EE120 Fall 2014 PS 4, synthesizing periodic signal: vowels

To output python notebook, use "ipython nbconvert file.ipynb" from command prompt. This will generate an .html file which you can open in a browser and then print. Printing directly from the iPython notebook gives only a single page. You may need to install pandoc first.

In [23]:
```
print 4+5  # check to see if iPython is running...
```

```
9
```

In [24]:
```
%pylab
```

```
Using matplotlib backend: Qt4Agg
Populating the interactive namespace from numpy and matplotlib

WARNING: pylab import has clobbered these variables: ['sum']
`%matplotlib` prevents importing * from pylab and numpy
```

In [25]:
```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
%matplotlib inline
```

In [26]:
```
# Graphing helper function
def setup_graph(title='', x_label='', y_label='', fig_size=None):
    fig = plt.figure()
    if fig_size != None:
        fig.set_size_inches(fig_size[0], fig_size[1])
    ax = fig.add_subplot(111)
    ax.set_title(title)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
```

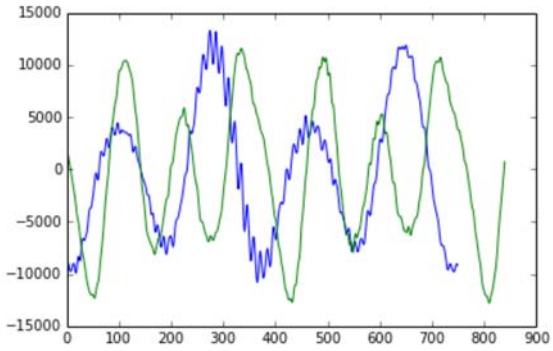
Before running the cell below, capture two short audio files of the vowels "eeee" and "oooo". Audacity works nicely with File -> Export Selection to WAV signed 16 bit PCM.

In [27]:
```
# import file
from scipy.io import wavfile
rate1,data1= wavfile.read('vowel-e.wav')  # 16 bit data if from Audacity
rate1,data2= wavfile.read('vowel-o.wav')  # 16 bit data if from Audacity
print 'rate1 =', rate1
print 'data1 =', data1
lengthE = np.size(data1)/2
print 'length E = ', lengthE
t1 = np.linspace(0, float(lengthE)/float(rate1), lengthE)
#
print 'data2 =', data2
lengthO = np.size(data2)/2
print 'length O = ', lengthO
```

```
rate1 = 44100
data1 = [[-9454 -9455]
 [-9181 -9182]
 [-8998 -8997]
 ...,
 [ 8798  8792]
 [ 8122  8119]
 [ 7492  7489]]
length E =  21504
data2 = [[2412 2411]
 [2012 2005]
 [1610 1602]
 ...,
 [9315 9326]
 [9087 9096]
 [8860 8866]]
length O =  22528
```

In the cell below, select the period (number of samples) for the 2 vowels. This can be determined approximately by inspection of the waveform.

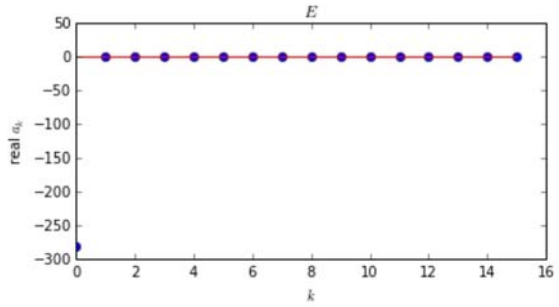
In [28]:
```
# plot data
# t1 = np.linspace(0, float(length)/float(rate1), length)\
length = max(lengthE, lengthO)
vowelE = np.zeros(length)
vowelO = np.zeros(length)
for i in range(0,length):
    if i < lengthE:
        vowelE[i] = data1[i][0] # copy left channel

    if i < lengthO:
        vowelO[i] = data2[i][1]  # copy left channel
# Choose N
Ne = 375 # estimated period of vowel O  TO BE DETERMINED
No = 420 # estimated period of vowel O  TO BE DETERMINED
_ = plt.plot(range(0,2*Ne), vowelE[0:2*Ne])
_ = plt.plot(range(0,2*No), vowelO[0:2*No])
```

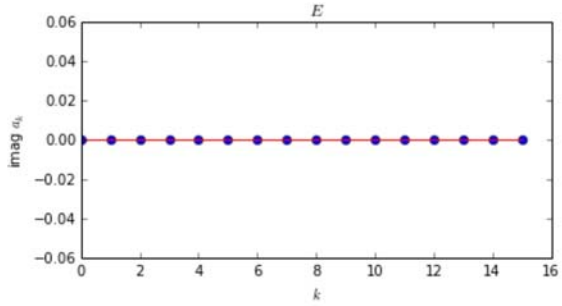


Select appropriate k for reasonable approximation.

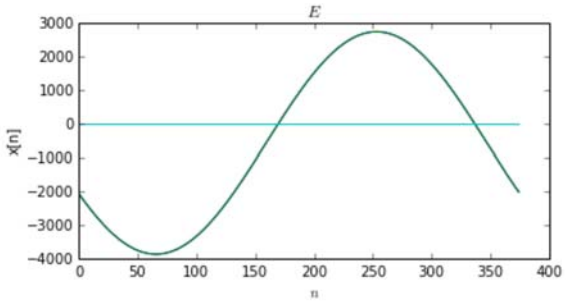
In [35]:
```
# calculate Fourier Series
coeffE = np.exp(1j* np.zeros(16))  # a_k. Since real signal, can use conjugate symmetry for a_{-k}
omega = 2.0 * np.pi/Ne
sum = complex(0,0)  # initialize to complex
for k in range(0,1):   # pick number of Fourier Series coefficients
    sum = 0.0+ 0j
    for n in range(0,Ne):
            sum = sum + vowelE[n] * np.exp(-1j * omega * n * k)
    coeffE[k] = sum/Ne
print 'coefficients:', np.real(coeffE), np.imag(coeffE)
setup_graph(title='$E$', x_label='$k$', y_label='real $a_k$', fig_size=(6,3))
_ = plt.stem(range(0,16), np.real(coeffE))
setup_graph(title='$E$', x_label='$k$', y_label='imag $a_k$', fig_size=(6,3))
_ = plt.stem(range(0,16), np.imag(coeffE))
```

```
coefficients: [-281.704    1.       1.       1.       1.       1.       1.       1.       1.
    1.       1.       1.       1.       1.       1.       1.   ] [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.]
```



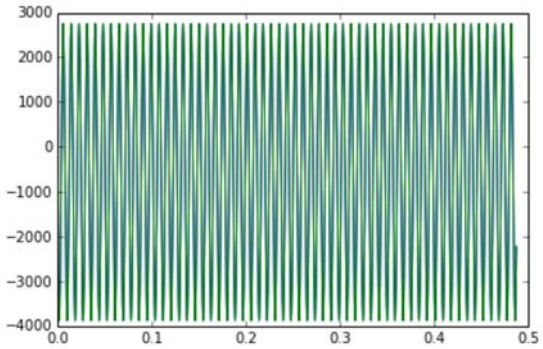


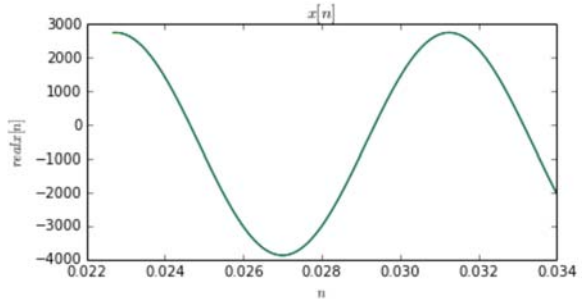
In [34]:
```
# synthesize vowel using a_k coefficients
xe = np.zeros((Ne,2))#just repeat one period
xe_imag = np.zeros((Ne,2))#just repeat one period
for n in range(0,Ne):
    sum = 0+0j
    for k in range(0,2): # pick number of Fourier Series coefficients
            sum = sum + coeffE[k]*np.exp(1j * omega *n*k) + \
                    np.conjugate(coeffE[k]) * np.exp(-1j * omega * n * k)
    xe[n,0]=np.real(sum) # left channel
    xe[n,1]=np.real(sum) # right channel
    xe_imag[n,0] = np.imag(sum)
setup_graph(title='$E$', x_label='$n$', y_label='x[n]', fig_size=(6,3))
_ = plt.plot(range(0,Ne), xe)
_ = plt.plot(range(0,Ne), xe_imag)  # plot to verify zero
```



In [36]:
```
# convert single period to full length
synthE = np.zeros((lengthE,2)) # initialize to zero
for i in range(0,lengthE):
    synthE[i] = xe[np.mod(i,Ne)]
_ = plt.plot(t1, synthE)
print 'synthE', synthE[0:8]
# now write  data file
wavfile.write('synthE.wav', rate1, synthE.astype(int16)) # 16 bit integer
setup_graph(title='$x[n]$', x_label='$n$', y_label='$ real x[n]$', fig_size=(6,3))
_ = plt.plot(t1[1000:1500],synthE[1000:1500])
```

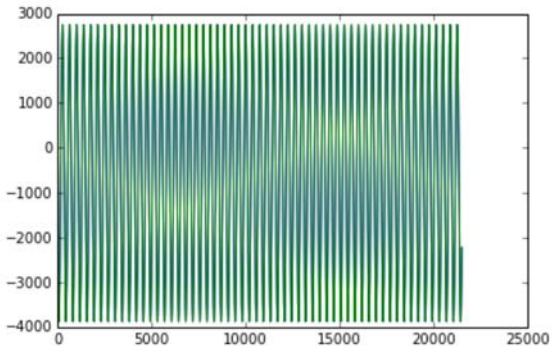
```
synthE [[-2070.78864907 -2070.78864907]
 [-2120.02556399 -2120.02556399]
 [-2168.82549146 -2168.82549146]
 [-2217.17473193 -2217.17473193]
 [-2265.05971237 -2265.05971237]
 [-2312.46699009 -2312.46699009]
 [-2359.38325649 -2359.38325649]
 [-2405.79534084 -2405.79534084]]
```





In [37]:
```
# check tthat can read .wav file that was written
rate1,data3= wavfile.read('synthE.wav')  # 16 bit data if from Audacity
print 'rate1 =', rate1
print 'data3 =', data3
lengthE = np.size(data1)/2
print 'length E = ', lengthE
_ = plt.plot(range(0,lengthE), data3)
```

```
rate1 = 44100
data3 = [[-2070 -2070]
 [-2120 -2120]
 [-2168 -2168]
 ...,
 [-2318 -2318]
 [-2271 -2271]
 [-2223 -2223]]
length E =  21504
```



In [32]: