Untitled

May 26, 2015

In [169]:
```
import numpy as np
import matplotlib.pyplot as plt
from math import exp, sin
import math
import random

t = np.loadtxt("C:\Users\IL\Documents\SciComp\HWK7\data.txt",usecols=[0,1])

plt.plot(t[:,0],t[:,1])
plt.show()
```

Smoothing Function, I'll use moving average of width 5

In [170]:
```
bCar = np.zeros((len(t)-4,2))

for i in range(len(t)-4):
    bCar[i,0] = t[i,0]
    bCar[i,1] = 1/5.*(t[i,1]+t[i+1,1]+t[i+2,1]+t[i+3,1]+t[i+4,1])
print bCar
plt.plot(bCar[:,0],bCar[:,1])
plt.show()
```

```
[[  1.00000000e+00   7.79440000e+03]
 [  2.00000000e+00   7.78200000e+03]
 [  3.00000000e+00   7.77420000e+03]
 ...,
 [  9.96000000e+02   8.68960000e+03]
 [  9.97000000e+02   8.69080000e+03]
 [  9.98000000e+02   8.72980000e+03]]
```

Function Fitter. Note I am ignoring 4 points of data for my smoothing function, this might reflect in the chiˆ2 value

In [171]:
```
#x should be a monotonic, evenly space time series
#sShift and pShift are the locations to switch the slope and period, fractions of the way thr
def fit(x, slope1,slope2,amp, per1, per2, sShift, pShift, offset, phase):
    temp = np.zeros(len(x))
    ind = int(sShift*len(x))
    for i in range(len(x)):
        if(i<int(sShift*len(x))):
            if(x[i]<x[int(pShift*len(x))]):
                temp[i] = amp*sin(x[i]/float(per1))+ x[i]*slope1+offset
            else:
                temp[i] = amp*sin(x[i]/float(per2))+ x[i]*slope1+offset
```

```
            else:
                if(x[i]<x[int(pShift*len(x))]):
                    temp[i] = amp*sin(x[i-ind]/float(per1)+phase)+ x[i-ind]*slope2+x[ind]*slope1+o
                else:
                    temp[i] = amp*sin(x[i-ind]/float(per2)+phase)+ x[i-ind]*slope2+x[ind]*slope1+o

        return temp

    test = fit(bCar[:,0],2.75,-1.2,500,9.5,9,.6,.3,7700, 3.14*.25)
    plt.plot(bCar[:,0],test)
    plt.plot(bCar[:,0],bCar[:,1])
    plt.show()
```

```
In [172]:
    #Black Box way to find the baseline, nothing to vary/iterate though?
    z = np.polyfit(bCar[:,0],bCar[:,1] ,2)
    fi = np.poly1d(z)
    plt.plot(bCar[:,0],fi(bCar[:,0]))
    plt.show()
```

Random parameter searching

```
In [175]: chi = 10**5
    temp = 0
    ite = 10000
    para = np.zeros(9)
    for n in range(ite):

        sShift = random.uniform(.5,.7)
        pShift = random.uniform(.25,.5)
        sl1 = random.uniform(2,3)
        sl2 = random.uniform(-1,-2)
        amp = random.uniform(450,500)
        per1 = random.uniform(8,9)
        per2 = random.uniform(8,9)
        offset = random.uniform(7600,7900)
        phase = random.uniform(0,3.14)
        randFit = fit( bCar[:,0],sl1,sl2,amp, per1, per2, sShift,pShift, offset,phase)

        for i in range(len(bCar)):
            temp += (bCar[i,1]-randFit[i])**2/randFit[i]

        if(temp < chi):
            chi = temp
            print chi
            para = [sl1, sl2, amp,per1,per2,sShift,pShift,offset,phase]
        temp =0

    print para
    test = fit(bCar[:,0],para[0],para[1],para[2],para[3],para[4],para[5],para[6],para[7], para[8])
    plt.plot(bCar[:,0],test)
    plt.plot(bCar[:,0],bCar[:,1])
    plt.show()
```

```
22007.4894721
16426.1426318
```

```
13427.4510599
7052.73341045
6831.57643345
6205.52194887
[2.801345622417074, -1.442161893949937, 455.1054023705578, 8.678189267905662, 8.488490518127653, 0.5795
```

```
In [154]: #save best parameter parameters
          best = [5886.37821516,2.788567091085996, -1.772548039994956, 463.5936329098231, 8.528950176949
          print best
```

```
[5886.37821516, 2.788567091085996, -1.772548039994956, 463.5936329098231, 8.528950176949616, 8.395827045
```

I ran the above code multiple times changing the ranges of the random number generators absed on where the best fit appeared.

Value at x = 1200

```
In [167]: ind = int(para[5]*len(bCar))
          pred = para[2]*sin(1200/float(para[4])+para[8])+ 1200-bCar[ind,0]*para[1]+bCar[ind,0]*para[1]+
          print pred
          plt.plot(bCar[:,0],test)
          plt.plot(bCar[:,0],bCar[:,1])
          plt.plot(1200,pred,'ro')
          plt.show()
```

```
8597.18846389
```

```
In [176]: #Check Uncertainty, new value, chi^2 6205
          para = [2.801345622417074, -1.442161893949937, 455.1054023705578, 8.678189267905662, 8.488490
          pred = para[2]*sin(1200/float(para[4])+para[8])+ 1200-bCar[ind,0]*para[1]+bCar[ind,0]*para[1]+
          print pred
```

```
8751.75002506
```

So a different set of parameters gives a fairly different answer, expected since the answer is oscillating and I am predicting a value that is 200 units (multiple wavelengths) away.

```
In [ ]:
```