Mechatronic Systems Identification

Lab 2 - Digital filtration: FIR and IIR

Khaldoun Fayad - 409597

Witold Surdej - 407100

18.03.2024

Description

The aim of this laboratory is for us to learn the basics of signal filtration and its different utilization methods.

Task1:

This laboratory aims to explore different filtration methods available in Matlab.

Task 1.1:

The first task aims to explore filtration's effects using a rectangular window with a length of 21 samples on the magnitude and phase of the input signal.

The characteristics of the filter can be seen below.

```
fs = 1000; % sampling
frequency
N = 1000; % number of
samples
t = (0:N-1)/fs; % time range
f1 = 10; %frequency one
f2 = 300; %frequency two
a1 = 1; % amplitude one
a2 = 0.2; % amplitude two
y = a1*sin(2*pi*f1*t) +
a2*sin(2*pi*f2*t);
%Rectangular Window
h = 1/21*rectwin(21);
%Freqz Filter
figure(3)
freqz(h,1,N,fs);
```

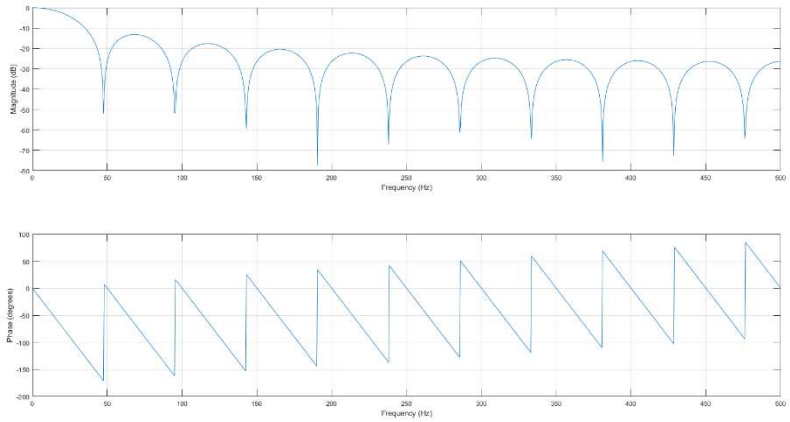


fig.1 Characteristics of the applied filter

Once applied to the signal we obtain the following output.

```
figure(2)
plot(t,y)
hold on
plot(t,yf)
plot(t,yc,'-.')
hold off
xlabel('time (s)');
ylabel('amplitude (~)');
```

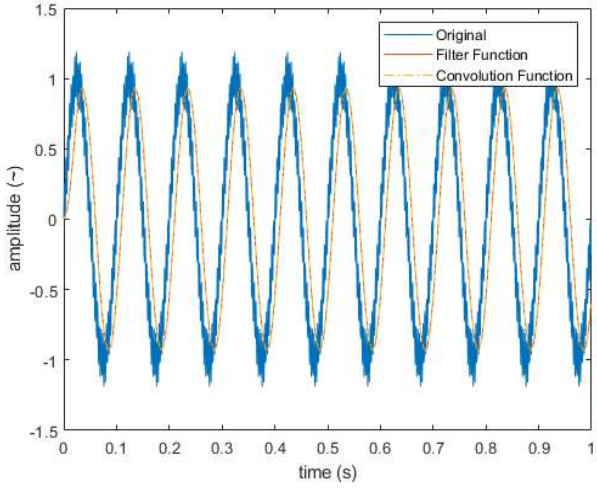


fig.2 Comparison of time signals from different types of filtration

The results of filtration using convolution and filter() function are identical. This result was expected since both operations result in the convolution of the original signal with the same window function. The only difference lies in the added flexibility of the filter() function when dealing with more complex filter designs.

The impact of applying the filter on the signal is a 10% decrease in the amplitude of the peak located at 10Hz and a 20x decrease in the amplitude of the peak at 300Hz. Additionally, a linear phase shift in the time domain could be observed. The result was to be expected given the magnitude characteristics of the applied filter (fig.1), The 1st had his amplitude diminished by -0.64dB while the other by -26.4dB

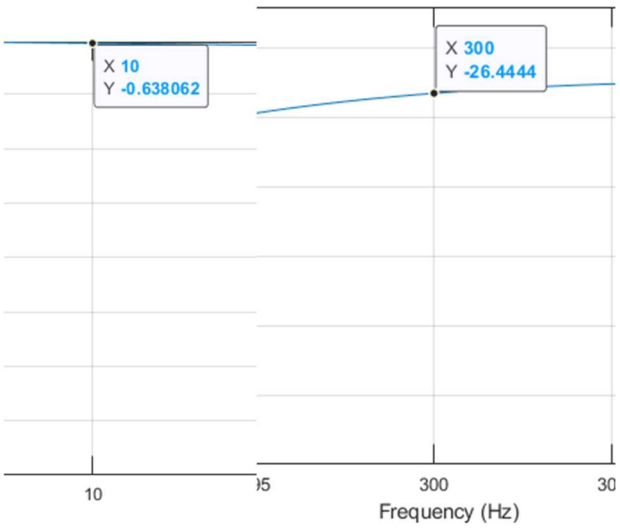


fig.3 Amplitude change in the characteristic frequencies introduced by filtering.

More broadly, the frequency characteristics of a filter in decibels (dB) are directly connected to the amplitude of the filtered signal in the frequency domain. This connection arises from the way filters affect signals in the frequency domain.

In the frequency domain, the amplitude response of a filter is often shown in decibels (dB). Decibels represent a logarithmic scale of relative power. When a filter attenuates certain frequencies, it means it reduces their amplitude. This attenuation is represented as a negative value in dB.

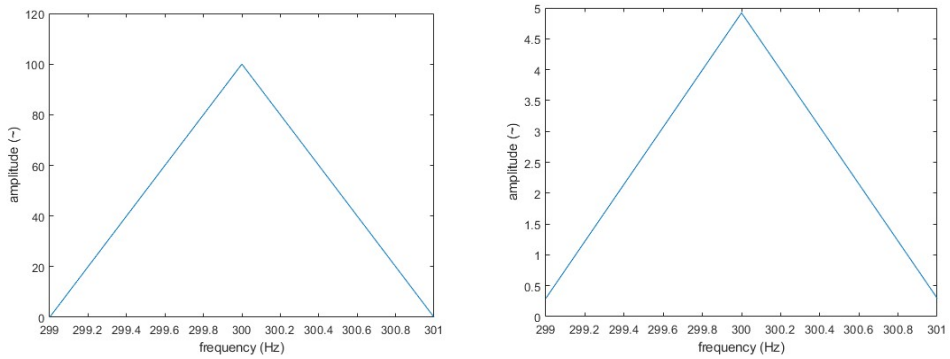


fig.4 The amplitude of the 300Hz peak pre and post-filtration.

task 1.2:

In this task, we have implemented filtration using filtfilt() function which offers zero-phase filtration capabilities. We have compared the results to the original filter() function and we have manually implemented zero-phase filtration in Matlab.

We can implement the zero-phase filtration by following the procedure showcased on the plot below:

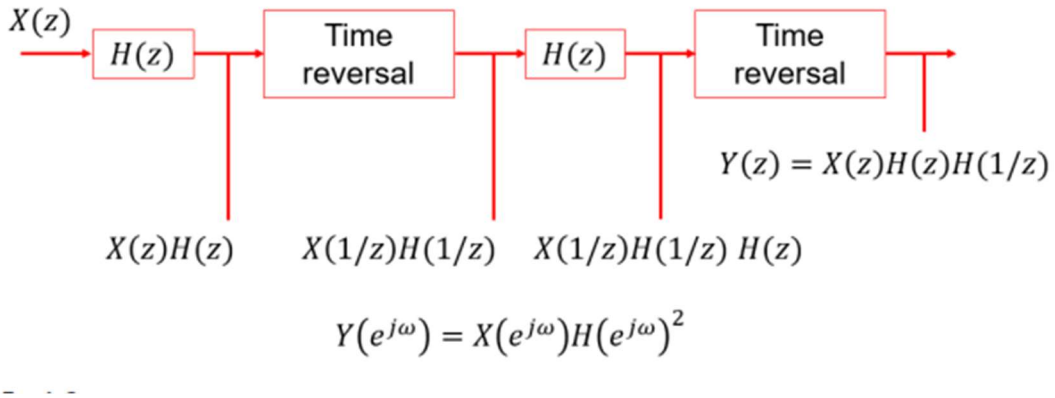


fig.5 Procedure for implementation of zero-phase filtration

By transcribing it into Matlab code we obtain the following:

```
yc2 = filter(h,1,y);
yc_reverse = yc2(end:-1:1);
yc_rev_conv =  filter(h,1,yc_reverse);
yz2 = yc_rev_conv(end:-1:1);
yz2 = yz2(1:1000);
```

fig.6 The implementation of the zero-phase filtration in Matlab

By running the code above and comparing it with the results obtained in task 1.1 and the filtfilt() function we obtain the following plots.

```
yz = filtfilt(h,1,y);
figure(4)
plot(t,y)
hold on
plot(t,yf)
plot(t,yz,'-.')
hold off
xlabel('time (s)')
ylabel('amplitude (~)')
```

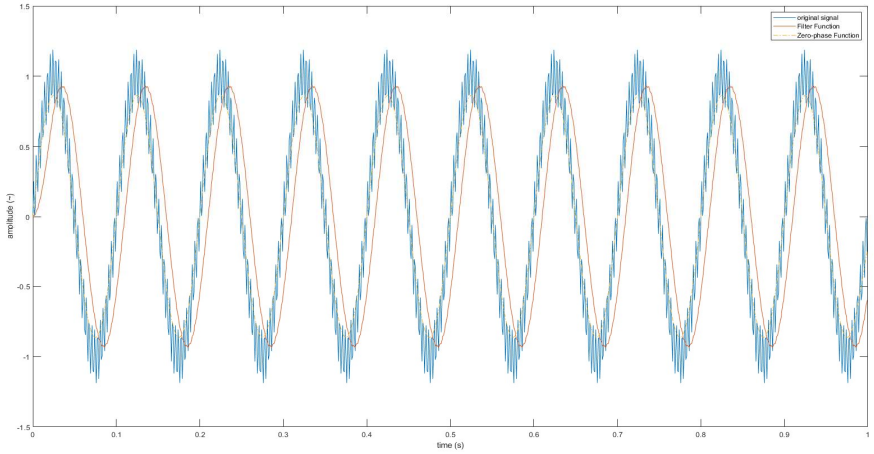


fig.7 Comparison of the signal pre and post-filtration using filter and zero-phase filtration

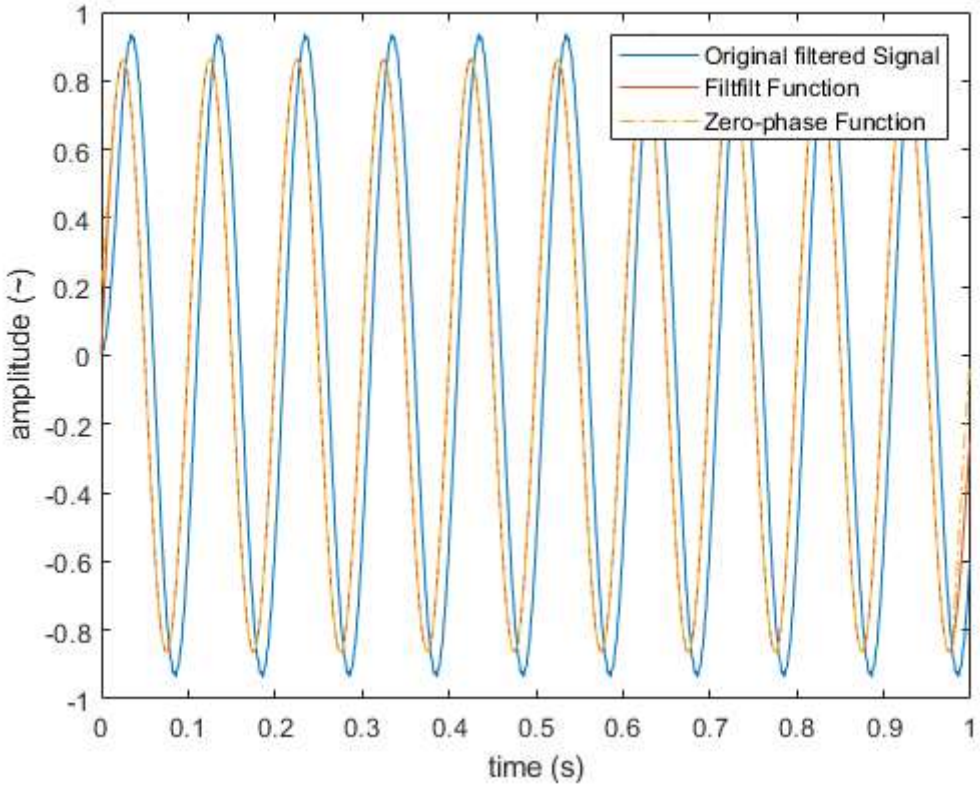


fig.8 Comparaison of the filtered signal using filter(), filtfilt() and zero-phase filtration.

The results obtained through filtering using filtfilt() and custom-implemented zero-phase filtration are identical since the filtfilt() function is simply a Matlab-implemented zero-phase filtration. We can see that the shift in the time introduced by the standard filter() function disappeared, although at the cost of slight amplitude change. In general zero-phase filtration is applied whenever the preservation of the original phase characteristics is crucial, but the tradeoff of this approach is increased computational cost and real-time processing constraints.

task 1.3:

In this task, we want to observe the changes in the filter characteristics when the window length changes. To this end, two new filters were designed with length of 11 and 51 samples, where each element has amplitudes of 1/11 and 1/51 respectively. Subsequently, the new filters were compared to the characteristics obtained in the previous example, and conclusions were drawn.

```
[h11,ph1]=freqz(h,1,N,fs);
[h22,ph2]=freqz(h2,1,N,fs);
[h33,ph3]=freqz(h3,1,N,fs);
h_fin=[h11,h22,h33];
ph=[ph1,ph2,ph3];
figure
hold on
for i= 1:1:3
    plot(ph(:,i),20*log10(abs(h_fin(:,i))));
end
legend('21 samples','11 samples','51 samples')
xlabel('Frequency (Hz)')
```

```
ylabel('Magnitude (dB)')
hold off
```

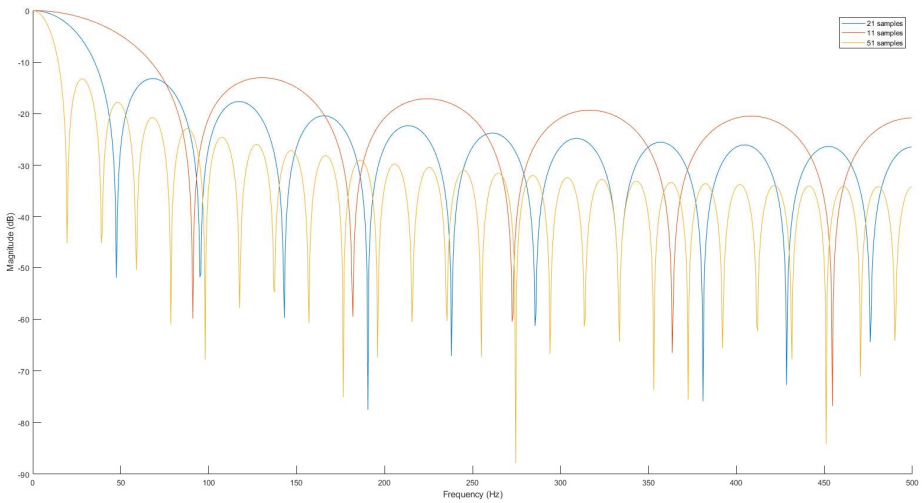


fig.9 Magnitude characteristics of the filters with 11, 21, and 51 samples

The window length affects the frequency response by influencing frequency resolution and smoothing. Longer windows generally provide better frequency resolution, as well as improved attenuation of unwanted frequencies, but they come with trade-offs in terms of computational complexity and potentially more artifacts in the time-domain characteristics of the signal.

Task 2:

This task aims to explore the characteristics of IIR (infinite impulse response) and FIR(finite impulse response) filters. Both filters were prepared using Matlab's inbuilt filterDesigner GUI, allowing for easier tuning of the filter's parameters.

The analyzed signal consists of four sine waves with different instantaneous frequencies at different time intervals.

```
% samping frequency
fs = 4000;
% time duration
tmax = 1-1/fs;
% time vector
t = 0:1/fs:tmax;
s = length(t);
% frequency vector
f = linspace(0,fs-fs/s,s);
% centra lfrequency of sine waves
w = [200,400,700,850];
% hann window 200 samples
mask = hann(200);
% time delay of the signals
inds = [1400,1000,1200,800];
% allocare memory for y
y = zeros(length(t),1);
% add sine waves
for i = 1:length(w)
 m = zeros(s,1);
 m(inds(i):inds(i)+length(mask)-1) = mask;
 y_temp = sin(2*pi*t*w(i)).*m.';
 y = y+y_temp.';
end
```

The time and frequency spectrums of the signal look as follows:

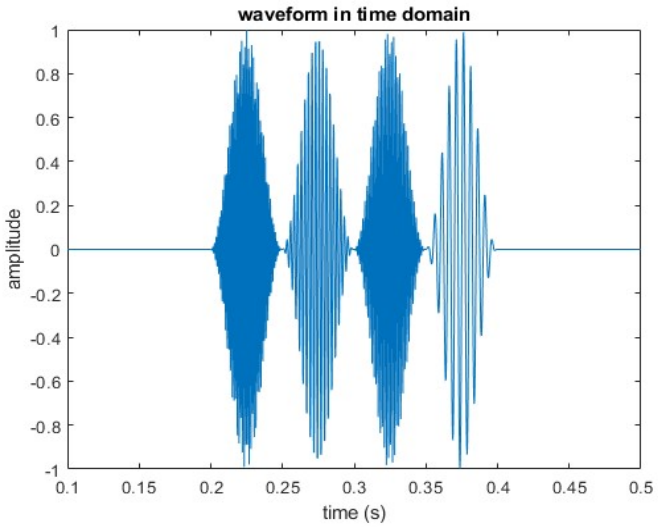


fig.10 Time domain signal

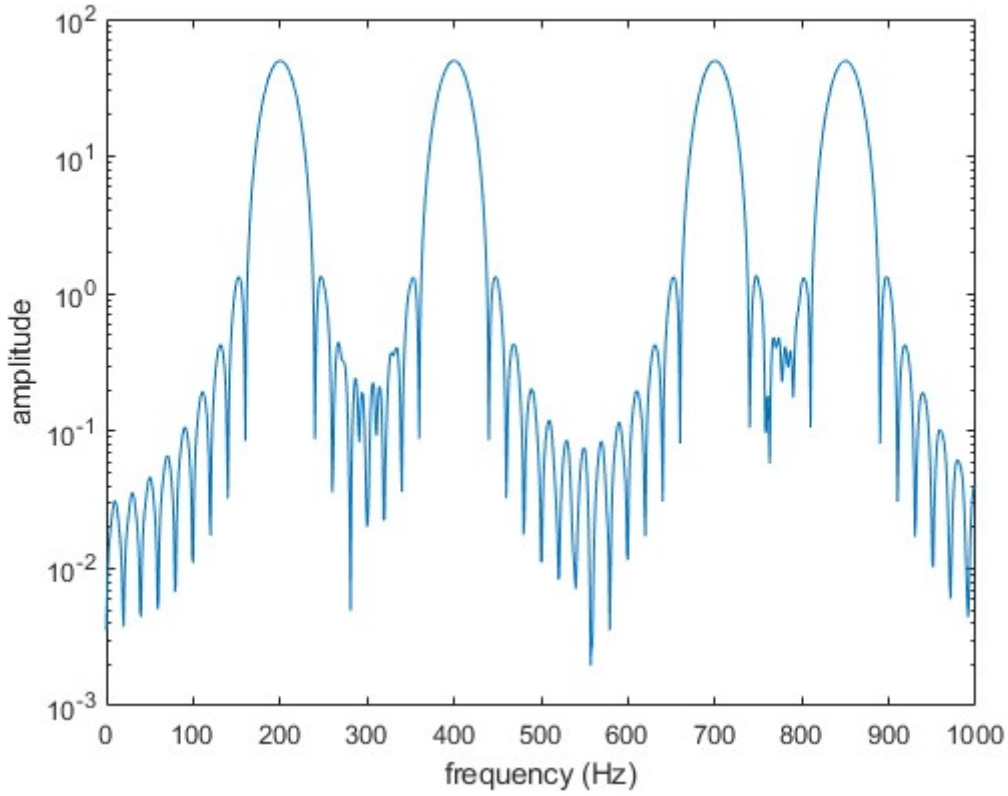


fig.11 Frequency spectrum of the original signal

Ex. 2.1:

In the first part of the task we aim to create an FIR lowpass filter with the following parameters:

parameter:	selected value:
window type	Kaiser
Astop	40dB
Apass	1dB
Fs	4000

table.1 Selected filter parameters

Adjusting the remaining two parameters (Fpass and Fstop) such that the filter cuts out the 850Hz center frequency wavelet and keeps the other wavelets intact.

After several tests, the following filter was obtained:

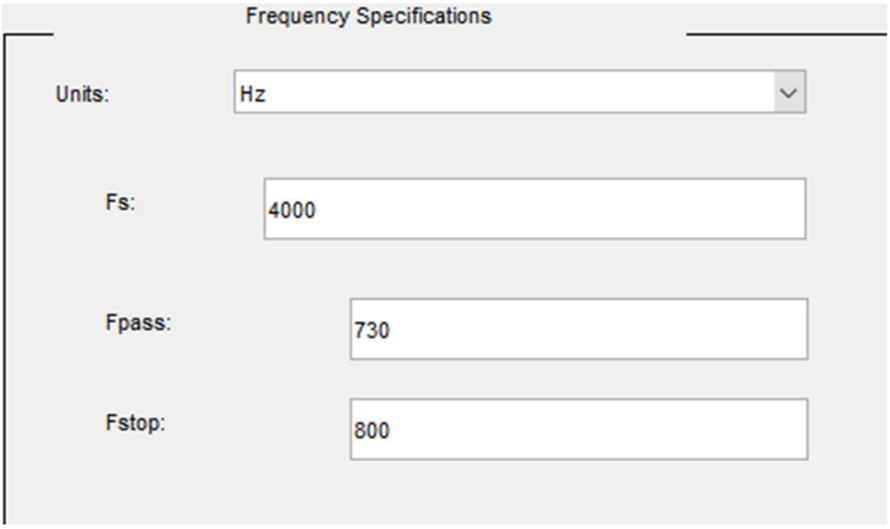


fig.12 Frequency characteristics

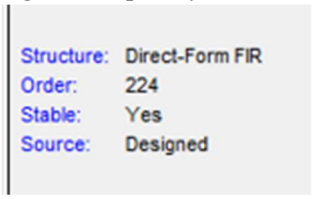


fig.13 Order of the filter

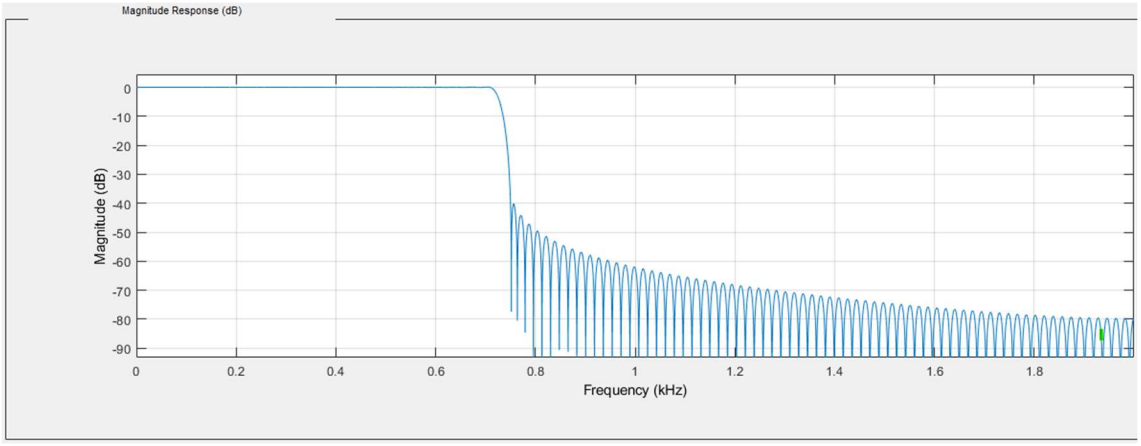


fig.14 magnitude characteristics of the FIR filter

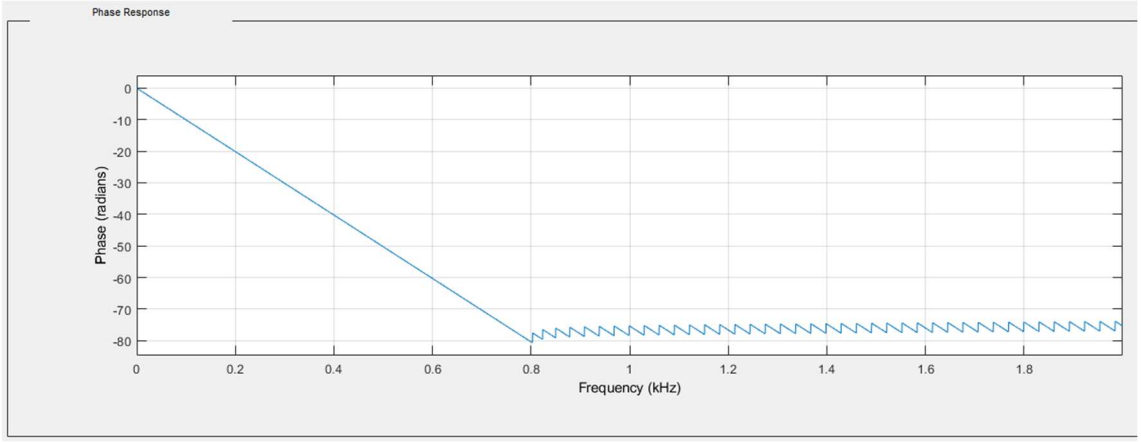


fig.15 phase characteristics of the FIR filter

From the plots above we can conclude that the designed FIR filter had no impact on the amplitude of the signal components present below ~750Hz. On the other hand, the phase of the signal has been linearly shifted and is equal to -80dB for any frequencies above 800Hz.

By applying the filter to the signal we obtain the following results:

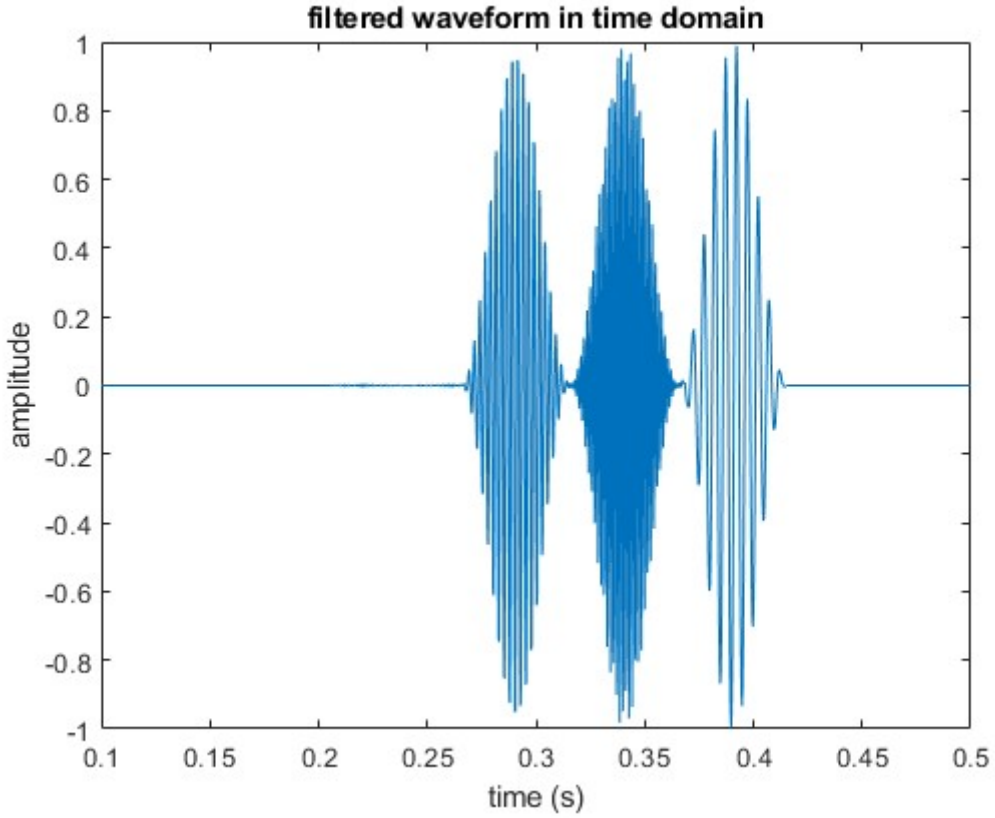


fig.16 Time domain signal after application of the FIR filter

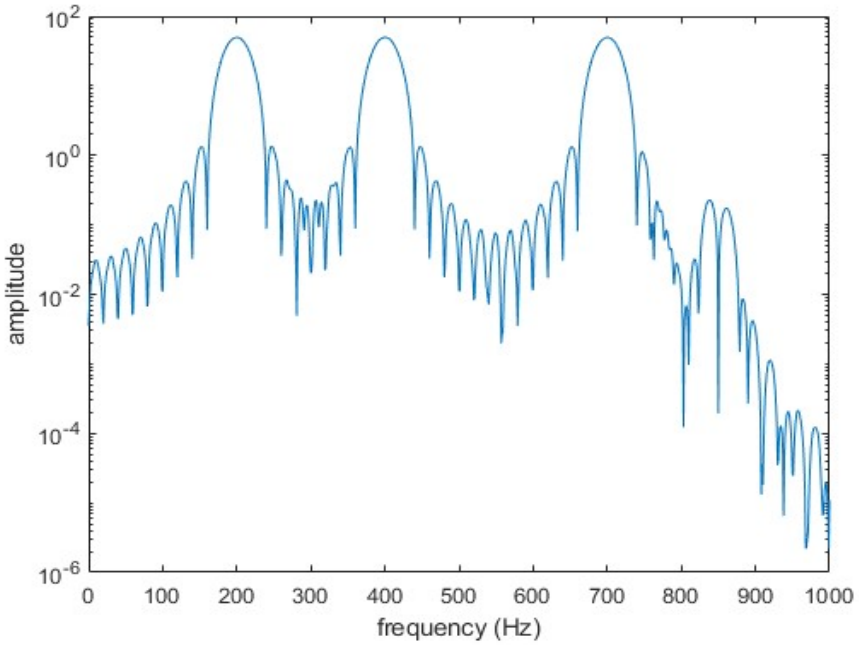


fig.17 Frequency spectrum of the signal after application of the FIR filter

From the fig.16 we can see that the wavelet corresponding to 850Hz previously present between 0.2 and 0.25s completely disappeared. On the other hand, the frequency spectrum (fig.17) plotted in semilogarithmic scale showcases that the designed FIR filter correctly attenuated the frequencies present at 850Hz, reducing them by a factor of 10^4
Therefore we can conclude that the filter was correctly designed and the selected Stop and Pass frequencies are appropriate.

Ex.2.2:

In this part of the task, we will repeat the procedure from the previous part of the task, but this time utilizing an IIR filter with the same parameters as the FIR filter. We aim to compare the performance of both filters in order to better understand the differences between them. Additionally, the IIR filtering will be applied using two different methods: classical filtration using the filter() function and zero-phase filtration using filtfilt() function.

The designed filter looks as shown in fig.18 and 19.

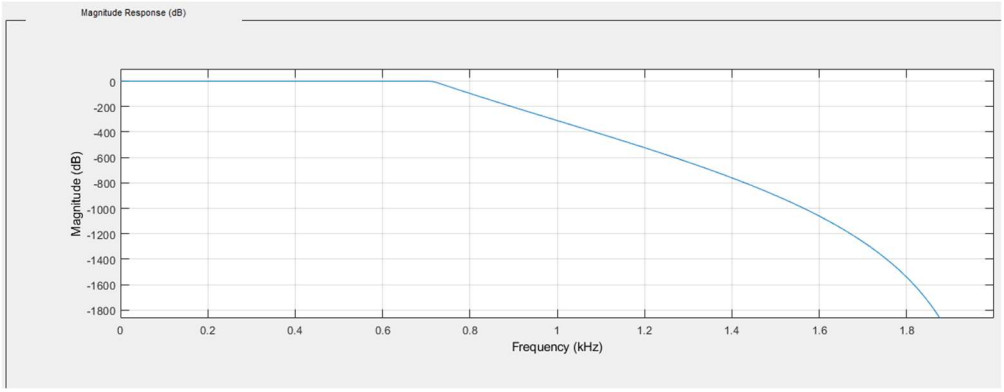


fig.18 magnitude characteristics of the IIR filter

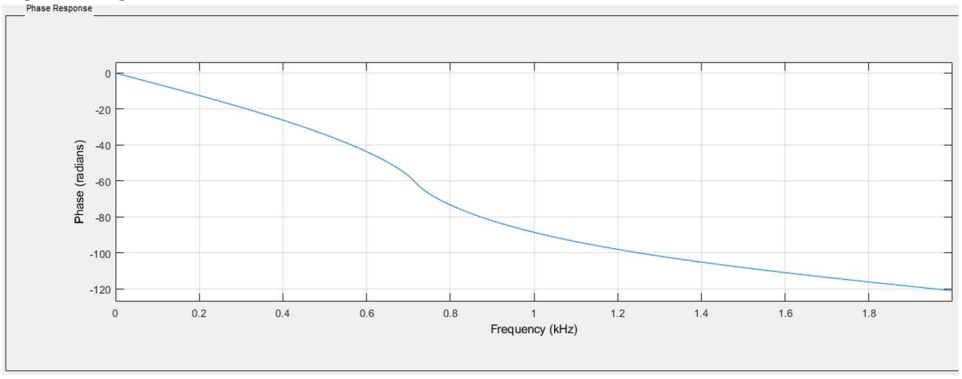


fig.19 Phase characteristics of the FIR filter

fig.20 Order of the IIR filter

Noticeably the order of the filter is almost 3 times smaller than the FIR equivalent. Additionally, we can observe a much larger phase shift. Once the filter was applied to the signal we obtained the following results when viewing both the time domain signal and frequency spectrum:

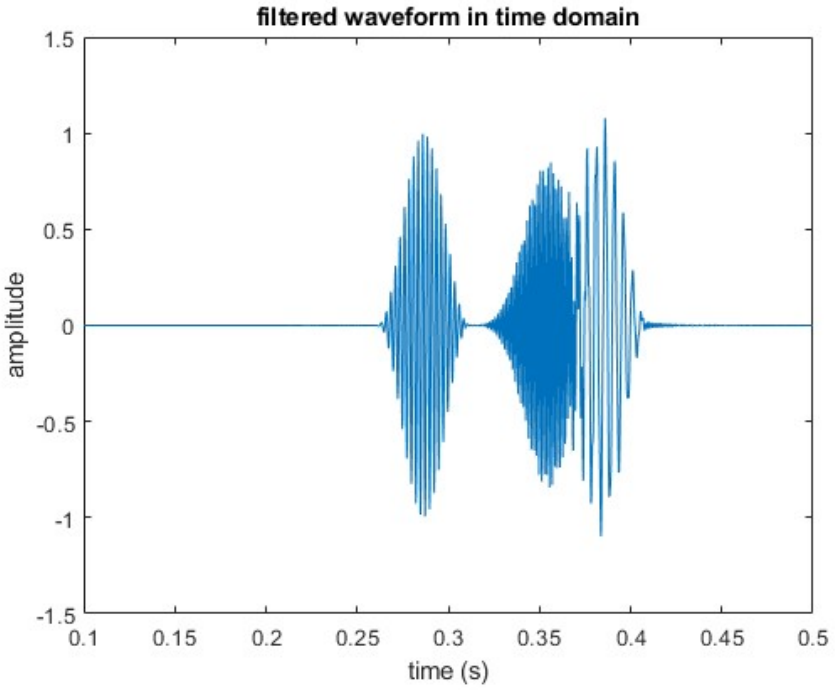


fig.21 Time domain signal after application of the IIR filter

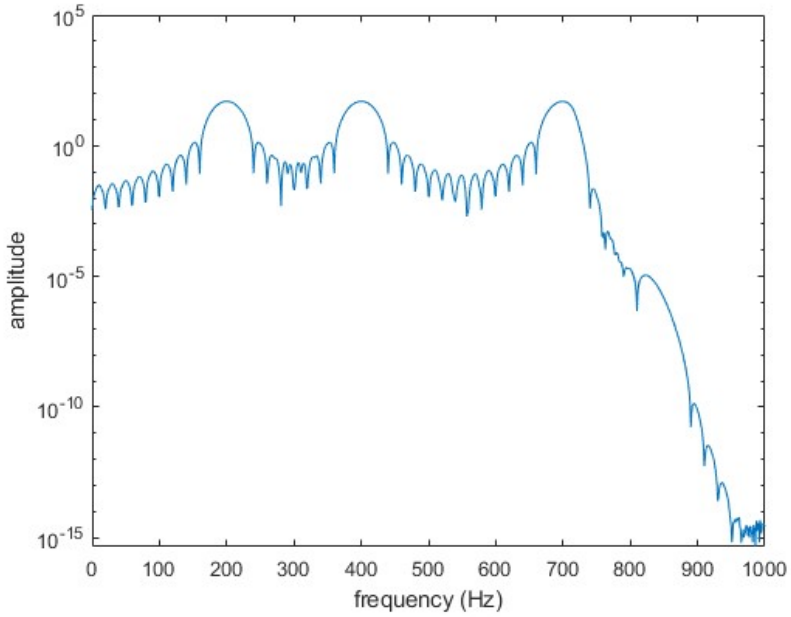


fig.22 Frequency spectrum of the signal after application of the FIR filter

As in the previous case the desired frequency peaks were attenuated, and the time domain signal no longer displayed the undesired wavelet, however, we have observed a significant shift in the remaining wavelets, causing them to overlap with each other.

Finally, we have compared the obtained results against the same filter applied to the signal utilizing zero-phase shift function filtfilt(). Once again the desired frequency was attenuated, but this time there was no overlapping in the remaining wavelets.

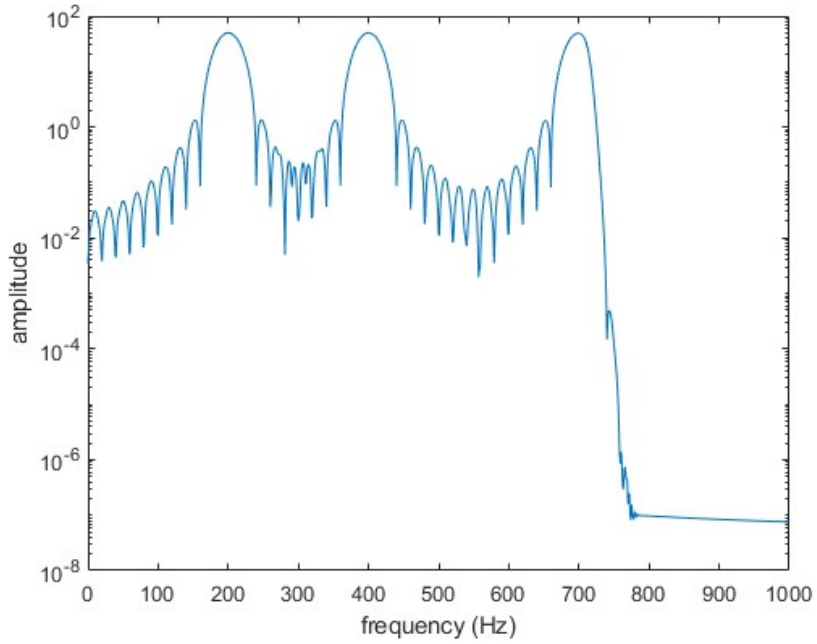


fig.23 Frequency spectrum of the signal after application of the IIR filter using filtfilt()

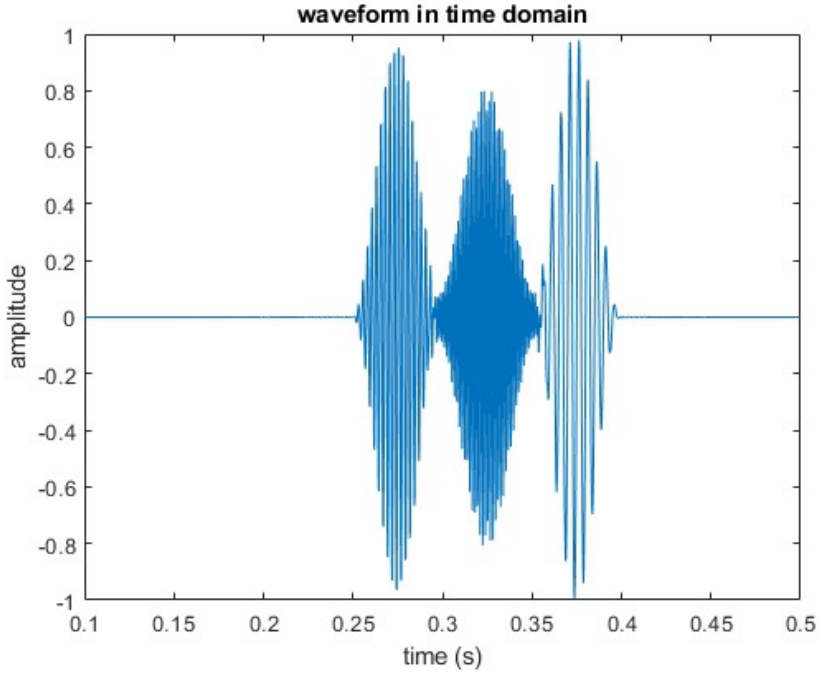


fig.24 Time domain signal after application of the IIR filter using filtfilt()

Utilizing zero-phase filtering prevented distortion of the signal in the time domain, although it caused a small amplitude change.

Conclusions:

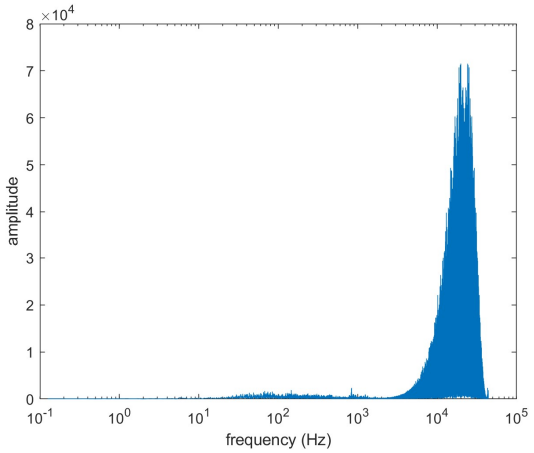
Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters are two common types of digital filters used for signal processing. One major difference between them lies in their impulse response characteristics. FIR filters have a finite duration impulse response, on the other hand, IIR filters possess an infinite impulse response. While FIR filters offer linear phase response and are suitable for precise control in the frequency domain, IIR filters are typically more computationally efficient due to feedback mechanisms but may introduce phase distortions. In the context of filtering a signal with characteristic frequencies FIR filters generally offer sharper cutoffs with less ripple in the passband but require more computational resources compared to IIR filters.

The influence of FIR and IIR filters on the remaining parts of the signal in the time domain can introduce certain effects on the signal beyond their intended filtering objectives. FIR filters typically have linear phase characteristics, meaning they introduce a constant delay across all frequencies (observed by comparing fig.10 and fig.16) resulting in a uniform shift in the timing of the signal, preserving the relative relationships between different components of the signal. However, FIR filters with sharp cutoffs may introduce ringing or overshoot artifacts in the time domain, especially if the filter order is high.

On the other hand, IIR filters, particularly those with complex poles, may introduce nonlinear phase distortions, causing a phase shift that varies with frequency. This can lead to distortion in the time domain (see fig.21).
In summary, while FIR filters generally offer more predictable behavior in the time domain, IIR filters may introduce more complex distortions due to their feedback nature and nonlinear phase response. Those side effects can be alleviated by carefully adjusting the filter parameters.

In particular, when it comes to the IIR filters their side effects can be minimized by applying zero-phase filtering which prevents the phase shift resulting in the distortion of the signal.

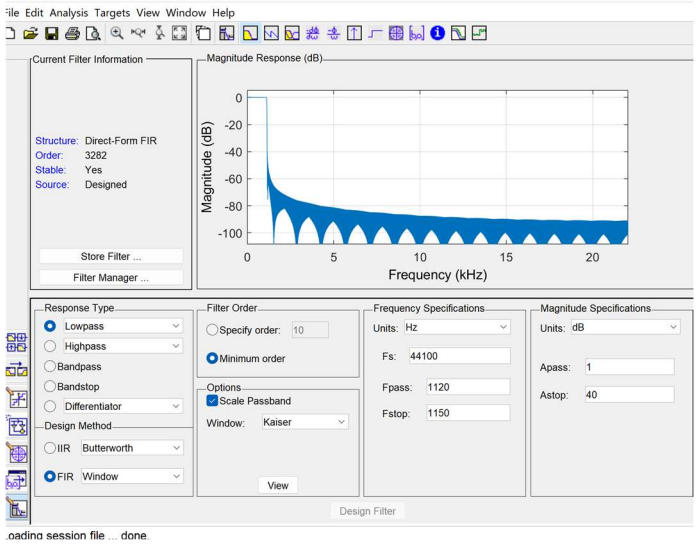
Task 3



This task aims to teach us how to denoise a signal and extract the relevant information.
The signal provided was very noisy with very high noise in high frequencies. The filter needed is a low-pass filter to remove all the high-frequency noise. The preferred filter is an FIR filter because preserving the original phase of the signal is important and computational resources are not a major constraint.
After multiple tires, the filter design is between a window FIR with the Kaiser function due to it being the most suitable option 1120 and 1150 Hz and the amplitude pass is between 1 and 40 dB.

```
%% Task 3
[data,fs]                    =
audioread(['ktoto.wav']);
%  Creating  the  fourier
transform
Y = fft(data(:,1));
Y2 = fft(data(:,2));
N = length(Y);
df  =  fs/N;  %  frequency
resolution
fv    =    (0:N-1)*df;    %
frequency vector

Nf = (abs(Y)/N*2);
figure(1)
semilogx(fv , abs(Y))
xlabel('frequency (Hz)')
ylabel('amplitude')
% Num - numerator of FIR
dataF(:,1) = filter(Hd,squeeze(data(:,1)));
dataF(:,2) = filter(Hd,squeeze(data(:,2)));
```



Conclusions:

After analyzing the signal and filtering it out we found out that the signal entails a famous movie phrase “You shall not pass!” by Gandalf the Grey in The Lord of the Rings movie. This character is played by Ian McKellen.