Norwegian University of Science and Technology
Department of Electronics and Telecommunications

TTT4120 Digital Signal Processing - Suggested solution for problem set 9 (Python)

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
import pandas as pd
from scipy.io.wavfile import read
import time
import sounddevice as sd
```

Problem 1

(a).

The impulse response can be found as follows.

$$\begin{aligned}
h_d(n) &= \text{IDTFT} H_d(f) \\
&= \int_{-\frac{1}{2}}^{\frac{1}{2}} H_d(f) e^{j2\pi fn} df \\
&= \int_{-f_c}^{f_c} e^{j2\pi fn} df \\
&= \frac{1}{2\pi jn}\left[e^{j2\pi fn}\right]_{-f_c}^{f_c}, \; n \neq 0 \\
&= \frac{e^{j2\pi f_c n} - e^{-j2\pi f_c n}}{2\pi jn}, \; n \neq 0 \\
&= \frac{\sin(2\pi f_c n)}{\pi n}, \; n \neq 0
\end{aligned}$$

For $n = 0$, we get

$$h_d(0) = \int_{-f_c}^{f_c} 1 df = 2f_c$$

(b).

First, in order to obtain a causal filter after windowing, we shift $h_d(n) = (N-1)/2$ samples to the right (we assume that N is an odd number). Then, we multiply by $w(n)$, and obtain the following unit sample response

$$\begin{aligned}
h(n) &= h_d\left(n - \frac{N-1}{2}\right) w(n) \\
&= \begin{cases} \frac{\sin[2\pi f_c(n - \frac{N-1}{2})]}{\pi(n - \frac{N-1}{2})} \cdot w(n) & n \neq \frac{N-1}{2} \\ 2f_c \cdot w\left(\frac{N-1}{2}\right) & n = \frac{N-1}{2} \end{cases}
\end{aligned}$$

(c).

The required python function can be implemented as follows.

In [2]:

```
fc = 0.2
N = 31

def problem1(w, fc):
    N = len(w)
    h = np.zeros(N)
    for i in range(int((N-1)/2)):
        h[i] = w[i]*np.sin(2*np.pi*fc*(i-(N-1)/2))/(np.pi*(i - (N-1)/2))
    for i in range(int((N+1)/2), N):
        h[i] = w[i]*np.sin(2*np.pi*fc*(i-(N-1)/2))/(np.pi*(i - (N-1)/2))
    h[int((N-1)/2)] = 2*fc*w[int((N-1)/2)]
    return h
```

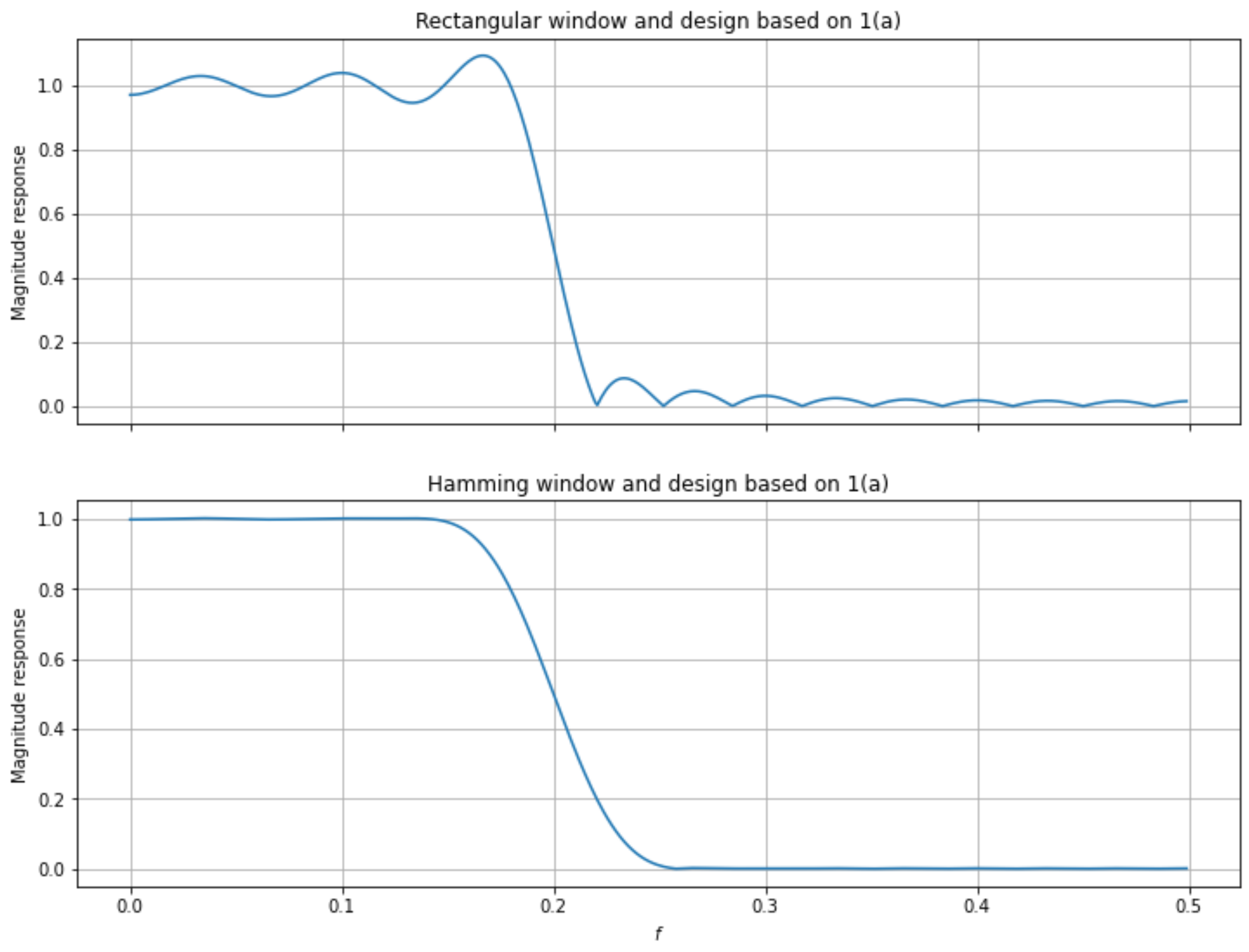
(d).

The magnitude response for the filters are shown in figure 1. By using rectangular window the resulting filter will have smaller transient band but at the same time it will have bigger passband and stopband ripples. The following code is used to test the function in 1(c) and plot figure 1.

In [3]:

```
fc = 0.2
N = 31

w1 = np.ones(N)
w2 = np.hamming(N)

h1 = problem1(w1, fc)
h2 = problem1(w2, fc)

omega1, H1 = signal.freqz(h1, 1)
omega2, H2 = signal.freqz(h2, 1)

# plot
plt.rcParams['figure.figsize'] = [12, 9]
fig, axs = plt.subplots(2)
fig.suptitle('Figure 1: Filters designed using the windowing method from 1
axs[0].plot(omega1/(2*np.pi), np.abs(H1))
axs[0].grid()
axs[0].set_title('Rectangular window and design based on 1(a)')
axs[0].set(ylabel = 'Magnitude response')
axs[0].label_outer()
axs[1].plot(omega2/(2*np.pi), np.abs(H2))
axs[1].grid()
axs[1].set_title('Hamming window and design based on 1(a)')
axs[1].set(xlabel = '$f$', ylabel = 'Magnitude response')
plt.show()
```

Figure 1: Filters designed using the windowing method from 1(a)

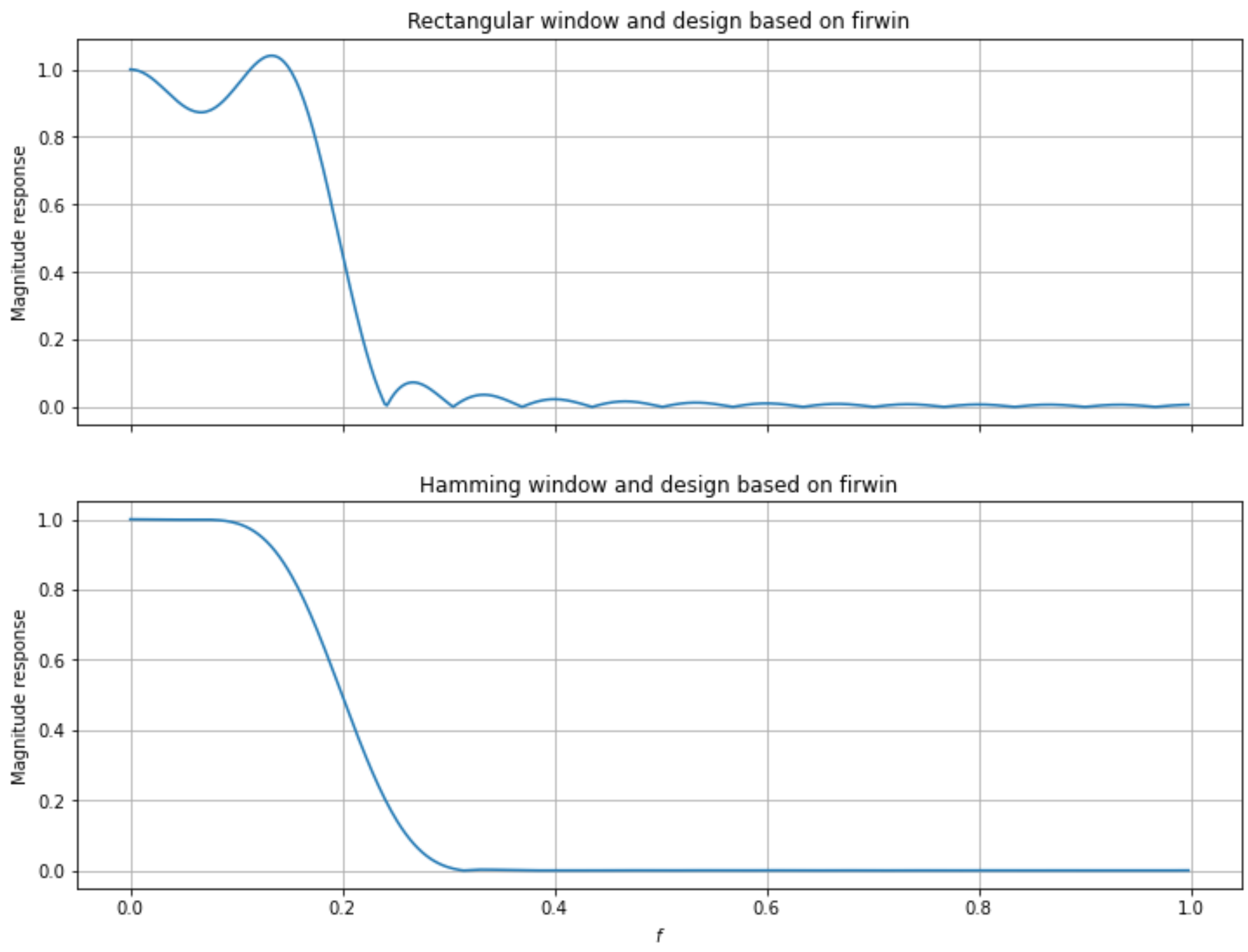


(e).

We can use the following code to generate and plot the magnitude responses of the filters. The results when plotted are equal to those from 1(c).

In [4]:
```
h1 = signal.firwin(N, 0.2, window='boxcar')
h2 = signal.firwin(N, 0.2, window='hamming')

omega1, H1 = signal.freqz(h1, 1)
omega2, H2 = signal.freqz(h2, 1)

# plot
plt.rcParams['figure.figsize'] = [12, 9]
fig, axs = plt.subplots(2)
fig.suptitle('Figure 2: Filters designed using the windowing method using s
axs[0].plot(omega1/(np.pi), np.abs(H1))
axs[0].grid()
axs[0].set_title('Rectangular window and design based on firwin')
axs[0].set(ylabel = 'Magnitude response')
axs[0].label_outer()
axs[1].plot(omega2/(np.pi), np.abs(H2))
axs[1].grid()
axs[1].set_title('Hamming window and design based on firwin')
axs[1].set(xlabel = '$f$', ylabel = 'Magnitude response')
plt.show()
```

Figure 2: Filters designed using the windowing method using signal.firwin function



Problem 2

(a).

The cut-off frequency can be found by solving $|H_a(\Omega_c)| = \frac{1}{\sqrt{2}}$. Thus, we get

$$\begin{aligned}
\left|\frac{1/RC}{j+\Omega_c+1/RC}\right| &= \frac{1}{\sqrt{2}} \\
\frac{1/RC}{|j+\Omega_c+1/RC|} &= \frac{1}{\sqrt{2}} \\
\sqrt{\Omega_c^2+(1/RC)^2} &= \frac{\sqrt{2}}{RC} \implies \\
\Omega_c^2+\frac{1}{(RC)^2} &= \frac{2}{(RC)^2} \implies \\
\Omega_c &= \frac{1}{RC}
\end{aligned}$$

since Ω_c is, by definition, a positive value.

(b).

Since $s = j\Omega$ is mapped into $z = e^{j\omega}$ by the bilinear transform, we have that

$$j\Omega = \frac{2}{T}\frac{1-e^{-j\omega}}{1+e^{-j\omega}}$$

Now, we multiply both the numerator and denominator on the right by $e^{j\omega/2}$, and get the following.

$$\begin{aligned}
\Omega &= \frac{2}{T}\frac{1}{j}\frac{(1-e^{-j\omega})e^{j\omega/2}}{(1+e^{-j\omega})e^{j\omega/2}} \\
&= \frac{2}{T}\frac{1}{j}\frac{e^{j\omega/2}-e^{-j\omega/2}}{e^{j\omega/2}+e^{-j\omega/2}} \\
&= \frac{2}{T}\frac{e^{j\omega/2}-e^{-j\omega/2}}{j}\frac{1}{e^{j\omega/2}+e^{-j\omega/2}} \\
&= \frac{2}{T}\frac{2\sin(\omega/2)}{2\cos(\omega/2)} \\
&= \frac{2}{T}\tan(\frac{\omega}{2})
\end{aligned}$$

(c).

We can use the frequency transformation expression to get the following relationship between analog and digital cut-off frequency

$$\Omega_c = \frac{2}{T}\tan(\frac{\omega_c}{2})$$

In addition, we know that $\Omega_c = 1/RC$. Thus, we get the following.

$$\begin{aligned} H(z) &= H_a(s)\big|_{s=\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}} \\ &= \frac{1/RC}{\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} + 1/RC} \\ &= \frac{\Omega_c}{\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} + \Omega_c} \\ &= \frac{\frac{2}{T}\tan(\frac{\omega_c}{2})}{\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} + \frac{2}{T}\tan(\frac{\omega_c}{2})} \\ &= \frac{\tan(\frac{\omega_c}{2})}{\frac{1-z^{-1}}{1+z^{-1}} + \tan(\frac{\omega_c}{2})} \\ &= \frac{\tan(\frac{\omega_c}{2})(1+z^{-1})}{1-z^{-1}+\tan(\frac{\omega_c}{2})+\tan(\frac{\omega_c}{2})z^{-1}} \\ &= \frac{\tan(\frac{\omega_c}{2})(1+z^{-1})}{1+\tan(\frac{\omega_c}{2})+(\tan(\frac{\omega_c}{2})-1)z^{-1}} \\ &= \frac{\frac{\tan(\frac{\omega_c}{2})}{\tan(\frac{\omega_c}{2})+1}1+z^{-1}}{\frac{\tan(\frac{\omega_c}{2})-1}{\tan(\frac{\omega_c}{2})+1}z^{-1}} \end{aligned}$$

By inserting $\omega_c = 0.2\pi$ we get

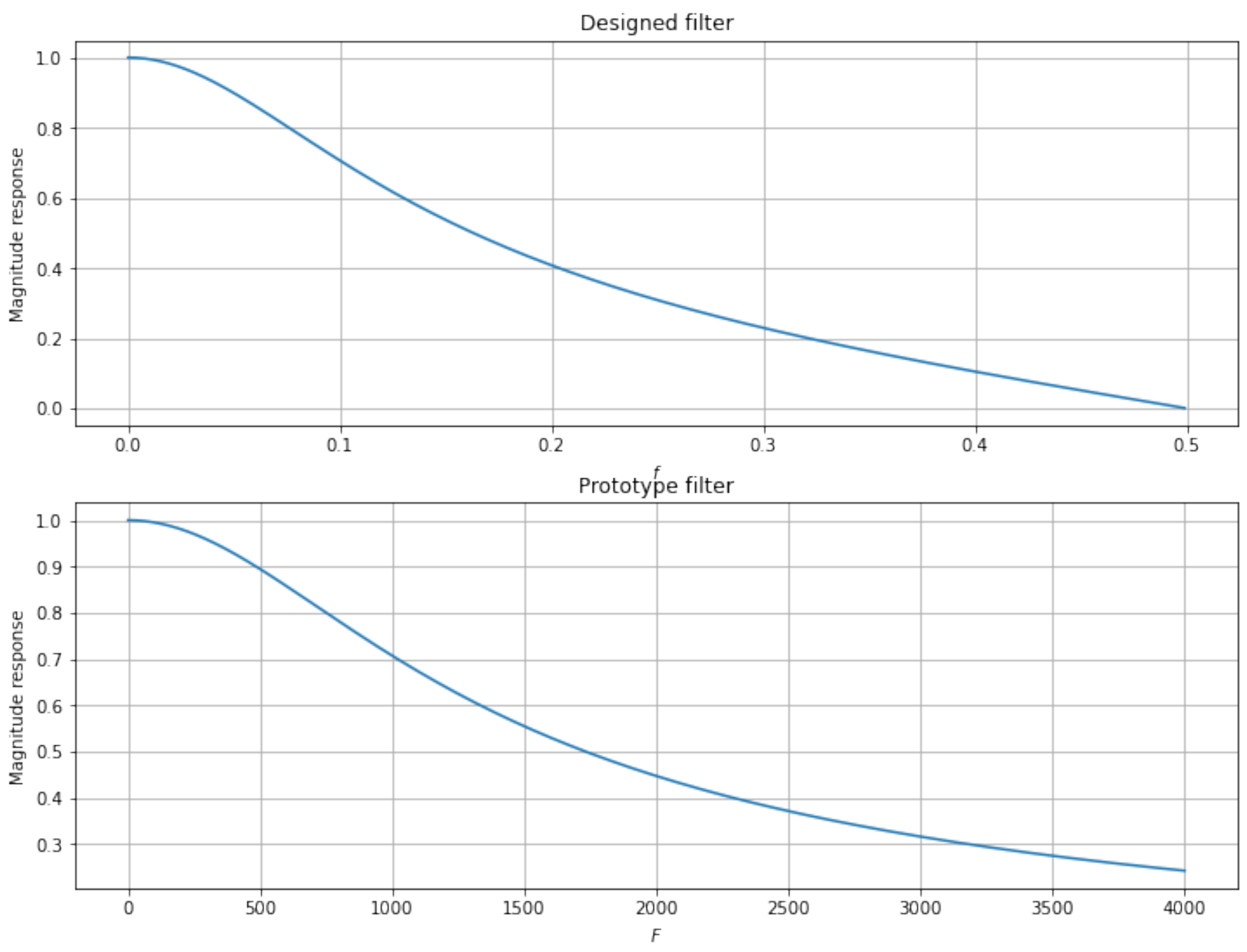
$$H(z) = \frac{0.245(1+z^{-1})}{1-0.51z^{-1}}.$$

Plots of the magnitude responses of the analog prototype filter, and the resulting digital filter are shown in figure 3. The magnitude responses are plotted as functions of F and f , respectively.

In [5]:

```
b = [0.245, 0.245]
a = [1, -0.51]

omega1, H1 = signal.freqz(b, a)

omega_c = 2000*np.pi
Hf = (omega_c)/(1j*2*np.pi*np.linspace(0, 4000, 500) + omega_c)


# plot
plt.rcParams['figure.figsize'] = [12, 9]
fig, axs = plt.subplots(2)
fig.suptitle('Figure 3: Magnitude responses of the prototype filter and the
axs[0].plot(omega1/(2*np.pi), np.abs(H1))
axs[0].grid()
axs[0].set_title('Designed filter')
axs[0].set(xlabel = '$f$', ylabel = 'Magnitude response')
axs[1].plot(np.linspace(0, 4000, 500), np.abs(Hf))
axs[1].grid()
axs[1].set_title('Prototype filter')
axs[1].set(xlabel = '$F$', ylabel = 'Magnitude response')
plt.show()
```

Figure 3: Magnitude responses of the prototype filter and the designed filter.



Note that for the analog filter, the frequency axis is not uniquely specified since we have not specified T. For the plot in figure 3, we have chosen an analog prototype filter with cut-off frequency $F_c = 1000$ Hz ($\Omega_c = 2000\pi$rad/s). The corresponding value of T can be found as follows.

$$2000\pi = \frac{2}{T}\tan\left(\frac{0.2\pi}{2}\right)$$
$$T = \frac{2}{2000\pi}\tan\left(\frac{0.2\pi}{2}\right) \approx 1.75 \cdot 10^{-6}$$

We see that the resulting digital filter is a lowpass filter with a similar shape as the prototype analog filter. However the two magnitude spectra differ especially at higher frequencies due to the nonlinear compression of the physical frequency axis into digital frequencies in the range $[0, \frac{1}{2}]$. Furthermore, the magnitude response of the resulting digital filter is approximately equal to $\frac{1}{\sqrt{2}} \approx 0.707$ at $f_c = 0.1$, which corresponds to $\omega_c = 2\pi f_c = 0.2\pi$, as required in the design specifications. The design specification have thus been met.

Problem 3

(a).

The magnitude response of a second order Butterworth filter is given by

$$|H_a(\Omega)| = \frac{1}{\sqrt{1 + (\Omega/\Omega_c)^4}},$$

where Ω_c is the cut-off frequency of the filter. Therefore we have to show that the magnitude response of the given filter can be written as

$$|H_a(\Omega)| = \frac{1}{\sqrt{1 + \Omega^4}},$$

The frequency response is given by

$$H_a(\Omega) = H_a(s)|_{s=j\Omega} = \frac{1}{-\Omega^2 + \sqrt{2}j\Omega + 1},$$

and the magnitude response is thus given by

$$|H_a(\Omega)| = \frac{1}{|-\Omega^2 + \sqrt{2}j\Omega + 1|} = \frac{1}{\sqrt{(1-\Omega^2)^2 + 2\Omega^2}} = \frac{1}{\Omega^4 + 1}$$

Alternatively we can find the cut-off frequency by solving

$$\begin{gathered}
\left|\frac{1}{-\Omega_c^2 + \sqrt{2}j\Omega_c + 1}\right| = \frac{1}{\sqrt{2}} \\
\left|-\Omega_c^2 + \sqrt{2}j\Omega_c + 1\right| = \sqrt{2} \\
\sqrt{(1-\Omega^2)^2 + 2\Omega^2} = \sqrt{2} \\
\sqrt{\Omega_c^4 + 1} = \sqrt{2} \\
\Omega_c = 1
\end{gathered}$$

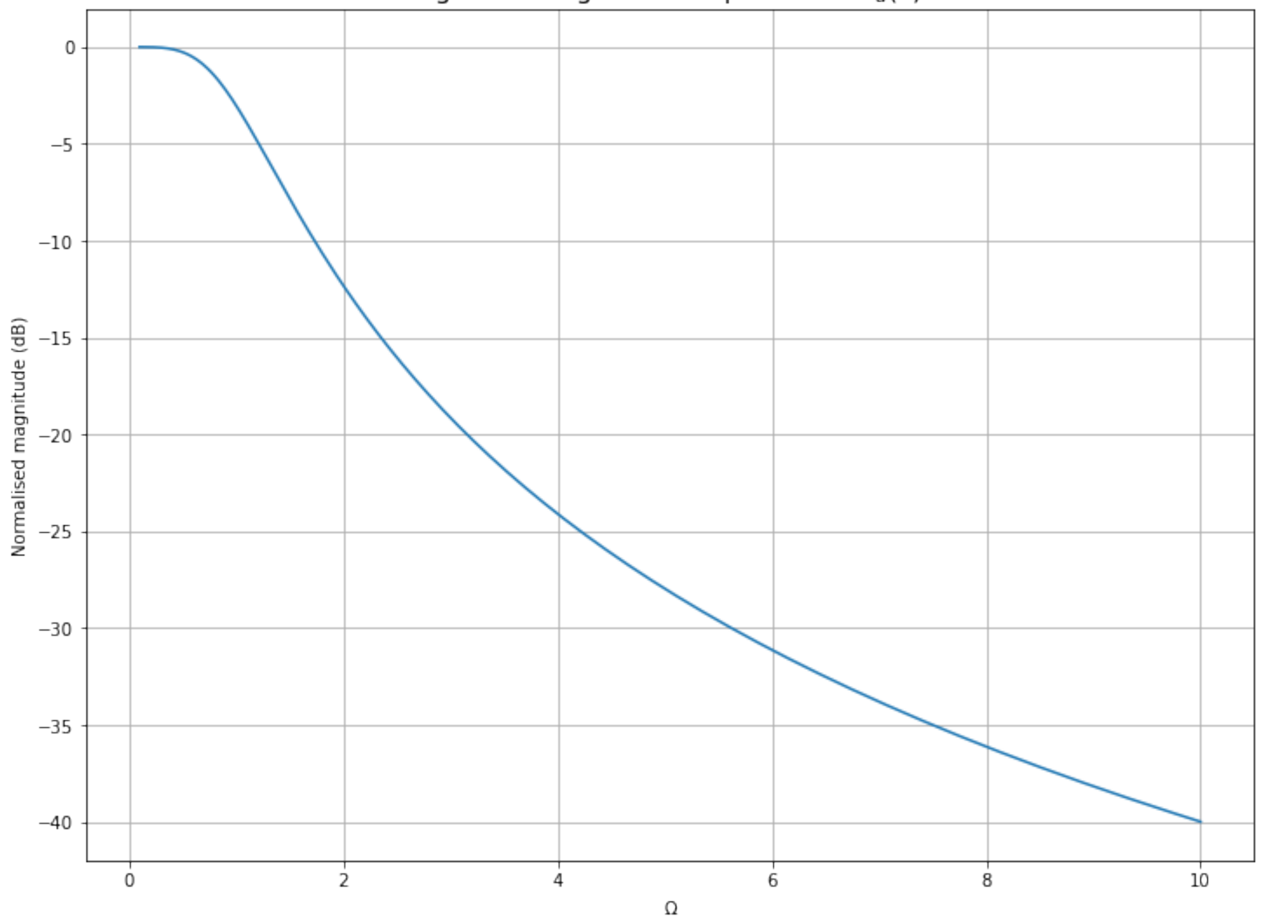
(b).

The plot of the magnitude response is shown in Figure 4. The python code is given below.

In [6]:

```python
b = [1]
a = [1, np.sqrt(2), 1]

omega1, H1 = signal.freqs(b, a)

plt.plot(omega1, 20*np.log10(np.abs(H1)))
plt.grid()
plt.xlabel('$\Omega$')
plt.ylabel('Normalised magnitude (dB)')
plt.title('Figure 4: Magnitude response for $H_a(s)$.', fontsize = 15)
plt.show()
```

Figure 4: Magnitude response for $H_a(s)$.



(c).

The poles of the filter are found by solving the following equation

$$s^2 + \sqrt{2}s + 1 = 0.$$

This gives

$$p_1 = \frac{1}{\sqrt{2}}(-1 + j)$$
$$p_2 = \frac{1}{\sqrt{2}}(-1 - j)$$

Note that $p_2 = p_1^*$.

(d).

We will now use the impulse invariance method to convert the analog filter to a digital IIR filter.

We can write $H_a(s)$ as

$$H_a(s) = \frac{1}{(s-p_1)(s-p_2)}.$$

By using the partial fraction expansion this can be written as

$$H_a(s) = \frac{c_1}{s-p_1} + \frac{c_2}{s-p_2}$$

where

$$c_1 = H_a(s)(s-p_1)\big|_{s=p_1} = \frac{1}{p_1-p_2} = \frac{\sqrt{2}}{2j} = -\frac{\sqrt{2}}{2}j.$$

Since $p_2 = p_1^*$, we have that

$$c_2 = c_1^* = -c_1 = -\frac{\sqrt{2}}{2}j$$

We can then find $H(z)$ as

$$\begin{aligned} H(z) &= \sum_{k=1}^{2} \frac{c_k}{1-e^{p_k T}z^{-1}} = \frac{c_1}{1-e^{p_1 T}z^{-1}} - \frac{c_1}{1-e^{p_1^* T}z^{-1}} \\ &= \frac{c_i(e^{p_1 T} - e^{p_1^* T})z^{-1}}{1-(e^{p_1 T}+e^{p_1^* T})z^{-1} + e^{(p_1+p_1^*)T}z^{-2}} \\ &= \frac{\sqrt{2}e^{-\frac{T}{\sqrt{2}}}\sin(\frac{T}{\sqrt{2}})z^{-1}}{1-2e^{-\frac{T}{\sqrt{2}}}\cos(\frac{T}{\sqrt{2}})z^{-1} + e^{-\sqrt{2}T}z^{-2}} \end{aligned}$$

The constant T is found from the frequency mapping $\omega = \Omega T$, and the requirement that the cut-off frequency of the analogue filter Ω_c should map to the required cut-off frequencies of the digital filters, ω_{c1} and ω_{c2}.

This gives

$$T_1 = \frac{\omega_c 1}{\Omega_c} = 0.25$$

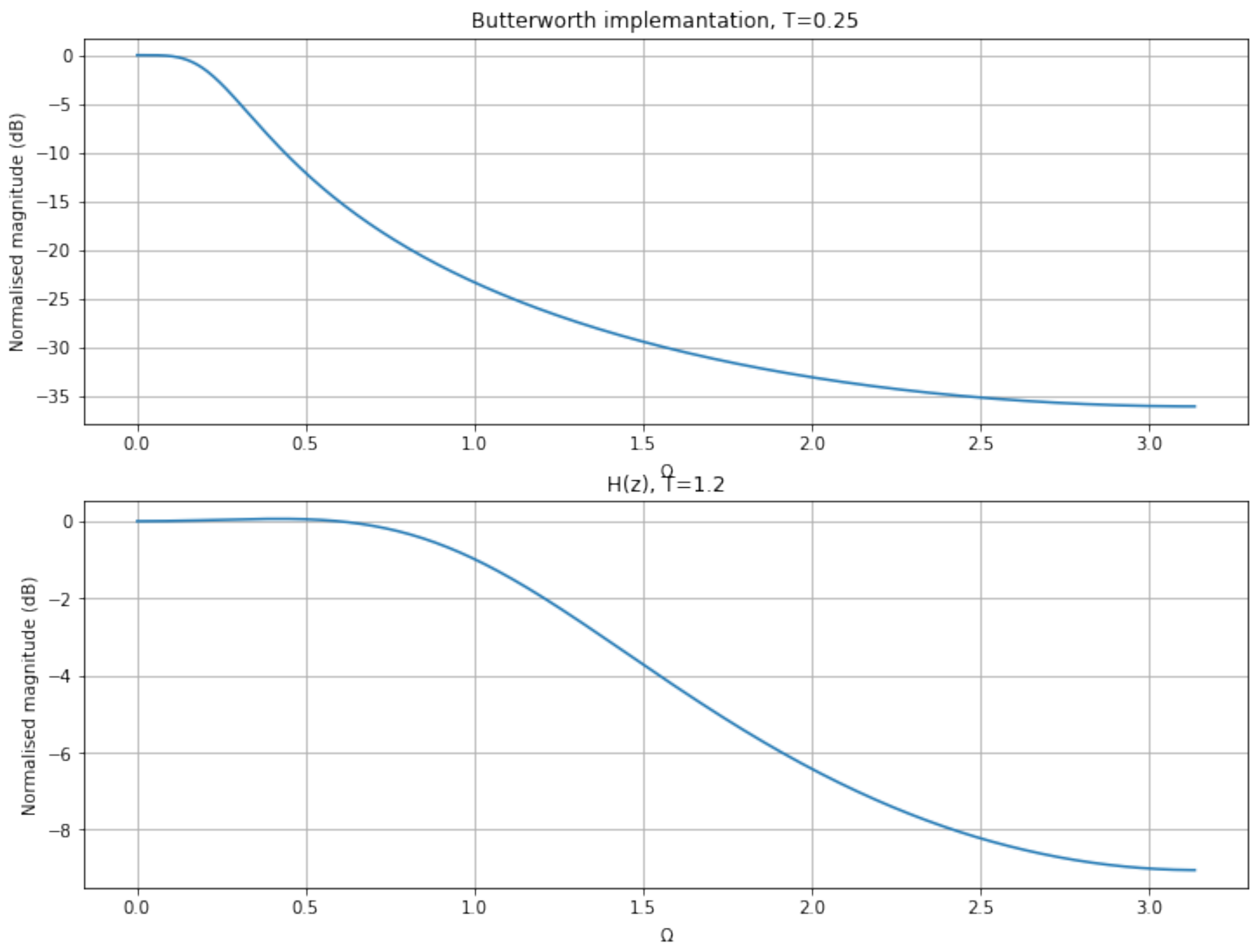
$$T_2 = \frac{\omega_{c2}}{\Omega_c} = 1.2$$

(e).

Magnitude responses of the digital IIR filters are shown in Figure 5 and 6 together with the corresponding part of the of the magnitude response of the prototype filter. (Notice that $\omega = \pi$ in figure 5 corresponds to $\omega = \frac{\pi}{T_1} = 15.7$ and $\omega = \pi$ in figure 6 corresponds to $\omega = \frac{\pi}{T_2} = 2.6$.) The python code is given below.

In [7]:
```
fig, axs = plt.subplots(2)
fig.suptitle('Figure 5: Magnitude responses of the prototype filter and the

T = 0.25
b = [0, np.sqrt(2)*np.exp(-T/np.sqrt(2))*np.sin(T/np.sqrt(2))]
a = [1, -2*np.exp(-T/np.sqrt(2))*np.cos(T/np.sqrt(2)), np.exp(-T*np.sqrt(2
omega1, H1 = signal.freqz(b, a)
axs[0].plot(omega1, 20*np.log10(np.abs(H1)/np.abs(np.amax(H1))))
axs[0].grid()
axs[0].set_title('Butterworth implemantation, T=' + str(T))
axs[0].set(xlabel = '$\Omega$', ylabel = 'Normalised magnitude (dB)')

T = 1.2
b = [0, np.sqrt(2)*np.exp(-T/np.sqrt(2))*np.sin(T/np.sqrt(2))]
a = [1, -2*np.exp(-T/np.sqrt(2))*np.cos(T/np.sqrt(2)), np.exp(-T*np.sqrt(2
omega1, H1 = signal.freqz(b, a)
axs[1].plot(omega1, 20*np.log10(np.abs(H1)/np.abs(np.amax(H1))))
axs[1].grid()
axs[1].set_title('H(z), T=' + str(T))
axs[1].set(xlabel = '$\Omega$', ylabel = 'Normalised magnitude (dB)')

plt.show()
```

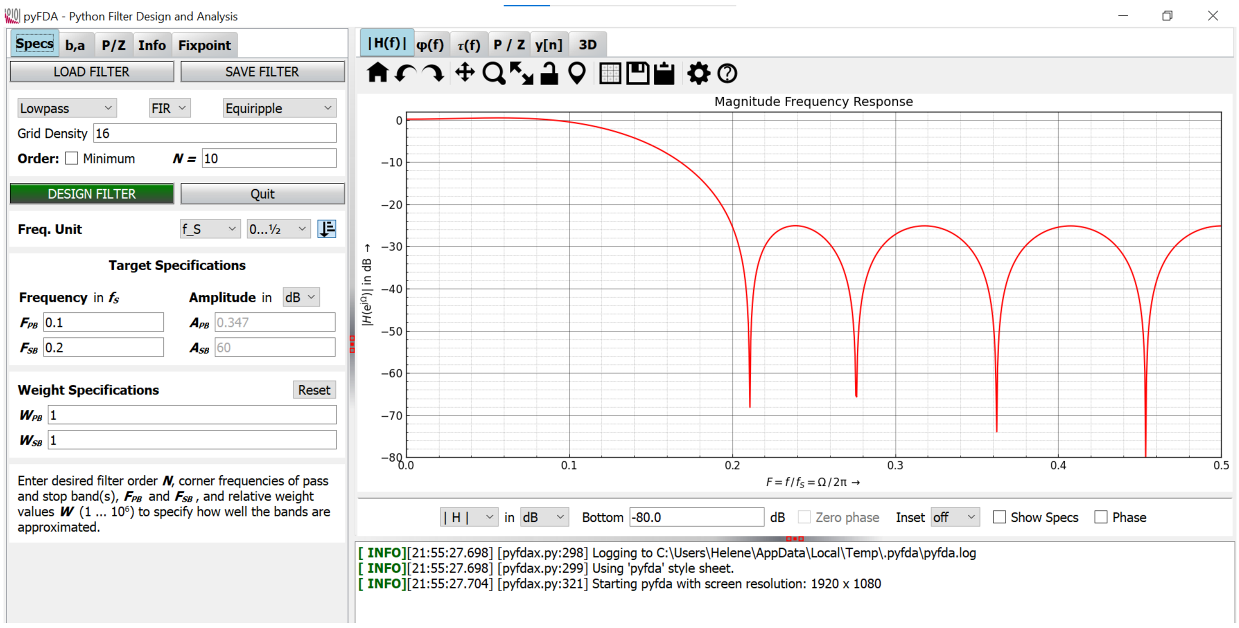
Figure 5: Magnitude responses of the prototype filter and the designed filter.



We can observe that the filter in figure 5 has a magnitude response that is almost the same as the analog filter. For the other filter with $T = 1.2$ we can see that the attenuation its much lower then the prototype filter. This is due to aliasing in the digital filter. We can also observe that in the filter with $T = 0.2$ the cut-off frequency is correct at $\omega_c = 0.2$, but for the filter with $T = 1.2$ aliasing has caused the cut-off frequency to shift to approximated $\omega_c = 1.35$.

Probelm 4

The pyFDA module can be downloaded from https://github.com/chipmuenk/pyfda. Here you can also find description on how to run the program. Below is a picture of what the program look like.



Lowpass | FIR | Equiripple

Grid Density 16

Order: ☐ Minimum N = 10

Freq. Unit f_S 0...½

Frequency in f_S: F_{PB} 0.1, F_{SB} 0.2

Amplitude in dB: A_{PB} 0.347, A_{SB} 60

W_{PB} 1

W_{SB} 1

Enter desired filter order N, corner frequencies of pass and stop band(s), F_{PB} and F_{SB}, and relative weight values W (1 ... 10^6) to specify how well the bands are approximated.

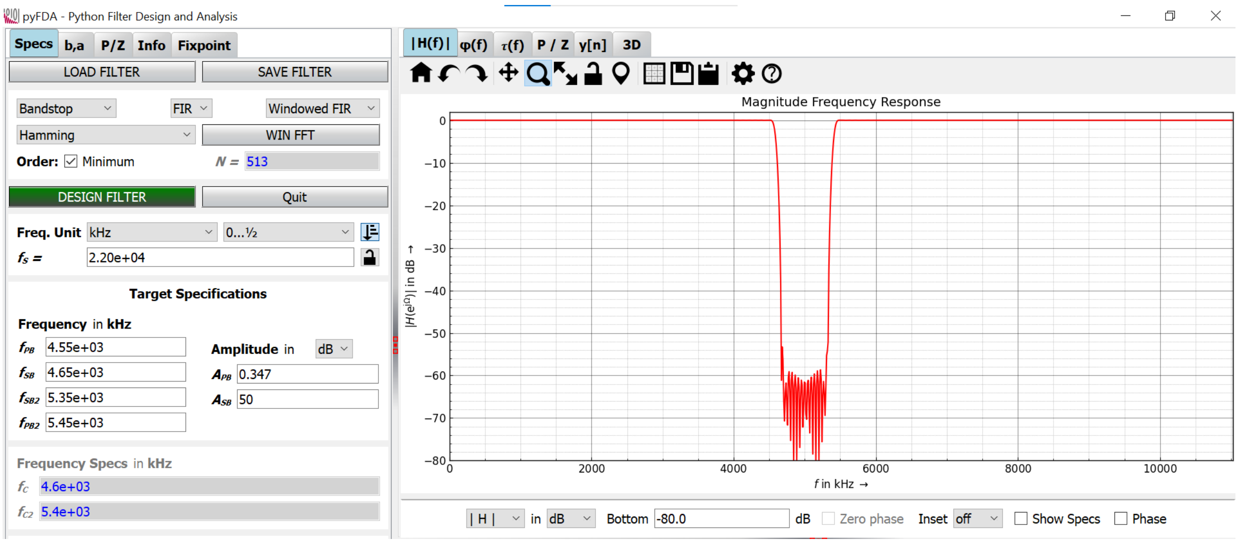
| H | in dB Bottom -80.0 dB Inset off

```
[ INFO][21:55:27.698] [pyfdax.py:298] Logging to C:\Users\Helene\AppData\Local\Temp\.pyfda\pyfda.log
[ INFO][21:55:27.698] [pyfdax.py:299] Using 'pyfda' style sheet.
[ INFO][21:55:27.704] [pyfdax.py:321] Starting pyfda with screen resolution: 1920 x 1080
```

When you have designed a filter you are happy with, you can export the data, and use it in your code.

(a).

Inspecting the magnitude response for the Hamming window method we can see that the specifications are met at an filter order of approximated $N = 513$. Before this we are not able to get a -50dB at 4650Hz. The magnitude response is shown in the figure below. The phase response is shown in the othr figure. From this figure we can see that the phase is linear.



pyFDA - Python Filter Design and Analysis

Specs | b,a | P/Z | Info | Fixpoint

LOAD FILTER | SAVE FILTER

Bandstop | FIR | Windowed FIR

Hamming | WIN FFT

Order: ☑ Minimum N = 513

DESIGN FILTER | Quit

Freq. Unit: kHz | 0...½

f_S = 2.20e+04

Target Specifications

Frequency in kHz

f_{PB} 4.55e+03

f_{SB} 4.65e+03

f_{SB2} 5.35e+03

f_{PB2} 5.45e+03

Amplitude in dB

A_{PB} 0.347

A_{SB} 50

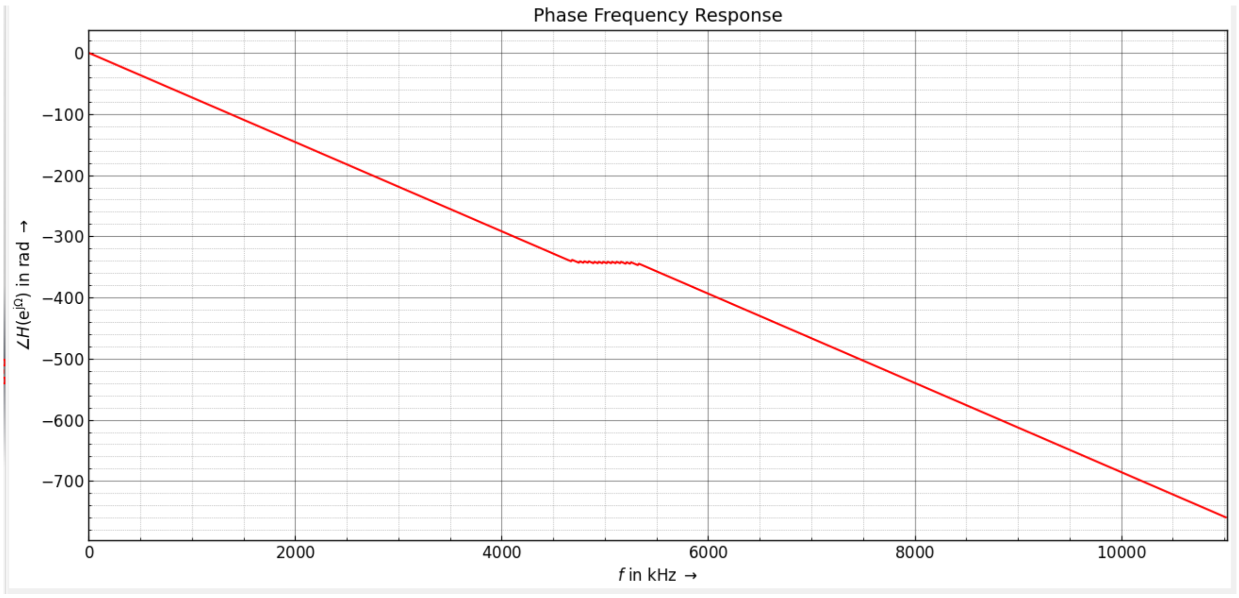
Frequency Specs in kHz

f_C 4.6e+03

f_{C2} 5.4e+03

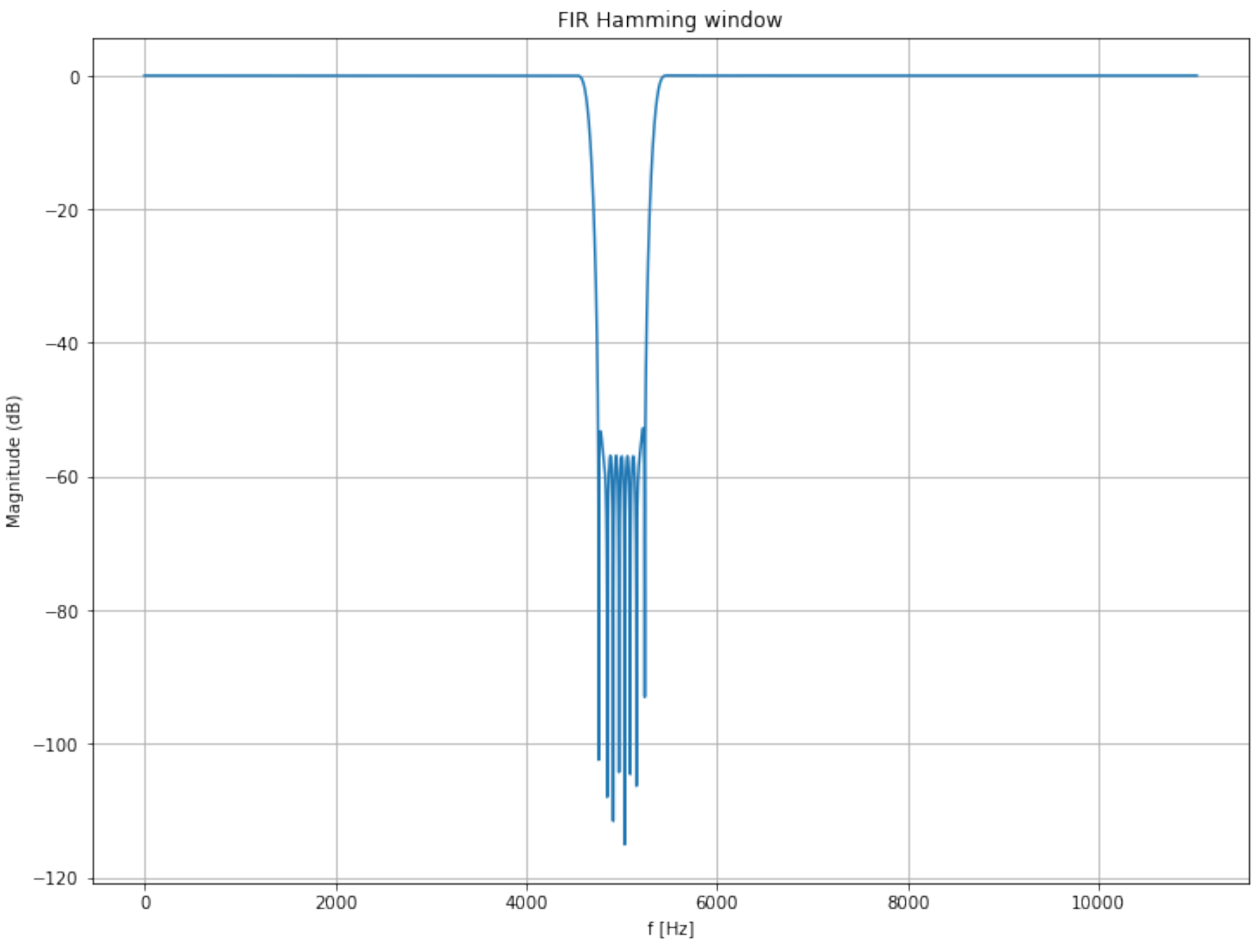
|H(f)| | φ(f) | τ(f) | P / Z | y[n] | 3D

| H | in dB Bottom -80.0 dB ☐ Zero phase Inset off ☐ Show Specs ☐ Phase



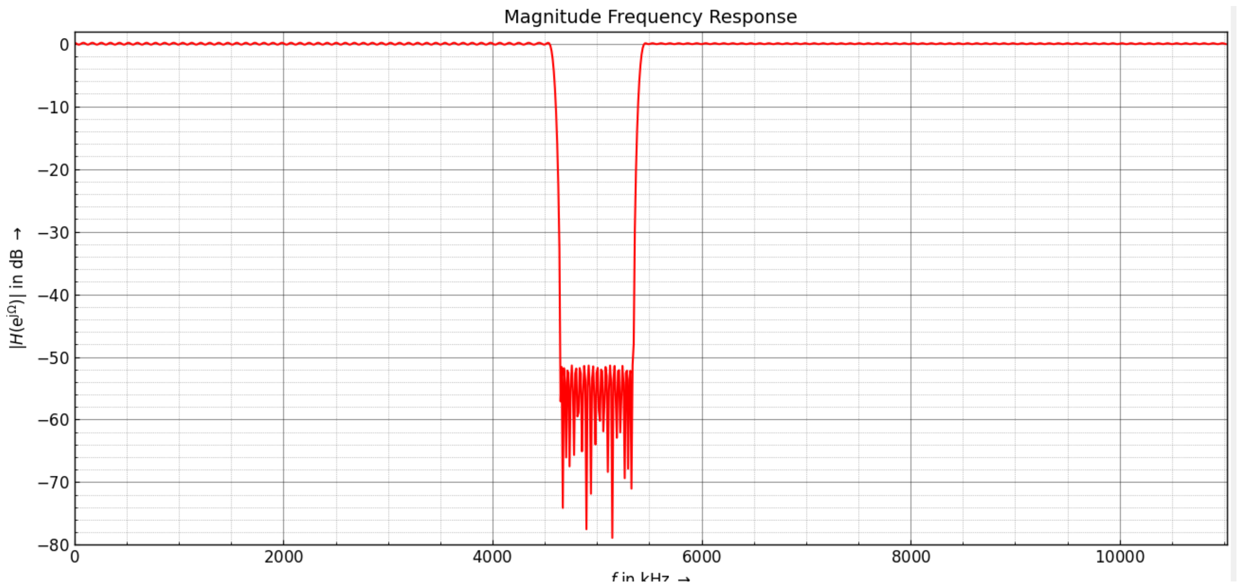
In the code below there is a example on how to plot exported data in python. For some reason, the format varies, thus the for-loop.

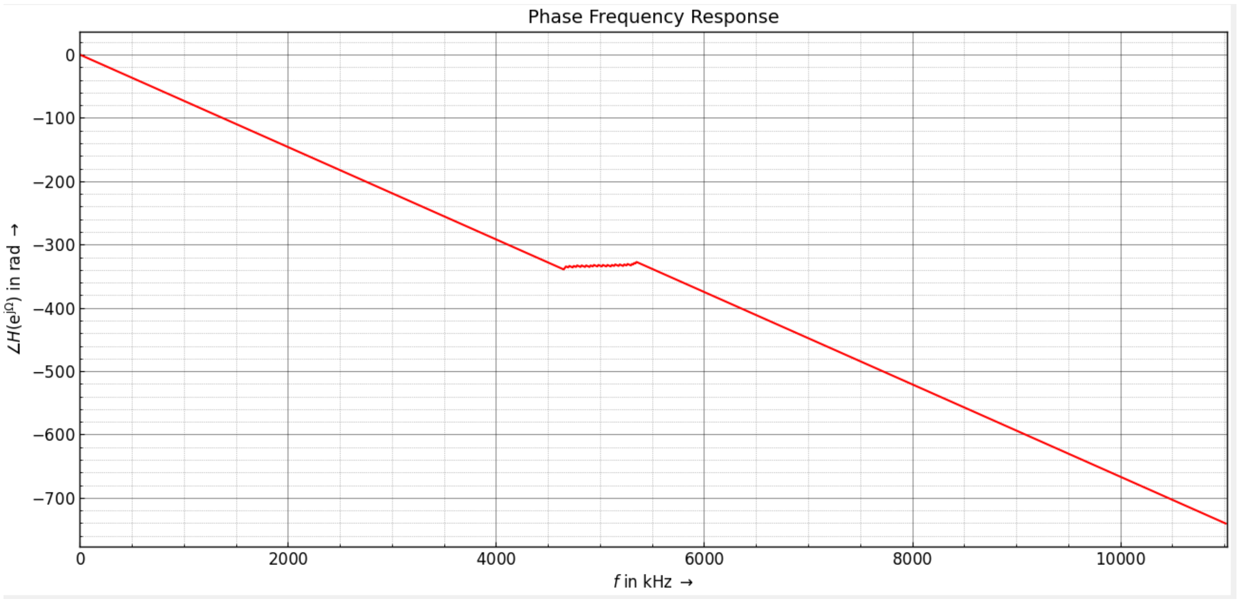
In [8]:
```
fs = 22050
df = pd.read_csv('4a.csv')

b = []
for value in df:
    try:
        b.append(float(value))
    except:
        b.append(float(value[:-2]))

omega1, H1 = signal.freqz(b, [1], worN = 4096*16)

plt.plot(omega1*fs/(2*np.pi), 20*np.log10(np.abs(H1)))
plt.grid()
plt.xlabel('f [Hz]')
plt.ylabel('Magnitude (dB)')
plt.title('FIR Hamming window')
plt.show()
```



(b).

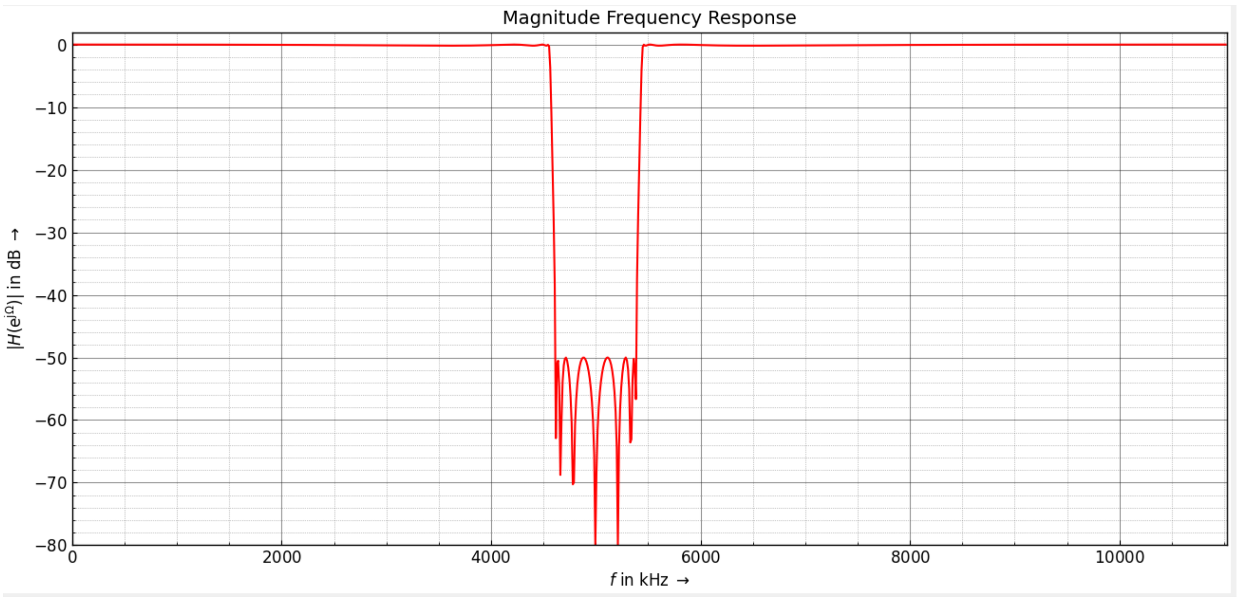
We use the pyFDA and find that the minimum filter order that satisfies the specifications is N = 512. The magnitude response and the phase response of this filter is shown in the figures below. From this we can see that the phase is linear.

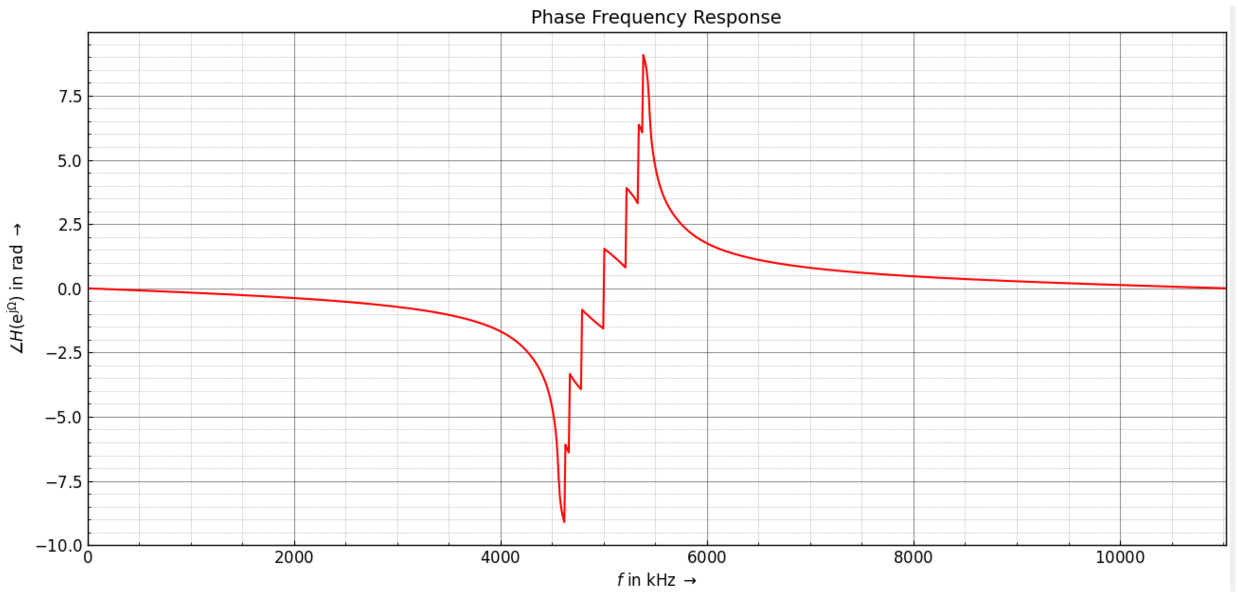




(c).

We use the pyFDA and find that the minimum filter order that satisfies the specifications is N = 7. The magnitude response and the phase response of this filter is shown in the figures below.





(d).

The windowing method has the advantage that it is very simple. However, it is also the method that results in the filter of highest order. The optimal equiripple design method is more complicated, but it produces a linear phase FIR filter with lowest possible order that the given specifications. This is achieved by controlling the size of the errors in the passband and stopband at the same time. Both windowing method and the equiripple method produce filters with linear phase response.

The elliptic IIR filter has significantly lower filter order than both the above methods. Thus, compared to the FIR filters, this filter is less computationally demanding to implement, requires less memory, and allows lower delay. The disadvantage of this filter is that we cannot obtain linear phase response.

(e).

In [9]:
```
fs, pianowav = read('pianoise.wav')

sd.default.samplerate = fs
sd.play(pianowav)
print('Playing original sound')
time.sleep(len(pianowav)/fs)

filtered_pianowav = signal.lfilter(b, [1], pianowav)
filtered_pianowav /= np.amax(filtered_pianowav)/2
sd.play(filtered_pianowav.real)
print('Playing filtered sound')
```

```
Playing original sound
Playing filtered sound
```

We can hear that we remove the noise when we use a high filter length(a high stopband attenuation).

In [ ]:
```
```