070928/ Thomas Munther
Halmstad University
School of Information Technology, Computer Science and Electrical Engineering

Matlab Exercise IV

In this laboratory we will focus on digital filters like FIR and IIR and also mention window functions. The goal is that you should be able to make some simple digital filters using Matlab.

FIR filter:

Matlab Signal processing Toolbox has two types of routines fir1 and fir2. They can both be used to design FIR filters based on the window methods.
Fir2 designs frequency sampling-based digital FIR filters with arbitrarily shaped frequency response.

```
b = fir2(n,f,m)      % returns row vector b containing the n+1 coefficients of an order n
                     % FIR filter.
```

The frequency-magnitude characteristics of this filter match those given by vectors f and m: f is a vector of frequency points in the range from 0 to 1, where 1 corresponds to the Nyquist frequency. The first point of f must be 0 and the last point 1. The frequency points must be in increasing order. m is a vector containing the desired magnitude response at the points specified in f.
f and m must be the same length.

Fir1 uses the Hamming window as default, but we can of course use other window types. The cutoff frequency is normalized and is between [0 1]. One corresponds to one-half the sampling rate.
When you have a discrete time signal of infinite length in order to calculate the DFT you truncate the signal to N samples. This is achieved by multiplying x[n] with a window function. The window function can have very different behaviour, but they window function must be non-zero in the interval 0 to N-1. The window functions affect two interesting properties in the spectrum, the width of the mainlobe and the attenuation of the sidelobes. In general if the width of the mainlobe is small then the sidelobe attenuation decreases. This is shown in the table below. The main purpose of the window is to damp out the effects of Gibbs phenomena. A result from truncation of an infinite series.

Window type	Width of mainlobe	sidelobe [dB]
Rectangular (rectwin)	2 π/N	-13
Bartlett	4 π/N	-27
Blackman	6 π/N	-58

If you want to separate two adjacent frequency components then the most suitable is the rectangular window.
Let's start with some FIR filter construction:
We will start up by using the fir1 command.

```
>>M=10;
```

>>wc=0.35;
>> b=fir1(M,wc) **%** M:th order FIR filter with normalized cutoff frequency.
b =
-0.0036 -0.0127 -0.0066 0.0881 0.2594 0.3510 0.2594 0.0881 -0.0066
-0.0127 -0.0036
% if M=10, then we have M+1 coefficients in the polynomial.

This means that we have a impulse response of the FIR filter like:

$h[n]=-0.0036*\delta[n] - 0.0127*\delta[n-1]-0.0066*\delta[n-2] + 0.0881*\delta[n-3] +0.2594*\delta[n-4]$
$+0.3510*\delta[n-5] + 0.2594*\delta[n-6] + 0.0881*\delta[n-7] - 0.0066*\delta[n-8]$
$- 0.0127*\delta[n-9] - 0.0036*\delta[n-10]$

Let's see how the frequency response look like:

>> [H,w]=freqz(b,1,512) % calculates the frequency response in 512 points.

>> HdB=20*log10(abs(H)); % conversion of magnitude response to decibel.
>> plot(w,HdB), grid % plots magnitude response versus frequency.
>> ylabel('dB');
>> xlabel('rad/sec');
>> title('Magnitude response')

Notice that the plot ends at approximately 3. The plot is carried out until 3.14. This means half the sampling frequency.

See figure 1 !

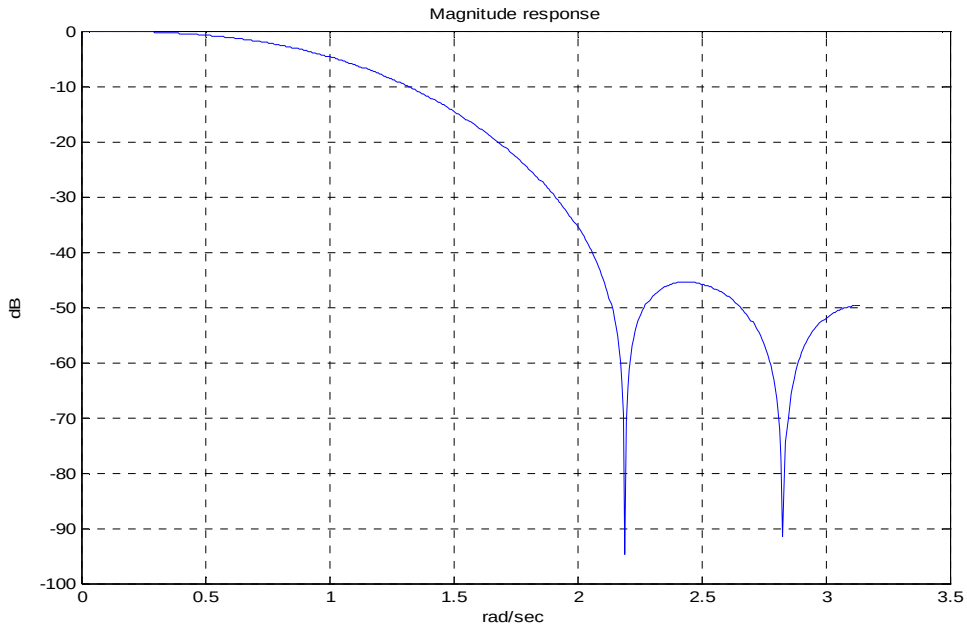


Figure 1

Repeat the commands and also use a window function .
Try the rectangular window function !.
>> b=fir1(M,wc,rectwin(M+1))

Don’t forget to repeat the other commands and then we will end up with figure 2.
Also try a Hamming window.

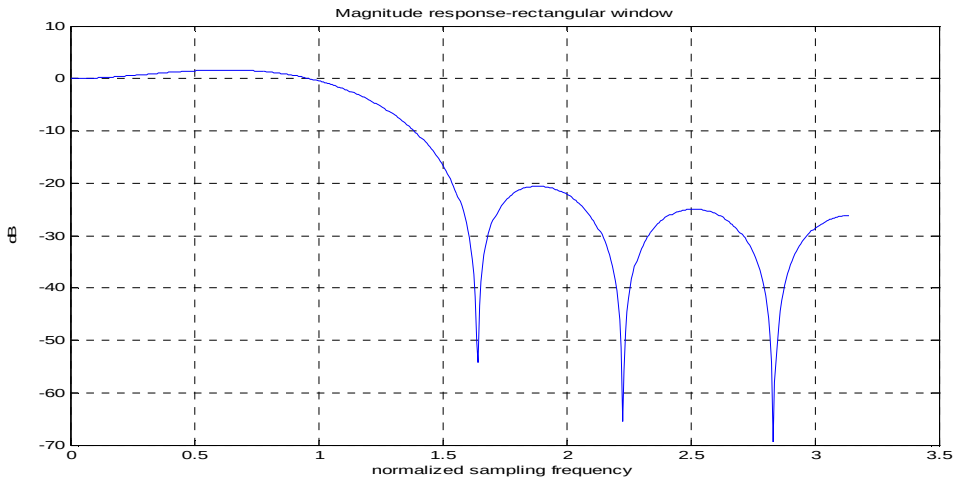


Figure 2

Whats the difference ?

If we instead change the order M=100 and the cutoff frequency is still 0.4. How will it then look like? See figure 3!

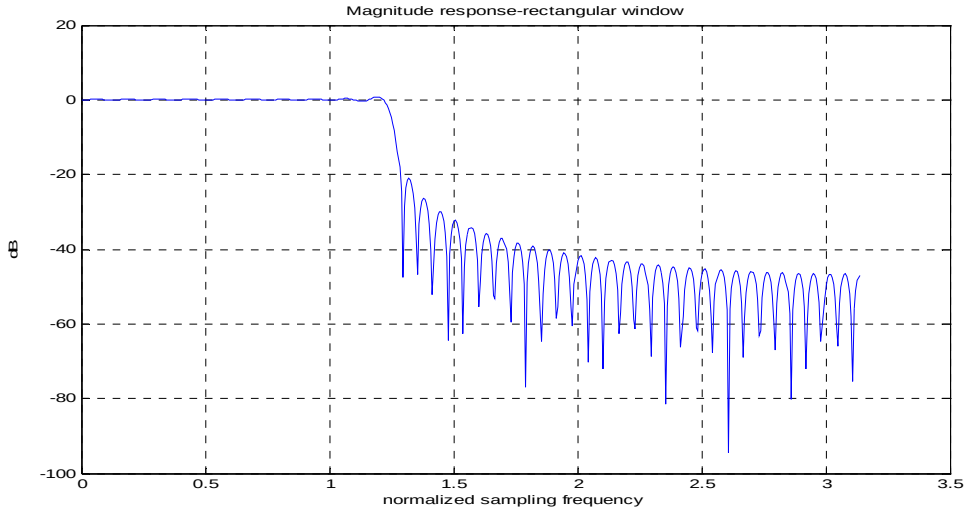


Figure 3

As you can see. The mainlobe (0 to 1.25) is wide and the sidelobes are greatly reduced relatively to the mainlobe. If we alter the order to M=100 for the Hamming window as well.Then see below in figure 4 for the change. Sidelobes are even more suppressed than before, but the disadvantage is that the mainlobe is slightly wider also.
This means that the Hamming window reduces oscillations in the frequency response of a FIR filter. However the price to be paid is larger transition band between passband and stopband. There is always benefits and disadvantages for every window function.
There are many other different window functions to investigate like: Kaiser, Hann, Bartlett, Tukey, Blackmann and so on.

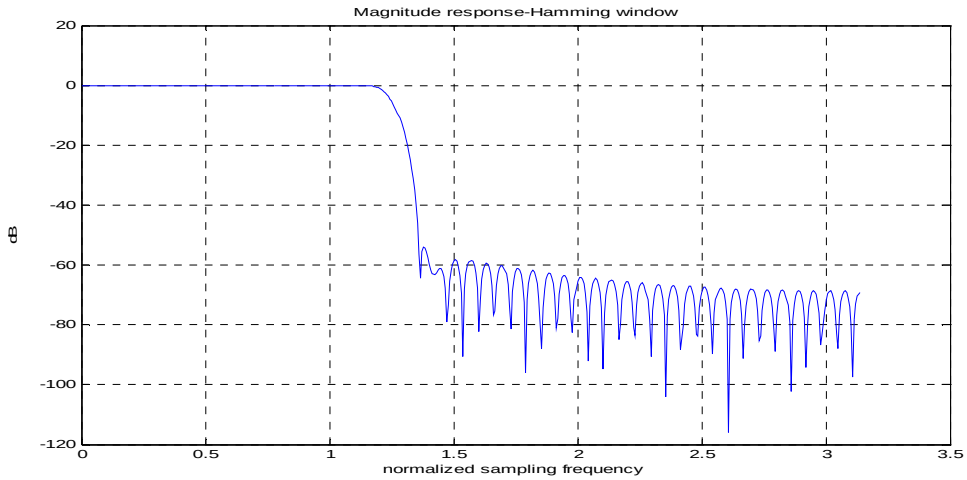


Figure 4

Before you continue with FIR filter construction. Do the following simple example which illustrate the window effect on a sampled sine wave signal.
>> t=0:0.01:10;
>> y=sin(40*t);
If you write help window you will find some of the windows to be used.
>> psd(y,512,100,hamming(512)) % plots power spectral density versus frequency.

The second argument gives number of calculation points to be used in the FFT algorithm. Then the third and fourth argument is the sampling frequency 1/100 and window function.
Try also the following.
>> psd(y,512,100, tukeywin(512))
>> psd(y,512,100,hanning(512))
I

We will now return to the construction of FIR filter. Design a 48: th order FIR bandpass filter. With the normalized cutoff frequencies ranging from 0.35 to 0.65 of half the sampling frequency (π). Notice the midband frequency ! It seems to be 0.5 and also notice the linear phase response in the passband. See figure 5!

>> b = fir1(48,[0.35 0.65]); % creates the fir coefficients using Hamming window.
>> freqz(b,1,512) % creates magnitude and phase response in figure 5.

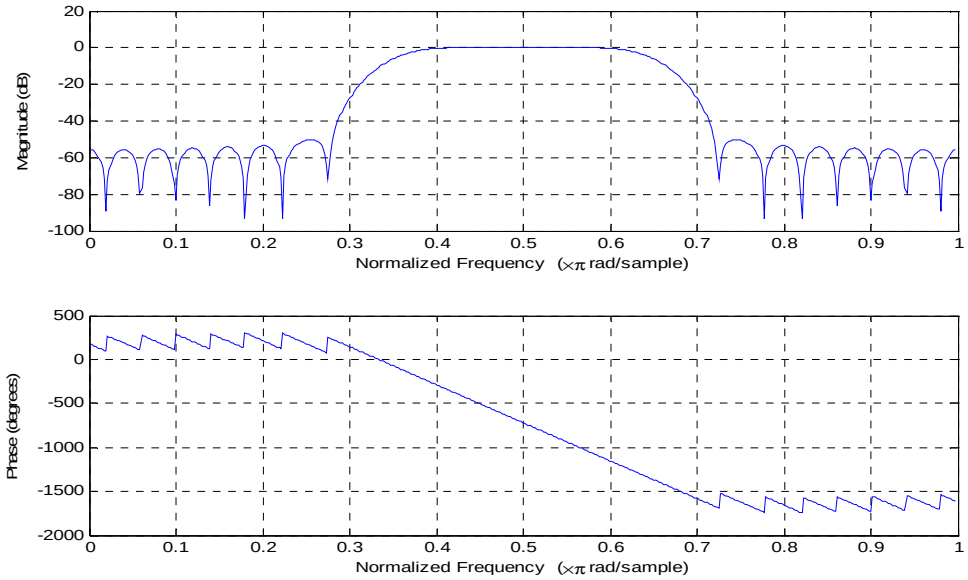
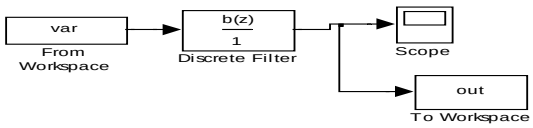


Figure 5

Suppose we now load a datafile containing some lovely bird song. The name of the file is chirp.mat . The file has most of its power above $F_s/4$.

>> load chirp % it gives two output arguments a vector y and a variable with the
% sampling frequency F_s.Here F_s is 8192 Hz.
>> plot(y) % look upon the vector.
>> soundsc(y) % listen to the vector.

We should try to design a highpass FIR filter and then let the vector pass through the filter. Listen to the ouput signal.
Ok, first we try with the bandstop filter.
>> b = fir1(48,[0.35 0.65]);
We will try to export this filter to a simulink model file.



Where can you find the different blocks ?

Simulink-> Sources-> From Workspace
Simulink-> Sinks-> To Workspace
Simulink-> Discrete-> Discrete Filter
Set the sample time to 1/8192 sec both in Discrete Filter as well as in From Workspace.
I have renamed the variables to var and out.
The var is really a matrix and is created like:
>> t=0:1/8192: (13129-1)/8192;
>> var=[t' y];
Now run the simulink model file. Before you run the mdl-file enter Simulation-> Configuration Parameters and change **Solver:** Discrete and **Type:** Fixed Step
Alter the format in the To Workspace block to array.
After the simulation return to matlab command window and write:
>> soundsc(out) % play the bird who was filtered through a bandpass filter.
Compare it to the original signal !
>> soundsc(y)

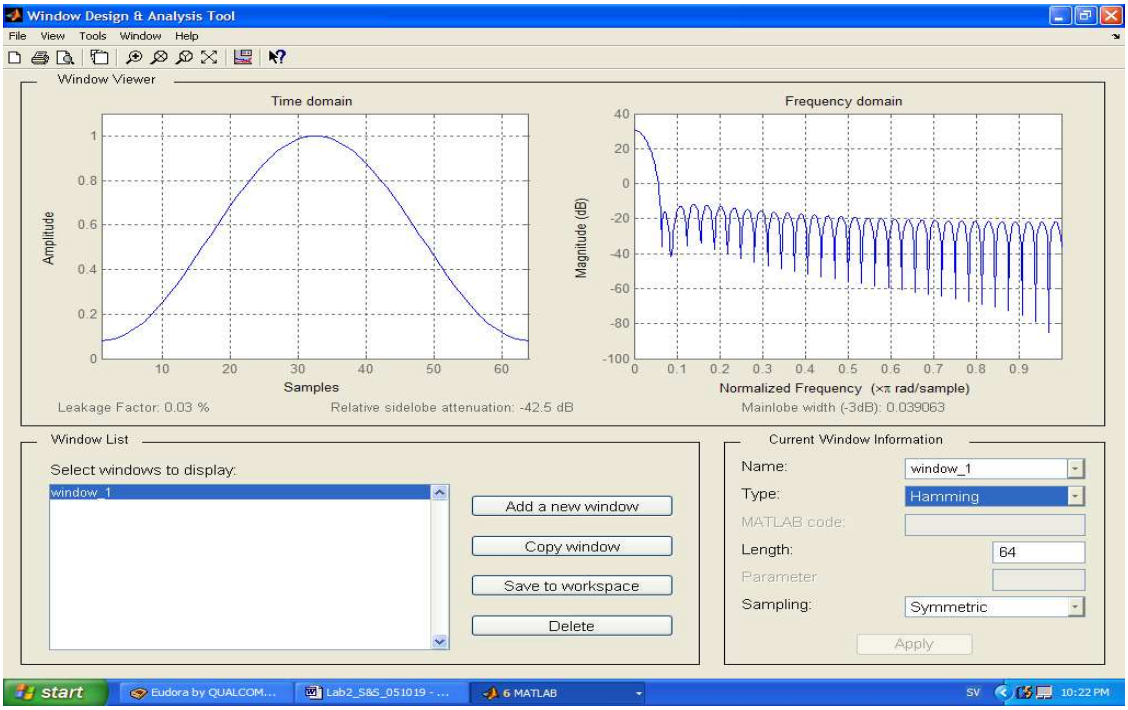
I think I can hear the difference. Can you ?

Try to change your filter to a highpass filter and repeat the previous steps and see if you can change the frequency content in the bird file.
I choose the cutoff frequency to 0.5 there is a significant difference between the original and filtered bird song.

Experiment with different sampling frequencies to hear the difference !

Window functions: if we are trying to design a FIR filter and we use ideal filter as a desired filter. This will correspond to an infinite impulse response. This is of course not possible to implement in a computer, so therefore one multiplies with the window function. We have truncated the impulse response to make it finite. This truncation can be done in different ways actually. We have a great number of window functions to select from as mentioned earlier.

In matlab you can start the interactive graphical user interface (GUI) for window function design. There is at least 15 different window functions and you can yourself build your own window functions as well.
>> wintool % starts the GUI for window function design



IIR filter: We are going to use a butterworthfilter and convert into the discrete time filter. Below can you see the help offered by matlab.

BUTTER Butterworth digital and analog filter design.

[B,A] = BUTTER(N,Wn) designs an Nth order lowpass digital Butterworth filter and returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator). The coefficients are listed in descending powers of z. The cutoff frequency Wn must be 0.0 < Wn < 1.0, with 1.0 corresponding to half the sample rate.

In this example will we try to create a third order IIR filter with a normalized frequency 0.3 .

```
>> [b,a]=butter(3,0.3);
>> [H,w]=freqz(b,a,512);
>> HdB=20*log10(abs(H));
>> subplot(2,1,1)
>> plot(w,HdB)
>> title('magnitude of IIR filter M=3')
>> phi=angle(H)*180/pi;
>> subplot(2,1,2)
>> plot(w,phi)
>> title('phase response IIR filter')
```

When you look upon the plot you can't ignore the jump in the phase response. We did put a cutoff frequency to 0.3 (of half the sampling frequency). This seems to be the same

location as the phase jump. The x-axis is showing normalized frequency from 0 to half the sampling frequency.
Read the magnitude response at the w_c ! It should of course be -3dB.

Please notice, everything you have learned about Bode plots concerning asymptotes and the total phase response is not valid for a discrete time system. Conclusions regarding the slope in magnitude response due to the order can no longer be made. See figure 6!

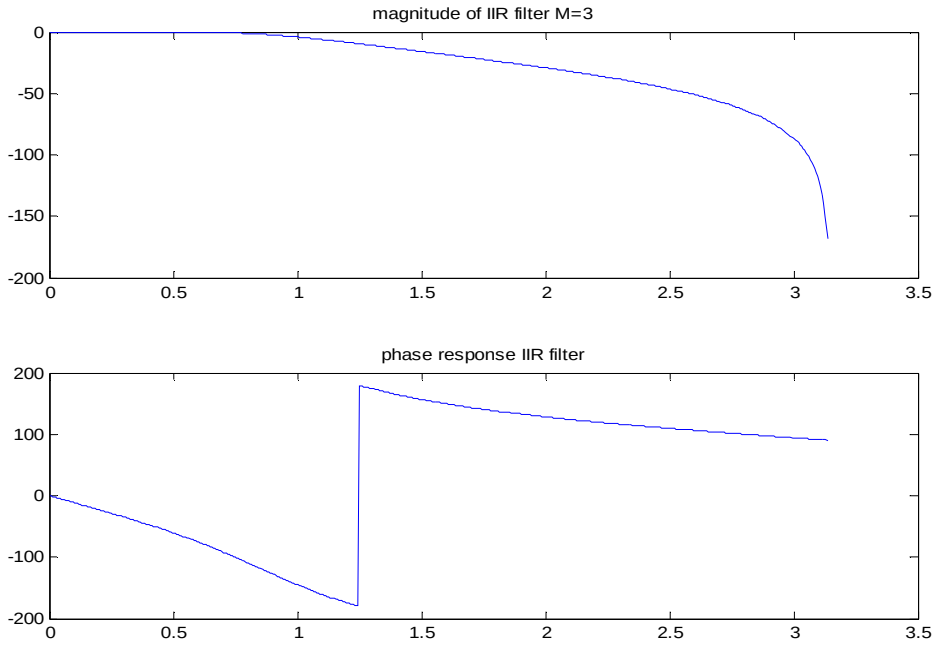


Figure 6

How does the filter look like. Let's use the transfer function. We can use the same command for both continuous and discrete time systems. The only difference is the third argument.
It contains the sampling frequency and therefore matlab assumes it should present the discrete transfer function.

```
>> HH=tf(b,a,1)
```

Transfer function:

$$\frac{0.04953\,z^3 + 0.1486\,z^2 + 0.1486\,z + 0.04953}{z^3 - 1.162\,z^2 + 0.6959\,z - 0.1378}$$

If we send in a pulse train through this IIR filter. Knowing that the phase response is hardly linear. How will the output look like ?
In the model below I've inserted two transfer functions one with a IIR filter and the other containing a FIR filter. The pulse generator is to be found in the sublibrary **Simulink->Sources .** Due to the fact I have changed the Pulse Type in the block from Time to Sample based it also effects the outside off the block.
I have also created a FIR filter of third order for comparison. They should also have the same cutoff frequency.
>> bfir=fir1(3,0.3);
Before we run the simulink model file: **Change Simulation-> Configuration Parameters** to **Type:** Fixed Step and **Solver:** discrete. Simulation Time 10 sec.
Set the sample time to 0.1 sec both in Pulse Generator and in both Discrete Filters.
Finally also adjust the Period (Pulse Generator) to 20 samples. Start simulation !

In the model window above you can see there is a clear difference between the filters. Are you surprised ? You should not be !
Here we can see the impact both regarding the magnitude as well as the phase response.
What happens if we lower the cutoff frequency in the FIR and IIR filter ?
If we make the cutoff frequency higher in both filters. Does it have any effect on the output ?

It could also be interesting to see the impulse responses from both the FIR and IIR filter.
>> impz(bfir,1,10); % Gives the first 10 samples from the impulse response of the FIR
% filter.
>> impz(b,a,10); % Impulse response from the IIR filter.
Please notice that for FIR filter of order 3, the length of the filter is 4, but for the IIR of the same order the response has a length far higher.

Problems that should be handed in before the exam (if you want bonus points). The exercises must be solved to pass the course.

1. Design a bandpass filter with order 50. The passband should lie between 0.25π and 0.75π. Use Hamming, Rectwin and Blackman windows. Give me a m-file that produces a figure window splitted into three subplots. They should contain magnitude responses from these three different filters using separate window types. Which one of these three window functions has the shortest transition band?

2. Design a recursive filter with the following requirements:passband between normalized frequency 0 and 0.3π. The maximum allowed ripple is 0.1 dB. Stopband between 0.5π and π. Attenuation in this frequency range minimum 40 dB. Make a m-file !

3. Load the datafile train.mat . This contains a sound coming from a train whistle. If you load the file it will give you a vector y and a constant F_s with the sampling frequency in Hz.
   You should create a m-file which produces a plot over the frequency content.
   Read the three first pages Matlab Exercise 4 to solve this assignment.
   Further, also construct three different FIR filters of LP, BP and HP-type with the following cutoff frequencies 2.5kHz, [2.5 2.6] kHz and 2.5kHz. Use order 10 !
   Play the resulting train whistle from each filter !
   Guidance:
   >> conv(h, y) %gives the filtered output from the FIR filter, if h is a vector that contains the coefficients in a FIR polynomial or if you prefer the impulse response. We can also consider it to be the convolution between impulse response and the data signal.
   Separate the display of the sounds by pause commands !

4. Identify the phone number in the given data file: **Numbers.wav**
   The phone number is 5 numbers long. Each number consists of two frequencies. For example number one has frequencies 697 and 1209 Hz.
   The file can be found on my homepage.

		Frekvens 2 [Hz]	
Frekvens 1 [Hz]	**1209**	**1336**	**1477**
697	**1**	**2**	**3**
770	**4**	**5**	**6**
852	**7**	**8**	**9**
941	*****	**0**	#

   Hand-in a m-file that produces this phone number. Like 5 separate frequency plots. Where I can more or less see the number(frequencies) myself.
   Guidance: [Y,Fs]=wavread('Siffror.wav'); % gives a sampled datavector & sampling frequency. The command psd can also be useful. It gives the the power spectrum density

>> psd(y(1:150000,1),512,Fs,Hanning(512)) % arguments are: datavector, points of fft ,sampling frequency and window function.