This example implements obtaining the frequency response of a discrete time system given in its Z-transform, and applying an input signal to it to calculate the output of the system.

In [ ]:
```python
# import the necessary libraries
import numpy as np                # for using basic array functions
import matplotlib.pyplot as plt # for this example, it may not be necessary

# the main package for signal processing is called "scipy" and we will use "signal"
import scipy.signal as sgnl
# alternative syntax: from scipy import signal as sgnl
%matplotlib notebook
```

In this example, we will use $freqz()$ (and $freqz\_zpk()$) function of the *scipy* library to obtain the frequency response of a signal/system from its Z-transform. There are two different syntaxes for it: (i) the transfer function form and (ii) the pole-zero-gain (zpk) form.

We are given a system in the Z-domain as:

$$H(z) = \frac{1+z^{-1}}{(1-j\frac{1}{2}z^{-1})(1+j\frac{1}{2}z^{-1})(1+\frac{1}{4}z^{-1})}$$

Note the complex poles exist with their conjugates!

Define the this system with its poles and zeros:

In [ ]:
```python
zeros = np.array([-1])          # observe that the numerator can be defined as transf
poles = np.array([1j/2, -1j/2, 1.0/4])

w, H = sgnl.freqz_zpk(zeros, poles, 1)

plt.plot(w/np.pi, abs(H))        # plot the magnitude in logarithmic scale with blue
plt.title('Frequency Response of $H(z)$')
plt.ylabel('Magnitude'), plt.xlabel('$\omega$ x $\pi$  (rad/sample)')
plt.grid()
```

Now, we will convert the system into the transfer function form and repeat the same steps.

In [ ]:
```python
num, denum = sgnl.zpk2tf(zeros, poles, 1)  # will return the coefficients b and a, r

w1, H_tf = sgnl.freqz(num, denum)

plt.figure()
plt.plot(w1/np.pi, abs(H_tf))       # plot the magnitude in logarithmic scale with b
plt.title('Frequency Response of $H(z)$')
plt.ylabel('Magnitude'), plt.xlabel('$\omega$ x $\pi$  (rad/sample)')
plt.grid()
```

The two figures appear the same and this result confirms that the two functions peform the same.

Our input is given as

$$x[n] = 2cos(0.2\pi n) + sin(0.9\pi n)$$

Now, we will define the input signal, and apply it to the given system and interpret the system behavior by looking at the input and output.

In [ ]:
```python
n = np.arange(0, 40, 1)          # define the index vector
xn = 2*np.cos(0.2*np.pi*n) + np.sin(0.9*np.pi*n)          # define the input signal
```

```
yn = sgnl.lfilter(num, denum, xn)
plt.figure()
plt.stem(n, yn)
```

You have calculated the response of the system in the first part for the given input signal analytically. Now, define the output you have calculated and plot for 40 points, i.e. $n = 0...39$, with *stem()* function. Compare these two plots, are they the same? Should they be the same?

Now, try to interpret the system behavior. Do you think that you have obtained (approximately) the same results as the first part? Look at the output signal you calculated in the first part of the problem. Try to observe each term of the output signal in the plot.