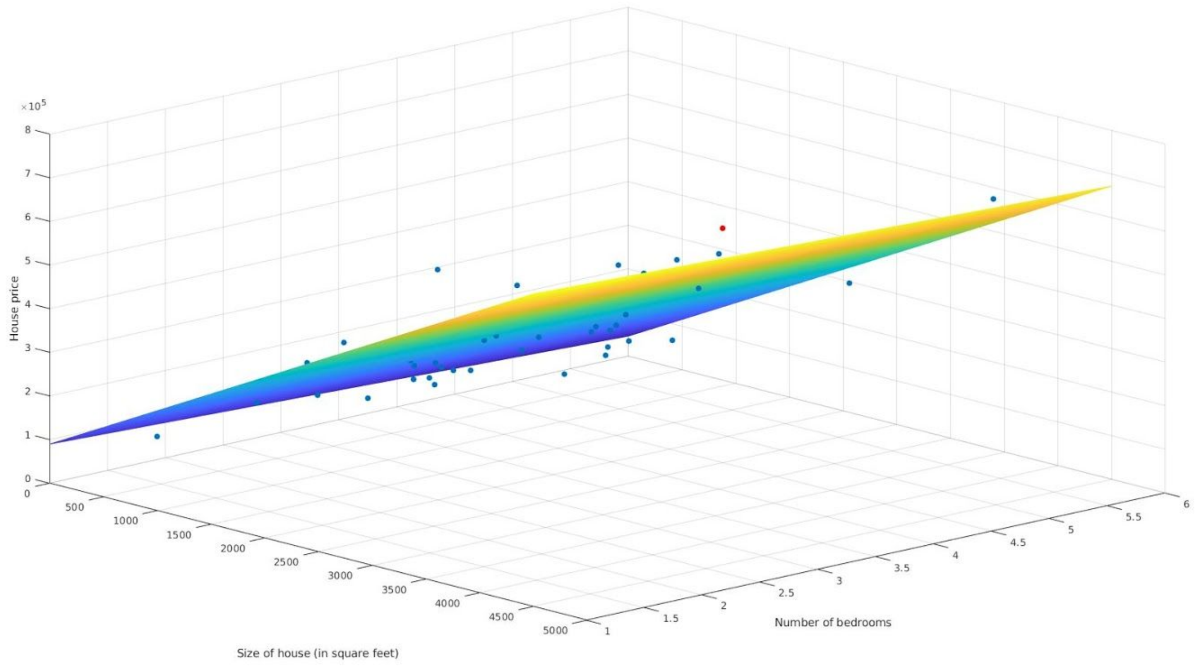
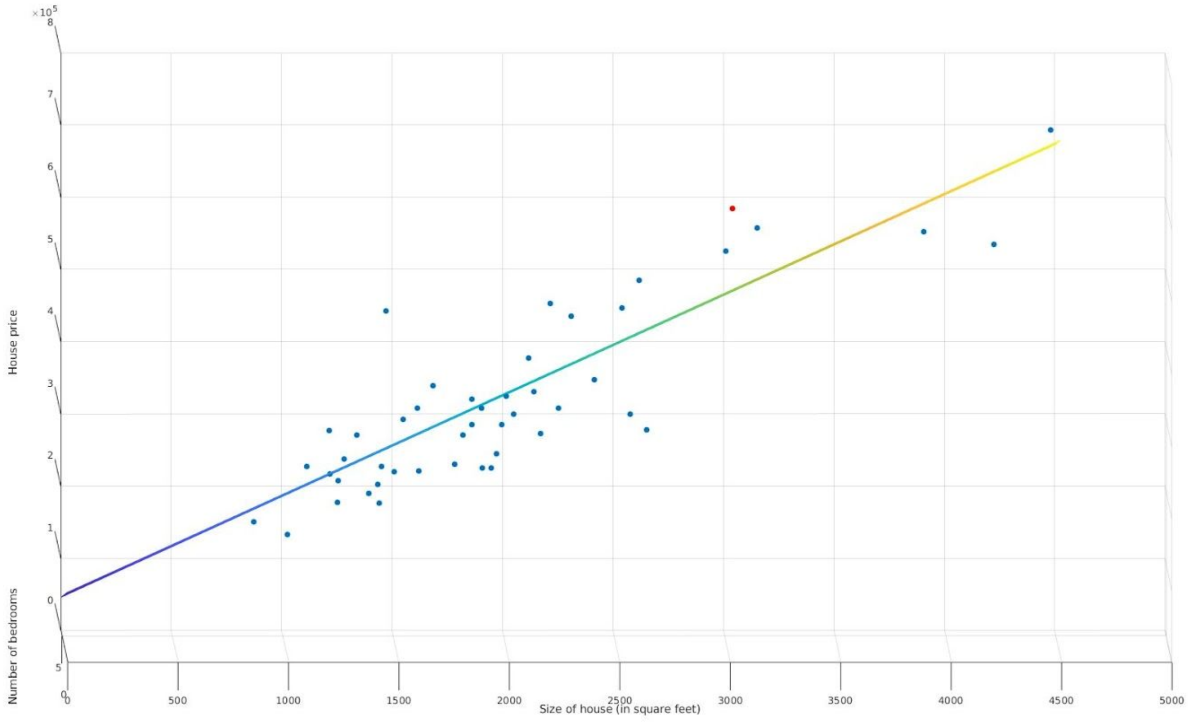
DSAA Assignment 3

-Meher Shashwat Nigam, 20171062

Question 1:





- Performed linear regression on the dataset given in the file houses.csv using method taught in class: Normal equation method.

$$\boxed{\theta = \left(X^T X\right)^{-1} . \left(X^T y\right)}$$

θ : hypothesis parameters that define it the best.
X : Input feature value of each instance.
Y : Output value of each instance.

- The columns in the data file represent size of house (Square feet), no. of bedrooms & price of house respectively. For this question, splitted the dataset into two parts:
- 90% for training and remaining 10% to test results.
- To estimate the correctness of solution, generated the L2 norm of predicted and ground truth house prices in the test set.
- L2 norm of predictions: 6.4191e+05
- L2 norm of actual values: 6.0635e+05
- Percentage error : 5.8637%

Part 1:

The coefficients corresponding to each feature of the house:

```
Theta  =          [ 77731 // constant term
              142     // size of house
             -6362  // no. of bedrooms
             ]
```

Predicted house price of a 1400 sq feet, 4 bedroom house = 250850

Part 2:

Normalizing the features doesn't help in any way for the method that we are using. Feature scaling
helps in faster convergence of gradient descent. The results that we get with and without normalization are the same .

Part 3:

The mean of all features passes through the regression plane we generated.

Been verified by putting mean of the features as input to our regression and noticed the value is same
as the mean of the actual output vector.

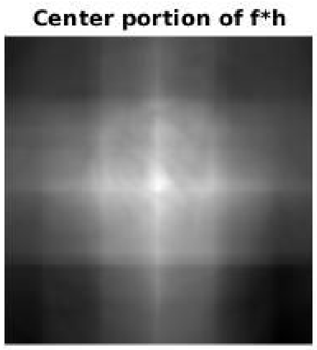
Part 4:

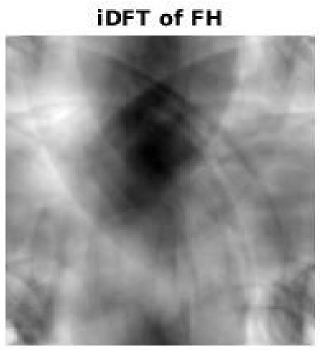
This method can’t be used when the number of rows of data is 1 million. Due to the involved
complexities of matrix multiplications and inverses:
For a least squares regression with N training examples and C features, it takes:

- O(C2N) to multiply XT by X
- O(CN) to multiply XT by Y
- O(C3) to compute the LU (or Cholesky) factorization of XTX and use that to compute the product (XTX)−1(XTY)
  Asymptotically N dominates C so we can say that the time complexity is O(C2N).

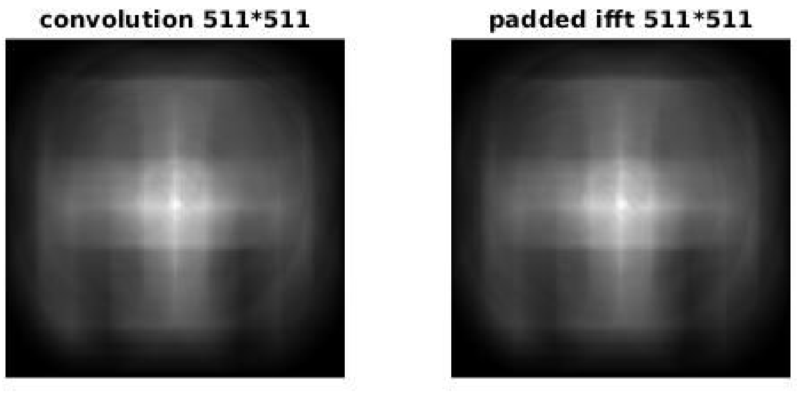
Question 2:

- **Part 1:**Convolution function built into MATLAB does linear convolution. This cannot be compared with the IFFT of the product of the FFTs of the individual images.





- **Part 2:** Average square difference between the two: 6.370079554142548e+13
- **Part 3:**

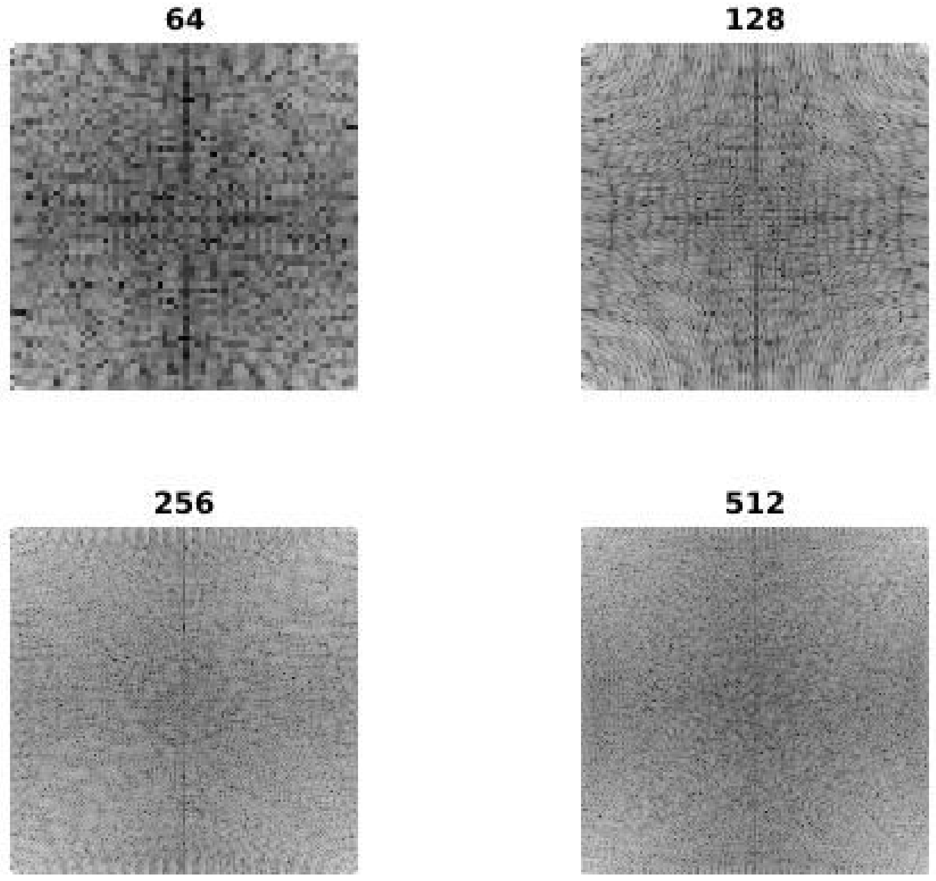


- Linear convolution after padding however becomes a special case of circular convolutions, thereby making the convolution of the padded image and the FFT method (Convolution Theorem), will match, with average squared difference of *2.587805541403562e-18(approx 0)* which tells us it is equal (with minor errors introduced while taking fft and ifft etc.)

Question 3:

- The fourier transform of each of the images 64x64, with no padding, and 128x128, 256x256 and 512x512, with padding, when visualized by taking the log of their absolute values, we can see a greater resolution in each of the visualizations.
- The observation is due to the larger sample size, i.e., the resolution for frequency domain increases due to the fact that with each padded image, the total number of samples increases with the same distribution in frequency.

- This agrees with the observation for 1-D signals, where more samples for the same sampling frequency provides us with a greater resolution in frequency for the same signal.
- The visualizations are as follows:



Question 4:

- The frequencies of the particular key pressed are clearly observable from the fft of the sound signal. The frequencies are 294 Hz and 440 Hz.
- To clean the signal, I used a threshold filter that makes values zero in the fourier domain if the it exceeds a certain limit, and then took the inverse fourier transform.
- The peaks are found at X = 882 and 1321.
- These are scaled using the scaling factor = sampling frequency / number of samples = 44100/132301 = 0.33

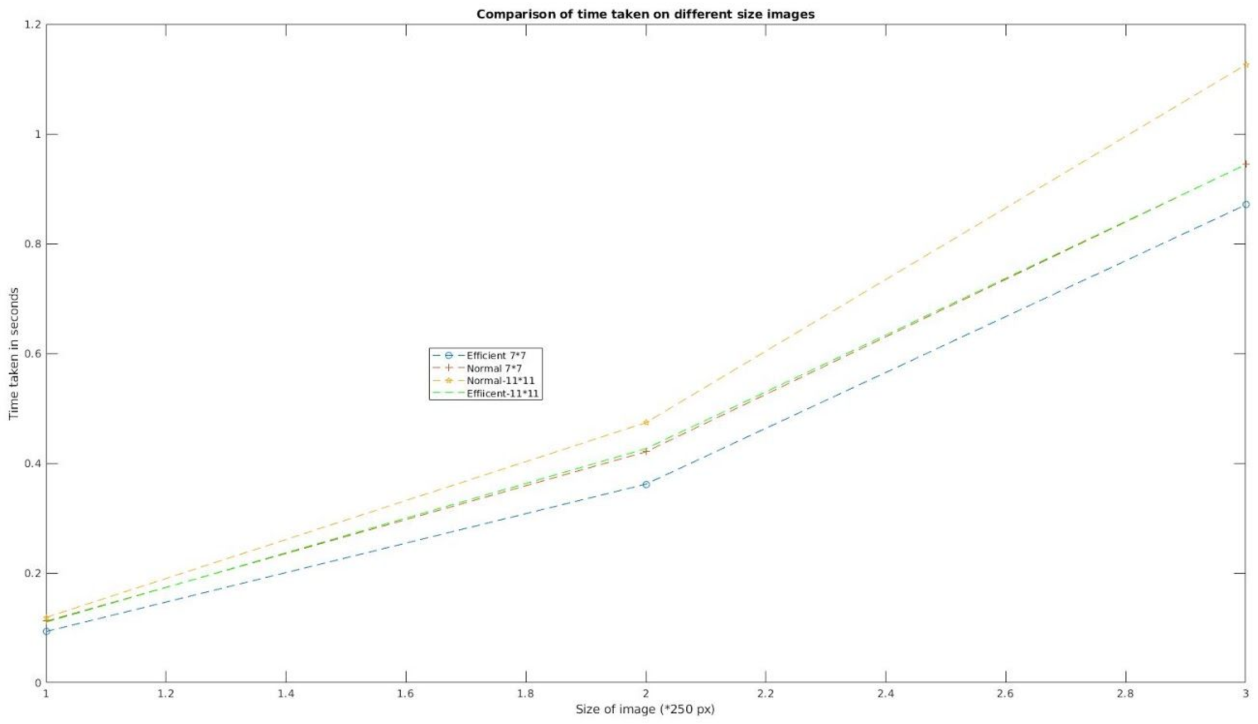
Question 5:

The correct ordering of the packets is

3 5 1 2 4

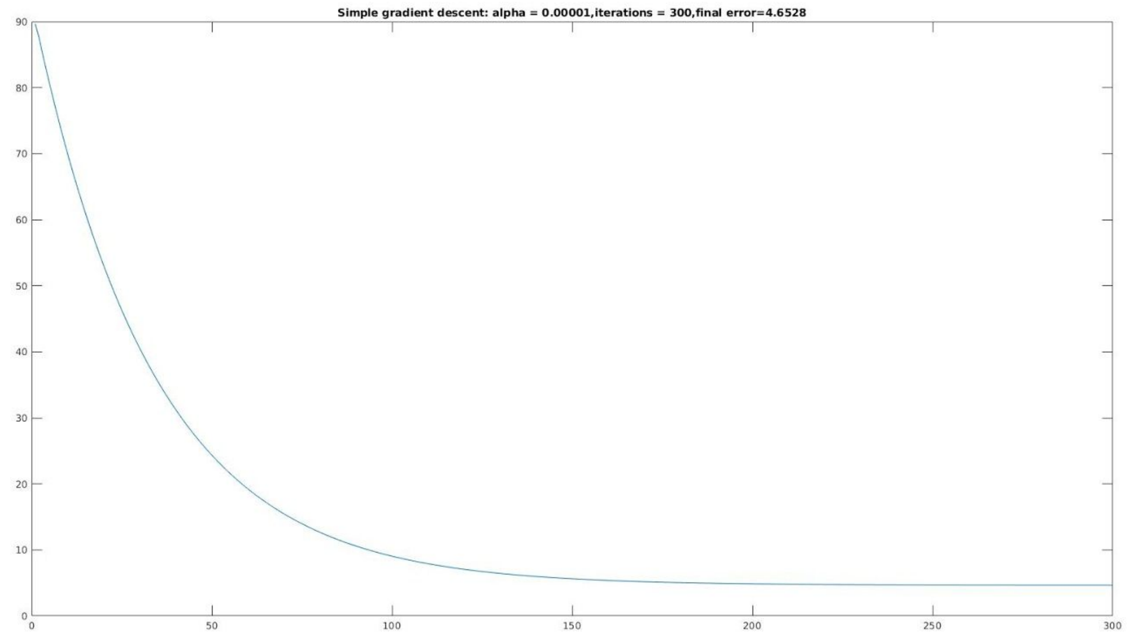
- To find out the ordering of the packets, separate out the first 5 seconds and last 5 seconds of the packets, then generate all permutations of the numbers 1 to 5 and find out the cross correlation for the corresponding pairs of the ends of packets to starts of packets based on the permutation.
- Then, we can obtain the peaks of each of these cross correlations, sum up all of these peaks for each ordering, and pick the maximum of these (sum of peaks). The maximum will correspond to permutation of correct sequence.
- To remove noise, a Butterworth low pass filter is used with frequency cap of 800Hz, followed by smoothdata function for a Gaussian filter with a window size of 5.

Question 6:

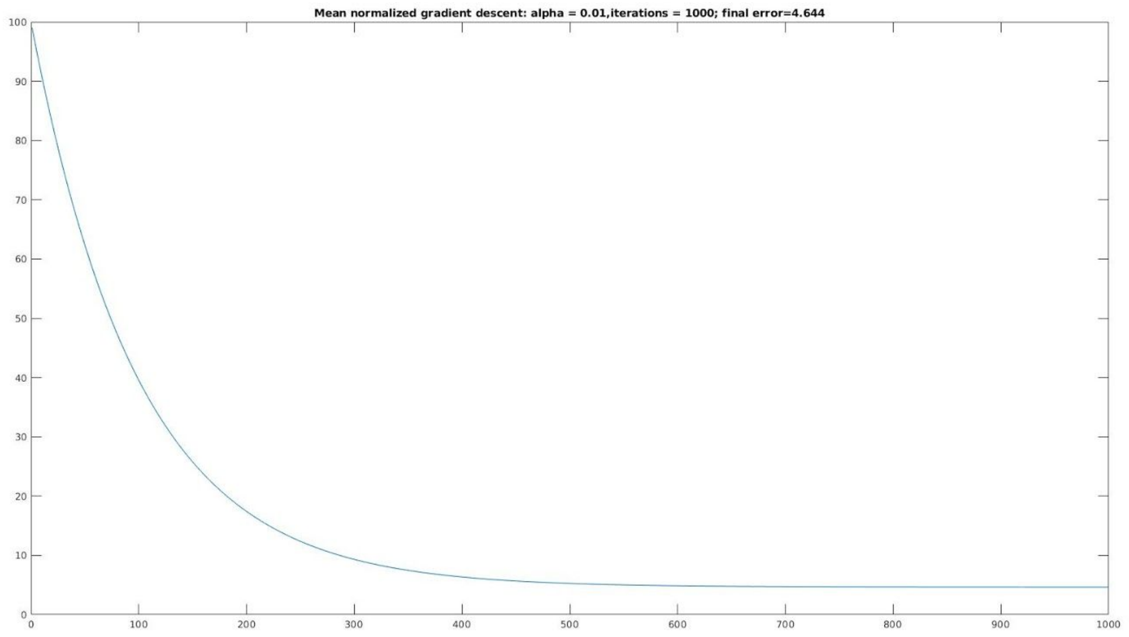


The implementation of linear spatial filters requires moving a mask centered at each pixel of the image, computing sum of products of mask coefficients and corresponding pixels.
PART1 Implemented an algorithm for low-pass filtering a grayscale image by moving a k k averaging filter of the form ones(k)/(k^2).
PART2 As the filter is moved from one spatial location to the next one, the filter window shares many common pixels in adjacent neighborhoods. Exploiting this observation, implemented a more efficient version of averaging filter.

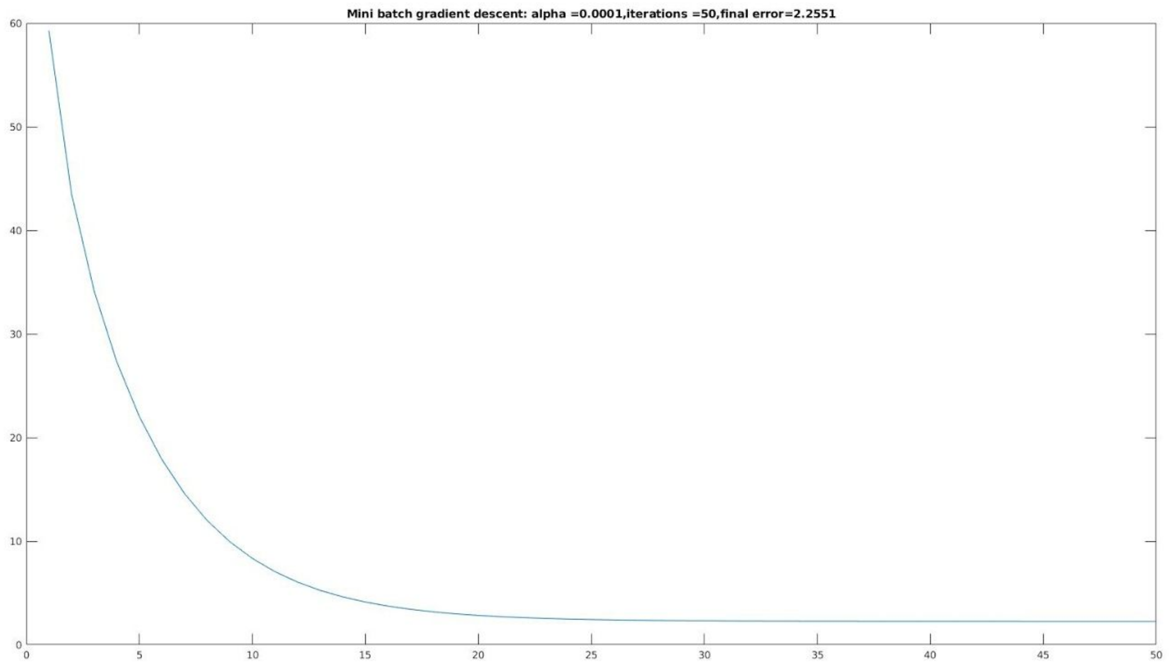
Question 7:



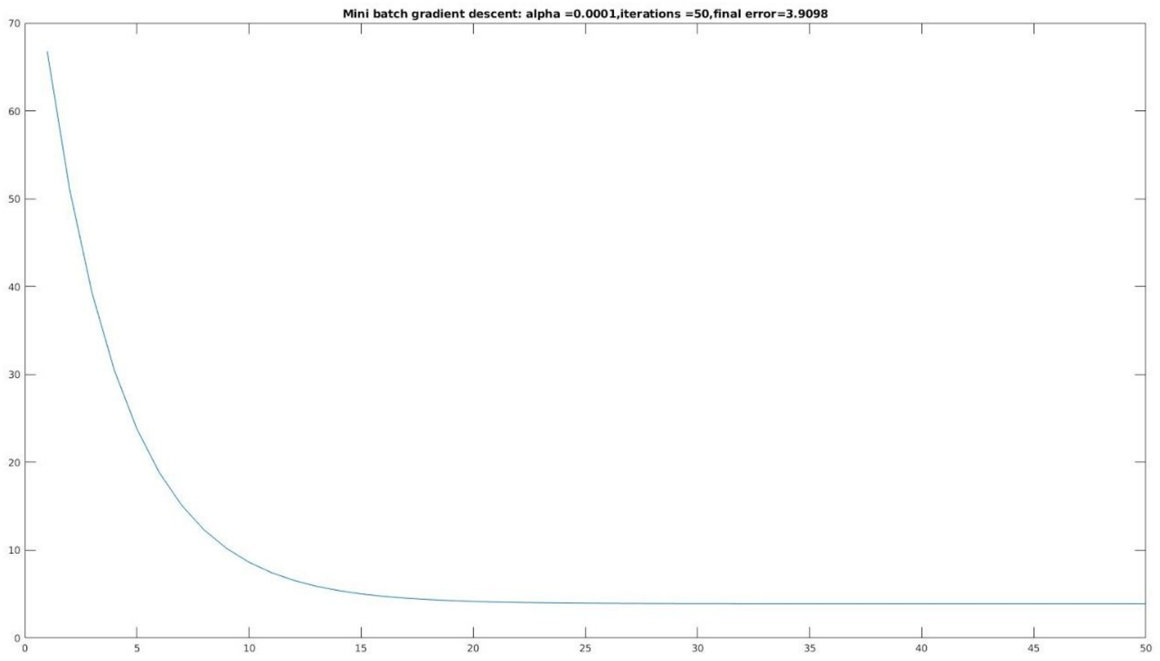
Simple gradient descent



Mean Normalized gradient descent



Mini batch gradient descent ; Batch size = 450



Mini batch gradient descent ; Batch size = 150

- To approach this question I have implemented gradient descent in 3 different methods.
- Batch Gradient Descent
    - Error rate of *4.6528%*.
    - Learning Rate = 0.00001 and iterations = 300
- Mini Batch Gradient Descent
    - Batch size of *450*, Error rate of *2.2551%*
    - Batch size of *150*, Error rate of *3.9098%*.
    - Learning rate = 0.0001 and iterations = 50
- Mean Normalized Gradient Descent
    - Error rate of 4.644%
    - Learning rate of 0.01 and iterations = 1000