School of Engineering and Technology

Course:	Final Year of Beng/CEE Communications & Electronic Engineering
Names of students:	Ignacio Salan
Assignment title:	Digital Signal Processing Assignment.
Supervisor:	Mr. G. Sexton.
Date:	06-05-2003

TABLE OF CONTENTS

1 INTRODUCTION

This assignment can be divided in two main parts the first one corresponds to the Matlab session in the lab, and the second one is the design and realisation of a digital filter.

The realisation of a digital filter is carried out with a TI TMS320C5402 DSP platform of Texas instrument. All the design and calculations are made in Matlab, so all the files are shown and the simulation.

2 MATLAB LAB

The following Matlab code creates a signal with a sampling frequency of 10000 Hz, containing three sine waves of frequencies 100, 200 and 500 Hz respectively and some additional white noise.

```
fs=1e4; f=[100;200;500]; n=0:1023;
a=[1,2,0.5,0.1];
s=sin(f/fs*n*2*pi);    %making the signals in function of the sampling
frequency
s=diag(a)*[s;randn(1,1024)];   %Adding noise to the signals and the amplitudes
s2=sum(s);
figure(1)
subplot(2,1,1);
plot(n,s);
title('The four signals');
xlabel('sample');
ylabel('Amplitude');
subplot(2,1,2);
plot(n,s2);
title('Signal with noise');
xlabel('sample');
ylabel('Amplitude');
t=n./fs;
figure(2)
subplot(2,1,1)
plot(t,s)
title('The four signals');
xlabel('time');
ylabel('Amplitude');
subplot(2,1,2)
plot(t,s2);
title('signal with noise');
xlabel('time');
```

```
ylabel('Amplitude');
```

The previous code creates the signal and plot both signal ‘s’ and ‘s2’, using the transpose operator (‘) where necessary. So now are displayed the different figures,

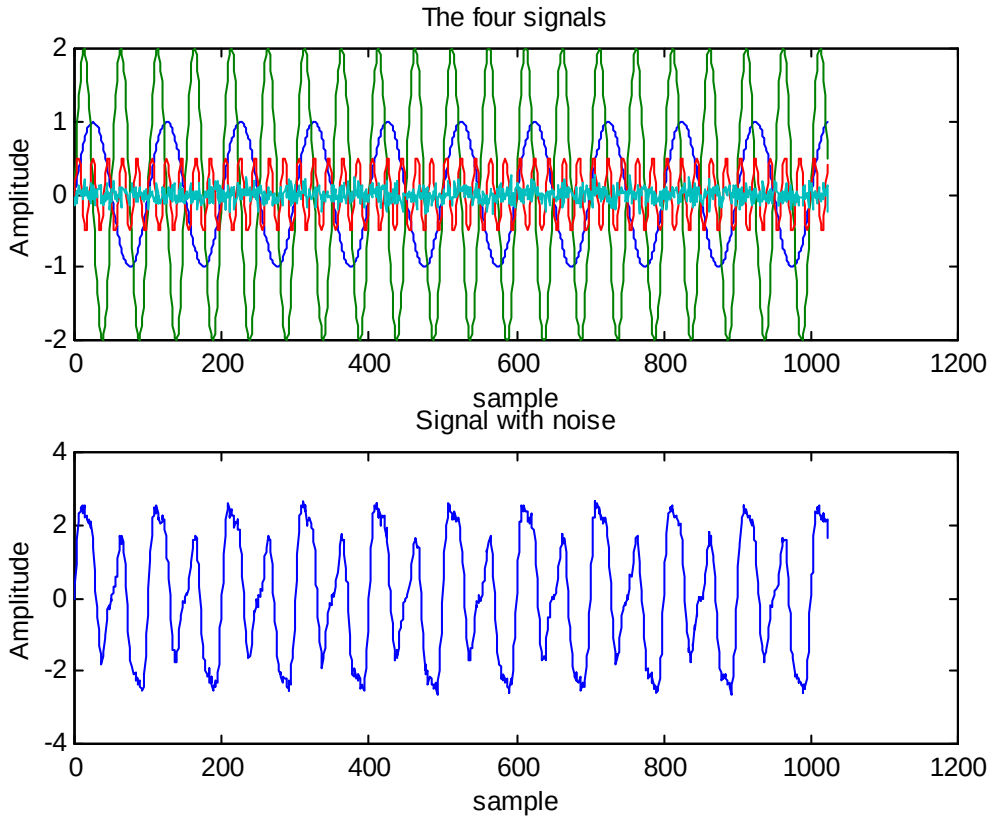


Figure 1. *Matlab simulation of the previous code.*

So as it can be observed in the graph and in the code the amplitude of the four signal that appears in the upper half of figure 1, are the following the green signal has an amplitude of 2, the blue one of 1, the red of 0.5 and the last one has an amplitude of 0.1.

All the signals have been plotted versus the sampling rate; next figure shows the signal versus time where it is demonstrated that it is possible.

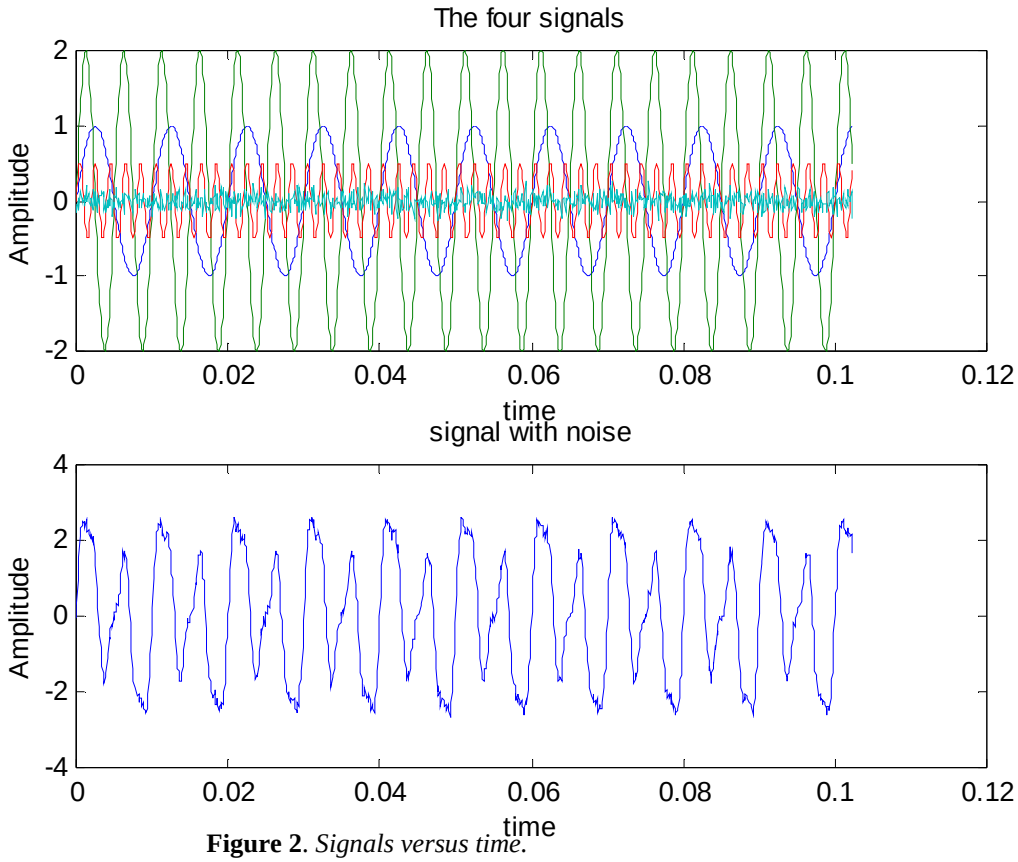


Figure 2. *Signals versus time.*

Now the code and the next figure next figure illustrates a signal with three sine waves of frequencies 100, 250, 400 Hz respectively and amplitudes (rms) of 1, 2, and 3 V. The sampling frequency to 1000 Hz and it is added noise to produce a signal to noise ratio of 20 dB for the 100 Hz signal.

```
fs=1e3; f=[100;250;400]; n=0:1023;
a=[1, 2, 3, 0.1];
s=sin(f/fs*n*2*pi);    %making the signals in function of the sampling frequency
s=diag(a)*[s;randn(1,1024)];   %Adding noise to the signals and the amplitudes
s2=sum(s);
figure(1)
subplot(2,1,1);
plot(n,s);
title('The four signals');
xlabel('sample');
ylabel('Amplitude');
subplot(2,1,2);
plot(n,s2);
```

```
title('Signal with noise');
xlabel('sample');
ylabel('Amplitude');
```

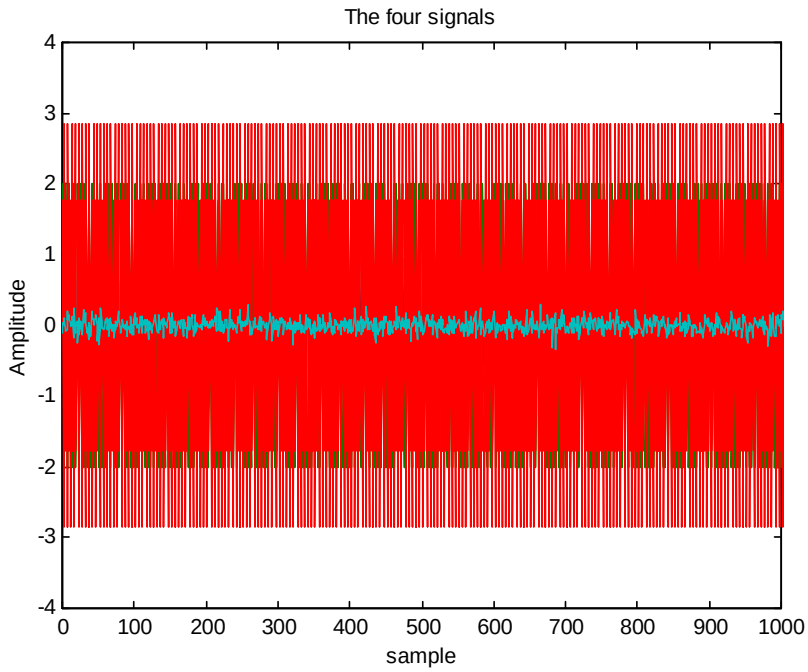


Figure 3. *The four signals.*

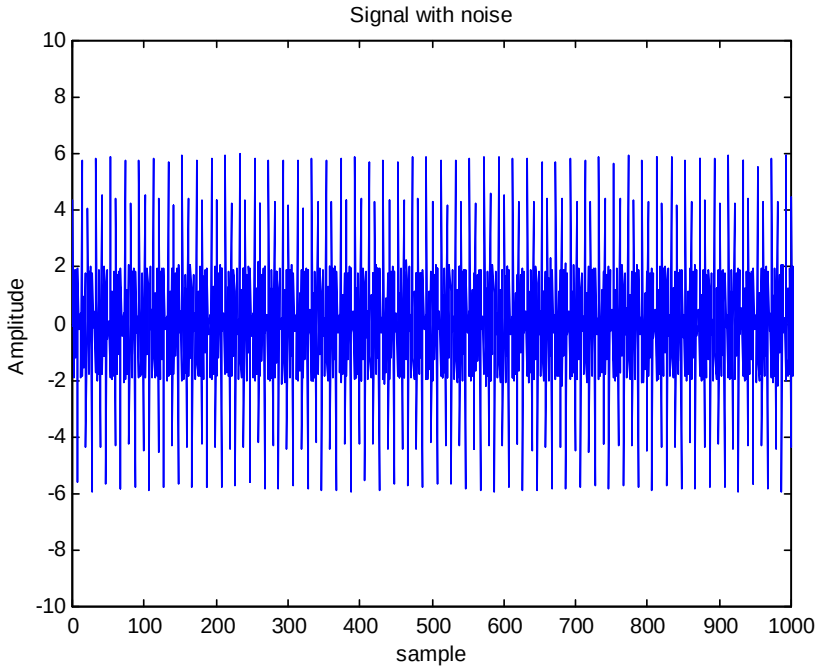


Figure 4. *Signal with noise.*

2.1 Sptool

In this section it used one of the Matlab tools the Sptool, and the first example is analysed with the same sampling frequency of 1000 Hz. So the signal is going to be processed with this tool is a signal with three sine waves and some additional noise, like appears in the upper half of figure 4.

Next figure illustrates the spectrum of this signal using the spectrum viewer of Sptool.

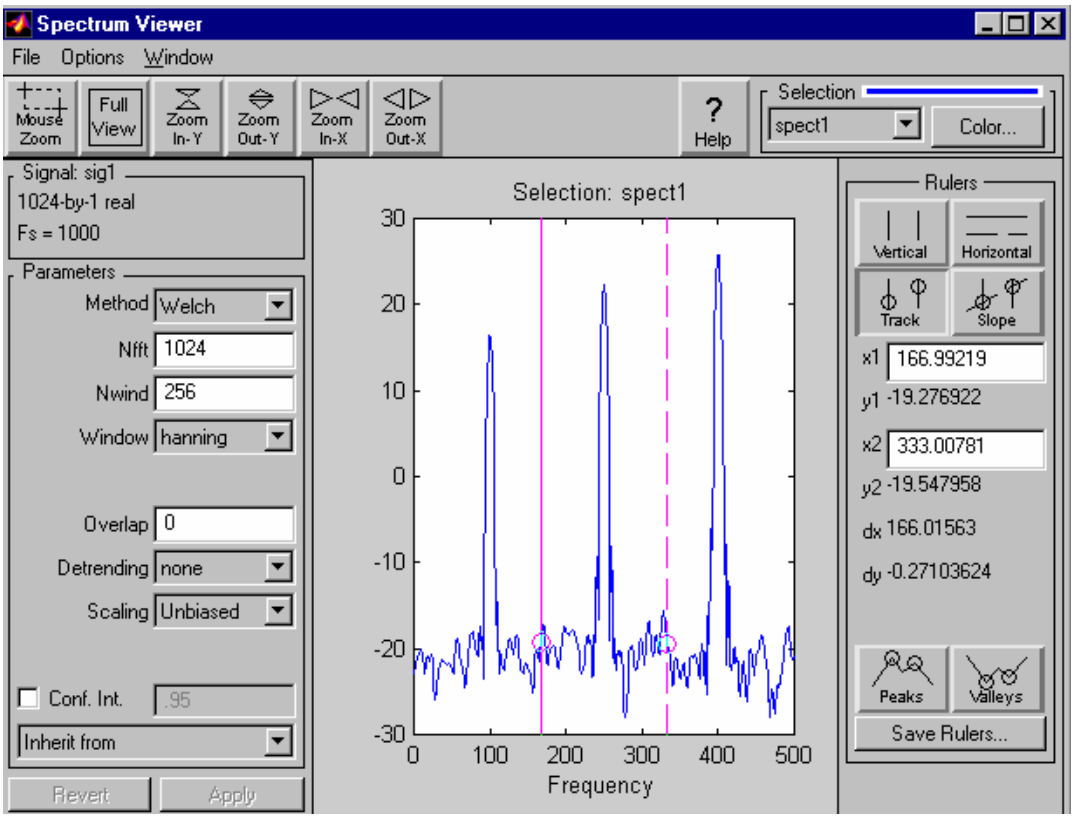


Figure 5. *Spectrum of the signal.*

In the previous figure the magnitude is in dB and the frequency range is from (0 to fs/2), the position and amplitude of the three signals in the figure are:

- The first signal corresponds to the first peak in the spectrum viewer has an amplitude of 16.32 dB and the frequency is 99.60 Hz.

- The second signal is 250 Hz and 22.34 dB.
- And the third one is 400.39 Hz and 25.7 dB.

The value of the noise floor if we take the highest peak of noise that appears in figure 5 the value is –15.45 dB, that is 5 dB higher that the one it was expected –20 dB.

Next step is to create a filter using the filter designer of Sptool, to filter out the three signals and separate them. Now are shown each signal after filtering with the original signal to compare both and the value of the amplitude and frequency is calculated to observe differences with the original signal.

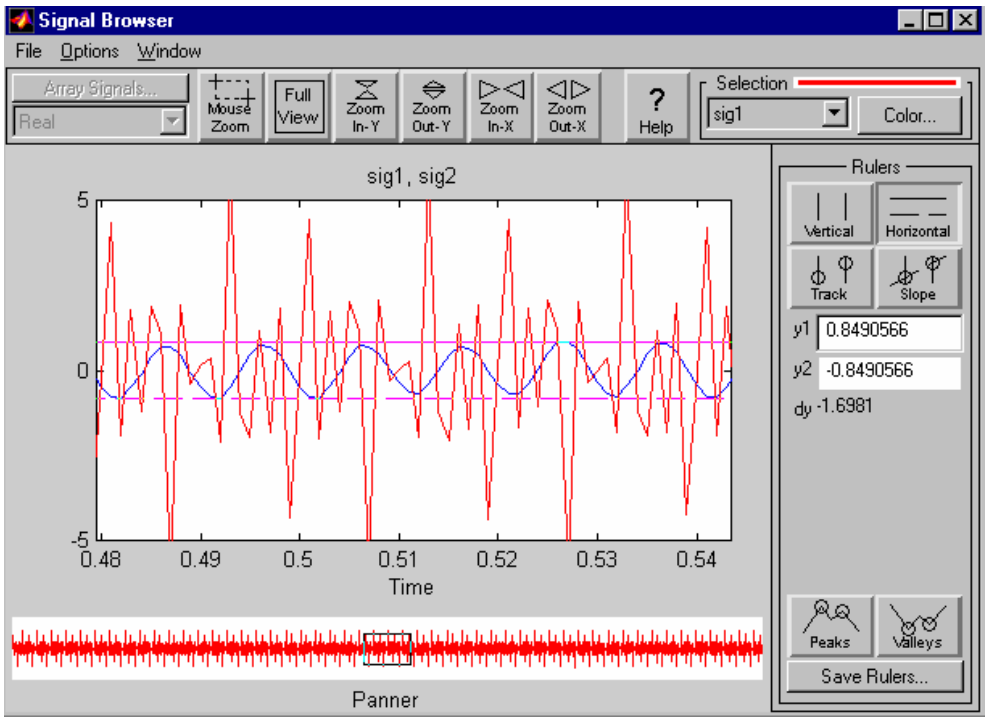


Figure 6. *This is the signal and the first signal after filtering.*

The previous figure shows the first signal after filtering applying a low pass filter as it can be observed the amplitude does not corresponds to the original signal, now the level is 0.84 V and the original one is 1 V, the frequency is 100 Hz that is the same as the original. The spectrum is shown where it can be compared the level of the filtered signal and the rejected ones. Also the pattern is not exactly the same of a sine wave.

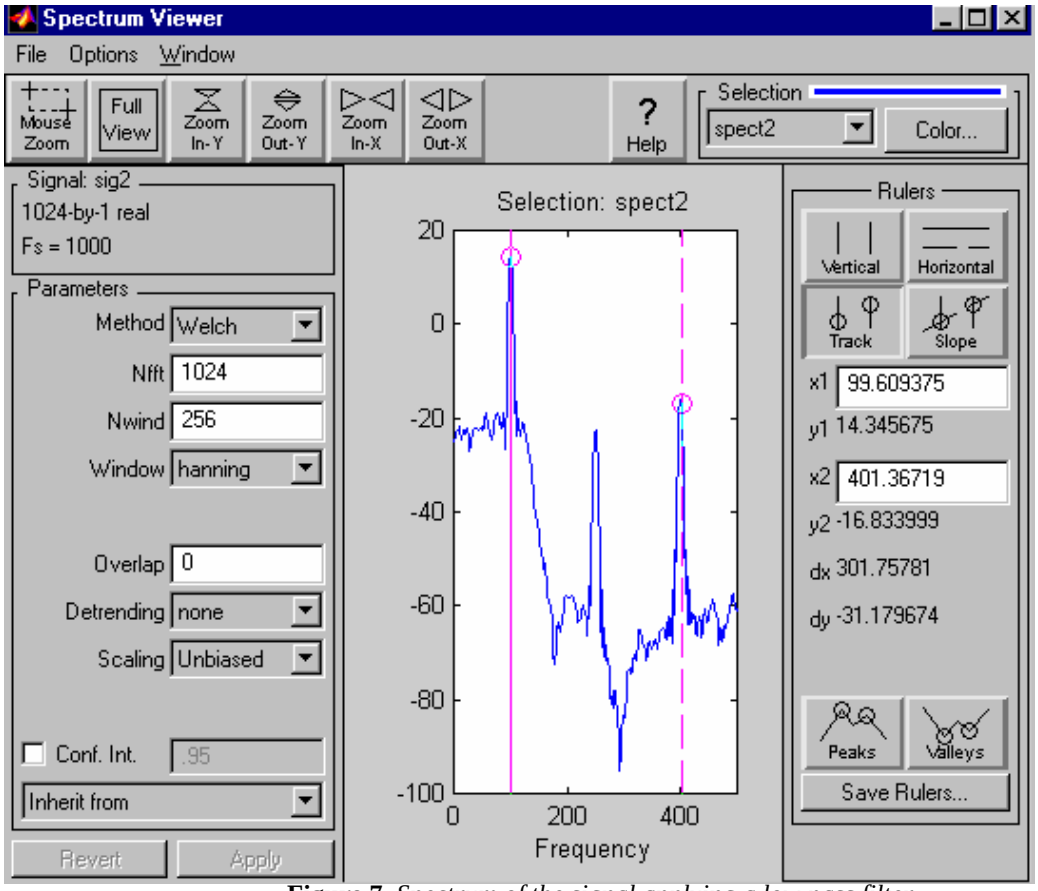


Figure 7. *Spectrum of the signal applying a low pass filter.*

Now is applied a band-pass filter to obtain the second signal so the next figures are obtained,

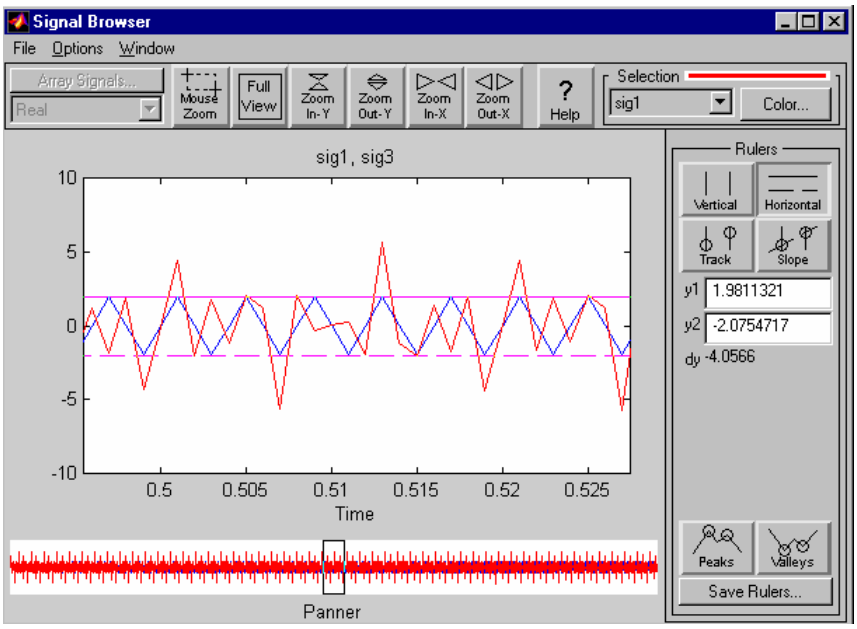


Figure 8. *This is the signal and the second signal after filtering.*

As it can be seen from this figure the signal is not exactly a sine wave, the amplitude is 1.98 V when it should be 2 V and the frequency is 245 Hz when it should be 250 Hz. The spectrum of the three signals after applying the band-pass filter is,

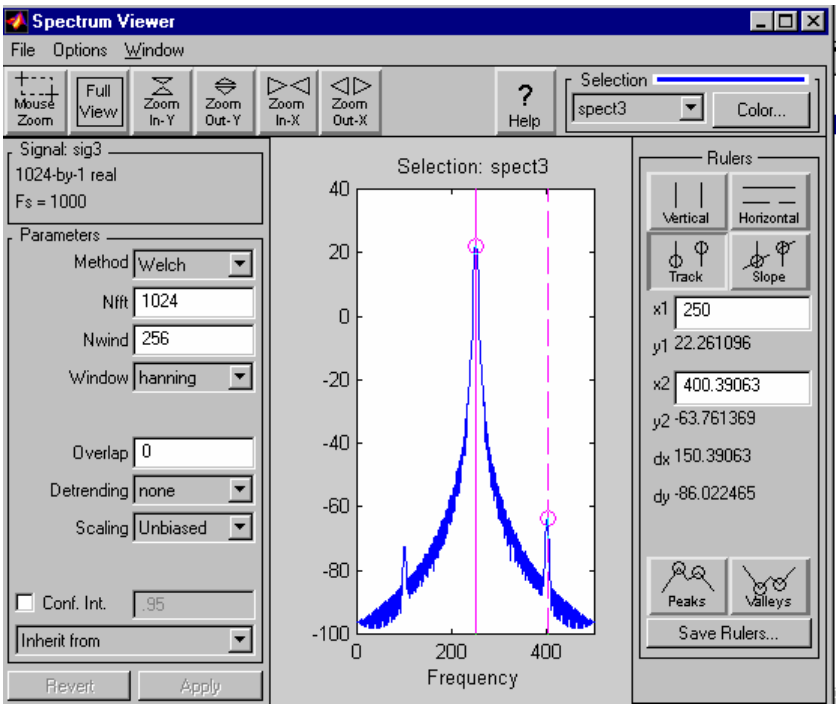


Figure 9. *The spectrum of the three signals after applying the band-pass filter.*

Now for the third signal is applied a high-pass filter so these figures are obtained,

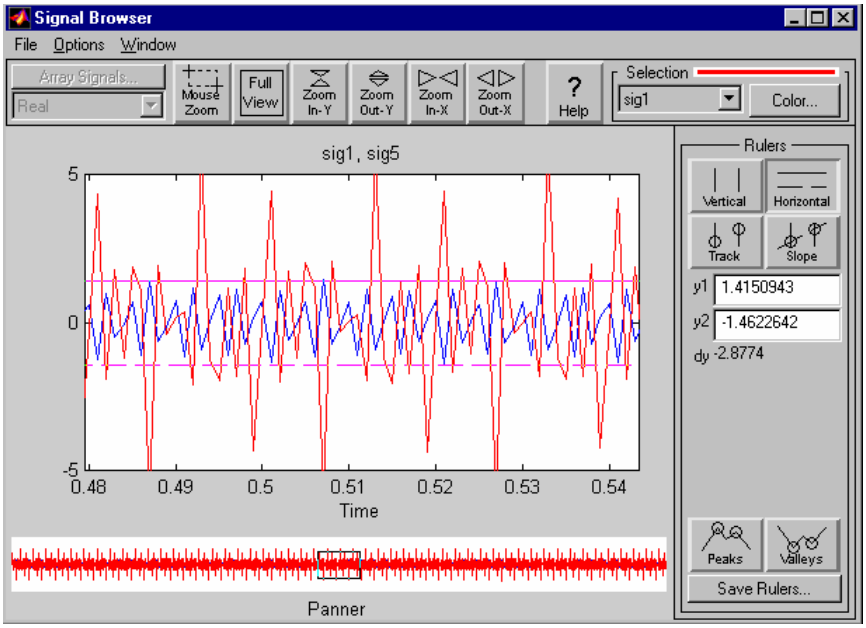


Figure 10. *This is the signal and the third signal after filtering.*

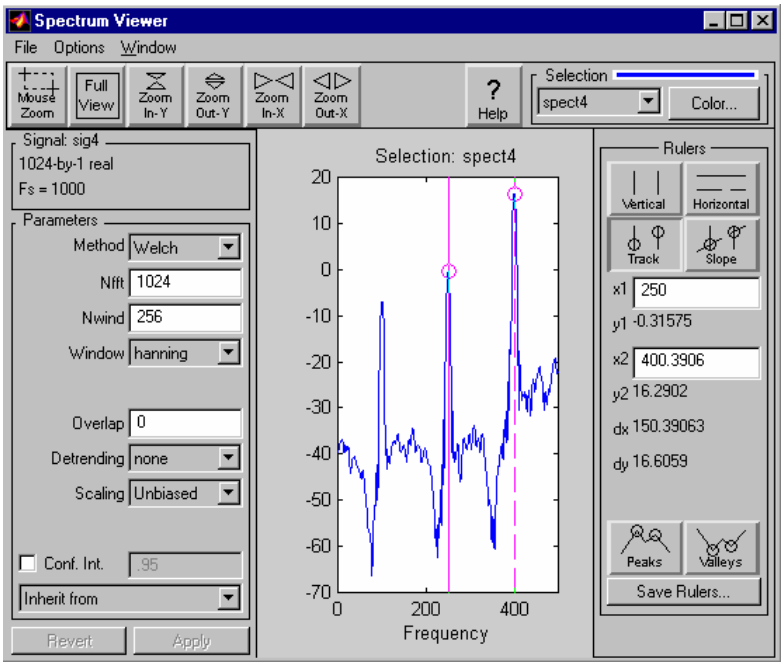


Figure 11. *The spectrum of the three signals after applying the high-pass filter.*

As it can be observed from figure 10 the signal is not a sine wave and the amplitude is 1.41 V when it should be 3 V. The frequency cannot measured with the signal browser because it is not clear that the signal is periodic, but it has been illustrated in the spectrum viewer that is the same as the original signal 400.39 Hz, when the original should be 400 Hz.

So the results do not meet the specifications, also the low-pass filter should has these specifications:

Ideal Specifications
- F1= 125
- F2= 225
- Rp= 2
- Rs= 40

The last part is to obtain the coefficients of the three filters and the transfer function in terms of H(z), are the next ones obtained after exporting the filters to the workspace in Matlab and extracting from the function,

Low-pass Filter

num: [0.02 -0.01 0.04 -0.01 0.02],

den: [1.00 -2.84 3.56 -2.20 0.58].

$$H(z)=\frac{0.02\cdot Z^4-0.01\cdot Z^3+0.04\cdot Z^2-0.01\cdot Z+0.02}{1\cdot Z^4-2.84\cdot Z^3+3.56\cdot Z^2-2.20\cdot Z+0.58}$$

Band-pass Filter

num: [0.00007 0 -0.00023 0 0.00023 0 -0.00007]

den: [1.00 0.00 2.82 0.00 2.66 0.00 0.84]

$$H(z)=\frac{\left(0.07\cdot Z^6-0.23\cdot Z^4+0.23\cdot Z^2-0.07\right)\cdot 10^{-3}}{1\cdot Z^6+2.82\cdot Z^4+2.66\cdot Z^2+0.84}$$

High-pass Filter

num: [0.01 -0.17 0.14 -0.17 0.18 -0.17 0.14 -0.17 0.01]

den: 1.00

$$H(z)=\frac{0.01\cdot Z^8-0.17\cdot Z^7+0.14\,Z^6-0.17\cdot Z^5+0.18\,Z^4-0.17\cdot Z^3+0.14\,Z^2-0.17\cdot Z+0.01}{1}$$

3 THE DESIGN AND REALISATION OF A DIGITAL FILTER

This is the second part of the assignments and it is divided in two sections.

3.1 Section 1: Design of an FIR lowpass filter using the Fourier Transform method.

Using Matlab is design a seven tap lowpass filter with wc= 0.3π (Note: fs= 0.2π), using the rectangular window. The function used in Matlab is 'fir1' and the impulse response of the filter can be written as,

$$h(n) = \frac{\sin(0.3 \cdot \pi(n-3))}{\pi(n-3)} \quad n = 0,1.....,6$$

That it is a sinc function, calculating manually the values of h(n) for the six first terms and compare these with the obtained by Matlab the results are,

$h(0) = 0.0327 \quad h(1) = 0.1513 \quad h(2) = 0.2575 \quad h(3) = 0.3 \quad h(4) = 0.2575 \quad h(5) = 0.1513 \quad h(6) = 0.0327$

h = 0.0328 0.1514 0.2575 0.3000 0.2575 0.1514 0.0328 $\Rightarrow$ Matlab

Bearing in mind that for h(3) L'Hopital rule may be applied in order to solve the indeterminacy of dividing 0 by 0. Also tables of the sinc function can be consulted.

In the next program is shown all the code implemented to answer all the questions in this section.

```
%Design of a seven taps low pass filter using
%the rectangular window.

wc=0.3*pi;
n=[0:6];
fs=2*pi;
n2=[0:7];
h=0.3*sinc(0.3*(-3:3));
[h2,w]=freqz(h,1,512,fs);
figure(1)
stem(n,h);
title('Low-Pass Filter with 7 taps Impulse Response')
xlabel('n');
ylabel('h[n]');
a=fir1(7,0.3,boxcar(8));
[H,w] = freqz(a,1,512,fs);
figure(2)
plot(w,abs(H)), grid
title('Frequency Response of the Filter')
xlabel('f(Hz)');
ylabel('Abs h(w)');
figure(6)
stem(n2,a);
% More taps introduced and comparison between them
n1=[0:6];
h1=0.3*sinc(0.3*(-3:3));
figure(3)
subplot(2,1,2)
stem(n1,h1);
title('Low-Pass Filter with 11 taps Impulse response')
xlabel('n');
ylabel('h[n]');
subplot(2,1,1)
stem(n,h);
title('Low-Pass Filter with 7 taps Impulse response')
xlabel('n');
ylabel('h[n]');
a1=fir1(11,0.3,boxcar(12));
[H1,w1] = freqz(a1,1,512,fs);
figure(2)
subplot(2,1,2)
plot(w,abs(H1)), grid
title('Frequency Response of the Filter with 11 taps')
xlabel('f(Hz)');
ylabel('Abs h1(w)');
subplot(2,1,1)
plot(w,abs(H)), grid
title('Frequency Response of the Filter with 7 taps')
xlabel('f(Hz)');
ylabel('Abs h(w)');
% end of comparison
%Different windows and comparison between them
a3=fir1(7,0.3,triang(8));
[H2,w] = freqz(a3,1,512,fs);
figure(4)
subplot(2,2,4)
plot(w,abs(H2)), grid;
title('Frequency Response  Triangular window')
xlabel('f(Hz)');
ylabel('Abs h(w)');

a4=fir1(7,0.3,hamming(8));
[H3,w] = freqz(a4,1,512,fs);
subplot(2,2,3)
plot(w,abs(H3)), grid;
title('Frequency Response Hamming window')
xlabel('f(Hz)');
```

```
ylabel('Abs h(w)');

subplot(2,2,2)
plot(w,abs(H)), grid;
title('Frequency Response  Rectangular window')
xlabel('f(Hz)');
ylabel('Abs h(w)');

subplot(2,2,1)
plot(w,abs(h2)), grid;
title('Frequency Response  Theoretical')
xlabel('f(Hz)');
ylabel('Abs h(w)');

figure(5)
subplot(2,2,2)
stem(n2,a);
title('Impulse Response rectangular window')
xlabel('n');
ylabel('h(n)');
subplot(2,2,3)
stem(n2,a4)
title('Impulse Response Hamming window')
xlabel('n');
ylabel('h(n)');
subplot(2,2,4)
stem(n2,a3)
title('Impulse Response Triangular window')
xlabel('n');
ylabel('h(n)');
subplot(2,2,1)
stem(n,h);
title('Impulse Response theoretical')
xlabel('n');
ylabel('h[n]');

%The filter coefficients of the first filter are
'the filter coefficients are'
h
```

The frequency response of the filter is shown in the next figure,

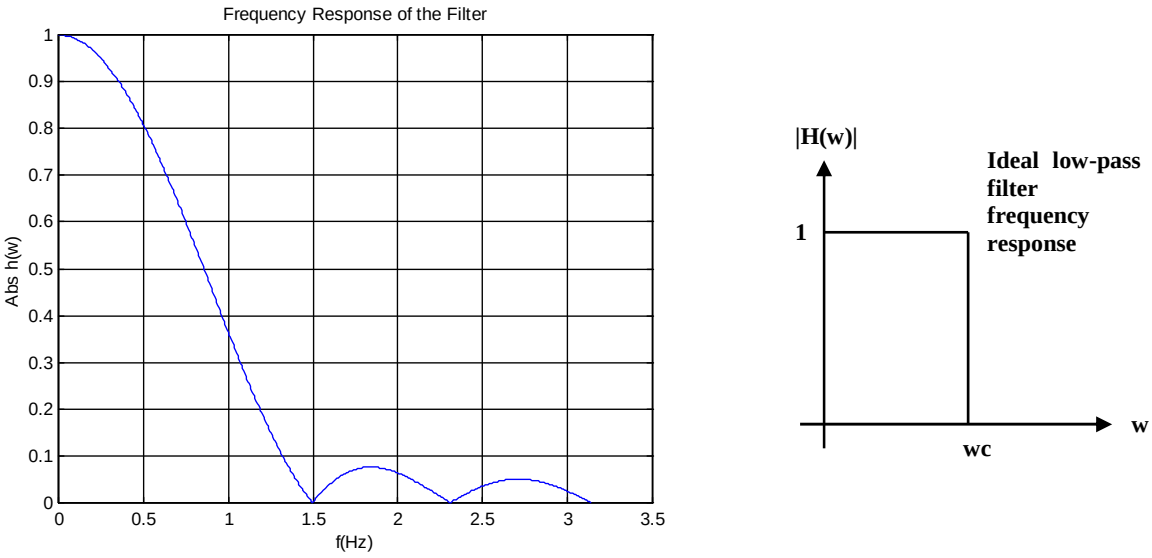


Figure 12. *Frequency response of the low-pass filter comparing with the ideal one.*

From the previous figure it is demonstrated that a real low-pass filter has not a sharp decrease in the stop band, the roll-off of the filter is softer. There is a transition band and also the response of the filter does not go to cero exactly. The use of the sinc function to describe a filter like this it is the reason of all these differences between the ideal characteristic and the real one.

If more taps are introduced in this filter the pattern of the filter becomes more similar to the ideal characteristic and the decreased after the cut-off frequency will be sharper than the previous one, the next figure demonstrates this.

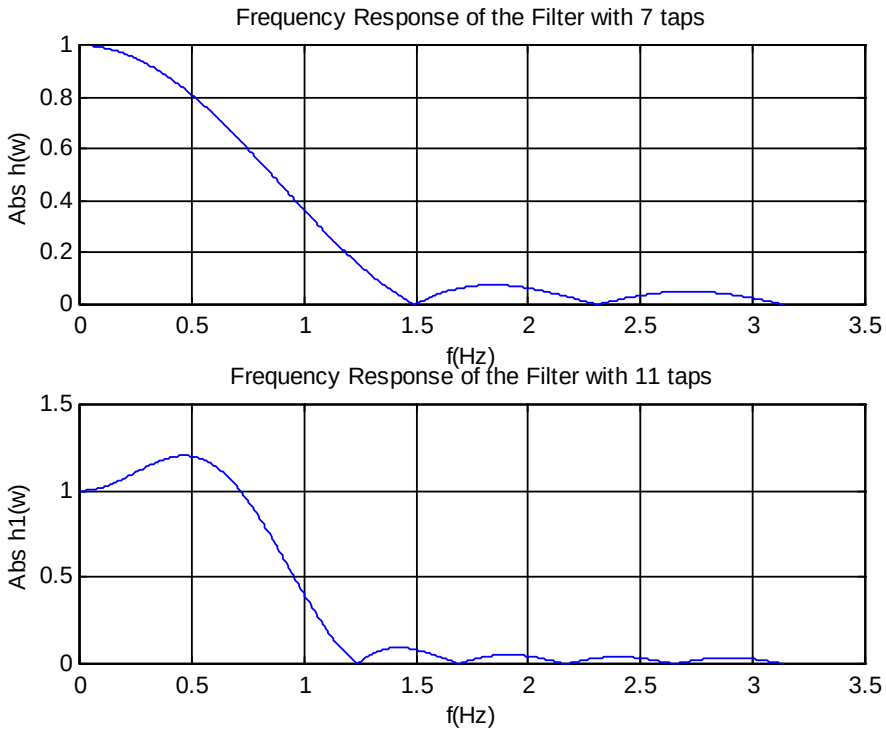


Figure 13. *Frequency response of the low-pass filter comparing one with seven taps and other with eleven.*

The roll-off of this filter as it has been commented before is sharper but also appears a ripple in the pass band that was not in the seven taps filter.

The same low-pass filter with seven taps is redesign using different windows like it was shown in the program at the beginning of this section and the following figures are obtained.

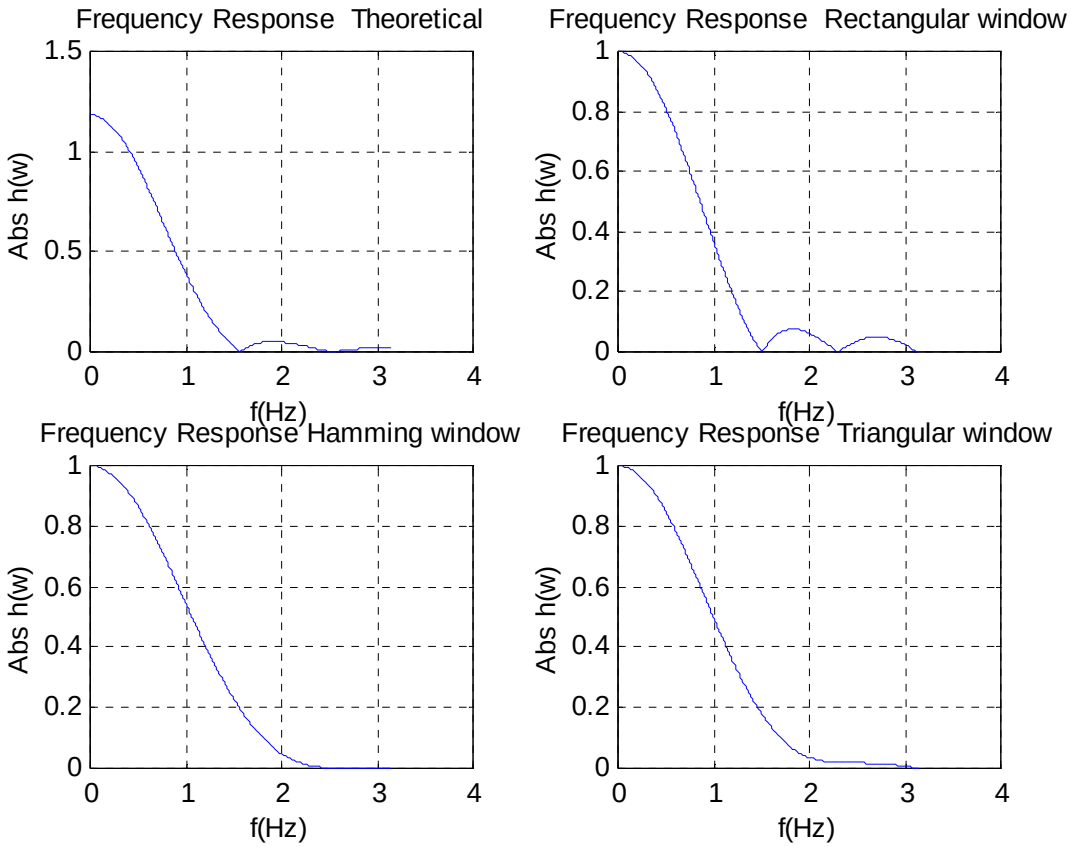


Figure 14. *Frequency response of the low-pass filter comparing the effect of the different windows.*

The differences between the windows used are explained by the method of windowing,

$$h_d(n) = DTFT^{-1}\{H_d(w)\} = \frac{1}{2\pi}\int_{-\pi}^{+\pi} H(w)\cdot e^{jwn}\,dw = \frac{1}{2\pi}\int_{-wc}^{+wc} e^{jwn}\,dw = \frac{\sin(nwc)}{n\pi} = \frac{wc}{\pi}\frac{\sin(nwc)}{n\pi} = \frac{wc}{\pi}\sin c(nwc)$$

Then if w(n) is a window of length M+1 that is symmetric at 0, then w(n) should satisfy

$$w\left(\frac{-M}{2}+n\right) = w\left(\frac{M}{2}-n\right)$$

Consequently the FIR filter of length M+1 will be given by,

$$h(n) = h_d \cdot w(n)$$

Where h_d is given by the previous expression and w(n) is going to be the window used in each case. So here are the windows that have been used for this filter.

$$\text{Rectangular} \qquad w(n) = 1 \quad for \quad \frac{-M}{2} \le \frac{M}{2}$$
$$= 0 \quad otherwise$$

$$Hamming \qquad w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{M}\right) \quad for \begin{cases} \frac{-M}{2} \le n \le \frac{M}{2} & for \quad M \quad even \\ \frac{-M-2}{2} \le n \le \frac{M-1}{2} & for \quad M \quad odd \end{cases}$$
$$= 0 \quad otherwise$$

$$Triangular \qquad w(n) = \frac{(M+1)-|n|}{(M+1)^2} \quad for \quad -M \le n \le M$$
$$= 0 \quad otherwise$$

So it is obvious that if a sharper decrease of the filter is desirable more taps should be introduced, loosing accuracy in the passband because of the ripple introduced. The main advantage using windowing method is that at the stopband the response is cero. But the roll-off is a bit worse observing the previous graph; the transition band between the pass-band and the stop band is wider. Now it is shown the differences between the impulse responses of the different windows respect to the original expression of the filter.

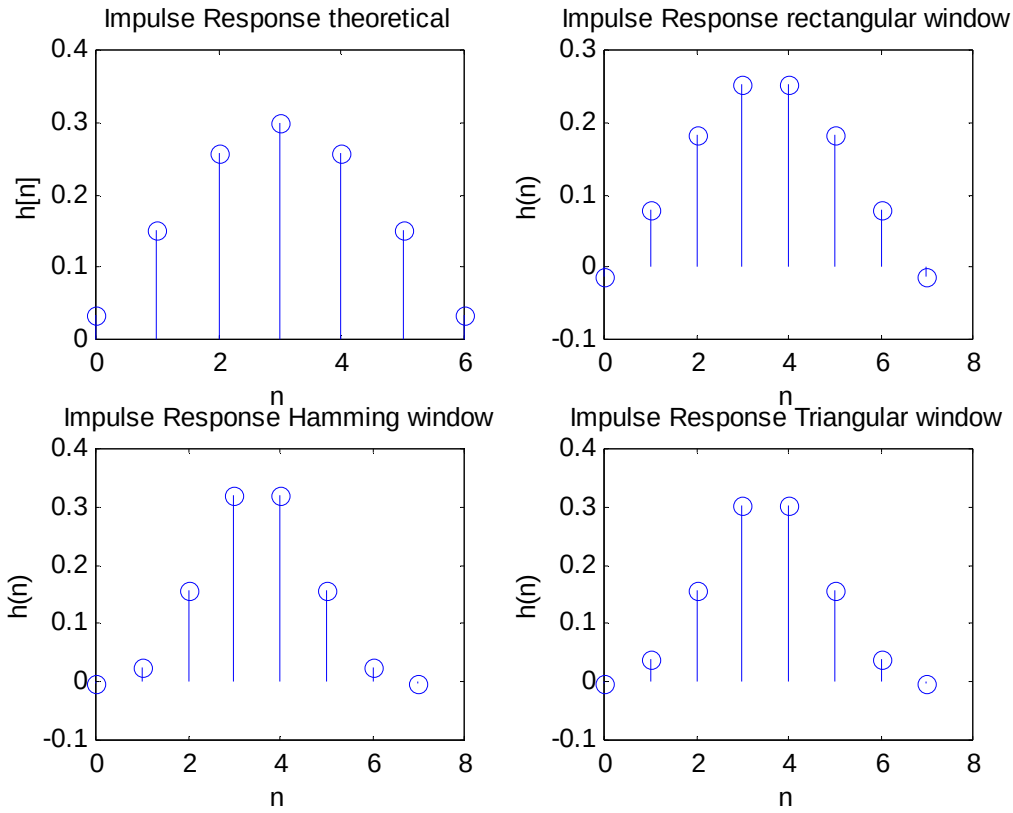


Figure 15. *Frequency response of the low-pass filter comparing the effect of the different windows.*

Differences in the values of the coefficients that are determined by the different windows expressions.

- **Filter Realisation**

Using the audio codec example program as a starting point, the following program was designed to implement the filter and run it in the TMS320C5402 DSP platform.

```
/************************************************************************/
/* FIR.c                                                                */
/* Digital Loopback example                                             */
/************************************************************************/
#include <type.h>
#include <board.h>
#include <codec.h>
#include <mcbsp54.h>
/************************************************************************/
/* Function Prototypes                                                  */
/************************************************************************/
/* This delay routine does not conflict with DSP/BIOS.  It is used in this  */
/* example rather than brd_delay_msec which causes DSP/BIOS conflicts just  */
/* because of this.  If you are not using DSP/BIOS, you can change the code */
/* to use brd_delay_msec.                                                   */

void delay(s16 period);
/************************************************************************/
/* Global Variables                                                     */
/************************************************************************/
HANDLE hHandset;
s16 Yn;
s16 Xn;
int a0,a1,a2,a3,a4,a5,a6;
/************************************************************************/
```

```
/* MAIN                                                    */
/*************************************************************************/
void main()
{
   s16 cnt=2;
   a0=0,a1=0,a2=0,a3=0,a4=0,a5=0,a6=0;

   if (brd_init(100))
      return;

         /* blink the leds a couple times */
         while ( cnt-- )
         {
                  brd_led_toggle(BRD_LED0);
                  /* brd_delay_msec(1000); */
                  delay(1000);
                  brd_led_toggle(BRD_LED1);
                  /* brd_delay_msec(1000); */
                  delay(1000);
                  brd_led_toggle(BRD_LED2);
                  /* brd_delay_msec(1000); */
                  delay(1000);
         }
   /* Open Handset Codec */
   hHandset = codec_open(HANDSET_CODEC);            /* Acquire handle to codec */

   /* Set codec parameters */
   codec_dac_mode(hHandset, CODEC_DAC_15BIT);       /* DAC in 15-bit mode */
   codec_adc_mode(hHandset, CODEC_ADC_15BIT);       /* ADC in 15-bit mode */
   codec_ain_gain(hHandset, CODEC_AIN_6dB);         /* 6dB gain on analog input to ADC */
   codec_aout_gain(hHandset, CODEC_AOUT_MINUS_6dB);  /* -6dB gain on analog output from DAC */
```

```
    codec_sample_rate(hHandset,SR_10667);              /* 10KHz sampling rate */


    /* Polling and digital loopback */
    while (1)
    {
      /* Wait for sample from handset */
      while (!MCBSP_RRDY(HANDSET_CODEC)) {};


      /* Read sample from and write back to handset codec */
      Xn = *(volatile u16*)DRR1_ADDR(HANDSET_CODEC);
            a6=a5;
            a5=a4;
            a4=a3;
            a3=a2;
            a2=a1;
            a1=a0;
            a0=Xn;
      Yn= (a0*0.0328) + (a1*0.1514) + (a2*0.2575) + (a3*0.3) + (a4*0.2575) + (a5*0.1514) + (a6*0.0328);
  /*The coefficients have been quantized*/


      *(volatile u16*)DXR1_ADDR(HANDSET_CODEC) = Yn;
    }
}
void delay(s16 period)
{
    int i, j;
    for(i=0; i<period; i++)
    {
       for(j=0; j<period>>1; j++);
    }
}
```

And this is the Frequency Response obtained using a frequency generator and a oscilloscope, setting the sampling frequency to 10 Khz And the input voltage to 20 mV peak to peak.

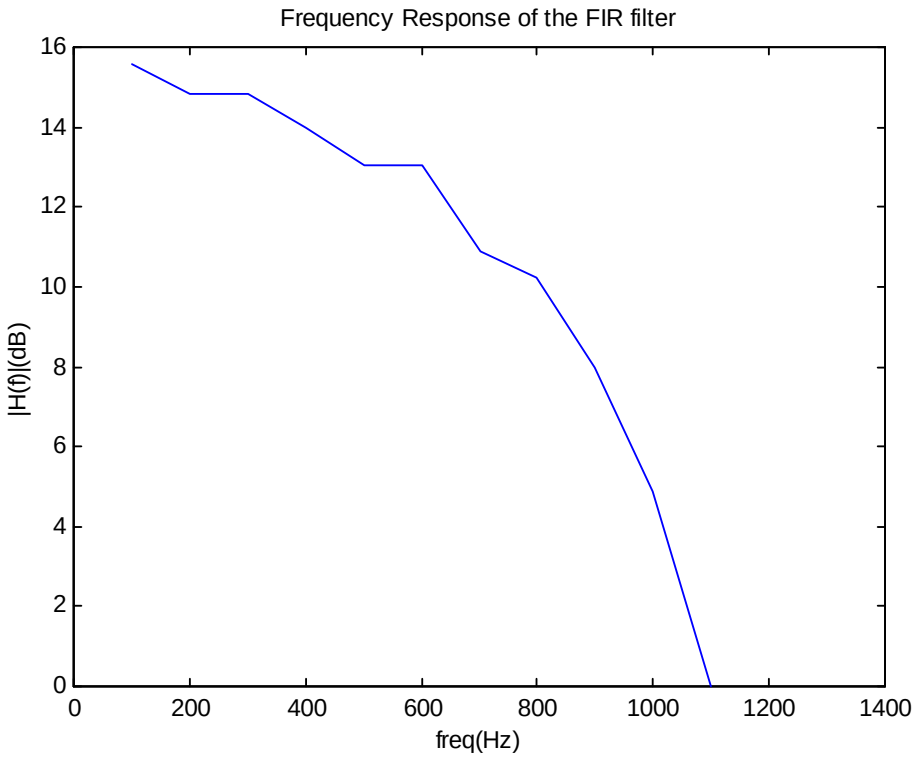


Figure 16. *Frequency response of the Digital low-pass filter FIR obtained with the oscilloscope and the frequency generator.*

Frequency (Hz)	Input Voltage pp (V)	Output Voltage pp (V)
100	0.02	0.12
200	0.02	0.11
300	0.02	0.11
400	0.02	0.10
500	0.02	0.09
600	0.02	0.09
700	0.02	0.09
800	0.02	0.065
900	0.02	0.05
1000	0.02	0.045
1100	0.02	0.035
1200	0.02	0.02
1300	0.02	0

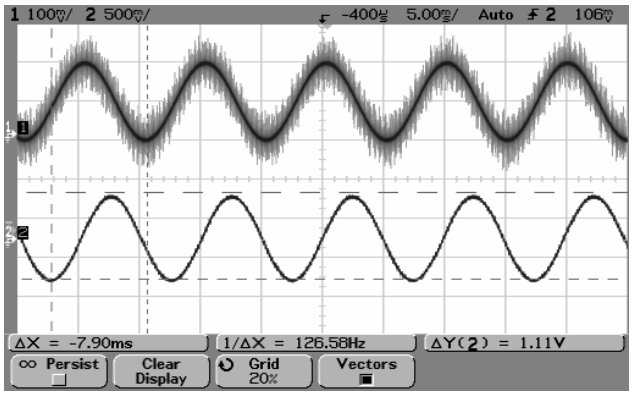
Table 1. *Values of the input and output for the Digital low-pass filter FIR obtained with the oscilloscope and the frequency generator.*

In order to plot with Matlab the frequency response this is the formula applied.

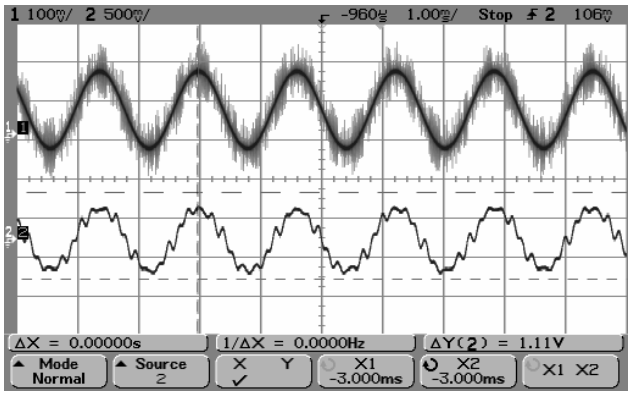
$$|H(f)|(dB) = \left| 20\log\left(\frac{Voutput}{vinput}\right)\right|$$

The previous expression is applied later for the IIR filter. The pattern obtained for the FIR filter is different from the ones obtained from Matlab, approximately the cut-off frequency is obtained when the frequency response has a value of 12.56 dB, this is for an output of 0.084 V peak to peak. At that point and looking at the graph of the frequency response the cut-off frequency is approximately of 625 Hz. The maximum sampling rate could be because of the limitations of this DSP platform of 16 Khz, this is defined in the file codec.h that is a header including at the beginning of the program. Now it is going to be shown two samples taking from a digital oscilloscope of the filter.

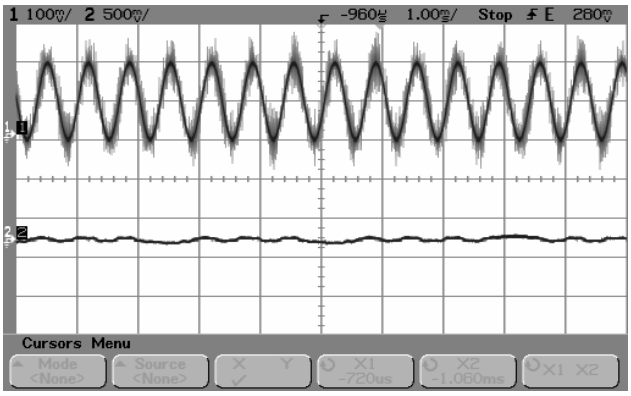
Frequency of 100 Hz



Frequency of 600 Hz



Frequency of 1400 Hz



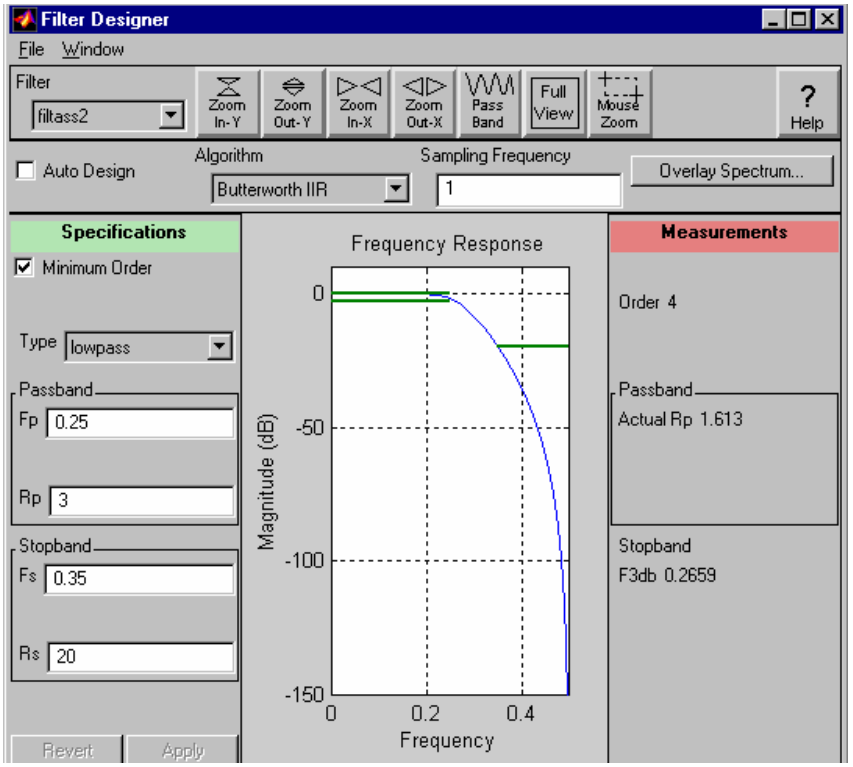
Signal 1 is the input in the upper half of the screen and signal 2 is the output for the three graphs.

Figure 17. *Some graphs obtained with the digital oscilloscope and the frequency generator.*

Note that at the previous graphs the input voltage and the output do not correspond to the values shown at table 1, this figure is only to demonstrate that the filter was working properly. The response of this filter will improve using other techniques in its implementation like circular buffers instead of shifting that it is the method used in the ‘c’ code shown before.

3.2 Section 2: Design a 4th order IIR filter.

Here is designed a low pass filter using Sptool and with the following specifications.



Specifications

- F1= 0.25
- F2= 0.35
- Rp= 3
- Rs= 20
- Fs= 1
- Order= 4

Figure 18. *Filter Designer and the specifications followed.*

After this and exporting this filter to the workspace of Matlab through the next program in Matlab is obtained the frequency response, the impulse function and the pole zero map of this filter it is also possible to make this with the Sptool but for the

author it is more interesting to learn the Matlab functions that provides the same results so the Matlab code is,

```
%Assignment Section 2
%Load the filter to obtain the coefficients

load filtass2.mat;
den=filtass2.tf.den;
num=filtass2.tf.num;

% Frequency Response
[H,w] = freqz(num,den,512,1);
figure(2)
plot(w,abs(H)), grid
title('Frequency Response of the Filter')
xlabel('f(Hz)');
ylabel('Abs h(w)');

%impulse response
[H1,T]=impz(num,den);
figure(3)
stem(T,H1);
title('Impulse Response of the Filter')
xlabel('n');
ylabel('h[n]');

%ploting the pole zero map
z=filtass2.zpk.z;
p=filtass2.zpk.p;
figure(4)
zplane(z,p)
```

```
title('pole zero map of the IIR Filter')

'the coefficients are'

num

den
```

And here are shown the figures produced by this code,

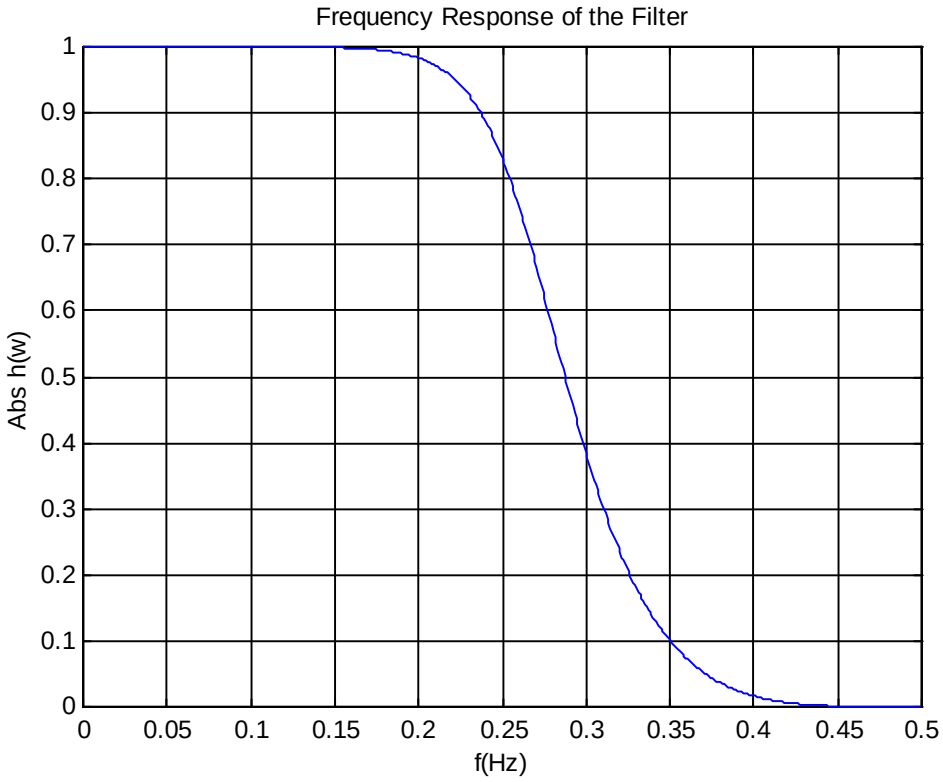


Figure 19. *Frequency response of the low-pass IIR filter of 4th order.*

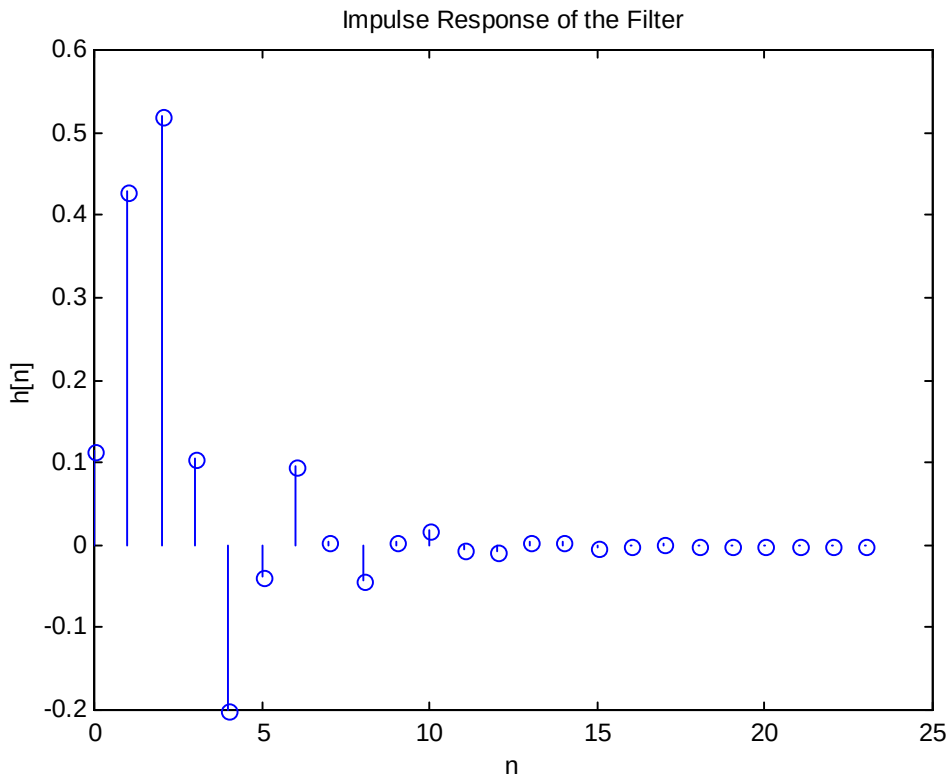


Figure 20. *Impulse function of the low-pass filter IIR of 4th order.*

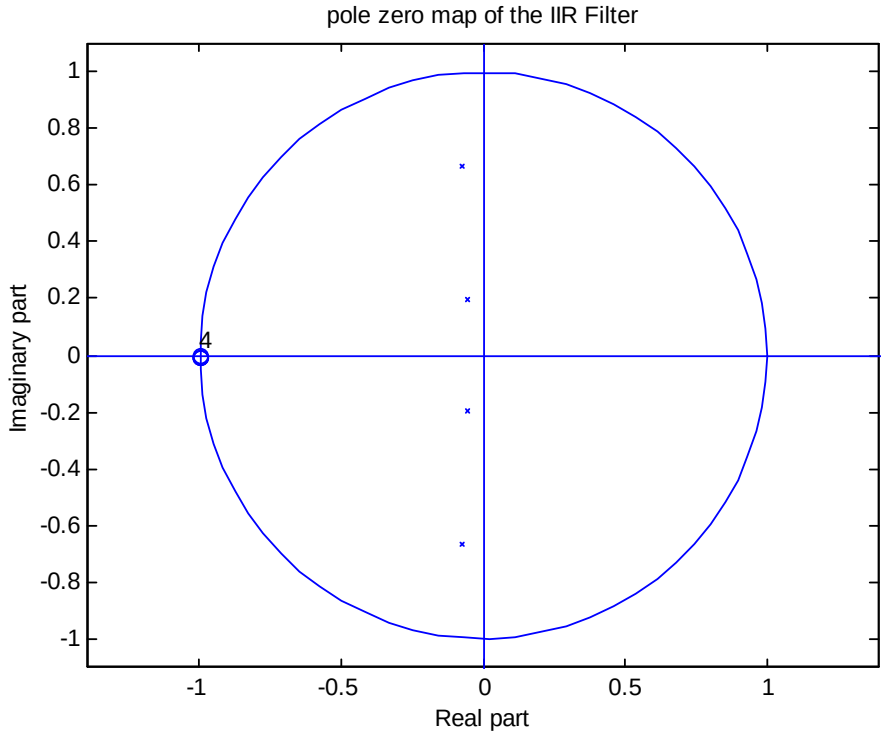


Figure 21. *Pole zero map of the low-pass filter IIR of 4th order.*

It is preferred to plot the graph using the function zplane because provides more information that plotting with Sptool the zero-pole map. It can be seen that the filter is stable, because all the ceros and poles are inside the unity circle and also in the left hand side of the Z-plane. It also important to notice that at –1 there is a pole of 4^{th} order.

- **Filter Realisation**

Using the audio codec example program as a starting point, the following program was designed to implement the filter and run it in the TMS320C5402 DSP platform.

```
/************************************************************************/
/* IIR.c                                                                */
/* 4th Order Butterworth IIR Low-pass filter                            */
/************************************************************************/
#include <type.h>
#include <board.h>
#include <codec.h>
#include <mcbsp54.h>

/************************************************************************/
/* Function Prototypes                                                  */
/************************************************************************/
void delay(s16 period);
/************************************************************************/
/* Global Variables                                                     */
/************************************************************************/
```

```
HANDLE hHandset;
s16 Xn;
s16 Y0;
s16 Yn;

/*****************************************************************/
/* MAIN                                                          */
/*****************************************************************/
void main()
{
   s16 cnt=2;
        int a0=0,a1=0,a2=0,a3=0,a4=0,b0=0,b1=0,b2=0,b3=0;


   if (brd_init(100))
      return;

        /* blink the leds a couple times */
        while ( cnt-- )
        {
                brd_led_toggle(BRD_LED0);
                /* brd_delay_msec(1000); */
                delay(1000);
                brd_led_toggle(BRD_LED1);
                /* brd_delay_msec(1000); */
                delay(1000);
                brd_led_toggle(BRD_LED2);
                /* brd_delay_msec(1000); */
                delay(1000);
        }
```

```
   /* Open Handset Codec */
hHandset = codec_open(HANDSET_CODEC);              /* Acquire handle to codec */

   /* Set codec parameters */
codec_dac_mode(hHandset, CODEC_DAC_15BIT);   /* DAC in 15-bit mode */
codec_adc_mode(hHandset, CODEC_ADC_15BIT);         /* ADC in 15-bit mode */
codec_ain_gain(hHandset, CODEC_AIN_6dB);           /* 6dB gain on analog input to ADC */
codec_aout_gain(hHandset, CODEC_AOUT_MINUS_6dB);    /* -6dB gain on analog output from DAC */
codec_sample_rate(hHandset,SR_10667);              /* 10KHz sampling rate */

   /* Polling and digital loopback */
   while (1)
   {
     /* Wait for sample from handset */
     while (!MCBSP_RRDY(HANDSET_CODEC)) {};

     /* Read sample from and write back to handset codec */
     Xn = *(volatile u16*)DRR1_ADDR(HANDSET_CODEC);

          a4= a3;
          a3= a2;
          a2= a1;
          a1= a0;
          a0= Xn;

          Y0 = (0.1141*a0 + 0.4562*a1 +0.6843*a2+0.4562*a3+0.1141*a4);

          b3= b2;
          b2= b1;
          b1= b0;
          b0= Y0;
```

```
            Yn = Y0-(0.2479*b0+0.5055*b1+0.0526*b2+0.0189*b3);


        *(volatile u16*)DXR1_ADDR(HANDSET_CODEC) = Yn;
    }
    }

void delay(s16 period)
{
int i, j;

for(i=0; i<period; i++)
{
      for(j=0; j<period>>1; j++);
}
}
```

And in the next page it is shown the Frequency Response obtained using a frequency generator and a oscilloscope, setting the sampling frequency to 10 Khz And the input voltage to 20 mV peak to peak.

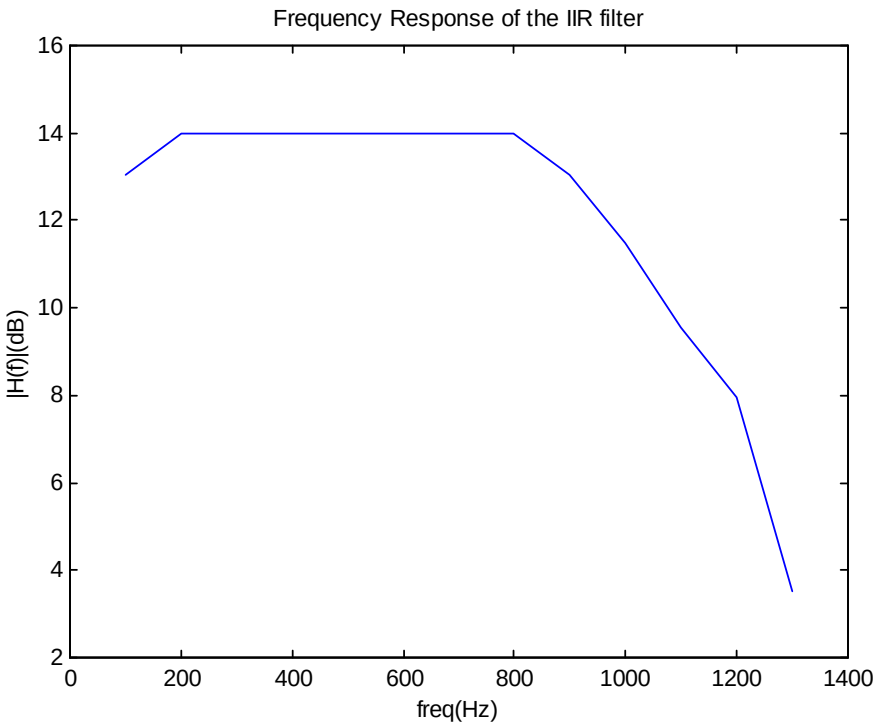


Figure 22. *Frequency response of the low-pass filter IIR of 4th order obtained with the frequency generator and the oscilloscope.*

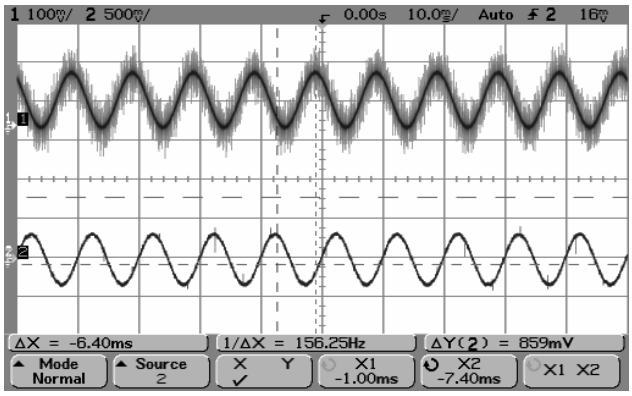
Frequency (Hz)	Input Voltage pp (V)	Output Voltage pp (V)
100	0.02	0.09
200	0.02	0.1
300	0.02	0.1
400	0.02	0.1
500	0.02	0.1
600	0.02	0.1
700	0.02	0.1
800	0.02	0.1
900	0.02	0.1
1000	0.02	0.09
1100	0.02	0.075
1200	0.02	0.060
1300	0.02	0.05

Table 2. *Values of the input and output for the Digital filter IIR obtained with the oscilloscope and the frequency generator.*

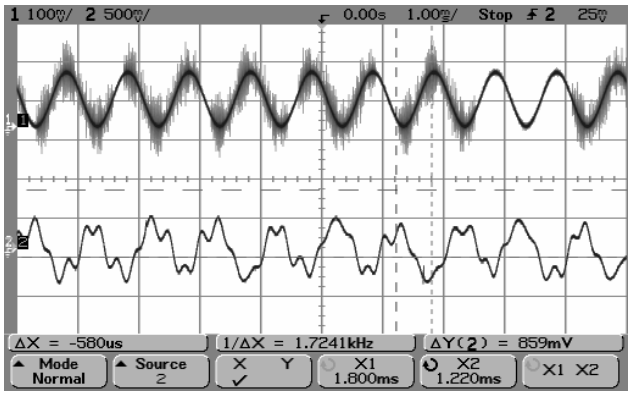
The same considerations in order to calculate the frequency response than for the FIR filter have been made here.

Approximately the cut-off frequency is obtained when the frequency response has a value of 10.97 dB, this is for an output of 0.070 V peak to peak. At that point and looking at the graph of the frequency response the cut-off frequency is approximately of 1050 Hz. The maximum sampling rate could be because of the limitations of this DSP platform of 16 Khz, this parameter is defined in the file codec.h that is a header including at the beginning of the program. Now it is going to be shown three samples taking from a digital oscilloscope of the filter, where the upper signal is the input and the bottom one is the output.

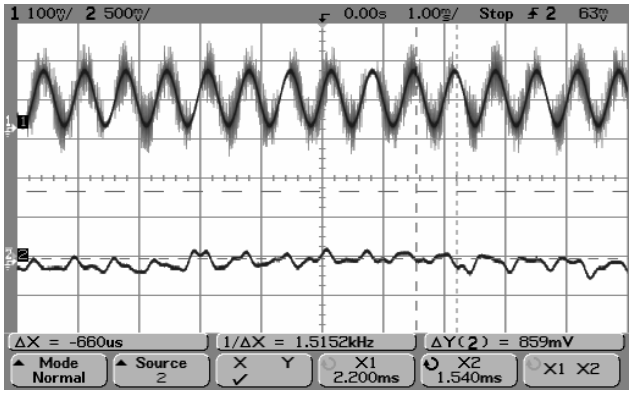
Frequency of 100 Hz



Frequency of 1000 Hz



Frequency of 1500 Hz



Frequency of 3800 Hz

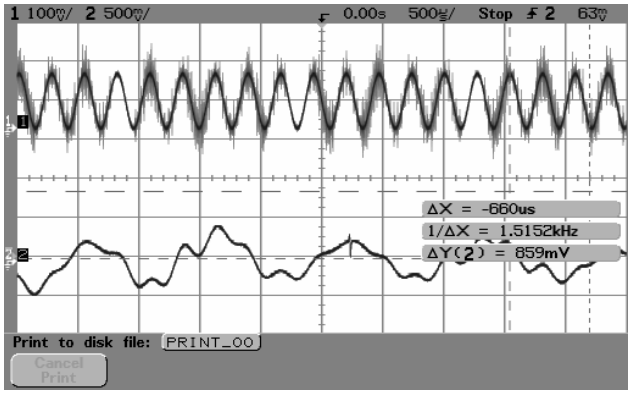


Figure 23. *Some graphs obtained with the digital oscilloscope and the frequency generator.*

Looking at the previous graphs it has to be noticed that it does not correspond to the values shown at table 2. All the measurements with the digital oscilloscope were made with the intention of demonstrate that the assignment and the aim have been achieved. The last filter has a higher cut-off frequency than the FIR filter it also important that at 3.8 Khz the output appears again but with higher frequencies this does not occur again, that it is one on the IIR filter characteristic infinite response. So as conclusion of this assignment other techniques might be implemented in order to achieve better accuracy and quality in the response of the filter.