EE120 Fall 2014 PS 4, synthesizing periodic signal: vowels

To ouput python notebook, use ``ipython nbconvert file.ipynb" from command prompt. You may need to install pandoc first.

In [6]:
```
print 4+5  # check to see if iPython is running...
```

```
9
```

In [7]:
```
%pylab
```

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

In [8]:
```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
%matplotlib inline
```

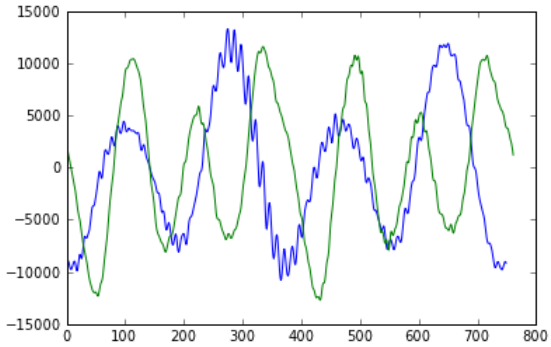
In [9]:
```
# Graphing helper function
def setup_graph(title='', x_label='', y_label='', fig_size=None):
    fig = plt.figure()
    if fig_size != None:
        fig.set_size_inches(fig_size[0], fig_size[1])
    ax = fig.add_subplot(111)
    ax.set_title(title)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
```

In [24]:
```
# 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
length E =  21504
length O =  22528
```

In [71]:
```
# 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 E
print 'period E, freq E=', np.float(Ne)/np.float(rate1), 1.0/(np.float(Ne)/np.float(rate1))
No = 381 # estimated period of vowel O
print 'period O, freq O=', np.float(No)/np.float(rate1), 1.0/(np.float(No)/np.float(rate1))
_ = plt.plot(range(0,2*Ne), vowelE[0:2*Ne])
_ = plt.plot(range(0,2*No), vowelO[0:2*No])
```

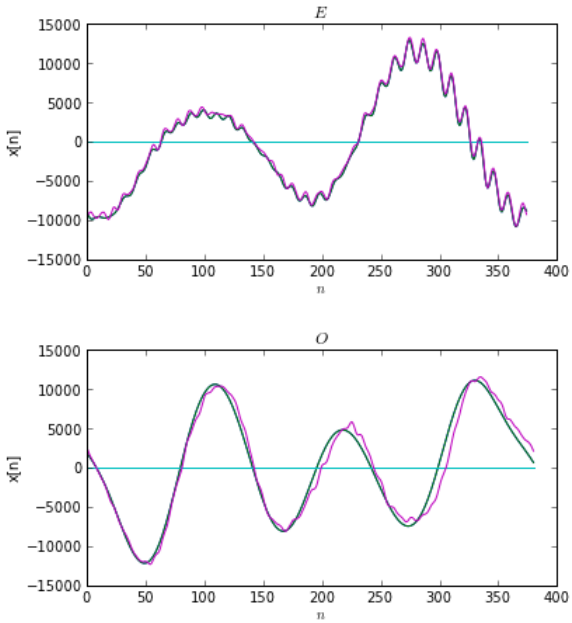
```
period E, freq E= 0.00850340136054 117.6
period O, freq O= 0.00863945578231 115.748031496
```



In [82]:
```
# calculate Fourier Series
numK = 64 # number of FS coefficients
coeffE = np.exp(1j* np.zeros(numK))  # a_k. Since real signal, can use conjugate symmetry for a_{-k}
coeffO = np.exp(1j* np.zeros(numK))
omegaE = 2.0 * np.pi/Ne
omegaO = 2.0 * np.pi/No
sum = complex(0,0)  # initialize to complex
for k in range(0,numK):   # pick number of Fourier Series coefficients
    sum = 0.0+ 0j
    for n in range(0,Ne):
            sum = sum + vowelE[n] * np.exp(-1j * omegaE * n * k)
    coeffE[k] = sum/Ne
    sum = 0.0+ 0j
    for n in range(0,No):
            sum = sum + vowelO[n] * np.exp(-1j * omegaO * n * k)
    coeffO[k] = sum/No

print 'coefficients:', np.real(coeffE), np.imag(coeffE)
setup_graph(title='$E$', x_label='$k$', y_label='real $a_k$', fig_size=(12,8))
plt.subplot(2,2,1)
xlabel('k')
ylabel('Re($a_k$)')
title('E')
_ = plt.stem(range(0,numK), np.real(coeffE))

plt.subplot(2,2,3)
xlabel('k')
ylabel('Im($a_k$)')
_ = plt.stem(range(0,numK), np.imag(coeffE))

plt.subplot(2,2,2)
xlabel('k')
ylabel('Re($a_k$)')
title('O')
_ = plt.stem(range(0,numK), np.real(coeffO))

plt.subplot(2,2,4)
_ = plt.stem(range(0,numK), np.imag(coeffO))
xlabel('k')
ylabel('Im($a_k$)')
```

```
coefficients: [ -2.81704000e+02  -7.53690325e+02  -4.07334962e+03   9.16103334e+01
  -2.37944489e+01   8.02893768e+01   6.34728673e+01   3.90792272e+01
   3.33276472e+01   1.73294056e+01   1.68666496e+01   2.19393771e+01
   3.45522896e+01   2.16027270e+01   1.59838568e+01   1.65259813e+01
   2.28721589e+01   1.93592227e-01  -1.28101553e+01   5.72922128e+01
  -1.49119852e+01   1.21312234e+01   2.75094974e+01   3.44344280e+01
   2.30621517e+01   2.05932611e+01   4.27791146e+01   4.79687080e+01
   5.80627513e+01   1.24924672e+02   1.34135437e+02  -2.26182738e+02
  -2.28339965e+02  -9.35862492e+01   7.60128988e+01   4.34448801e-01
   2.70079042e+00  -4.93132960e+01   1.51160466e+01  -1.12808028e+01
  -5.92642092e+00  -3.49669885e+01  -3.48201599e+01   3.42937460e+00
  -7.95692348e+01  -2.83716418e+00   2.93011065e+00  -9.86553822e+00
   5.55949880e+00  -7.83125197e+00   5.35428455e+00   2.75683115e+00
   1.62537113e+01  -1.24783832e+01   7.37001693e+00  -2.97504364e+01
  -1.37486449e+01  -5.83106991e+00  -1.25445645e+00   2.97185429e+00
  -3.05073552e+00  -4.56881021e+00   1.29852088e+00  -3.55281029e+00] [  0.00000000e+00   1.47568895e+03   3.09781222e+02  -5.
06929747e+02
   1.24384791e+02   2.56480221e+01   6.57678158e+00  -2.92266534e+01
  -3.24958944e+01  -1.79873849e+01  -2.17478173e+01   1.18721053e+01
   5.39417798e+00  -2.35885941e+01  -3.31660871e+00  -5.42860284e+00
  -6.38673401e+00  -2.39738319e+01  -4.89455159e+01   2.83037218e+01
   1.97811167e+01   1.85415298e+01   1.33108479e+01  -7.78091480e+00
   3.46385756e+01   1.67921182e+01   8.26637711e+00   7.73234454e+00
   1.11620566e+01  -3.03520760e+01   2.68787021e+02   2.01285540e+02
  -5.35820868e+00  -5.78653266e+01  -8.83011781e+00  -8.98682072e+01
  -4.91833620e+01  -5.38998565e+01   1.65052281e+01  -3.24289803e+00
  -2.69838730e+01  -3.74745001e+01   1.69495369e+01  -4.61942075e+00
   9.55502346e-01  -1.54675935e+01   5.60511798e+00   4.73673827e-01
   3.97957892e+00   5.59807447e+00  -7.26875317e+00   3.08430805e+00
   1.16156591e+01  -1.92015347e+01  -4.47527431e+00  -3.17528207e+01
  -1.03302933e+01  -5.00129957e+00  -6.63402427e+00   2.09115170e+00
```

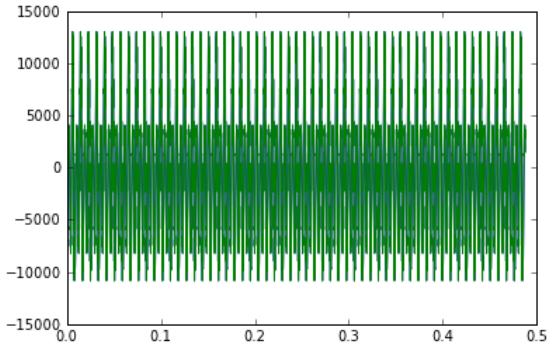
In [86]:
```
# synthesize vowel E using a_k coefficients
xe = np.zeros((Ne,2))#just repeat one period
xe_imag = np.zeros((Ne,2))#just repeat one period
for n in range(0,Ne):
    sumE = 0+0j
    for k in range(0,36): # pick number of Fourier Series coefficients
            sumE = sumE + coeffE[k]*np.exp(1j * omega *n*k) + \
                    np.conjugate(coeffE[k]) * np.exp(-1j * omega * n * k)
    xe[n,0]=np.real(sumE) # left channel
    xe[n,1]=np.real(sumE) # right channel
    xe_imag[n,0] = np.imag(sumE)

# synthesize vowel O using a_k coefficients
xo = np.zeros((No,2))#just repeat one period
xo_imag = np.zeros((No,2))#just repeat one period
for n in range(0,No):
    sumO = 0+0j
    for k in range(0,7): # pick number of Fourier Series coefficients
            sumO = sumO + coeffO[k]*np.exp(1j * omega *n*k) + \
                    np.conjugate(coeffO[k]) * np.exp(-1j * omega * n * k)
    xo[n,0]=np.real(sumO) # left channel
    xo[n,1]=np.real(sumO) # right channel
    xo_imag[n,0] = np.imag(sumO)

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
_ = plt.plot(range(0,Ne), vowelE[0:Ne])  # plot to verify zero
setup_graph(title='$O$', x_label='$n$', y_label='x[n]', fig_size=(6,3))
_ = plt.plot(range(0,No), xo)
_ = plt.plot(range(0,No), xo_imag)  # plot to verify zero
_ = plt.plot(range(0,No), vowelO[0:No])  # plot to verify zero
```

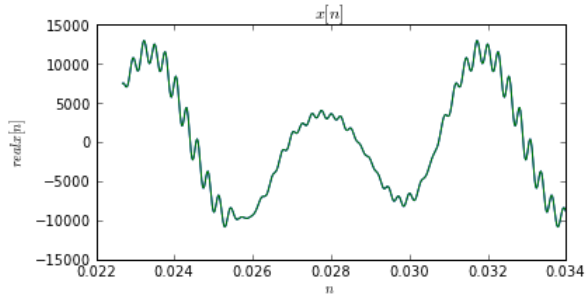


For "ooo" there is a reasonable match with only 7 complex terms for a_k. However the "ee" is not matching the high frequency wiggles, unless k is chosen greater than 32, picking up the harmonic at about 4 kHz. The basic vowel sound is reasonable with k = 7, with only a small visual difference.

In [89]:
```
# 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])
```

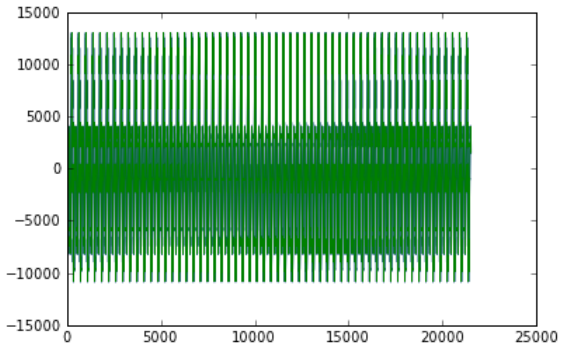
```
synthE [[-9146.76638236 -9146.76638236]
 [-9463.80779821 -9463.80779821]
 [-9705.78726535 -9705.78726535]
 [-9845.38766591 -9845.38766591]
 [-9884.09283144 -9884.09283144]
 [-9845.38949787 -9845.38949787]
 [-9764.19499065 -9764.19499065]
 [-9675.92796568 -9675.92796568]]
```





In [90]:
```
# 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 = [[-9146 -9146]
 [-9463 -9463]
 [-9705 -9705]
 ...,
 [ 1679  1679]
 [ 1529  1529]
 [ 1514  1514]]
length E =  21504
```



In [90]:

In []: