Systems & Signals 414

Practical 5

D.S. van der Westhuizen - 17158680

In [1]:

```
import scipy.fftpack
import copy
```

In [2]:

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

pl.rcParams['figure.figsize'] = (9,2)

def setup_plot(title, y_label, x_label):
    pl.box(False)
    pl.margins(*(pl.array(pl.margins())+0.05))
    pl.title(title)
    pl.ylabel(y_label)
    pl.xlabel(x_label)
```

In [3]:

```
#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)
```

In [4]:

```
import os
import urllib
import scipy.io
from scipy.io import wavfile

#Download yesterday.wav from courses.ee.sun.ac.za and return it as a numpy array
def yesterday_wav():
    url = 'http://courses.ee.sun.ac.za/Stelsels_en_Seine_414/content/yesterday.wav'
    filename = os.path.split(url)[-1]
    #Download if path does not already exist
    if not os.path.isfile(filename):
        urllib.request.urlretrieve(url, filename)
    sample_frequency, signal_array = wavfile.read(filename)
    #Normalise signal and return
    signal_array = signal_array/np.max([np.max(signal_array), -np.min(signal_array)])
    return sample_frequency, signal_array
```

Question 1.1

In [5]:

```
beatles_sample_freq, beatles_sample = yesterday_wav()
beatles_sample.shape
x1 = beatles_sample
t1 = np.linspace(0,x1.shape[0]/44100,x1.shape[0],False)
#pl.plot(t1,beatles_sample)
print(x1.shape[0]/44100)
print(x1.shape)
print(t1[t1.shape[0]-1])
X1 = np.fft.fft(x1)
```

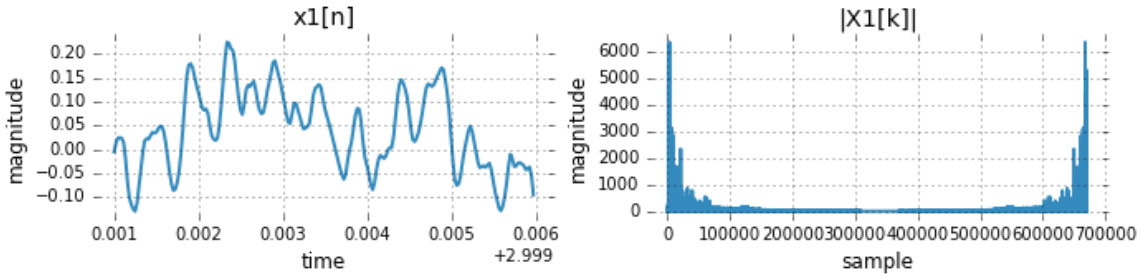
```
15.209070294784581
(670720,)
15.209047619
```

In [6]:

```
#Example of subplotting
pl.figure(figsize=(11,2))
pl.subplot(1, 2, 1)
setup_plot('x1[n]', 'magnitude', 'time')
pl.plot(t1[44100*3:44100*3 + (int)(44100*0.005)],x1[44100*3:44100*3 + (int)(44100*0.005
)])

pl.subplot(1, 2, 2)
setup_plot('|X1[k]|', 'magnitude', 'sample')
pl.plot(np.abs(X1))
```

Out[6]:

```
[<matplotlib.lines.Line2D at 0x82f39e8>]
```



In [7]:

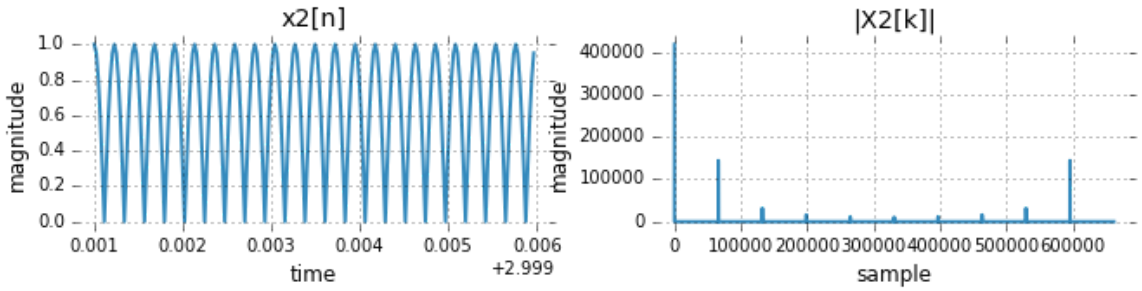
```
fs = 44100
f0 = 2205
s = 0
e = 15
t2 = np.linspace(s,e,(e-s)*fs,False)
x0 = np.cos(f0*2*np.pi*t2)
x2 = np.abs(x0)
X2 = np.fft.fft(x2)
```

In [8]:

```
pl.figure(figsize=(11,2))
pl.subplot(1, 2, 1)
setup_plot('x2[n]', 'magnitude', 'time')
pl.plot(t2[44100*3:44100*3 + (int)(44100*0.005)],x2[44100*3:44100*3 + (int)(44100*0.005
)])

pl.subplot(1, 2, 2)
setup_plot('|X2[k]|', 'magnitude', 'sample')
pl.plot(np.abs(X2))
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0xfee74e0>]
```



In [9]:

```
print(x2.shape)
print(x1.shape)
x2_ = np.append(x2,np.zeros(x1.shape[0]-x2.shape[0]))
x = x1 + x2_
print(x.shape)
X = np.fft.fft(x)
```

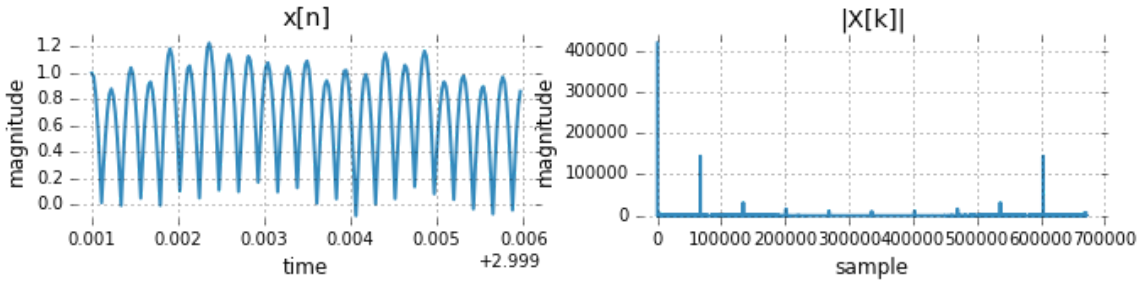
```
(661500,)
(670720,)
(670720,)
```

In [10]:

```
pl.figure(figsize=(11,2))
pl.subplot(1, 2, 1)
setup_plot('x[n]', 'magnitude', 'time')
pl.plot(t1[44100*3:44100*3 + (int)(44100*0.005)],x[44100*3:44100*3 + (int)(44100*0.005
)])

pl.subplot(1, 2, 2)
setup_plot('|X[k]|', 'magnitude', 'sample')
pl.plot(np.abs(X))
```

Out[10]:

```
[<matplotlib.lines.Line2D at 0x1286ba58>]
```



Question 1.2

In [11]:

```
IPython.display.Audio(data=x1,rate=44100)
```

Out[11]:

0:00 / 0:15

In [12]:

```
IPython.display.Audio(data=x2,rate=44100)
```

Out[12]:

0:00 / 0:15

In [13]:

```
IPython.display.Audio(data=x,rate=44100)
```

Out[13]:

0:00 / 0:15

Question 1.3

In [28]:

```
k=10
alpha = 0.99

a = np.array([1])
a = np.append(a,np.zeros([k]))
b = np.array([1])
b = np.append(b,np.zeros([k]))
b = np.append(b,alpha)
print(b)
print(a)
```

```
[ 1.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.99]
[ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

In [29]:

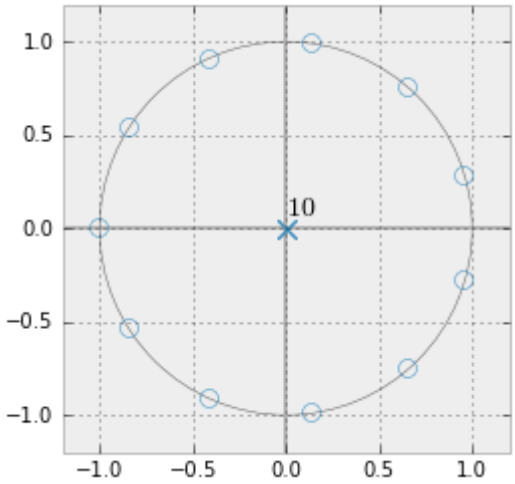
```
z,p,k = signal.tf2zpk(b, a)
print(z)
print(p)
print(k)
```

```
[-0.99908675+0.j         -0.84048526+0.54014708j -0.84048526-0.54014708j
 -0.41503564+0.90880127j -0.41503564-0.90880127j  0.14218487+0.98891749j
  0.14218487-0.98891749j  0.65426268+0.75505939j  0.65426268-0.75505939j
  0.95861672+0.28147526j  0.95861672-0.28147526j]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
1.0
```

This demonstrates what the z-plane will look like if k=10. This plot takes far too long too generate if I use the desired value of k=4410.

In [30]:

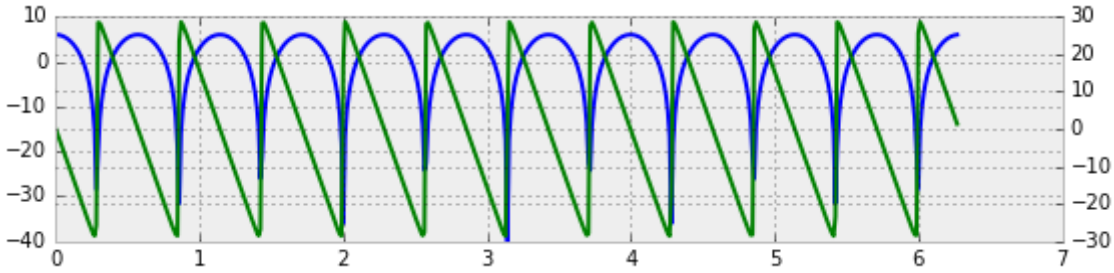
```
pl.figure(figsize=(12,4))
foo = zplane(z,p)
```



Question 4

In [31]:

```
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')

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



The comb filter looks something like this

In [32]:

```
foo[0].shape
```

Out[32]:

```
(512,)
```

Question 5

I will use the transfer function below on x[n] by coding the second equation with k=4410 and α almost 1.

$$H(z) = \frac{1 + \alpha z^{-k}}{1}$$
$$y[n] = x[n] + \alpha x[n - k]$$

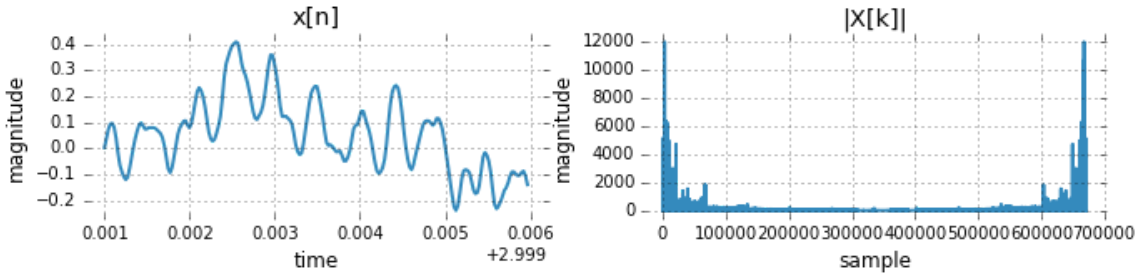
In [33]:

```
pl.figure(figsize=(5.5,2))
y = copy.copy(x)
k = 4410
for q in range(0,x.shape[0]):
    if(q > k):
        y[q] = x[q] - 0.9999*x[q-k]

Y = np.fft.fft(y)

pl.figure(figsize=(11,2))
pl.subplot(1, 2, 1)
setup_plot('x[n]', 'magnitude', 'time')
pl.plot(t1[44100*3:44100*3 + (int)(44100*0.005)],y[44100*3:44100*3 + (int)(44100*0.005
)])

pl.subplot(1, 2, 2)
setup_plot('|X[k]|', 'magnitude', 'sample')
pl.plot(np.abs(Y))
```

Out[33]:

```
[<matplotlib.lines.Line2D at 0x14738588>]
```

```
<matplotlib.figure.Figure at 0x14694e10>
```



This filter eliminated the interfering sinusoid, but now the music clip has an echo.

In [34]:

```
IPython.display.Audio(data=y,rate=44100)
```

Out[34]:

0:00 / 0:15

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
```

In [ ]:

```
```