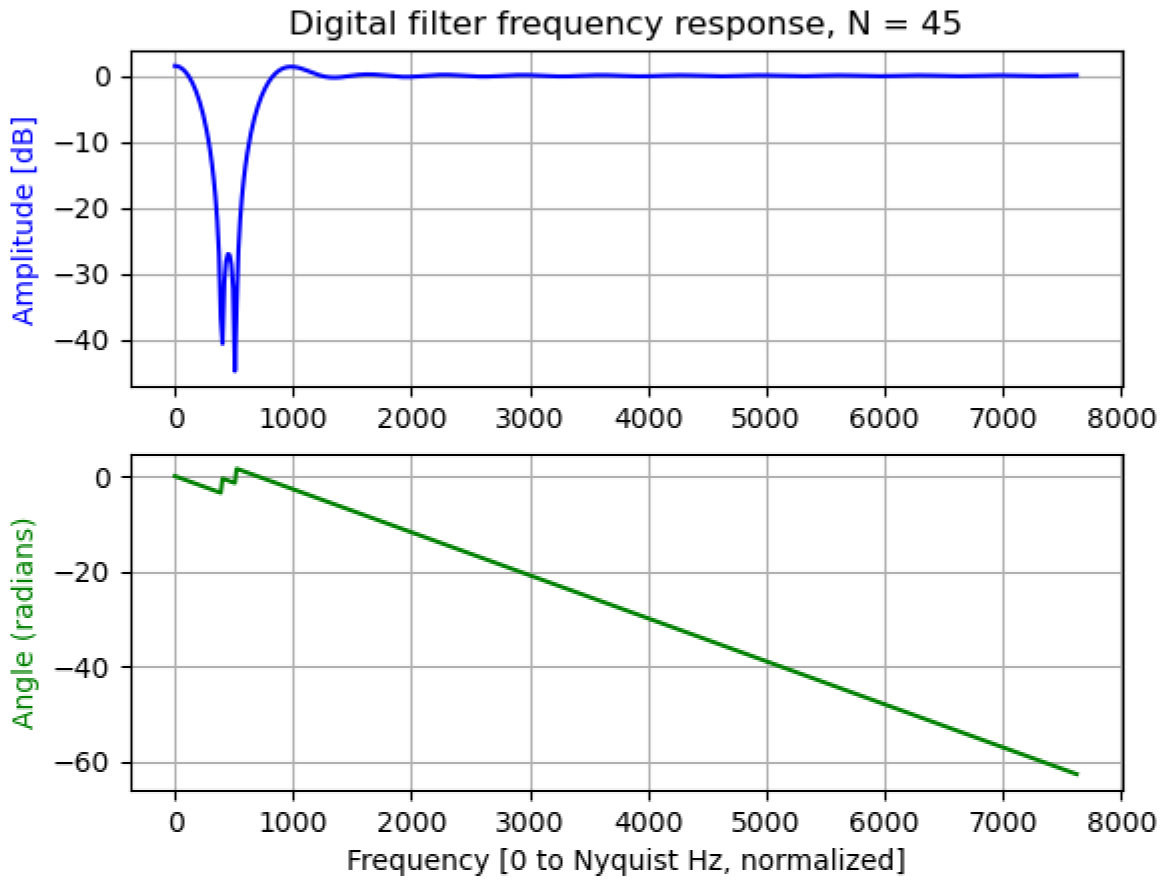
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

# Your filter design here
# firls() can be called via signal.firls()
sampeling_freq = 48*1000
#b1 = signal.firwin(10,900,window = 'hamming',fs = sampeling_freq)
#b2 = signal.firwin(10,2100,window = 'hamming',fs = sampeling_freq)
bands = [0,700,1000,2000,2200,3000,4000,24000]

b= signal.firls(45,bands,[1,1,0,0,1,1,1,1],fs=sampeling_freq)

# Signal analysis
w, h = signal.freqz(b,fs=sampeling_freq)

plt.figure()
plt.subplot(2,1,1)
plt.title('Digital filter frequency response, N = ' + str(len(b)))
plt.plot(w / np.pi, 20 * np.log10(abs(h)), 'b')
plt.ylabel('Amplitude [dB]', color='b')
plt.grid()
plt.axis('tight')

plt.subplot(2,1,2)
angles = np.unwrap(np.angle(h))
plt.plot(w / np.pi, angles, 'g')
plt.ylabel('Angle (radians)', color='g')
plt.grid()
plt.axis('tight')
plt.xlabel('Frequency [0 to Nyquist Hz, normalized]')
plt.show()
np.savetxt('B_FIR',b,delimiter=',')
print (b)
```

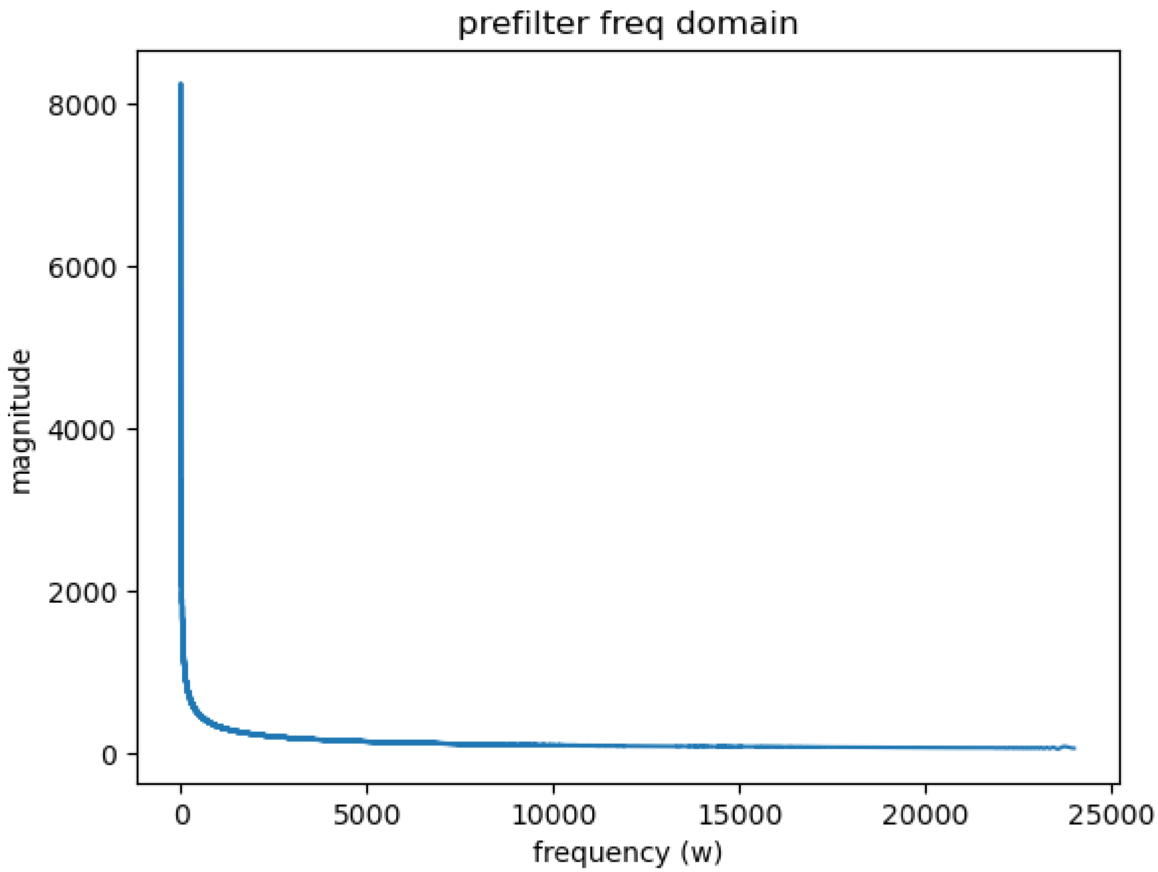



```
[ 0.00948419  0.01511763  0.02121561  0.02734276  0.0329743   0.0375527
  0.04055468  0.04155847  0.04030054  0.03671254  0.03093253  0.02328863
  0.01425797  0.00440811 -0.00566935 -0.01542203 -0.02438708 -0.03222144
 -0.03870909 -0.04374667 -0.0473131  -0.04943154  0.94986682 -0.04943154
 -0.0473131  -0.04374667 -0.03870909 -0.03222144 -0.02438708 -0.01542203
 -0.00566935  0.00440811  0.01425797  0.02328863  0.03093253  0.03671254
  0.04030054  0.04155847  0.04055468  0.0375527   0.0329743   0.02734276
  0.02121561  0.01511763  0.00948419]
```

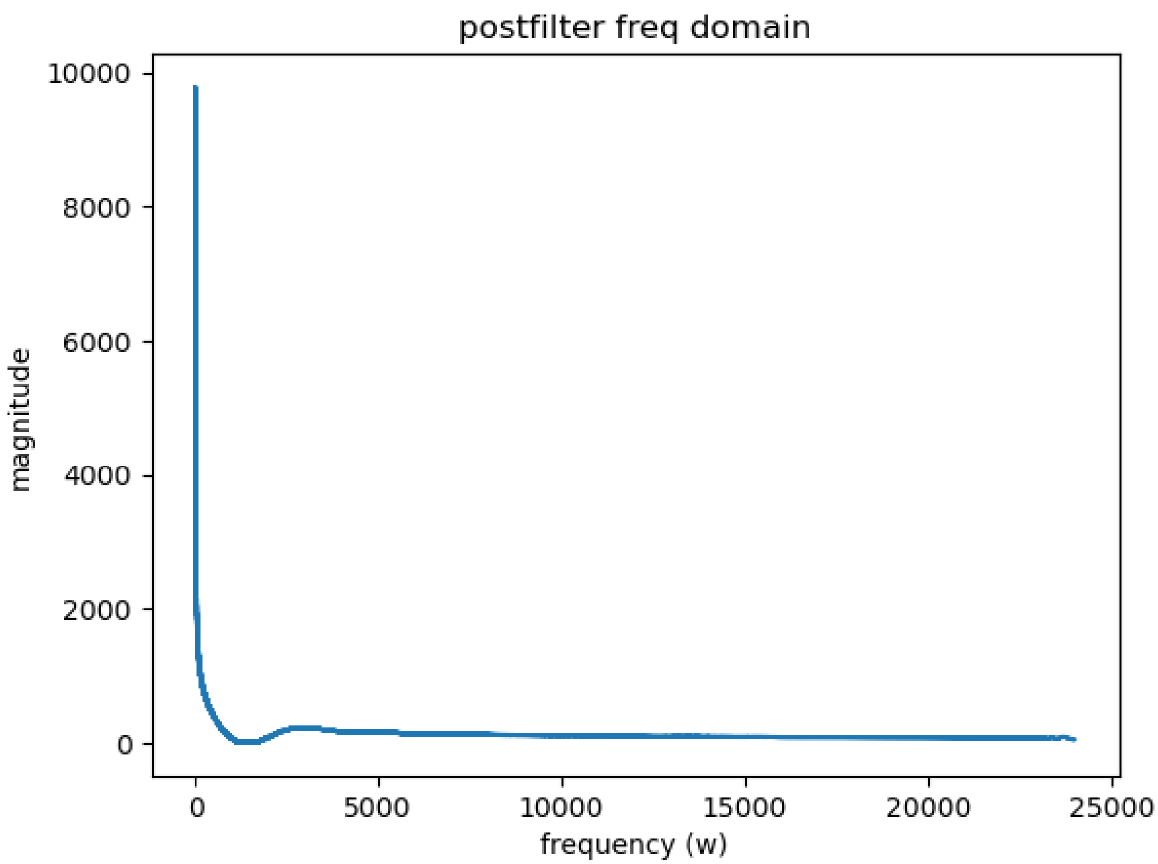
In [2]:
```
##the lower tabs results in better magnitude response, and less taps means les
##In this case, with too few taps, the pass band rippel is too large to be acc
##number 45 was chosen.
```

In [2]:
```
def overflow(a,b):
    if a < b:
        return a
    else:
        return a-b

def apply_FIR(b,data_in):
    filter_len = len(b)
    data_o=[]
    buffer=[]
    #load initial data
    for i in range(0,filter_len):
        buffer.append(data_in[i])
    #start pointer
    pointer = 0
    for j in range(filter_len,len(data_in)):
        data_new = 0
        for k in range(0,filter_len):
            itr = overflow(pointer+k,filter_len)
            data_new = data_new+b[k]*buffer[itr]
        data_o.append(data_new)
        buffer[pointer] = data_in[j]
        pointer = overflow(pointer+1,filter_len)
    return data_o
```

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
F_s = 48000
t = [i / F_s for i in range(2 * F_s)]
test_data = signal.chirp(t, 1, t[-1], 24000, method='logarithmic')


x = np.linspace(0,len(test_data))


##plt.plot(test_data,)

test_fft = np.fft.rfft(test_data)
test_freq = np.fft.fftfreq(test_fft.size,1/F_s)
test_freq1 = np.linspace(0,24000,len(test_fft))
plt.figure()
plt.title('prefilter freq domain')
plt.xlabel('frequency (w)')
plt.ylabel('magnitude')
plt.plot(test_freq1,abs(test_fft))
plt.figure()

data_out = apply_FIR(b,test_data)
data_out_fft = np.fft.fft(data_out)
data_freq = np.fft.fftfreq(data_out_fft.size,1/F_s)

plt.plot(abs(data_freq),abs(data_out_fft))
plt.title('postfilter freq domain')
plt.xlabel('frequency (w)')
plt.ylabel('magnitude')
plt.show()
```



prefilter freq domain
magnitude
frequency (w)
0
2000
4000
6000
8000
0
5000
10000
15000
20000
25000



postfilter freq domain
magnitude
frequency (w)
0
2000
4000
6000
8000
10000
0
5000
10000
15000
20000
25000