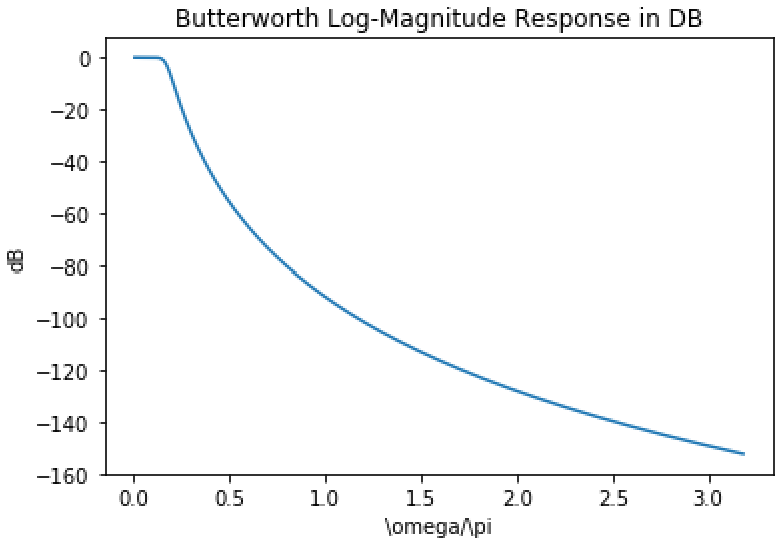
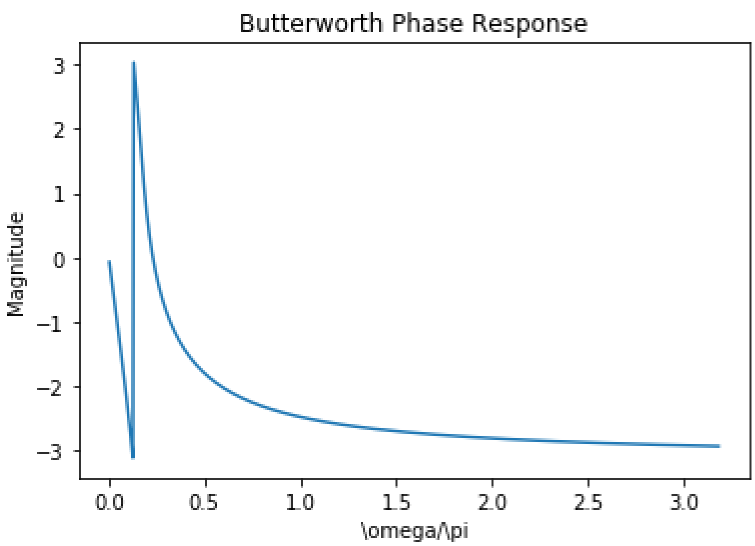
In [117]:
```
import numpy as np
import scipy
import math
import scipy.signal
from scipy import signal
import matplotlib.pyplot as plt
```

Problem 1

In [126]:
```
omegap = 0.15 * np.pi
omegas = 0.25 * np.pi
Rp = 1.2
As = 18

numerator = np.log10((10**(Rp/10) - 1) / (10**(As/10) - 1))
denominator = 2 * np.log10(omegap / omegas)

N = math.ceil((numerator / denominator))
k = np.arange(N)

omegac1 = omegap / ((10**(Rp/10) - 1)**(1 / (2 * N)))
omegac2 = omegas / ((10**(As/10) - 1)**(1 / (2 * N)))
print(omegac1)
print(omegac2)

omegac = (omegac1 + omegac2) / 2

pk = omegac * np.exp(1j * np.pi/12 * (2*k + n + 1))
real = np.real(np.poly(pk))

w, h = signal.freqs([omegac ** n], real)

mag = np.abs(h)
dB = 20 * np.log10((mag) / np.max(mag))
phase = np.angle(h)

plt.plot(w / np.pi, dB)
plt.title("Butterworth Log-Magnitude Response in DB")
plt.ylabel('dB')
plt.xlabel('\omega/\pi')
plt.show()


plt.plot(w / np.pi, phase)
plt.title("Butterworth Phase Response")
plt.ylabel('Magnitude')
plt.xlabel('\omega/\pi')
plt.show()
```

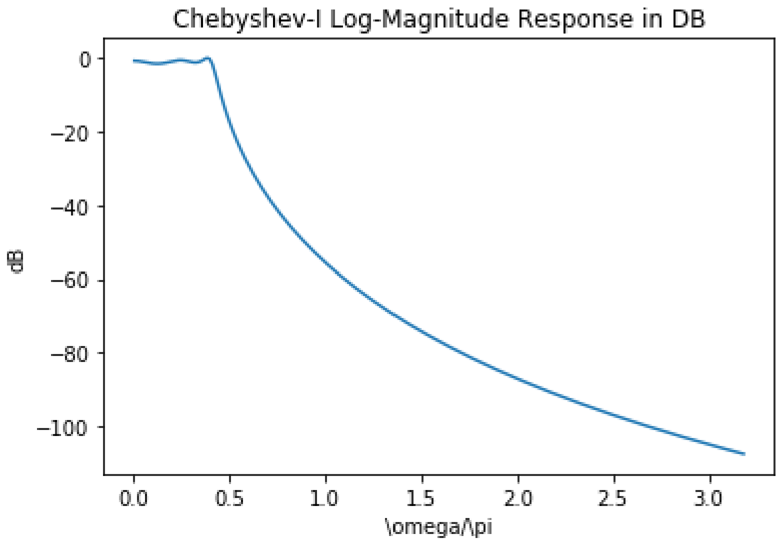
```
0.5184135341697546
0.5567600528034354
```



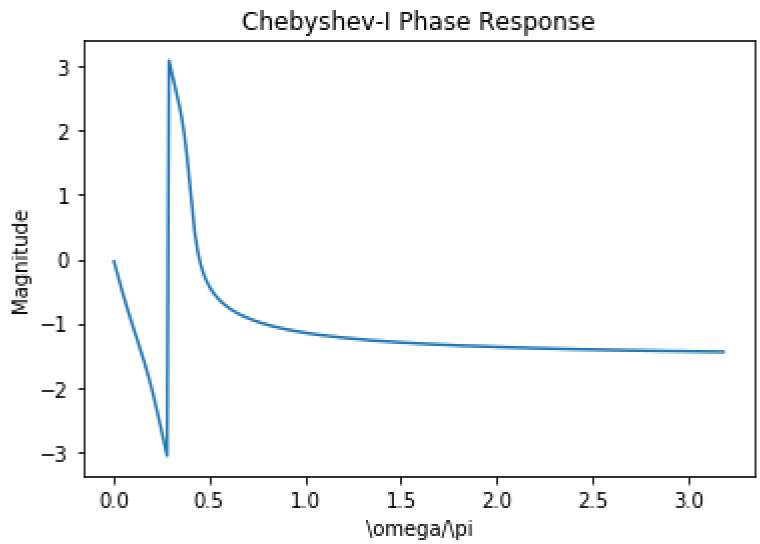


Problem 2

In [128]:
```
omegap = 0.4 * np.pi
omegas = 0.6 * np.pi
omegac = omegap
Rp = 0.8
As = 25
epsilon = np.sqrt(10**(0.1 * Rp) - 1)
A = 10**(As / 20)
omegar = omegas / omegap
g = np.sqrt((A**2 - 1) / epsilon**2)

N = np.log10(g + np.sqrt(g**2 - 1)) / np.log10((omegar + np.sqrt(omegar**2 - 1
)))
k = np.arange(N)

alpha = (1 / epsilon) + np.sqrt(1 + (1 / epsilon**2))

a = (1 / 2) * (alpha**(1/N) - alpha**(-1 / N))
b = (1 / 2) * (alpha**(1/N) + alpha**(-1 / N))

sigmak = a * omegac * np.cos((np.pi / 2) + ((2 * k + 1) * np.pi) / (2 * N))
omegak = b * omegac * np.sin((np.pi / 2) + ((2 * k + 1) * np.pi) / (2 * N))

pk = sigmak + 1j * omegak
real = np.real(np.poly(pk))
w, h = signal.freqs([omegac ** n], real)

mag = np.abs(h)
dB = 20 * np.log10((mag) / np.max(mag))
phase = np.angle(h)

plt.plot(w / np.pi, dB)
plt.title("Chebyshev-I Log-Magnitude Response in DB")
plt.ylabel('dB')
plt.xlabel('\omega/\pi')
plt.show()


plt.plot(w / np.pi, phase)
plt.title("Chebyshev-I Phase Response")
plt.ylabel('Magnitude')
plt.xlabel('\omega/\pi')
plt.show()
```



Chebyshev-I Log-Magnitude Response in DB
dB
0
−20
−40
−60
−80
−100
0.0
0.5
1.0
1.5
2.0
2.5
3.0
\omega/\pi

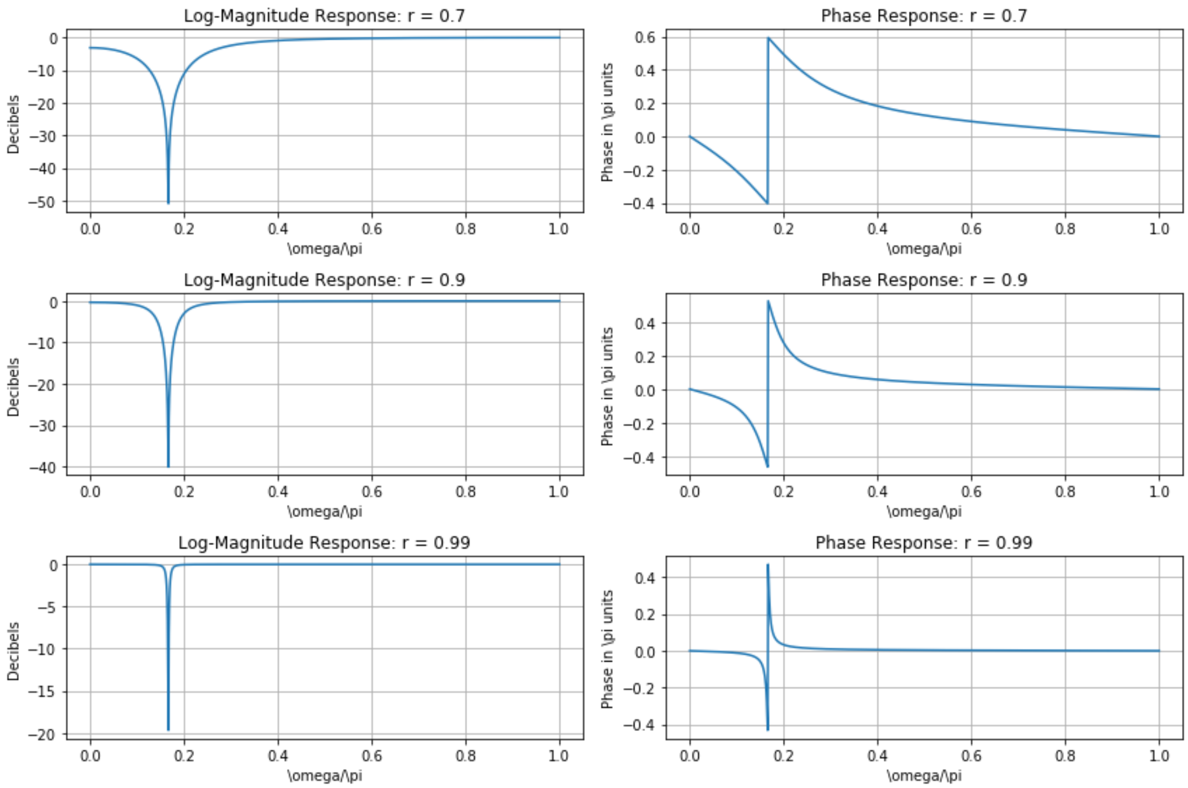


Chebyshev-I Phase Response
Magnitude
3
2
1
0
−1
−2
−3
0.0
0.5
1.0
1.5
2.0
2.5
3.0
\omega/\pi

Problem 3

In [132]:
```
omega0 = np.pi / 6
z1 = np.exp(1j * omega0)
z2 = np.conj(z1)
b = np.poly([z1, z2])

""" (a) Frequency Response Plots """
# r1 = 0.7
r1 = 0.7
p1 = r1 * np.exp(1j * omega0)
p2 = np.conj(p1)
a = np.poly([p1, p2])
om = np.pi * np.linspace(0, 1, 1001)
W1, H1 = scipy.signal.freqz(b, a, om)
Hmag1 = np.abs(H1)
Hpha1 = np.angle(H1)
Hdb1 = 20 * np.log10(Hmag1 / np.max(Hmag1))

# Log-Magnitude Response
fig = plt.subplots(3, 2, figsize = (12, 8))
plt.subplot(3, 2, 1)
plt.plot(om / np.pi, Hdb1)
plt.xlabel('\omega/\pi')
plt.ylabel('Decibels')
plt.title('Log-Magnitude Response: r = 0.7')
plt.grid(True)

# Phase Response
plt.subplot(3, 2, 2)
plt.plot(om / np.pi, Hpha1 / np.pi)
plt.xlabel('\omega/\pi')
plt.ylabel('Phase in \pi units')
plt.title('Phase Response: r = 0.7')
plt.grid(True)

# r2 = 0.9
r2 = 0.9
p1 = r2 * np.exp(1j * omega0)
p2 = np.conj(p1)
a = np.poly([p1, p2])
om = np.pi * np.linspace(0, 1, 1001)
W2, H2 = scipy.signal.freqz(b, a, om)
Hmag2 = np.abs(H2)
Hpha2 = np.angle(H2)
Hdb2 = 20 * np.log10(Hmag2 / np.max(Hmag2))

# Log-Magnitude Response
plt.subplot(3, 2, 3)
plt.plot(om / np.pi, Hdb2)
plt.xlabel('\omega/\pi')
plt.ylabel('Decibels')
plt.title('Log-Magnitude Response: r = 0.9')
plt.grid(True)

# Phase Response
plt.subplot(3, 2, 4)
plt.plot(om / np.pi, Hpha2 / np.pi)
```

```
plt.xlabel('\omega/\pi')
plt.ylabel('Phase in \pi units')
plt.title('Phase Response: r = 0.9')
plt.grid(True)


# r3 = 0.99
r3 = 0.99
p1 = r3 * np.exp(1j * omega0)
p2 = np.conj(p1)
a = np.poly([p1, p2])
om = np.pi * np.linspace(0, 1, 1001)
W3, H3 = scipy.signal.freqz(b, a, om)
Hmag3 = np.abs(H3)
Hpha3 = np.angle(H3)
Hdb3 = 20 * np.log10(Hmag3 / np.max(Hmag3))

# Log-Magnitude Response
plt.subplot(3, 2, 5)
plt.plot(om / np.pi, Hdb3)
plt.xlabel('\omega/\pi')
plt.ylabel('Decibels')
plt.title('Log-Magnitude Response: r = 0.99')
plt.grid(True)

# Phase Response
plt.subplot(3, 2, 6)
plt.plot(om / np.pi, Hpha3 / np.pi)
plt.xlabel('\omega/\pi')
plt.ylabel('Phase in \pi units')
plt.title('Phase Response: r = 0.99')
plt.grid(True)

plt.tight_layout()
# plt.show()
```



In [133]:
```
""" (b) 3dB bandwidths """
r = 0.7
print('r={}'.format(r))
I = np.where(Hdb1 < -3)[0]
Imin = np.min(I)
Imax = np.max(I)
BW3db_computed = (Imax - Imin) * np.pi / 1000
print('BW3db_computed={}'.format(BW3db_computed))

r = 0.9
print('r={}'.format(r))
I = np.where(Hdb2 < -3)[0]
Imin = np.min(I)
Imax = np.max(I)
BW3db_computed = (Imax - Imin) * np.pi / 1000
print('BW3db_computed={}'.format(BW3db_computed))

r = 0.99
print('r={}'.format(r))
I = np.where(Hdb3 < -3)[0]
Imin = np.min(I)
Imax = np.max(I)
BW3db_computed = (Imax - Imin) * np.pi / 1000
print('BW3db_computed={}'.format(BW3db_computed))
```

```
r=0.7
BW3db_computed=0.8859291283123216
r=0.9
BW3db_computed=0.21048670779051615
r=0.99
BW3db_computed=0.015707963267948967
```

In [134]:
```
""" (c) Value of r for 3dB bandwidth of 0.04 radians """
# 'Trial and Error' Approach stands for 'Guess and Check', let's try r = 0.979
0, the computed 3dB bandwidth is 0.041 radians, which makes sense.

r4 = 0.9790
p1 = r4 * np.exp(1j * omega0)
p2 = np.conj(p1)
a = np.poly([p1, p2])
om = np.pi * np.linspace(0, 1, 1001)
W4, H4 = scipy.signal.freqz(b, a, om)
Hmag4 = np.abs(H4)
Hpha4 = np.angle(H4)
Hdb4 = 20 * np.log10(Hmag4 / np.max(Hmag4))

I = np.where(Hdb4 < -3)[0]
Imin = np.min(I)
Imax = np.max(I)
BW3db_computed = (Imax - Imin) * np.pi / 1000
print('r={}'.format(r))
print('BW3db_computed={}'.format(BW3db_computed))
```

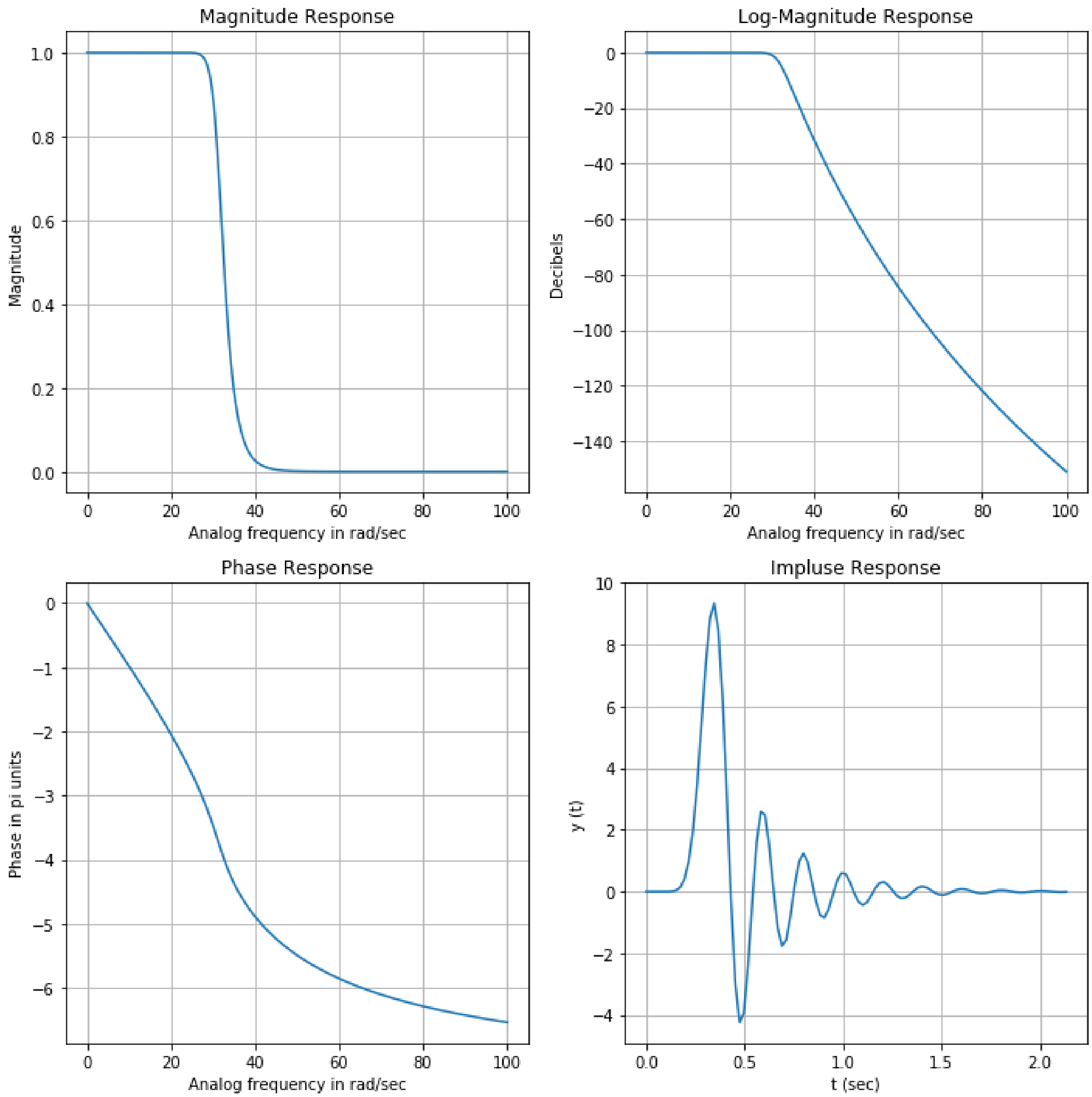
```
r=0.99
BW3db_computed=0.04084070449666732
```

Problem 4

In [135]:
```
def u_buttap(N, Wc):
    # b, a = u_buttap(N, Wc);
    # b = numerator polynomial coefficients of Ha(s)
    # a = denominator polynomial coefficients of Ha(s)
    # N = Order of the Butterworth Filter
    # Wc = Cutoff frequency in radians/sec
    z, p, k = scipy.signal.buttap(N)
    p = p * Wc
    k = k * (Wc ** N)
    B = np.real(np.poly(z))
    b0 = k
    b = k * B
    a = np.real(np.poly(p))
    return b, a


def afd_butt(Wp, Ws, Rp, As):
    # b, a = afd_butt(Wp, Ws, Rp, As);
    # b = Numerator coefficients of Ha(s)
    # a = Denominator coefficients of Ha(s)
    # Wp = Passband edge frequency in rad/sec (Wp > 0)
    # Ws = Stopband edge frequency in rad/sec (Ws > Wp > 0)
    # Rp = Passband ripple in +dB (Rp > 0)
    # As = Stopband attenuation in +dB (As > 0)
    if Wp <= 0:
        print('Passband edge must be larger than 0')
        return
    if Ws <= Wp:
        print('Stopband edge must be larger than Passband edge')
        return

    if (Rp <= 0) or (As < 0):
        print('PB ripple and/or SB attenuation ust be larger than 0')
        return
    N = np.ceil(np.log10((10**(Rp / 10) - 1) / (10**(As / 10) - 1)) /
                (2 * np.log10(Wp / Ws)))
    print('*** Butterworth Filter Order = {} \n'.format(N))

    Wc = Wp / ((10**(Rp / 10) - 1) ** (1 / (2 * N)))
    b, a = u_buttap(N, Wc)
    return b, a


def freqs_m(b, a, wmax):
    w = np.arange(0, 501) * wmax / 500
    W, H = scipy.signal.freqs([b], a, w)
    mag = np.abs(H)
    db = 20 * np.log10((mag) / np.max(mag))
    pha = np.angle(H)
    return db, mag, pha, w


Wp = 30
Ws = 40
Rp = 1
As = 30
```

```
b, a = afd_butt(Wp, Ws, Rp, As)
Wmax = 100
db, mag, pha, w = freqs_m(b, a, Wmax)
pha = np.unwrap(pha)
# print(b, a)
t, y = signal.impulse2(([b], a))


fig = plt.subplots(2,2, figsize = (10, 10))
plt.subplot(2, 2, 1)
plt.plot(w, mag)
plt.xlabel('Analog frequency in rad/sec')
plt.ylabel('Magnitude')
plt.title('Magnitude Response')
plt.grid(True)

plt.subplot(2, 2, 2)
plt.plot(w, db)
plt.xlabel('Analog frequency in rad/sec')
plt.ylabel('Decibels')
plt.title('Log-Magnitude Response')
plt.grid(True)

plt.subplot(2, 2, 3)
plt.plot(w, pha/np.pi)
plt.xlabel('Analog frequency in rad/sec')
plt.ylabel('Phase in pi units')
plt.title('Phase Response')
plt.grid(True)

plt.subplot(2, 2, 4)
plt.plot(t, y)
plt.xlabel('t (sec)')
plt.ylabel('y (t)')
plt.title('Impluse Response')
plt.grid(True)

plt.tight_layout()
plt.show()
```

```
*** Butterworth Filter Order = 15.0
```




```
In [ ]:
```