Systems & Signals 414

Practical 4

D.S. van der Westhuizen - 17158680

In [1]:

```
#All the necessary imports
%matplotlib inline
import pylab as pl
pl.style.use('ggplot') #pretty plots
import numpy as np
from scipy import signal

pl.rcParams['figure.figsize'] = (12,4)

def prepare_plot(title, y_label, x_label):
    pl.figure()
    pl.title(title)
    pl.ylabel(y_label)
    pl.xlabel(x_label)

def twin_plot(xy_data1, xy_data2, title='', xlabel='', ylabel1='', ylabel2=''):
    pl.title(title)
    pl.plot(*xy_data1, color='b')
    pl.ylabel(ylabel1, color='b')
    pl.xlabel(xlabel)

    ax2 = pl.gca().twinx()
    pl.plot(*xy_data2, color='g')
    pl.ylabel(ylabel2, color='g')
```

In [2]:

```
#Adapted from https://gist.github.com/endolith/4625838
def zplane(zeros, poles):
    """
    Plot the complex z-plane given zeros and poles.
    """
    zeros=np.array(zeros);
    poles=np.array(poles);
    ax = pl.gca()

    # Add unit circle and zero axes
    unit_circle = pl.matplotlib.patches.Circle((0,0), radius=1, fill=False,
                                color='black', ls='solid', alpha=0.6)
    ax.add_patch(unit_circle)
    pl.axvline(0, color='0.7')
    pl.axhline(0, color='0.7')

    #Rescale to a nice size
    rscale = 1.2 * np.amax(np.concatenate((abs(zeros), abs(poles), [1])))
    pl.axis('scaled')
    pl.axis([-rscale, rscale, -rscale, rscale])

    # Plot the poles and zeros
    polesplot = pl.plot(poles.real, poles.imag, 'x', markersize=9)
    zerosplot = pl.plot(zeros.real, zeros.imag,  'o', markersize=9, color='none',
                            markeredgecolor=polesplot[0].get_color(),
                            )

    #Draw overlap text
    overlap_txt = []
    def draw_overlap_text():
        for txt in overlap_txt:
            try:    txt.remove()
            except: txt.set_visible(False)
        del overlap_txt[:]

        poles_pixel_positions = ax.transData.transform(np.vstack(polesplot[0].get_data
()).T)
        zeros_pixel_positions = ax.transData.transform(np.vstack(zerosplot[0].get_data
()).T)

        for (zps_pixels, zps) in [(poles_pixel_positions, poles), (zeros_pixel_position
s, zeros)]:
            superscript = np.ones(len(zps))
            for i in range(len(zps)):
                for j in range(i+1,len(zps)):
                    if superscript[i]!=-1:
                        if np.all(np.abs(zps_pixels[i] - zps_pixels[j]) < 0.9):
                            superscript[i]+=1;
                            superscript[j]=-1;
            for i in range(len(zps)):
                if superscript[i] > 1:
                    txt = pl.text(zps[i].real, zps[i].imag,
                            r'${}^{%d}$'%superscript[i], fontsize=20
                            )
                    overlap_txt.append(txt)
    draw_overlap_text()

    #Reset when zooming
    def on_zoom_change(axes): draw_overlap_text()
```

```
    ax.callbacks.connect('xlim_changed', on_zoom_change)
    ax.callbacks.connect('ylim_changed', on_zoom_change)
```

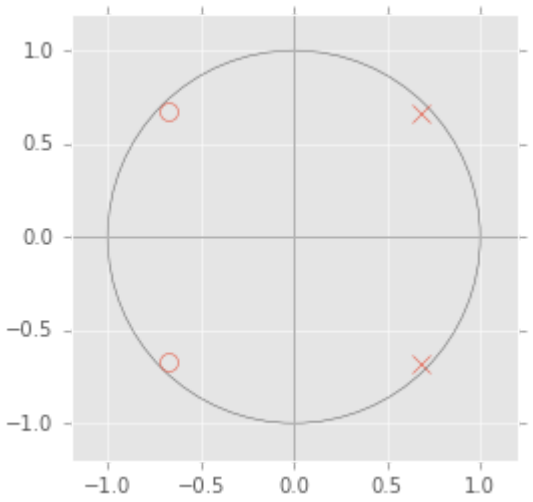
1.1.

In [3]:

```
r1=0.95
r2=0.95
theta1=2*3*np.pi/8
theta2=2*1*np.pi/8
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
```

In [4]:

```
%matplotlib inline
foo = zplane(zeros,poles)
```



1.2.

In [5]:

```
r1=0.95
r2=0.95
theta1=2*3*np.pi/8
theta2=2*1*np.pi/8
```

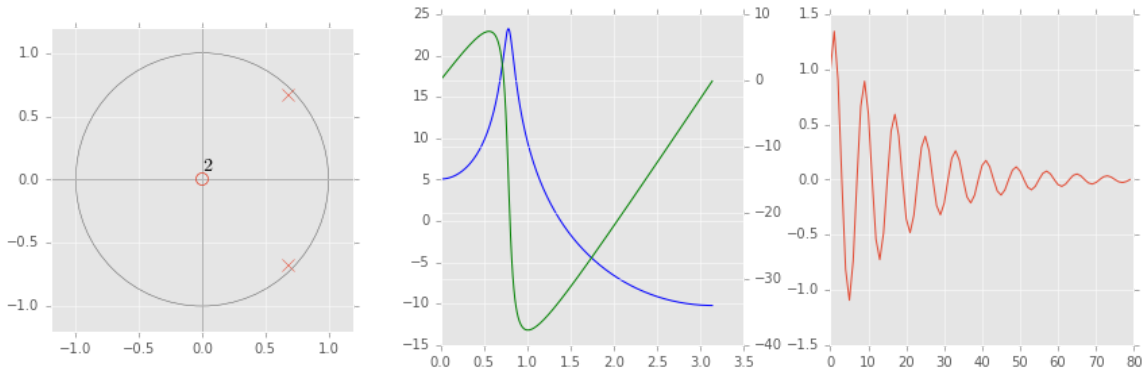
In [6]:

```
%matplotlib inline
r1=0
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[6]:

```
[<matplotlib.lines.Line2D at 0x790eba8>]
```



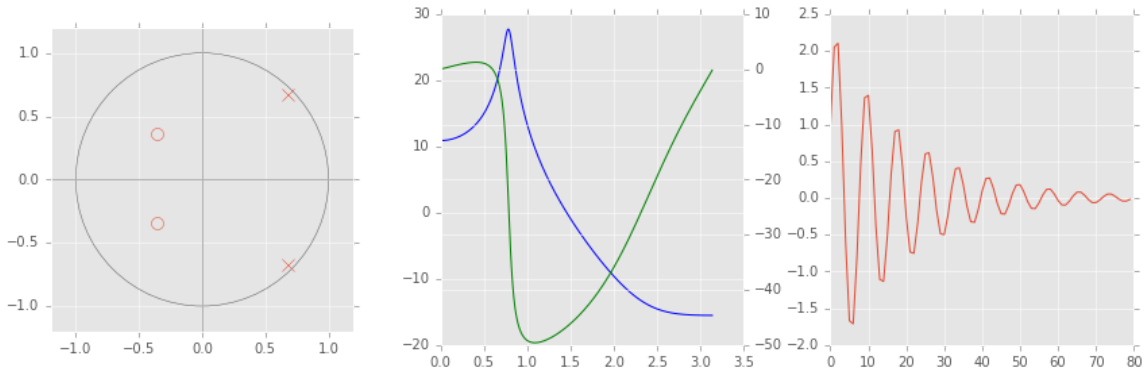
In [7]:

```
%matplotlib inline
r1=0.5
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[7]:

```
[<matplotlib.lines.Line2D at 0x81d8550>]
```



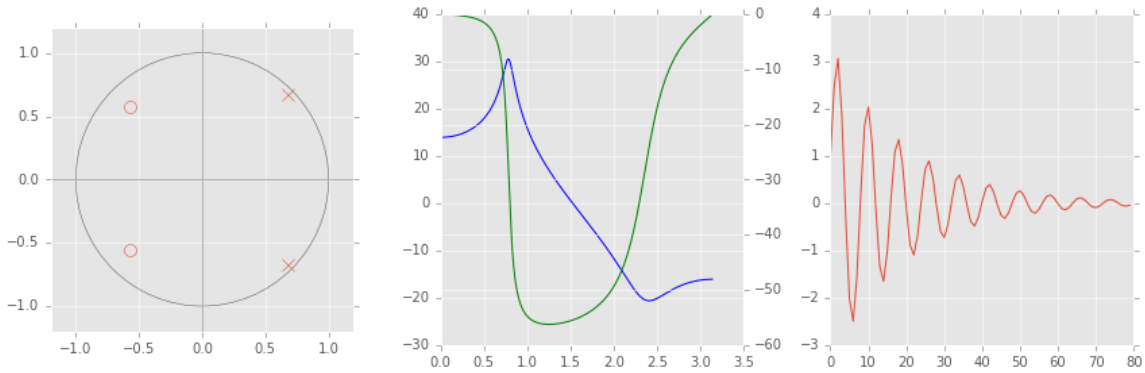
In [8]:

```
%matplotlib inline
r1=0.8
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x830c8d0>]
```



In [9]:

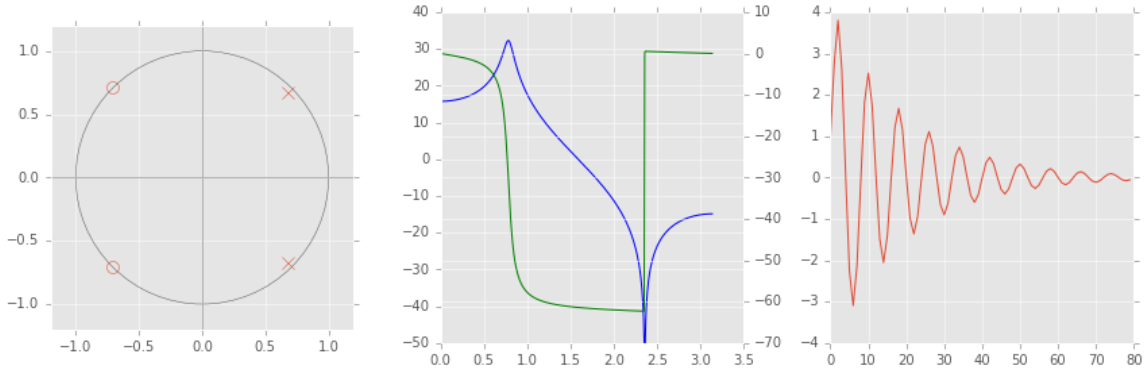
```
%matplotlib inline
r1=1
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

```
F:\Program Files\WinPython-64bit-3.4.3.7\python-3.4.3.amd64\lib\site-packa
ges\ipykernel\__main__.py:14: RuntimeWarning: divide by zero encountered i
n log10
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x96b4d68>]
```



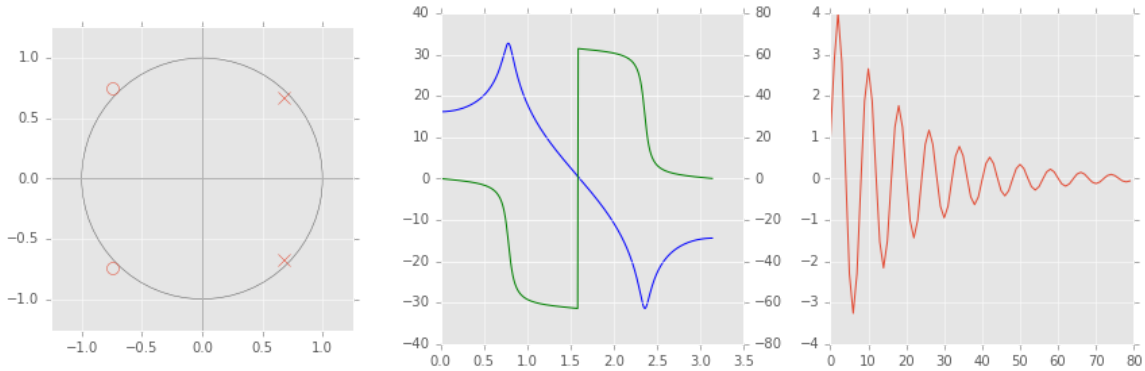
In [10]:

```
%matplotlib inline
r1=1.05
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[10]:

```
[<matplotlib.lines.Line2D at 0x962c2b0>]
```



1.3.

In [11]:

```
r1=0.95
r2=0.95
theta1=2*3*np.pi/8
theta2=2*1*np.pi/8
```

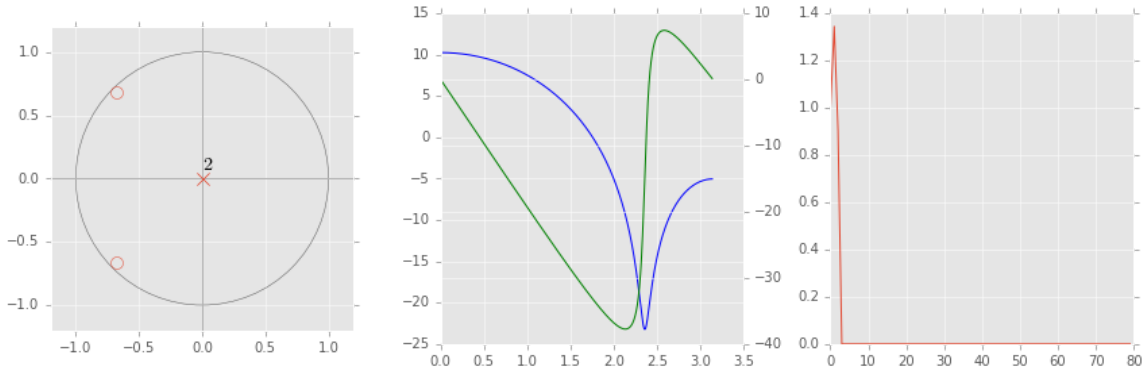
In [12]:

```
%matplotlib inline
r2=0
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[12]:

```
[<matplotlib.lines.Line2D at 0x95d32b0>]
```



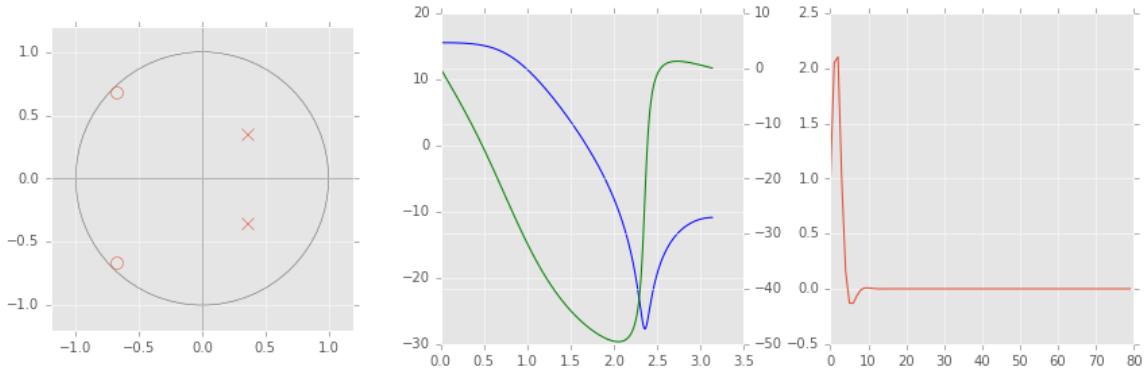
In [13]:

```
%matplotlib inline
r2=0.5
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[13]:

```
[<matplotlib.lines.Line2D at 0x99e6ef0>]
```



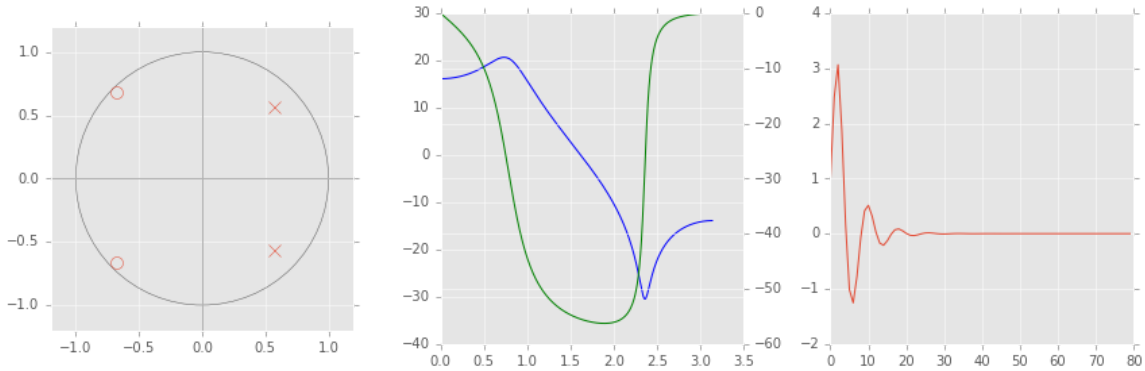
In [14]:

```
%matplotlib inline
r2=0.8
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[14]:

```
[<matplotlib.lines.Line2D at 0x9bc24a8>]
```



^The increasing radius of the pole causes the response oscillation to increase

In [15]:

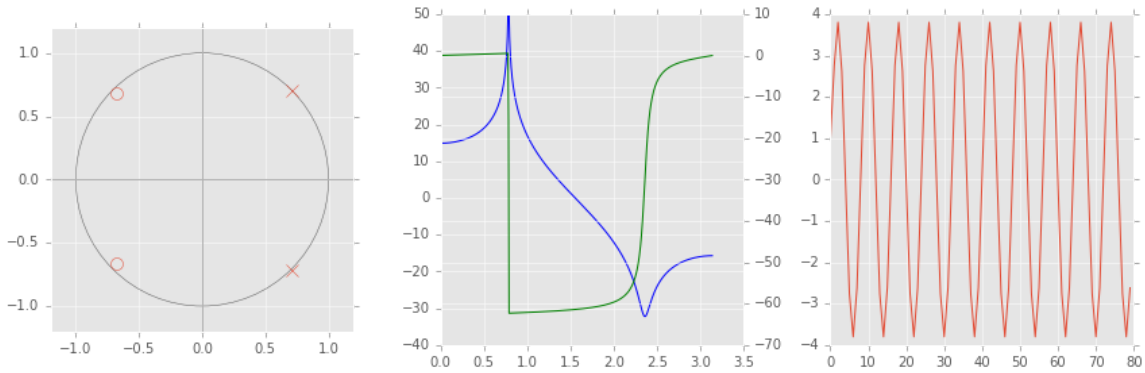
```
%matplotlib inline
r2=1
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

```
F:\Program Files\WinPython-64bit-3.4.3.7\python-3.4.3.amd64\lib\site-packa
ges\scipy\signal\filter_design.py:244: RuntimeWarning: divide by zero enco
untered in true_divide
  h = polyval(b[::-1], zm1) / polyval(a[::-1], zm1)
```

Out[15]:

```
[<matplotlib.lines.Line2D at 0x9dbb9b0>]
```



^The pole pair has now reached the unity circle and the response is thus on the border between stable and unstable, and is thus maintaining an unchanging oscillation.

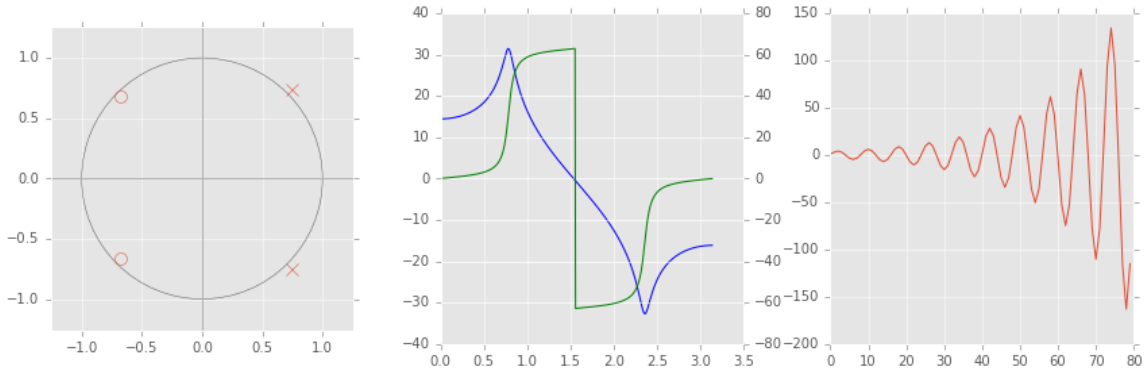
In [16]:

```
%matplotlib inline
r2=1.05
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[16]:

```
[<matplotlib.lines.Line2D at 0xaea1908>]
```



^The pole pair has now moved outside the unity circle and thus the impulse response is unstable (continously increasing)

1.4.

In [17]:

```
r1=0.95
r2=0.95
theta1=2*3*np.pi/8
theta2=2*1*np.pi/8
```

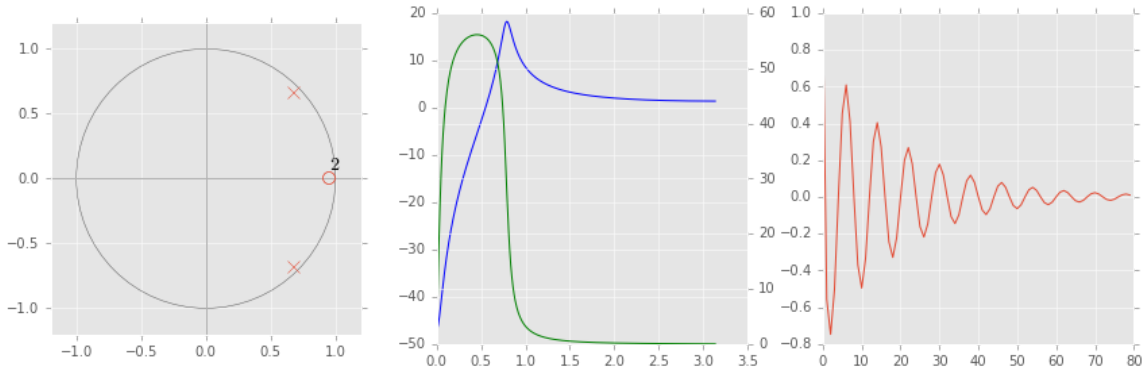
In [18]:

```
%matplotlib inline
theta1=0
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[18]:

```
[<matplotlib.lines.Line2D at 0xb1026a0>]
```



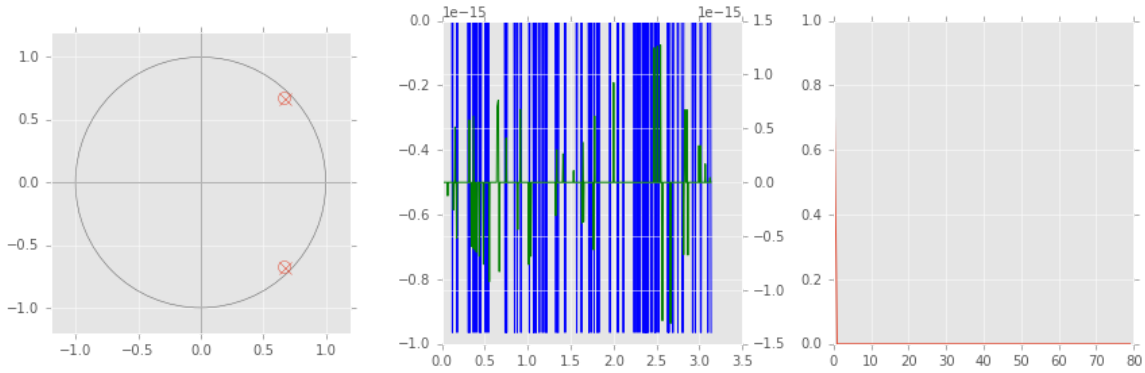
In [19]:

```
%matplotlib inline
theta1=2*np.pi/8
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[19]:

```
[<matplotlib.lines.Line2D at 0xb213b00>]
```



^pole and zero canceling each other out, making the frequency response zero.

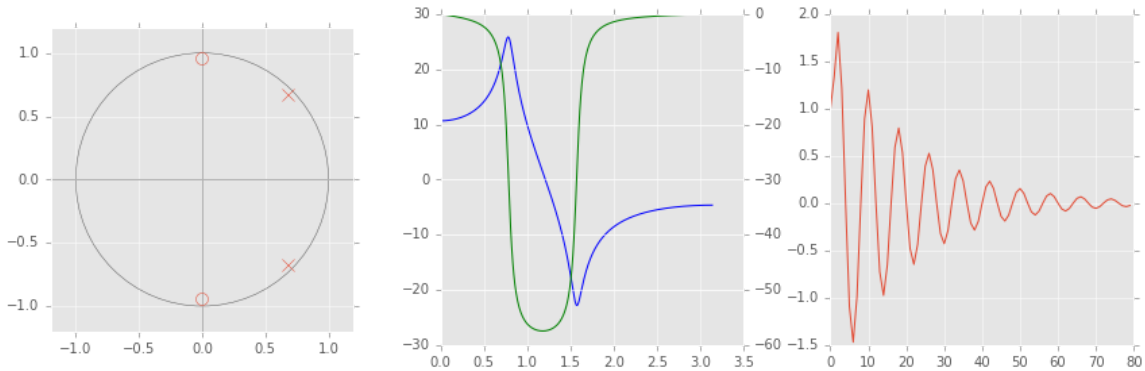
In [20]:

```
%matplotlib inline
theta1=4*np.pi/8
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[20]:

```
[<matplotlib.lines.Line2D at 0xc6bc198>]
```



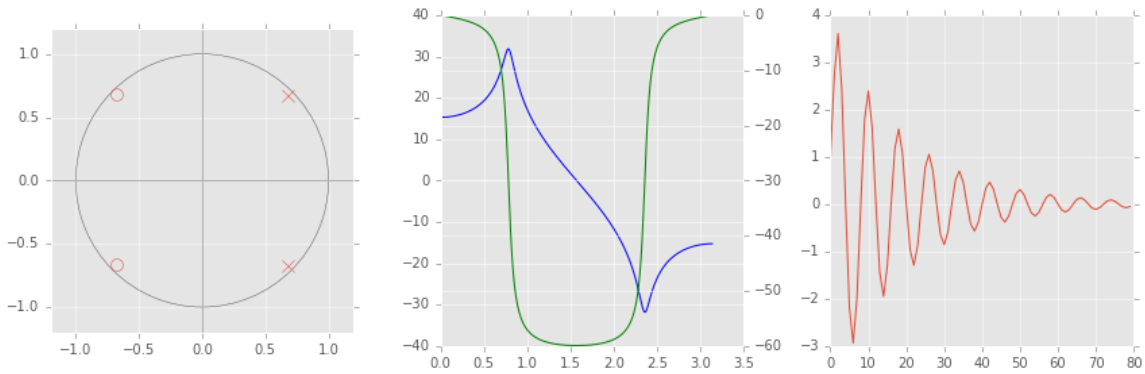
In [21]:

```
%matplotlib inline
theta1=6*np.pi/8
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[21]:

```
[<matplotlib.lines.Line2D at 0xc614e48>]
```



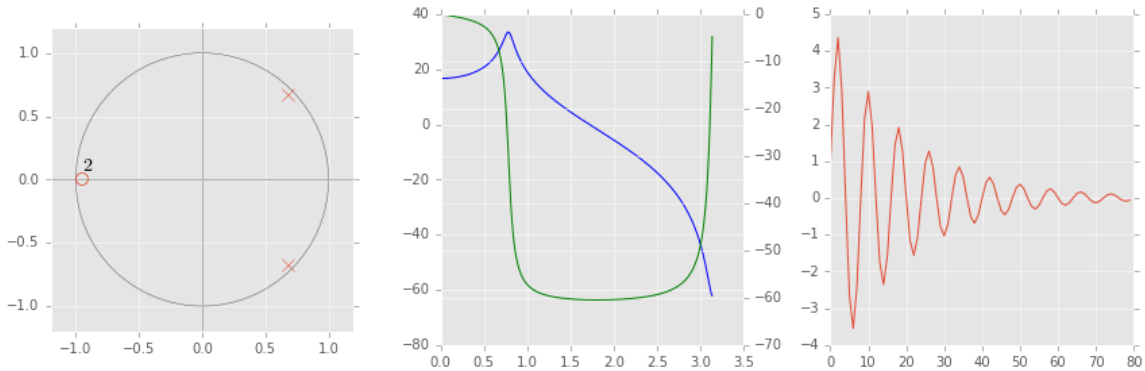
In [22]:

```
%matplotlib inline
theta1=np.pi
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[22]:

```
[<matplotlib.lines.Line2D at 0xc851f28>]
```



1.5.

In [23]:

```
r1=0.95
r2=0.95
theta1=2*3*np.pi/8
theta2=2*1*np.pi/8
```

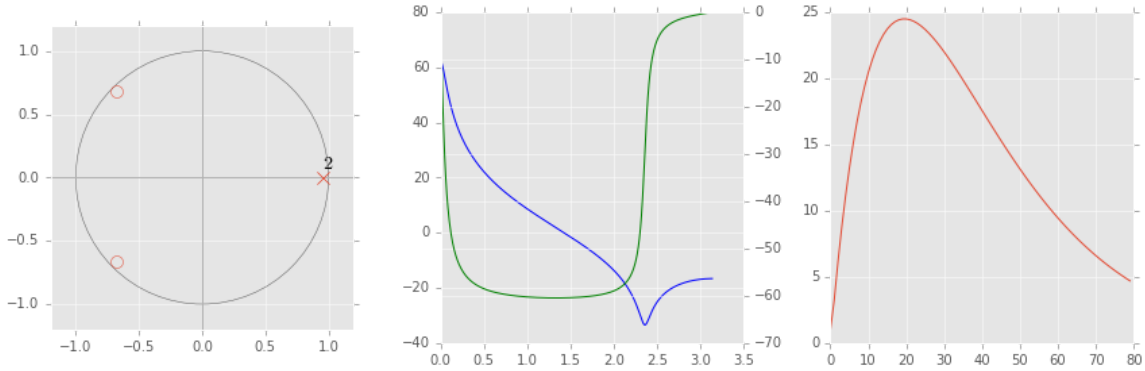
In [24]:

```
%matplotlib inline
theta2=0
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[24]:

```
[<matplotlib.lines.Line2D at 0xc9c9f98>]
```



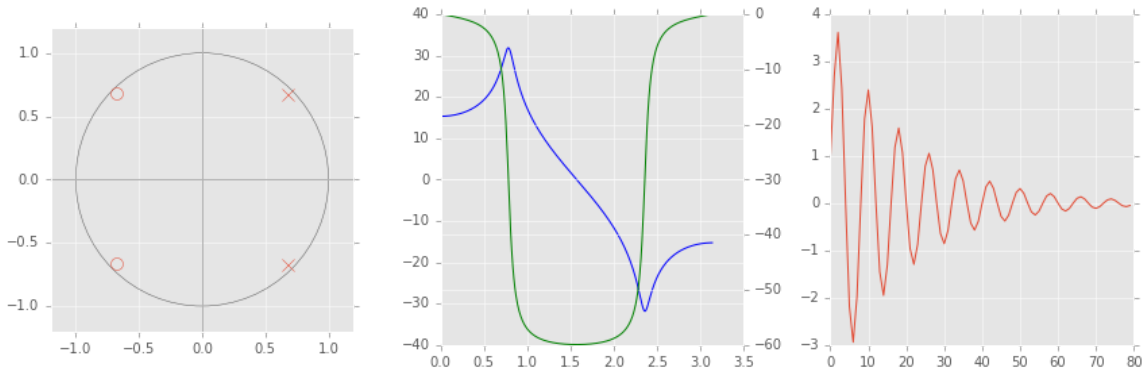
In [25]:

```
%matplotlib inline
theta2=2*np.pi/8
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[25]:

```
[<matplotlib.lines.Line2D at 0xcb21cf8>]
```



^The pole pair, having become complex, now causes the response to oscillate

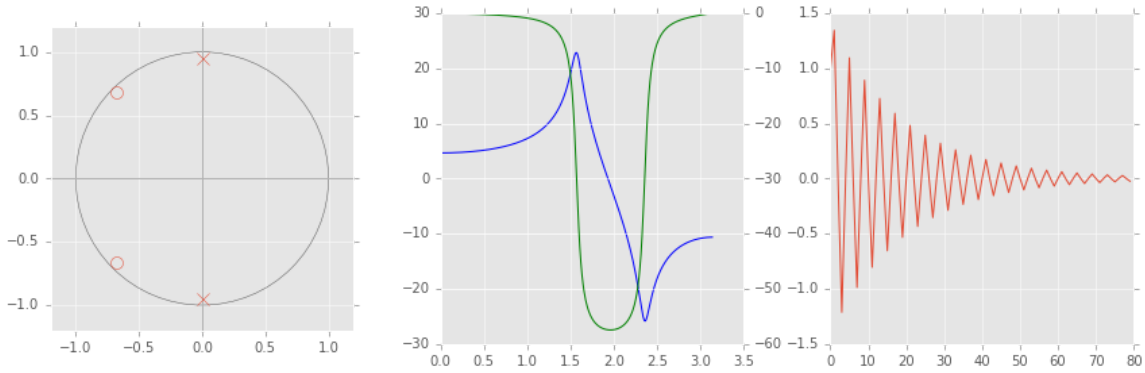
In [26]:

```
%matplotlib inline
theta2=4*np.pi/8
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[26]:

```
[<matplotlib.lines.Line2D at 0xccfaac8>]
```



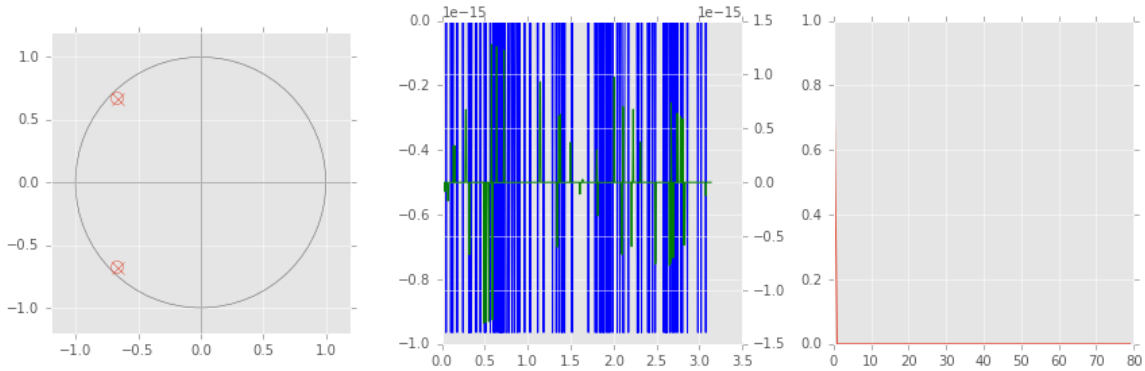
In [27]:

```
%matplotlib inline
theta2=6*np.pi/8
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[27]:

```
[<matplotlib.lines.Line2D at 0xde5fcf8>]
```



^pole and zero canceling each other out, making the frequency response zero.

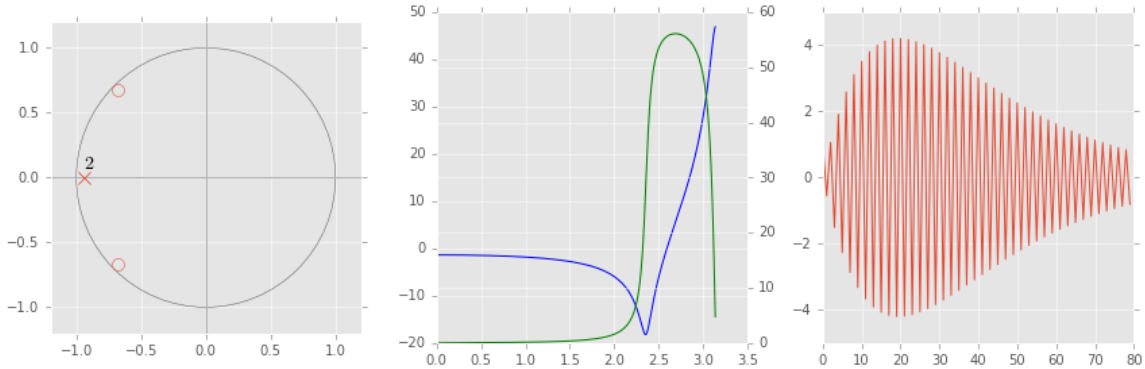
In [28]:

```
%matplotlib inline
theta2=np.pi
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])

pl.figure(figsize=(12,4))
pl.subplot(1, 3, 1)
foo = zplane(zeros,poles)
pl.subplot(1, 3, 2)
pl.tight_layout() #Reduce label overflow
foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
pl.subplot(1, 3, 3)
pl.tight_layout() #Reduce label overflow
t = np.array(range(0,80))
x = np.zeros(80)
x[0] = 1
foo = signal.lfilter(b,a,x)
pl.plot(t,foo)
```

Out[28]:

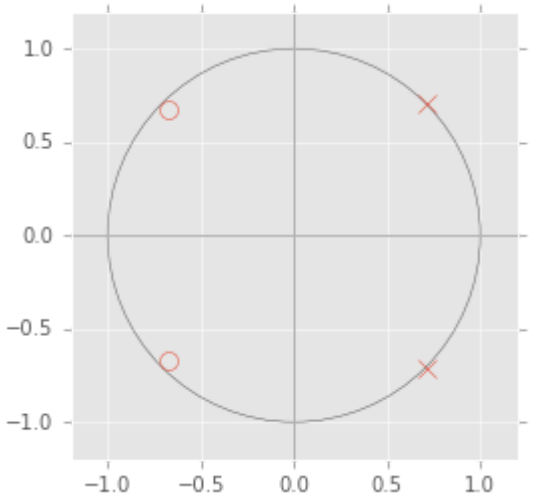
```
[<matplotlib.lines.Line2D at 0xe12bc88>]
```



1.6

In [29]:

```
r1=0.95
r2=1
theta1=2*3*np.pi/8
theta2=2*1*np.pi/8
zeros = np.array([r1*np.cos(theta1) + r1*np.sin(theta1)*1j,r1*np.cos(theta1) - r1*np.si
n(theta1)*1j])
poles = np.array([r2*np.cos(theta2) + r2*np.sin(theta2)*1j,r2*np.cos(theta2) - r2*np.si
n(theta2)*1j])
b=np.array([1,-r1*2*np.cos(theta1),r1**2])
a=np.array([1,-r2*2*np.cos(theta2),r2**2])
foo = zplane(zeros,poles)
```

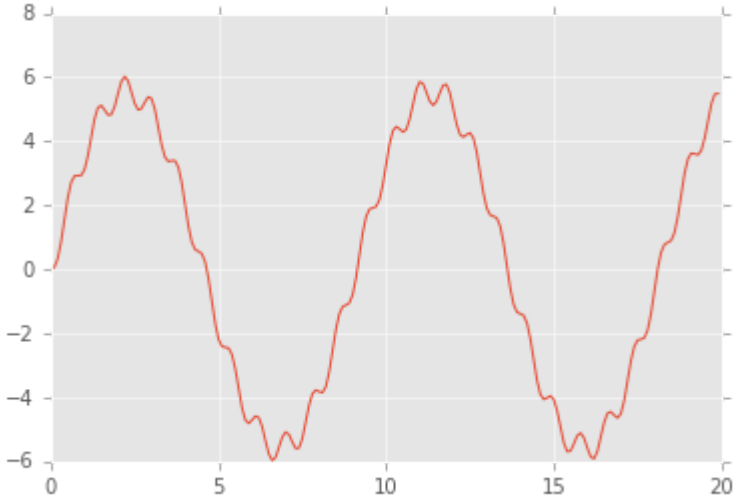


In [30]:

```
t = np.array(range(0,200))
x = np.sin(0.11*2*np.pi*(t/10))
foo = signal.lfilter(b,a,x)
pl.plot(t/10,foo)
```

Out[30]:

```
[<matplotlib.lines.Line2D at 0xe9f1a90>]
```

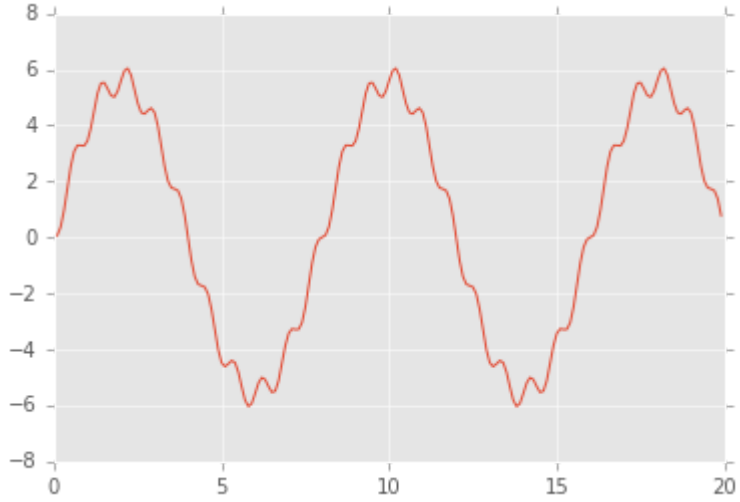


In [31]:

```
t = np.array(range(0,200))
x = np.sin(0.125*2*np.pi*(t/10))
foo = signal.lfilter(b,a,x)
pl.plot(t/10,foo)
```

Out[31]:

[<matplotlib.lines.Line2D at 0xea55ac8>]

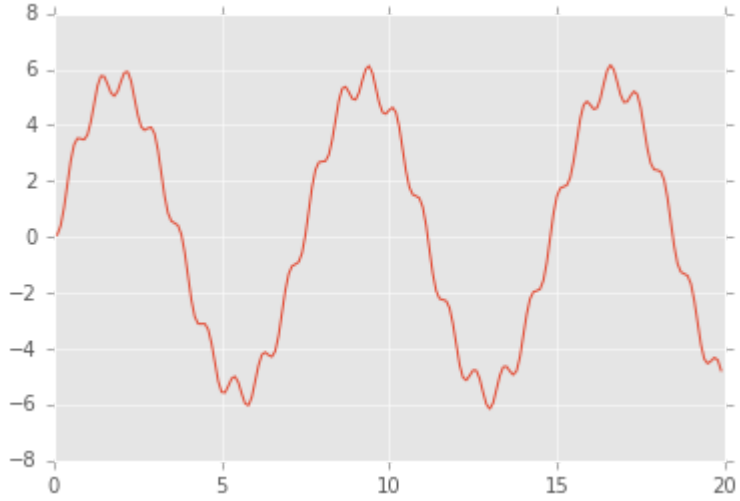


In [32]:

```
t = np.array(range(0,200))
x = np.sin(0.135*2*np.pi*(t/10))
foo = signal.lfilter(b,a,x)
pl.plot(t/10,foo)
```

Out[32]:

[<matplotlib.lines.Line2D at 0xeab88d0>]

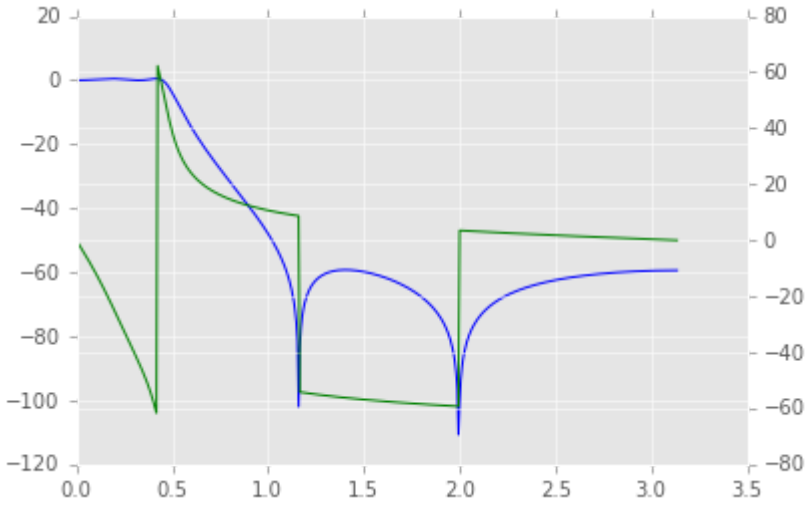


2.1.

In [33]:

```
%matplotlib inline
b=np.array([0.0038,0.0001,0.0051,0.0001,0.0038])
a=np.array([1,-3.2821,4.236,-2.5275,0.5865])

foo = signal.freqz(b,a)
twin_plot((foo[0],20*np.log10(np.abs(foo[1]))),(foo[0],20*np.angle(foo[1])))
```



In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```