Name: _____

EE354 Spring 2011 Lab 10: Superheterodyne Receivers

In this lab you will be using MATLAB to design and simulate the functions of a communication system mixer, upconverter and downconverter. Your design will be verified via hardware implementation..

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 Signal: DSB-TC AM with sinusoidal modulation (f_{max} = 5 kHz).
Modulation Index: 70%
RF Frequency: 10.7 MHz.
Receiver IF Frequency: 1.0 – 2.5 MHz (Choose an IF frequency within this range)

At a minimum, your design must specify the following:

Receiver IF frequency: _____

Receiver LO frequency (HSI): _____

Receiver LO frequency (LSI): _____

Image frequency (HSI): _____

Image frequency (LSI): _____

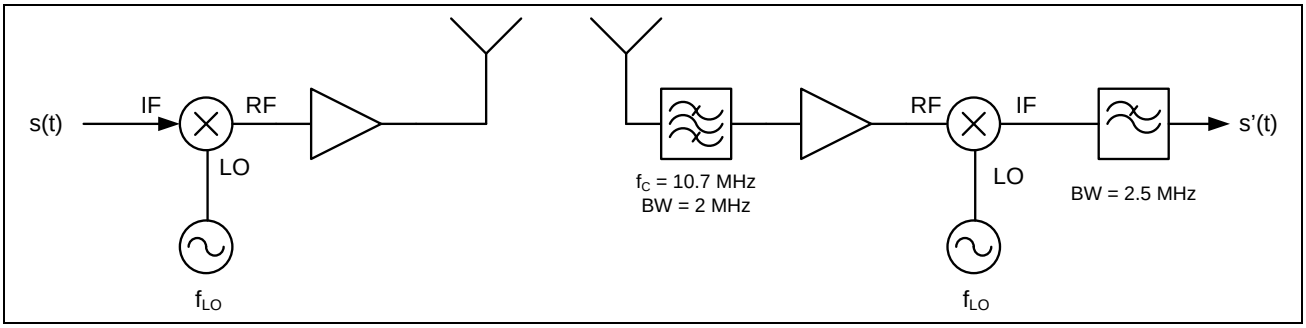


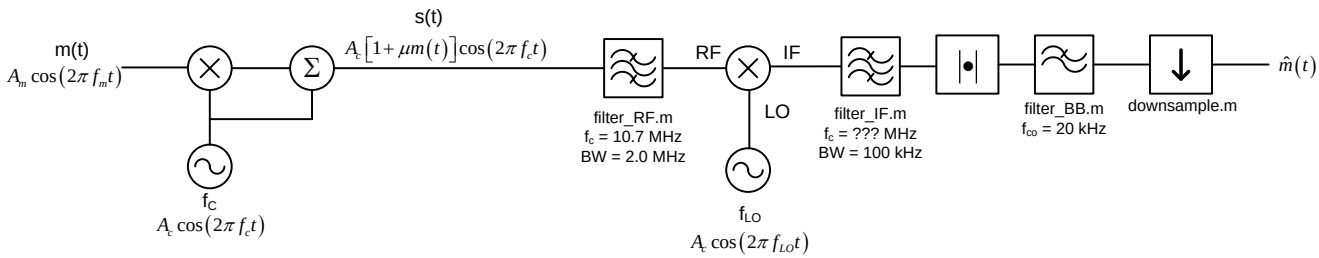
Figure 1: Block diagram of a DSB-TC Superheterodyne AM Communication System

Instructor Verification:_____

Part 2: Simulation

This part of the lab will step you through creating a superheterodyne receiver in Matlab. Many of these building blocks will be used in later labs where you will capture and demodulate a real AM or FM signal. Use the `pwelch` function to generate power spectral density plots where required.

For the simulation, you will be connecting Matlab software routines based on the following block diagram:



For your filters, use the following parameters:

- RF Bandpass Filter: BW = 2.0 MHz
- IF Bandpass Filter: BW = 100 kHz
- Baseband Lowpass Filter: BW = 30 kHz

You will need the `filter_RF` , the `filter_IF,` and the `filter_BB` functions available from the course website. These functions implement the RF bandpass filter, IF bandpass filter, and demodulator lowpass filter. You can use the `test_filter` function to verify that the parameters you choose for your filter will result in a filter that behaves as intended.

1. Generate the DSB-TC AM signal in Matlab. Verify that you have the proper signal in both the time and frequency domains and that your parameters (center frequency, modulation index, etc.) are correct.

2. Using the `filter_RF` function, bandpass filter your RF signal, and verify that you get the output you intend (in both the time-domain and frequency-domain).

3. Generate your receiver local oscillator signal (plot it to verify that it is correct), and use that signal to downconvert the RF signal to IF.

4. Use the `filter_IF` function (download from the website) to implement the IF filter. Verify that you get the output you intend (in both the time-domain and frequency-domain).

5. Generate the frequency spectrum of the received IF signal. Zoom in around the portion close to where the peak occurs so that you can see whether the AM spectrum appears at the IF frequency. Include a copy of this plot with your report.

6. Did the IF signal show up at the desired frequency? Is the signal bandwidth and number/amplitude of sidebands correct? Why or why not? If you have errors, correct your design before proceeding.

   _____

   _____

   _____

7. 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_audio` function to implement the LPF.

   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:

```
dec_rate = floor(fs./fs_soundcard); % Soundcard Sampling frequency is 44.1 kHz
fs_audio = fs./dec_rate;
s_demod = downsample(s_envelope_detected, dec_rate);
t_demod = downsample(timevector, dec_rate); % Must downsample time as well
```

   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)

8. Output the recovered audio signal to the PC's sound card and verify that you hear the 5.0 kHz tone. Plot the recovered signal and turn in a copy with your report.

   Does the recovered signal match the original signal? If not, explain why not?

   _____

   _____

   _____

Instructor Verification of Audio Recovery:_____

Part 3: Hardware Measurements of the AM Signal

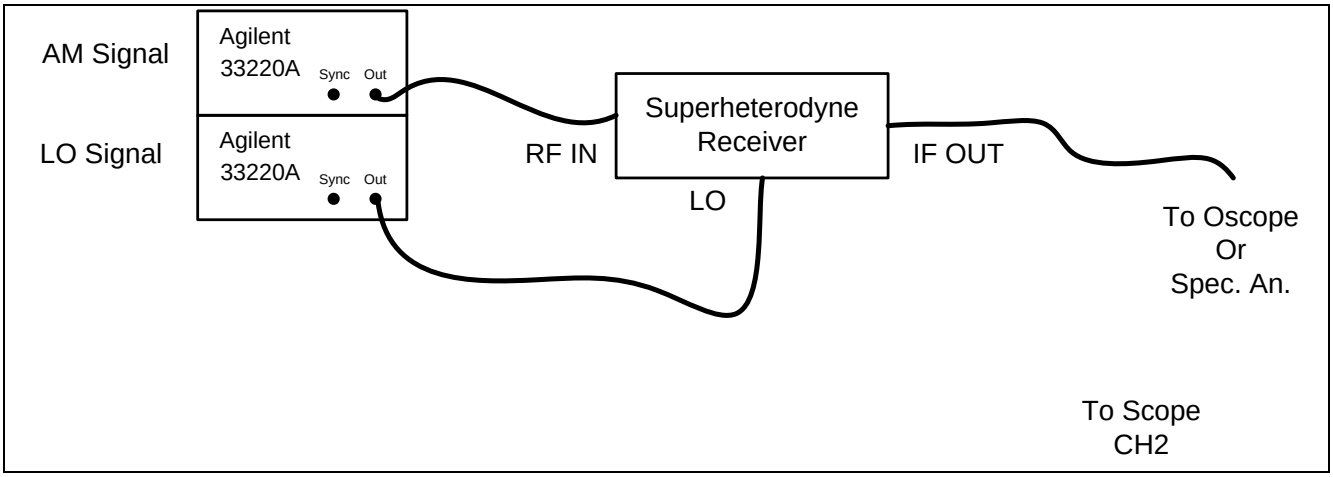


Figure 2: Block diagram of the equipment connection for evaluating the operation of a superheterodyne receiver.

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

Amplifier: ZFL-500-BNC
RF Filter: 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).

1. Configure one of the Agilent 33220A signal generators to produce an AM signal with a carrier frequency of 10.7 MHz, Internal 5.0 kHz modulation, Modulation Index of 70%, and output power of -34.0 dBm.

2. Connect the output of the AM signal generator to the RF Filter, connect the output of the RF Filter to the amplifier, and connect the output of the amplifier to the input of the Anritsu Spectrum Analyzer. The amplifier requires +16 to +18 VDC supply in order to operate, so connect the amplifier power to the benchtop DC power supply. DO NOT power on the amplifier at this stage.

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.

3. Power on the amplifier, and, using the spectrum analyzer, verify the presence of your AM signal at the output of the AM signal generator (you should see the carrier and both sidebands). Print out a copy of your RF spectrum and include it with your report. Use the spectrum analyzer Marker Demod function to demodulate the AM signal and verify that you get a 5.0 kHz tone.

4. Power down the amplifier and disconnect the amplifier output from the spectrum analyzer. 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.

5. Use a second 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.

6. 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 and chosen IF Frequency. Change the LO to your HSI LO frequency, and verify that superheterodyne receivers operate with either LO frequency. Print out a copy of your IF spectrum and include it with your report.

7. Connect the IF filter to the output of the Mixer, and connect the filter output to Channel 1 of the LeCroy oscilloscope. Connect the Sync output from the AM signal generator to Channel 2 of the oscilloscope, and configure the scope to trigger on Channel 2.

8. Configure the LeCroy oscilloscope to display several cycles of the IF signal. Use the oscilloscope to measure the IF carrier frequency and message frequency and record these values below:

   $f_{carrier}$ = _____

   $f_{message}$ = _____

9. Configure the oscilloscope to operate at a sampling frequency of 5.0 MHz and record 500,000 samples. Record the IF signal to a Matlab .dat file. Using Matlab, demodulate your captured data via an envelope detector (similar to what you did in the simulation → Note that the sampling frequency is different!), and plot the output in the time domain. Include a copy of the plot in your lab report.

10. Output the recovered signal on the PC's sound card and demonstrate the operation of your receiver to the instructor.

Instructor Verification of Successful Downconversion and Demodulation: _____

<u>Items to turn in with your report</u>

Plot of the receiver IF signal spectrum from Part 1, Step 4.
Plot of the recovered signal from Part 1, Step 6.
Plot of the received RF spectra from Part 2, Step 2.
Plot of the received IF spectra from Part 2, Step 5.
Plot of the recovered signal after envelope detection from Part 2, Step 6.

Software Demodulation of a Pure Tone

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 $|r(t)|$ (where $r(t)$ 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 ($f_s < 44.1kHz$).
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:

`filter_bb(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 N^{th} sample starting with the first.

Note 1: The cutoff frequency you use in the filter_bb.m function should be no lower than approximately $0.005 f_s - 0.01 f_s$, 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 <u>decimate</u> the demodulated signal. Decimation works by throwing away unnecessary samples. To implement, use the following commands:

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