HW4_4&5

February 19, 2024

Cogs 118C HW4

Student name: Edwin Ruiz

Student PID: A17136339

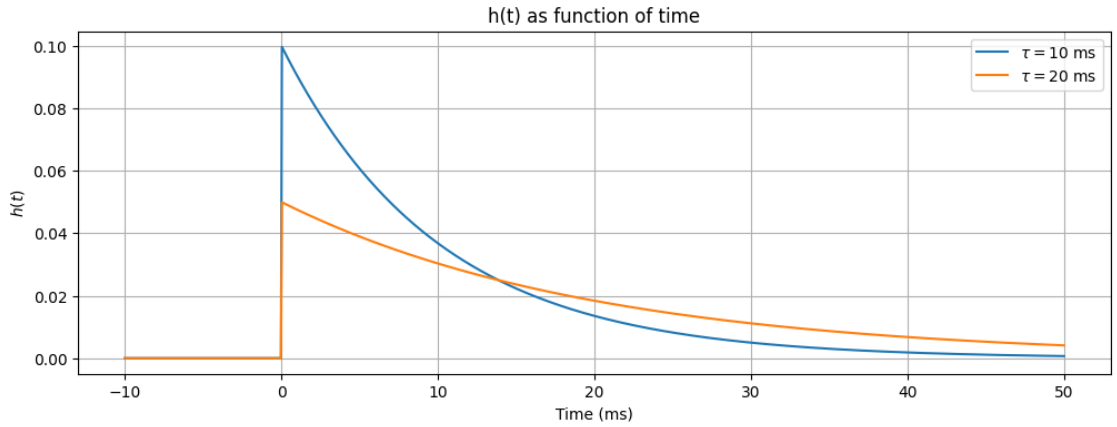
1 4

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

1.1 a.

Answer

[17]:
```
def h(t, tau):
    return (1/tau) * (t >= 0) * np.exp(-t / tau)


t = np.linspace(-10, 50, 1000)
tau_values = [10, 20]

plt.figure(figsize = (12, 4))

for tau in tau_values:
    plt.plot(t, h(t, tau), label = f'$\\tau = {tau}$ ms')

plt.title('h(t) as function of time')
plt.xlabel('Time (ms)')
plt.ylabel('$h(t)$')
plt.legend()
plt.grid(True)
plt.show()
```



1.2 b

Answer > increases = slower impulse response

decreases = faster impulse response

1.3 c

Answer > $H(\omega) = \int_{-\infty}^{\infty} h(t)e^{-i\omega t}\,dt$

$$= \int_{0}^{\infty} \frac{1}{\tau} e^{-\frac{t}{\tau}} e^{-i\omega t}\,dt = \frac{1}{1+iw\tau}$$

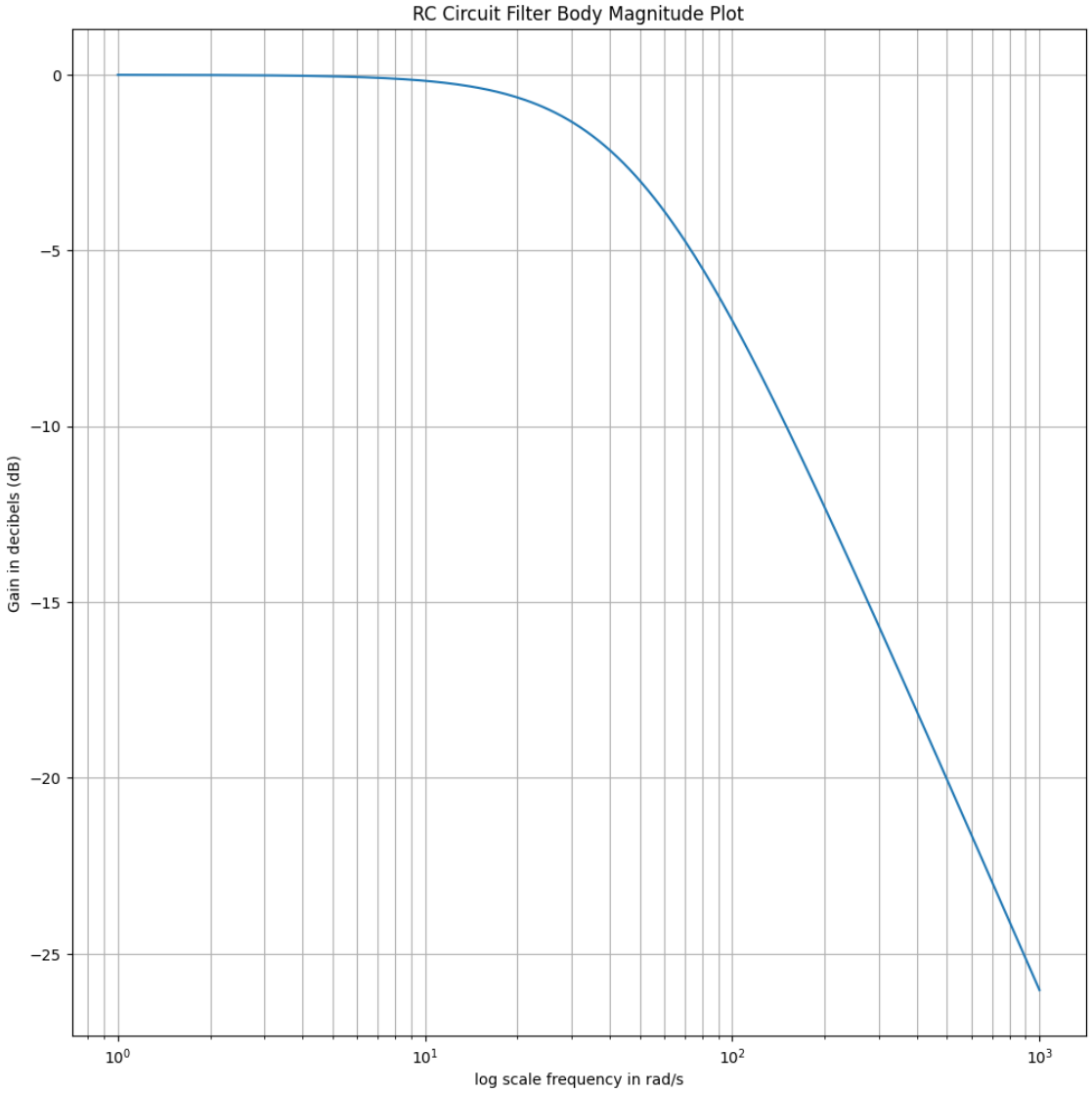
1.4 d

Answer > $g(w) = |\frac{1}{1+iw\tau}| = \frac{1}{\sqrt{1+(w\tau)^2}} = \frac{1}{\sqrt{1+(0\cdot\tau)^2}} = \frac{1}{\sqrt{1+0)^2}} = \frac{1}{1} = 1$

1.5 e

Answer

[21]:
```
tau_20ms = 20e-3
omega_range = np.logspace(0, 3, 300)
omega_g_function_gain = 1 / np.sqrt(1 + (omega_range * tau_20ms)**2)

omega_g_function_db = 20 * np.log10(omega_g_function_gain)

plt.figure(figsize = (12, 12))
plt.semilogx(omega_range, omega_g_function_db)
plt.title('RC Circuit Filter Body Magnitude Plot')
plt.xlabel('log scale frequency in rad/s')
plt.ylabel('Gain in decibels (dB)')
plt.grid(True, which = "both", ls = "-")
plt.show()
```



1.6 f

Answer $> 20\ log_{10}\ \frac{1}{\sqrt{1+(2\pi\cdot 60\cdot 0.02)^2}} = -17.622952655178295\ db$

- $20\ ms\ \cdot\ \frac{1}{1000} = 0.02\ s$
- Angular function: $w = \frac{2\pi}{T} \rightarrow f_{frequency} = \frac{1}{T} \rightarrow w = \Theta = 2\pi f$
- – T = period

1.7 g

$Im[H(\omega)] = -\frac{1}{1+(w\tau)^2}$

$Re[H(\omega)] = \frac{1}{1+(w\tau)^2}$

$tan\phi(\omega) = \frac{Im[H(\omega)]}{Re[H(\omega)]} = \frac{-\frac{1}{1+(w\tau)^2}}{\frac{1}{1+(w\tau)^2}} = -1 \rightarrow -w\tau = (2\pi \cdot 60)(0.02) = 7.53982236$

Answer > 7.53982236

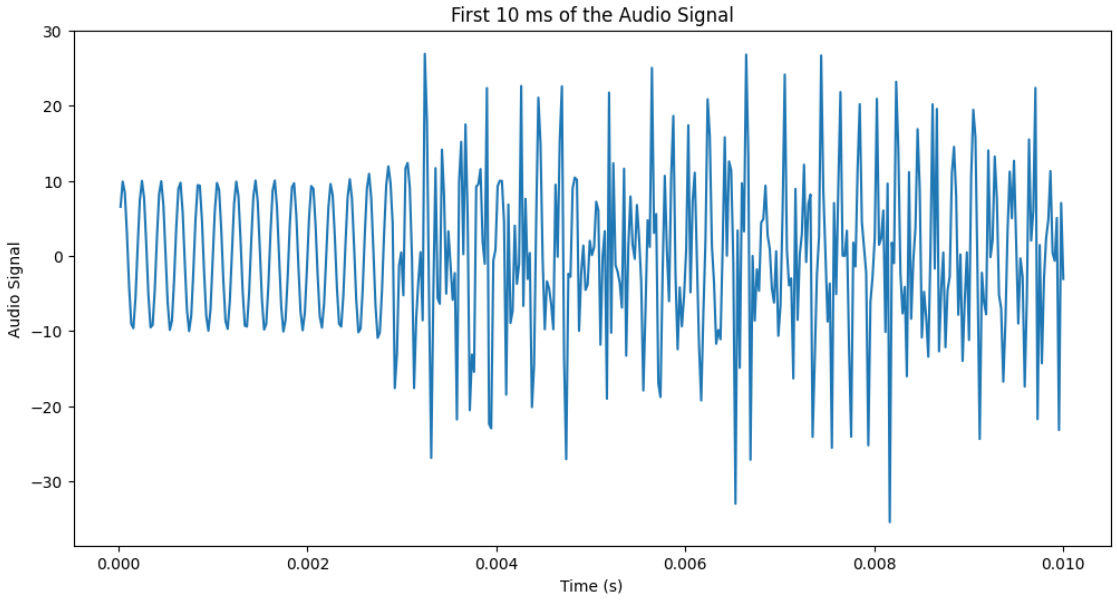
---

2 5

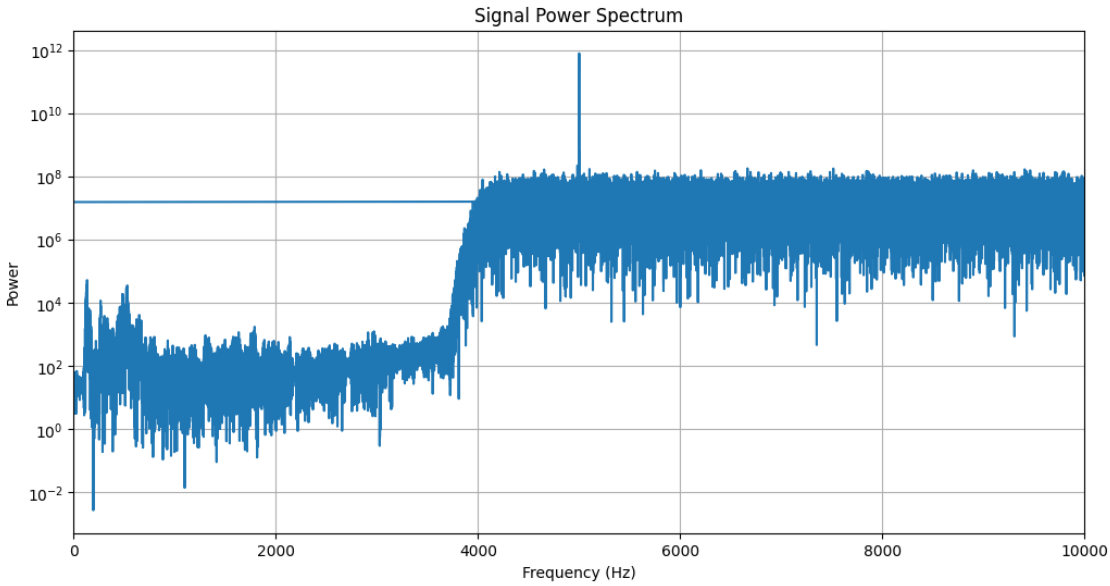
[2]:
```
# Load the audio file
data = np.loadtxt('hw4_audio.csv',delimiter=',')
t = data[:,0]
x = data[:,1]
Fs = 44100 # Sampling rate in Hz
```

[3]:
```
# (a) Save a .wav audio file. You should be able to download
# this file and play it to your speakers. How does it sound?
from scipy.io import wavfile
wavfile.write('hw4_audio.wav',Fs,x)
```

[4]:
```
# (b) Make a plot showing the first 10 ms of the signal.
# Do you notice a strong sinusoidal oscillation?
plt.figure(figsize = (12, 6))
first_10ms = 0.01
number_of_samples_in_10ms = Fs * first_10ms
time_range = int(number_of_samples_in_10ms)
plt.plot(t[0:time_range], x[0:time_range])
plt.title('First 10 ms of the Audio Signal')
plt.xlabel('Time (s)')
plt.ylabel('Audio Signal')
plt.show()
```

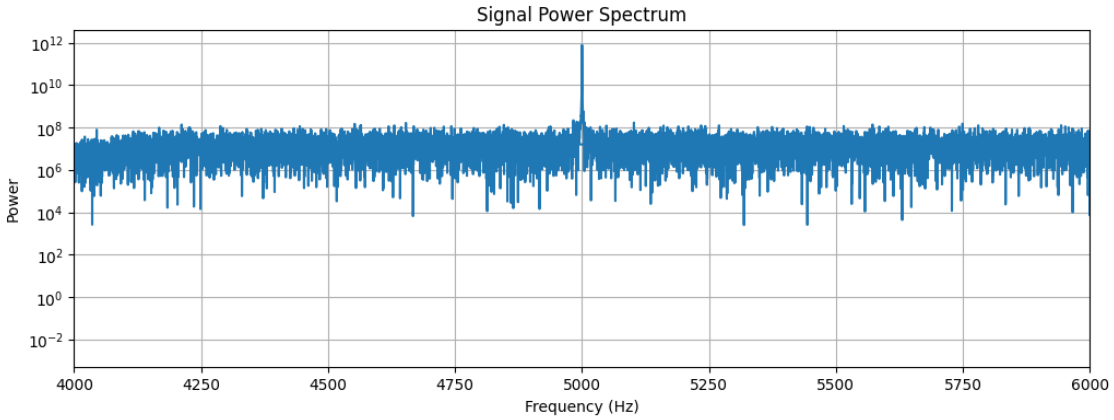


[5]:
```
# (c) Plot the power spectrum of the signal. Use
# log scale for the y-axis. Use plt.xlim() to zoom
# in on the frequency range from 0 to 10 kHz. Finally,
# make sure to label the axes.
#
# Hint: use np.fft.fft, np.fft.fftfreq

X = np.fft.fft(x)
f = np.fft.fftfreq(len(x), 1 / Fs)

plt.figure(figsize = (12, 6))
plt.plot(f, np.abs(X)**2)
plt.yscale('log')
plt.xlim(0, 10000)
plt.title('Signal Power Spectrum')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power')
plt.grid(True)
plt.show()
```

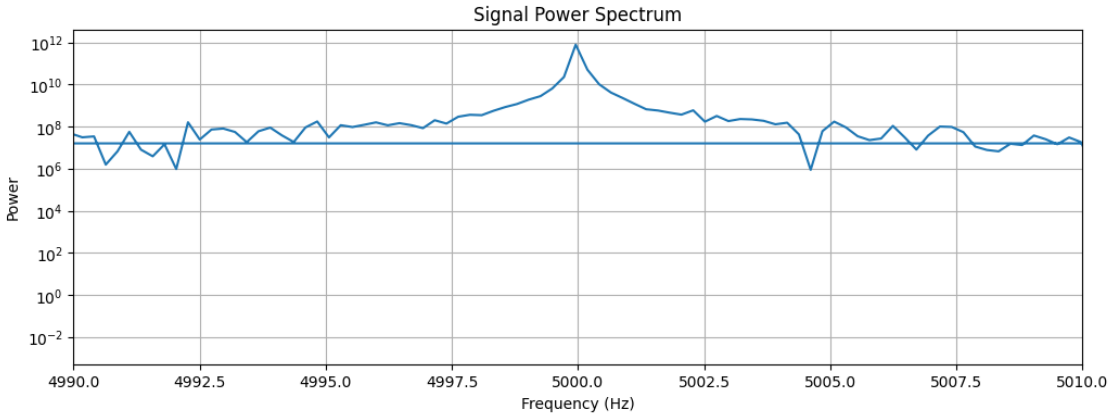


[6]:
```
plt.figure(figsize = (12, 4))
plt.plot(f, np.abs(X)**2)
plt.yscale('log')
plt.xlim(4000, 6000)
plt.title('Signal Power Spectrum')
```

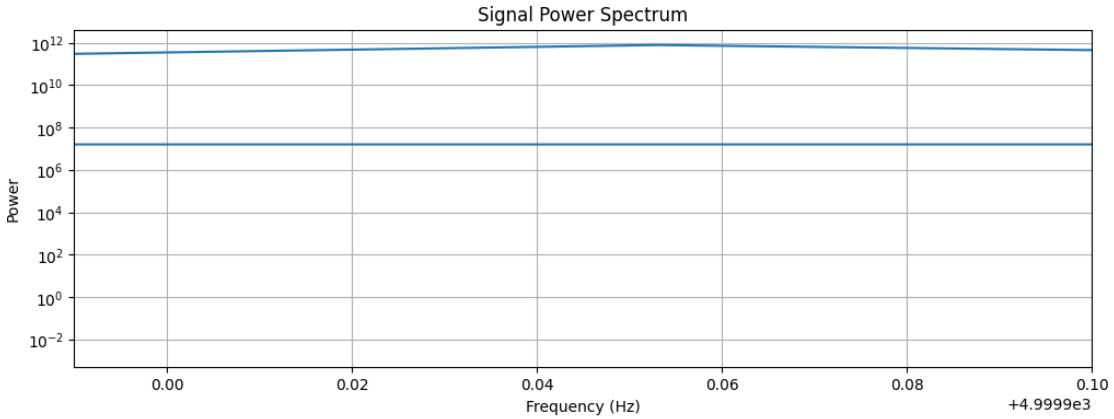
```
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power')
plt.grid(True)
plt.show()
```



[7]:
```
plt.figure(figsize = (12, 4))
plt.plot(f, np.abs(X)**2)
plt.yscale('log')
plt.xlim(4990, 5010)
plt.title('Signal Power Spectrum')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power')
plt.grid(True)
plt.show()
```



[8]:
```
plt.figure(figsize = (12, 4))
plt.plot(f, np.abs(X)**2)
plt.yscale('log')
plt.xlim(4999.89, 5000)
plt.title('Signal Power Spectrum')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power')
plt.grid(True)
plt.show()
```



3 (d) What is the exact frequency of the largest peak

4 in the power spectrum? Hint: You may need to zoom in even

5 further on the plot you made in (c) to see exactly where

6 the peak is.

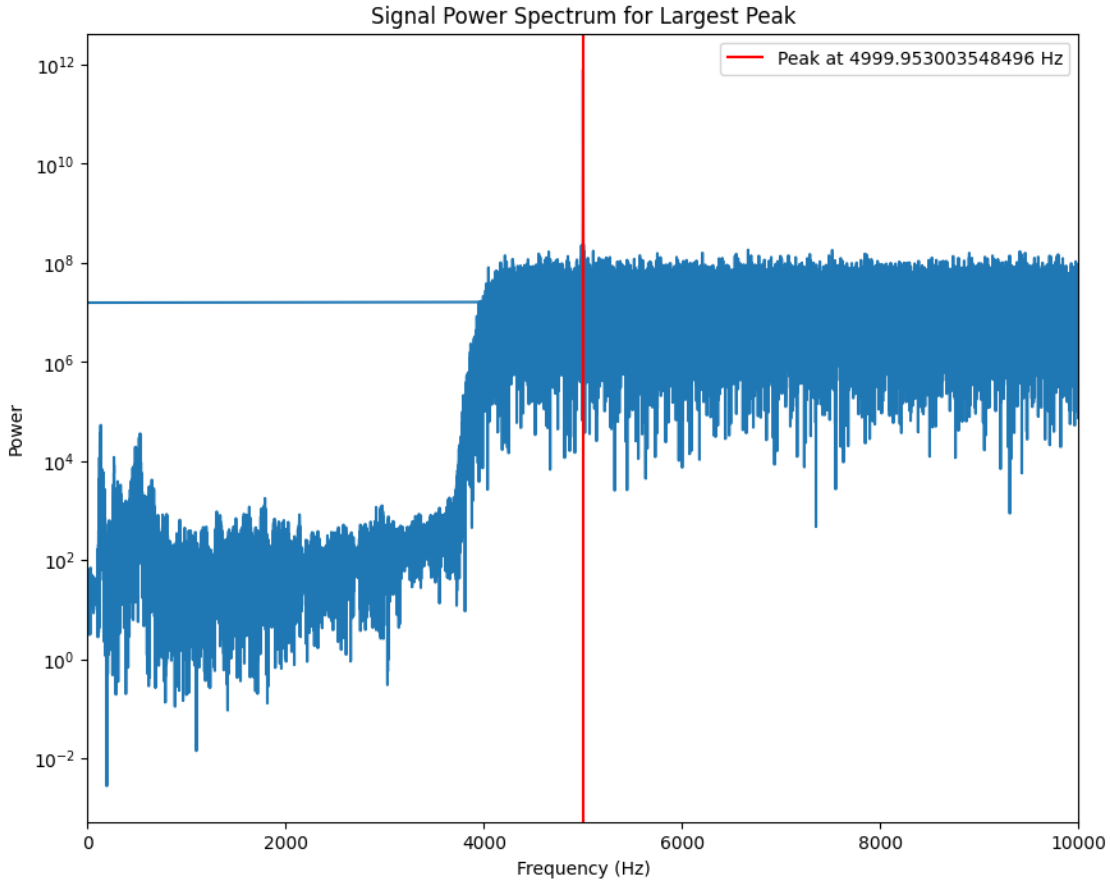
Answer > 4999.953003548496 Hz

[9]:
```
largest_peak_index = np.argmax(np.abs(X[:len(f)//2])**2)
largest_peak_frequency = f[largest_peak_index]
print("The exact frequency of the largest peak is: ", largest_peak_frequency,␣
↪"Hz")
```

```
The exact frequency of the largest peak is:  4999.953003548496 Hz
```

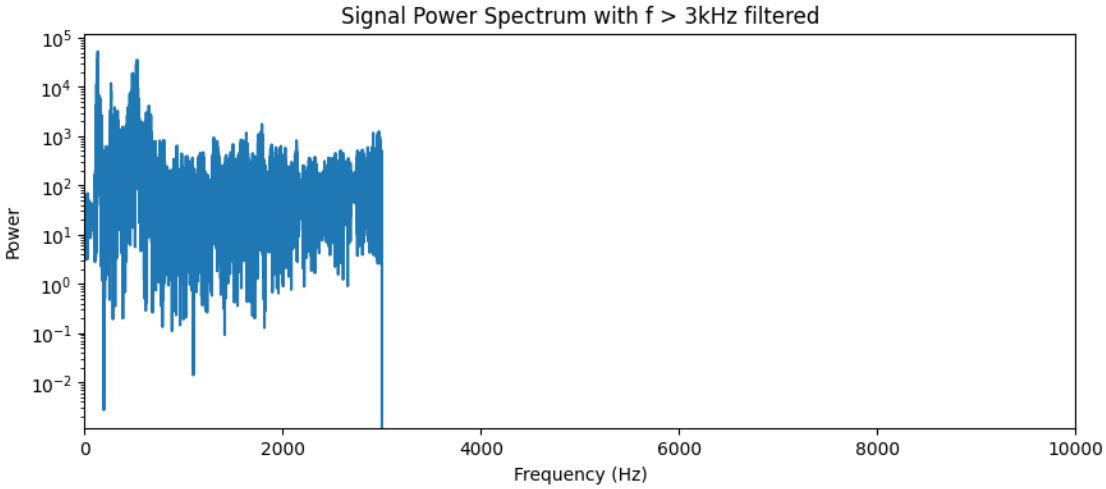
[10]:
```
plt.figure(figsize = (10, 8))
plt.plot(f, np.abs(X)**2)
plt.axvline(x = largest_peak_frequency, color = 'r', linestyle = '-', label =␣
↪f'Peak at {largest_peak_frequency} Hz')
plt.yscale('log')
```

```
plt.xlim(0, 10000)
plt.title('Signal Power Spectrum for Largest Peak')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power')
plt.legend()
plt.show()
```



[11]:
```
# (e) Ideal filter: Make a filter, H, that will remove frequencies
# f>3 kHz. Plot the product of H with X, the Fourier transform of x

# To get you started, here is how to create H:
H = np.abs(f) < 3000
filtering_X = X * H

plt.figure(figsize = (10, 4))
plt.plot(f, np.abs(filtering_X)**2)
plt.yscale('log')
plt.xlim(0, 10000)
plt.title('Signal Power Spectrum with f > 3kHz filtered')
```

```
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power')
plt.show()
```



Plot e shows a limited range of the original audio and filters frequencies above 3 kHz

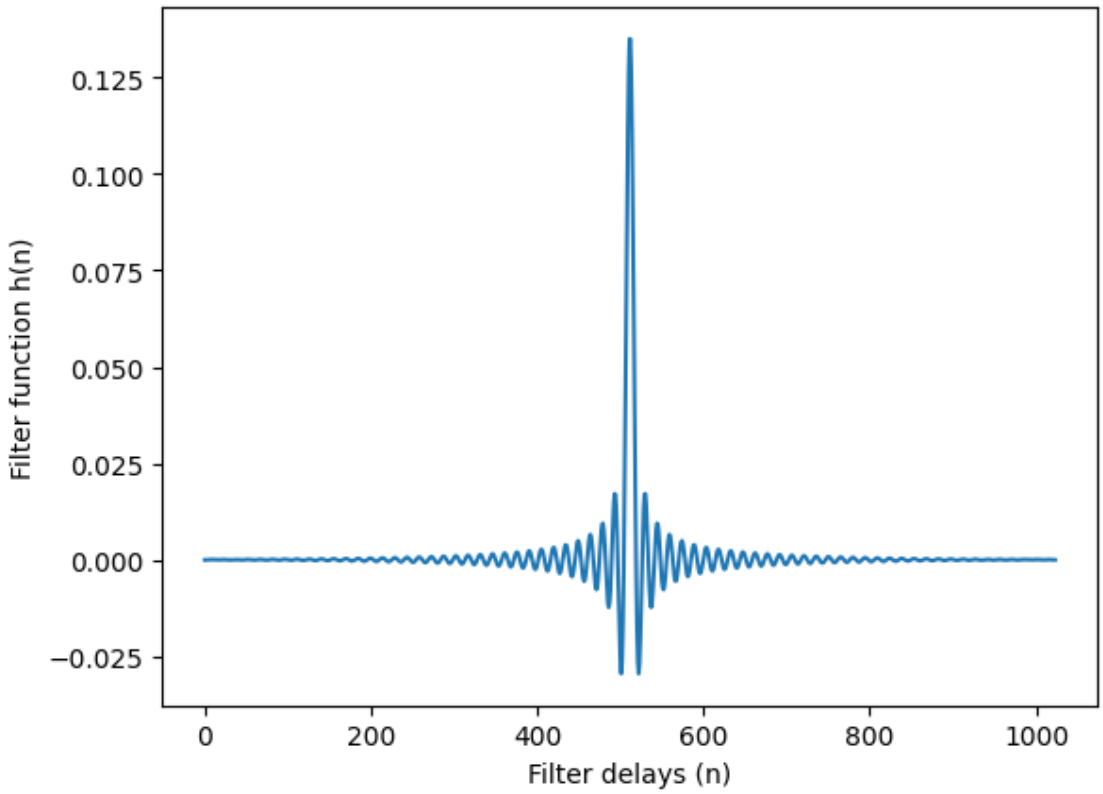
[12]:
```
# (f) Use the convolution theorem to calculate the filtered signal.
# To do this, note that the filtered signal is the inverse Fourier transform
# (ifft) of the product of the Fourier transforms of x and h.
# Save a .wav audio file containing the filtered signal. Download and listen to it.
# Describe what you hear.

# Hint: use wavfile.write

filtered_x = np.real(np.fft.ifft(filtering_X))
wavfile.write('hw4_audio_filtered.wav', Fs, filtered_x)
```

[13]:
```
# (g) Apply the high-pass filter we designed to remove the high frequencies
# from the data. You may use np.convolve, with mode='same' so that the output
# has the same size as the input. Plot the power spectrum of the filtered signal,
# again showing the range 0 to 10 kHz. Comment on the similarity and difference
# of this spectrum from the original spectrum (part c).
```

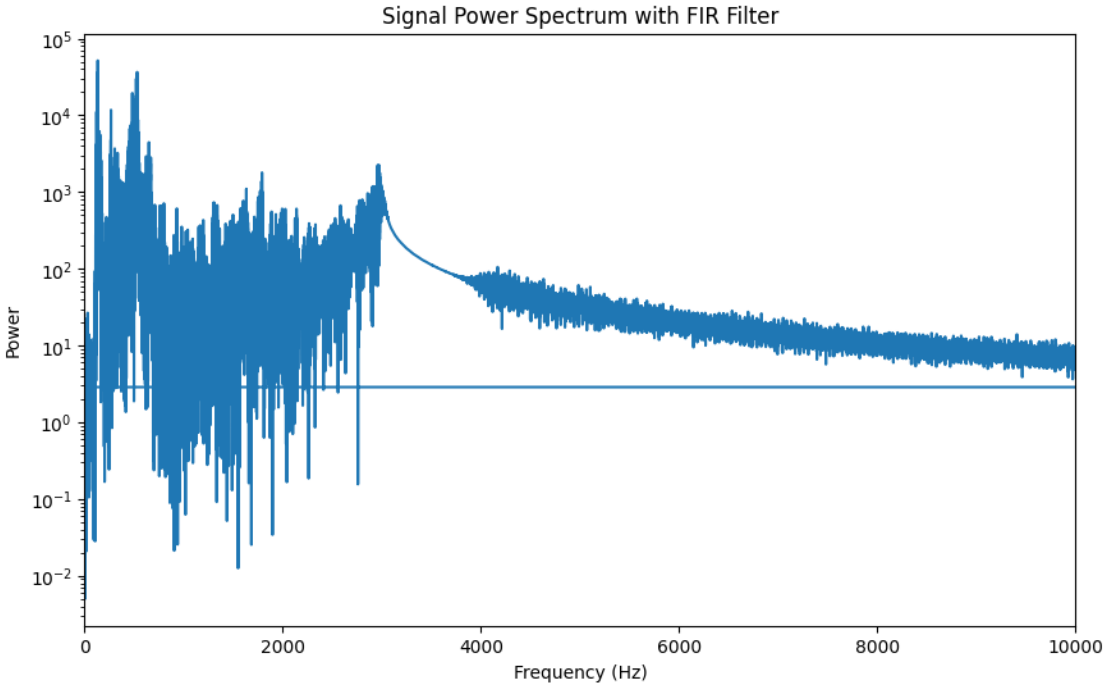
[14]:
```
# Design a better high-pass filter using a finite impulse response (FIR)
# convolution. We will create a filter with order n=1024
from scipy import signal
n = 1024
h = signal.firwin(n,cutoff=3000,fs=Fs) # This function designs a FIR filter
```

```
plt.plot(h)
plt.xlabel('Filter delays (n)')
plt.ylabel('Filter function h(n)')
```

[14]: Text(0, 0.5, 'Filter function h(n)')



[15]:
```
filtered_x_fir = np.convolve(x, h, mode = 'same')
filtered_X_fir = np.fft.fft(filtered_x_fir)

plt.figure(figsize=(10, 6))
plt.plot(f, np.abs(filtered_X_fir)**2)
plt.yscale('log')
plt.xlim(0, 10000)
plt.title('Signal Power Spectrum with FIR Filter')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power')
plt.show()
```



[16]:
```
# (h) Save the filtered signal as a .wav file and listen
# to it on your speakers. What is the "secret" message
# (actually a quote from the eminent UCSD neuro-philosopher
# Patricia Churchland)?

wavfile.write('hw4_audio_lp_filter.wav', Fs, filtered_x_fir))
```