Lab 4: Solutions

Part 1: FIR and IIR filters and Z-Transform

1. **(Q)**: Suppose you have a filter with coefficients $b = [-2,1,0,-1,2]$defined for *k* = [0:4]. Write the analytical equation of the filtered signal y as a function of x.

   The analytical equation for the FIR filter is:

$$y[n] = -2x[n] + x[n-1] - 1x[n-3] + 2x[n-4]$$

2. **(Q)**: Now suppose that on top of the previous coefficients $b = [-2,1,0,-1,2]$ defined for *k* = [0:4], you have the coefficients $a = [3,2,1]$ defined for $l = [1:3]$. Write the analytical equation of the filtered signal y as a function of x and the previous values of y.

   The analytical equation for the IIR filter is:

$$\begin{aligned} y[n] = &-2x[n] + x[n-1] - 1x[n-3] + 2x[n-4] \\ &+3y[n-1] + 2y[n-2] + 3y[n-3] \end{aligned}$$

3. **(Q)**: Assume the system has an IIR form with the same coefficients as previously: $b = [-2,1,0,-1,2]$ and $a = [3,2,1]$. Take the Z-transform of both sides of the difference equation you wrote in the previous question by using the Z-transform tables given in the *Appendix*. Manipulate the equation to obtain the discrete transfer function $H_{IIR}(z) = \frac{Y(z)}{X(z)}$ symbolically. What is the transfer function $H_{FIR}(z)$ for the FIR form (where $a = [0,0,0]$)?

   We apply the "linearity" and "time shift" properties of the Z-transform to the analytical equation for the IIR filter, hence:

$$\begin{aligned} Y = &-2X + z^{-1}X - z^{-3}X + 2z^{-4}X \\ &+3z^{-1}Y + 2z^{-2}Y + z^{-3}Y \end{aligned}$$

   We rearrange the terms to obtain the desired transfer function $H_{IIR}(z) = \frac{Y(z)}{X(z)}$:

$$H_{IIR} = \frac{Y}{X} = \frac{-2 + z^{-1} - z^{-3} + 2z^{-4}}{1 - 3z^{-1} - 2z^{-2} - z^{-3}}$$

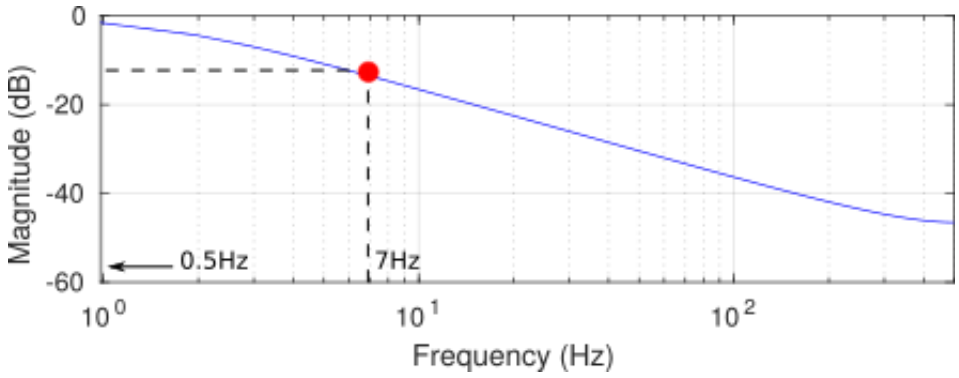
   In the FIR case, the transfer function $H_{FIR}(z) = \frac{Y(z)}{X(z)}$ *is simply given by:*

$$H_{FIR} = \frac{Y}{X} = -2 + z^{-1} - z^{-3} + 2z^{-4}$$

Part 2: Filtering of signals generated in Matlab

4. **(S)**: Create a signal made of two sines, with respective frequencies of 0.5 and 7 Hz, and amplitude of 1 for both. Add a simple low pass filter with a cutoff frequency (`Fc`) of 3 Hz. Compare the FFT of the original signal with the FFT of the filtered signal. Are there any differences? Explain the changes from the Bode plot.

   The filtered signal has a nearly unaffected 0.5Hz sinusoid, but the 7 Hz signal is strongly attenuated, because 7 Hz is larger than the cut-off frequency of the filter. The attenuation can be seen in the amplitude plot of the Bode plot:



The attenuation at 0.5Hz is close to 0dB (which means no attenuation) while the attenuation at 7Hz is around 12dB (note that the dB scale is exponential!)

5. **(S)**: Keep the same cutoff frequency and change to a high pass Butterworth filter of order 1. Compare the FFT of the original signal with the FFT of the filtered signal. Are there any differences? Explain the changes from the Bode plot.

   Now we see the reverse effect. The high-pass filter leaves the 7Hz sinusoid nearly untouched, while the 0.5Hz sinusoid is strongly attenuated. One can do an analysis on the Bode plot similarly to what was done previously.

6. **(S)**: Change the filter type back to a low pass Butterworth filter and run the script for filter orders of 1 and 3. Compare the two filtered signals. Are there any differences? Explain the changes from the Bode plots.

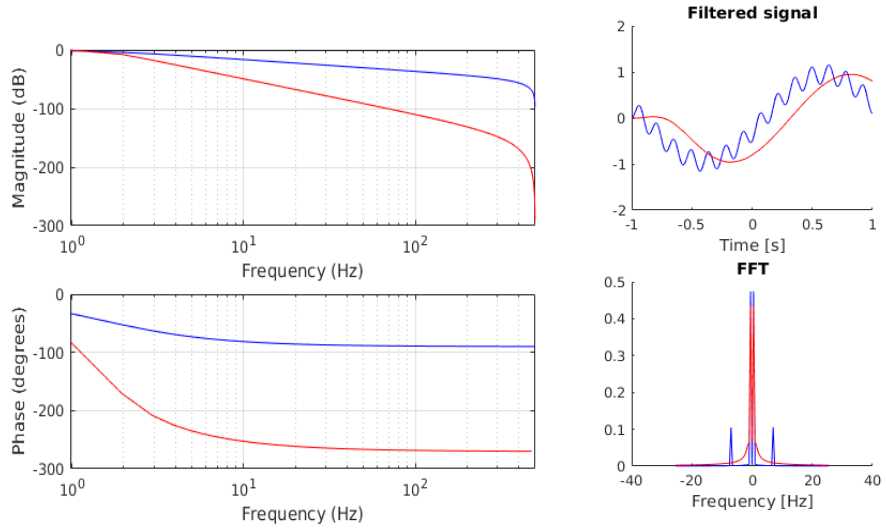
The 3rd order filter attenuates more the 7 Hz component, because its frequency response drops faster (as it can be observed in the Bode plots). Note that the filter of order 3 applied to the input signal gives an output signal with an FFT that contains residual frequencies. This is because the filter is being applied to a signal that is finite in time (from -1 to 1).

7. **(S)**: Increase the cutoff frequency to 100 Hz and compare the bode plots for a simple low pass filter and 1st and 3rd order low pass Butterworth filters. What is different in the frequency response? What are the differences in the phase response? Explain the changes from the Bode plot.

The simple low pass filter has the lowest and slowest attenuation of frequencies in the stopband. It also has the least phase distortion. However, the phase is not monotonic as the signal frequency increases. The Butterworth filters show a monotonic increase in phase shift and have a sharper and stronger increase in attenuation in the stopband. These qualities are amplified as we increase the Butterworth filter order.

8. **(Q)**: What parameter of the Butterworth filter do we have to change to have a better attenuation in high frequencies? What is the trade-off when changing that parameter? *(hint: check the phase plot of the filters that you designed)*

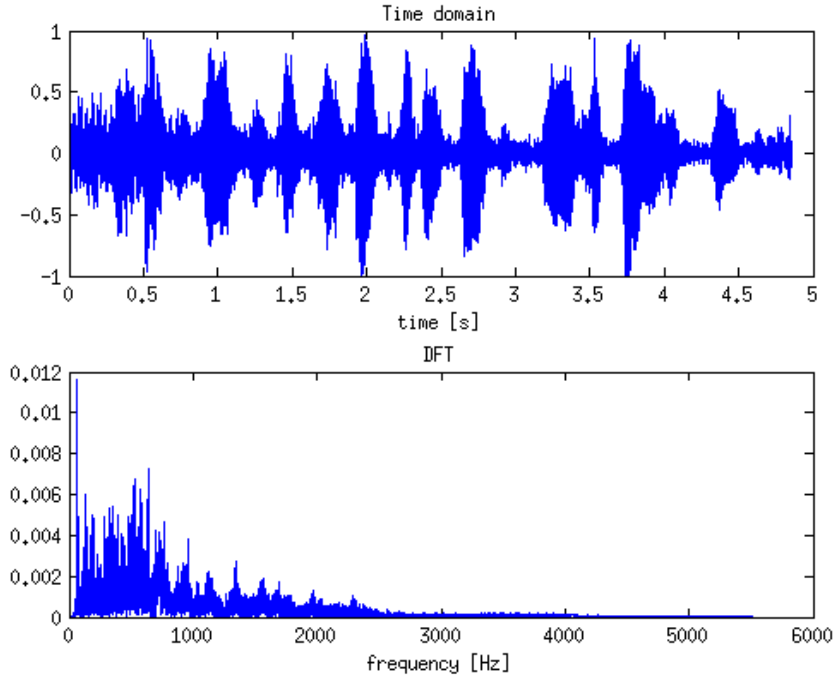
The order of the filter can be adjusted (increased) to increase the attenuation of high frequencies. If we plot the frequency domain and time domain plots for the Butterworth filter of order 1 and 3 together we get:



The blue lines correspond to the frequency response of the filter (left) of order 1 and the respective filtered signal (right). The red lines are related to the same information for the filter of order 3. It is possible to observe that, while the filter of order 3 yields better attenuation, it also produces a larger phase delay, which translates to a signal delay in time domain (as clearly seen in the top right plot).

Part 3: Filter and Fourier transform on sound samples

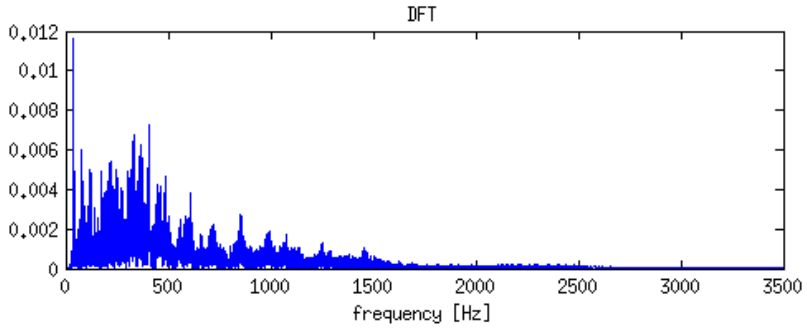
9. **(S)**: Use the provided Matlab function `analyzeSoundSignal(y,fs)` to look at the time and frequency domain of the signal you just played. How does the spectrum look like? Where can you find the voice components?



Almost all the spectrum is in the range [0-2500Hz], being most of its components in the range [0-1000Kz]. There is almost no continuous component.

10. **(S)**: Now play the same sound signal again setting higher and lower frequencies as `Fs` (e.g., 7000, 14000, 18000). How does the voice sound? Why does the pitch change?

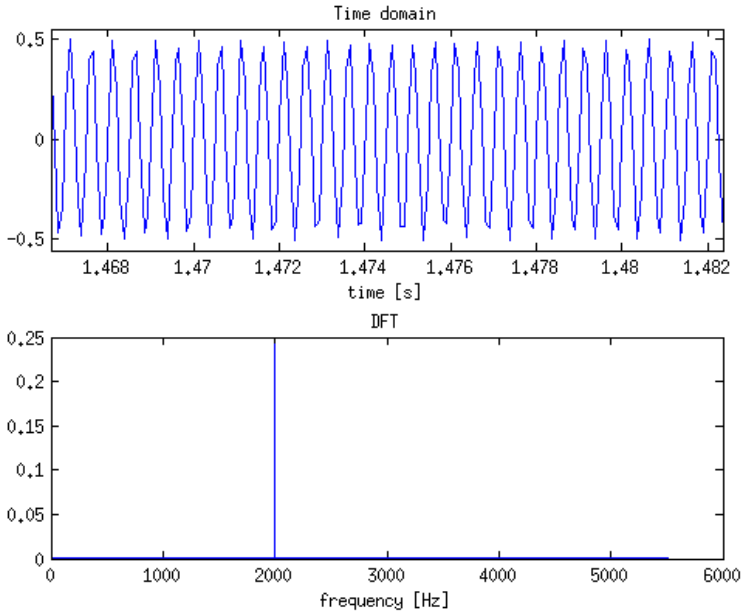
If you set a lower value of the sampling frequency then the player will reconstruct the signal and play it at lower speed, having as an effect a compression of the signal in the frequency domain and so sounding with a lower pitch. If you run `analyzeSoundSignal` *changing the* `dataFrequency` *in line 5, for instance to 7000Hz, you can observe the following spectrum (note that what has changed is the axis, although the signal looks the same).*



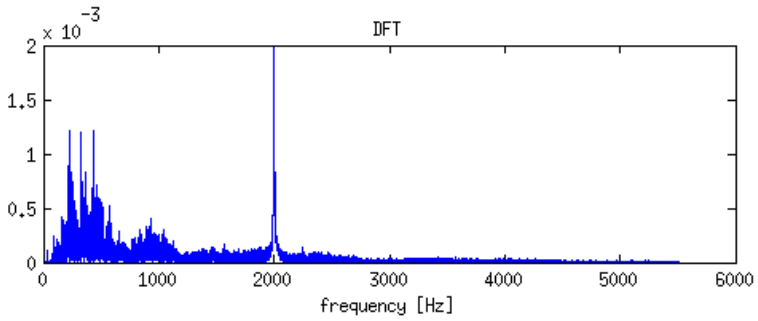
The opposite effect takes place if you set a higher sampling frequency, the spectrum will expand and the played signal will have a higher pitch.

11. **(S)**: Now load `sample2.wav`, listen to it (**watch out, sound might not be pleasant**) and look at how it looks in the time and frequency domain. The signal is composed from a dialogue from the same movie and an added signal. Can you guess which kind of signal is it? Where is it found in the frequency domain? *Hint: Use the provided Matlab function* `analyzeSoundSignal` *to look at the time and frequency domain of the signal and uncomment the line 18 to set the y axis as needed.*

The added signal is a sinusoidal signal of 2 KHz. We can see it clearly in the frequency domain, but also when zooming close in the time signal.



If we look in the frequency domain closer for y axes between 0 and 0.002, we can still appreciate a spectrum that looks similar to the one in sample1.wav. It is the spectrum of the hidden voice.



12. **(Q)**: As you could probably guess, `sample2.wav` is the sum a signal representing a voice and a sinusoid of 2 KHz. If we want to remove the sinusoid signal to better listen the speaker, what kind of filter would we need? Low-pass, high-pass, band stop or band pass? Consider that we might be able to discard sound components above 2 KHz and still be able to recognize a voice.

We want to filter out the sinusoid at 2 KHz, we could use a stop-band filter centered at that high frequency that will remove only that component. Another possibility is to use a low-pass filter to remove frequencies higher than 2 KHz. This low pass filter will not only remove the sinusoid at 2 KHz but also other voice components from the original signal. Still the voice would be recognized.

13. **(I)**: Change `Specify order` from `10` to `140`, then hit `Design Filter` to recompute the filter. What happens to magnitude response?

The maximum of the ripples in the magnitude drops from ca. – 10 dB to ca. -58 dB.

14. **(Q)**: Which filter is "better"? The one with order 10 or the one with order 140?

The filter of order 140 attenuates (filters) the high frequencies better.

15. **(Q)**: What could be the disadvantage of a filter with a high filter order? *(hint: look at the phase plot of the filters you designed in the* `filterDesigner`*).*

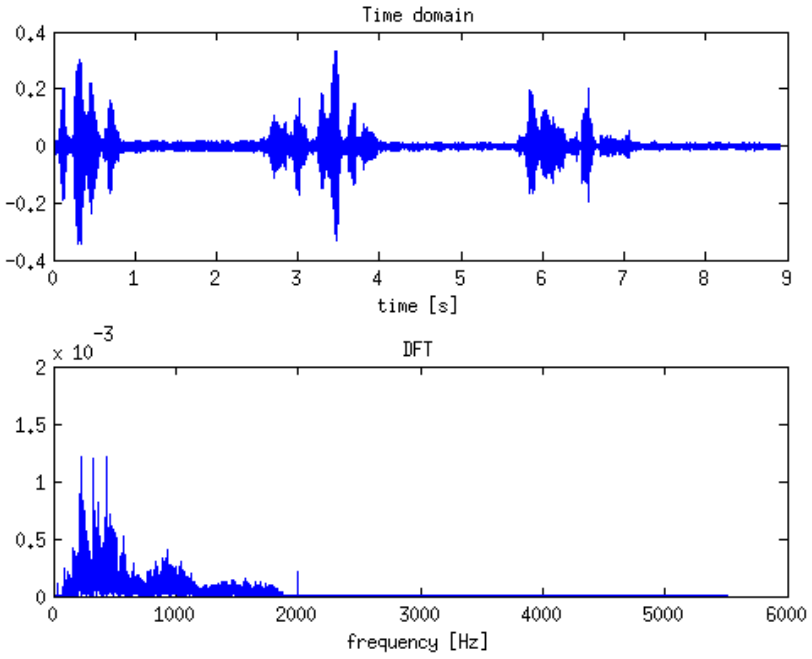
More processing is required (the processing requirement grows linearly with the order of the filter). Additionally, a higher order increases the phase delay of the filter response, which increases the latency of the filter (the filtered signal is delayed with respect to the original signal). This can be a major drawback, especially if the filtered signal is to be used as a control input.

16. **(S)**: What do these 141 coefficients stored in the `Num` variable correspond to in the FIR equation of Part 1?

The coefficients stored in `Num` *correspond to the coefficients* b_k *in the FIR filter equation.*

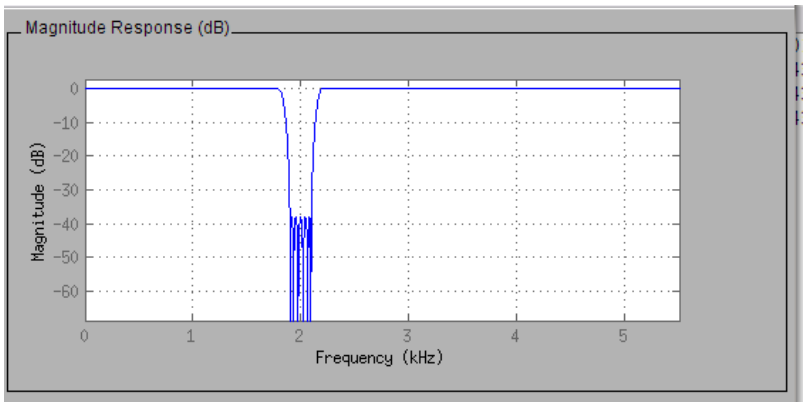
17. **(S)**: Filter the data using the MATLAB routine `filter_data.m`, play the resulting signal and look at its time and frequency domains. Can you listen properly to the voice? Is the tone at 2 KHz still present? Compare also with the original signal.

Now the voice can be much better understood, although if we pay attention there is still the sound at 2 KHz. It can be also appreciated when looking at the frequency domain graph. In addition, we see how the frequencies of the voice above 2 KHz have been also removed.

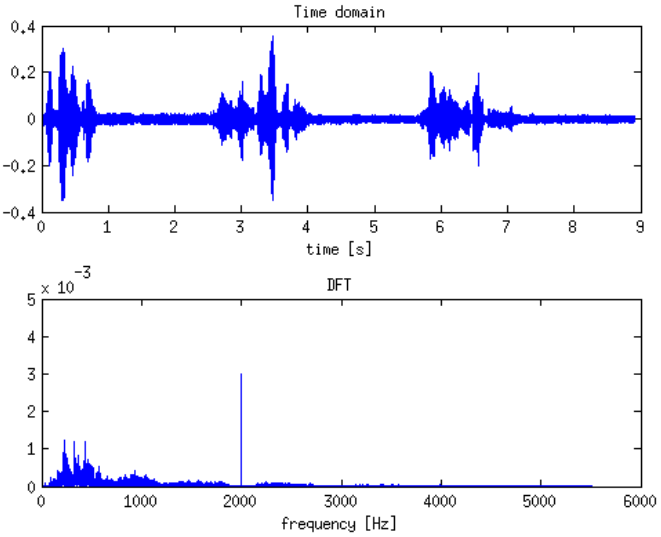


18. **(S)**: Find your desired filter by tuning if needed the above parameters, export it as previously and repeat Question **17** for the new filter. How does the frequency domain plot compare for the three signals (original, filtered in Question **17** and filtered here)?

With the proposed setup the magnitude response of the filter looks as follows. It indicates that the signal at 2 KHz is less attenuated (40 dB) than with the low pass filter. We could increase the order to increase the attenuation. On the other hand it keeps other frequencies above 2 KHz.

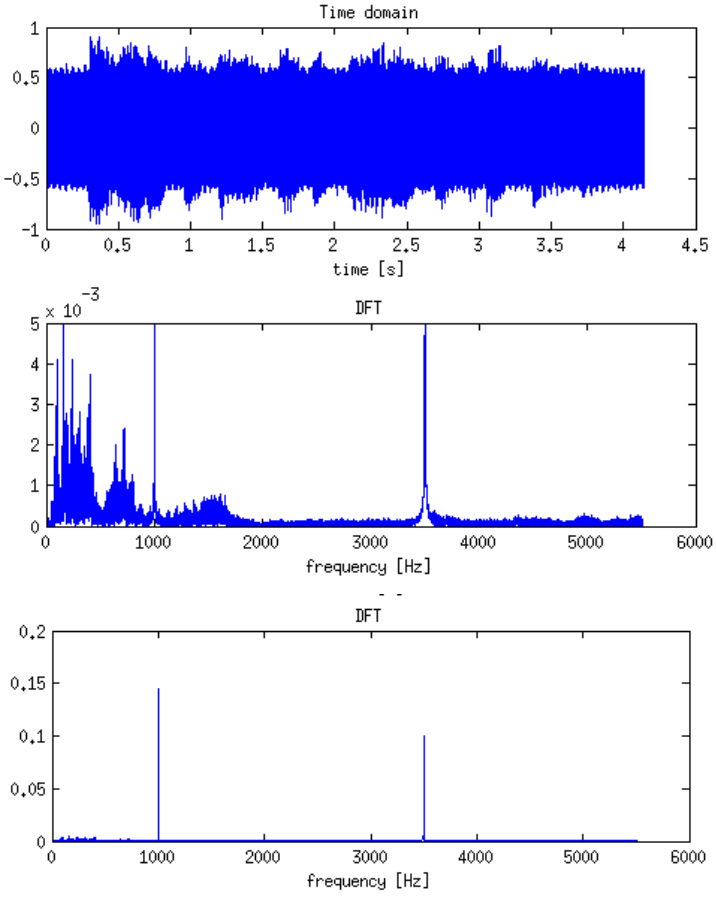


If we look at the filtered signal we observe the same behavior. That sinusoid signal is a little stronger and frequencies above 2 KHz are still present.



19. **(B)**: Lastly you will work with `sample3.wav` file which corresponds to a voice of a well-known movie with two sine signals of different frequencies added. Using `analyzeSoundSignal`, determine these two frequencies and later implement the filtering needed to clean the signal. *Hint: you might want to design and apply two different filters using the* `filterDesigner` *and apply them sequentially.* Do you know to whom the voice corresponds?

Looking at the spectrum of the signal we can see that there are to peaks at 1 KHz and 3.5 KHz.



We then want to implement two filters. The first a stop band filter at 1 KHz, and the second could be a low pass filter or a stop band filter to remove the component at 3.5 KHz or above it. We have to apply the two filters sequentially:

```
>> [y3, fs]  = audioread('sample3.wav');
>> analyzeSoundSignal(y3, fs)
>> y3_filt1 = filter_data(y3,Num_fil1);
>> y3_final = filter_data(y3_filt1,Num_fil3);
```

The voice is HAL from 2001: A Space Odyssey