In [74]:
```
import numpy as np
def GradientDescent(A,b,x,alpha):
    return x - alpha * A.T.dot(A.dot(x) - b)
def SoftThresholdingOp(x,p,alpha):
    for c in x:
        c = np.sign(c) * max(abs(c)-p*alpha, 0)
    return x
A = np.load("assignment0828_A-Copy1.npy")
b = np.load("assignment0828_b-Copy1.npy")
p = 1
alpha = 1/max(np.linalg.eig(A.T.dot(A))[0])
#alpha = 1e-5
#x = range(2000)
#x = np.reshape(x,(2000,1))
for n in range (2000):
    x[n] = [1]
for k in range(5000):
    x = GradientDescent(A,b,x,alpha)
    #x = SoftThresholdingOp(x,p,alpha)
    i = 0
    for c in x:
        c = np.sign(c) * max(abs(c)-p*alpha, 0)
        x[i] = c
        i += 1
    #print(np.linalg.norm(A.dot(x) - b))
    #print(x)
    #print(sum(x))
x
```

Out[74]:
```
array([[ 0.00134817+0.j],
       [-0.        +0.j],
       [-0.01850942+0.j],
       ...,
       [ 0.16413786+0.j],
       [-0.        +0.j],
       [ 0.        +0.j]])
```

In [75]:
```
#A.T.dot(A.dot(x)-b).shape
#A.T.shape
#(A.dot(x)-b).shape
#A.dot(x).shape
#b.shape
#np.linalg.eig(A.T.dot(A))
#x = range(2000)
#x = np.reshape(x,(2000,1))
#for n in range (2000):
#    x[n] = [1]
#print(x)
print("Printing the number of zeros in x (what we want to maximize):")
counter = 0
for c in x:
    if c == 0:
        counter += 1
print(counter)
print("printing the norm of x:")
print(sum(x))
print("Sanity Check, printing ||Ax-b|| (should be close to 0):")
print(np.linalg.norm(A.dot(x) - b))
```

```
Printing the number of zeros in x (what we want to maximize):
1063
printing the norm of x:
[62.12315385+0.j]
Sanity Check, printing ||Ax-b|| (should be close to 0):
0.8430575973963907
```