Signals and Filtering: Overview

- Today: signals and data acquisition
  - Source of deterministic errors and random noises
  - Basics of signal spectrums and Fourier Transform
  - Nyquist theorem, modulation and aliasing
  - Basics of low pass filters

- Signal processing is at the interface of embedded software engineering and ECE.
- Objective:
  - understand the fundamental concepts
  - master issues that a software engineer should do and can do

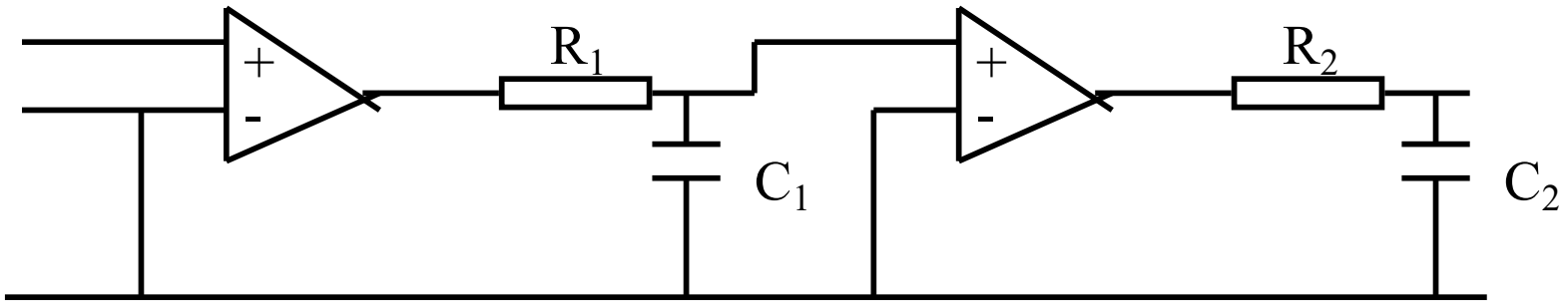
- **A basic tutorial** (written by Hagit Shatkay) on Fourier transform is posted (see lecture2_add_material)

➔ **Do not get confused:**

- **In the lecture ω = angular velocity = 2π * frequency**
- **In the tutorial ω = frequency**

Basic Signal Processing Concepts:
Low Pass filters - 1

Signals are often contaminated by high frequency noises that need to be removed. The following is a simple two stage active analog filter.

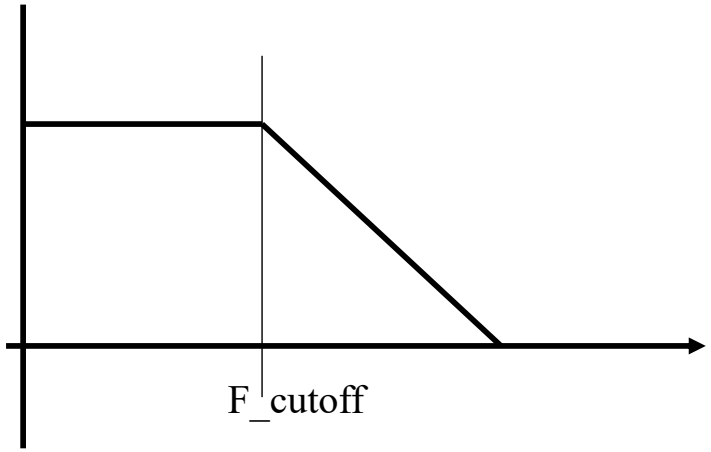


The Op-Am isolates the interactions between two RC circuits.

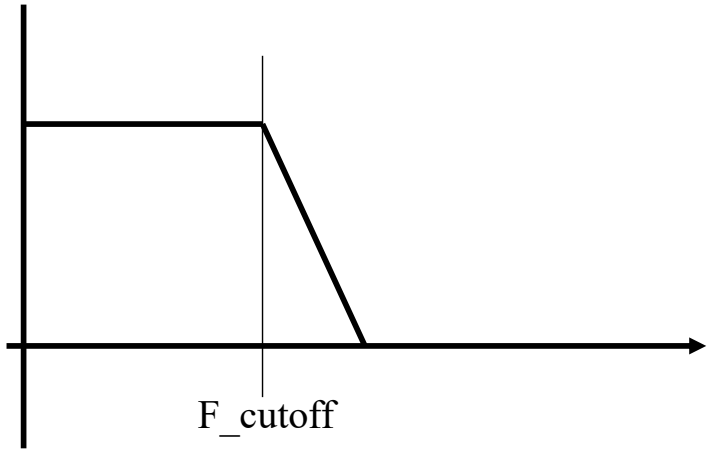
This allows a simple analysis. You can get such filters on a chip

Basic Signal Processing Concepts:
Low Pass filters - 2

- The more are the stages, the sharper is the rate of reduction of the signal after the cut off frequency. If you want sharper rate of reduction, you need more stages. (In filter literature, the number of stages is called the order of the filter.)



1st order



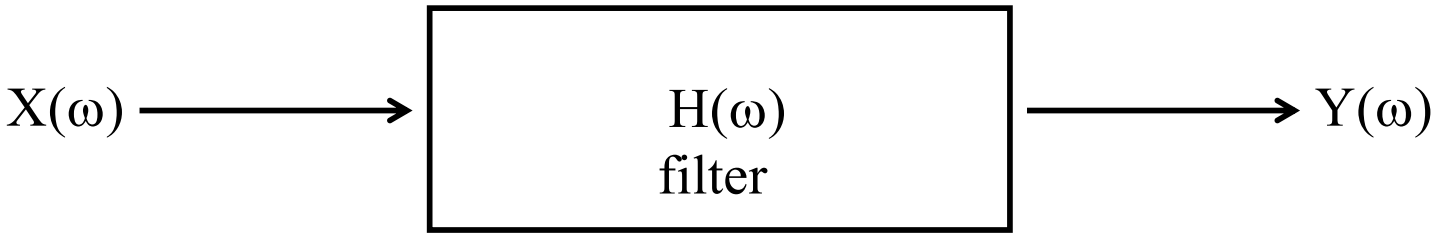
2nd order

Appendix: Frequency Response Function of a filter

- Knowing the frequency response function H(ω) of a filter (it is a complex number!), and the continuous fourier transform X(ω) of the input signal, the continuous fourier transform Y(ω) of the output signal is:

Continuous fourier transfer of y(t)

$$Y(\omega) = H(\omega)X(\omega)$$



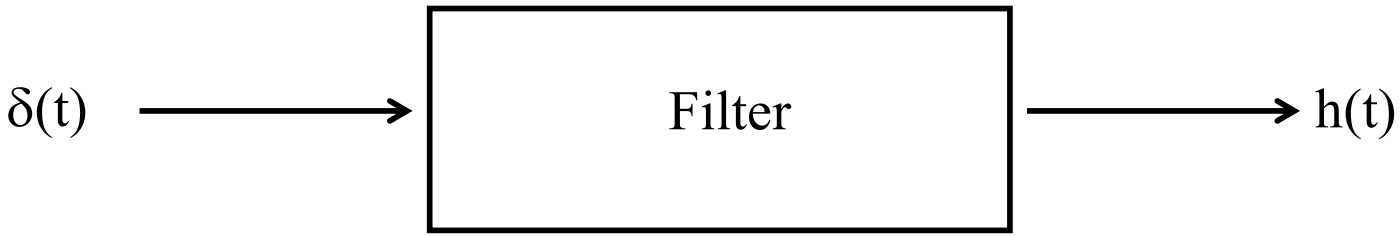
Amplitude response

Phase response

$$H(\omega) = \left|H(\omega)\right| \cdot e^{i\theta(\omega)}$$

Appendix: Frequency Response Function of a filter

- What is the meaning of frequency response function H(ω) of a filter?
- It is the continuous fourier transform of the output signal when the input signal is a delta function δ(t).



Amplitude response

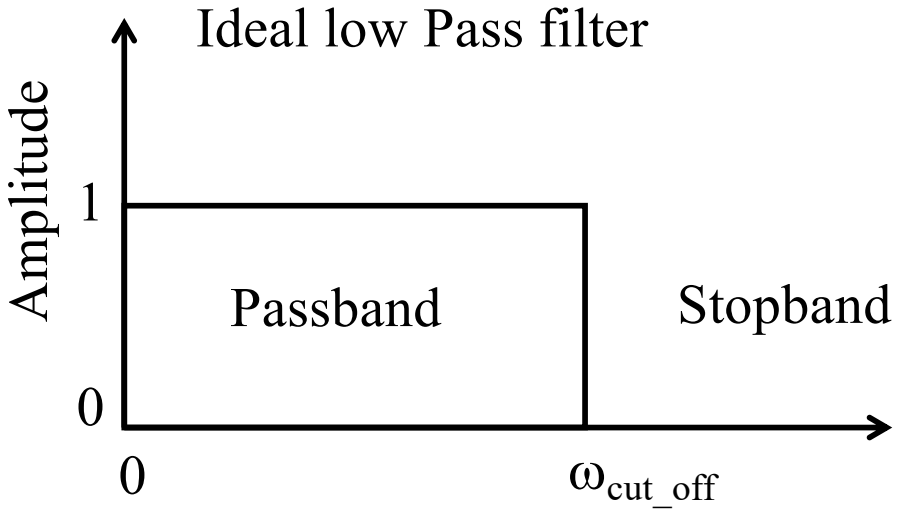
Phase response

$$H(\omega) = \left|H(\omega)\right| \cdot e^{i\theta(\omega)}$$

Appendix: Ideal Low Pass filters - 1

- A filter is a device designed to attenuate specific ranges of frequencies, while allowing others to pass. In this way, a filter can limit the frequency spectrum of a signal.

- **Ideal filter**: The passband amplitude response is continuously flat at a value of 1. The frequencies which are allowed to pass through the filter do so completely undistorted. The stopband amplitude response is continuously flat at a value of 0. The undesiderable frequencies are completely suppressed.

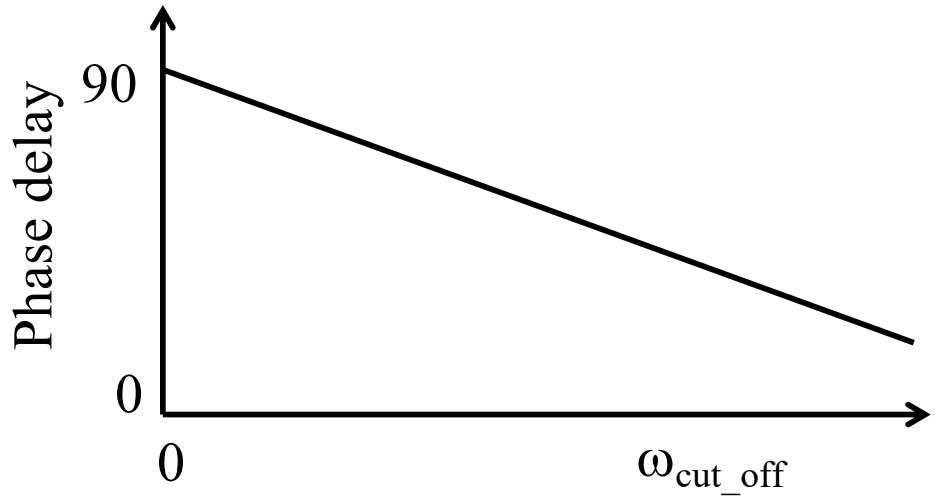


Frequency domain!

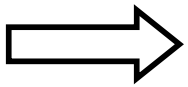
Appendix: Ideal Low Pass filters - 2

- **Ideal Phase Response of a filter**: The brick-wall response of an ideal filter (see previous slide) only describes changes in the magnitude of the signals as a function of frequency. The complete specification of a filter must include its **phase response.** Filters introduce a time delay at the output. **An ideal filter's time delay is independent of the frequency!**

Ideal phase response of low pass filter



Frequency domain!



It applies a time shifting to the filtered signal:

$f(t) \rightarrow f(t - t_0)$

Basic Signal Process Concepts:
Low Pass Filters 3

The magnitude of attenuation and the phase delay of a N stage filter illustrated in page 49 are as follows, where $\omega_{cutoff} = 1/R_1C_1 = 1/R_2C_{2.} = \ldots.$

$$\left|H(\omega)\right| = \left(\frac{1}{\sqrt{1+\left(\frac{\omega}{\omega_{cutoff}}\right)^2}} \right)^N$$

$$\theta(\omega) = -N\left(\arctan \frac{\omega}{\omega_{cutoff}} \right)$$

The formula for the commonly used Butterworth filter is SQRT(1 /(1 + (w/w_cutoff)^{2N}))

Application Notes

- We have two control variables, the cutoff frequency and the order of the filter.

1. The lower the cut-off frequency, the more effective is the filtering. But if the cut-off frequency is too close to the useful signal, the signal will also be reduced.

2. The higher the order, the more powerful is the filter. But it also introduces more phase delay (the shift of the wave to the right on the time-line)

- The design of a filter is iterative. You have to adjust the cut-off frequency and the order of the filter until the design requirements are met.

Class Exercise

- The frequencies of interest in the signal are from 10 to 60 Hz. There are serious noises with frequencies in the range 500 - 1000 Hz seen on the scope. The simple anti-aliasing filter (page 49) is used to filter out the high frequency noises.

- Suppose that we pick cut-off frequency at 100 Hz, what should be the order (stages) of the filter so that the signal will be reduced no more than 30% while the noise will be reduced at least 96%?

$$\left|H(\omega)\right| = \left(\frac{1}{\sqrt{1+\left(\frac{\omega}{\omega_{cutoff}}\right)^2}} \right)^N$$

The frequencies of interest in the signal are from 10 to 60 Hz. There are serious noises with frequencies in the range 500 - 1000 Hz seen on the scope. The simple anti-aliasing filter (page 49) is used to filter out the high frequency noises.

$$\left|H(\omega)\right| = \left(\frac{1}{\sqrt{1+\left(\frac{\omega}{\omega_{cutoff}}\right)^2}} \right)^N$$

Design a Digital Low Pass Filter

- **The general structure of a digital filter is:**
  - y = b(1)*x + b(2)*x(1) + ... + b(n+1)*x(n) - a(2)*y(1) - ... - a(n+1)*y(n), where
  - y is the current (filtered) output that you want to compute
  - x is the current raw input before filter
  - y(k) and x(k) are k-step previous output and input respectively.

- **You can find a's and b's by using Matlab.**

- [b, a] = butter[n, wn] gives a n-th order (n stage) butterworth filter with cut-off frequency wn. wn must be in the form of percentage of the Nyquist frequency, which is sampling rate/2.

- Suppose that sampling rate is 200 Hz and the cut-off frequency is 20, then the Nyquist frequency is 200/2 = 100 Hz and wn = 20/100 = 0.2.

Digital Filter Design - 2

- [b, a] = butter(2, 0.2)
- b = 0.0675 0.1349 0.0675
- a = 1.0000 -1.1430 0.4128
- y = b(1)*x + b(2)*x(1) + ... + b(n+1)*x(n) - a(2)*y(1) - ... - a(n+1)*y(n)
- y = 0.0675x + 0.1349x(1) +0.0675x(2) + 1.143y(1) – 0.4128y(2)

n is the order of the filter

1 step previous input

2 step previous output

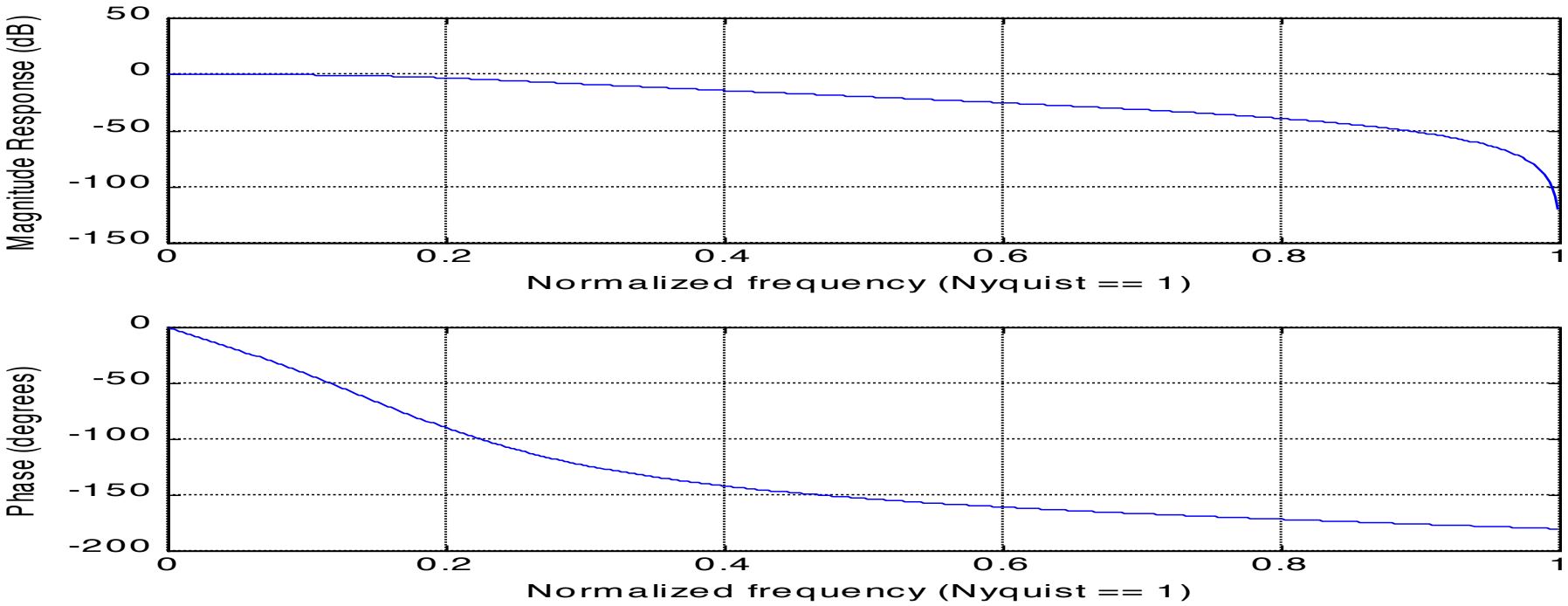
- The above line represents the formula that can be implemented in software to have a digital low pass filter. The formula can also be derived by the help pages of Matlab ➔ See appendix

- **Note**: if your program is sampling the analog signal at 200Hz, then the cut-off frequency of the above filter will be 20Hz. Changing the sampling frequency will affect the cut-off frequency of the digital filter!!!

Digital Filter Design - 3

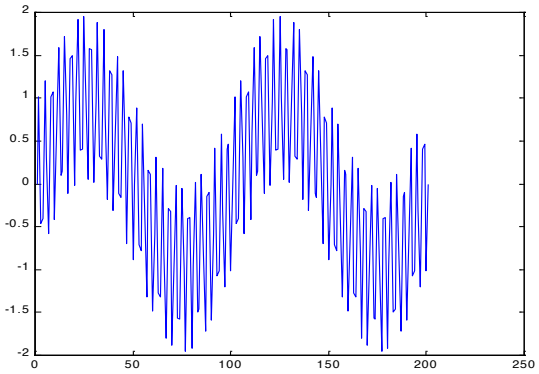
- [b, a] = butter(2, 0.2)
- Freqz(b,a) // Amplitude is expressed in decibel: 20 log (v1/v2), the base of log is 10.

- Examples:
- if the amplitude response of a filter is $|H(\omega)| = 1$ ➔ 20 log(1) dB = 0 dB
- if the amplitude response of a filter is $|H(\omega)| = 100$ ➔ 20 log(100) dB = 40 dB
- if the amplitude response of a filter is $|H(\omega)| = 1/100$ ➔ 20 log(1/100) dB = -40 dB

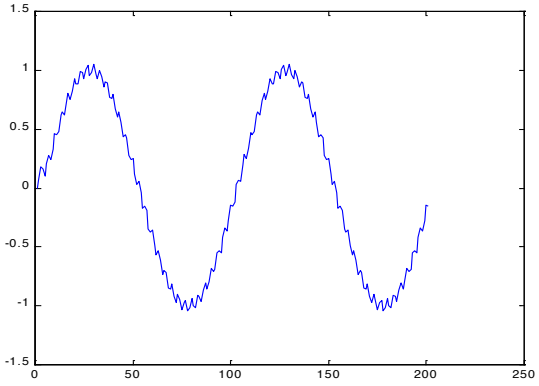


Example: filtering noise from a signal

- t = 0:0.005:2; %sample at 200 Hz for 2 sec.
- x = sin(2*pi*t); % 1 Hz signal
- y = sin(2*pi*30*t); % 30 Hz noise
- plot(x+y)



- wn = 20/(200*0.5)=0.2 % cut off at 20 Hz
- Nyquist
- [b, a] = butter(2, 0.2) % 2nd order filter
  % cutoff 20 Hz
- z = filter(b, a, (x+y));
- plot(z)



- "%" comment. ";" suppress output.

Appendix: Digital Filter Design

- [b, a] = butter(2, 0.2)
- b = 0.0675 0.1349 0.0675
- a = 1.0000 -1.1430 0.4128
- y = 0.0675x + 0.1349x(1) +0.0675x(2) + 1.143y(1) – 0.4128y(2)
- y = b(1)*x + b(2)*x(1) + ... + b(n+1)*x(n) - a(2)*y(1) - ... - a(n+1)*y(n)
- Try Matlab's help and search "**butter**" and select "**Filter Design Issues**"

Digital Domain

`[b,a] = butter(n,Wn)` designs an order `n` lowpass digital Butterworth filter with cutoff frequency `Wn`. It returns the filter coefficients in length `n+1` row vectors `b` and `a`, with coefficients in descending powers of `z`.

$$H(z) = \frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + \cdots + b(n+1)z^{-n}}{1 + a(2)z^{-1} + \cdots + a(n+1)z^{-n}} \quad \Longrightarrow \quad H = \frac{Y}{X}$$

$$Y\left[1 + a(2)z^{-1} + a(3)z^{-2}\right] = X\left[b(1) + b(2)z^{-1} + b(3)z^{-2}\right]$$

- Apply the following transformation: Z^{-1} Y = y(1)…… Z^{-N} Y = y(n)
- Apply the following transformation: Z^{-1} X = x(1)…… Z^{-N} X = x(n)

Appendix: Digital Filter Design

- Apply the following transformation: Z^{-1} Y = y(1)…… Z^{-N} Y = y(n)
- Apply the following transformation: Z^{-1} X = x(1)…… Z^{-N} X = x(n)

$$Y\left[1 + a(2)z^{-1} + a(3)z^{-2}\right] = X\left[b(1) + b(2)z^{-1} + b(3)z^{-2}\right]$$

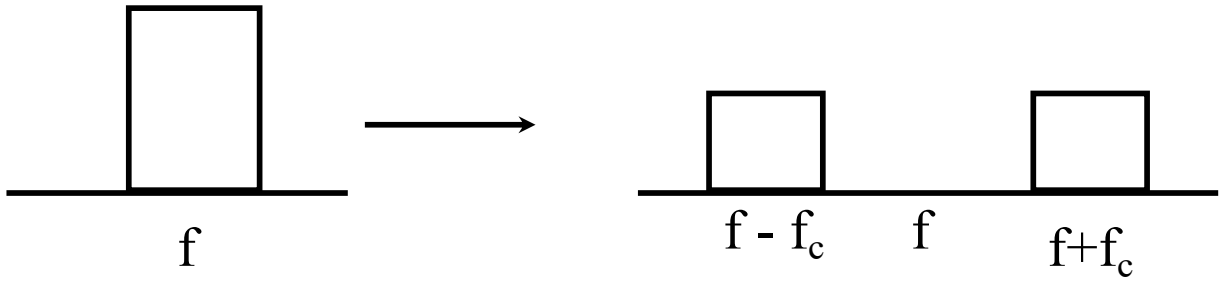
$$y + a(2)y(1) + a(3)y(2) = b(1)x + b(2)x(1) + b(3)x(2)$$

$$y = b(1)x + b(2)x(1) + b(3)x(2) - a(2)y(1) - a(3)y(2)$$

Appendix: Modulation Theorem

Multiplying a signal $x(t)$ by a sin function with frequency f_c splits the signal into two parts and frequency shifted by f_c and $-f_c$

$$x_c(t) = x(t)\sin\omega_c t \longrightarrow |X_c(f)| = 1/2\,[(X(f - f_c) + X(f + f_c)]$$



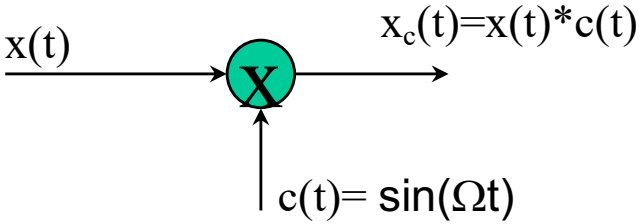
- **Quiz: how can we re-obtain the original signal after we apply a modulation?**

Appendix: a better understanding of Modulation Theorem

- The process of multiplying a signal x(t) by a sin/cos function is called **modulation**

$$\mathcal{F}[y_1(t)y_2(t)] = \underbrace{\frac{1}{2\pi}\int_{-\infty}^{\infty} Y_1(\omega-\Omega)Y_2(\Omega)d\Omega}_{\text{Convolution of } Y_1(\omega) \text{ and } Y_2(\omega)} = \frac{1}{2\pi}\int_{-\infty}^{+\infty} Y_1(\Omega)Y_2(\omega-\Omega)d\Omega$$

Appendix: a better understanding of Modulation Theorem



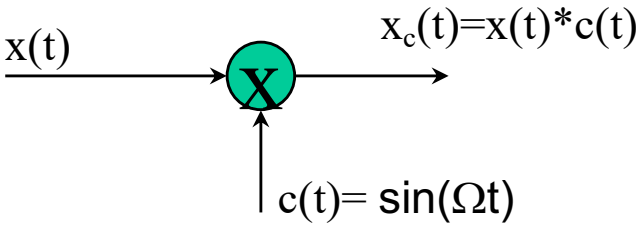
$$F[y_1(t)y_2(t)] = \frac{1}{2\pi}\int_{-\infty}^{+\infty} Y_1(\alpha)Y_2(\omega-\alpha)d\alpha$$

$$\mathcal{F}[\sin(\Omega t)] = \frac{\pi}{i}\delta(\omega-\Omega) - \frac{\pi}{i}\delta(\omega+\Omega)$$

$$F[x(t)\cdot c(t)] = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X(\alpha)\left[\frac{\pi}{i}\delta(\omega-\Omega-\alpha) - \frac{\pi}{i}\delta(\omega+\Omega-\alpha)\right]d\alpha =$$

$$= \frac{1}{2i}\left(\int_{-\infty}^{+\infty} X(\alpha)\delta(\alpha-\omega+\Omega)d\alpha - \int_{-\infty}^{+\infty} X(\alpha)\delta(\alpha-\omega-\Omega)d\alpha\right) =$$

Appendix: a better understanding of Modulation Theorem



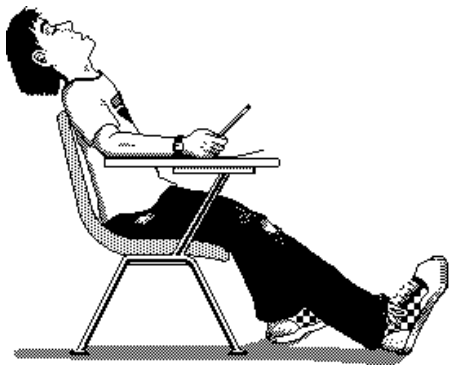
$$F[y_1(t)y_2(t)] = \frac{1}{2\pi}\int_{-\infty}^{+\infty} Y_1(\alpha)Y_2(\omega-\alpha)d\alpha$$

$$\mathcal{F}[\sin(\Omega t)] = \frac{\pi}{i}\delta(\omega-\Omega) - \frac{\pi}{i}\delta(\omega+\Omega)$$

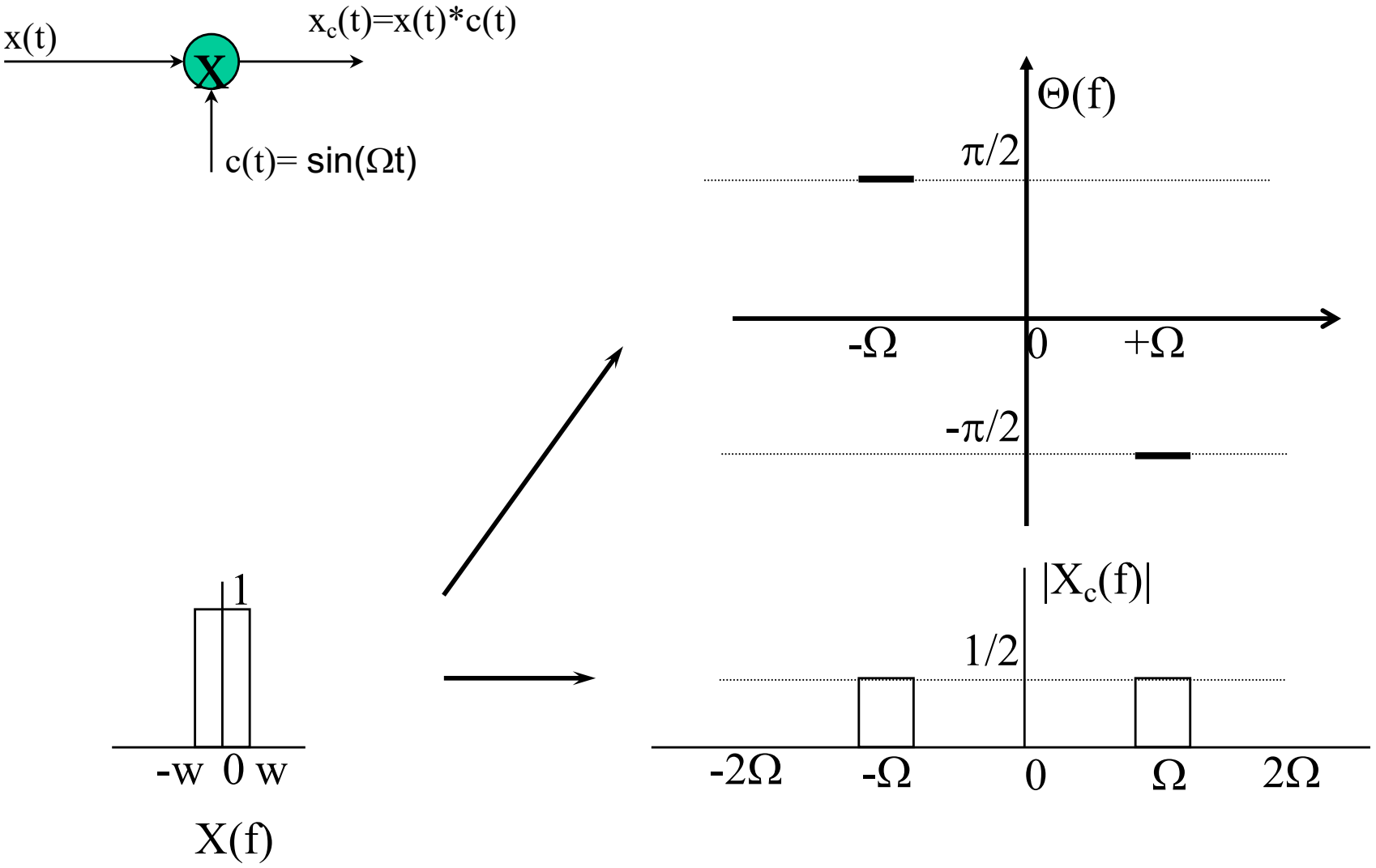
$$F[x(t)\cdot c(t)] = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X(\alpha)\left[\frac{\pi}{i}\delta(\omega-\Omega-\alpha) - \frac{\pi}{i}\delta(\omega+\Omega-\alpha)\right]d\alpha =$$

$$= \frac{1}{2i}\left(\int_{-\infty}^{+\infty} X(\alpha)\delta(\alpha-\omega+\Omega)d\alpha - \int_{-\infty}^{+\infty} X(\alpha)\delta(\alpha-\omega-\Omega)d\alpha\right) =$$

$$= \frac{1}{2i}\left[X(\omega-\Omega) - X(\omega+\Omega)\right]$$



Appendix: a better understanding of Modulation Theorem



Example: amplitude modulation/demodulation

- % Example of amplitude modulation and demodulation with Matlab:

- % digital sampling: 500Hz
- % s(t) 10Hz sine
- % c(t) 100Hz AM cosine carrier
- % sam(t) modulated signal
- % s_dem(t) demodulated signal
- % s_dem_f(t) filtered and demodulated signal

- t = 0:0.002:1;
- s = .1 + .1*sin(2*pi*10*t);
- c = cos(2*pi*100*t);

- sam = times(s,c); % modulating signal s

- s_dem = times(sam,c); % demodulating signal with AM modulation

Example: amplitude modulation/demodulation

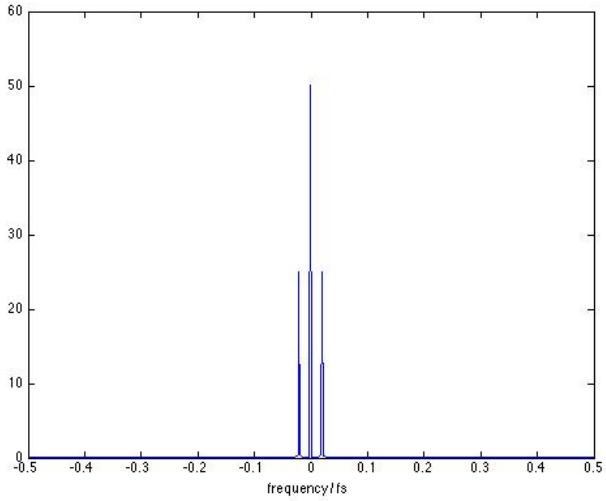
- % plot fast fourier transform for the demodulated signal s_dem
- N = 501
- F = [-N/2:N/2-1]/N;
- FTs_dem = abs(fft(s_dem));
- FTs_dem = fftshift(FTs_dem);
- plot(F,FTs_dem), xlabel('frequency / fs')

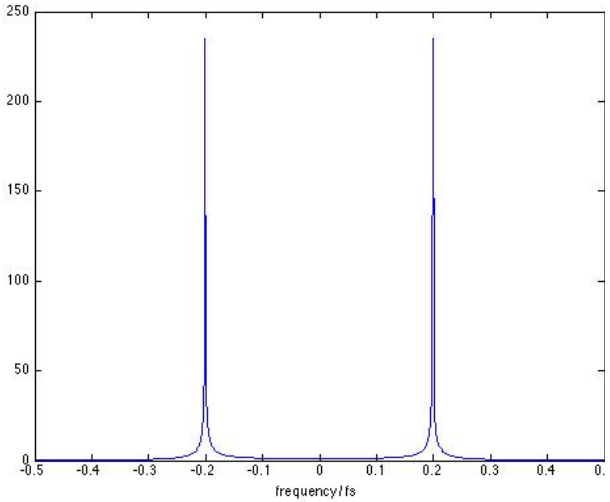
- % designing a Butterworth digital filter
- [b, a] = butter(4, 0.08); % filter: cutoff at 20Hz, 4th order

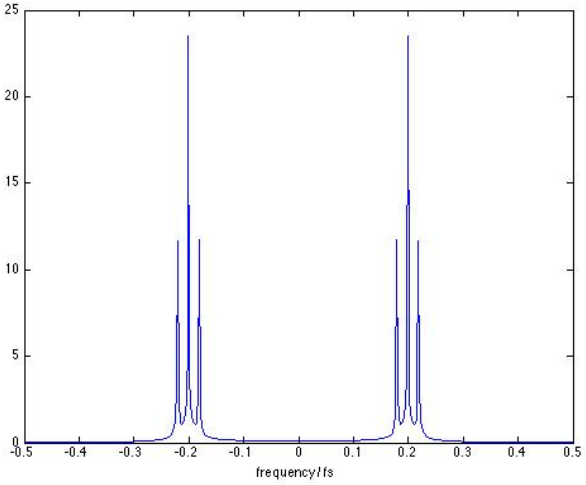
- % filtering the demodulated signal
- s_dem_f = filter(b, a, s_dem);

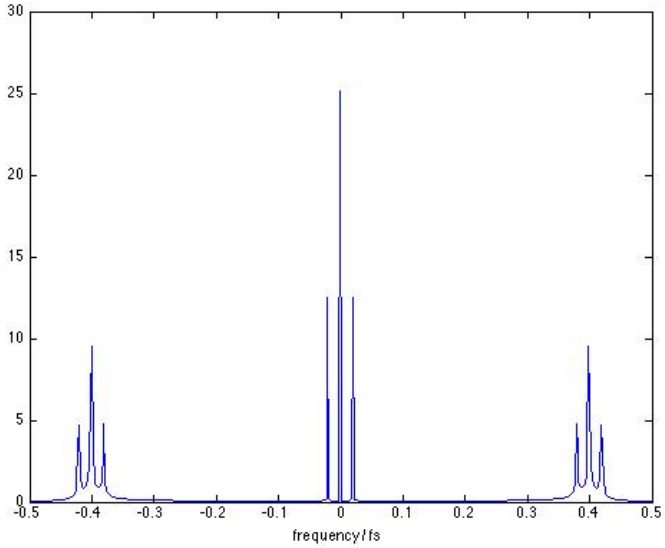
- % plotting original, demodulated, and demodulated_filtered signals
- plot(t, s, t, s_dem, t, s_dem_f); figure(gcf);

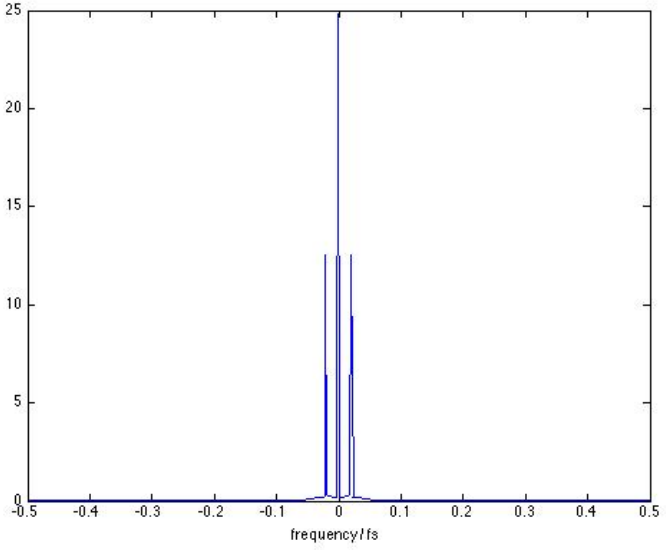
Example: amplitude modulation/demodulation











Example: amplitude modulation/demodulation

- Blue signal: ?
- Green signal: ?
- Red signal: ?