\left(e^{j\omega}\right)\right|e^{j\omega n_0}X\left(e^{j\omega}\right)=\left|H\left(e^{j\omega}\right)\right|X_{n_0}\left(e^{j\omega}\right)$$

DTFT - Linear Phase

- A filter with linear phase delays all frequencies by the same amount.
- If we add a constant delay to a zero phase filter we obtain a linear phase filter.
- The shift in time can be removed if the filter can be non-causal. In which case we get a zero phase filter.

Example: delay here 11 samples



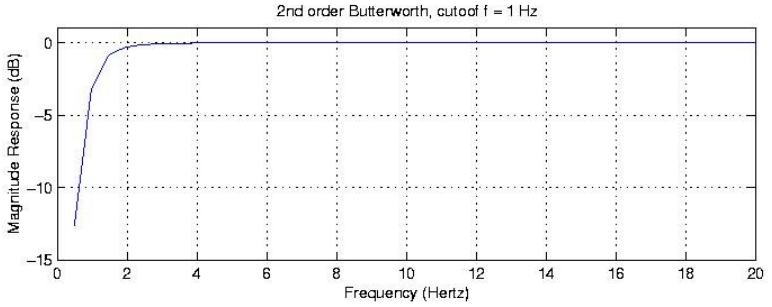


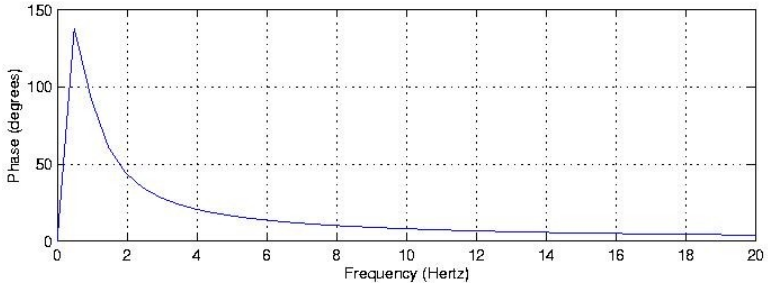
DTFT - Pragmatic filter design example

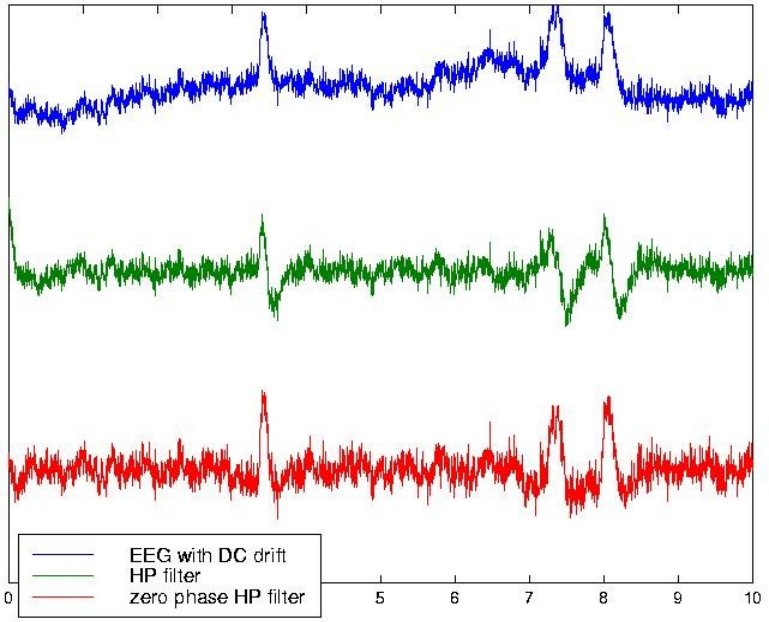
If the filter is allowed to be non causal a **quick fix** is to apply the filter to the time inverted signal resulting in a **zero phase** filter:

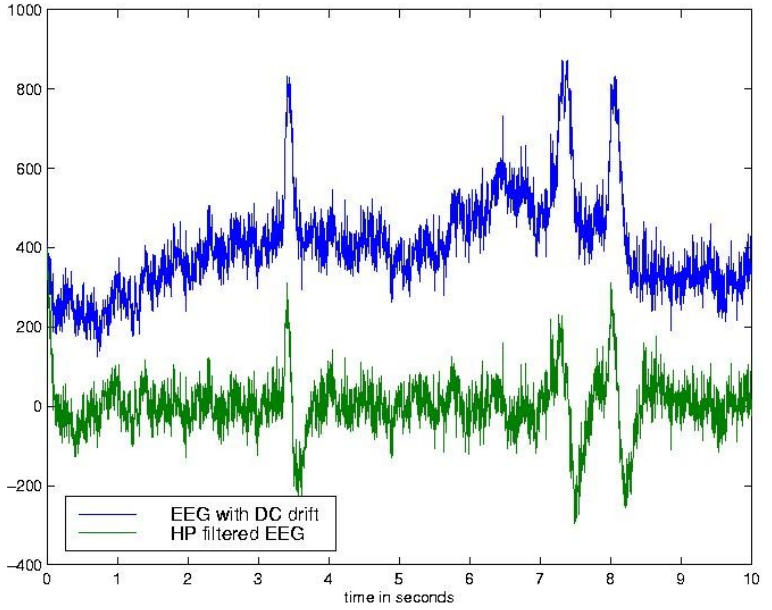
$$Y\left(e^{j\omega}\right)=\left|H\left(e^{j\omega}\right)\right|^2 X\left(e^{j\omega}\right)$$

>> filtfilt(b,a,x)









DTFT - Pragmatic filter design

Assignment 4:

Use `sptool()` to design 10Hz bandpass filter (5Hz bandwidth) to measure the power of alpha activity in EEG.

1. IIR filter of low order AND
2. FIR with linear phase

Show corresponding impulse response, magnitude and phase response. The last two should look like freqz output, but, implement this yourself using fft (abs and angle). Apply this to the EEG signal posted on the web site (EEG_128Hz.txt). Show signal before and after filtering (one channel is enough).

After finding a good filter take note of the parameters, and use the corresponding filter design function to compute the ARMA filter coefficients. To find the corresponding filter function use for instance:

```
>> lookfor chebyshev
```

3. Apply filter to all EEG channels and display power in the 10Hz band across channels. Power is defined as

$$P_y = \frac{1}{T}\sum_{t=1}^{T} |y(t)|^2$$

```
>> Power = mean(abs(y).^2);
```

Gabor Quadrature Pair Filter (Morlet wavelets)

A pair of filters (real and complex values) that has compact support in time and frequency domain and 90° phase delay – also know as Morlet wavelets:

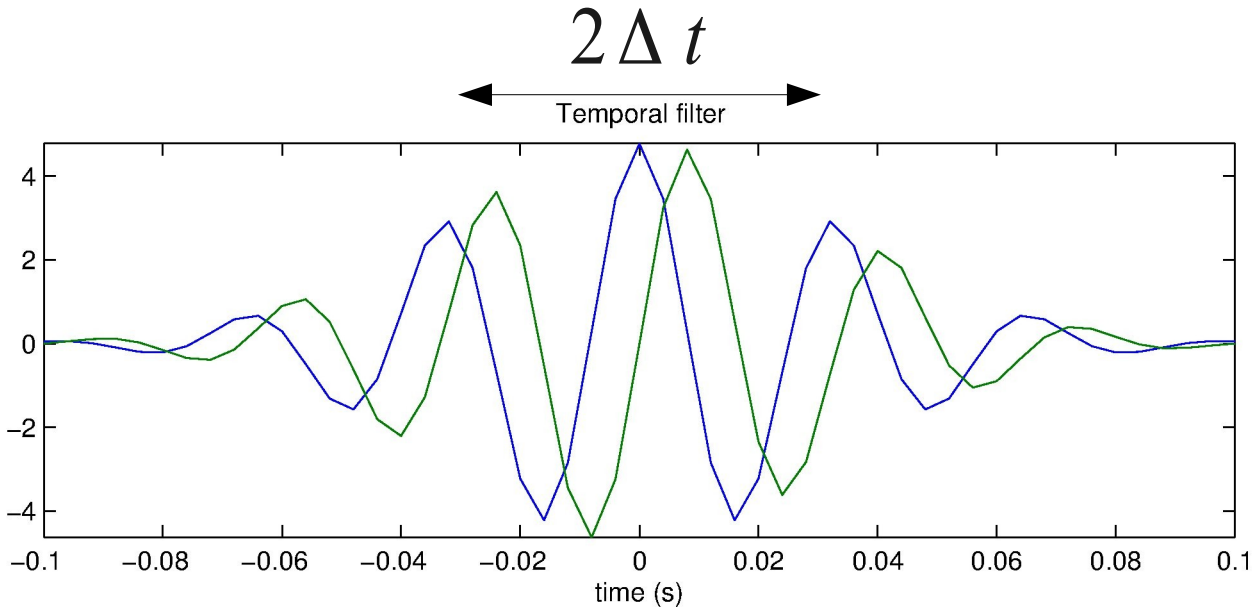
$$h(t) = \exp(j\omega_o t)\exp\left(-\frac{t^2}{2\Delta t^2}\right)$$

$$H(e^{j\omega}) \propto \exp\left(-\frac{(\omega-\omega_o)^2}{2\Delta\omega^2}\right)$$

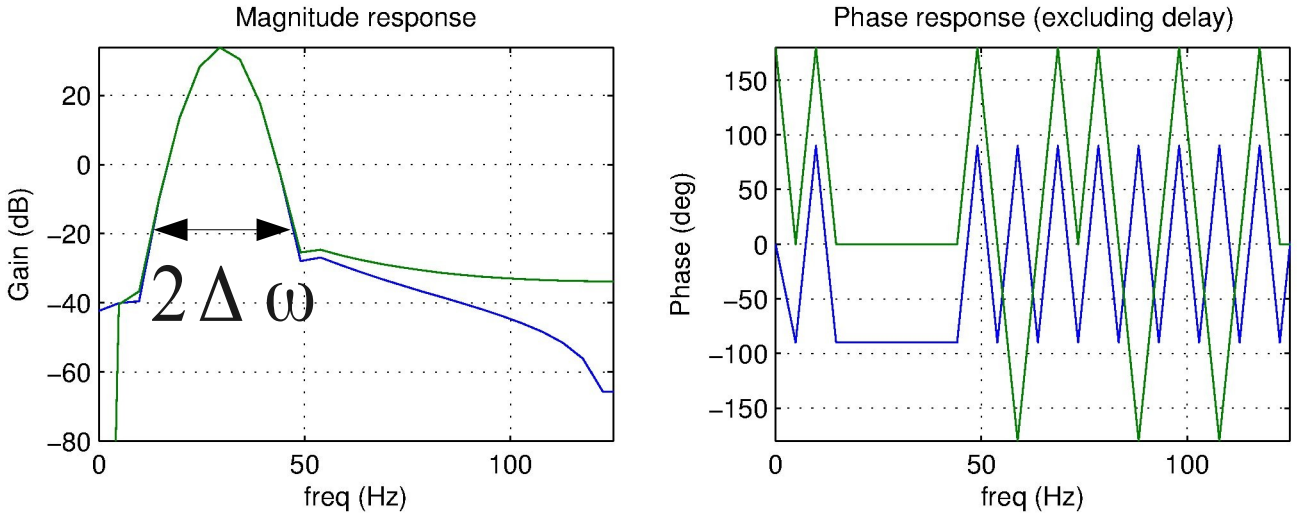
Time-Frequency Product:
(optimal value for this filter)

$$\Delta t \, \Delta \omega = \frac{1}{2}$$

Gabor Quadrature Pair Filter (Morlet wavelets)



$f = 30\text{Hz}$
$f_s = 250\text{Hz}$
$Q = 1$



Q-factor: $Q = \frac{\omega}{\Delta\omega}$ (determines number of oscillations)

Gabor Quadrature Pair Filter (Morlet wavelets)

```
function b=gaborfir(fc,fs,Q)
df = fc/Q; % bandwidth in Hz,
dt = 1/df;
t  = (-3*dt*fs:3*dt*fs)'/fs;
b  = 1/sqrt(pi/2)/fs/dt*exp(-t.^2/2/dt^2).*exp(sqrt(-1)*2*pi*fc*t);

fs = 250; % sampling rate in Hz
fc = 10; % center frequency in Hz
Q  = 1;  % Q-factor = f/df;
b = gaborfir(fc,fs,Q);

subplot(2,1,1);
plot(t,[real(b) imag(b)]); axis tight
title('Temporal filter'); xlabel('time (s)');
fbin = (0:length(b)-1)/length(b)*fs;

subplot(2,2,3);
plot(fbin,db(abs(fft([real(b) imag(b)])))); axis tight
xlim([0 fs/2]); ax=axis; ylim([-80 ax(4)]); grid on
title('Magnitude response'); xlabel('freq (Hz)'); ylabel('Gain (dB)')

subplot(2,2,4);
plot(fbin,180/pi*angle(fft(ifftshift([real(b) imag(b)])))); axis tight
xlim([0 fs/2]); ax=axis; ylim([-180 180]); grid on
title('Phase response (excluding delay)'); xlabel('freq (Hz)');
ylabel('Phase (deg)')
```

Assignment

Alpha power
Create a bandpass filter for alpha activity (~10 Hz). Filter the data and estimate at what times the person has their eyes open and closed (alpha activity is stronger when the eyes are closed).
The output of the homework should be a figure with the following panels.
1) PSD before and after filtering in one figure.
2) Magnitude response of the filter
3) Phase response of the filter
4) Show the instantaneous amplitude and instantaneous frequency.
5) Signal before and after filtering with vertical lines indicating the moments where the eyes are open or closed.
The data is online stored as alpha.mat

Assignment

Spectrogram vs wavelets

1)Use Morlet wavelets to band-pass a signal.
2)Use multiple center frequencies separated by ½ octave
3)Save the the instantaneous amplitude for all frequencies
4)Display the amplitudes as an image (freq, time).
5)Compared that to a spetrogram (as in your previous homework).
6)When doing the spectrogram be sure to multiply with a windowing function prior to the FFT.
7)Display both time-frequency results for the signal in speech.wav

Analytic signal

For a band-passed signal $y(t)$ one can estimate an instantaneous power, phase and frequency from the "**analytic signal**":

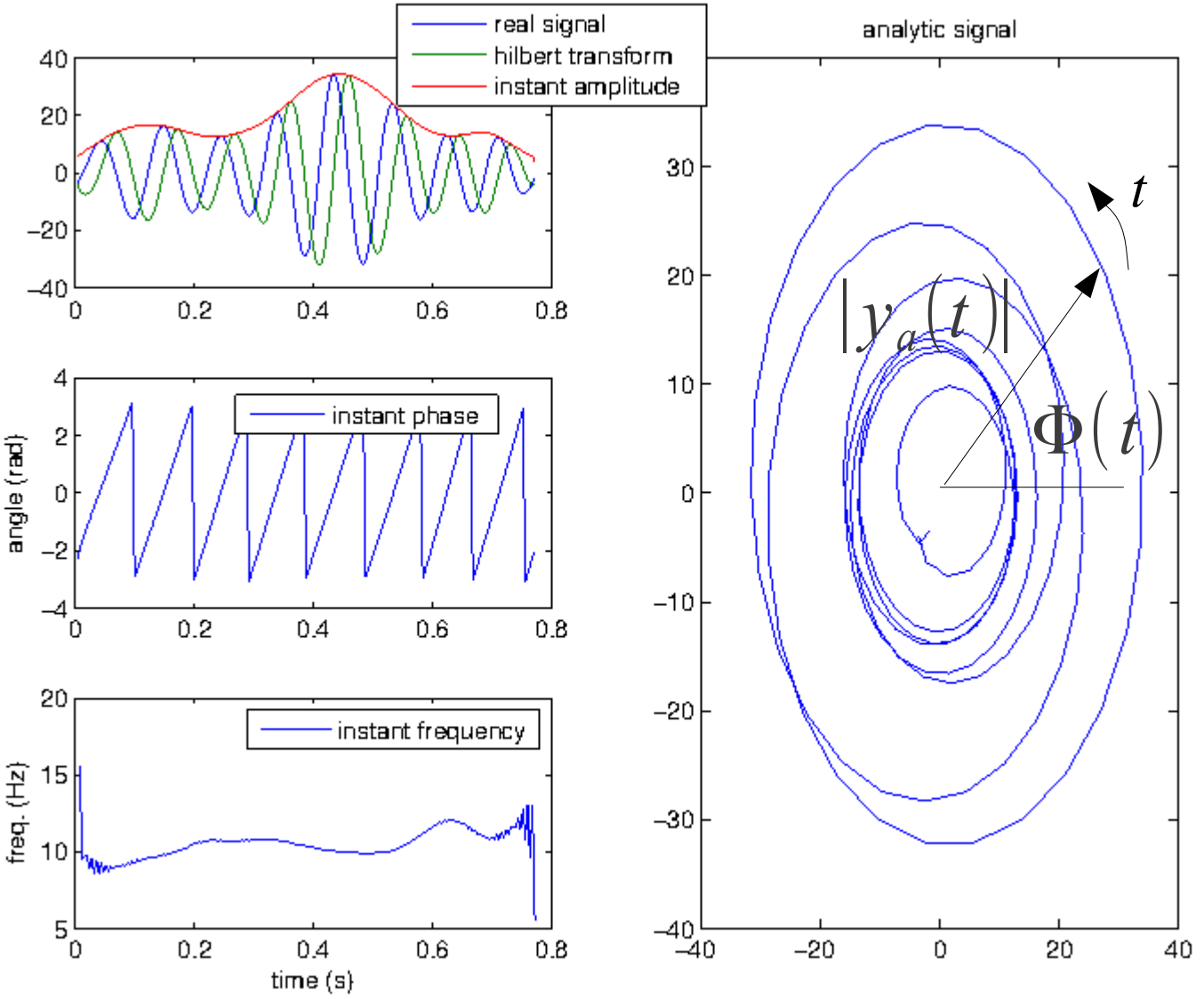
$$y_a(t) = y(t) + i\frac{1}{\pi t} * y(t)$$

The convolution in the second (complex) term is known at the Hilbert transform:

$$H[y(t)] = \frac{1}{\pi t} * y(t) = \frac{1}{\pi}\int_{-\infty}^{\infty} dt' \frac{1}{t} y(t - t')$$

```
>> ya = hilbert(y)
```

Instantaneous power, phase, frequency



Instantaneous power, phase, frequency

Instantaneous power: $P(t)=|y_a(t)|^2$

Instantaneous phase: $\Phi(t)=arg(y_a(t))$

Inst. angular frequency: $\omega(t)=\partial\Phi(t)/\partial t$

Inst. Temporal Freq: $f(t)=\frac{1}{2\pi}\frac{\partial\Phi(t)}{\partial t}$

All this only makes sense for band-pass signals.

Coherence in a frequency band

Coherence is usually defined in the frequency domain. It can be computed from band-pass analytic signals:

$$C_{xy} = \frac{\left| \sum_t x_a^*(t)\, y_a(t) \right|}{\sqrt{\sum_t |x_a(t)|^2 \sum_t |y_a(t)|^2}}$$

where x^* is the complex conjugate of x. The real part captures the instantaneous correlation. The imaginary part captures the correlation of 90° delayed signals. The absolute value here captures correlation at any delay, i.e. the coherence.

In matlab this is simply

```
b  = gaborfir(fc,fs,1);
xa = filter(b,1,x);
ya = filter(b,1,y);
C  = abs(corr(xa,ya));
```

Discrete Fourier Transform (DFT)

The **Discrete Fourier Transform** (DFT) is the **sampled** DTFT

$$X\left(e^{j\omega}\right)_{\omega=2\pi k/N} = X[k]$$

The ***N*-point DFT** is defined for a signal of length N: (analysis)

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$$

with **inverse *N*-point DFT**: (synthesis)

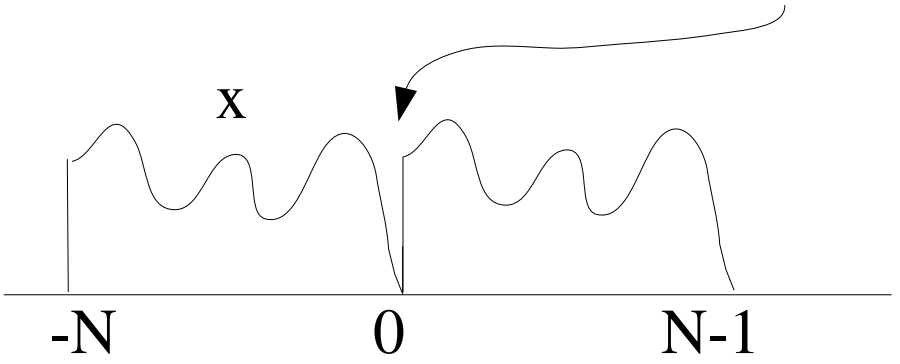
$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N}$$

Note that specifying $X[0]...X[N\text{-}1]$ implies a *synthesized* periodic signal outside $n = 0...N\text{-}1$:

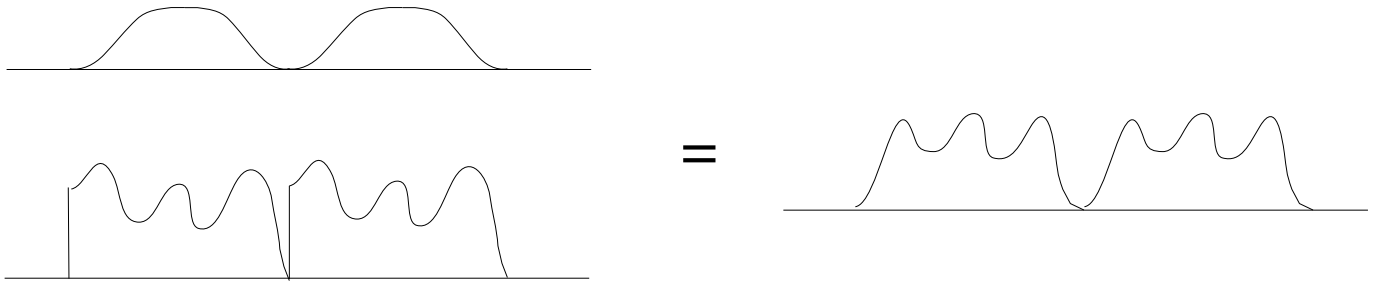
$$x[n] = x[n \bmod N]$$

Windowing

Implicitly, the DFT assumes are periodic signal and thus the analysis and thus the analysis equation will capture a potential discontinuity and the period end.



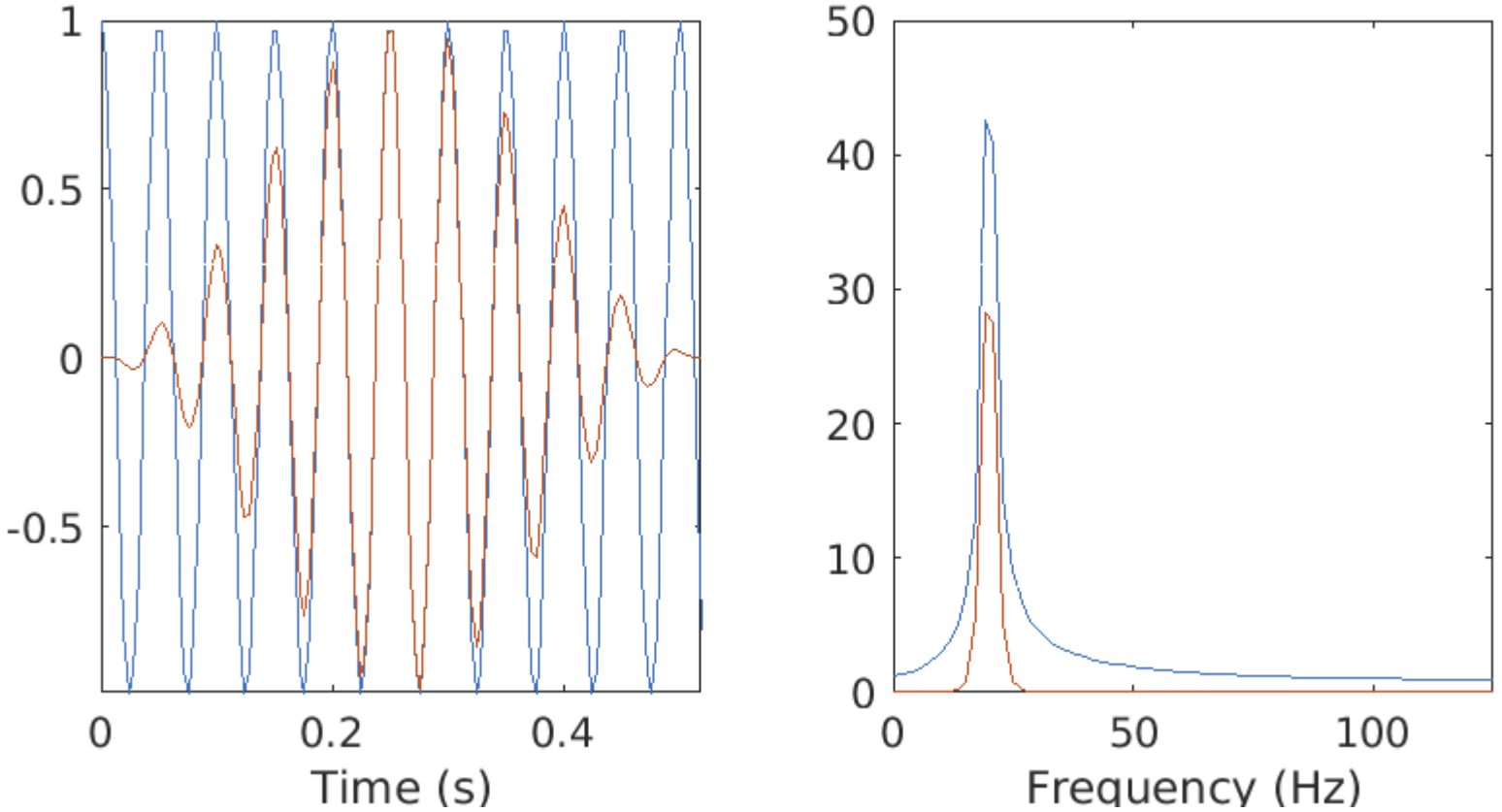
To avoid such artifactual high frequency content one should "window", i.e. multiply the time signal with a windowing function that makes a smooth transition.



Examples are Hanning and Hamming windows.

Effect of windowing in the frequency domain

1)It removes artefactual high-frequency content due to edges (because it removed edges of periodic repetition).
2)It "smooths" the spectrum across frequencies (because product in time domain is that same as convolution in frequency domain).



Discrete Fourier Transform - FFT

In signal processing we always work with the DFT since we can compute Frourier transform only for discrete frequencies.

Important result on computational cost: While computing DFT values $X[k]$, $k=1...N$, would seem to take N^2 operations there is an efficient method called **Fast Fourier Transform** (FFT) of order:

$$N \log_2 N$$

```
>> X =  fft(x);
>> x = ifft(X);
```

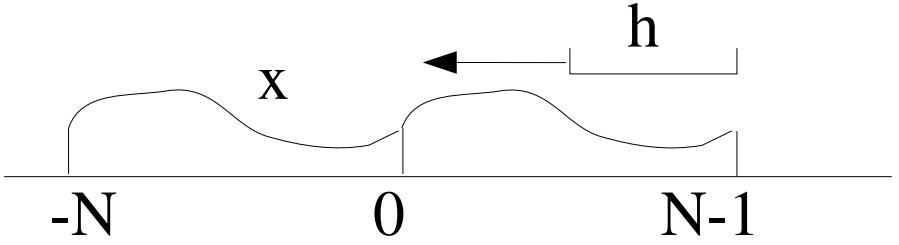
With this one can implement convolution in $\log_2(P)$ operations per sample rather than **P**!

DFT - circular convolution

Because $X[k]$ corresponds to a periodic $x[n]$ with period N the convolution of two signals is equivalent to a **circular convolution**:

$$h[n] \circ x[n] = \sum_{k=0}^{N-1} h[k]\, x[(n-k) \bmod N]$$

That is, the circular convolution "wraps around"



It can be implemented with a circular Toeplitz matrix:

$$\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ \vdots \\ y[N-1] \end{bmatrix} = \begin{bmatrix} h[0] & h[N-1] & h[N-2] & \cdots & h[1] \\ h[1] & h[0] & h[N-1] & \cdots & h[2] \\ h[2] & h[1] & h[0] & \cdots & h[3] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h[N-1] & h[N-2] & h[N-3] & \cdots & h[0] \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix}$$

DFT - circular convolution

The convolution theorem for the DFT corresponds now to a **circular convolution:**

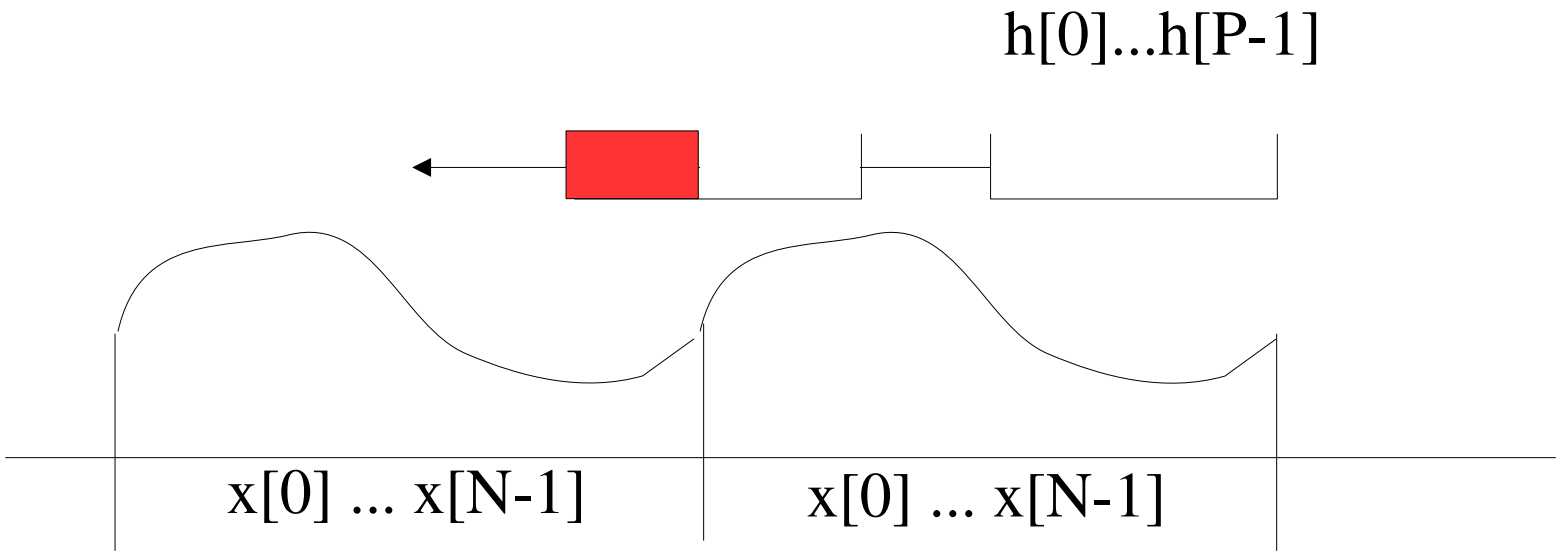
$$y[n] = h[n] \circ x[n] \quad \Leftrightarrow \quad Y[k] = H[k]X[k]$$

We can use this for a fast implement the linear convolution

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

DFT - linear and circular convolution

The N-point circular convolution with filter $h[n]$ of length P corresponds to the linear convolution for the last N-P samples. The first P samples do not correspond to the linear convolution as the filter $h[k]$ multiplies with values in the next period.



Hence, $y[n]$ computed with the product, $H[k]\,X[k]$, of the DFTs corresponds to the linear convolution for $n=N\text{-}P+1 \ldots N$.

DFT - linear convolution with "overlap-save"

The **overlap and save** method for computing the linear convolution using the FFT for a filter of length N consists in:

1. Compute the 2N point FFT of segment $x[n\text{-}N]$... $x[n+N\text{-}1]$
2. Multiply by the 2N point FFT of $h[0]$...$h[N\text{-}1]$
3. Invert the result with IFFT
4. Save the last N samples into $y[n]$...$y[n+N\text{-}1]$
5. Increment n by N and return to 1.

Assignment 5:

- Implement linear convolutions with "overlap and save" method, and apply to random signal x with some arbitrary filter h. Compare this result (y_{os}) with a linear convolution implemented with Toeplitz matrix multiplication (y_{toep}), and `filter()` function (y_{filt}). Plot results y_{os}, y_{toep}, y_{filt}. Should be the same.
- Graph speed of all three methods (using `tic; toc;`) for N=2^n, n=1..10, where N is the size of the filter.

Sampling - Sampling Theorem*

What is the relation between the continuous time Fourier transform (CTFT) of a continuous time signal $x(t)$ given by,

$$X_c(\Omega) = \int_{-\infty}^{\infty} dt\, x(t) e^{-j\Omega t}$$

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} d\Omega\, X_c(\Omega) e^{j\Omega t}$$

and the DTFT, $X(e^{j\omega})$ of the signal sampled at frequency $f_s = 1/T$, $x[n] = x(nT)$?

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-jn\omega}$$

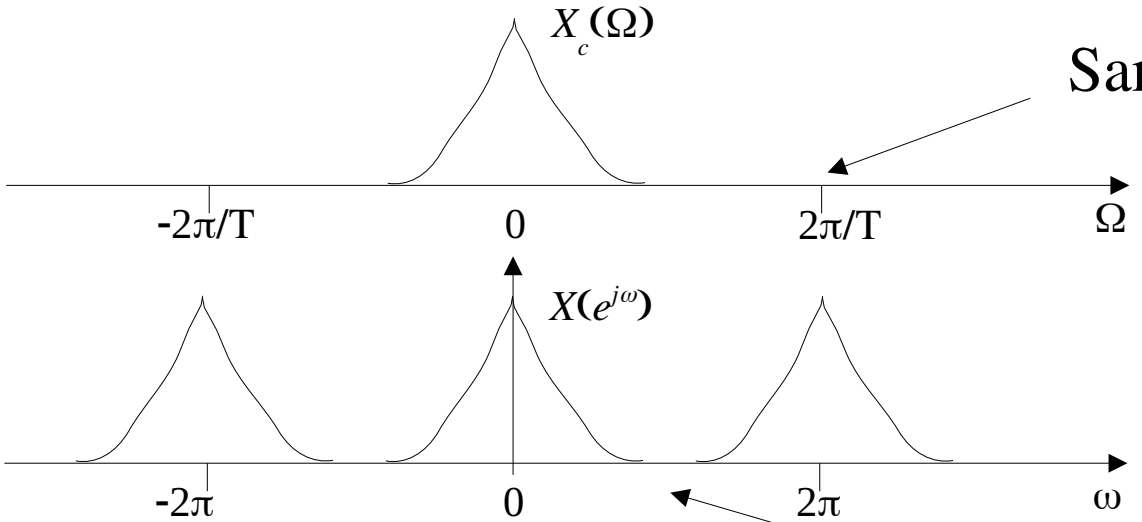
* See Oppenheim and Schaefer for derivation (the *classic* SP book)

Sampling - Sampling Theorem

According to the Sampling Theorem the relation is:

$$X\left(e^{j\omega}\right)=\frac{1}{T}\sum_{k=-\infty}^{\infty}X_c\left(\frac{\omega}{T}+\frac{2\pi k}{T}\right)$$

The DTFT repeats the CTFT with a period 2π.



Contributions above π will overlap with different period!

Sampling - Sampling Theorem

Under which conditions can we determine continuous time $x(t)$ from discrete time $x[n]$, $t=nT$?

If the signal is *bandlimited:* $X_c(\omega/T)=0, \qquad |\omega|\geq\pi$

Then we can determine $X_c(\Omega)$ from $X(e^{j\omega})$ according to the Sampling Theorem:

$$X\left(e^{j\omega}\right)=\frac{1}{T}X_c\left(\frac{\omega}{T}\right)$$

In that case we can determine x[t] $\rightarrow$ $X(e^{j\omega})$ $\rightarrow$ $X_c(\Omega)$ $\rightarrow$ x(t).

After some algebra:

$$x(t)=\sum_{n=-\infty}^{\infty}x[n]\,sinc\left(\frac{t-nT}{T}\right)$$

$x(t)$ is $x[n]$ convolved with $sinc(t)=\sin(\pi t)/(\pi t)$

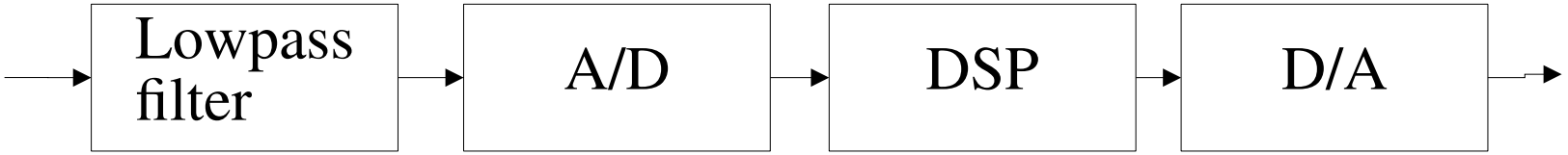
Assignment

Use the sampling theorem to reconstruct a signal with high temporal resolution from its low-frequency sampled version:

1. At a high sampling rate, say 1kHz, generate the following 5 signals signals:
   - Sinusoids at frequencies f=30, 45, and 60 Hz.
   - Gaussian random noise low-pass filtered with stop-band frequency of 40Hz and stop-band frequency of 100Hz.
1. Subsample these 5 signals at a low sampling rate, say 100Hz.
2. Reconstruct the 1kHz signals from the sub-sampled signals.
3. Show the original 1kHz, the sub-sampled and the reconstructed signals.
4. Give the error of you reconstruction as % of variance in the signal for each example (in total you should have 5 numbers giving the % error).
5. For which of the 5 signals above to you expect a vanishing error and for which may there be a significant reconstruction error? Be sure your results agree with that.

Sampling

Because of the sampling theorem always!:



Make sure you lowpass filter the signal to half the sampling frequency (Nyquist) before you sample.

If you can not filter prior to sampling make sure that you choose the sampling frequency to be twice the highest frequency that contains significant signal power.

Do not down sample by simply taking every other sample. First lowpass filter then subsample. Better yet, use either

```
>> x = resample(x,P,Q);
>> x = decimate(x,Q/P);
```