In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sp
from scipy import signal

# x = np.array(np.ones(150))
# x = x.ravel()

# b,a = signal.butter(2, .1, 'low') # two pole butterworth filter
# y = signal.lfilter(b,a,x)
# plt.plot(y)
```

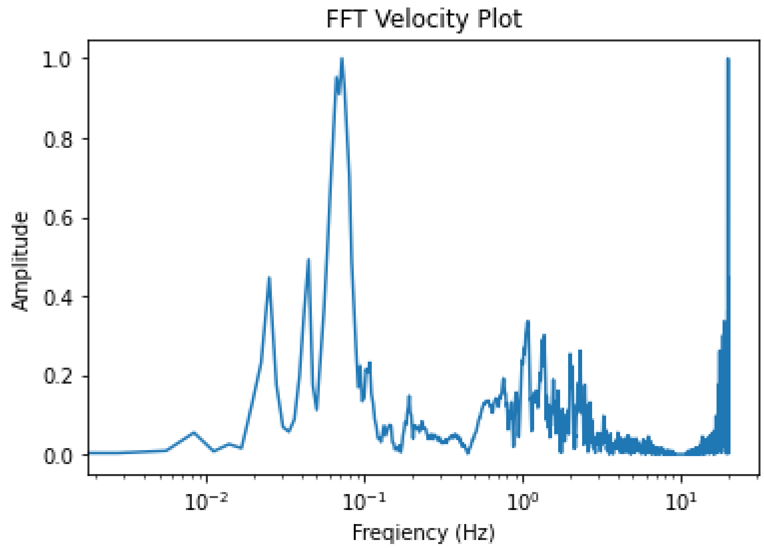
Problem 1: Load the Erebus earthquake velocity signal (load Erebus_velocity.mat) and plot its amplitude spectrum using abs and fft. Make the frequency axis logarithmic and the amplitude axis linear. Only show frequency information from 0 Hz through Nyquist frequency at 20 Hz. Normalize your spectrum to a peak value of 1.

In [ ]:
```
seis = sp.loadmat('./Erebus_velocity.mat')
print(seis.keys())
vel = seis['vel']
vel = vel.ravel()
sr = 40 #sample rate in Hz
freq = np.fft.fft(vel) # create the frequency spectrum


amp_spectrum = np.abs(freq)# amplitide spectrum
freq_ax = np.linspace(0, sr/2, len(freq))
norm = (amp_spectrum - np.min(amp_spectrum))/(np.max(amp_spectrum)-np.min(amp_s
# zi = (xi – min(x)) / (max(x) – min(x)) normalize equation
# fig = plt.figure()
plt.plot(freq_ax, norm)
plt.title('FFT Velocity Plot')
plt.xlabel('Freqiency (Hz)')
plt.xscale('log')
plt.yscale('linear')
plt.ylabel('Amplitude')
```

```
dict_keys(['__header__', '__version__', '__globals__', 'hdr', 'vel'])
```

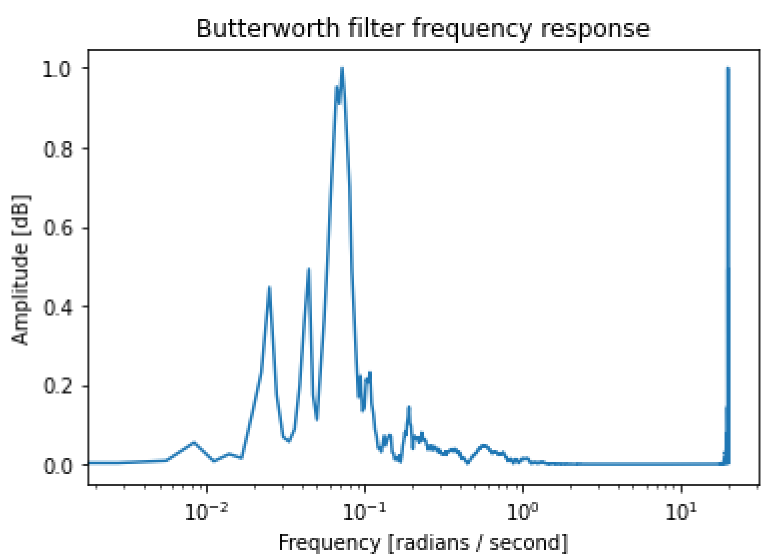
Out[ ]:
```
Text(0, 0.5, 'Amplitude')
```



Problem %% 2: Apply a two-pole low-pass Butterworth filter (butter) to the velocity seismogram using a corner frequency of 1 Hz. Plot the amplitude spectrum as in part 1 and normalize its amplitude to 1.

In [ ]:
```
b,a = signal.butter(2, 1/(40/2), 'low') #  two pole butterworth filter the Wn h
h = signal.filtfilt(b,a,vel)
# w, h = signal.freqs(b, a)
fft = np.fft.fft(h)
amp = np.abs(fft)
norm = (amp - np.min(amp))/(np.max(amp)-np.min(amp))
plt.plot(freq_ax, norm)
plt.xscale('log')
plt.title('Butterworth filter frequency response')
plt.xlabel('Frequency [radians / second]')
plt.ylabel('Amplitude [dB]')
```

Out[ ]:
```
Text(0, 0.5, 'Amplitude [dB]')
```

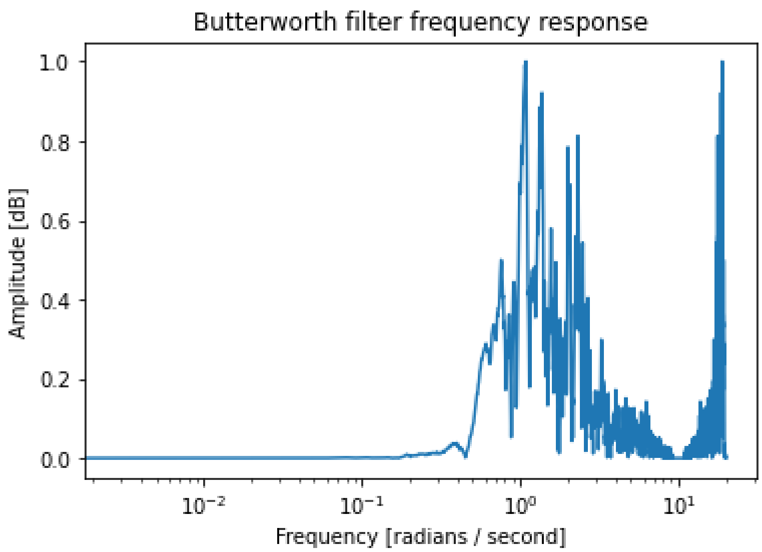


Problem 3: Same as problem two bit with high pass filter

In [ ]:
```
w,g = signal.butter(2, 1/(40/2), 'high') #  two pole butterworth filter the Wn
z = signal.filtfilt(w,g,vel)
```

```
# w, h = signal.freqs(b, a)
fft = np.fft.fft(z)
amp = np.abs(fft)
norm = (amp - np.min(amp))/(np.max(amp)-np.min(amp))
plt.plot(freq_ax, norm)
plt.xscale('log')
plt.title('Butterworth filter frequency response')
plt.xlabel('Frequency [radians / second]')
plt.ylabel('Amplitude [dB]')
```

Out[ ]:
```
Text(0, 0.5, 'Amplitude [dB]')
```



%% 4: Calculate the phase spectra for parts 1 and 2 (up to Nyquist) and use these to get the phase ‘difference’ between the velocity and low-pass filtered spectra of velocity. Plot the spectral phase difference between -pi and pi. Note that you might want to use the command unwrap.

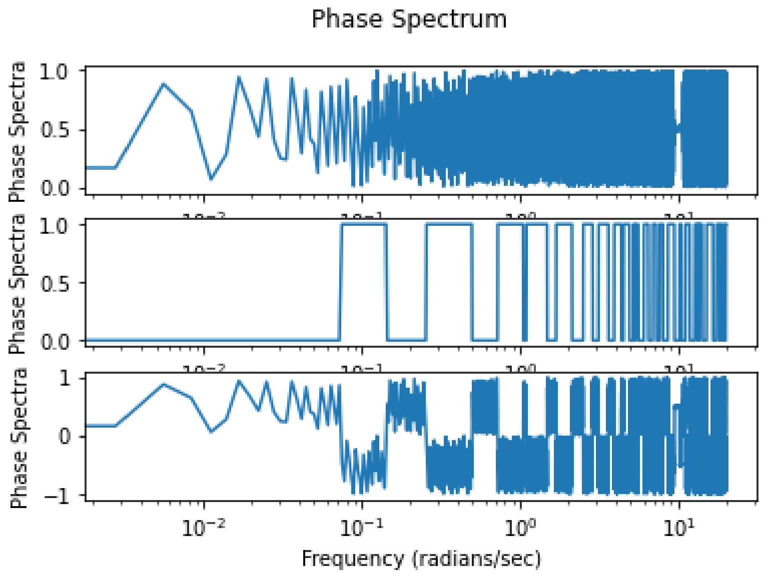
In [ ]:
```
phase_spectrum = np.angle(freq)# amplitide spectrum
norm = (phase_spectrum - np.min(phase_spectrum))/(np.max(phase_spectrum)-np.mir

phase_butter = np.angle(h) #get phase from butterworth output
norm_b = (phase_butter - np.min(phase_butter))/(np.max(phase_butter)-np.min(pha

# d_theta = (phase_spectrum-phase_butter) #difference in phase
d_theta = (norm-norm_b)
xx = np.unwrap(phase_spectrum)
yy =np.unwrap(phase_butter)


fig, ax = plt.subplots(3)
fig.suptitle('Phase Spectrum')
ax[0].plot(freq_ax, norm)
ax[1].plot(freq_ax, norm_b)
ax[2].plot(freq_ax, d_unwrap)

for y in ax.flat:
    y.set(xlabel='Frequency (radians/sec)', ylabel='Phase Spectra')
    y.set(xscale = 'log', yscale = 'linear')
```



Problem 5: same as 4 but witt high pass

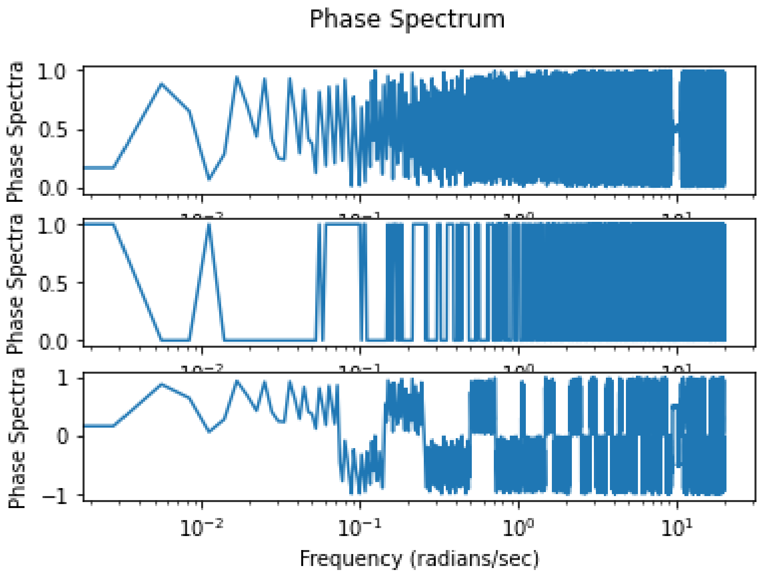
In [ ]:
```
phase_spectrum = np.angle(freq)# amplitide spectrum
norm = (phase_spectrum - np.min(phase_spectrum))/(np.max(phase_spectrum)-np.mir

phase_butter = np.angle(z) #get phase from butterworth output
norm_b = (phase_butter - np.min(phase_butter))/(np.max(phase_butter)-np.min(pha

# d_theta = (phase_spectrum-phase_butter) #difference in phase
d_theta = (norm-norm_b)
xx = np.unwrap(phase_spectrum)
yy =np.unwrap(phase_butter)


fig, ax = plt.subplots(3)
fig.suptitle('Phase Spectrum')
ax[0].plot(freq_ax, norm)
ax[1].plot(freq_ax, norm_b)
ax[2].plot(freq_ax, d_unwrap)

for y in ax.flat:
    y.set(xlabel='Frequency (radians/sec)', ylabel='Phase Spectra')
    y.set(xscale = 'log', yscale = 'linear')
```


Phase Spectrum
Phase Spectra
1.0
0.5
0.0
Phase Spectra
1.0
0.5
0.0
Phase Spectra
1
0
−1
10^{-2}
10^{-1}
10^{0}
10^{1}
Frequency (radians/sec)