Massachusetts Institute of Technology
Dept. of Electrical Engineering and Computer Science
Spring Semester, 2007
6.082 Introduction to EECS 2

Lab #3: Modulation and Filtering

Goal:

Learn principles of modulation and filtering through hands on interaction with Matlab and the USRP board. In particular, students will become familiar with a wireless communication scheme known as frequency division multiplexing, and will write Matlab code to receive voice signals that have been modulated to different frequency bands. The issue of frequency offset in demodulation will also be examined.

Instructions:

1. Complete the Prelab exercises ***BEFORE*** Wednesday's lab.
2. Complete the activities for Wednesday's lab (see below) and get checked off by one of the TAs before leaving. Be sure to work in pairs with one computer per pair.
3. Complete the Postlab exercises ***BEFORE*** Friday, and turn them in at the beginning of lecture on Friday.

Prelab:

Modulation, *demodulation*, and *filtering* are integral to the frequency division multiplexing (FDM) communication system you will be building in Lab 3. The purpose of these prelab exercises is to familiarize you with the implementation of these signal processing operations using simple sinusoidal signals.

A. Understanding Modulation

The purpose of this exercise is to understand the mechanics of modulation and demodulation as well as to study the effects of these operations on signals in the time and frequency domains. To achieve this purpose we will study the modulation and demodulation of a 1 Hz sinusoid by 10 Hz.

Consider the signal $x(t) = \cos(2\pi * t)$. This is a sinusoid with amplitude of 1 and frequency of 1 Hz. Create a Matlab file called "modulation.m" and enter the following commands to create and visualize x(t) in the time domain. Your code should generate Figure 1.

```
Fs = 1000;                  % Sample Frequency in Hz
t = 0:1/Fs:5;               % Time vector form 0-5 seconds in increments of
                            %  0.001 seconds (eg. 0 0.001, 0.002, 0.003 ...)

x = cos(2*pi*1*t);          % Cosine with amplitude 1 and frequency 1 Hz

figure(1)                   % Create Figure 1
plot(t,x)                   % Plot Signal x(t) in Figure 1
title('Time-Domain');       % Add Title Time-Domain to Figure 1
xlabel('Time (sec)');       % Add Time (sec) as x-axis label of Figure 1
ylabel('Amplitude');        % Add Amplitude as y-axis label of Figure 1
axis([0 1 -2 2])            % Set x-axis range [0-1] sec and y-axis range [-2 2]
```

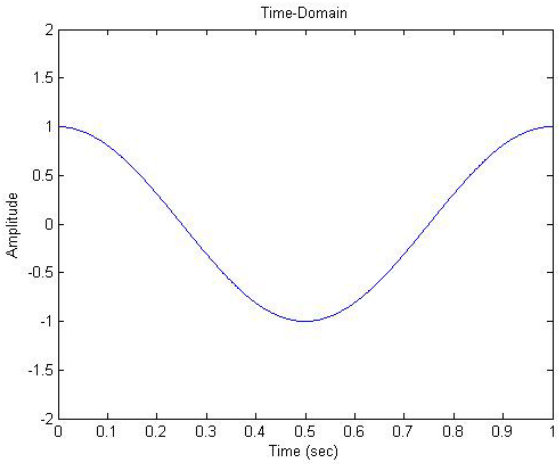


Figure 1

Now lets visualize the magnitude of the spectrum of x(t). Since x(t) is a real signal, the magnitude of its spectrum is an even function (symmetric about 0 Hz). We expect to see a peak at 1 Hz and another at -1 Hz in the magnitude of the spectrum of x(t). Add the following commands to your Matlab file, and rerun it to visualize the magnitude of the spectrum of x(t). Your code should now generate Figure 2.

```
X = fft(x);                    % perform FFT on signal x(t)
N = length(x);                 % Make frequency vector f
f = [-N/2:N/2-1]*(Fs/N);       % that spans frequencies from -500 Hz to 500 Hz

figure(2)                      % Create Figure 2
plot(f,abs(fftshift(X(1:N))))  % Plot positive and negative spectrum of x(t)
axis([-5 5 0 3000]);           % Set x-axis range [-5 5] Hz and y-axis [0 3000]
title('Frequency Domain');     % Add Frequency Domain as title of Figure 2
xlabel('Frequency (Hz)');      % Add Frequency (Hz) as x-axis label of Figure 2
ylabel('Magnitude');           % Add Magnitude as y-axis label of Figure 2
```

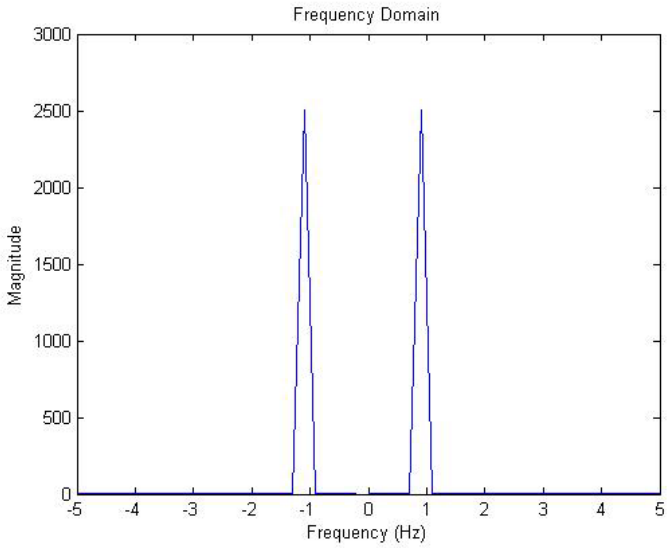


Figure 2

Now lets modulate x(t) upwards by 10 Hz. To do this we need to generate the modulating signal $y(t) = \cos(2\pi * 10t)$, and then create the modulated signal m(t) = x(t)y(t). Recall from lecture that we can use convolution operations in the frequency domain to analyze modulation, but we can also use a simple trigonometric identity as well:

$$\cos(2\pi f_1 t) * \cos(2\pi f_2 t) = \frac{\cos\{2\pi(f_1 + f_2)t\} + \cos\{2\pi(f_1 - f_2)t\}}{2}$$

$$m(t) = \cos(2\pi t) * \cos(2\pi * 10 * t) = \frac{1}{2}\cos(2\pi * 11t) + \frac{1}{2}\cos(2\pi * 9t).$$

Add the following commands to your Matlab file, and rerun it to visualize the in the time-domain representation of the modulated signal m(t). Your code should generate a plot similar to that in Figure 3.

```
y = cos(2*pi*10*t);          % Create modulating signal y(t); a cosine
                             % with amplitude 1 and frequency 10 Hz

m = x.*y;                    % Multiply signal x(t) by modulating signal
                             % y(t) to create the modulated signal m(t).

figure(3)                    % Create Figure 3
plot(t,m)                    % Plot Signal m(t) in Figure 3
title('Time-Domain');        % Add Title Time-Domain to Figure 3
xlabel('Time (sec)');        % Add Time (sec) as x-axis label of Figure 3
ylabel('Amplitude');         % Add Amplitude as y-axis label of Figure 3
axis([0 1 -2 2])             % Set x-axis range [0-1] sec and y-axis range [-2 2]
```

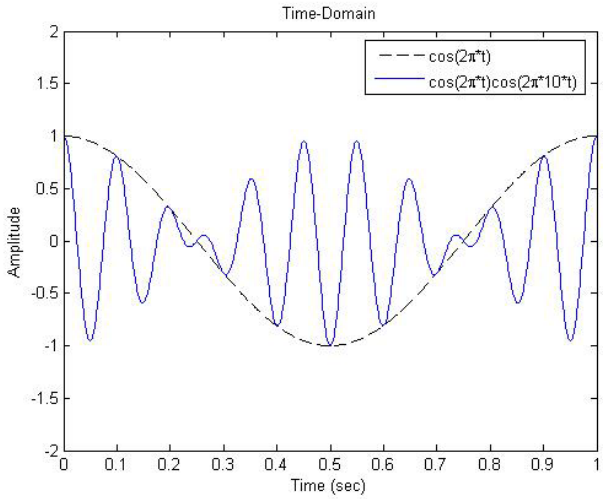


Figure 3

Now lets visualize the magnitude of the spectrum of m(t). Since m(t) is a real signal, the magnitude of its spectrum is an even function (symmetric about 0 Hz). We expect to see a peak at +/- 9 Hz and +/- 11 Hz in the magnitude of the spectrum of m(t). Add the following commands to your Matlab file, and rerun it to visualize the magnitude of the spectrum of x(t). Your code should now generate a plot similar to Figure 4.

```
M = fft(m);                  % perform FFT on signal m(t)
f = [-N/2:N/2-1]*(Fs/N);     % Make frequency vector f
                             % that spans frequencies from -500 Hz to 500 Hz

figure(4)                    % Create Figure 4
plot(f,abs(fftshift(M(1:N))));% Plot positive and negative spectrum of m(t)
axis([-20 20 0 1500]);       % Set x-axis range [-20 20] Hz and y-axis [0 1500]
```

```
title('Frequency Domain');  % Add Frequency Domain  as title of  Figure 4
xlabel('Frequency (Hz)');   % Add Frequency (Hz) as x-axis label of Figure 4
ylabel('Magnitude');        % Add Magnitude as y-axis label of Figure 4
```

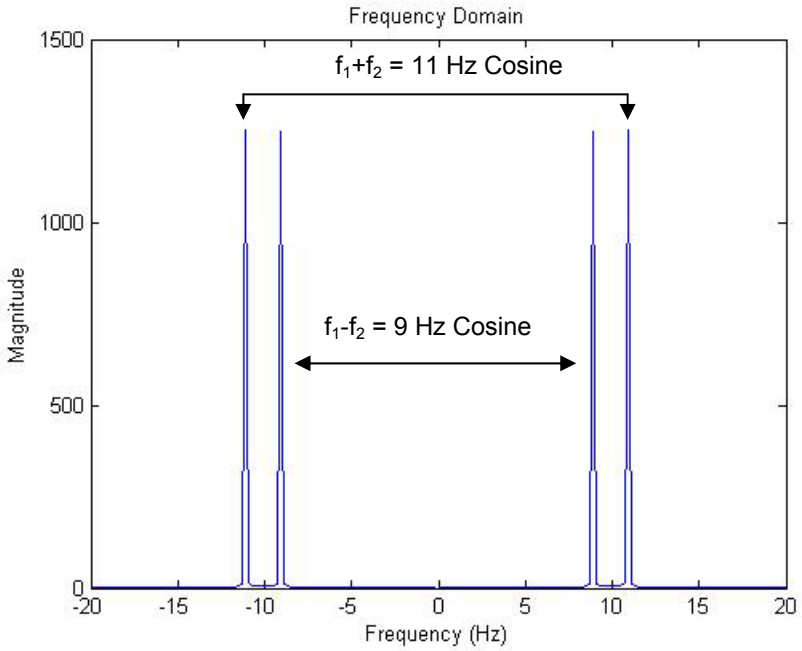


Figure 4

At this point you have completed modulating the signal x(t) by 10 Hz. You have implemented the block diagram shown in Figure 5.

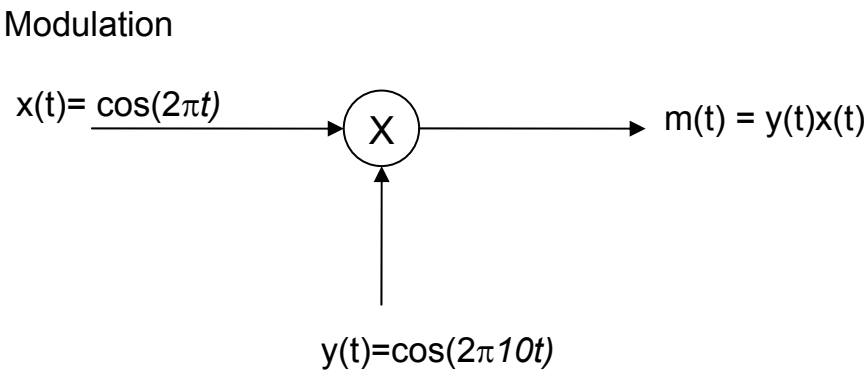


Figure 5

B. Understanding Demodulation

Now let's demodulate the signal m(t) downward by 10 Hz in order to recover the signal x(t). To do this we multiply m(t) by y(t). From trigonometry, we see that the resulting signal c(t) should have the form:

$$c(t) = \frac{1}{4}\cos(2\pi(f_1 - 2f_2)t) + \frac{1}{2}\cos(2\pi f_1 t) + \frac{1}{4}\cos(2\pi(f_1 + 2f_2)t)$$

$$c(t) = \frac{1}{4}\cos(2\pi * 19t) + \frac{1}{2}\cos(2\pi * t) + \frac{1}{4}\cos(2\pi * 21t)$$

Add the following commands to your Matlab file, and rerun it to visualize the in the time-domain representation of the demodulated signal c(t) You should see a plot like that in Figure 6.

```
c = m.*y;                    % Multipy modulated signal m(t) by y(t) to
                             % create demodulated signal c(t)

figure(5)                    % Create Figure 5
plot(t,c)                    % Plot Signal c(t) in Figure 5
title('Time-Domain');        % Add Title Time-Domain to Figure 5
xlabel('Time (sec)');        % Add Time (sec) as x-axis label of Figure 5
ylabel('Amplitude');         % Add Amplitude as y-axis label of Figure 5
axis([0 1 -2 2])             % Set x-axis range [0-1] sec and y-axis range [-2 2]
```

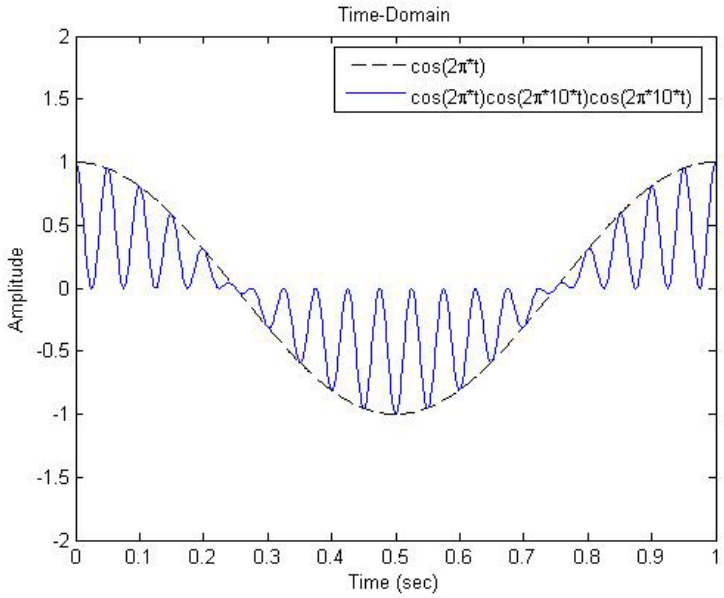


Figure 6

Now visualize the magnitude of the spectrum of c(t). Since c(t) is a real signal, the magnitude of its spectrum is an even function (symmetric about 0 Hz). We expect to see

a peak at +/- 21 Hz, +/- 19 Hz, and +/-1 Hz. Add the following commands to your Matlab file, and rerun it to visualize the magnitude of the spectrum of c(t). Your code should now generate a plot similar to Figure 7.

```
C = fft(c);                    % perform FFT on signal m(t)
f = [-N/2:N/2-1]*(Fs/N);       % Make frequency vector f
                               % that spans frequencies from -500 Hz to 500 Hz

figure(6)                      % Create Figure 6
plot(f,abs(fftshift(C(1:N))));% Plot positive and negative spectrum of c(t)
axis([-30 30 0 1500]);         % Set x-axis range [-30 30] Hz and y-axis [0 1500]
title('Frequency Domain');     % Add Frequency Domain as title of Figure 6
xlabel('Frequency (Hz)');      % Add Frequency (Hz) as x-axis label of Figure 6
ylabel('Magnitude');           % Add Magnitude as y-axis label of Figure 6
```

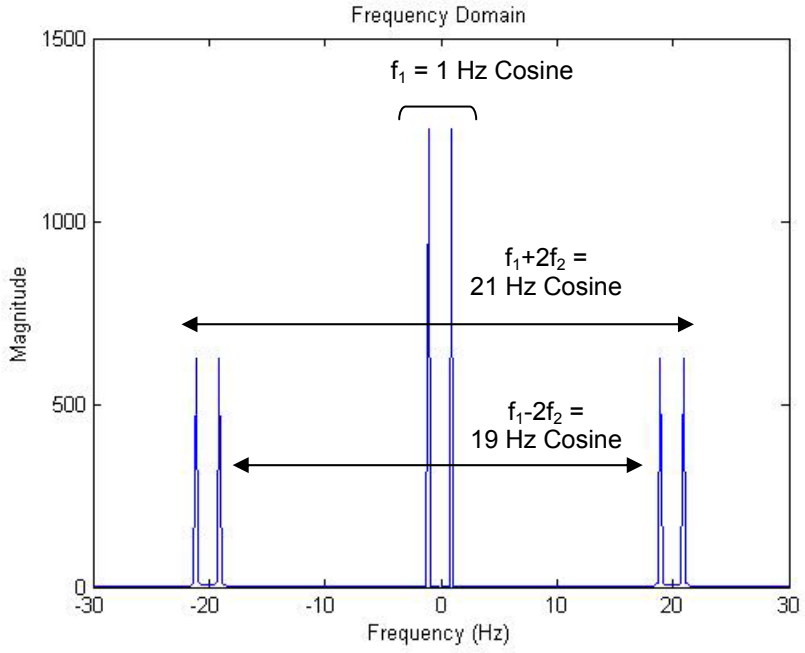


Figure 7

Note that we did not recover x(t); we recovered x(t) as well as some higher frequency components. To remove the higher frequency components (19Hz and 21Hz components) we need to apply a filter that passes only low frequencies to c(t). Filtering is the topic of the next pre-lab exercises.

C. Low Pass and Band Pass Filter Design

Recall from the filtering lecture notes that, in general, the output of a difference equation y(n) can depend on present and past values of the input x(n) as well as past values of the output. This dependence is captured in the equation below

$$y(n) = a_1 y(n-1) + a_2 y(n-2) + \cdots a_M y(n-M) + b_o x(n) + b_1 x(n-1) + b_2 x(n-2) + \ldots + b_N x(n-N)$$

$$y(n) = \sum_{j=1}^{M} a_i y(n-j) + \sum_{i=0}^{N} b_i x(n-i)$$

In this pre-lab, we will be implementing filters using difference equations whose output y(n) depends only on present and past values of the input x(n), which means the coefficients $a_j = 0$. Recall that these are called FIR filters. This reduces our difference equation to that shown below

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + \ldots + b_N x(n-N)$$

$$y(n) = \sum_{i=0}^{N} b_i x(n-i)$$

Also recall from lecture that the values and number of the b_i coefficients determine the type and performance of the filter implemented by the difference equation. Through the following exercises you we will become familiar with Matlab commands for building filters (*fir1* command); visualizing the frequency response of filters (*freqz* command); and applying filters to signals (*filter* command).

Let's design a low pass filter with order 128 and a cutoff frequency f_c=25 Hz to extract the 10Hz component of

$$x(n) = \sin(2\pi * 10 * n * Ts) + \sin(2\pi * 50 * n * Ts) + \sin(2\pi * 90 * n * Ts)$$

Create the file "filtering.m", and add the following commands to create the signal x(n) and its time-domain plot. You should see a plot such as that in Figure 8.

```
Fs = 1000;                  % Sample Frequency in Hz
t = 0:1/Fs:5;               % Time vector form 0-5 seconds in increments of
                            %  0.001 seconds (eg. 0 0.001, 0.002, 0.003 ...)
x = sin(2*pi*10*t) + sin(2*pi*50*t)+sin(2*pi*90*t);
figure(1)                   % Create Figure 1
plot(t,x)                   % Plot Signal x(n) in Figure 1
title('Time-Domain');       % Add Title Time-Domain to Figure 1
xlabel('Time (sec)');       % Add Time (sec) as x-axis label of Figure 1
ylabel('Amplitude');        % Add Amplitude as y-axis label of Figure 1
axis([0 0.5 -4 4])          % X-axis range [0-0.5] sec and y-axis range [-4 4]
```

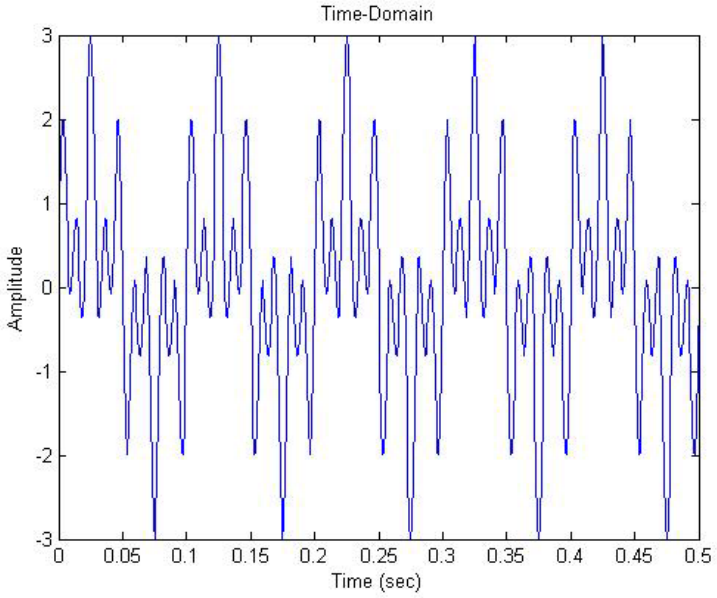


Figure 8

Now let's visualize the magnitude of the spectrum of x(n). Since x(n) is a real signal, the magnitude of its spectrum is an even function (symmetric about 0 Hz). We expect to see a peak at +/-10 Hz, +/- 50 Hz, +/- 90 Hz. Add the following commands to your Matlab file, and rerun it to visualize the magnitude of the spectrum of x(n). Your code should now generate Figure 9.

```
X = fft(x);                 % perform FFT on signal x(n)
N = length(x);              % Make frequency vector f
f = [-N/2:N/2-1]*(Fs/N);    % that spans frequencies from -500 Hz to 500 Hz

figure(2)                   % Create Figure 2
plot(f,abs(fftshift(X(1:N)))) % Plot positive and negative spectrum of x(n)
axis([-100 100 0 3000]);    % X-axis range [-100 100] Hz and y-axis [0 3000]
title('Frequency Domain');  % Add Frequency Domain as title of Figure 2
xlabel('Frequency (Hz)');   % Add Frequency (Hz) as x-axis label of Figure 2
ylabel('Magnitude');        % Add Magnitue as y-axis label of Figure 2
```

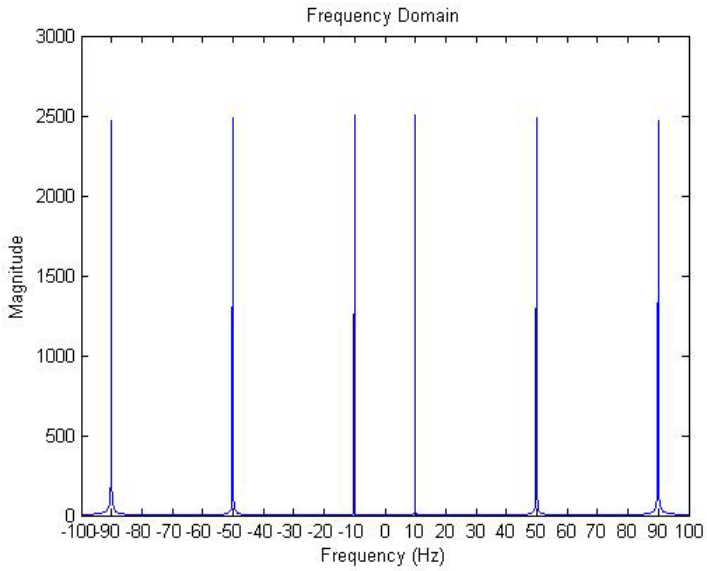


Figure 9

Now let's create the low pass filter that will extract the 10Hz component in x(n). The low pass filter will have an order of 128 and a cutoff frequency f_c=25 Hz. Before getting to the code let's learn about the commands we will be using to create filters and visualize their frequency response.

To create our filter we will use the *fir1* command. To produce a low pass filter using the *fir1* command, use the following format:

B_Coefficients = **fir1**(*Filter_Order*, *Cuttoff_Frequency*, 'low')

The variable *Filter_Order* specifies the order of the filter (in our case 128). The variable *Cuttoff_Frequency* specifies the cutoff frequency of the filter (in our case 25 Hz). The string argument *`low'* specifies to the *fir1* command to create a low pass filter. After executing this command, the variable *B_Coefficients* will be a vector whose elements are the b_i coefficients of a difference equation that implements a low pass filter.

To obtain the frequency response of a filter we will use the *freqz* command. In general, this command computes the frequency response of filter specified by a_i and b_i coefficients; in our case we only have b_i. Use the following format to call the *freqz*

H = **freqz**(*B_Coefficients*, [1], 'whole');

*F=(-256:255)*Fs/512;*

The variable *B_Coefficients* will be a vector of b_i coefficients produced by the *fir1* command. The argument [1] specifies to the *freqz* command that there are no a_i coefficients. The string argument *whole* specifies to the *freqz* command to compute the frequency response for positive and negative frequencies defined in the variable *F*.

Now add the following commands to your Matlab file to create the low pass filter and view its frequency response. You should see a plot like that Figure 10 (we added the Figure 10A to give you practice with viewing spectra on log scales).

```
Filter_Order = 128;              % Set filter order to be 128
Cuttoff_Frequency = 25*(2/Fs);   % Set Cuttoff Frequecy to 25 Hz
%  Following are commands for creating filter and filter frequency response
B_Coefficients= fir1(Filter_Order, Cuttoff_Frequency,'low');
H = freqz(B_Coefficients, [1], 'whole');
F=(-256:255)*Fs/512;

figure(3)                        % Create Figure 3
plot(F,abs(fftshift(H)))         % Plot positive and negative frequencies
axis([-50 50 0 2]);              % X-axis range [-50 50] Hz and y-axis [0 2]
title('Frequency Response');     % Add Frequency Response  as title
xlabel('Frequency (Hz)');        % Add Frequency (Hz) as x-axis label
ylabel('Magnitude');             % Add Magnitude as y-axis label
```

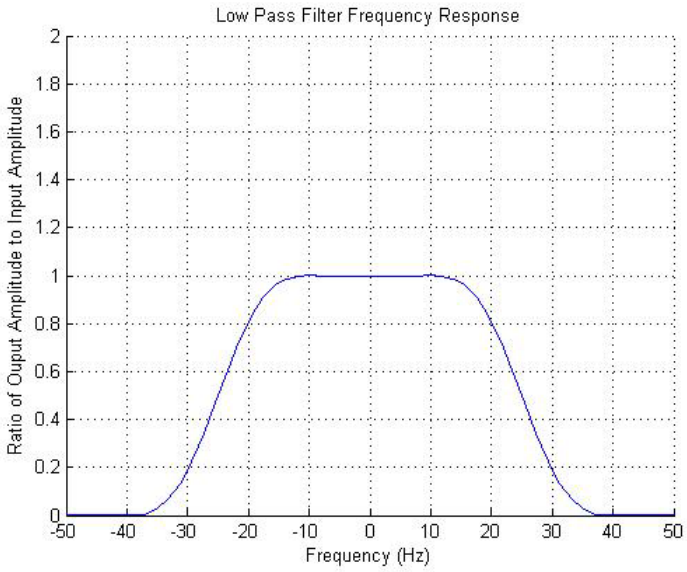


Figure 10

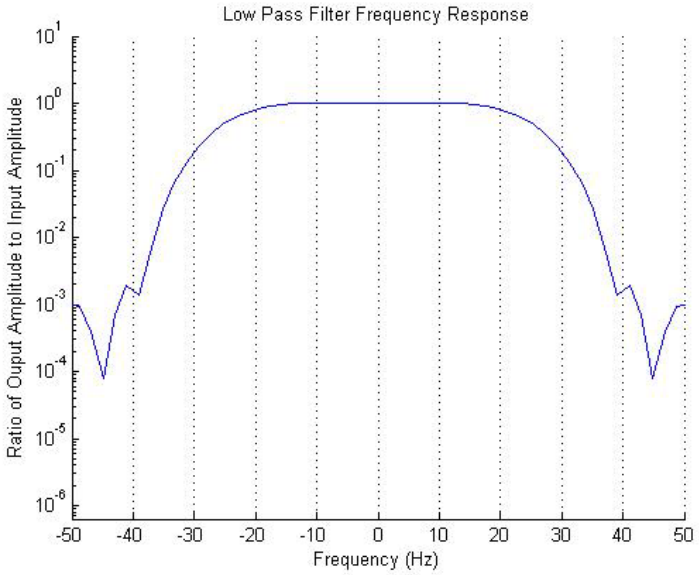


Figure 10A

Now let's apply this low pass filter to the signal x(n). To filter signals we will use the Matlab commands *filter*. Use the following format to call the command *filter*

output_signal = **filter**(*B_Coefficients*, [1], input_*signal*);

The variable *B_Coefficients* is a vector of b_i coefficients (produced by the *fir1* command) of the difference equation representing the filter. The argument [1] specifies to the *filter* command that there are no a_i coefficients in the difference equation. The variable *input_signal* refers to the input signal to be filtered. After executing this command, the variable *output_signal* holds the filtered signal.

Now add the following code to your Matlab file in order to filter the signal x(n). You should see a plot like that in Figure 11.

```
yLPF = filter(B_Coefficients,[1],x);     %Apply the filter to the signal x(n)
figure(4)                                % Create Figure 4
plot((0:length(yLPF)-1)/Fs, yLPF)        % Plot Signal y(n) in Figure 4
title('Time-Domain');                    % Add Title Time-Domain
xlabel('Time (sec)');                    % Add Time (sec) as x-axis label
ylabel('Amplitude');                     % Add Amplitude as y-axis label
axis([.1 1 -1 1])                        % Set x and y axis range
```

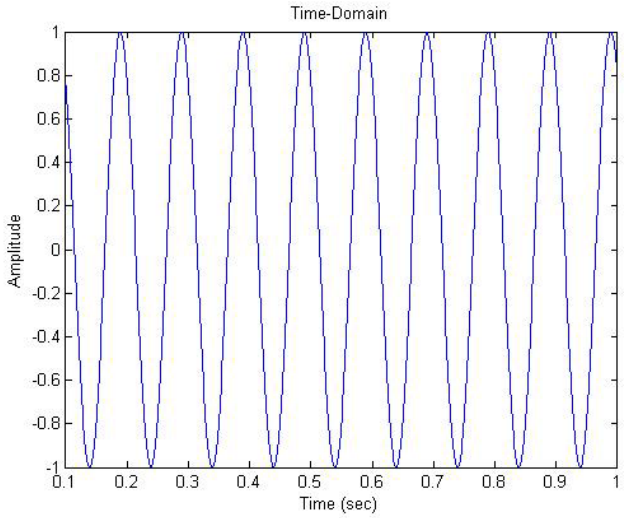


Figure 11

Now let's design a band pass filter to extract the 50Hz frequency component from $x(n) = \sin(2\pi * 10 * n * Ts) + \sin(2\pi * 50 * n * Ts) + \sin(2\pi * 90 * n * Ts)$. The band pass filter will have an order 128 and cutoff frequencies $f_{c,low} = 35$ Hz and $f_{c,high} = 65$ Hz. To design a low pass filter using the *fir1* command use the following format

B_Coefficients = **fir1**(*Filter_Order*, [*Cuttoff_Frequency_Low Cuttoff_Frequency_High]*)

The variable *Filter_Order* specify the order of the filter (in our case 128). The variables *Cuttoff_Frequency_Low* and *Cuttoff_Frequency_High* specify $f_{c,low}$ and $f_{c,high}$ respectively (in our case $f_{c,low} = 35$ Hz and $f_{c,high} = 65$ Hz). After executing this command, the variable *B_Coefficients* will be a vector whose elements are the b_i coefficients of a difference equation that implements a bandpass filter. There is no change in how we call the command *freqz* to evaluate the frequency response of the bandpass filter.

Now add the following commands your Matlab file to create the bandpass filter and view its frequency response. You should see a plot such as that in Figure 12 (we added the Figure 12A to give you practice with viewing spectra on log scales).

```
Filter_Order = 128;                 % Set filter order to be 128
Cuttoff_Frequency_Low = 35*(2/Fs);  % Set f_c,low to 35 Hz
Cuttoff_Frequency_High = 65*(2/Fs); % Set f_c,high = 65 Hz

% Following are commands for creating filter and filter frequency response
B_Coefficients= fir1(Filter_Order, [Cuttoff_Frequency_Low Cuttoff_Frequency_High]);
H = freqz(B_Coefficients, [1], 'whole');
F=(-256:255)*Fs/512;
figure(5)                           % Create Figure 5
```

```
plot(F,abs(fftshift(H)))        % Plot positive and negative spectrum of filter
axis([-100 100 0 2]);           % X-axis range [-100 100] Hz and y-axis [0 2]
title('Frequency Response'); % Add Frequency Response  as title
xlabel('Frequency (Hz)');       % Add Frequency (Hz) as x-axis label
ylabel('Magnitude');            % Add Magnitude as y-axis label
```

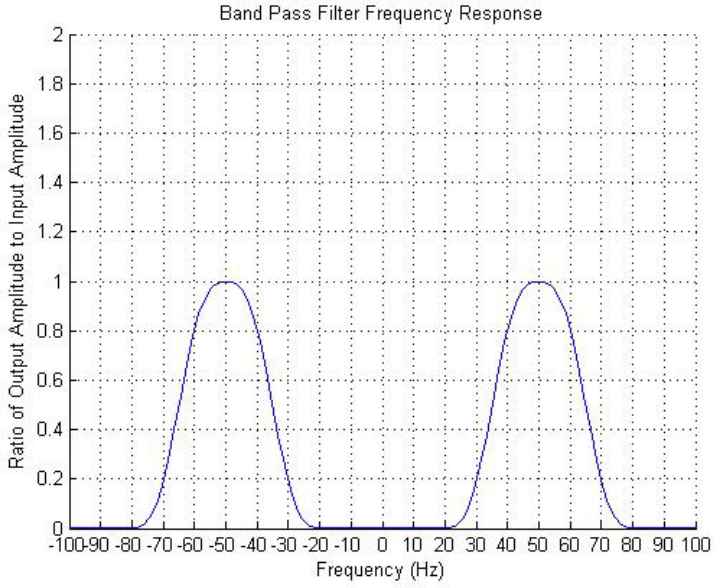


Figure 12

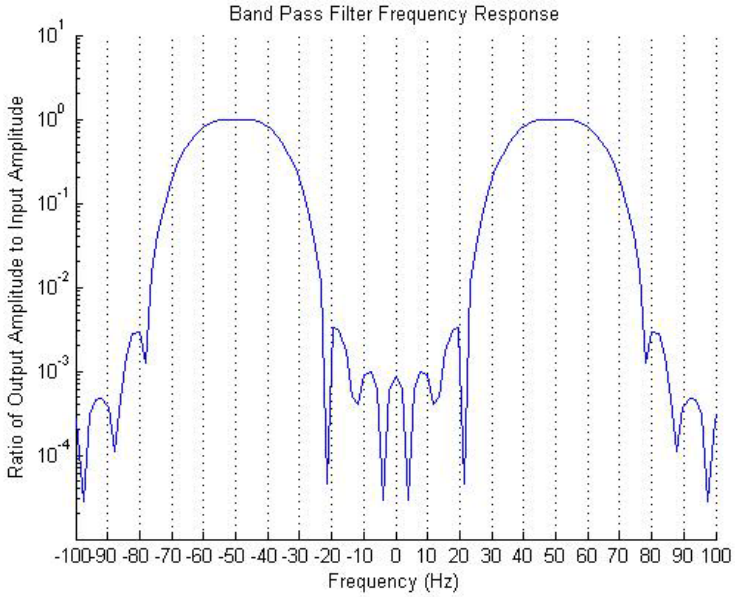


Figure 12A

Now let's apply this bandpass filter to the signal x(n). There is no change to the way in which we invoke the command *filter*. Now add the following code to your Matlab file in order to filter the signal x(n). You should see a plot like that in Figure 13.

```
yHBPF = filter(B_Coefficients,[1],x);      %Apply the filter to the signal x(n)
figure(6)                                  % Create Figure 6
plot((0:length(yHBPF)-1)/Fs, yHBPF)        % Plot Signal y(n) in Figure 6
title('Time-Domain');          % Add Title Time-Domain to Figure 6
xlabel('Time (sec)');          % Add Time (sec) as x-axis label of Figure 6
ylabel('Amplitude');           % Add Amplitude as y-axis label of Figure 6
axis([.1 .3 -1 1])             % X-axis range [0.1 .3] sec and y-axis range [-1 1]
```

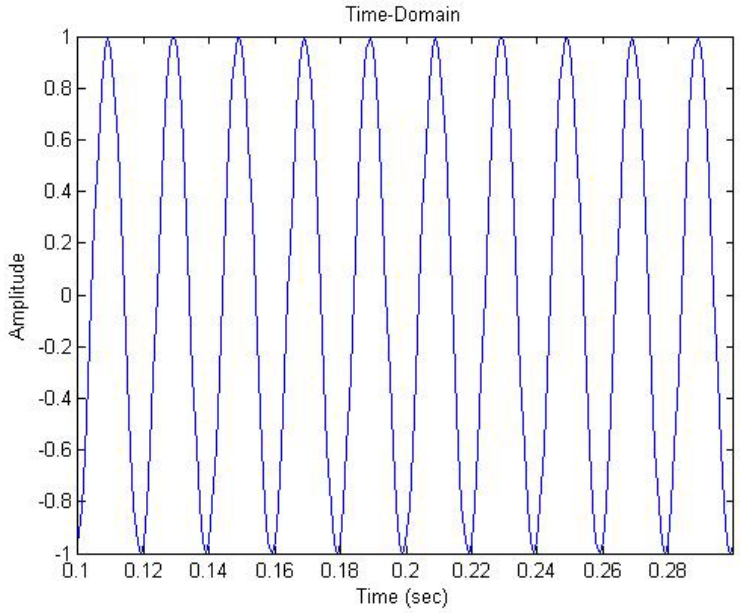


Figure 13

D: Exercises: Filtering and Modulating Signals With Nonzero Bandwidth

So far we have been dealing with the modulation and filtering of sinusoids; these signals have a single frequency component and zero bandwidth. In this exercise you will modulate and filter a voice signal; the voice signal is sampled at Fs = 250kHz, has a bandwidth of 4 kHz, and is centered at 10 kHz. **Complete Tasks 1-6, and for each task save and hand-in the plots you produce.**

***Task 1*:** Obtain the voice signal "voice_sample.mat" by entering the following commands at an Athena workstation:

```
setup 6.082
cd ~/6.082
cp –rf /mit/6.082/Labs/Lab3  .
matlab &
```

Once Matlab starts, run the following commands within the Matlab execution window:

```
cd Lab3
ls
```

You should see voice_sample.mat as one of the files shown. The voice signal is the word "cat" sampled at a sampling frequency Fs = 250kHz and modulated up to a frequency of 10kHz. Plot the spectrum of the voice signal spectrum.

***Task 2*:** Modulate the voice signal so that it is centered at 40kHz and plot the spectrum of the modulated signal.

***Task 3*:** Create a bandpass filter of order 256 and $f_{c,low}$ = 35kHz and $f_{c,high}$ = 45kHz. Plot the frequency response of the bandpass filter.

Task 4: Filter the modulated signal resulting from Task 2 using the bandpass filter, and plot the spectrum of the filtered signal.

***Task 5*:** Demodulate the filtered signal from Task 4 back to DC (0Hz). Plot the spectrum of the demodulated signal.

***Task 6*:** Create a low pass filter of order 256 and f_c = 4.5 kHz. Filter the demodulated signal from Task 5 using the low pass filter. Plot the spectrum of the low pass filtered signal.

Lab Exercises (2pm – 5pm, Wed., February 21, 2007):

A. Objectives

In this lab you will learn about a communication scheme known as frequency division multiplexing. You will then implement, in software, the signal processing operations necessary to receive voice signals transmitted using this communication scheme.

B. Introduction

Frequency Division Multiplexing (FDM) is a communication scheme that allows multiple information signals to be sent simultaneously across a communication channel. FDM assigns to each information signal a frequency band within the communication channel's spectrum. Since the frequency bands are chosen so they don't overlap, the information signals travel through the channel without interference. Figure 1 illustrates the transmission of two information signals using FDM.

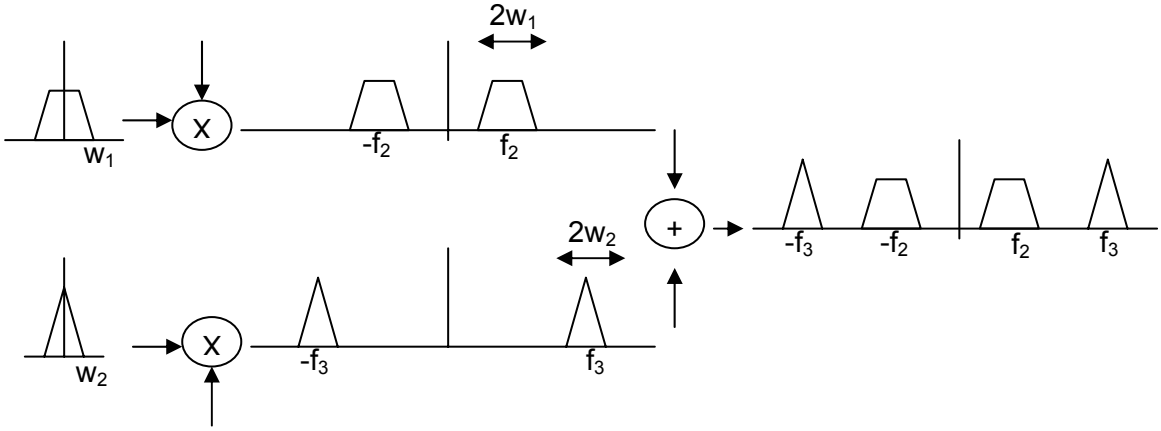


Figure 1

In Figure 1 we have two information signals with spectra shaped as a trapezium and a triangle respectively. To send these information signals simultaneously over the same communication channel using FDM, we *modulate* each of the information signals to a different frequency band. In this example, we modulated the information signal corresponding to the trapezium spectrum to a frequency band with center frequency f_2 and bandwidth w_1; and we modulated the information signal with the triangular spectrum to a frequency band with center f_3 and bandwidth w_2. By adding these two modulated information signals we obtain a *transmission signal* that can carry the spectra of both information signals across the communication channel simultaneously.

At the receiver the transmission signal has to be processed in order to extract the information signal of interest. This processing involves removing the unwanted spectral components from the transmission signal, and then demodulating the spectral component

of interest back to DC (0Hz). As an example, Figure 2 illustrates the processing steps necessary in order to extract the information signal with the trapezium spectrum.

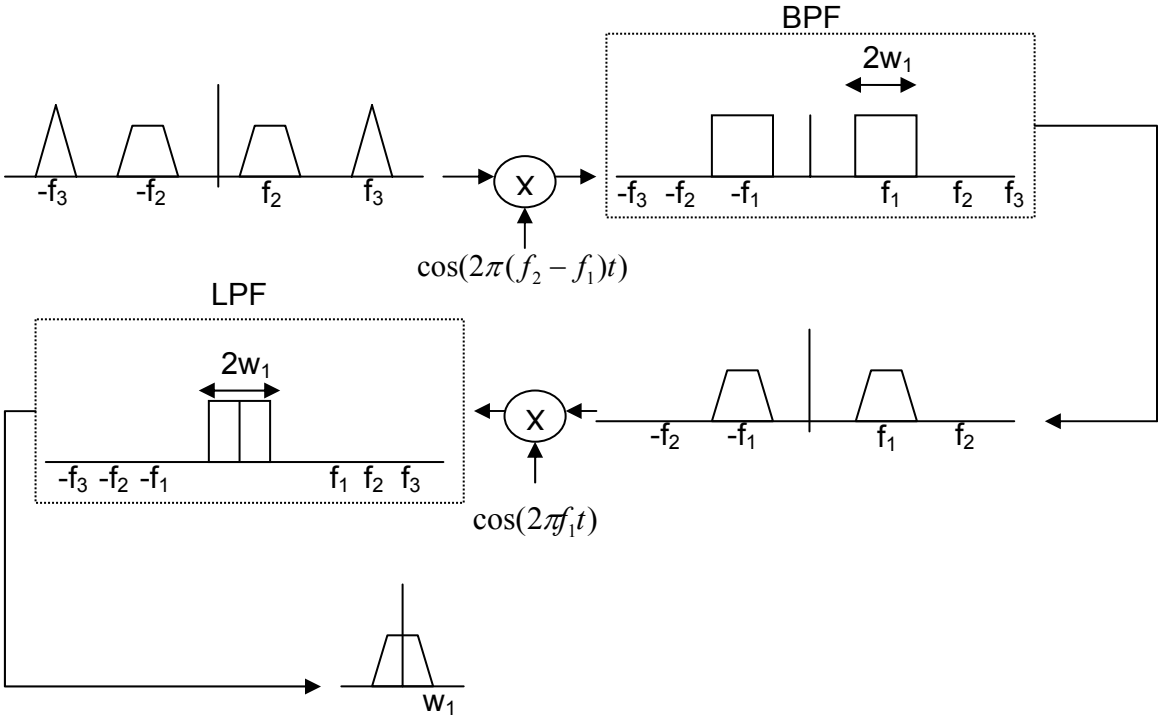


Figure 2

First the transmission signal is modulated so that the spectral component of interest is positioned at the center frequency of a bandpass filter; in this case the trapezium spectrum is shifted so that it is centered at f_1. The bandpass filter keeps only the spectral components of the information signal of interest (the trapezium) and removes the spectral components corresponding to the second information signal (the triangle).

Next, the trapezium spectrum is demodulated to DC (the original center frequency of the information signal spectrum) and low pass filtered in order to remove the higher frequency demodulation products (at $2f_1$ and $-2f_1$). The result is the information signal with the trapezium spectrum.

The same process is used to extract the information signal with the triangular spectrum except now the receiver modulates the received signal by $\cos(2\pi(f_3 - f_1)t)$ as opposed to $\cos(2\pi(f_2 - f_1)t)$. Modulation by the f_3-f_1 frequency cosine centers the triangular spectrum at f_1. Now when the bandpass filter is applied only the spectral components of the information signal of interest (the triangle) are kept. The steps necessary to extract the triangular spectrum are illustrated in Figure 3.

The receiver just discussed is known as the *superheterodyne receiver*. It was developed by Edwin Armstrong in 1918. The main advantage of the superheterodyne receiver is

that it allows the majority of the processing blocks of a radio receiver (blocks such as signal amplifiers not illustrated in Figure 2) to be designed and operated at a narrow, low frequency range ($[f_1\text{-}w_1\ f_1\text{+}w_1]$ in Figure 2) as opposed to operating in a wide, high frequency range ($[f_2\text{-}w_1\ f_3\text{+}w_2]$ in Figure 2). Designing high performance processing blocks that operate in a narrow, low frequency range is far easier than designing processing blocks to operate in a wide, high frequency range.

In summary, the key step in receiving an information signal from a frequency division multiplexed transmission signal is selection of the modulation frequency at the receiver that will center the spectrum of the information signal of interest in the passband of the bandpass filter.

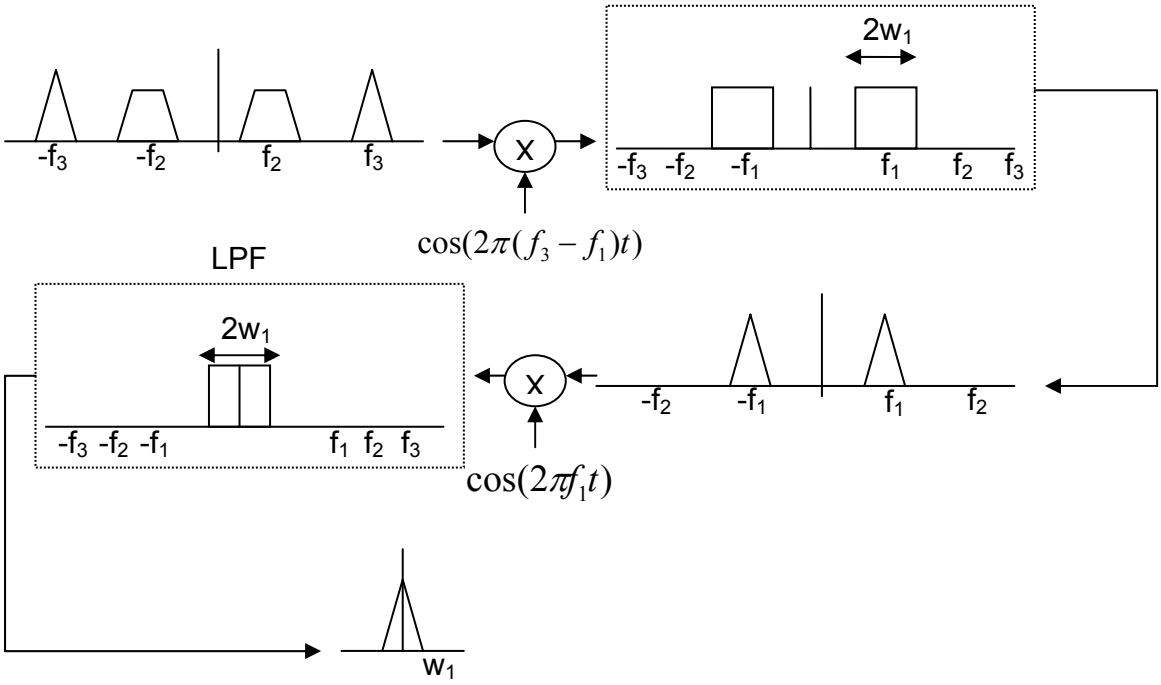


Figure 3

C. Building a FDM Receiver

In this lab a transmission signal carrying the spectra of three different voice signals is being broadcast. The voice signal spectra each have a bandwidth of 4kHz, and they are centered at f_1=30kHz, f_2=70kHz, and f_3=90kHz respectively. The spectrum of the transmission signal is illustrated in Figure 4. Your task is to extract and listen to each of the voice signals embedded in the transmission signal of Figure 4; each voice signal is a word. What is the word formed by concatenating the first letter of the words in Signal 1,2,3 in that order?

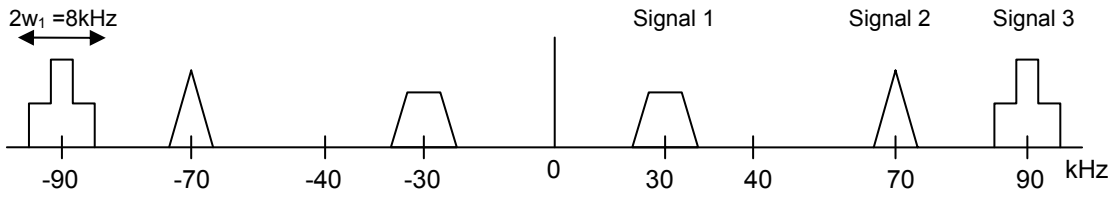


Figure 4

To complete this task, you will fill the blanks in the file "receive_FDM_SuperHet.m" This file is an implementation of the FDM receiving scheme discussed in the introduction and illustrated more compactly in Figure 5. In Figure 5 f_k for $k=1, 2, 3$ refers to the center frequency of voice signal k (e.g. f_3 = 90 kHz and is the center frequency of voice signal 3). Also note that in Figure 5 a correction term Δf has been added to the first modulator. The purpose of this correction term is to allow YOUR receiver to compensate for a frequency offset introduced by the transmitter; that is to allow your receiver to recognize that signal 1, 2, and 3 may not actually be at 30 kHz, 70 kHz, and 90 kHz but rather at 30 kHz + Δf , 70 kHz + Δf, and 90 kHz + Δf. A frequency offset can exist between the transmitter and receiver because the hardware components used to implement the modulators and demodulators in Figures 1 and 2 are not exact; that is both the transmitter and receiver modulators/demodulators operate at frequencies that are different from those we specify. The correction term Δf is somewhere between -100 and 100 Hz in 10 Hz steps (i.e., -10 Hz, 0 Hz, 10 Hz, 20 Hz etc.).

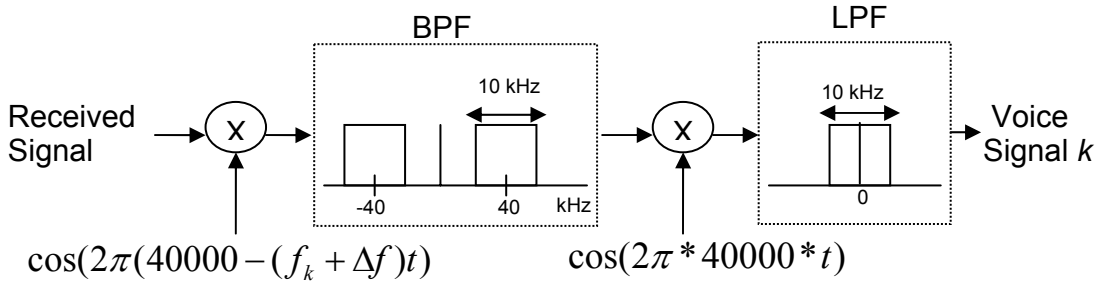


Figure 5

D. Exercises

Starting Matlab and creating a Lab3 Directory

- Choose a lab partner and a PC to work on.
- Log in to your Athena account and then type the following commands within the *same* shell window to start Matlab:

  setup 6.082
  cd 6.082

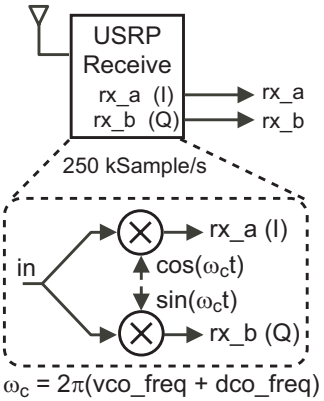
cp –rf /mit/6.082/Labs/Lab3 .
su (password: 4$jk88*)
matlab &

- Within the Matlab execution window, type the command:

cd Lab3

Exercise 1

Within the lab, a transmitter is broadcasting the signals described above in the RF frequency range of around 457 MHz. As shown in the figure below, the USRP board attached to your PC receives this signal and them modulates it down to the baseband frequency range. While not shown in the figure, the USRP also converts these signals to a form in which they can be retrieved by the PC through the USB cable.



Notice that *two* received signals are available at the baseband frequency range, **rx_a** and **rx_b**, corresponding to demodulation by cosine and sine waves, respectively. You can access each of these baseband signals within Matlab through the function:

[rx_a,rx_b] = rf_receive(vco_freq,dco_freq,adc_gain,mixer_gain,record_length);

Each time that **rf_receive()** is run, a new set of samples of **rx_a** and **rx_b** is retrieved..

You will use **rf_receive()** in the exercises below, but for now it is useful to just observe the frequency content of the received signal from the USRP board in `real time'. To do so, examine the received *spectrum* for signal **rx_a** by running the Matlab command

monitor_receive('rx_a');

Print out the spectrum and then label signals 1, 2 and 3 on it. Find a TA and have them check you off on this exercise on the last page of this document.

Exercise 2

The file "receive_FDM_SuperHet.m" is includes comments and blanks (marked using the symbol "??") where you should insert values and commands in order to implement the block diagram in Figure 5. You can edit this file in Matlab by typing:

edit receive_FDM_SuperHet.m;

The contents of the file "receive_FDM_SuperHet.m" are also shown below for convenience; commands are in bold and comments are preceded by the symbol "%". Once you have filled all the blanks, listen to each of the voice signals in the FDM transmission signal and identify the word being transmitted.

```
%%%%%%%%% Obtain received signal from USRP board  %%%%%%%%%%
%%%%%%%%%% DO NOT ALTER THIS SECTION  %%%%%%%%%%%%%

%%% Specification of RF hardware gains, oscillator frequencies
vco_freq = 452e6;
dco_freq = 5.0e6;
adc_gain = 0;
mixer_gain = -10;

%%% Specify number of samples of received data %%%
num_records = 20;
record_length = 50e3;

rx_a = zeros(1,num_records*record_length);
rx_b = zeros(1,num_records*record_length);

%%% receive data from USRP receiver board
for i = 1:num_records
  [rx_a_s,rx_b_s] = rf_receive(vco_freq,dco_freq,adc_gain,mixer_gain,record_length);

  rx_a((1+(i-1)*record_length):(i*record_length)) = rx_a_s;
  rx_b((1+(i-1)*record_length):(i*record_length)) = rx_b_s;
end

[rx_a,rx_b] = lab3_compensate_freq_offset(rx_a,rx_b);

%Sampling Frequency of USRP board equals 250kHz
Fs = 250e3;
% Number of samples we captured from the USRP board
N = length(rx_a);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%% Stuff that you will modify is below %%%%%%%%%%%%
%Specify center frequency (in FDM transmission signal) of voice signal of interest
fsignal = ???;                    % Can be 30kHz, 70kHz, or 90kHz plus offset freqeuncy

%Make the bandpass filter
Filter_Order = ???;
Cutoff_Frequency_Low  = ???*(2/Fs);
Cutoff_Frequency_High = ???*(2/Fs);
B_Coefficients_BPF = ???;

%Make the low pass filter
Filter_Order = ???;
Cutoff_Frequency = ???*(2/Fs);
B_Coefficients_LPF = ??? ;

%Make the cosine to modulate the voice signal to 40kHz
t = 0:1/Fs:(N-1)/Fs;
f1 = ????;
f2 = ????;
modulation_Signal_1 = cos(???*t);

%Modulate the received signal to the fixed bandpass filter
rx_demod = rx_a .* ????;

%Apply the bandpass filter
rx_bpf =????;

%Make the cosine to modulate the bandpass filtered signal to DC
f3 = ???;
modulation_Signal_2 = cos(???*t);

%Modulate the bandpass filtered signal to DC
rx_dc = ??? .* ????;

%Apply the low pass filter to remove higher frequency demodulation products
rx_dc_lpf = ????;

%Listen to the recieved signal
soundsc_linux(rx_dc_lpf, Fs);
```

Once you have completed this exercise, save your file and then find a TA to have them check you off on this exercise on the last page of this document.

Exercise 3

In this section of the lab we will use an alternate FDM receiver architecture shown in Figure 6 to extract each of the voice signals in Figure 4. To receive voice signal *k* using this architecture, one chooses the demodulator frequency so that it centers the spectrum of voice signal *k* at DC. Next, the demodulated signal is low pass filtered so that only the spectrum of voice signal *k* appears at the output. The receiver architecture outlined in Figure 6 is known as the *homodyne receiver*. It was developed by a team of British

scientists in 1932. The main advantage of the homodyne receiver is its low complexity (easier and cheaper to build) and low power consumption.

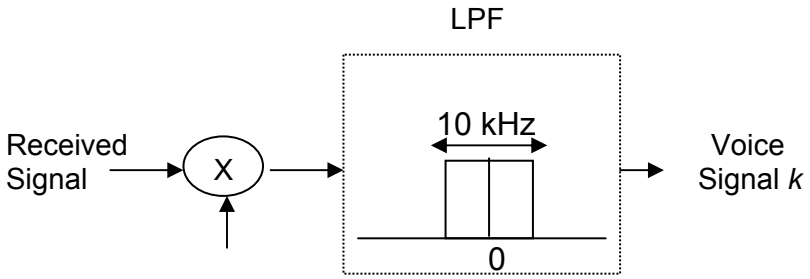


Figure 6

Edit your superheterodyne file to build the homodyne receiver shown in Figure 6. Once again you must include the correction term Δf when specifying the frequency of the homodyne receiver demodulator; which will be the same value as determined in the previous exercise. Listen to each of the voice signals in the FDM transmission signal and identify the word being transmitted.

Once you have completed this exercise, save your file and then find a TA to have them check you off on this exercise on the last page of this document.

Exercise 4

Add a large frequency mismatch between your receiver and the transmitter by adding to the modulation signal an additional frequency offset of x = ±500 Hz and ±1000

$$\cos(2\pi(f_k + \Delta f + x)t)$$

Hz and listen to the results.

Once you have completed this exercise, save your file and then find a TA to have them check you off on this exercise on the last page of this document.

Post-Lab Exercises (Due Friday, February 23)

1) When trying to recover a signal through demodulation, what is the impact of having a frequency offset? Explain by using the Fourier Transform in the form of frequency domain pictures. A good example would be to see the impact of having a frequency offset when trying to demodulate a higher frequency cosine wave.

2) When filtering a demodulated signal, what are the tradeoffs in making the filter bandwidth too low or too high? In other words, what are the negative effects of having the filter bandwidth too low, and what are the negative effects when it is too high?

3) Besides frequency offset, what other non-idealities did you notice when performing demodulation?

4) Consider the filterbank shown in Figure 1. Let the input to this filterbank be the FDM transmission signal shown in Figure 2; this is a similar set signals to what you received in lab.

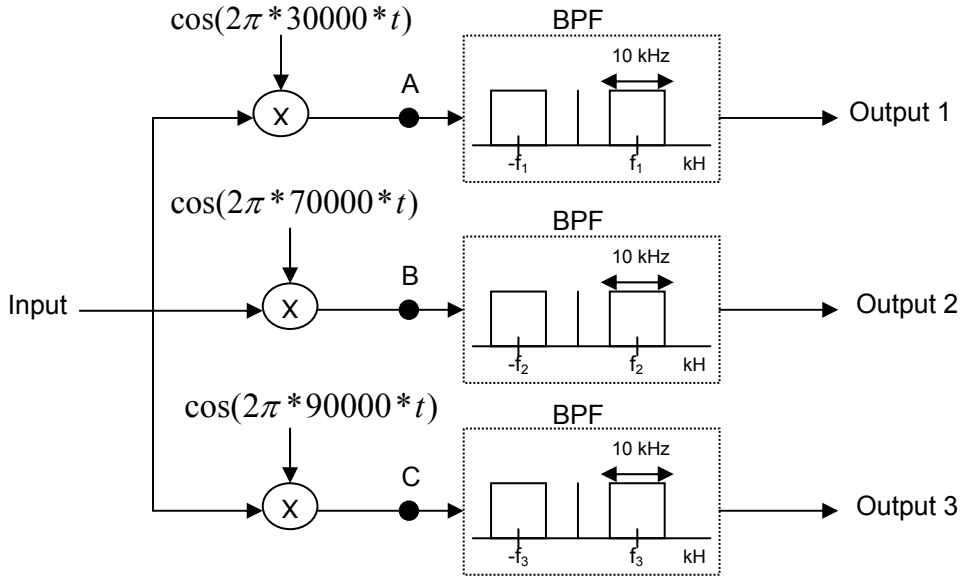


Figure 1

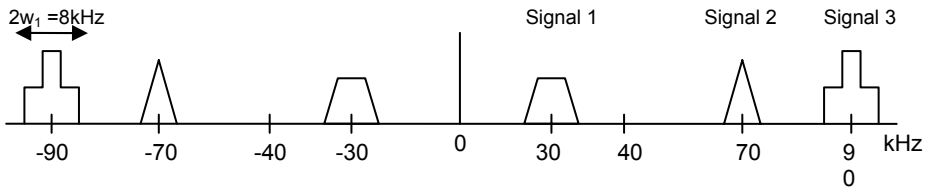


Figure 2

a) Sketch the spectrum of the input signal at points A, B, and C of the filterbank.
b) Determine the center frequencies f_1, f_2, and f_3 of the filterbank such that outputs 1, 2, and 3 all have the triangular spectrum of Signal 2.

Check-off for Lab 3

_____ _____

Student Name Partner Name

Check-off for Exercise 1

_____

TA Signature

Check-off for Exercise 2

_____

TA Signature

Check-off for Exercise 3

_____

TA Signature

Check-off for Exercise 4

_____

TA Signature