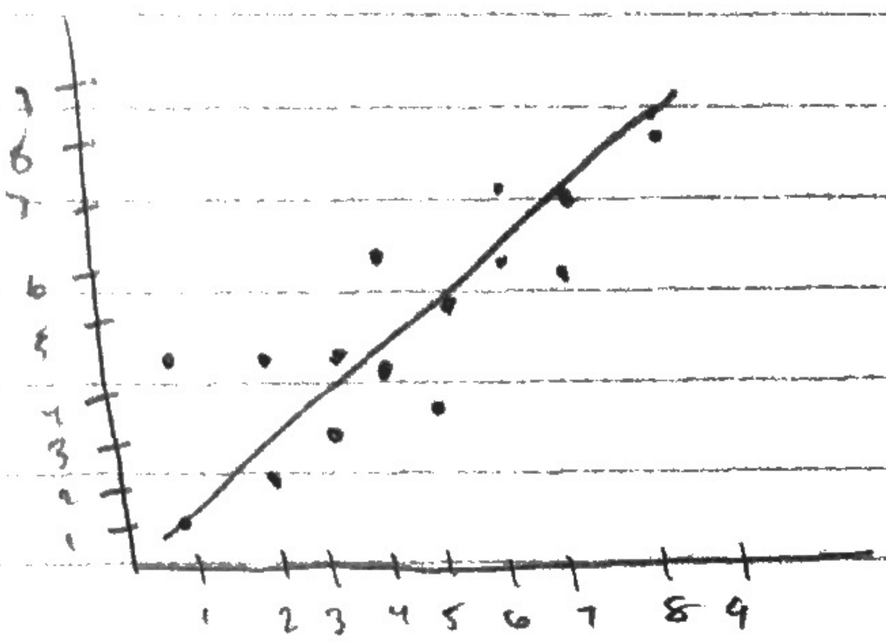
Mathew Joseph
EE16A HW

1. Nate Holmes - 25047248
Trevor Hansen - 25327650

2. a)



Finding x_i

$a^T a = 204$, $a^T b = 210$

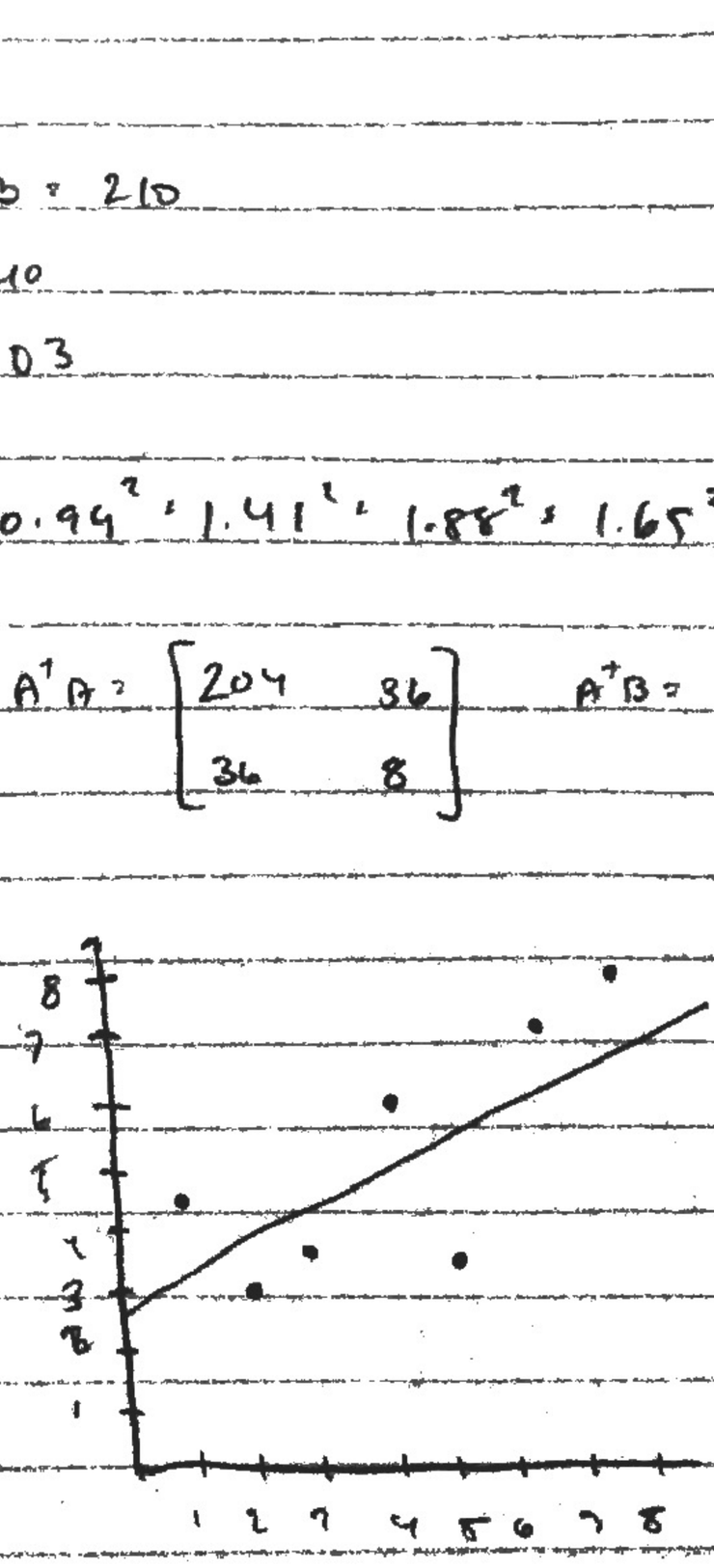
$\rightarrow 204x = 210$

$x = 1.03$

$|\varepsilon|^2 = 3.47^2 + 0.94^2 + 1.41^2 + 1.88^2 + 1.65^2 + 0.82^2 = 24.82$

b) x vector: $A^T A = \begin{bmatrix} 204 & 36 \\ 36 & 8 \end{bmatrix}$ $A^T B = \begin{bmatrix} 210 \\ 42 \end{bmatrix}$

$\rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.9 \\ 3 \end{bmatrix}$



$|3|^2 = 1^2 + 1^2 + 0 + 1^2 + 2^2 = 1$

$= 10$

c) $\langle \vec{b} - A\vec{x}, A \rangle = A^T(b - A\vec{x})$, which is equal to 0.

$A^T b - A^T A x = 0$

$\rightarrow A^T A x = A^T \vec{b}$

3 a) We select promotions in order of their similarity between $\vec{x}_c$ and $\vec{s}_n$. The higher, the faster we select it

b) $\|\vec{x}_c - \vec{s}_a\|$ would be bad as the users we are calculating vectors based on how different they are.

$\frac{1}{\|\vec{x}_c - \vec{s}_a\|}$ would be good as the value is higher as they are more similar

$\langle \vec{x}_c, \vec{s}_a \rangle$ is good as inner products are almost by definition a way for saying $\langle \vec{x}_c, \frac{\vec{s}_a}{\|\vec{s}_a\|} \rangle$ is just the normalized version of this, so it is still good.

c) Cpcake: 0.4 flour + 0.33 milk + 0.22 egg + 0.05 butter

Cream: 0.7 flour + 0.1 milk + 0.1 egg + 0.1 butter

Cookie: 0.2 flour + 0.1 milk + 0.15 egg + 0.55 butter

Cake: 0.05 flour + 0.02 milk + 0.2 egg + 0.73 butter

d) $\begin{bmatrix} 40 & 70 & 20 & 5 \\ 33 & 10 & 10 & 2 \\ 22 & 10 & 15 & 20 \\ 5 & 10 & 55 & 73 \end{bmatrix} \begin{bmatrix} \vec{s}_0 \end{bmatrix} \cdot \vec{x}_c$

e) Based on part d,

$\vec{x}_{C_1} = \begin{bmatrix} 47.5 \\ 17.5 \\ 18.5 \\ 16.5 \end{bmatrix}$ $\vec{x}_{C_2} = \begin{bmatrix} 3.33 \\ 22.67 \\ 23.8\overline{3} \\ 80.167 \end{bmatrix}$ $\vec{x}_{C_3} = \begin{bmatrix} -7.5 \\ 2.5 \\ 11.5 \\ 93.5 \end{bmatrix}$

$\vec{x}_{C_4} = \begin{bmatrix} 32.5 \\ 6 \\ 15 \\ 41.5 \end{bmatrix}$ so, your Doe: $\begin{bmatrix} 37.5 \\ 25 \\ 6.25 \\ 31.25 \end{bmatrix}$

e) Yes -- if the null space is empty or the system itself is overdetermined.

4. a) $[\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \alpha_4 \quad \alpha_5] \begin{bmatrix} \text{age} \\ \text{weight} \\ \text{ts2} \\ \text{ts1} \\ \text{Chol} \end{bmatrix} = \vec{b}$

b) $\vec{x} = \begin{bmatrix} -0.156 \\ 0.0924 \\ 0.48 \\ -0.555 \\ -0.354 \end{bmatrix}$ c) $[1, -1, -1, 1]$

5. a) $[a \quad d \quad e] \begin{bmatrix} x^2+y^2 \\ x \\ y \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 1 \end{bmatrix}$

b) $[a \; b \; c \; d \; e] \begin{bmatrix} x^2 \\ xy \\ y^2 \\ x \\ y \end{bmatrix} = \begin{bmatrix} \vec{1} \\ 1 \end{bmatrix}$

c) in iPython

d) in iPython. The error is smaller in this fit, so this technique is better.

6. a) We see that the error is small

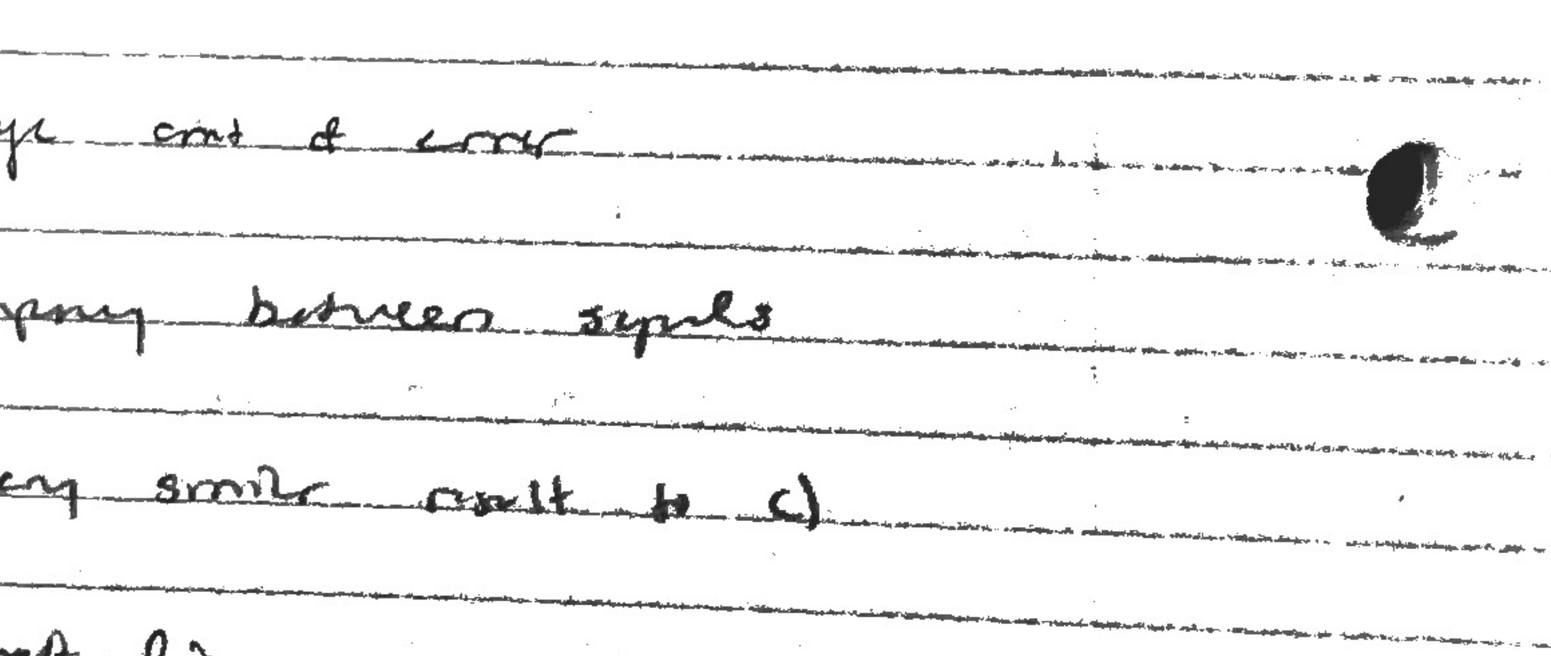
b) we see a large cost of error

c) we see a discrepancy between signals

d) we see a very similar result to c)

7. Find the Line of best fit

t	$\vec{b}$
0	-1
1	3
2	1



$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \\ 1 \end{bmatrix}$$

$A^T A$: $\begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 3 \\ 3 & 3 \end{bmatrix}$

$(A^T A)^{-1} = \frac{1}{6-3-3(3)} \begin{bmatrix} 3 & -3 \\ -3 & 5 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 3 & -3 \\ 3 & 8 \end{bmatrix}$

$x = (A^T A)^{-1} A^T b$

$x = \frac{1}{6} \begin{bmatrix} 3 & -3 \\ -3 & 5 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$

$y = t - 2$

In [1]:
```
%pylab inline
import numpy as np
import matplotlib.pyplot as plt
import scipy.io
import sys
```

```
Populating the interactive namespace from numpy and matplotlib
```

Labeling Patients

In [3]:
```
# Part B
A = np.load('gene_data_train.npy')
b = np.load('diabetes_train.npy')
np.linalg.lstsq(A, b)
```

Out[3]:
```
(array([[-0.15646169],
        [ 0.09239418],
        [ 0.48053974],
        [-0.5847018 ],
        [-0.35350734]]),
 array([ 0.63021776]),
 5,
 array([ 55.54141902,   5.44019995,   3.94168171,   2.38586289,   1.
90939811]))
```

In [11]:
```
# Part C
A_test = np.load('gene_data_test.npy')
b_test = np.load('diabetes_test.npy')
x = np.array([[-0.15646169], [ 0.09239418], [ 0.48053974], [-0.5847018
pred_b = np.dot(A_test, x).tolist()
used_vals = []
for i in pred_b:
    used_vals.append(i[0])
binary_list = [1 if i > 0 else -1 for i in used_vals]
binary_list
```

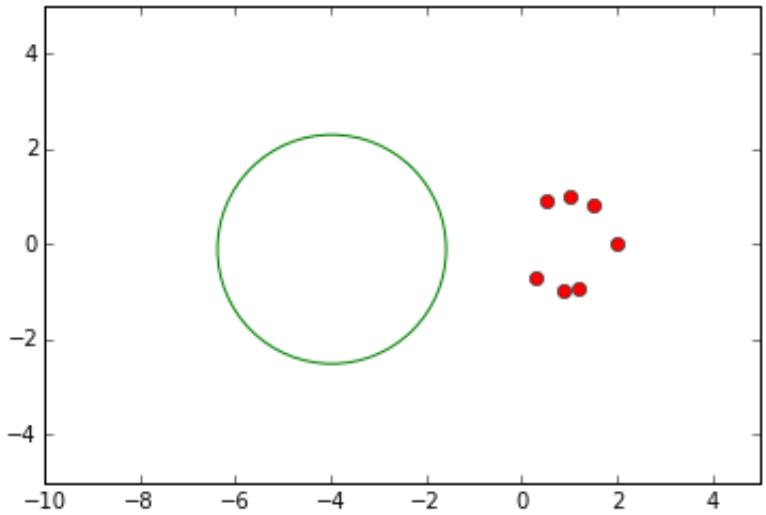
Out[11]:
```
[1, -1, -1, 1]
```

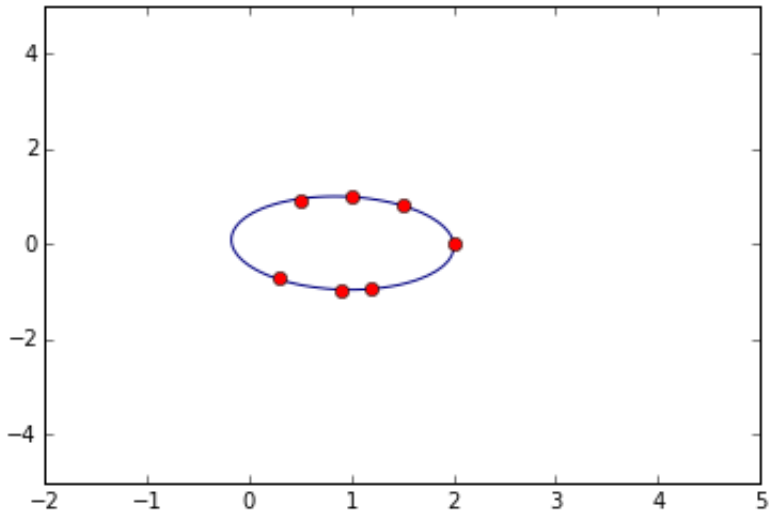
Image Analysis

In [18]:

```
# Our vars
data = [[0.3, -0.7], [0.5, 0.91], [0.9, -0.99], [1, 1.01], [1.2, -0.93]

r_sq = []
x_sq = []
y_sq = []

for i in data:
    r_sq.append(np.square(i[0]) + np.square(i[1]))
    x_sq.append(np.square(i[0]))
    y_sq.append(np.square(i[1]))

x_times_y = [(i[0] * i[1]) for i in data]

# creating A shapes

A_circle = np.array([
        [data[0][0], data[0][1], r_sq[0]],
        [data[1][0], data[1][1], r_sq[1]],
        [data[2][0], data[2][1], r_sq[2]],
        [data[3][0], data[3][1], r_sq[3]],
        [data[4][0], data[4][1], r_sq[4]],
        [data[5][0], data[5][1], r_sq[5]],
        [data[6][0], data[6][1], r_sq[6]]
    ])


A_ellps = np.array([
        [x_squared[0], xy[0], y_squared[0], data[0][0], data[0][1]],
        [x_squared[1], xy[1], y_squared[1], data[1][0], data[1][1]],
        [x_squared[2], xy[2], y_squared[2], data[2][0], data[2][1]],
        [x_squared[3], xy[3], y_squared[3], data[3][0], data[3][1]],
        [x_squared[4], xy[4], y_squared[4], data[4][0], data[4][1]],
        [x_squared[5], xy[5], y_squared[5], data[5][0], data[5][1]],
        [x_squared[6], xy[6], y_squared[6], data[6][0], data[6][1]]
    ])
b = np.array([
        [1],
        [1],
        [1],
        [1],
        [1],
        [1],
        [1],
    ])

circle_fit = np.linalg.lstsq(A_circle, b)
fig=plt.figure(1)
plt.axis([-10 , 5, -5, 5])
plt.plot([data[0][0], data[1][0], data[2][0], data[3][0], data[4][0], d
ax=fig.add_subplot(1,1,1)

circ=plt.Circle((-3.98007129,-0.11237247), radius=2.39722071, color='g'
ax.add_patch(circ)
```

```
plt.show()

# plot ellipse fit
ellipse_fit = np.linalg.lstsq(A_ellps, b)
x = np.linspace(-2.0, 5.0, 100)
y = np.linspace(-5.0, 5.0, 100)
X, Y = np.meshgrid(x,y)

F = 2.95103059*(X**2) + 3.66108715*(Y**2) + 0.4670502*(X*Y) -5.41586506
plt.contour(X,Y,F,[0])
plt.plot([data[0][0], data[1][0], data[2][0], data[3][0], data[4][0], d
plt.show()
```



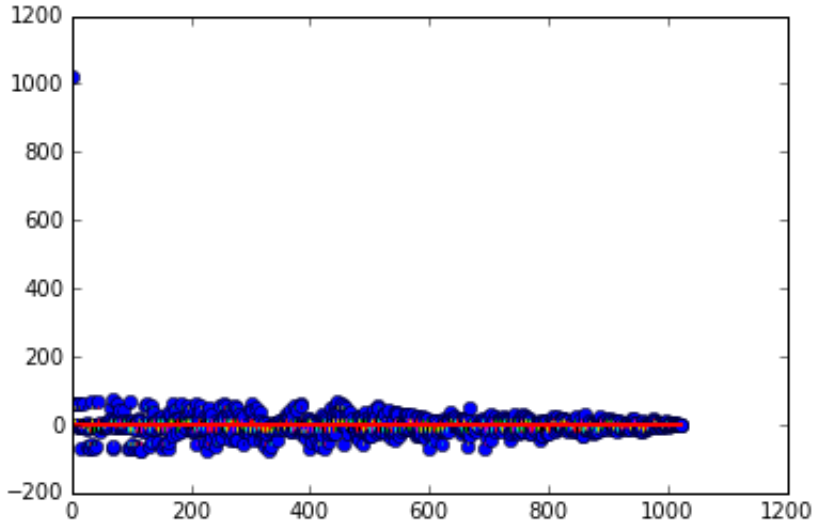


GPS Locationing

In [19]:
```
## RUN THIS FUNCTION BEFORE YOU START THIS PROBLEM
## This function will generate the gold code associated with the satell
## The satellite_ID can be any integer between 1 and 24
def Gold_code_satellite(satellite_ID):
    codelength = 1023
    registerlength = 10

    # Defining the MLS for G1 generator
    register1 = -1*np.ones(registerlength)
    MLS1 = np.zeros(codelength)
    for i in range(codelength):
        MLS1[i] = register1[9]
        modulo = register1[2]*register1[9]
        register1 = np.roll(register1,1)
        register1[0] = modulo

    # Defining the MLS for G2 generator
    register2 = -1*np.ones(registerlength)
    MLS2 = np.zeros(codelength)
    for j in range(codelength):
        MLS2[j] = register2[9]
        modulo = register2[1]*register2[2]*register2[5]*register2[7]*re
        register2 = np.roll(register2,1)
        register2[0] = modulo

    delay = np.array([5,6,7,8,17,18,139,140,141,251,252,254,255,256,257
    G1_out = MLS1;
    G2_out = np.roll(MLS2,delay[satellite_ID - 1])

    CA_code = G1_out * G2_out

    return CA_code
```

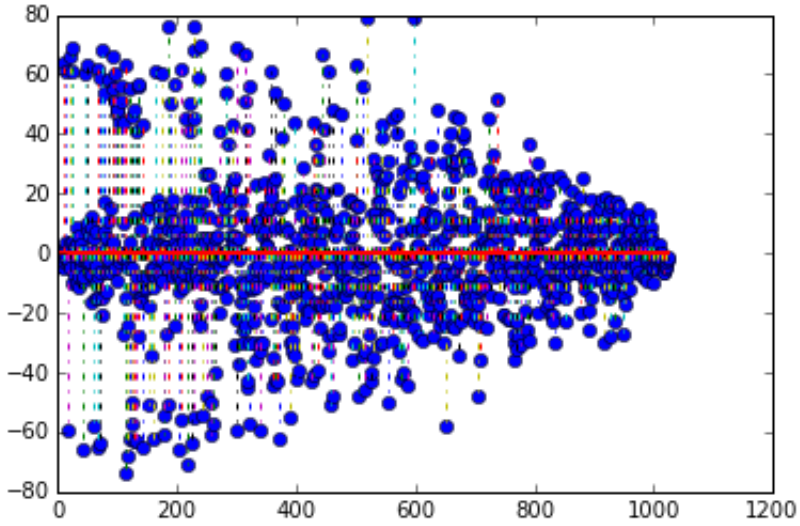
In [26]:
```
## PART A CODE HERE
def array_correlation(array1,array2):
    """ This function should return two array or a matrix with one row
    the offset and other to the correlation value
    """
    result = numpy.correlate(array1, array2, mode='full')
    result = result[result.size/2:]
    result = np.expand_dims(result, axis=1)
    x = np.array([range(0, result.size)]).transpose()
    new = np.c_[result, x].transpose()
    return new


correlation = array_correlation(Gold_code_satellite(10), Gold_code_sate

print('Autocorrelation of Signal:')
x = np.linspace(0, 1023, 1023)

markerline, stemlines, baseline = plt.stem(x, auto[0,:], '-.')
plt.setp(markerline, 'markerfacecolor', 'b')
plt.setp(baseline, 'color', 'r', 'linewidth', 2)

plt.show()
```

```
Autocorrelation of Signal:
```



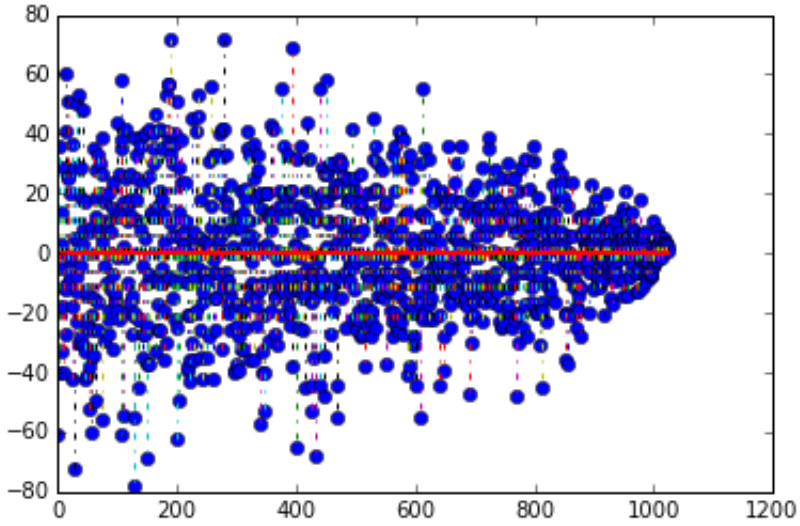
In [27]:
```
## PART B CODE HERE

#cross correlating 10,13
correlation = array_correlation(Gold_code_satellite(10), Gold_code_sate
print('Cross correlation of Signal:')
x = np.linspace(0, 1023, 1023)
markerline, stemlines, baseline = plt.stem(x, correlation[0,:], '-.')
plt.setp(markerline, 'markerfacecolor', 'b')
plt.setp(baseline, 'color', 'r', 'linewidth', 2)

plt.show()
```

```
Cross correlation of Signal:
```



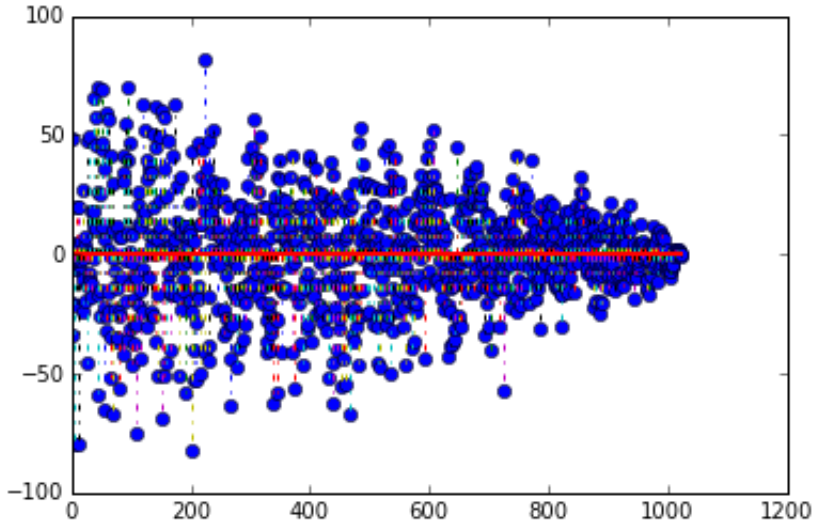
In [28]:
```
## PART C CODE HERE
## THIS IS A HELPER FUNCTION FOR PART C
def integernoise_generator(length_of_noise):
    noise_array = np.random.randint(2,size = length_of_noise)
    noise_array = 2*noise_array - np.ones(size(noise_array))
    return noise_array


correlation_noise = array_correlation(Gold_code_satellite(10), integern

print('Cross correlation of Signal:')
x = np.linspace(0, 1023, 1023)
markerline, stemlines, baseline = plt.stem(x, correlation_noise[0,:], '
plt.setp(markerline, 'markerfacecolor', 'b')
plt.setp(baseline, 'color', 'r', 'linewidth', 2)

plt.show()
```

```
Cross correlation of Signal:
```



In [31]:
```
## PART D CODE HERE
## THIS IS A HELPER FUNCTION FOR PART D
def gaussiannoise_generator(length_of_noise):
    noise_array = np.random.normal(0,1,length_of_noise)
    return noise_array

cross_gauss = array_correlation(Gold_code_satellite(10), gaussiannoise_

print('Cross correlation of Signal:')
x = np.linspace(0, 1023, 1023)
markerline, stemlines, baseline = plt.stem(x, cross_gauss[0,:], '-.')
plt.setp(markerline, 'markerfacecolor', 'b')
plt.setp(baseline, 'color', 'r', 'linewidth', 2)

plt.show()
```

```
Cross correlation of Signal:
```



In [2]:

In [1]:

In [ ]:

In [ ]: