EE354: Lab 5-AM & FM Demodulation via Superheterodyne

Objective: This lab explores SDR techniques for demodulating AM and FM waveforms. The lab also explores design and simulate the functions of a communication system mixer, upconverter and downconverter.

Part 1: Pre-Lab Theory

The following is a block diagram of a heterodyne transmitter and receiver for a DSB-TC AM communication system. Your objective is to design the system such that it meets the following specifications:

Transmitted Signals:
1. DSB-TC AM with audio modulation ($f_{\text{max}} = 3.3$ kHz) at 10.7 MHz.
2. Wideband FM with audio modulation ($f_{\text{max}} = 15$ kHz) at 21.4 MHz.

Modulation Index:
1. AM: $a_{\text{mod}} \approx 0.7$
2. FM: $\beta \approx 5$

Receiver IF Frequency: 300 kHz

At a minimum, your design must specify the following:

AM
- Receiver IF frequency:
- Receiver LO frequency (HSI):
- Receiver LO frequency (LSI):
- Image frequency (HSI):
- Image frequency (LSI):

FM
- Receiver IF frequency:
- Receiver LO frequency (HSI):
- Receiver LO frequency (LSI):
- Image frequency (HSI):
- Image frequency (LSI):

![Block diagram of a DSB-TC Superheterodyne AM Communication System](image)

Figure 1: Block diagram of a DSB-TC Superheterodyne AM Communication System

Instructor Verification:_____________
**Part 1a: Simulation of AM Demodulator**

This part of the lab will step you through creating an envelope detector in Matlab. A section entitled “Software Demodulation of a Pure Tone” is attached to the back of this lab and will aid you in your design.

First, use the following code to generate a DSB-TC AM signal that has a center frequency of 400 kHz, modulation index of 0.7, and modulating frequency of 500 Hz.

```matlab
clear all
close all

fs = #####;            % Establish our RF Sampling Frequency
fs_soundcard = ######; % Soundcard Sampling Frequency for audio playback
fc = #####;            % Carrier Frequency
fm = 500;              % message frequency 500 Hz
amod = ###;            % Modulation Index
A = ###;               % AM carrier amplitude

delt = 1/fs;
tstop = 0.5;          % Create signal for at least 1/2 of a second
t = 0:delt:tstop;     % Establish the time vector
m = cos(2*pi*fm*t);   % Create the Message Signal
u_am = A.*(1 + amod.*m).*cos(2*pi*fc*t);    %DSBTC Creation
```

1. In software, implement an envelope detector (recall that envelope detection in Matlab is basically, an absolute value and low pass filter. See the posted course notes for an example and the handout attached to the back of this lab) to recover the original information signal. Output the resulting demodulated signal via the PC’s sound card. Use the `filter_lowpass` function (available on the course website) to implement the LPF, and use a cutoff frequency of 20 kHz.

**Recall:** Your sampling frequency is likely set based on the frequency and bandwidth the RF signal. As a result, the sampling frequency will be many times larger than the PC’s sound card can handle. To obtain an acceptable audio sampling frequency, we have to decimate (remove redundant samples from) the demodulated signal. Decimation works by essentially throws away unnecessary samples. To implement, use the following commands:

```matlab
downsample_rate = floor(fs./fs_soundcard);  % Downsampling rate to audio
fs_audio = fs./downsample_rate;
s_demod = downsample(s_envelope_detected, downsample_rate);
t_demod = downsample(t, downsample_rate); % Must downsample time as well
```

The `downsample` command performs the decimation operation for us, and the variable `downsample_rate` tells Matlab how many samples to throw away in between samples that it retains. Notice that our net sampling frequency is reduced (which will be important when you are playing the demodulated signal on the PC’s sound card)

2. Output the recovered audio signal to the PC’s sound card and verify that you hear the 500 Hz tone. To play the sound, use the Matlab command `sound(s_demod, fs_soundcard)`. 
Part 1b: Simulation of FM Demodulator

First, use the following code to generate a FM signal with center frequency of 400 kHz, modulating frequency of 500 Hz, and frequency deviation of 40.0 kHz.

```matlab
clear all
close all

fs = #######; % Establish our Sampling Frequency
fs_soundcard = #######; % Soundcard Sampling Frequency (see previous labs)
fc = #######; % Carrier Frequency
fm = 500; % message frequency 500 Hz
Beta = ###; % FM Modulation Index
A = ###; % FM carrier amplitude

delt = 1/fs;
tstop = 0.5; % Create signal for 1/2 of a second
t = 0:delt:tstop; % Establish the time vector

m = sin(2*pi*fm*t); % Create the Message Signal
u_fm = A.*cos(2*pi*fc*t + Beta.*m); %FM Creation via PM
```

1. To demodulate the FM Signal, you will be connecting Matlab software routines based on the your choice of the three possible FM demodulators: Instantaneous Frequency, ArcTangent, or Complex Exponential. Block diagrams are provided below. Key components that should be present in any FM demodulator are:

   1. Limit the received signal to remove amplitude variations.
   2. Downsample the signal to avoid excess processing power and memory.
   3. Low-pass filter to remove undesired high frequency terms.
   4. Downsample the low-pass filter to an audio-level sampling frequency ($f_s < 44.1 kHz$)

2. In order to build your FM demodulator, you will need the following Matlab functions: `diff.m`, and `downsample.m`, and `filter_lowpass.m` (available on the course website). The `diff.m` command is used to implement the time derivative, and instructions for its use can be found in the Matlab Help files.

Recall: Your sampling frequency is likely set based on the frequency and bandwidth the RF signal. As a result, the sampling frequency will be many times larger than the PC’s sound card can handle. To obtain an acceptable audio sampling frequency, you need to decimate the demodulated signal just as you did with the AM signal.

3. Output the recovered audio signal to the PC’s sound card and verify that you hear the 500 Hz tone. To play the sound, use the Matlab command `sound(s_demod, fs_soundcard)`.
Figure 2: The three possible FM Demodulator Options: Instantaneous Frequency, ArcTangent, and Complex Exponential.
Part 2a: Setting up the Superheterodyne Receiver

At the instructor’s bench, you will find the following components:

- Amplifier: ZFL-500-BNC
- RF Filter: BBP-21.4 & BBP-10.7
- IF Filter: BLP-2.5
- Mixer: ZFM-2+
- Various BNC Cables and adapters

Specifications and datasheets for these components can be found on MiniCircuits website (www.minicircuits.com).

**Important:** Never power on the amplifier without connecting a load to the amplifier output (mixer, oscilloscope, or spectrum analyzer). Running the amplifier unloaded could cause it to go into oscillation and burn up.

1. The instructor’s bench is generating a combined AM and FM signal at the specified frequencies and at a power level of approximately -30.0 dBm. This signal is being distributed to all of the lab benches via coaxial cable. You will find the signal at the coaxial connection at the back of your lab bench. Use the spectrum analyzer to verify that all signals are present.

2. Connect the RF signal to the RF Filter, connect the output of the RF Filter to the amplifier. Using one of the short BNC cables, connect the amplifier output to the RF port on the mixer.

   **Note:** The Mixer ports are labeled R, I, and L, corresponding to the RF, IF, and LO ports shown in Figure 1.

3. Use an Agilent 33220A signal generator as the LO. Configure the signal generator for your LSI LO frequency, and set it to an output power of +7 dBm (Important: If the output power is not +7 dBm, the mixer will not be activated and frequency translation will not occur). Connect the output of the LO signal generator to the LO port on the Mixer.

4. Connect the IF output of the mixer to the spectrum analyzer, and power on the amplifier. Verify that the received signal has been translated to your designed IF Frequency.

![Figure 3: Block diagram of the equipment connection for evaluating the operation of a superheterodyne receiver.](image-url)
Part 2b: AM Demodulation

1. Connect the IF output of your superheterodyne receiver to the LeCroy Oscilloscope. Ensure the scope is set for a 50Ω input impedance. Adjust the oscilloscope to record 5.0 Million Samples, and configure the sampling frequency to be 1.0 MHz. Your timebase on the scope screen should be 500 msec/division. Turn off all measurement, math functions, and other channels.

2. Set your LO frequency to downconvert the AM signal to the 300 kHz IF.

3. Observe the AM signal on the scope (you should see a decent amount of amplitude change). Press the “Single” button on the scope, and when the scope is finished capturing the signal, save it to a Matlab .dat file.

4. Using your Matlab AM envelope-detector code, load the Oscilloscope captured data into Matlab (Note: it’s a big file, it may take some time!), filter the input samples, demodulate the AM signal, and play it using the PC’s sound card. Your demodulated signal in this case should be several seconds of whatever audio signal you were outputting from the PC.

5. Demonstrate successful AM Demodulation to the Instructor.

Part 2c: FM Demodulation

1. Connect the IF output of your superheterodyne receiver to the LeCroy Oscilloscope. Ensure the scope is set for a 50Ω input impedance. Adjust the oscilloscope to record 5.0 Million Samples, and configure the sampling frequency to be 1.0 MHz. Your timebase on the scope screen should be 500 msec/division. Turn off all measurement, math functions, and other channels.

2. Set your LO frequency to downconvert the FM signal to the 300 kHz IF.

3. Observe the FM signal on the scope (you should see a decent amount of amplitude change). Press the “Single” button on the scope, and when the scope is finished capturing the signal, save it to a Matlab .dat file.

4. Using your Matlab FM envelope-detector code, load the Oscilloscope captured data into Matlab (Note: it’s a big file, it may take some time!), filter the input samples, demodulate the AM signal, and play it using the PC’s sound card. Your demodulated signal in this case should be several seconds of whatever audio signal you were outputting from the PC.

5. Demonstrate successful FM Demodulation to the Instructor.
Software Demodulation of an AM Signal

To recover the original information signal from the received AM waveform (whether in simulation or from data captured off the scope), you will need to implement an envelope detector in Matlab. The envelope detector consists of the following four steps:

1. Take the absolute value of the AM waveform to form $|s_{am}(t)|$ (where $s_{am}(t)$ is the received AM signal).
2. Low-Pass filter the rectified signal to form $\hat{m}(t)$.
3. Downsample the low-pass filter to an audio-level sampling frequency ($f_s < 44.1 kHz$).
4. Output the resulting audio signal via the PC’s speaker using the `sound` command.

In order to build your AM demodulator, you will find the following Matlab functions useful:

```matlab
filter_baseband(x, fco, fs) – (Available on the EE354 Course Website) Implements a simple Low Pass Filter to prevent aliasing of the audio signal when sampled. Note that fco must be less than half of fs and that fco is the -3dB cutoff point.
downsample(x, N) – (a native Matlab function) downsamples input signal X by keeping every Nth sample starting with the first.
```

Note 1: The cutoff frequency you use in the `filter_baseband.m` function should be no lower than approximately $0.005f_s - 0.01f_s$, otherwise the filter will simply output a flat line.

Note 2: Because the sampling frequency is set for the RF signal, the sampling frequency is many times larger than the PC’s sound card can handle. To obtain an acceptable audio sampling frequency, we need to decimate the demodulated signal. Decimation works by throwing away unnecessary samples. To implement, use the following commands:

```matlab
downsample_rate = floor(fs_original./fs_soundcard);
fs_audio = fs_original./downsample_rate;
s_demod = downsample(m_hat, downsample_rate);
```

The `downsample` command performs the decimation operation for us, and the variable `dec_rate` tells Matlab how many samples to throw away in between samples that it retains. Notice that our net sampling frequency is reduced (which will be important when you are playing the demodulated signal on the PC’s sound card)

For our purposes, the soundcard operates at a sampling frequency of 44.1 kHz (i.e., $fs_{soundcard} = 44.1e3$). For the Matlab Simulation, $fs_{original}$ will be the sampling frequency you used to generate and simulate the AM signal. For the oscilloscope data, $fs_{original}$ will be the sampling frequency of the oscilloscope.