EE 3410 Take-Home-Quiz

Fall, 2014

Prof. Steven Grant

In this take-home quiz, you will design a receiver for an “Audio Radio” in Matlab. This is a completely useless device, but it might be kind of fun to play with. It will demonstrate the use of the modulation property of the DTFT for communications. The math is the same as that of a normal AM radio, but instead of using electromagnet waves as the communication medium we use sound. So instead of transmit and receive antenna’s we use loudspeakers and microphones.

Since you are only to design the receiver, I have provided a recorded received signal. The file, “radiodata.wav” is posted on Blackboard in the Content section.

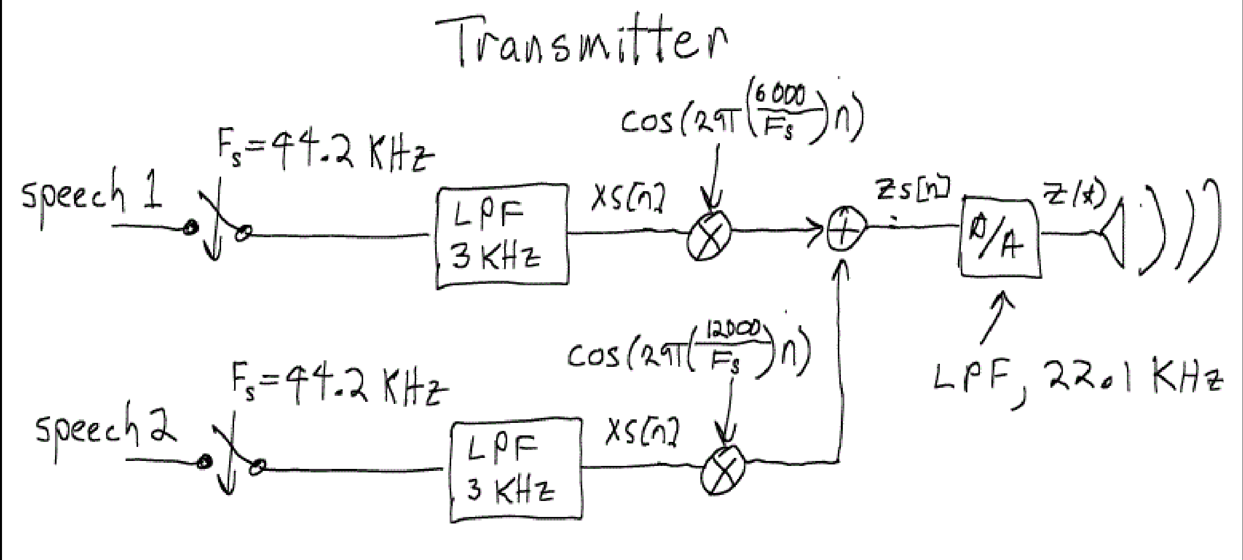

Figure 1

The “radiodata.wav” signal is generated in MATLAB as shown in figure 1. Two speech signals have been sampled at the sample rate of, Fs=44.2 KHz. Then, they were low-pass filtered (LPF) with a cutoff frequency of 3 KHz. The resulting signals, xs1[n and xs2[n] were modulated by sinusoids, cos(2*pi*(6000/Fs)n) and cos(2*pi*(12000/Fs)n), respectively. They were then added to obtain the sampled transmit signal, zs[n]. This was then sent to the computer’s loudspeaker through its digital-to-analog (D/A) converter that has an anti-aliasing filter which low pass filtered zs[n] by half the sample rate, Fs/2=22.1KHz. The output signal is z(t). The resulting magnitude spectra of of zs[n] and z(t) are shown in figures 2 and 3 respectively. z(t) was played out of a loudspeaker, recorded with the microphone in my computer and sent to the 44.2KHz sampled wav file, “radiodata.wav” .

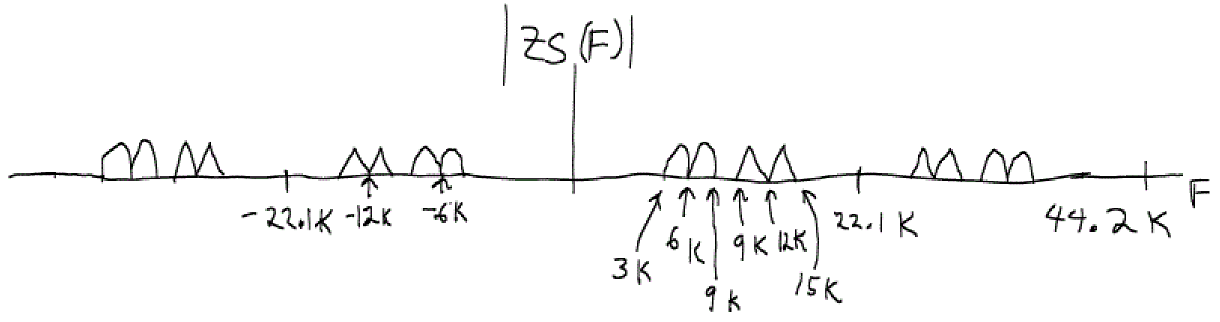

Figure 2

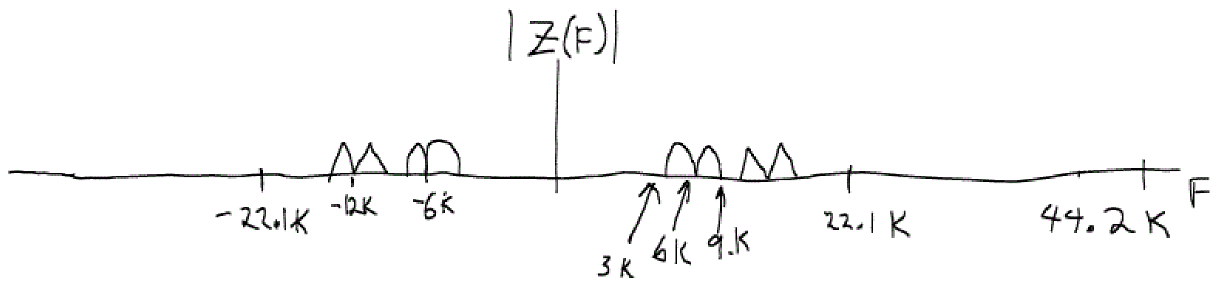

Figure 3

WHAT TO DO: You are to build a receiver in Matlab for each of the channels in the received signal.

1) Read in the received signal in the radiodata.wav file using the" wavread()" function
2) Design band-pass filters to select the desired signal and reject the other.
3) Filter the received signal.
4) Design a mixing signal to de-modulate the signal. (Hint, it's the same one that modulated it)
5) Demodulate the signal
6) Now get rid of the extra signal in the high band from the demodulation process with an appropriate low pass filter (design and implement this filter, too). The result is the "base-band" signal
7) Listen to both the received and base-band signal. Do you hear the difference? The original signal is unintelligible, but in the second you should be able to make out the speech pretty clearly. . (Don't expect the signal to sound perfect. My result had some buzzing in it due to the noise in the room at the frequencies around 6 KHz and 12 KHz. Don't worry about it.)

8) Create a “myanswer1.wav” file of the base-band signal for the lower channel.
9) Repeat steps 4 through 8, but this time process the upper channel. Create a “myanswer2.wav” file for the base-band signal of the upper channel.
10) **Then, email myanswer1.wav and myanswer2.wav, and your Matlab code to me at <sgrant@mst.edu>. Be sure to write in the subject line (exactly), “EE 3410 Take Home Quiz”. If you don’t write that exactly you may not get credit for the quiz.**

RULES for this quiz: Because this is a take-home quiz we will be going by the honor system. For this particular quiz you may discuss the solution with other students in the class using the “hands in your pocket” rule. That is, you can say anything you want to each other as long as both of you have your hands in your pockets. On the other hand, copied code will not be tolerated. If I find code has been copied, all involved will receive a zero for the quiz.

SOME USEFUL HINTS

Matlab’s normalized frequency: Unfortunately, Matlab does not use the standard normalized frequency, f=F/Fs, like we do in class (here F is the cyclic frequency, Fs is the sample rate, and f is the normalized cyclic frequency). Instead of normalizing at the sampling rate, Fs, Matlab normalizes at half the sampling rate, Fs/2. Let us call the Matlab normalized frequency, r. Then r=2F/Fs. Be aware of this if you use Matlab’s help command to get help with the frequency design program. (By the way, for any Matlab command you can get help at the Matlab command prompt by just typing, “help function_name.”)

Reading in the received signal: Use the Matlab function, “wavread()” to read in the received signal provided on Blackboard. Example:

```
[received_signal ,Fs,dontcare]= wavread(‘radiodata.wav’);
```

Where,

radiodata.wav is the name of the received data file,

received_signal is a very long vector of received signal,

Fs is the sampling frequency (should be 44.2 KHz), and

dontcare is something we don’t care about.

This might be a stereo file with two very long columns. If so, just get rid of the second one by using,

```
received_signal = received_signal (:,1);
```

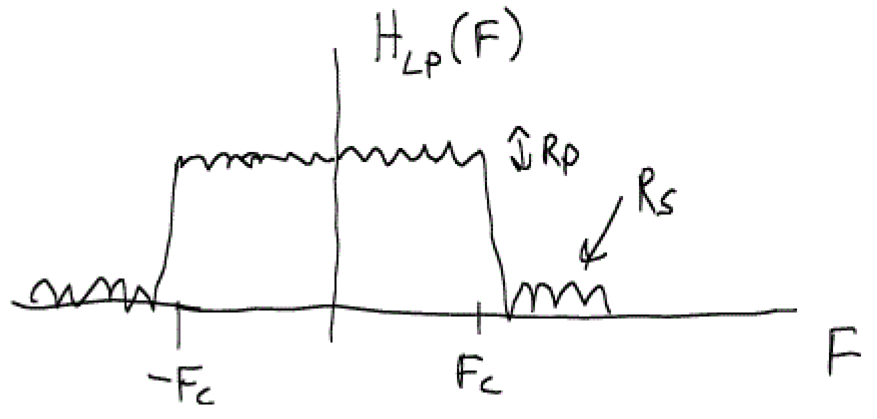

Figure 4: Lowpass filter

How to Design Filters in Matlab: For the filter design I recommend you use Matlab's "ellip()" function. It design's a so-called elliptical filter. Let's say I want to design a LPF with cutoff frequency of Fc. The magnitude frequency characteristic of the desired filter is shown in figure 4.

The command is typically written as,

```
[b,a]=elllip(N,Rp,Rs,2*Fc/Fs);
```

Where:

N is the filter order (8 is generally a good number),

Rp is the passband ripple in dB (0.5 works well),

Rs is the stopband rejection in dB (50 is good enough),

Fc is the filter's cutoff frequency,

Fs is the sampling frequency, and

b and a are the filter coefficients (you will use this when you implement the filter)

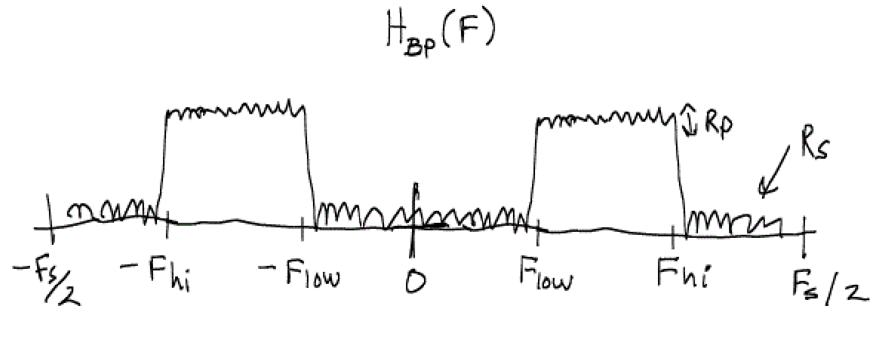

Figure 5: Bandpass filter

For a band pass filter with a low cutoff frequency of, Flow, and a high cutoff frequency of, Fhigh, one would write,

```
Wp=2.*[Flow,Fhigh]./Fs;
```

~~[b,a]=elllip(N,Rp,Rs,2*Fc/Fs,'high');~~

```
[b,a]=elllip(N,Rp,Rs,Wp);
```

To examine a filter: use the "freqz(b,a)" command. It plots the filter's magnitude and phase response between r=0 and r=1.

To implement a filter: use the "filter()" command. Here is an example,

```
y=filter(b,a,input_signal_vector);
```

Here,

b and a are the filter coefficients designed using the "ellip()" command,

input_signal_vector is the input signal expressed as a vector (usually a very big vector),

y is the output signal vector of the filter (as big as input_signal_vector).

Mixing (Modulating) two signals. See in figure 5 below where we multiply two signals, x_tilde[n]=x[n]*cos(2*pi*(F0/Fs)*n). We call this modulating a sinusoid with carrier frequency F0 with the signal x[n].

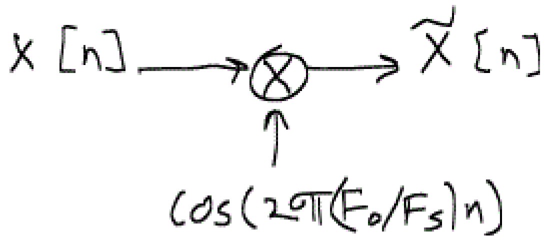

Figure 4

This would be implemented in Matlab as follows.

1) Let x be the signal vector (a very long column vector).

2) Create a time index vector, time=[1:length(x)]'; (note the ' at the end of the closing square bracket. This makes the vector a column vector, the kind I like to work with.
3) Create the mixing vector, mix=cos(2*pi*(Fo/Fs)*time);
4) Finally, mix the signals using the element -by-element multiply, x_tilde=x.*mix; (if you try the vector multiply, x*mix, Matlab will try to do an inner product and complain that the inner dimensions of the matrices, x and mix should be the same).

How to write a *.wav file: Use the "wavwrite()" function. Example:

```
wavwrite(output_signal_vector,Fs,16,'my_result');
```

Where

output_signal_vector is the signal vector you want to write to memory

Fs is the sampling rate

16 is the number of bits per sample (standard)

myresult.wav is the file the signal will appear in.

To listen to a signal vector: Use the "soundsc()" command. Example:

```
soundsc(sound_vector,Fs)
```

Where,

sound_vector is the signal you want to listen to, and

Fs is the sampling frequency.

Note that *.wav files only save signal samples with values between -1 and +1. To prevent "clipping" of the signal try the command,

```
output_signal_vector=0.98.*output_signal_vector./max(abs(output_signal_vector));
```

This will prevent clipping when using the wavwrite() command.