Ideally, I would like to have gathered way more data points. I was able to achieve an almost exact beta match at around n = 300 observations/samples. Below or my two attempts at the lasso regression algorithm implementation. The first uses essentially the exact same notation from the computer assignment pdf. However, it results in a lot of chatter. The second uses a separate implementation based on the beginning parts of the slides. In other words, I define the cost function similar to the gradient descent problem, and then add the lambda L1 regularization term. This graph came out much cleaner but does not seem to be affected by varying lambda.


```python
def lasso_regression(parmsl, lamda, x_batch, y_batch):

    betas = parmsl
    def soft_threshold(a, b):
        #could be g from slide 14
        #or beta from slide 15
        if a > b:
            return a - b
        elif a < -b:
            return a + b
        else:
            return 0

    #define cost function with L1 regulariation
    cost_b4_l1 = (1/2) * np.sum(  (y_batch - (np.sum(np.dot(x_batch , betas)))) **2
    L1_Regularization = lamda * np.sum(np.abs(betas))
    cost = cost_b4_l1  +  L1_Regularization

    new_betas = []
    for i in range(len(betas)):
        XiT = x_batch[:,i].T
        X_i = np.delete(x_batch,i,1)
        B_i = np.delete(betas,i)
        y = y_batch
        Xi = x_batch[:,i]

        gi =  np.dot(XiT , (y - np.dot(X_i , B_i) )) / np.dot(XiT , Xi)

        z = lamda / np.linalg.norm(Xi)**2

        new_beta = soft_threshold(gi, z)
        new_betas.append(new_beta)

    weights = np.array(new_betas)
    weight_next = weights

    return cost, weight_next
```



```python
################################################################################
# Here you create you own LASSO algorithm for the linear regression model
# Hint: please check the slides of the gradient descent (1.5, the 7th to 14th page)
################################################################################
"""
notes:
p=7
n= 30
y size = n x 1 = [30 x 1]
X size = n x (p+1) = [30 x 8] ?
beta size = (p+1) x 1 = [8 x 1]
"""

def lasso_regression(parmsl, lamda, x_batch, y_batch):

    betas = parmsl

    #define prediction
    y_pred = np.dot(x_batch, betas)

    #define cost function with L1 regulariation
    cost_b4_l1 = (1/2) * np.sum(  (y_batch - (np.sum(x_batch * betas))) **2    )
    L1_Regularization = lamda * np.sum(np.abs(betas))
    cost = cost_b4_l1  +  L1_Regularization

    #define derivtive
    dcost = np.dot(x_batch.T, y_pred - y_batch) + lamda * np.sign(betas)

    #calcualte new betas
    new_betas = betas - 0.0001  * dcost

    weights = np.array(new_betas)
    weight_next = weights

    return cost, weight_next

################################################################################
```