ELEN E3084: Signals and Systems Lab
Lab III: Amplitude Modulation

1 Introduction

As one of the simplest modulation schemes, amplitude modulation (AM) has found many applications both in broadcasting (AM radio and TV video broadcasting) as well as in point-to-point communications (vintage walkie-talkies). The purpose of this lab is to expose us to the practical side of this theory by presenting some simplified cases of real-life information transmission scenarios.

In the first part of the lab we are going to use elementary signals in order to recreate results similar to the ones that appear in section 4.7 Application to Communications: Amplitude Modulation of the Lathi course textbook. By doing so it will be evident how Matlab can help us built our intuition and provide guidance in solving theoretical problems. For this part only basic Matlab commands will be used.

In the second part of the lab we are going to simulate the transmission of voice over a realistic, non-ideal conditions. This part will be fun as the results of our work will be audible and the distortion will be analyzable both quantitatively as well as empirically by using our own ears.

At the end we should feel confident in both writing simple Matlab code that implements fundamental modulation functions as well as using existing packages that provide functionality that would take months for us to program.

Before we begin we will start again the diary function which will capture all our commands. Go the `lab3` directory using the `cd` command the same way as in UNIX. Then type:

```
>> diary on;
```

Note that you should reproduce all the examples that will be presented below. By receiving your diary file at the end of the lab session we will have a simple way to see that you successfully completed the lab.

2 Amplitude modulation of elementary signals

In this first part of the lab we will focus on a couple of simple examples as presented in section 4.7-1/p277 of the Lathi book. Assuming that one has already studied the theory presented in that section the execution of this part of the lab is going to help us solidify our knowledge.

Before we proceed with the actual modulation commands in Matlab we will first need to write two helper functions. The purpose of those functions is to provide functionality that will be needed more than once. By organizing our code in such a modular way we gain in the long term since we only have to make corrections in one place.

To Do 1

Write two Matlab functions. Save them under the `/lab3` directory using the filenames `am_spectrum.m` and `am_plot.m` and submit them. The first function computes the magnitude of the fourier spectrum as needed for visualization.

MATLAB 1.1

```
function X = am_spectrum(x)
% AM_SPECTRUM Calculates the spectrum of a signal
%   X = am_spectrum(x)
%
%   x: signal
%   X: spectrum of signal x

X = abs(fftshift(fft(x)));
```

The second function plots any three signals that are given to it using subplots. This is very useful for comparing the modulator, the carrier and the modulated signals. Notice how our comments are available if you get help for the function.

MATLAB 1.2

```
function am_plot(idx,m,c,u,rng)
% AM_PLOT Plots the three modulation signals
%   am_plot(m,c,u,rng)
%
%   idx: x index (it can represent time or frequency)
%   m:   modulating signal
%   c:   carrier signal
%   u:   modulated signal
%   rng: range of x axis to plot (optional)

if nargin < 5; rng = 1; end % default value for rng

figure; % create a new figure so we don't overwrite an existing one
subplot(311); % split the figure in three subplots
plot(idx,m); grid on;
axis([rng*min(idx) rng*max(idx) min(m)-0.1 max(m)+0.1]);
```

```
subplot(312);
plot(idx,c); grid on;
axis([rng*min(idx) rng*max(idx) min(c)-0.1 max(c)+0.1]);
subplot(313);
plot(idx,u); grid on;
axis([rng*min(idx) rng*max(idx) min(u)-0.1 max(u)+0.1]);
```

How does the `rng` parameter affect the range of the x axis ?

Now we will construct elementary signals and we will modulate them. Using the functions we already wrote, it is going to be easy to visualize them both in time as well as in frequency. The plots we will create will be very similar to the figures 4.31/p278 and 4.32/279 of the book. At the end we will have an intuitive understanding of the effect of each parameter involved.

To Do 2

Make elementary modulators, carriers and modulated signals and plot them. Submit the plots in `.eps` format.

MATLAB 2.1

```
% Define the sampling frequency in Hz
fs  = 2000;
len = 1; % length of signals in seconds
t   = (-len/2):1/fs:(len/2); % time index
% Make our first modulating (message) signal m(t)
fm  = 0.5; % modulator frequency (Hz)
m   = cos(2*pi*fm*t) - 0.25;
% Now make the carrier c(t)
fc = 20; % carrier frequency
c  = cos(2*pi*fc*t);
% Modulate the signal
u  = m.*c;
% Let's plot them in time to see how they look like
am_plot(t,m,c,u,1.1);
% Calculate the spectra
M = am_spectrum(m);
C = am_spectrum(c);
U = am_spectrum(u);
% Make the frequency index for plotting
f = (-fs/2):(1/len):(fs/2);
```

```
% Now let's plot them in frequency
am_plot(f,M,C,U,0.1);
```

Change the carrier frequency to 40 and repeat the same procedure.
Can you verify the carrier frequency from the plots ?

Now we will create a second message signal and a second carrier.
You should repeat the plotting procedure of Matlab 2.1. Submit the plots in `.eps` format.

MATLAB 2.2

```
% Now make the second message signal
pw = 0.1; % pulse width
m  = rectpuls(t,pw);
% Now make a new carrier
fc = 60; % carrier frequency
c  = cos(2*pi*fc*t);
% Modulate the signal
u  = m.*c;
```

How does the spectrum of the pulse change when we vary the pulse-width parameter (`pw`)? (Try values like `0.01, 0.2, 0.5` and `0.75`)
Can you identify the upper and the lower sidebands (USB, LSB) in the spectrum of the modulated signal?

Now change the parameters to `pw = 0.05` and `fc = 15`.
Can we have perfect demodulation in this case? (Justify your answer)
Submit the plots in `.eps` format.

Note that up to this point we didn't consider the demodulation and reconstruction of the modulating signal. This process requires filtering and/or envelope detection which we will examine through the functionality of the Communications Toolbox in the next section.

3 Amplitude modulation of speech

In this section we are going to simulate modulation and demodulation of a speech signal using Double Sideband, Transmitted Carrier (DSB-TC, or just AM). The theory as described on sections 4.7-1,2,3 of the book applies here. We are going to use two functions that encapsulate

the modulation and demodulation functionality. The functions `mymod.m` and `mydemod.m` are going to be provided by the LA.

To Do 3

Evaluate acoustically the quality of the demodulated speech as this is effected by different carrier frequencies.

MATLAB 3.1

```
% Load the modulating signal (message)
[m,fs] = wavread('sm1_cln_32k.wav');
% Keep only a small part of it to speed up calculations
m = m(1:42000);
% Listen to it 'cottage cheese'
soundsc(m,fs);
% Look at the time signal
t = linspace(0,length(m)/fs,length(m));
figure; plot(t,m); grid on; axis tight;
% Look at the spectrum
f = linspace(-fs/2,fs/2,length(m));
figure; plot(f,am_spectrum(m)); grid on; axis tight;
```

Listen to the original speech signal; the utterance is 'cottage cheese'. Notice in the plot of the spectrum that that the bandwidth of the signal is approximately 4.5 kHz. Now lets modulate our signal.

```
% Define a carrier frequency
fc = 8000;
% Modulate the signal
u  = mymod(m,fc,fs);
% And demodulate
dm = mydemod(u,fc,fs);
```

Using the following commands notice how the modulated signal's spectrum is shifted to $\pm$ 8kHz, our carrier frequency. There is no overlap on the LSB so perfect demodulation is possible. We can even listen to the modulated signal:

```
% Look at the modulated signal in the time domain
figure;
plot(t,u); grid on; axis tight;
% Look at the spectrum of the modulated signal
figure;
plot(f,am_spectrum(u)); grid on; axis tight;
```

```
% Now listen to it (everything is shifted to high frequencies)
soundsc(u,fs);
```

Note that the modulating signal `m` looks similar to the modulated one `u` in the time domain plot. This is because the temporal envelope of the modulating signal is preserved. Try to zoom-in using the magnifying glass tool so that you can notice the rapid modulation on the modulated signal `u` which exactly what the modulator introduces in the signal `m`.

Now we can empirically verify that we have perfect demodulation by listening to the reconstructed signal and looking at its spectrum:

```
% Look at the demodulated signal in the time domain
figure;
plot(t,dm); grid on; axis tight;
% Look at the spectrum of the demodulated signal
figure;
plot(f,am_spectrum(dm)); grid on; axis tight;
soundsc(dm,fs);
```

Notice that the demodulated signal sounds the same as the original. We can also use our plotting function to compare all the signals side by side. Typing:

```
am_plot(t,m,u,dm);
am_plot(f,am_spectrum(m),am_spectrum(u),am_spectrum(dm));
```

we get all the previous plots together for an easy comparison.

It is easier to experiment with different carrier frequencies by encapsulating the methodology we just presented in a function. Write the following function and save it under the `/lab3` directory using the filename `am_test.m`.

```
function [dm,e] = am_test(fc,dfc,bplot)
% AM_TEST Tests the effect of carrier frequency on a speech signal
%   [dm,e] = am_test(fc,dfc)
%
%   fc:    carrier frequency in Hertz
%   dfc:   delta carrier frequency to simulate lack of frequency coherence
%   bplot: plotting flag (1:plot, 0:don't plot)
%   dm:    demodulated signal
%   e:     the error, original minus demodulated

% Default values
if nargin < 2; dfc   = 0; end % No coherence error
```

```
if nargin < 3; bplot = 1; end % Plot by default
% Load the modulating signal (speech) and its sampling frequency
[m,fs] = wavread('sm1_cln_32k.wav');
% Keep only a small part of it to speed up calculations
m = m(1:42000);
% Modulate ...
u  = mymod(m,fc,fs);
% ... and demodulate
dm = mydemod(u,fc+dfc,fs);

if bplot
    % Make the frequency index for plotting
    t = linspace(0,length(m)/fs,length(m));
    f = linspace(-fs/2,fs/2,length(m));
    am_plot(t,m,u,dm);
    am_plot(f,am_spectrum(m),am_spectrum(u),am_spectrum(dm));
end
% How does the demodulated one sound like ?
soundsc(dm,fs);
% And the error is
e = m-dm;
```

Verify that MATLAB can find your function by typing `help am_test`.

Now we are ready to empirically assert the effect of choosing an inappropriate carrier frequency by listening to the demodulated signal. Call the function we just wrote for various carrier frequencies. For example start at 8000Hz and sweep the frequency going down to 6000, 4000, 2000 or as low as 200 Hz. You can do this by typing for example:

```
[dm8000,e8000] = am_test(8000);
[dm4000,e4000] = am_test(4000);
[dm200, e200 ] = am_test(200); % etc
```

Can you find the lowest carrier frequency for which we have almost perfect demodulation?

Can you verify your answer both acoustically as well as by looking at the spectrum of the demodulated signal in comparison with the original ?

At this point we are ready to evaluate the effect of lack of frequency coherence between the carriers at the modulator and the demodulator. We can find the theory in example 4.18/p281 of the lathi textbook. The parameter `dfc` in `am_test` corresponds to $\Delta\omega$ in the lathi example. Call the function we just wrote for different values of `dfc`. For example start with:

```
[dm8000_5,e8000_5] = am_test(8000,5);
```

Notice that the amplitude of the demodulated signal is modulated and speech sounds like `tremolo` in music terms. Notice the modulated envelope in the time domain plot of the demodulated signal. It is interesting to play with less realistic frequency errors by using values of 100, 500 or even 1000 for the `dfc` parameter. Type:

```
[dm8000_1000,e8000_1000] = am_test(8000,1000);
```

An interesting sound effect to say the least this situation could correspond to us turning some analog knob on an old tuner trying to tune in the right frequency. Notice that if you want to concentrate on listening to different examples you can suppress the plotting by adding a third argument:

```
[dm8000_4000,e8000_4000] = am_test(8000,4000,0);
```

Lastly the command that you can definitely appreciate at this point is `close all` to close all open figures at once.

At this point you should turn off the diary function we started in the beginning of this lab session. You can do that by typing:

```
>> diary off;
```

You should show the LA or the TA the diary file.