In [226]:

```
import numpy as np
import scipy
import math
import scipy.signal
from scipy import signal
import matplotlib.pyplot as plt
```

Problem 1b

In [227]:

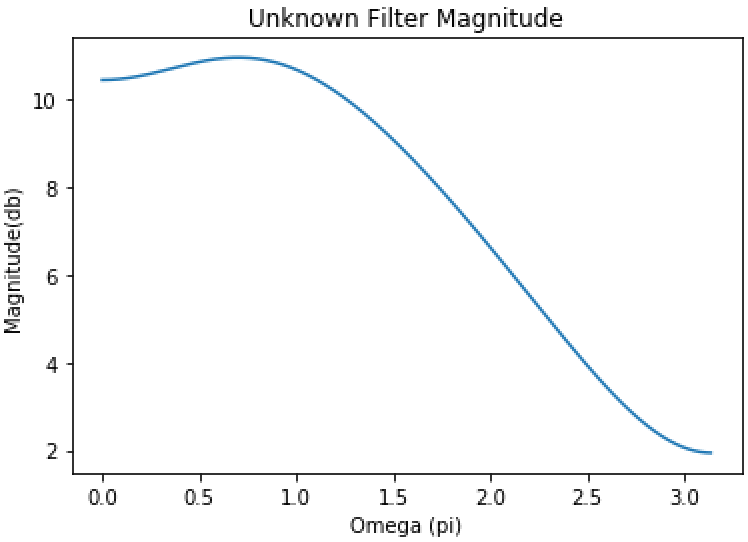
```
d = np.array([0.8, 0.6, 0.5, 0.2, -0.3])
x = np.array([1.0, 0.9, 0.9, 0.7, 0.6, 0.1, 0.1])

delta = 0.05
y = []

error = []

h0 = np.array([1/3, 1/3, 1/3])

def hvalues(h, delta, error, x):
    h1 = h[0] + delta * error * x[2]
    h2 = h[1] + delta * error * x[1]
    h3 = h[2] + delta * error * x[0]
    hvalues = np.array([h1, h2, h3])
    return hvalues

def findy(h, x):
    return x[2] * h[0] + x[1] * h[1] + x[0] * h[2]

def error (d, y):
    return d - y

y.append(findy(h0, x[0:3]))
error1 = d[0] - y[0]
print('The first iteration error is', error1)

h1 = hvalues(h0, delta, error1, x[0:3])

y.append(findy(h1, x[1:4]))
error2 = d[1] - y[1]
print('The second iteration error is', error2)

h2 = hvalues(h1, delta, error2, x[1:4])

y.append(findy(h2, x[2:5]))
error3 = d[2] - y[2]
print('The third iteration error is', error3)
```

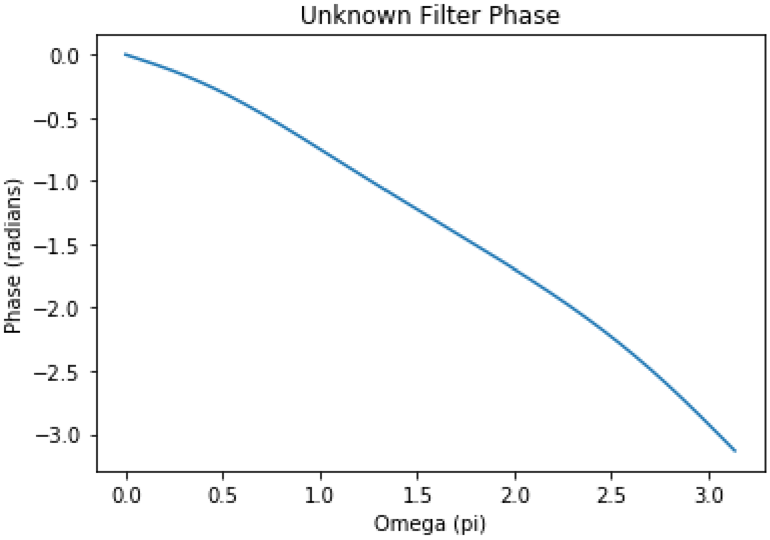
```
The first iteration error is -0.1333333333333333
The second iteration error is -0.21773333333333322
The third iteration error is -0.19928413333333328
```

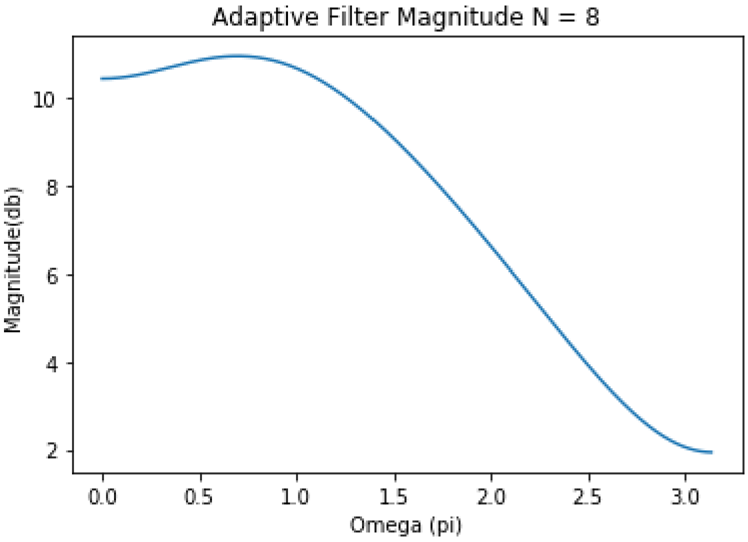
Problem 2

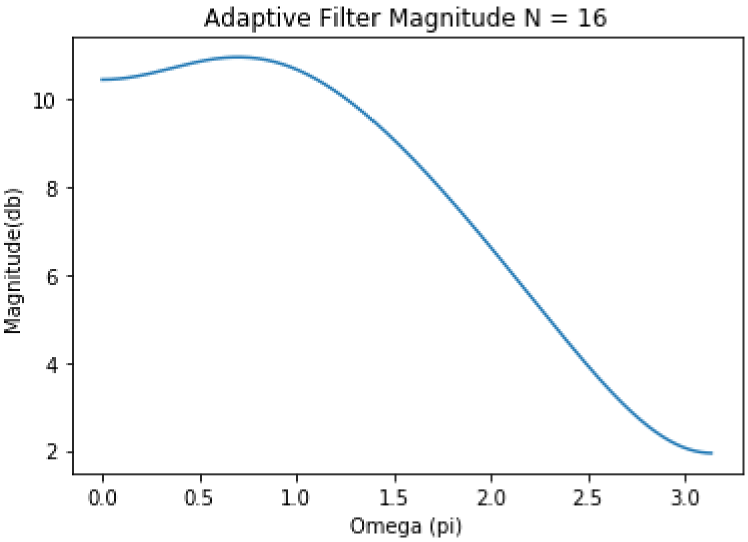
In [228]:

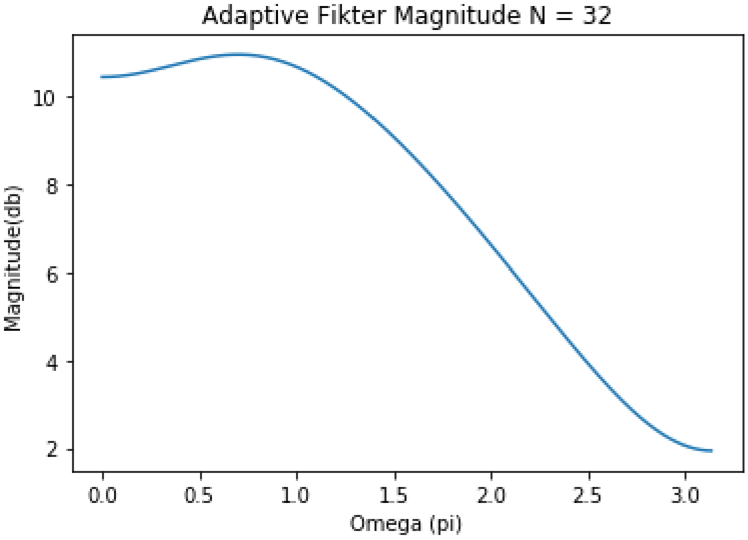
```
# part a - generate input signals
Nx = 1024
x = np.zeros(Nx)
for i in range(0, Nx):
    value = np.random.uniform(-2, 2)
    x[i] = value

# part b - generate output signals
A = np.array([1, -0.5, 0.1])
B = np.array([1, 2, -1])
d = scipy.signal.lfilter(B, A, x)
Nd = len(d)

# part c
Nh8 = 8
hn8 = np.zeros(8)
for i in range(0, Nh8):
    value = np.random.uniform(0, 1)
    hn8[i] = value

Nh16 = 16
hn16 = np.zeros(Nh16)
for i in range(0, Nh16):
    value = np.random.uniform(0, 1)
    hn16[i] = value

Nh32 = 32
hn32 = np.zeros(Nh32)
for i in range(0, Nh32):
    value = np.random.uniform(0, 1)
    hn32[i] = value

delta = 0.02
y8 = np.zeros(Nh8 - 1)
e8 = np.zeros(Nh8 - 1)

y16 = np.zeros(Nh16 - 1)
e16 = np.zeros(Nh16 - 1)

y32 = np.zeros(Nh32 - 1)
e32 = np.zeros(Nh32 - 1)

for i in range(Nh8, Nx - 1):
    y_temp = signal.lfilter(hn8, 1, x[i - Nh8 : i])
    y8 = np.append(y8, y_temp[-1])
    e8 = np.append(e8, d[i - 1] - y8[i - 1])
    hn8 = hn8 + (delta * e8[i - 1] * x[i - Nh8: i][::-1])

for i in range(Nh16, Nx - 1):
    y_temp = signal.lfilter(hn16, 1, x[i - Nh16 : i])
    y16 = np.append(y16, y_temp[-1])
    e16 = np.append(e16, d[i - 1] - y16[i - 1])
    hn16 = hn16 + (delta * e16[i - 1] * x[i - Nh16: i][::-1])

for i in range(Nh32, Nx - 1):
    y_temp = signal.lfilter(hn32, 1, x[i - Nh3 : i])
    y32 = np.append(y32, y_temp[-1])
    e32 = np.append(e32, d[i - 1] - y32[i - 1])
    hn32 = hn32 + (delta * e32[i - 1] * x[i - Nh32: i][::-1])

#part d
w, h = scipy.signal.freqz(b, a)
Hz_phase = np.angle(h)
Hz_db = 20 * np.log10(np.abs(h))

plt.title('Unknown Filter Magnitude')
plt.ylabel('Magnitude(db)')
plt.xlabel('Omega (pi)')
plt.plot(w, Hz_db)
plt.show()

plt.title('Unknown Filter Phase')
plt.ylabel('Phase (radians)')
plt.xlabel('Omega (pi)')
plt.plot(w, Hz_phase)
plt.show()

plt.title('Adaptive Filter Magnitude N = 8')
plt.ylabel('Magnitude(db)')
plt.xlabel('Omega (pi)')
wn8, hn_8 = scipy.signal.freqz(hn1, [1])
Hn_db8 = 20 * np.log10(np.abs(hn_f8))
plt.plot(wn8, Hn_db8)
plt.show()

plt.title('Adaptive Filter Magnitude N = 16')
plt.ylabel('Magnitude(db)')
plt.xlabel('Omega (pi)')
wn16, hn_16 = scipy.signal.freqz(hn2, [1])
Hn_db16 = 20 * np.log10(np.abs(hn_f16))
plt.plot(wn16, Hn_db16)
plt.show()

plt.title('Adaptive Fikter Magnitude N = 32')
plt.ylabel('Magnitude(db)')
plt.xlabel('Omega (pi)')
wn32, hn_32 = scipy.signal.freqz(hn3, [1])
Hn_db32 = 20 * np.log10(np.abs(hn_f32))
plt.plot(wn32, Hn_db32)
plt.show()

plt.title('Adaptive Filter Phase, N = 8')
plt.ylabel('Phase (radians)')
plt.xlabel('Omega (pi)')
plt.plot(wn, np.angle(hn_8))
plt.show()

plt.title('Adaptive Filter Phase, N = 16')
plt.ylabel('Phase (radians)')
plt.xlabel('Omega (pi)')
plt.plot(wn, np.angle(hn_16))
plt.show()
plt.title('Adaptive Filter Phase, N = 32')
plt.ylabel('Phase (radians)')
plt.xlabel('Omega (pi)')
plt.plot(wn, np.angle(hn_32))
plt.show()

plt.title('Error Plot N = 8')
plt.xlabel('n')
plt.ylabel('Error')
plt.plot(e8)
plt.show()

plt.title('Error Plot N = 16')
plt.xlabel('n')
plt.ylabel('Error')
plt.plot(e16)
plt.show()

plt.title('Error Plot N = 32')
plt.xlabel('n')
plt.ylabel('Error')
plt.plot(e32)
plt.show()
```

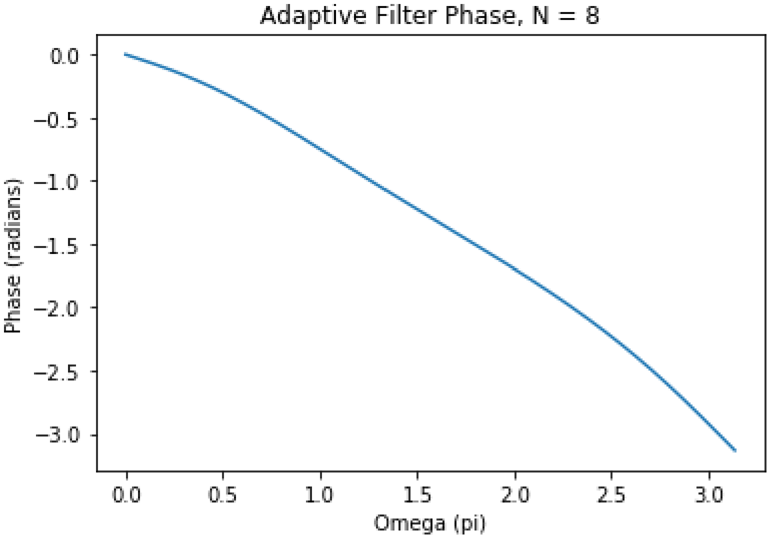

Unknown Filter Magnitude
Magnitude(db)
10
8
6
4
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Omega (pi)

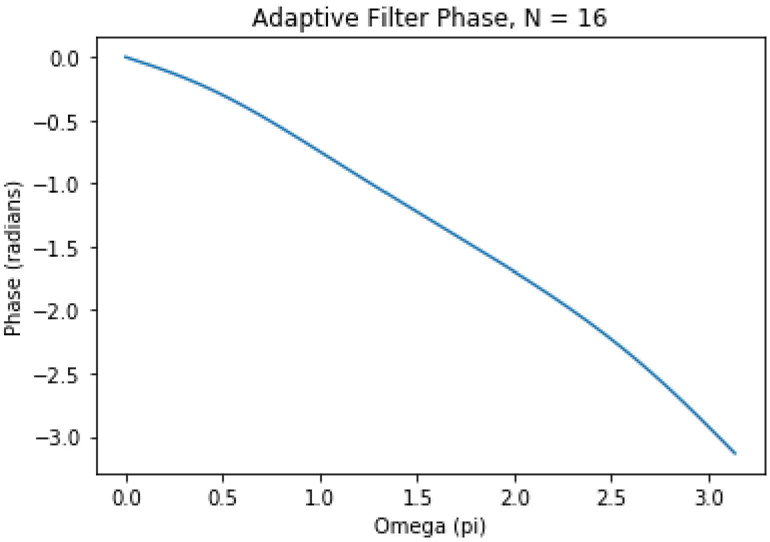

Unknown Filter Phase
Phase (radians)
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Omega (pi)

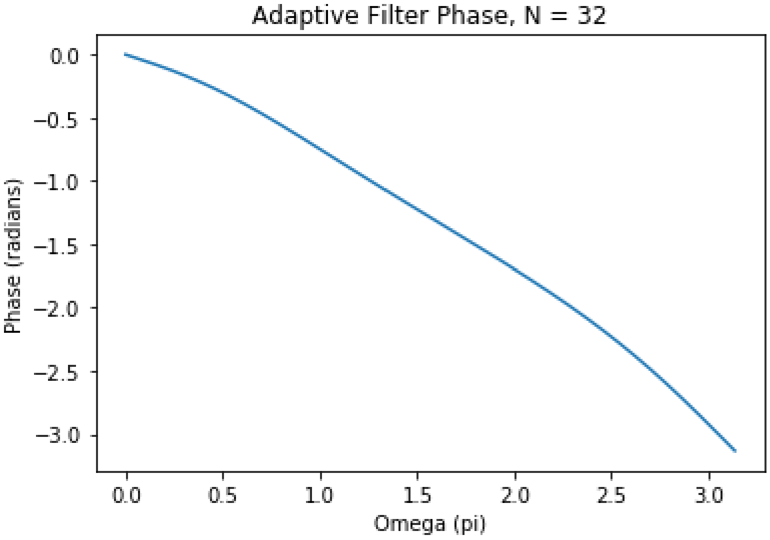

Adaptive Filter Magnitude N = 8
Magnitude(db)
10
8
6
4
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Omega (pi)

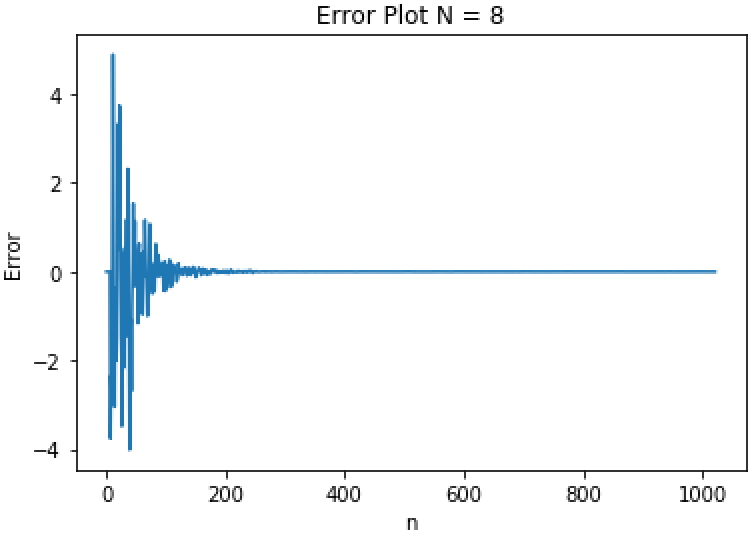

Adaptive Filter Magnitude N = 16
Magnitude(db)
10
8
6
4
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Omega (pi)

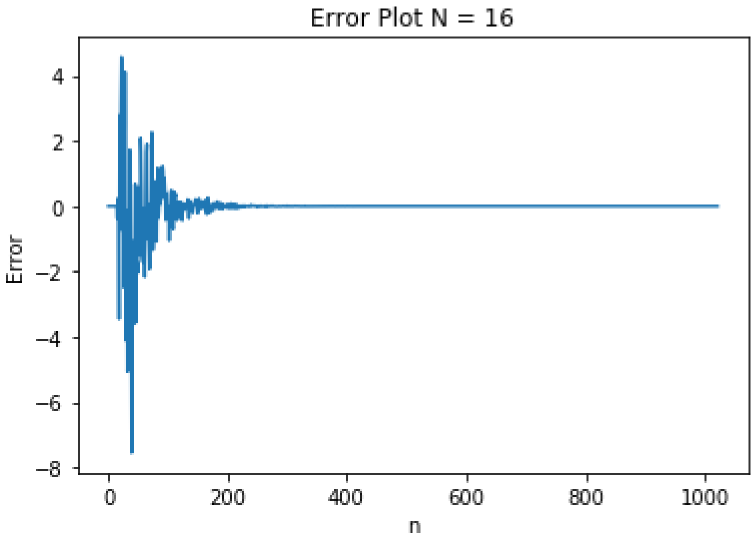

Adaptive Fikter Magnitude N = 32
Magnitude(db)
10
8
6
4
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Omega (pi)

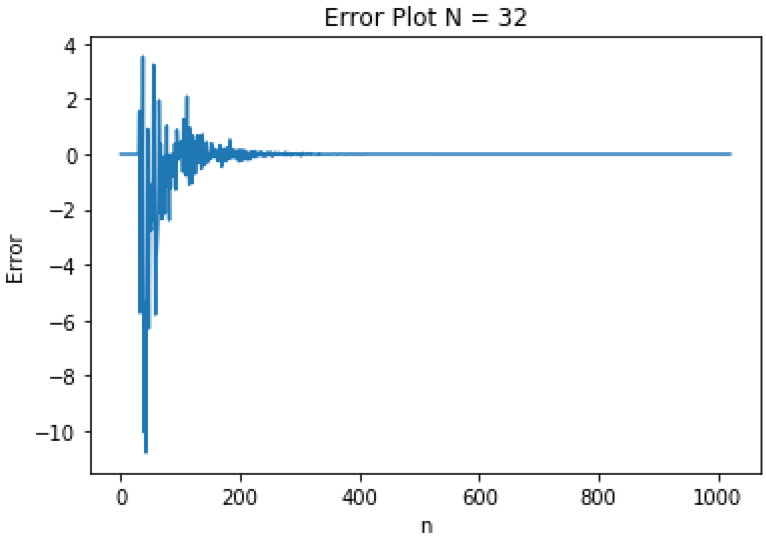

Adaptive Filter Phase, N = 8
Phase (radians)
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Omega (pi)


Adaptive Filter Phase, N = 16
Phase (radians)
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Omega (pi)


Adaptive Filter Phase, N = 32
Phase (radians)
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Omega (pi)


Error Plot N = 8
Error
4
2
0
−2
−4
0
200
400
600
800
1000
n





As N increases, I noticed that the error functions take slightly longer to converge. Showing that there is slightly more error as you increase N. However, all adaptive filters have the same general form as the unknown filter.

Problem 3

In [229]:

```
def lms(x, d, hn):
    delta = 0.0001
    delay = 1
    Nx = len(x)
    Nh = len(hn)
    w = []
    w.append(hn)

    s_hat = np.zeros(Nh)
    e = np.zeros(Nh)
    for i in range(Nh + delay, Nx - 1):
        sn = signal.lfilter(hn, 1, x[i - Nh : i])
        s_hat = np.append(s_hat, sn[-1])
        e = np.append(e, x[i - 1] - s_hat[i - 1])
        hn = hn + (delta * e[i - 1] * x[i - Nh : i][: : -1])
        w.append(hn)
    return s_hat[Nh:], e[Nh:], np.asarray(w)
```

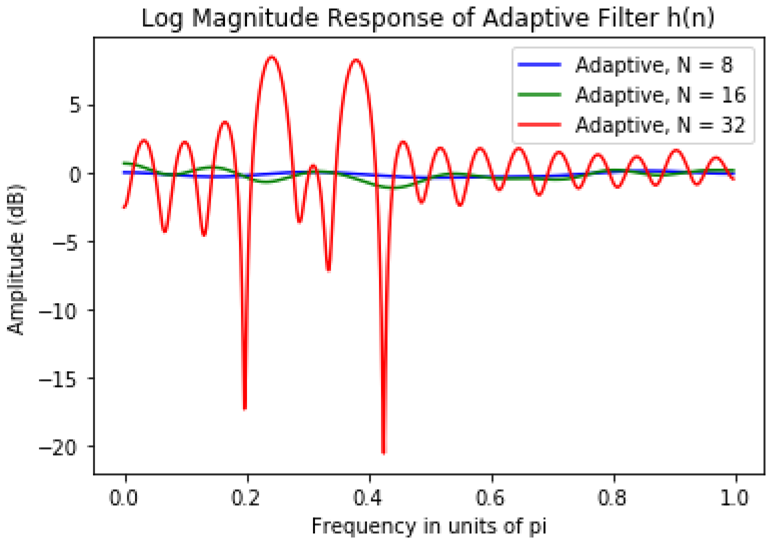
In [230]:

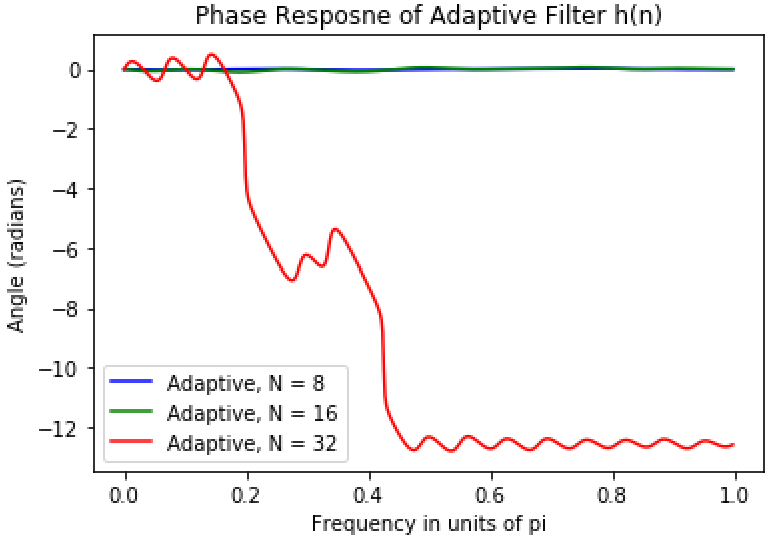
```
def generatehn(Nh):
    array = np.zeros(Nh)
    for i in range(0, Nh):
        array[i] = np.random.uniform(-1, 1)
    return array
```

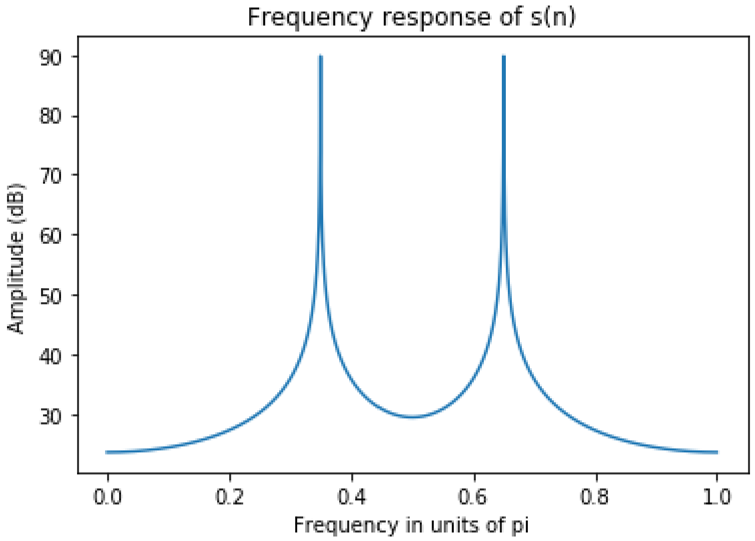
In [232]:

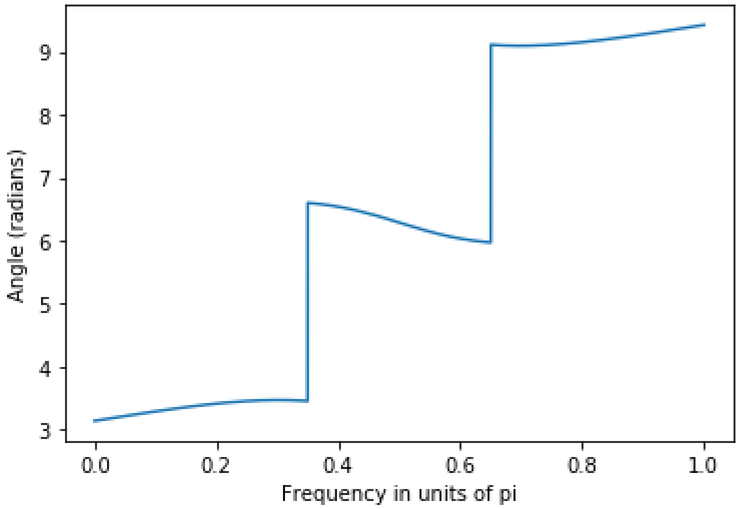
```
# part a
Ns = 4096
w = np.zeros(Ns)
for i in range(0, Ns):
    value = np.random.uniform(-5, 5)
    w[i] = value

n = np.arange(Ns)
s = 20 * np.sin(0.3 * np.pi * n)
x = w + s

# part b
hn8 = generatehn(8)
s8, e8final, w8 = lms(x, 'x', hn8)

hn16 = generatehn(16)
s16, e16final, w16 = lms(x, 'x', hn16)

hn32 = generatehn(32)
s32, e32final, w32 = lms(x, 'x', hn32)

# part c

#resulting adaptive filter h(n)
adapt8w, adapt8h = signal.freqz(w8[-1], 1)
adapt8phase = np.unwrap(np.angle(adapt8h))

adapt16w, adapt16h = signal.freqz(w16[-1], 1)
adapt16phase = np.unwrap(np.angle(adapt16h))

adapt32w, adapt32h = signal.freqz(w32[-1], 1)
adapt32phase = np.unwrap(np.angle(adapt32h))

plt.plot(adapt8w / np.pi, 20 * np.log10(np.abs(adapt8h)), 'b', label='Adaptiv
e, N = 8')
plt.plot(adapt16w / np.pi, 20 * np.log10(np.abs(adapt16h)), 'g', label='Adapti
ve, N = 16')
plt.plot(adapt32w / np.pi, 20 * np.log10(np.abs(adapt32h)), 'r', label='Adapti
ve, N = 32')
plt.xlabel('Frequency in units of pi')
plt.ylabel('Amplitude (dB)')
plt.title("Log Magnitude Response of Adaptive Filter h(n)")
plt.legend()
plt.show()

plt.plot(adapt8w / np.pi, adapt8phase, 'b', label='Adaptive, N = 8')
plt.plot(adapt16w / np.pi, adapt16phase, 'g', label='Adaptive, N = 16')
plt.plot(adapt32w / np.pi, adapt32phase, 'r', label='Adaptive, N = 32')
plt.title('Phase Resposne of Adaptive Filter h(n)')
plt.xlabel('Frequency in units of pi')
plt.ylabel('Angle (radians)')
plt.legend()
plt.show()

#sinusoidal signal s(n)
w = np.linspace(0, np.pi, Ns)
```

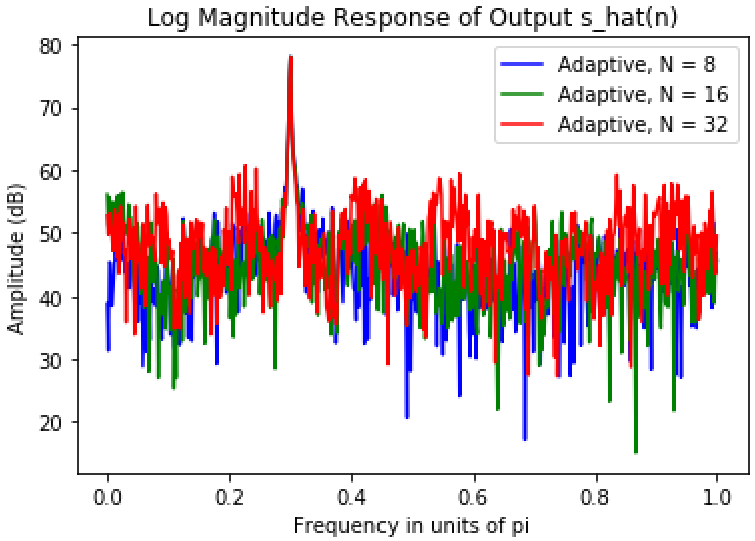
```
h = np.fft.fft(s)
h = np.fft.fftshift(h)

plt.plot(w / np.pi, 20 * np.log10(abs(h)))
plt.title('Frequency response of s(n)')
plt.xlabel('Frequency in units of pi')
plt.ylabel('Amplitude (dB)')
plt.show()

plt.xlabel('Frequency in units of pi')
plt.ylabel('Angle (radians)')
plt.plot(w / np.pi, np.unwrap(np.angle(h)))
plt.show()

#adaptive filter s_hat(n)
s8w, s8h = signal.freqz(s8, 1)
s16w, s16h = signal.freqz(s16, 1)
s32w, s32h = signal.freqz(s32, 1)
s8phase = np.angle(s8h)
s16phase = np.angle(s16h)
s32phase = np.angle(s32h)

w8 = np.linspace(0, np.pi, len(s8w))
w16 = np.linspace(0, np.pi, len(s16w))
w32 = np.linspace(0, np.pi, len(s32w))

plt.plot(w8 / np.pi, 20 * np.log10(np.abs(s8h)), 'b', label='Adaptive, N = 8')
plt.plot(w16 / np.pi, 20 * np.log10(np.abs(s16h)), 'g', label='Adaptive, N = 1
6')
plt.plot(w32 / np.pi, 20 * np.log10(np.abs(s32h)), 'r', label='Adaptive, N = 3
2')
plt.xlabel('Frequency in units of pi')
plt.ylabel('Amplitude (dB)')
plt.title("Log Magnitude Response of Output s_hat(n)")
plt.legend()
plt.show()

plt.plot(w8 / np.pi, s8phase, 'b', label='Adaptive, N = 8')
plt.plot(w16 / np.pi, s16phase, 'g', label='Adaptive, N = 16')
plt.plot(w32 / np.pi, s32phase, 'r', label='Adaptive, N = 32')
plt.title('Phase Response of Output s_hat(n)')
plt.xlabel('Frequency in units of pi')
plt.ylabel('Angle (radians)')
plt.legend()
plt.show()

#error
plt.title('Error of N = 8')
plt.xlabel('n')
plt.ylabel('Error')
plt.plot(e8final)
plt.show()

plt.title('Error of N = 16')
plt.xlabel('n')
plt.ylabel('Error')
plt.plot(e16final)
plt.show()

plt.title('Error of N = 32')
plt.xlabel('n')
plt.ylabel('Error')
plt.plot(e32final)
plt.show()
```

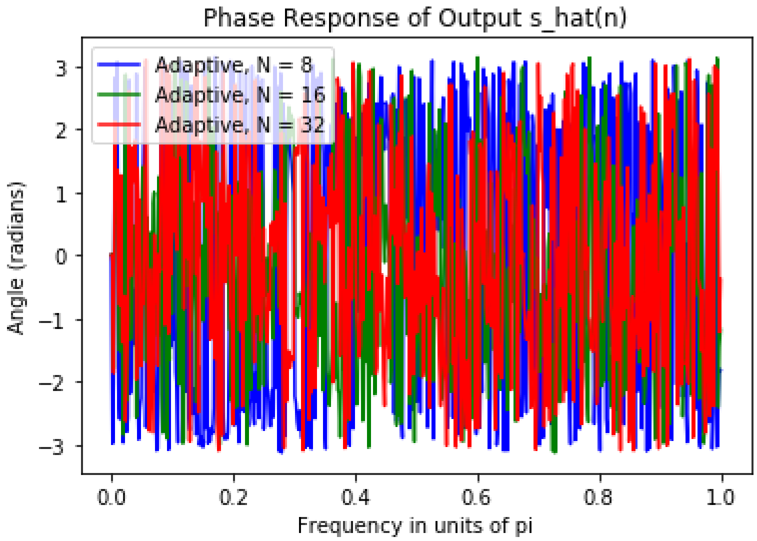

Log Magnitude Response of Adaptive Filter h(n)
Adaptive, N = 8
Adaptive, N = 16
Adaptive, N = 32
Amplitude (dB)
5
0
−5
−10
−15
−20
0.0
0.2
0.4
0.6
0.8
1.0
Frequency in units of pi

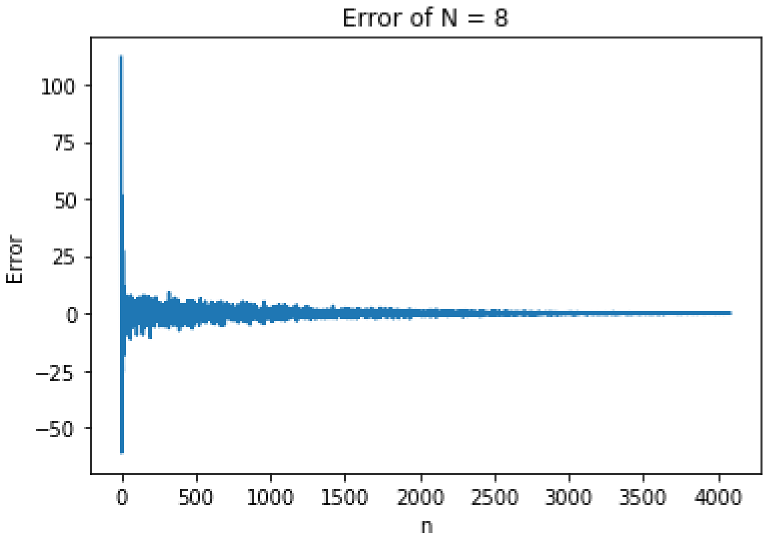

Phase Resposne of Adaptive Filter h(n)
Angle (radians)
0
−2
−4
−6
−8
−10
−12
Adaptive, N = 8
Adaptive, N = 16
Adaptive, N = 32
0.0
0.2
0.4
0.6
0.8
1.0
Frequency in units of pi

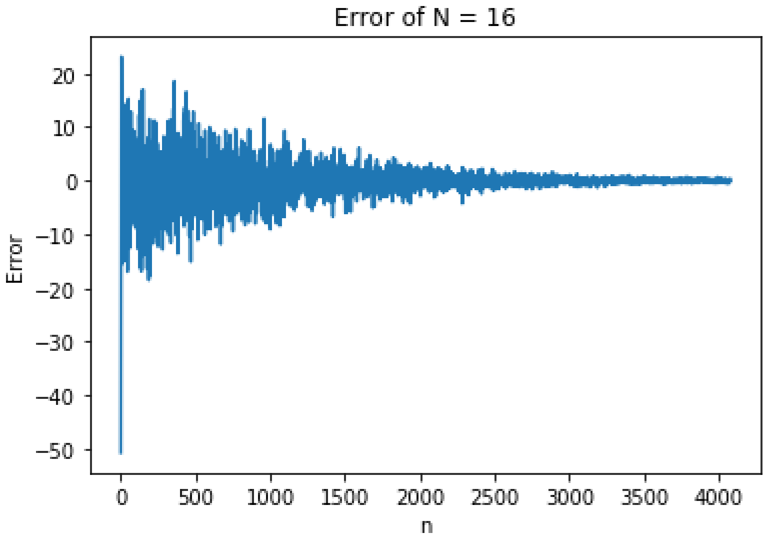

Frequency response of s(n)
Amplitude (dB)
90
80
70
60
50
40
30
0.0
0.2
0.4
0.6
0.8
1.0
Frequency in units of pi

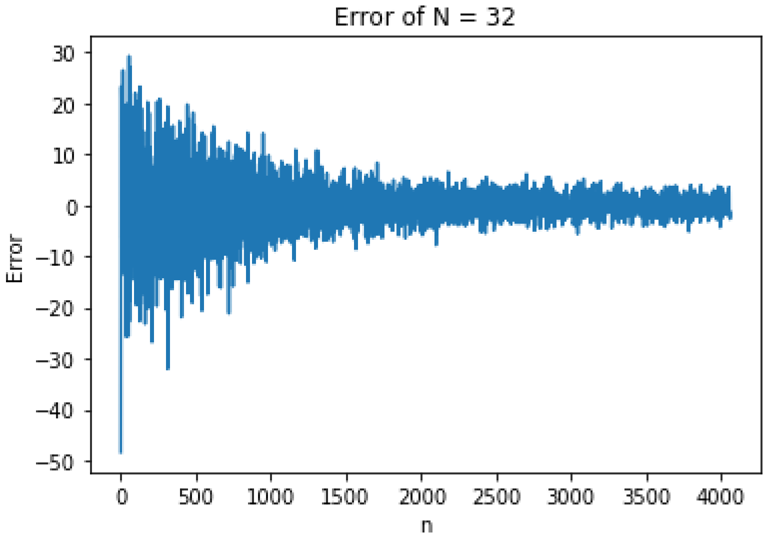

Angle (radians)
3
4
5
6
7
8
9
0.0
0.2
0.4
0.6
0.8
1.0
Frequency in units of pi


Log Magnitude Response of Output s_hat(n)
Adaptive, N = 8
Adaptive, N = 16
Adaptive, N = 32
Amplitude (dB)
20
30
40
50
60
70
80
0.0
0.2
0.4
0.6
0.8
1.0
Frequency in units of pi


Phase Response of Output s_hat(n)
Adaptive, N = 8
Adaptive, N = 16
Adaptive, N = 32
Angle (radians)
−3
−2
−1
0
1
2
3
0.0
0.2
0.4
0.6
0.8
1.0
Frequency in units of pi


Error of N = 8
Error
100
75
50
25
0
−25
−50
0
500
1000
1500
2000
2500
3000
3500
4000
n


Error of N = 16
Error
20
10
0
−10
−20
−30
−40
−50
0
500
1000
1500
2000
2500
3000
3500
4000
n


Error of N = 32
Error
30
20
10
0
−10
−20
−30
−40
−50
0
500
1000
1500
2000
2500
3000
3500
4000
n

In [ ]: