In [137]:
```
import numpy as np
import scipy
import scipy.signal
import matplotlib.pyplot as plt
```

Problem 1

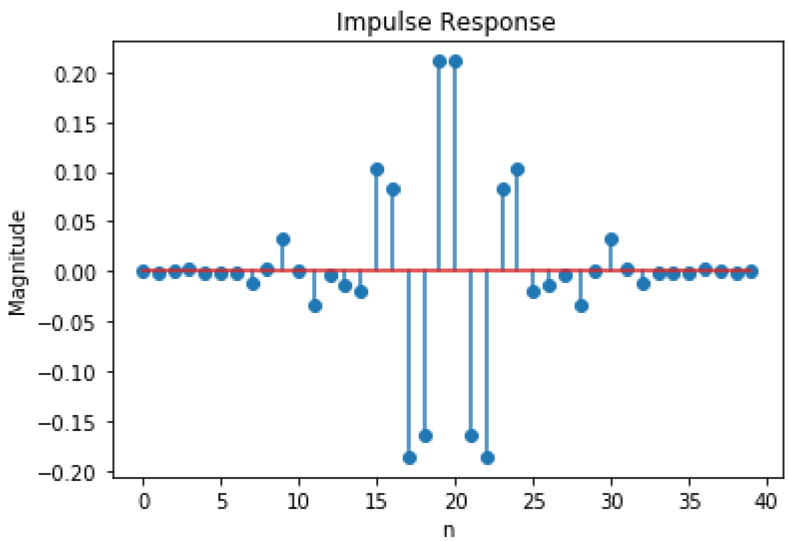
In [138]:
```
ws1 = 0.3 * np.pi
ws2 = 0.7 * np.pi
wp1 = 0.4 * np.pi
wp2 = 0.6 * np.pi
As = 40
Rp = 0.5
```

In [139]:
```
M = 40
T1 = 0.405
transition = np.array([T1])

alpha = (M - 1) / 2
wc = (ws1 + wp1) / 2
length_of_zeros_ws1 = 40 * (ws1 / (2 * np.pi)) + 1
length_of_zeros_after = length_of_zeros_ws1 - 1
length_of_pass = 40 * ((wp2 - wp1)/ (2 * np.pi)) + 1
```

In [140]:
```
first = np.zeros(6)
second = np.ones(5)
third = np.zeros(7)
fourth = np.zeros(7)
fifth = np.ones(5)
sixth = np.zeros(6)
hr = np.concatenate((first, transition, second, transition, third, fourth, tra
nsition, fifth, transition, sixth))
```

In [141]:
```
k1 = np.arange(0, 20)
k2 = np.arange(20, 40)

hk1 = -1 * alpha * (2 * np.pi * k1) / M
hk2 = alpha * 2 * (np.pi / M) * (M - k2)
hk = np.concatenate((hk1, hk2))
hk = np.exp (1j * hk)
H = hr * hk
impulse = np.fft.ifft(H)
plt.stem(impulse)
plt.title('Impulse Response')
plt.xlabel('n')
plt.ylabel('Magnitude')
```

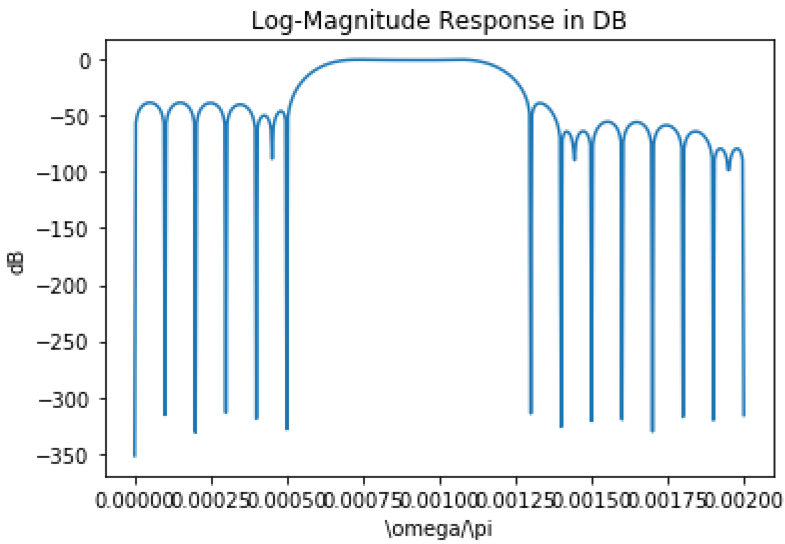
Out[141]:
```
Text(0, 0.5, 'Magnitude')
```



In [147]:
```
def freqz_m(b, a):
    # Modified version of scipy.signal.freqz()
    # db    = Relative magnitude in dB computed over 0 to pi radians
    # mag = absolute magnitude computed over 0 to pi radians
    # pha   = Phase response in radians over 0 to pi radians
    # W     = 501 frequency samples between 0 to pi radians
    # H     = The frequency response, as complex numbers.
    # b     = numerator polynomial of H(z) (for FIR: b=h)
    # a     = denominator polynomial of H(z) (for FIR: a=[1])

    W, H = scipy.signal.freqz(b, a, 1000, 'whole')
    # print(H)
    W = W[0:501]
    H = H[0:501]
    mag = np.abs(H)
    db = 20 * np.log10((mag) / np.max(mag))
    pha = np.angle(H)
    return db, mag, pha
```

In [148]:
```
db, mag, pha = freqz_m(impulse, 1)
delta_w = np.pi / 500
w = np.linspace(0, delta_w, 501)
plt.plot(w / np.pi, db)
plt.title("Log-Magnitude Response in DB")
plt.ylabel('dB')
plt.xlabel('\omega/\pi')
```

Out[148]: Text(0.5, 0, '\\omega/\\pi')



Problem 3

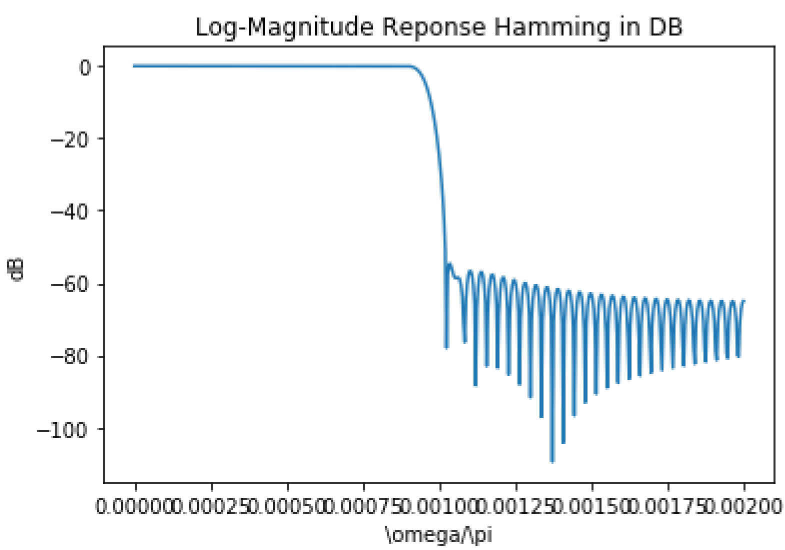
In [149]:
```
T1 = 0.107
T2 = 0.58895
wp = 0.45 * np.pi
ws = 0.51 * np.pi
Rp = 1
As = 50
wc = (wp + ws) / 2
M = ((6.6 * np.pi) / (ws - wp)) + 1
M = int(np.round(M))
```

In [150]:
```
def ideal_lp(wc, M):
    # Ideal LowPass filter computation
    # hd = ideal impulse response between 0 to M-1
    # wc = cutoff frequency in radians
    # M = length of the ideal filter

    # You need to fill in here
    n = np.arange(0, M)

    alpha = int((M - 1) / 2)

    m = n - alpha + np.spacing(1)

    hd = np.sin(wc * m) / (np.pi * m)
    return hd
```

In [151]:
```
low_pass = ideal_lp(wc, M)
```

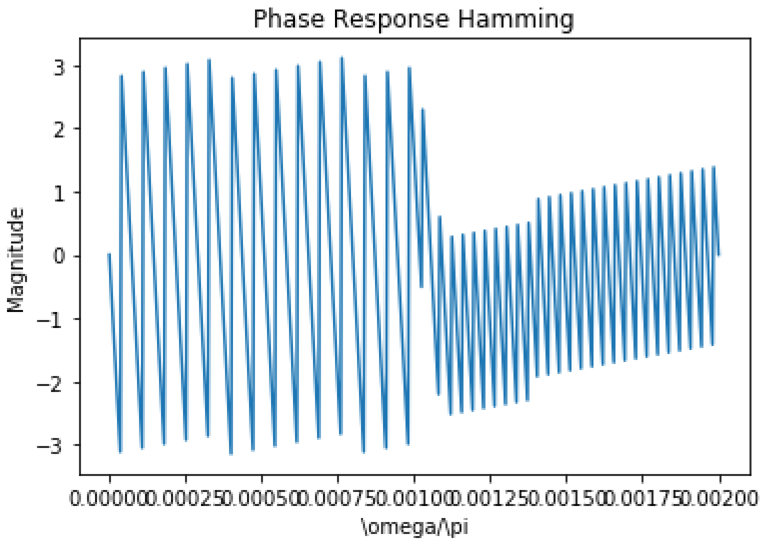
In [152]:
```
#part1
ham = np.hamming(M)
h_windowing = ham * low_pass
db, mag, pha = freqz_m(h_windowing, 1)
delta_w = np.pi / 500
w = np.linspace(0, delta_w, 501)
plt.plot(w / np.pi, db)
plt.title('Log-Magnitude Reponse Hamming in DB')
plt.ylabel('dB')
plt.xlabel('\omega/\pi')
```

Out[152]:
```
Text(0.5, 0, '\\omega/\\pi')
```



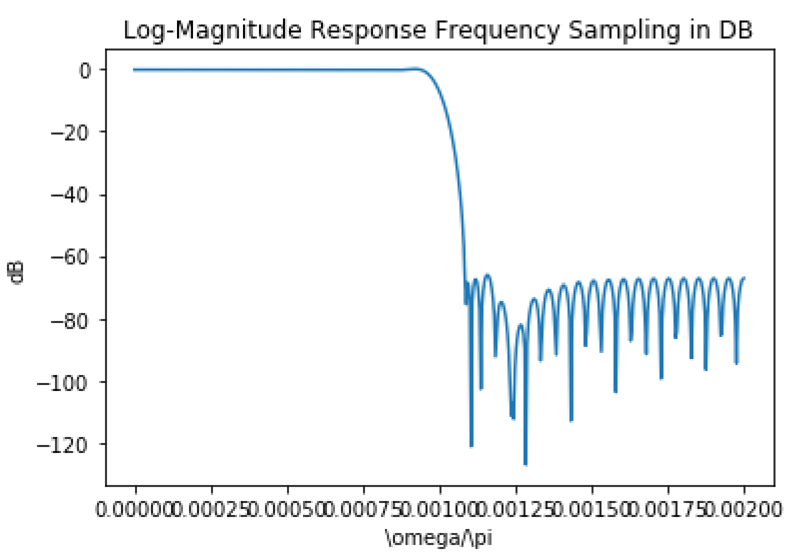
In [153]:
```
plt.title('Phase Response Hamming')
delta_w = np.pi / 500
w = np.linspace(0, delta_w, 501)
plt.plot(w / np.pi, pha)
plt.ylabel('Magnitude')
plt.xlabel('\omega/\pi')
```

Out[153]:
```
Text(0.5, 0, '\\omega/\\pi')
```



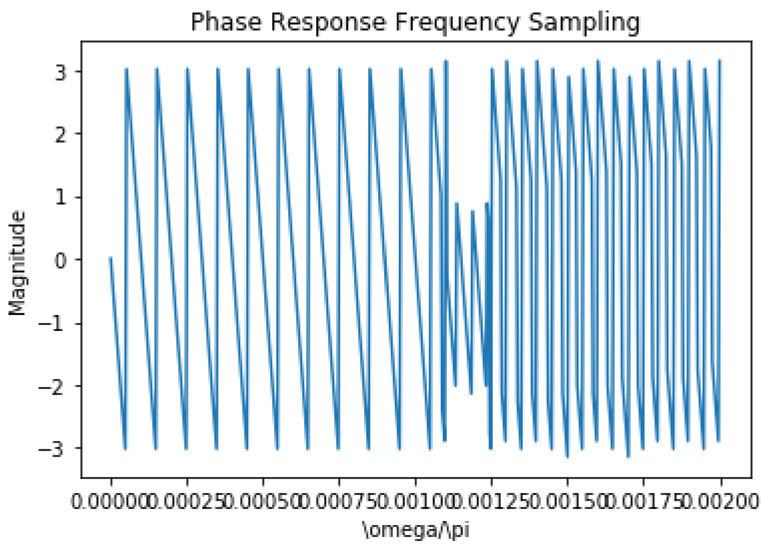
In [154]:
```
#part 2
T1 = 0.107
T2 = 0.58895
wp = 0.45 * np.pi
ws = 0.51 * np.pi
M = 81

alpha = (M - 1) / 2
pass_amount = int(np.ceil(M * (wp / (2 * np.pi))) + 1)
passband1 = np.ones(pass_amount)
passband2 = np.ones(19)

zeroes = np.zeros(38)

hr = np.concatenate((passband1, np.array([T2]), np.array([T1]), zeroes, np.arr
ay([T1]), np.array([T2]), passband2))

k1 = np.arange(0, 40)
k2 = np.arange(40, 81)

#type 1
hk1 = -1 * alpha * (2 * np.pi * k1) / M
hk2 = alpha * 2 * (np.pi / M) * (M - k2)
hk = np.concatenate((hk1, hk2))
hk = np.exp (1j * hk)
H = hr * hk
h = np.real(np.fft.ifft(H, M))

delta_w = np.pi / 500
w = np.linspace(0, delta_w, 501)
db2, mag2, pha2 = freqz_m(h, 1)
plt.plot(w / np.pi, db2)
plt.title("Log-Magnitude Response Frequency Sampling in DB")
plt.ylabel('dB')
plt.xlabel('\omega/\pi')
```

Out[154]:
```
Text(0.5, 0, '\\omega/\\pi')
```



In [155]:
```
plt.title("Phase Response Frequency Sampling")
delta_w = np.pi / 500
w = np.linspace(0, delta_w, 501)
plt.plot(w / np.pi, pha2)
plt.ylabel('Magnitude')
plt.xlabel('\omega/\pi')
```

Out[155]: Text(0.5, 0, '\\omega/\\pi')



Part 3 The hamming window has a more uniform ripple. The M for the Hamming technique is longer than the M for the frequency sampling technique. M (Hamming) was 111 and M (frequency sampling) was 81. You get more samples with the Hamming technique which is good because one problem with Frequency Sampling that occurs is if you don't have a lot of samples, then your samples may not reflect the reponse correctly.