# Assignment 3

Praffullitt Jain
20171142

**Question 1**

a. The coefficients calculated using closed form of linear regression are-

$b_0$ =77731, $b_1$=141.83, $b_2$= -6361.9

cost = b0 + b1*area + b2*rooms

Error calculated using L2 norm = 5.8 %

Price of a house with size 1400 square meters and 4 bedrooms = 250845

b. Normalization did not have any impact on the error calculated using L2 norm. It just scaled all the values between 0 and 1. And made the variables comparable.

c. Yes, the mean passes through the regression line.

d. No, the complexity of the closed form of linear regression is of the order N^3 i.e. it will be 10^18 for a million data points.

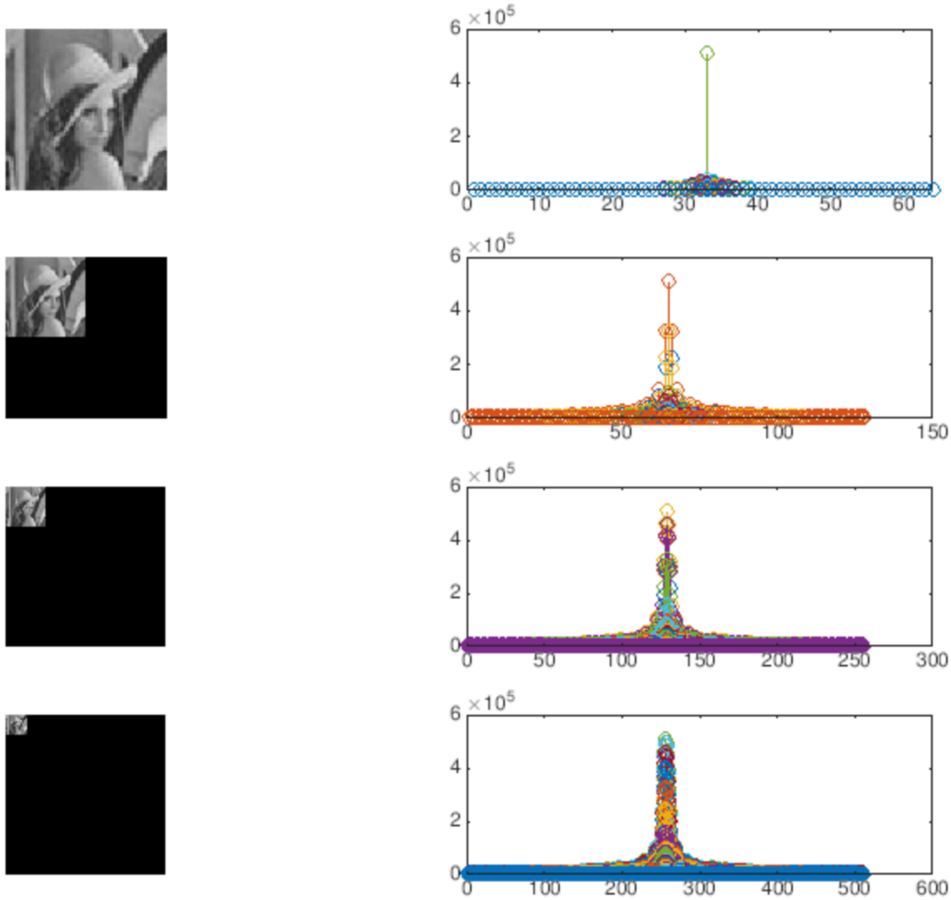
**Question 2**

a. iDFT[FH] does not correspond to the center of center of f*h because of lack of padding and unequal sizes etc.

b. The average of squared difference between pixel values in iDFT[FH] and center of f*h is of the order 10^17.

c. After padding, iDFT[FH] corresponds to the center of center of f*h. And the error reduces to 0.1017.

## Question 3




## Question 4

The two indices obtained after fft are 881 and 1320.

But frequency = index x (Fs/size)

= index x (1/3)

Therefore, frequencies are 293.66 and 439.99.

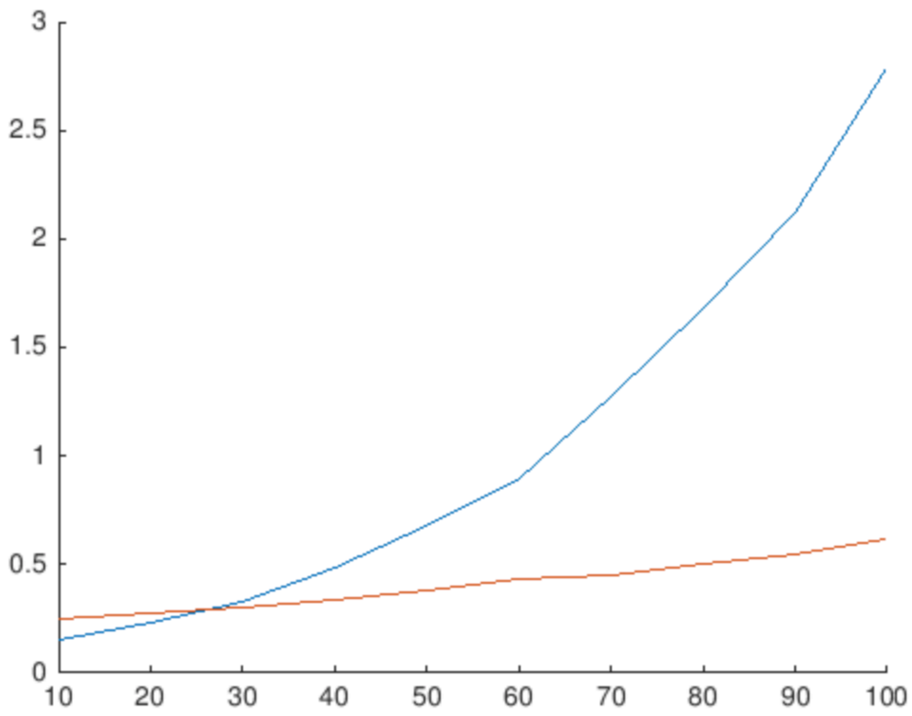
### Question 5

Correct sequence = 3, 5, 1, 2, 4

Steps:
- Isolate the first and last 5 seconds of each signal.
- Run correlation to identify which audio signals match.
- Starting signal is the one with minimum starting correlation with the end of the rest.

### Question 6

Complexity of default method = O(N^2 * K^2)
Complexity of improved method = O(N^2 * K)



## Question 7

**Gradient Descent Algorithm** is an iterative algorithm to find a **Global Minimum** of an objective function (cost function). In gradient descent, our first step is to initialize the parameters by some value and keep changing these values till we reach the global minimum. In this algorithm, we calculate the derivative of cost function in every iteration and update the values of parameters simultaneously using the formula:

$$\theta := \theta - \alpha \frac{\delta}{\delta\theta} J(\theta).$$

In linear regression we have a hypothesis function: $H(X) = \theta_0 + \theta_1 X_1 + \ldots + \theta_n X_n$

In order to solve the model, we try to find the parameter, such that the hypothesis fits the model in the best possible way.To find the value of parameters we develop a cost function J(θ) and use gradient descent to minimize this function.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2,$$

Cost function (ordinary least square error)

$$h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Gradient of Cost function

1. **Batch gradient descent**

It is the first basic type of gradient descent in which we use the complete dataset available to compute the gradient of cost function. As we need to calculate the gradient on the whole dataset to perform just one update, batch gradient descent can be very slow and is intractable for datasets that don't fit in memory.

2. **Stochastic gradient descent**

Batch Gradient Descent turns out to be a slower algorithm. So, for faster computation, we prefer to use stochastic gradient descent. The first step of algorithm is to randomize the whole training set.

Then, for updating every parameter we use only one training example in every iteration to compute the gradient of cost function. As it uses one training example in every iteration this algorithm is faster for larger data set. In SGD, one might not achieve accuracy, but the computation of results are faster.

**3. Mini batch gradient descent**

Mini batch algorithm is the most favorable and widely used algorithm that makes precise and faster results using a batch of ‘ m ‘ training examples. In mini batch algorithm rather than using the complete data set, in every iteration we use a set of ‘m’ training examples called batch to compute the gradient of the cost function.