Experiment 3

Amplitude Modulation

MATLAB Simulation

Objectives

The main objectives of this experiment are:

1) To gain a clearer understanding about double-side band suppressed carrier (DSB-SC) and amplitude modulation (AM)

2) To learn how to simulate modulation/demodulation systems for DSB-SC and AM using MATLAB for synthetic & real signals (such as speech).

Pre-Lab Work

1) Read the relevant material in your textbook (Chapter 4).

2) Using MATLAB, perform the following:

   a. X=[0,1,2,3,4,5]

   b. Y=[1,2,3,4,5,6]

   Now multiply X by Y using two ways. The first one is the usual MATLAB multiplication (star (X*Y)) and the other one is what is called as point-wise array multiplication (a dot followed by a star(X.*Y)). What is the difference between the two?

3) Using MATLAB, generate a vector t = [0:0.001:1]. Then generate m =cos(2*pi*t) and v =cos(4*pi*t). Plot m, v, and the product x=mv. Are you going to use multiplication between matrices or vectors that are representing functions?

Introduction

Amplitude modulation (AM) is the family of modulation schemes in which the amplitude of a sinusoidal carrier is changed as a function of the modulating message signal. This type of modulation schemes includes many variants, such as double-sideband suppressed carrier (DSB-SC), single-sideband (SSB), conventional AM, and vestigial-sideband (VSB). Refer to

your textbooks for ample details on amplitude modulation techniques. In this lab, we focus in particular on DSB-SC and conventional AM.

DSB-SC AM:

In DSB-SC AM, the amplitude of the modulated signal is proportional to the message signal. The time-domain representation of this scheme is given by:

$$y(t) = A_c x(t)\cos(2\pi f_c t)$$

where $A_c \cos(2\pi f_c t)$ is the carrier signal with a carrier frequency f_c, and $x(t)$ is the message signal. The transmission bandwidth is twice the bandwidth of the message signal.

Conventional AM:

AM is similar to DSB-SC, but it also includes a pure carrier (non-modulated) component in the transmitted signal. The message signal $x(t)$ is replaced by $[1+\mu x_n(t)]$, where $x_n(t)$ is the normalized message signal and μ is the modulation index. Therefore the AM signal will be:

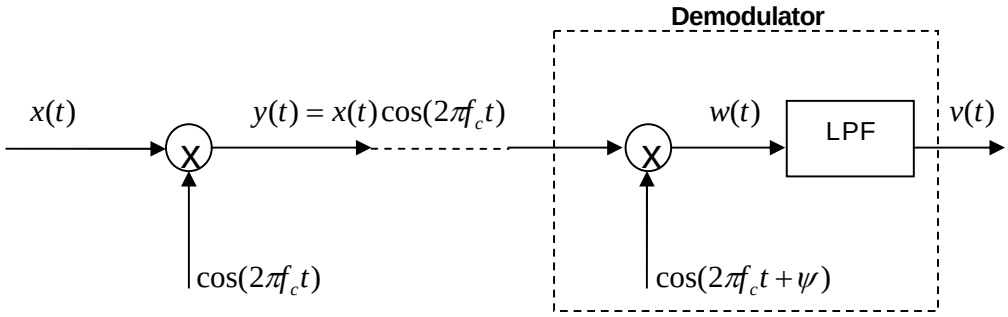
$$y(t) = A_c[1+\mu x_n(t)]\cos(2\pi f_c t)$$

The existence of the sinusoidal component makes the AM scheme less economical in terms of power utilization as compared to the DSB-SC scheme. However, the demodulation of AM signals is much cheaper than the demodulation process of DSB-SC signals. The conventional AM demodulation process is simply done by employing envelope detectors.

For the bandwidth, the AM signal has the same transmission bandwidth as the DSB-SC transmission bandwidth.

Lab Work

1) Use MATLAB to simulate the following block diagram



Assume $\psi = 0$ and let $x(t) = \cos(2\pi 2000t)$. Use a carrier frequency of $f_c = 20$ kHz. Plot $x(t)$, $y(t)$, $w(t)$, and $v(t)$, and their magnitude spectrums each

in a two-panel figure. Define the time vector t as [0:200]*ts where ts is the step size given by $t_s = 1/(10 f_c)$.

At the receiver end, you need to design a Low Pass Filter (LPF). In MATLAB, you can use a type of filters known as Butterworth filters. For example, you can design a given filter with some order n and cut-off frequency f_c which is typically normalized in Matlab and given by $2 f_c t_s$ (where t_s is the sampling period). To obtain the filter coefficients, the statement will be: [num,den]=butter(n, 2 $f_c t_s$), where **num** and **den** are the numerator and denominator coefficients of the rational function representing the analog filter. You can use $n = 5$ for example. Once you obtained these coefficients, you can use the Matlab function **filter** to filter the signal $w(t)$ using the designed LPF. That is, v = filter(num,den,w).

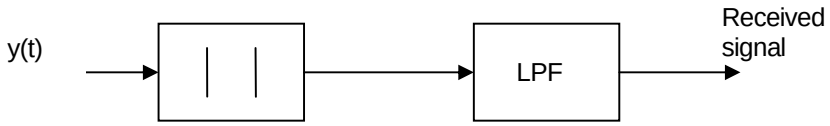
Refer to the additional notes below for further discussion on how to use filters in Matlab.

Useful MATLAB Functions: cos, fftshift(fft( )), butter, filter, abs, plot, subplot, figure, xlabel, ylabel, title.

2) Repeat Part 1 with2 different values for the receiver phase offset: $\psi = \pi/2 \,\&\, \pi$. What do you notice at the receiver end? Is there any difference between the recovered signal here and the one obtained in Part 1? Why is that? And what is the solution to this problem?

Repeat Part 1 by making $y(t) = A_c(1 + \mu x_n(t))\cos(2\pi f_c t)$ where μ is the modulation index of the AM wave, A_c is the carrier amplitude (set it equal to 4), and $x_n(t)$ is the normalized version of $x(t)$. Set it to be 0.5 (50% modulation).

3) At the demodulator, you can implement the functionality of the simple envelope detector that you studied in class (built with a diode, a capacitor and resistor) by using simple MATLAB code to produce full-wave rectification (absolute value function), followed by low-pass filtering. This is illustrated in the following block diagram. You need to think about setting the appropriate cutoff frequency for the LPF. In addition, you can also add a mechanism to remove the DC component from the signal



4) Repeat Part 3 by letting the modulation index equal to 1.2. What will happen to the received signal? Explain.

5) Load the file called Exp3Part5.mat. This data file contains:

   a. A vector called ***ms***, which is a speech signal sampled with $t_s = 1/96\text{E}3$ s.

   b. A vector called ***t*** that represents time.

With a carrier of 24kHz, transmit and receive ms using the AM system in part 3 with $\mu = 0.5$ & 1.5. For both cases show the following:

   a. In one figure with two panels, the time and frequency domain representations of the modulated waves

   b. Listen to **ms** by typing (**sound(ms,96E3), pause**, then press **Enter** to continue). Also listen to the received signal **v** by typing **sound(v,96E3)**. Comment on the differences between the two signals.

Note: in order to be able to see the spectrum of the signal **ms**, after plotting the magnitude spectrum of **ms**, (denoted by **Ms**) vs. **f**, type:

axis([-4E3,4E3,0,max(|Ms|)])

Useful MATLAB Functions: load, sound, pause, axis.

Additional Notes (Filters in MATLAB):

To further understand how to use filters in Matlab, recall from EE207 that the rational transfer function of a filter can be expressed in terms of the Laplace variable *s* by:

$$H(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0}$$

where the a's and b's define the transfer function coefficients. These coefficients completely characterize the filter response. Matlab returns these two vectors as a result of designing a filter with a certain type (e.g., Butterworth, etc) and cutoff frequency. For example, with num=$[b_m, b_{m-1}, \ldots, b_0]$ and den=$[a_n, a_{n-1}, \ldots, a_0]$, the filter design is done by : [num,den]=butter(n, 2 $f_c t_s$),

Notice also that the filter order *n* is important to specify. For example, t is shown that for these Butterworth-type filters, as the filter order increases the filter response will approach that of an ideal "brick-wall" response.