HW Assignment 7

Due date: Thursday 12/5/2016

In this assignment you will use a MATLAB add-on called SIMULINK in order to learn about amplitude modulated signals. If this is your first encounter with MATLAB or SIMULINK some of the terms I use below may be unfamiliar. The internet will help.

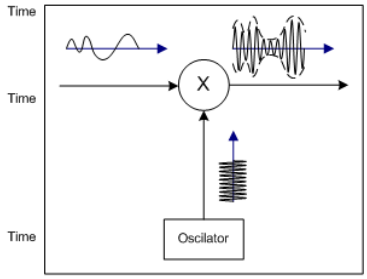
NOTE: The software was tested on MATLAB versions starting from 2014a. Most aquarium computers should run it.

Background - Amplitude Modulation

In order to better pass through a channel a steady radio signal or "radio carrier" must be changed or modulated in one way or another. In amplitude modulation, the amplitude (signal strength) of the carrier wave is varied in proportion to the waveform being transmitted.

Two of the most significant reasons to modulate the signal would be:

- Whether the original signal's frequencies decay fast in the air.
- we want to create lots of different channels without a major overlap between them.

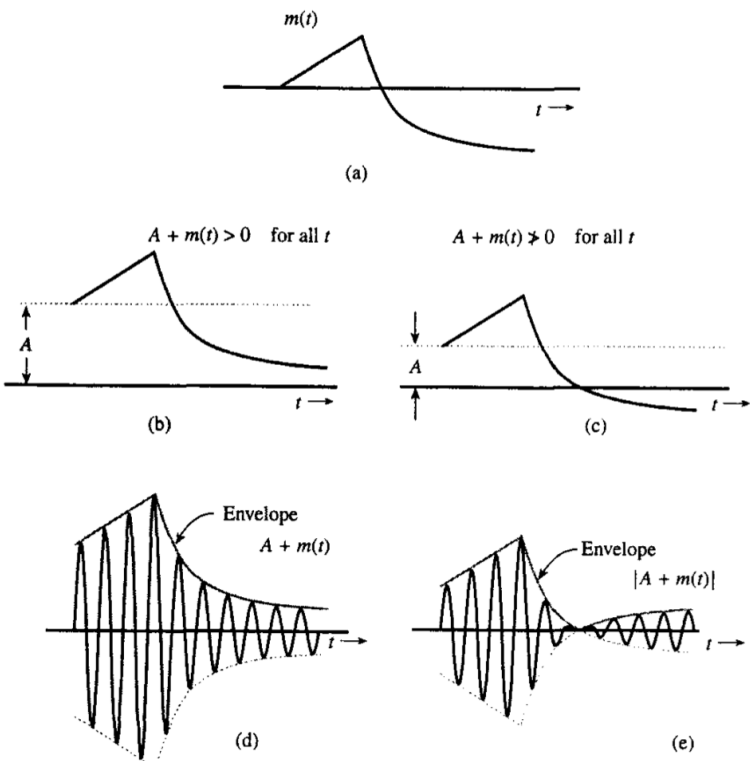


The Transmitter

- <u>In the time domain:</u>

We multiply the amplitude of the original signal with a local oscillator ω_0 (a cosine function $\cos(\omega_0 t)$).

That gives us a modulated signal with an envelope according to the original signal. If the original signal is fractional negative, we shall add a DC signal (just some constant) to the original signal so that the original signal+DC signal is now positive.

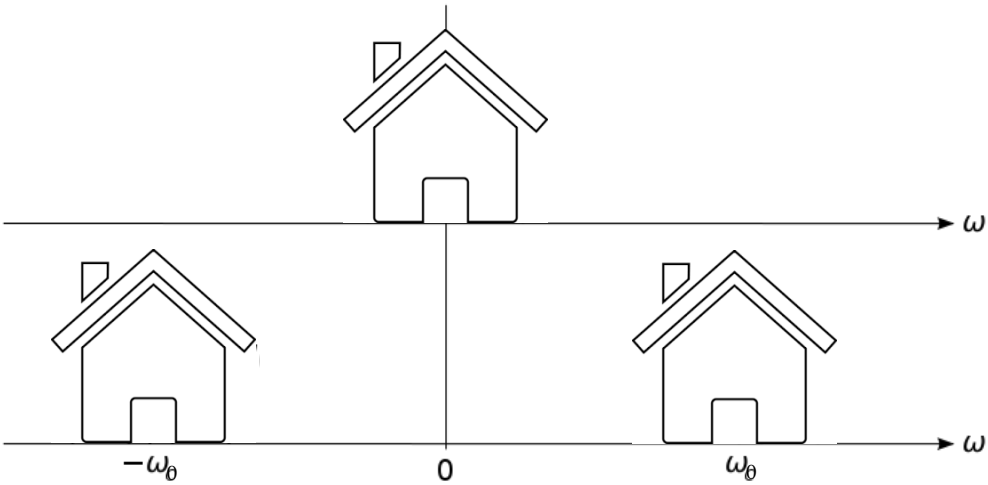


- <u>In frequency domain</u>

The Fourier transform of the original signal $f(t)$ multiplied by the local oscillator $\cos(\omega_0 t)$:

$$F_{\text{mod}}(\omega) = Fourier\{f(t)\cos(\omega_0 t)\} = \frac{1}{2}F(\omega-\omega_0) + \frac{1}{2}F(\omega+\omega_0)$$

So for a low frequencies signal centred at $\omega = 0$, for instance, we get now two copies of the original spectrum, centred at $\omega = \pm\omega_0$.



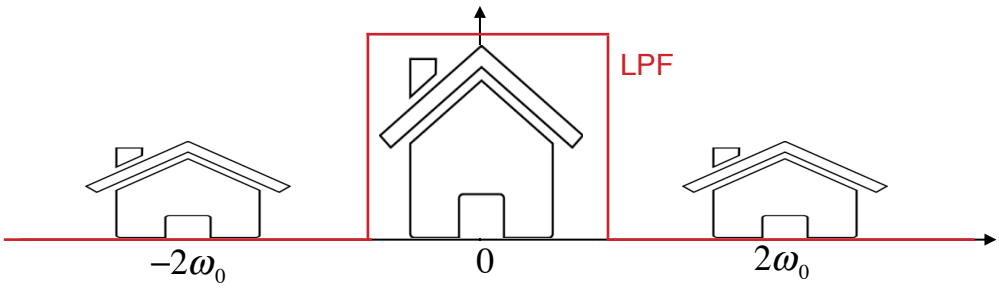
The Receiver

In this exercise We will examine two signal reconstruction ways:

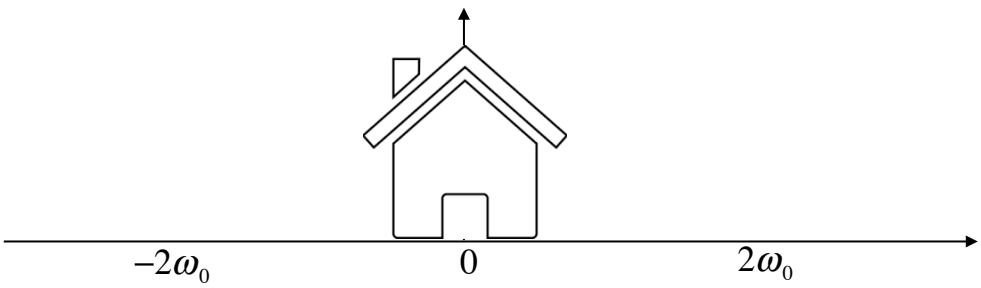
- <u>AM Demodulator</u>

The received modulated signal is multiplied again by the local oscillator $\cos(\omega_0 t)$ (with the same frequency ω_0), so that the signal spectrum now will be:

$$F_{\text{mod}}(\omega) * \frac{1}{2}\left[\delta(\omega-\omega_0)+\delta(\omega+\omega_0)\right] = \frac{1}{4}F(\omega-2\omega_0) + \frac{1}{2}F(\omega) + \frac{1}{4}F(\omega+2\omega_0)$$

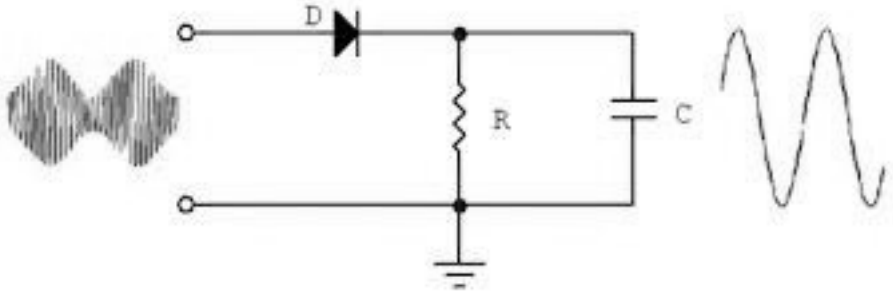


Applying a LPF on the signal now will give us the original signal's spectrum (the actual signal can be generated using inverse Fourier transform).

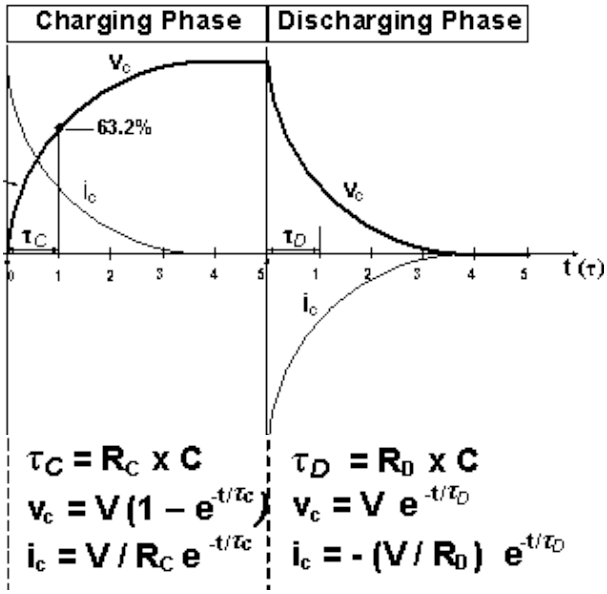


- <u>Envelope Follower</u>

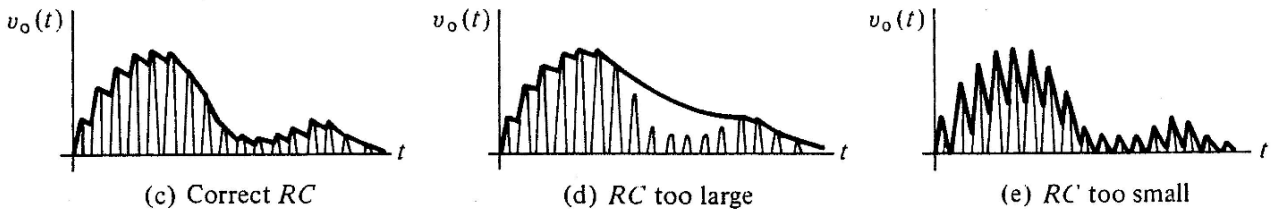
Can be implemented simply as a charged and discharged capacitor (a simple RC circuit).



The product of resistance R and capacitance C is called the Time Constant τ, which characterizes the rate of charging and discharging of a capacitor.



The time constant τ should be large enough so that the circuit output does not decay completely between every two peeks, but small enough so that the circuit output can follow the rapid changes in the signal.



Preliminaries

- Download the required files from the moodle and place them in a directory.
- Open MATLAB and change it's working directory to the location of the files.
- In the command window, type *open AMChannel2* (loading takes a bit of time - have a coffee…).

Simulink is an environment for 'real-time' simulation of complex systems. When you press the *PLAY* button, the simulation starts. If your computer is not powerful enough, the simulation may occur (a) in slow motion (b) in leaps, or (c) both.

- Press *PLAY,* wait a few seconds (after the simulation started) and press stop.
- Switch ti the matlab window. You wiki see in the *workspace* that several variables were created - one for each of the magnet boxes in the simulation (e.g *EnvFollower,RecieverInput*).
- You can plot the signal that reaches the box *AMDemod* by typing

  plot(squeeze(AMDemod.Data))

- Changing the parameters of a component is done by double-clicking the component. **Change the frequency of the *Demodulator*, and rerun the simulation. Plot the output of the demodulator again and see what happens[1], explain.**

In our setup, an input signal is modulated by a sine wave in order to better pass through a channel (in orange). After the channel there are 2 different receivers. we will test them in different configurations. When you press *PLAY*, you should hear the received signal (might be in slow motion if your computer cannot achieve real-time simulation).

Part 1 - Channel Identification

First, we want to estimate the frequency response of the channel $H(\omega)$. into the *MULTIMEDIA FILE* block, load the file *SILENCE.WAV*, which contains the zero signal $x(t)=0$ (double-click on the *INPUT SIGNAL* block and change the file name). The *MODULATOR* block produces a sine wave. For this experiment you can shorten the simulation duration (at the top) to 1 second.

1. For this input signal, write the output of the transmitter, and input of the receiver (plot it on the same graph). you can use the following code:

   plot(squeeze(TransmitterOutput.Data));

   hold on;

   plot(squeeze(ReceiverInput.Data),'r');

2. Run several simulations at different *MODULATOR* frequencies. For each frequency, run the simulation 5 times. After each run, record the maximum absolute value of *recieverInput* and *transmitterOutput* (use the command *max()*). For the 5 runs at each frequency, calculate the means of the values you recorded. Plot an estimate of $|H(\omega)|$ (hint: remember that $H(\omega)={Y(\omega)}/{X(\omega)}$). **Use the command *loglog() instead of plot.***

[1] Use the internet to learn how to display two plots in the same window with different colours.

Part 2 - AM Demodulator

Set the frequencies of the MODULATOR and the DEMODULATOR to $6 \cdot 10^4 \, {}^{rad}\!/_{\sec}$. Make sure that the *AUDIO-HPF* in the *AM DEMODULATOR* is bypassed (by double-clicking the switch).

3. Run several simulations with different values of the phase of the DEMODULATOR. for each run, plot *InputSig* and *AMDemon* on the same graph. What phase gives the best reconstructed signal? Explain the results.

Reload *AudioSegment.WAV* into the *MULTIMEDIA FILE* block. Set the simulation length to 1-3 seconds, and make sure to use the AUDIO-HPF (by clicking the switch). Consider the following error measure between the input signal $x(t)$ and the receiver signal $y(t)$

$$\varepsilon_{x,y} = 1 - \frac{\int x(t)y(y)dt}{\sqrt{\int x^2(t)dt}\sqrt{\int y^2(t)dt}}$$

where we shall approximate the integrals by summing over all samples. For example, to get the numerator integral use:

*sum(squeeze(Signal1.Data)).*Squeeze(Singnal2.Data));*

4. For different values of the *phase* of the demodulator, calculate the error ε between the input and the output. Plot the error as a function of the phase.

   NOTE: Use the given function *normXY(x,y)* in order to normalize *InputSig* and *AMDemon* before multiplying them.

5. Listen to the output of the *AM DEMODULATOR* at different values of the phase of the *DEMODULATOR.* is the sound quality consistent with the results of the previous item?

   NOTE: It is best to listen off-line (after the simulation has ended) by using the command:

   soundsc(squeeze(AMDemod.Data),1e5);

Part 3 - Envelope Follower

We are now interested in the *ENVELOPE FOLLOWER*

6. With the *AUDIO-HPF2* block bypass, plot a graph of the *RECIEVERINPUT* and the output of the *ENVFOLLOWER* (on the same axis with different colours).
7. Explain in your own words the operation of the envelope follower.
8. The main component in the envelope follower is the block named *RF-LPF2,* which is a low-pass filter with a single pole (the RC we've talked about earlier). Plot a graph of the error ε as a function of the pole of the RF-LPF (make sure to use the AUDIO-HPF2 by clicking the switch). Explain the results.

Part 4 - Conclusions

9. Which receiver performs better under ideal conditions?
10. In which cases would you prefer the other receiver?