**Homework 1**

**Qn 1**

This paper aims to illustrate the different techniques of optimization to solve for various minimization problems. For the first section, we implemented and plotted the results of Gradient Descent using

**Gradient Descent**

**Function** NormalGradientDescent:

1. Set start\_point, convergence\_criterion and step\_ size
2. Current\_point = start\_point
3. **While** location(start\_point – current\_point > convergence criterion):
4. Calculate g(w)
5. Calculate gradient of g(w)
6. Start\_point = makes an update to the weight vector based on one data point
7. Calculate distance(Start\_point – current guess\_
8. Current\_guess = Start\_point
9. **end while**

**end function**

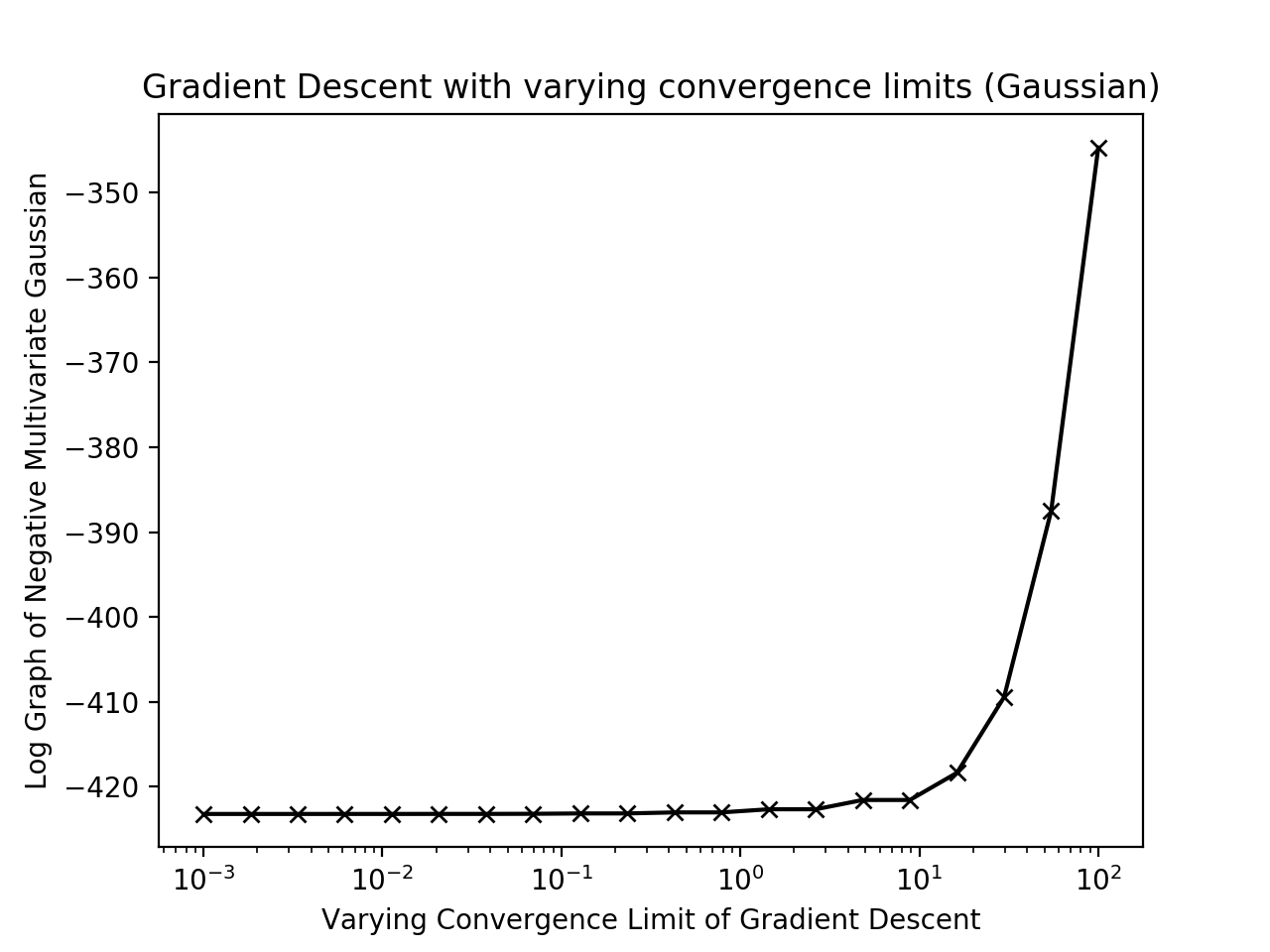
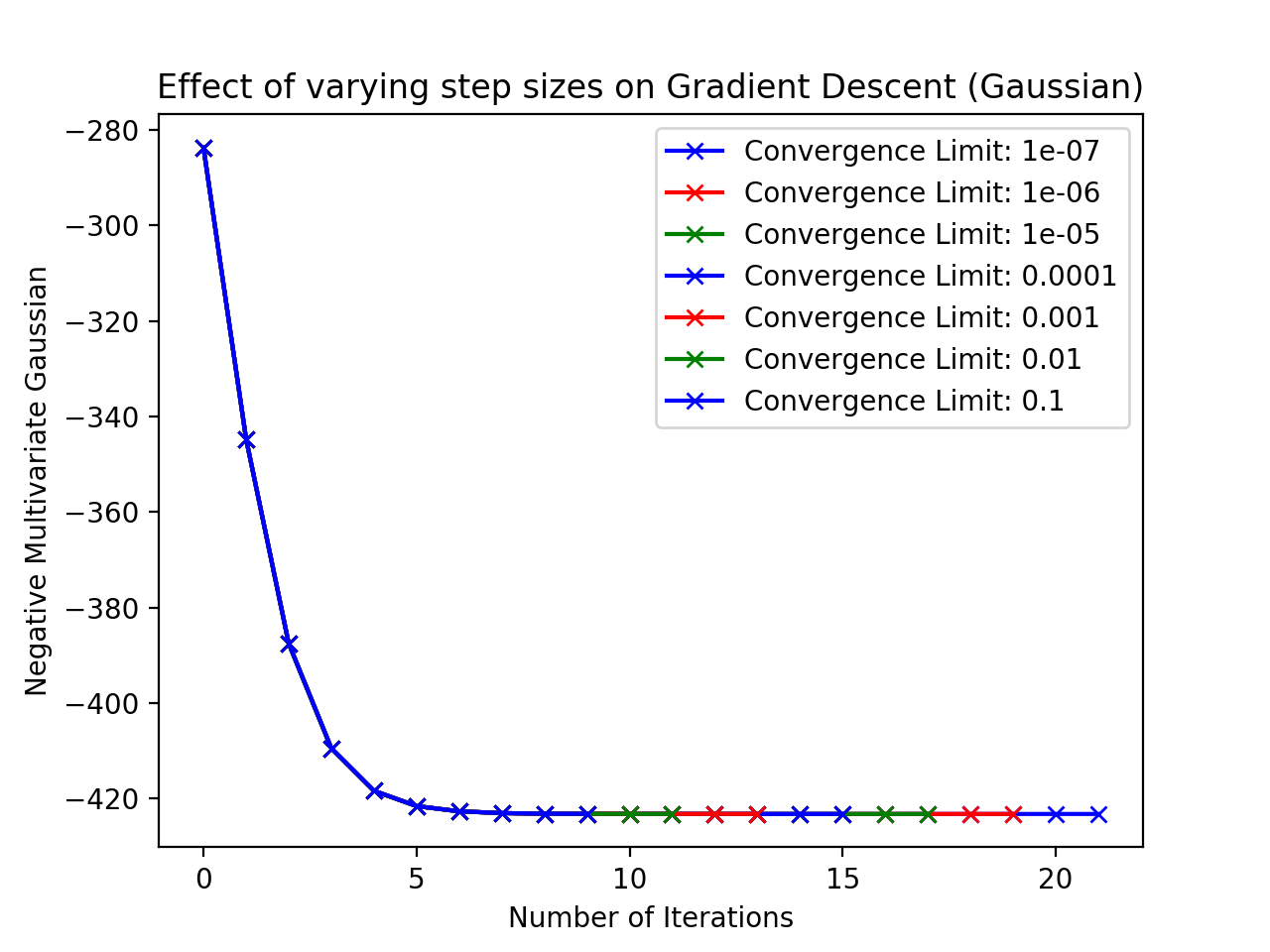
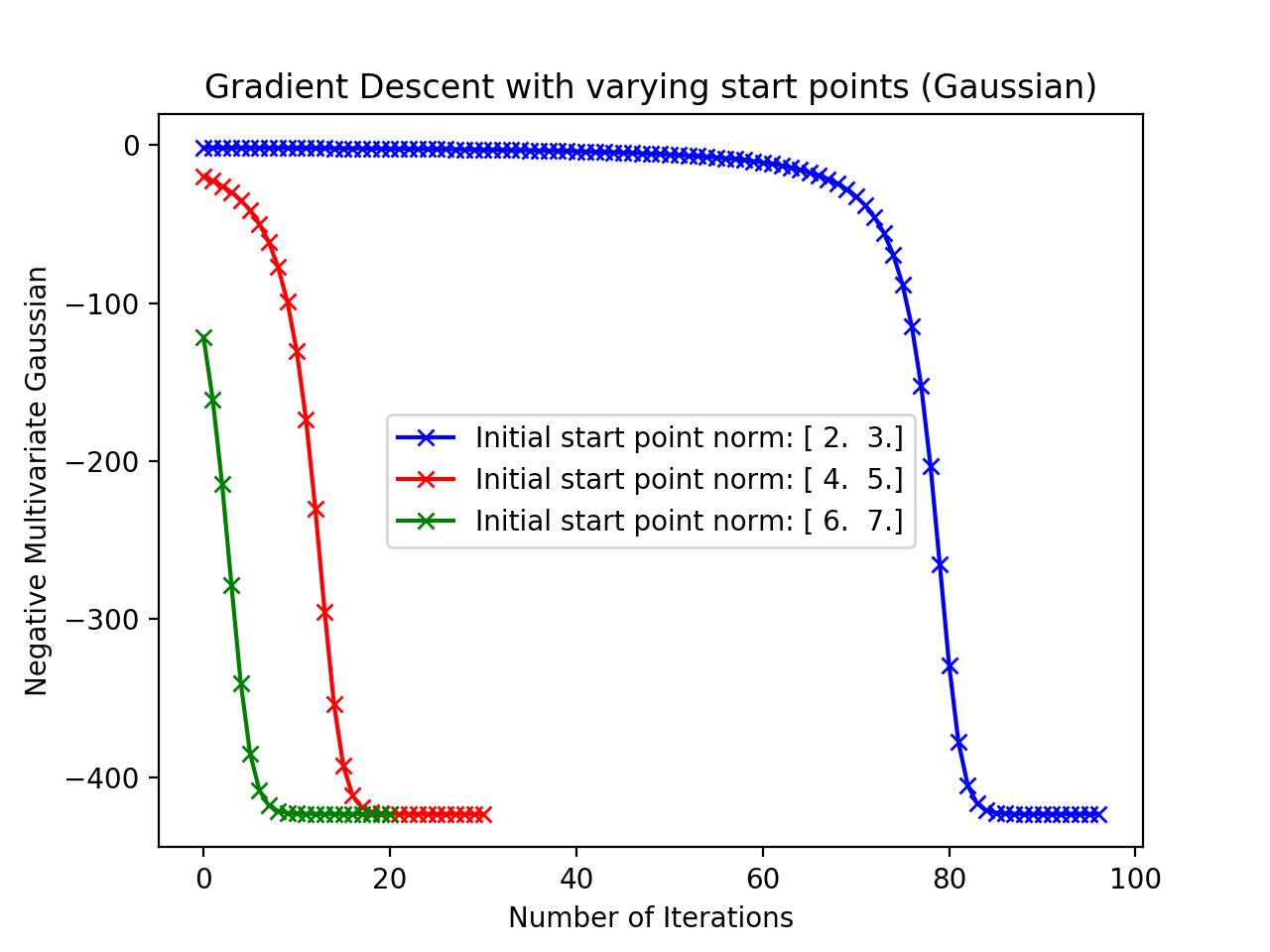
Python. The algorithm, in pseudocode, is described as below:

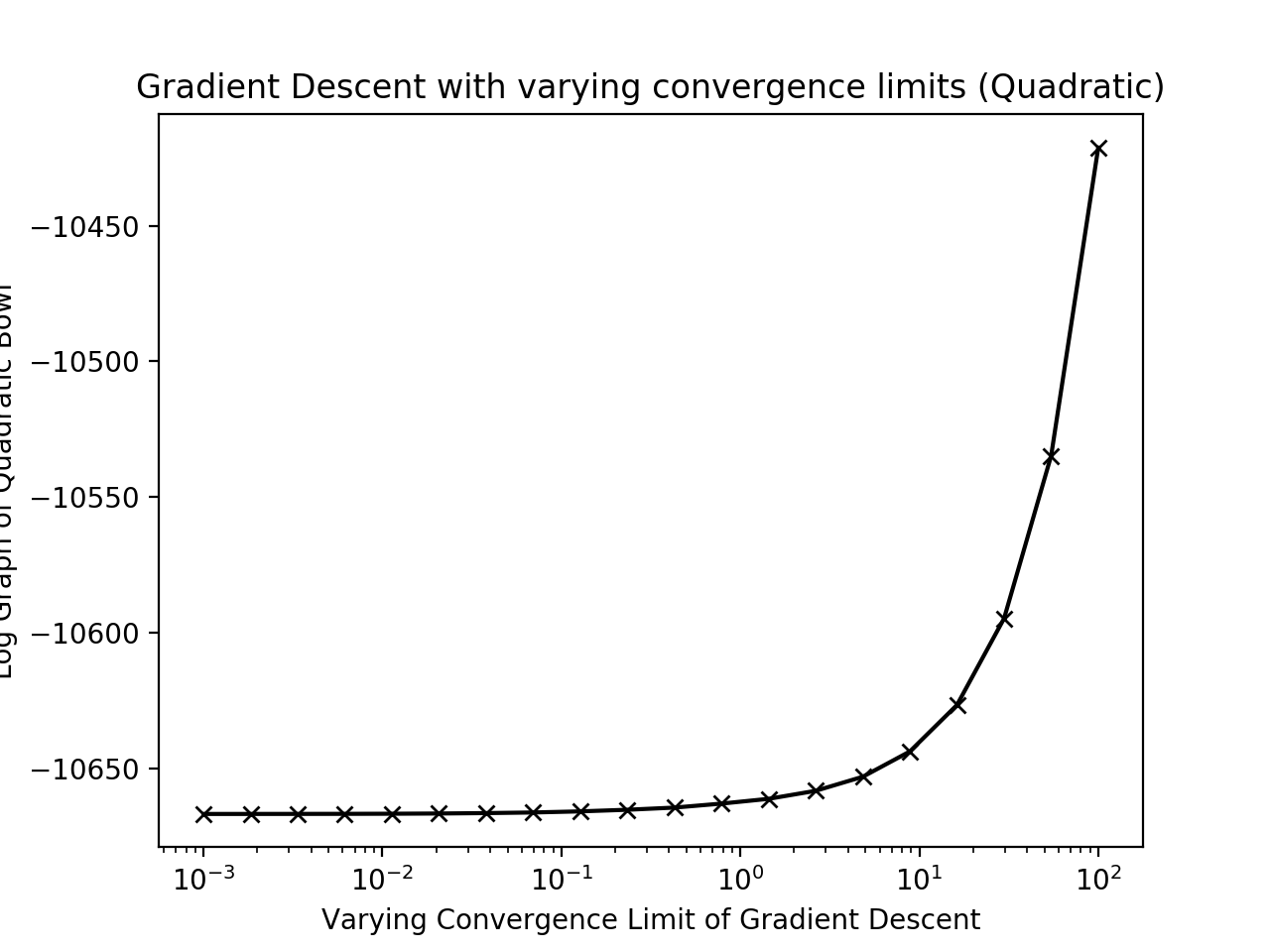
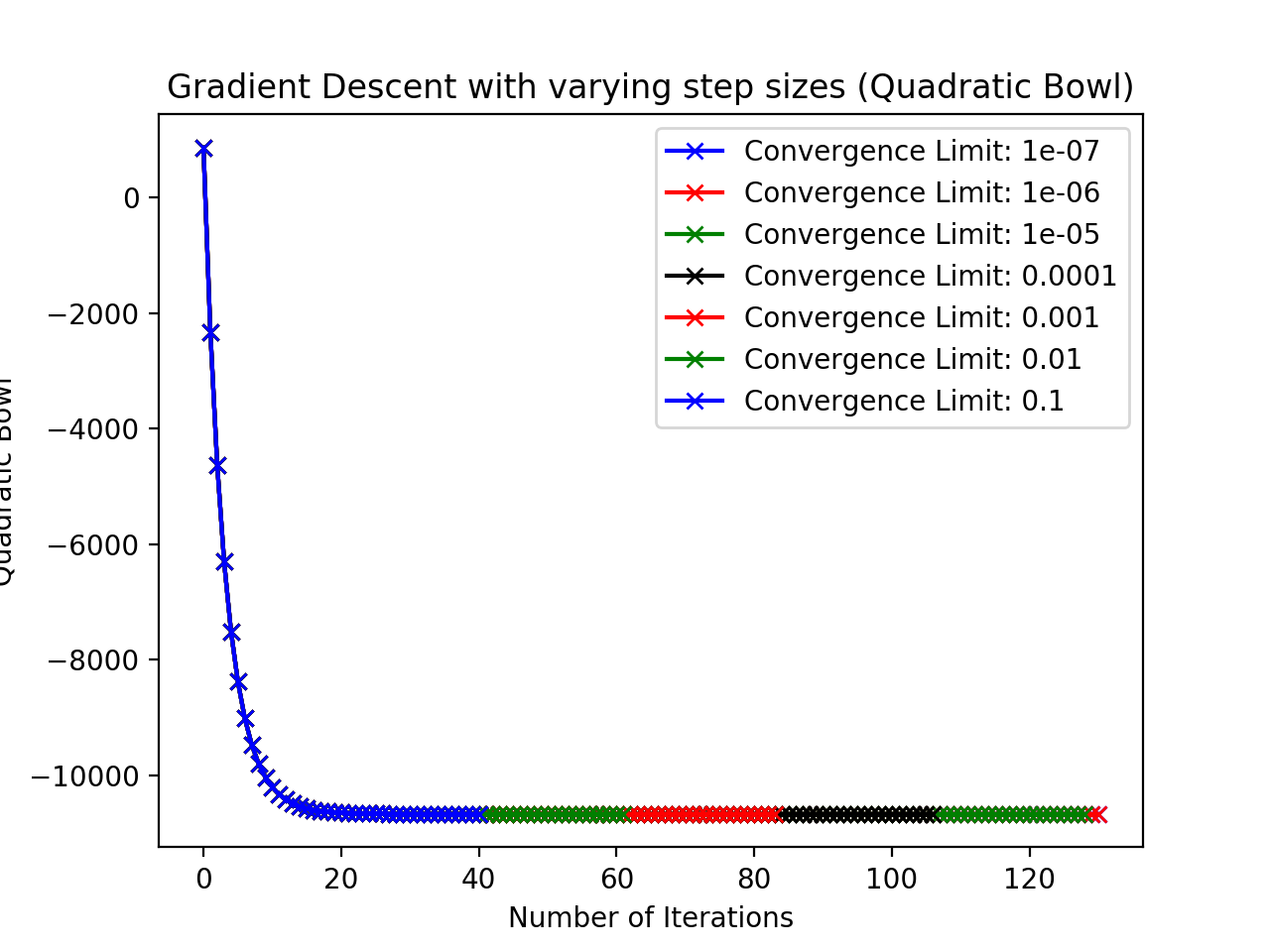
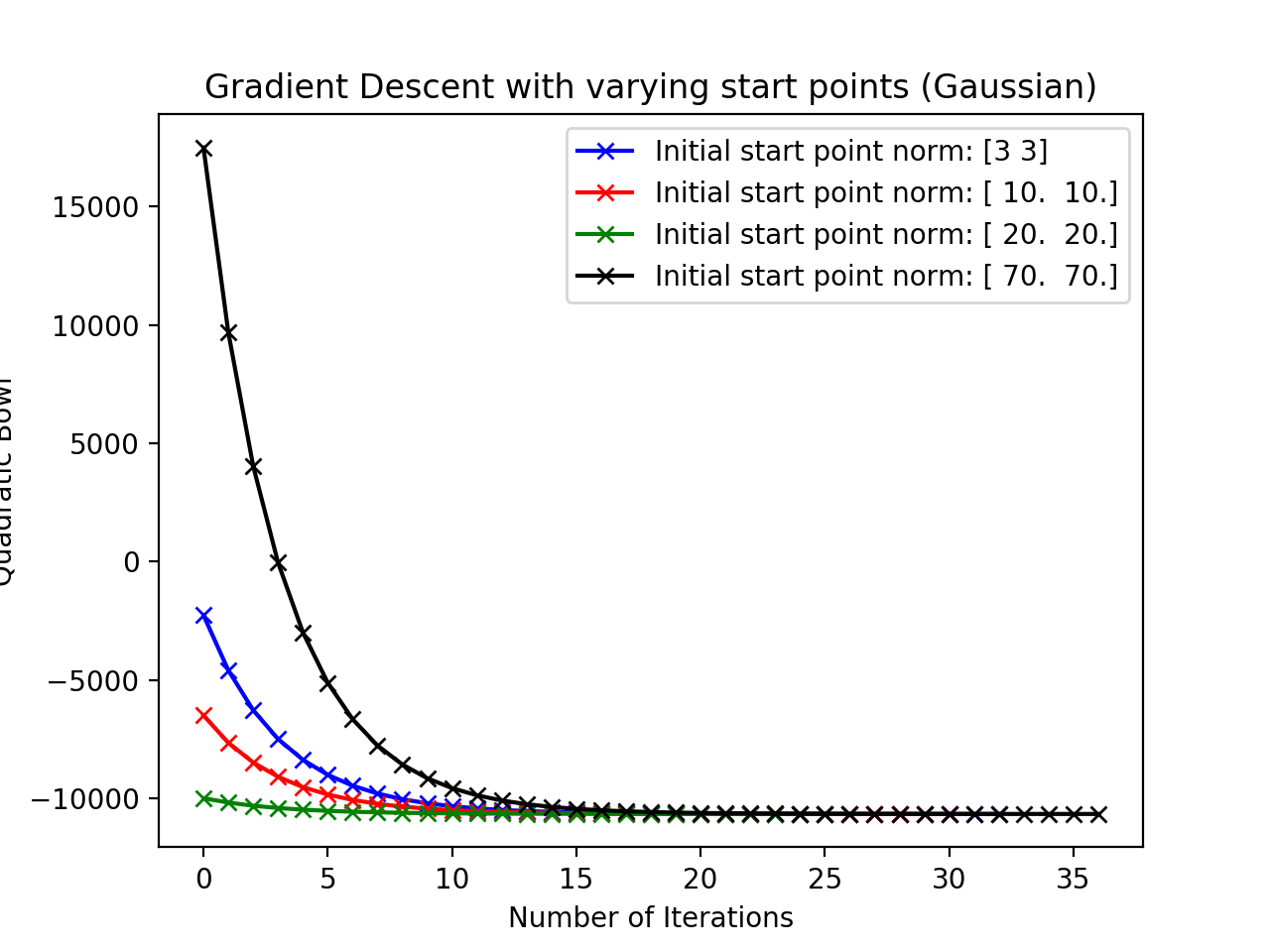
This program was tested on two functions:

1.A negative multivariate Gaussian function with mean (10, 10) and covariance matrix of :

2. And a quadratic bowl of:

Using the procedure described above, we found that the minima for each of the functions are and respectively. First, we vary the starting point, to examine the Gradient Descent’s effect on . For GD on , we set the as (2, 3), (4,5) and (6,7) respectively (Figure X). We find that the number of iterations to convergence becomes lower when the set points are closer to the real minimum. We also found that for some start points furthest away from the minima, convergence is not met even after 10000 iterations. This happens when is smaller than the convergence threshold. GD on (Figure X), on the other hand, shows that asis closer to the minima, the number of iterations it will take to reach convergence will be lower. This is due to the fact that as is set closer to the minima, the slope of the gradient will be less steep, resulting in a much smaller step sizes. We can sufficiently conclude the accuracy of GD does not depend on , if and only if this optimization converges.

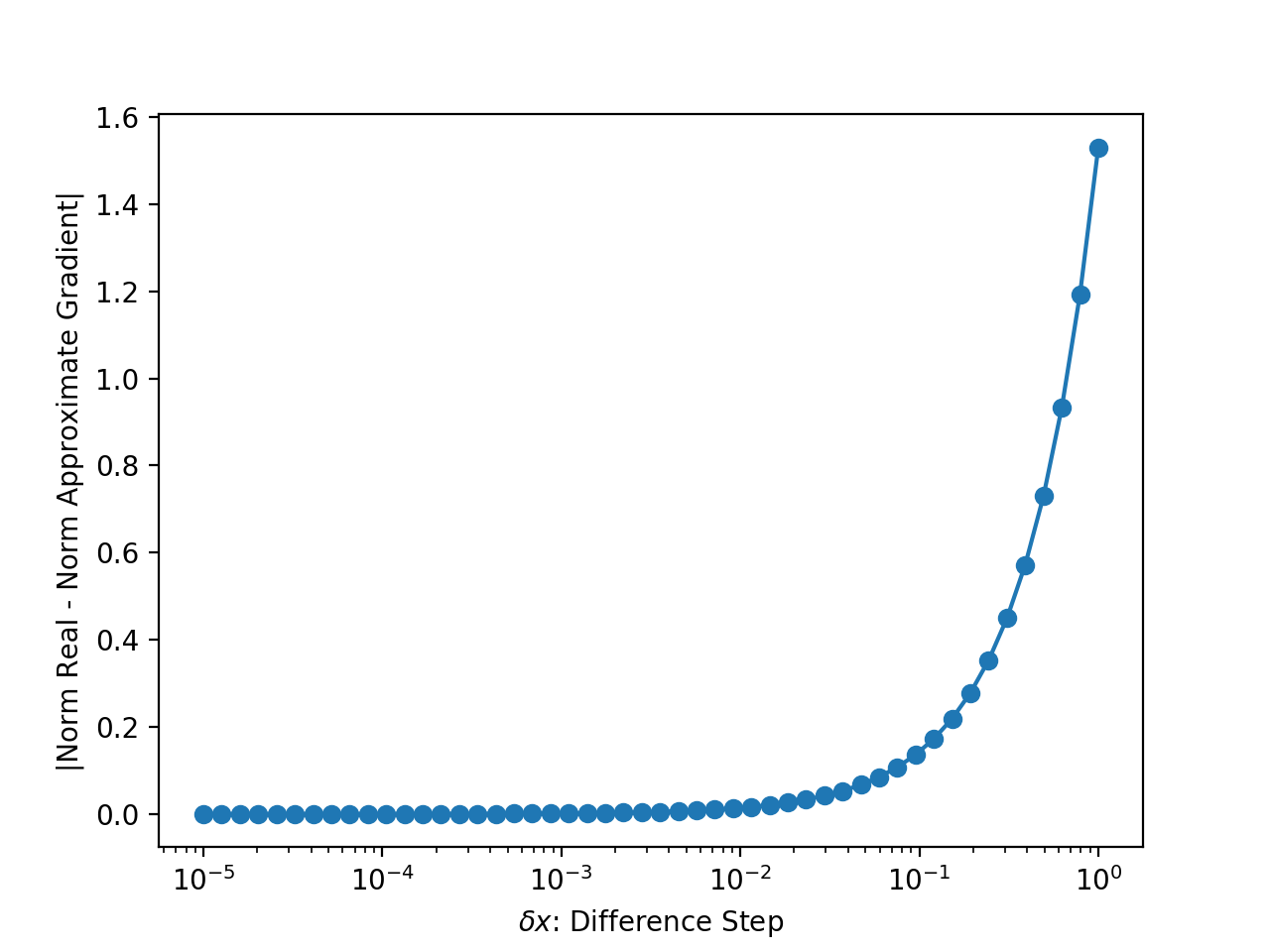




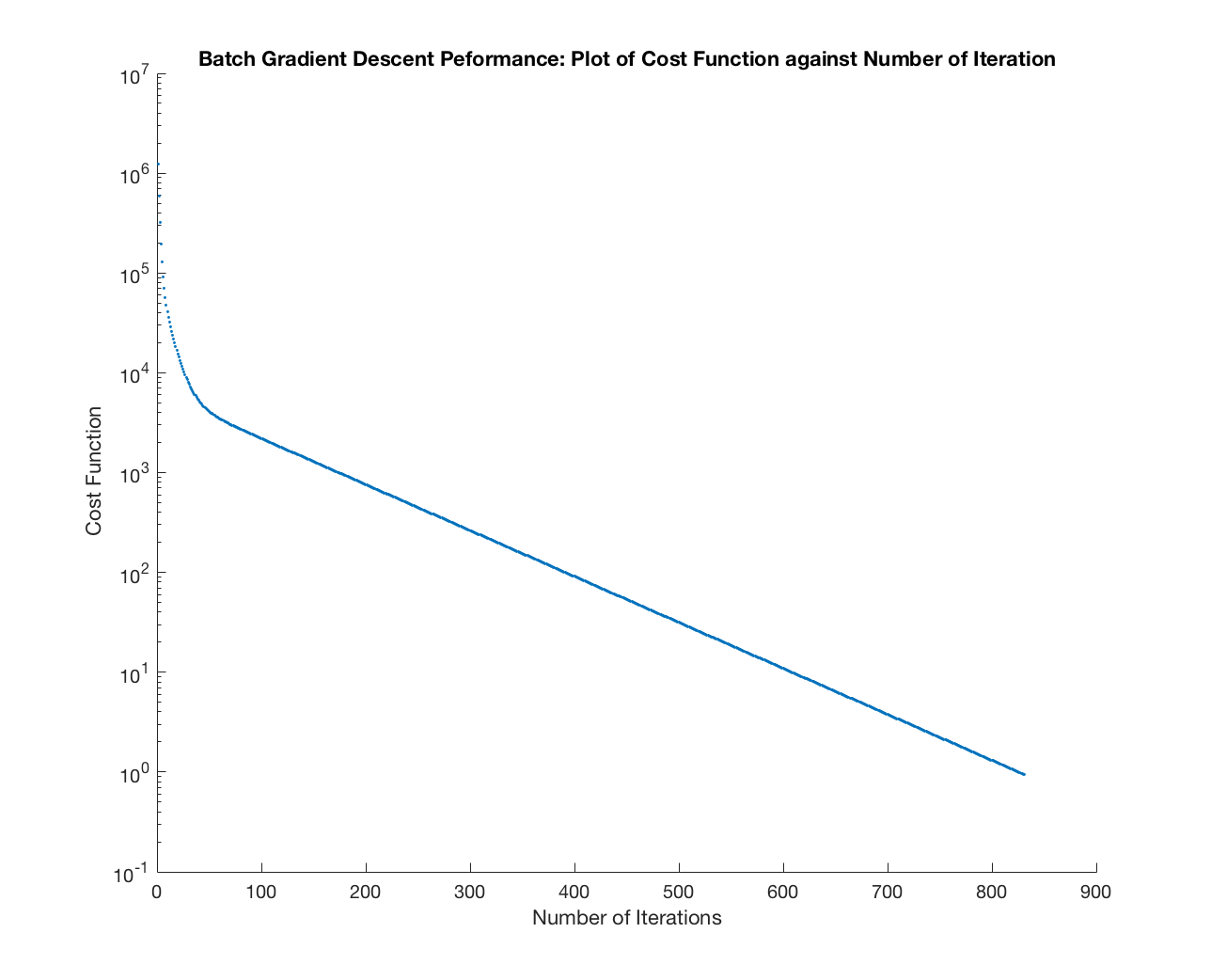
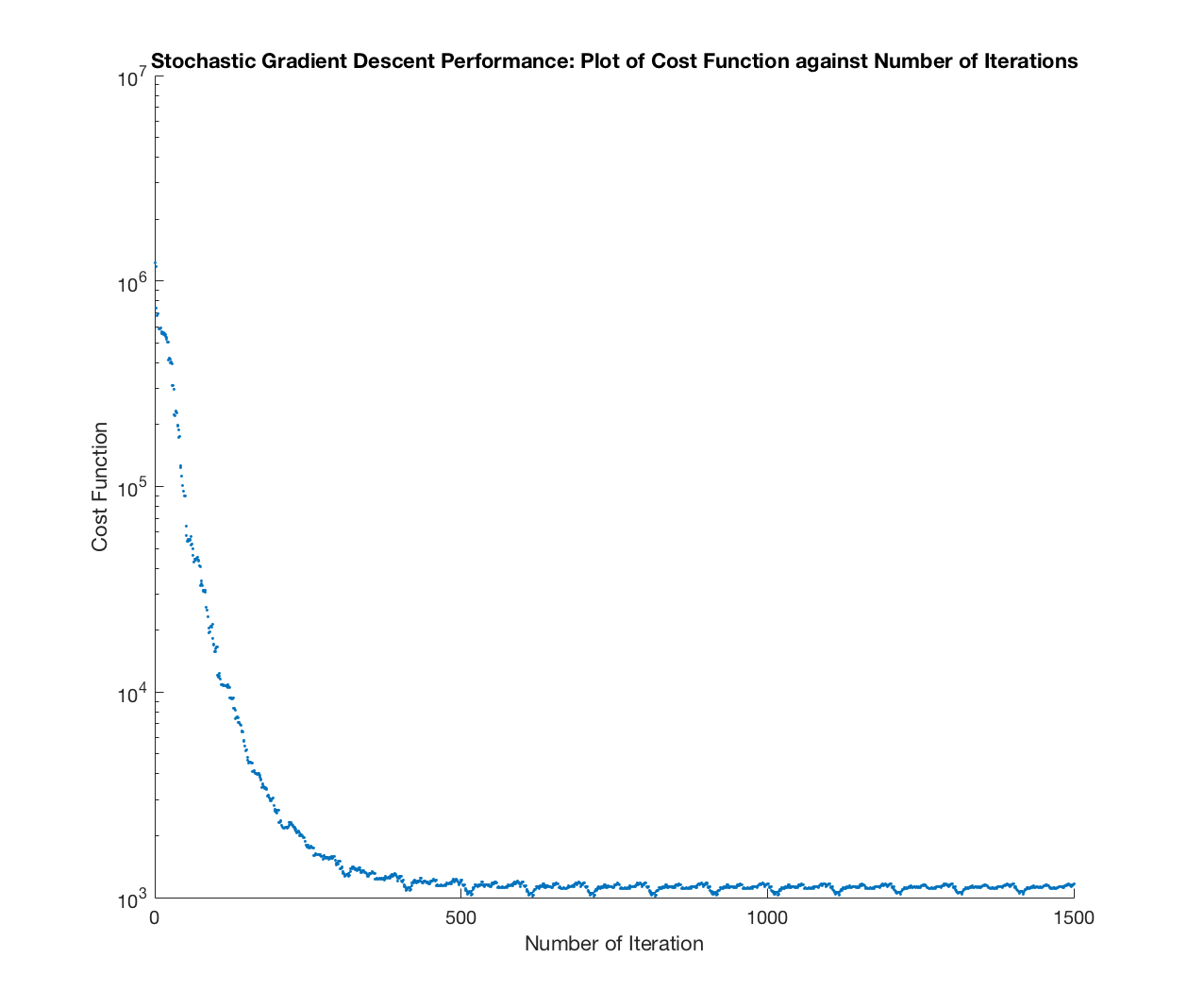
Next, we vary the step sizes while keeping the other parameters constant (Figures x and x). The application of GD illustrates a similar trend for and As the convergence limit becomes larger, the number of iterations it takes to reach the minima decreases. Similar to the first test, the accuracy of GD does not depend on the step size.

Lastly, the convergence limits are varied (Figure x and x). Since the convergence limit acts as thresholds that dictates to the algorithm when the minima has been found, we found that as the limit increases, less iterations it would take to reach a result. Unlike the first two parameters, the convergence limit determines how accurate GD will be.

We verify the gradient values on the functions we used in this problem by inspecting the norm of the error between the closed-loop form and numerical gradient at randomly-assigned points for at . As expected, as increases, the difference between the real and approximate gradient increases.(Figure XX)



Lastly, both stochastic (SD) and batch gradient (BD) descent were applied, obtaining Figures 3a and b. Knowing that each BD iteration takes about 100 times SD iteration, BD becomes considerably slow at 831 iterations, when compared to SD, which took 1500 iterations to complete.



**Q4 Sparsity and Lasso**

For this problem, we will use LASSO to estimate the sparse weights through the minimization of the formula:

Insert Lasso minimizer formula

Lasso has a property such that if the regularization coefficient, , which controls the relative importance of the data dependent error and the regularization term, is sufficiently large, then some of the coefficients of the weights will be driven to zero, resulting in a sparse model. In this case, the corresponding basis functions do not play a role. We expect LASSO to outperform ridge regression for better sparse estimation.

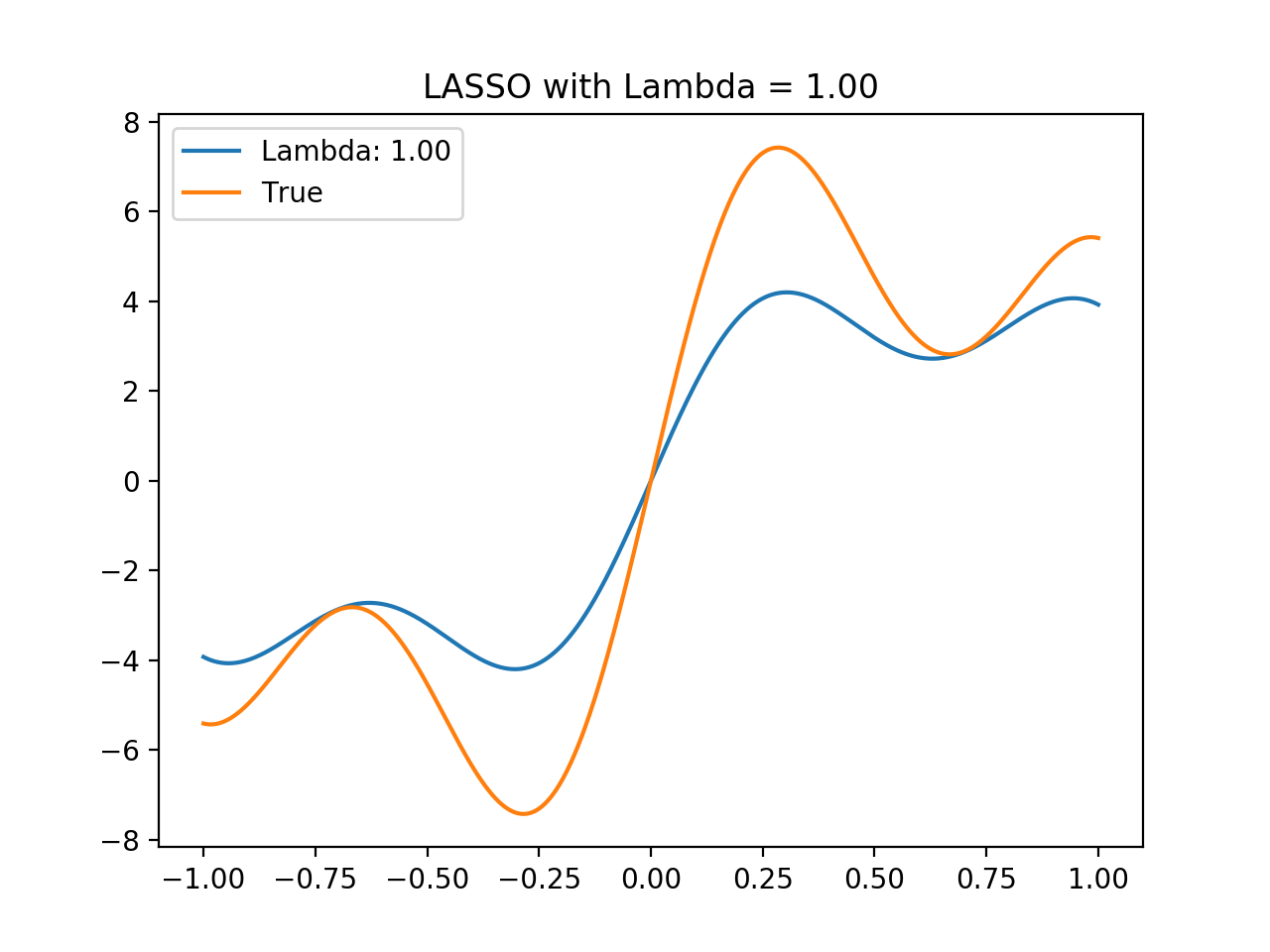
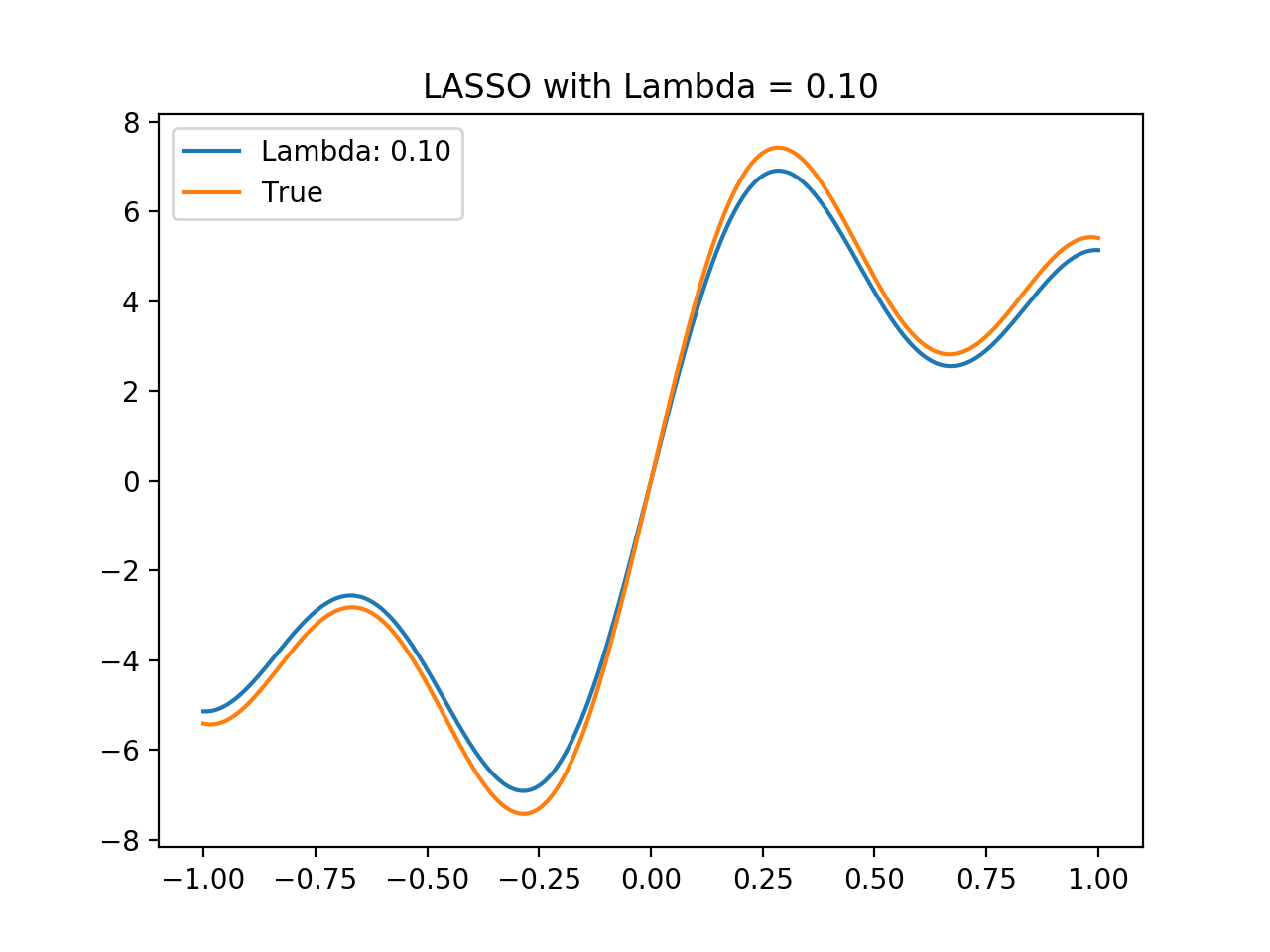
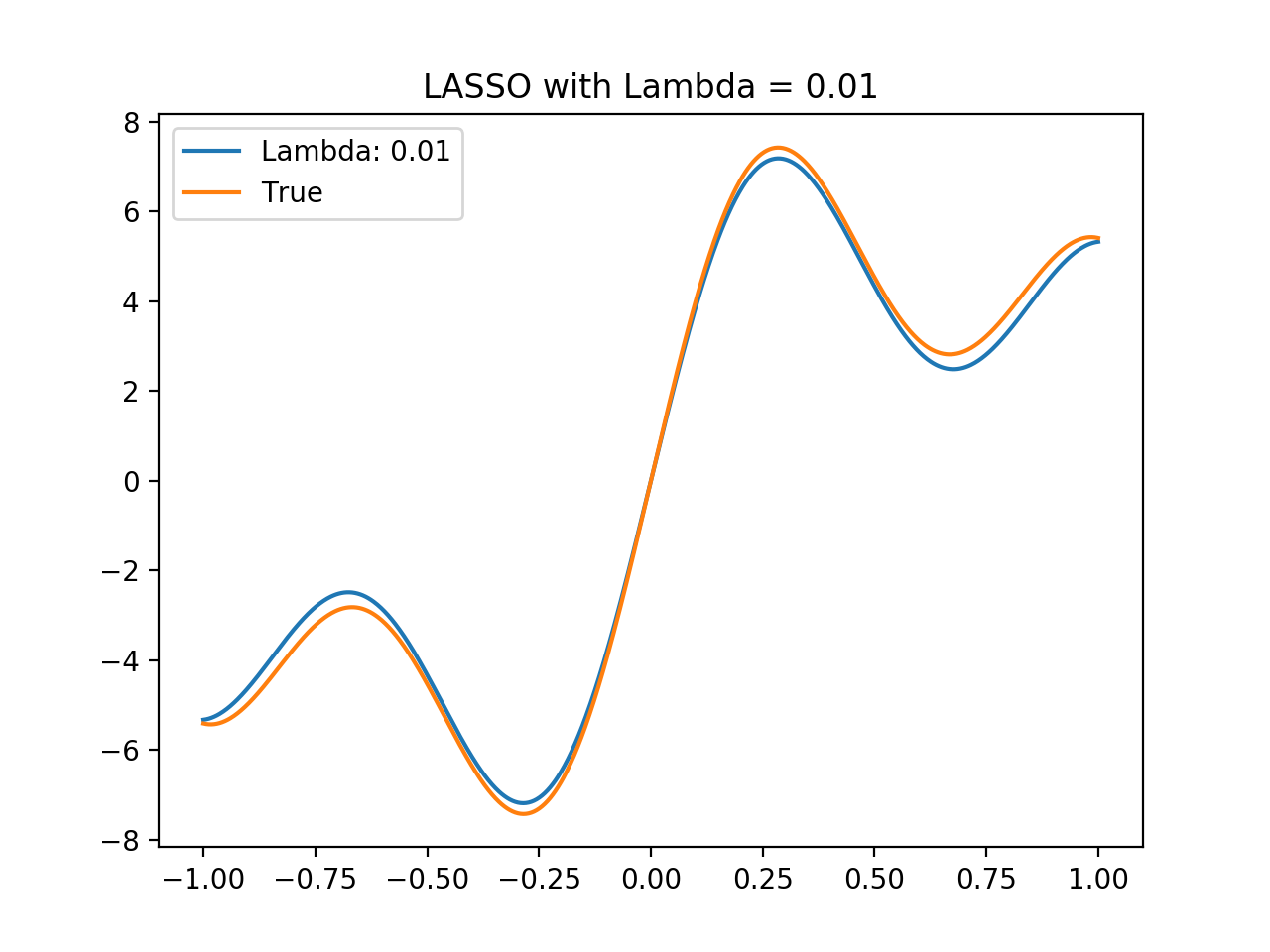
To prove this, we apply LASSO on the dataset where:

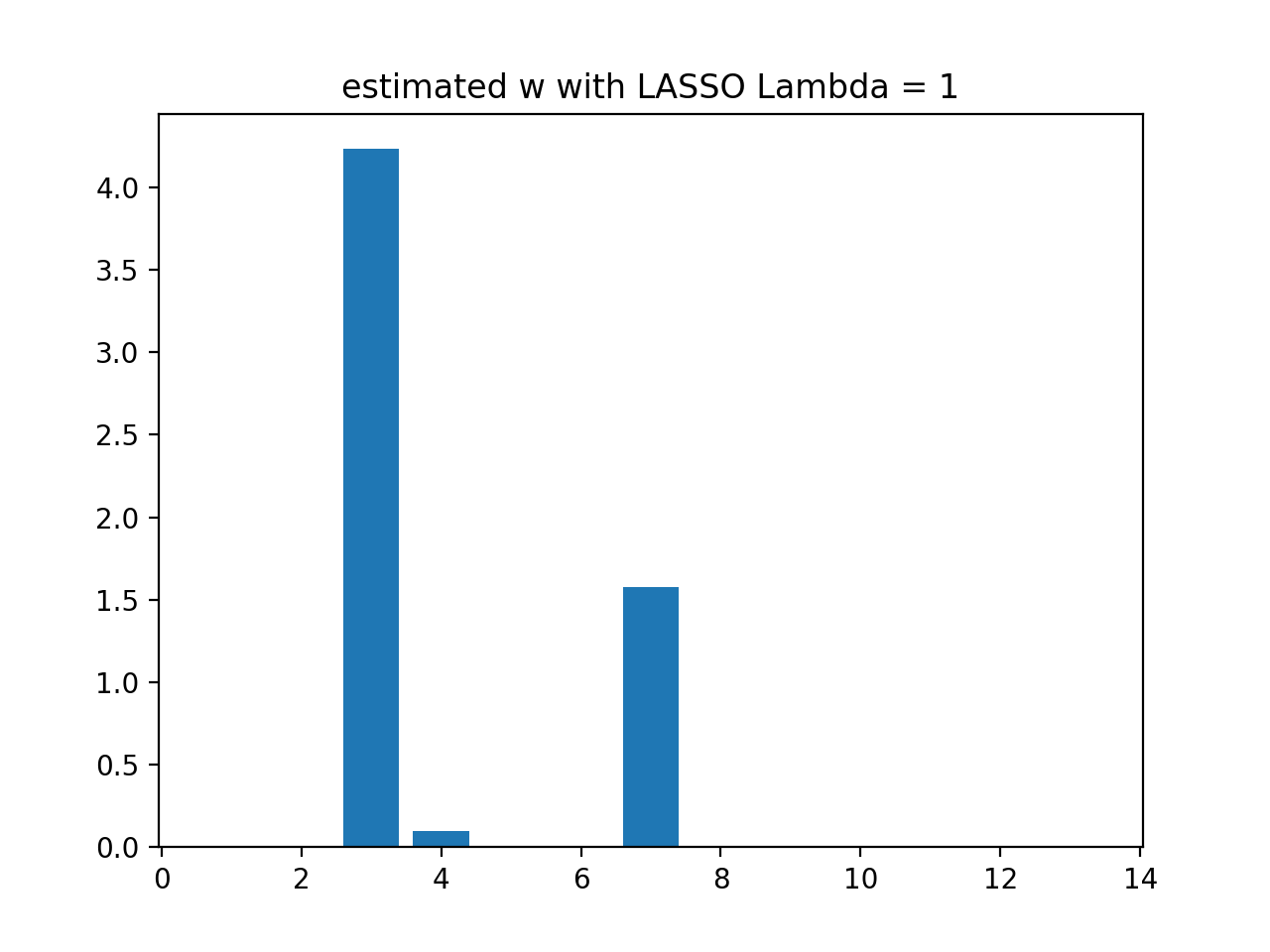
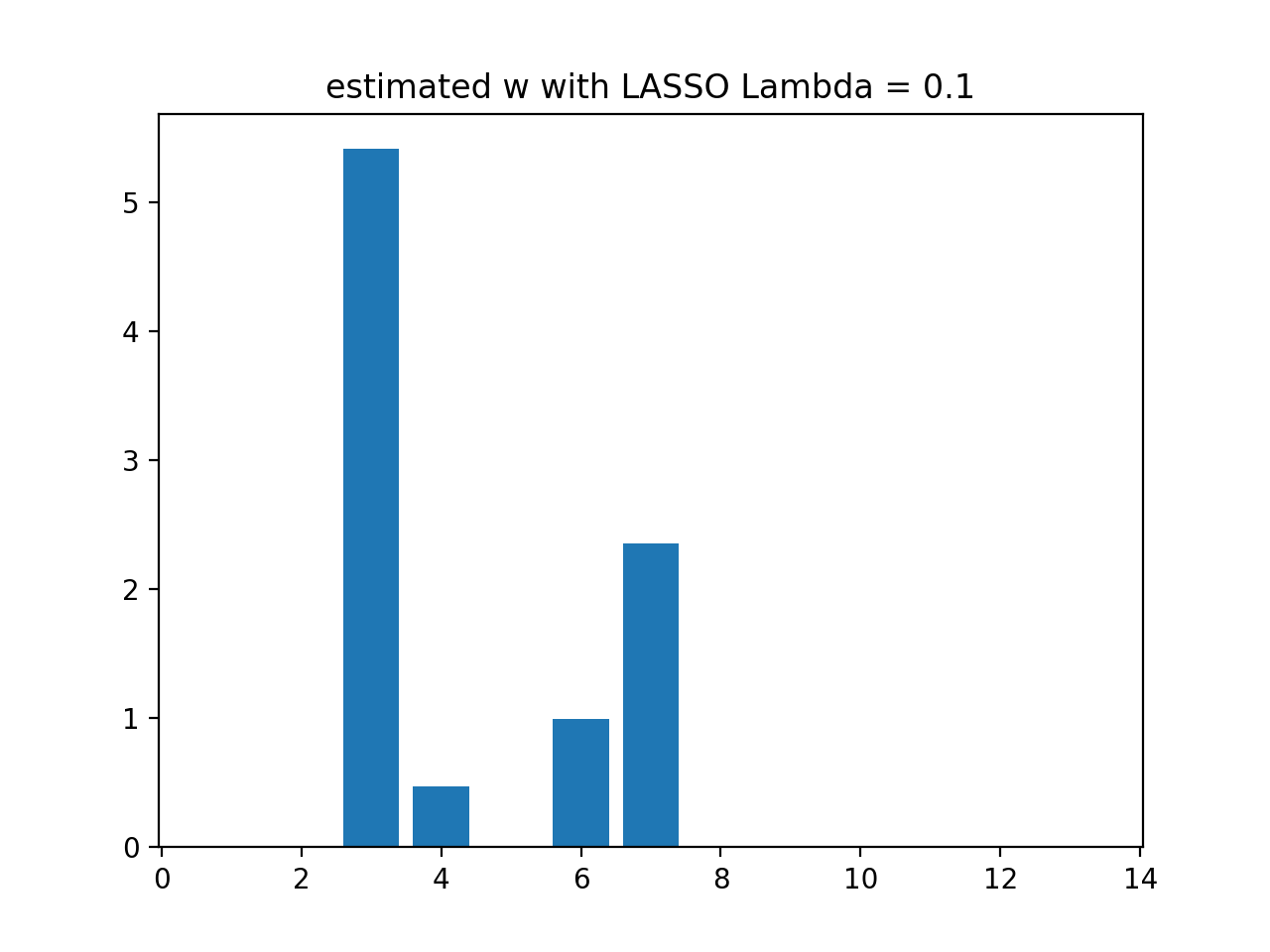
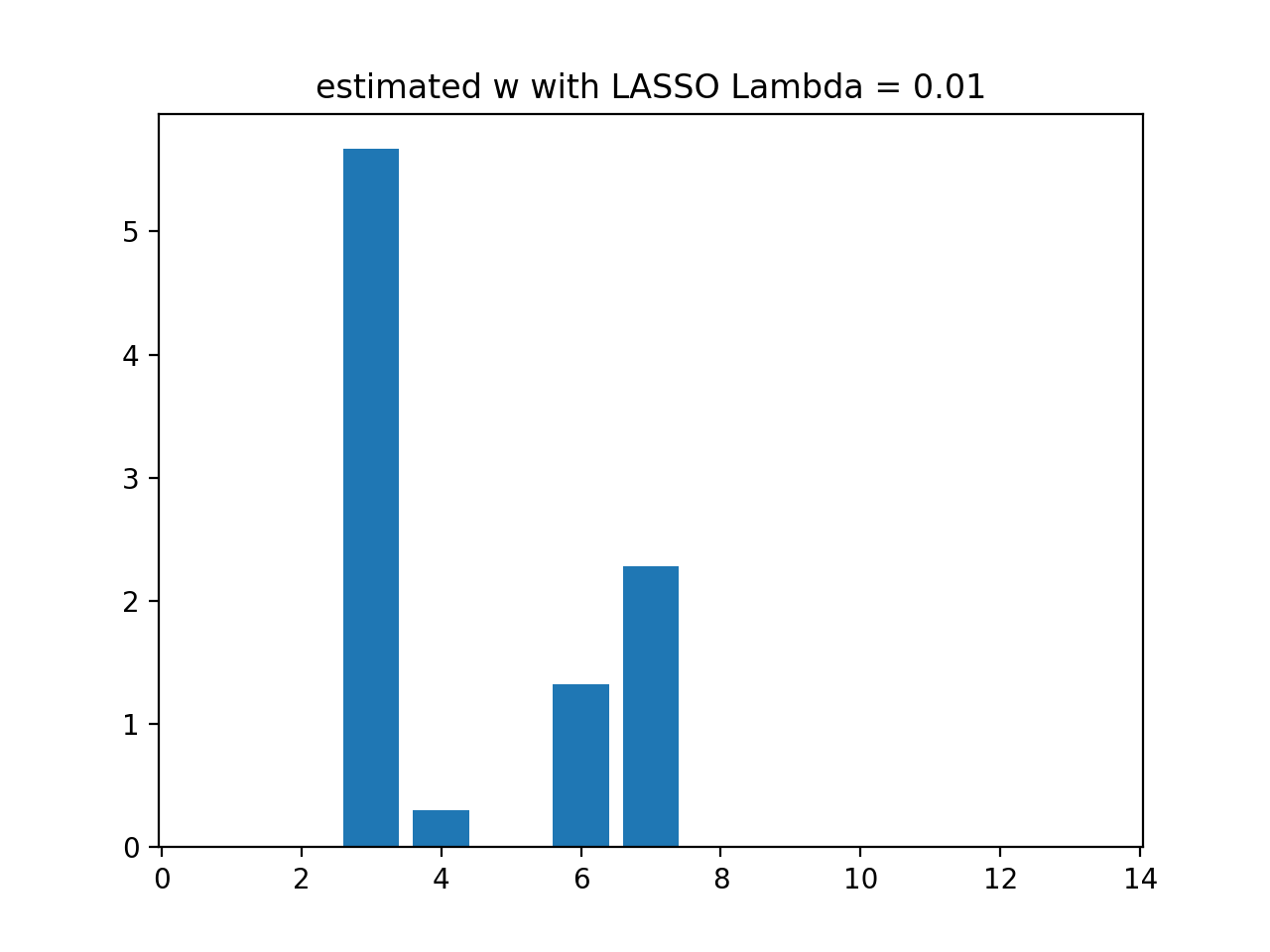
Insert equation question 4 = wT true

With the feature vector as:

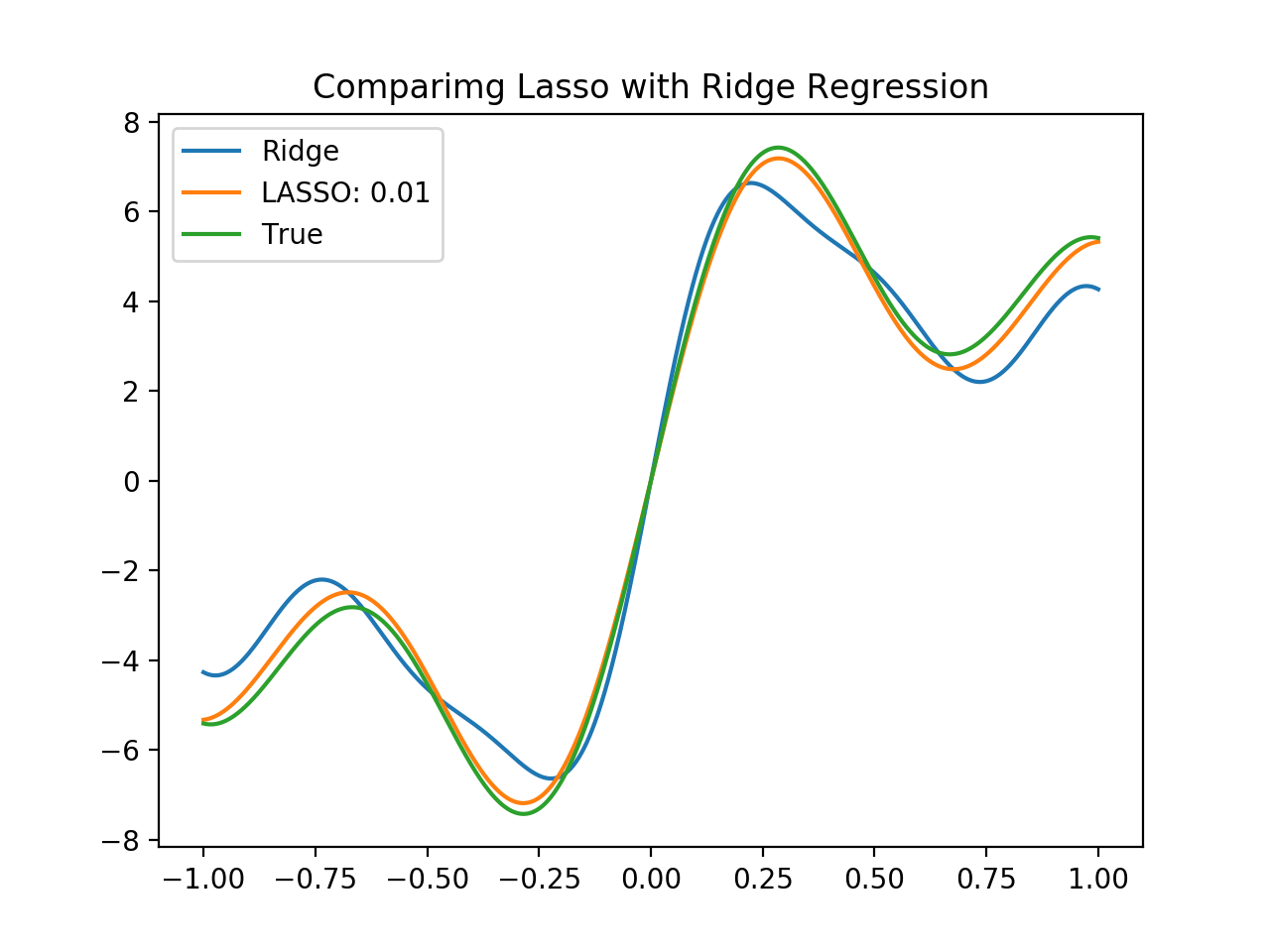
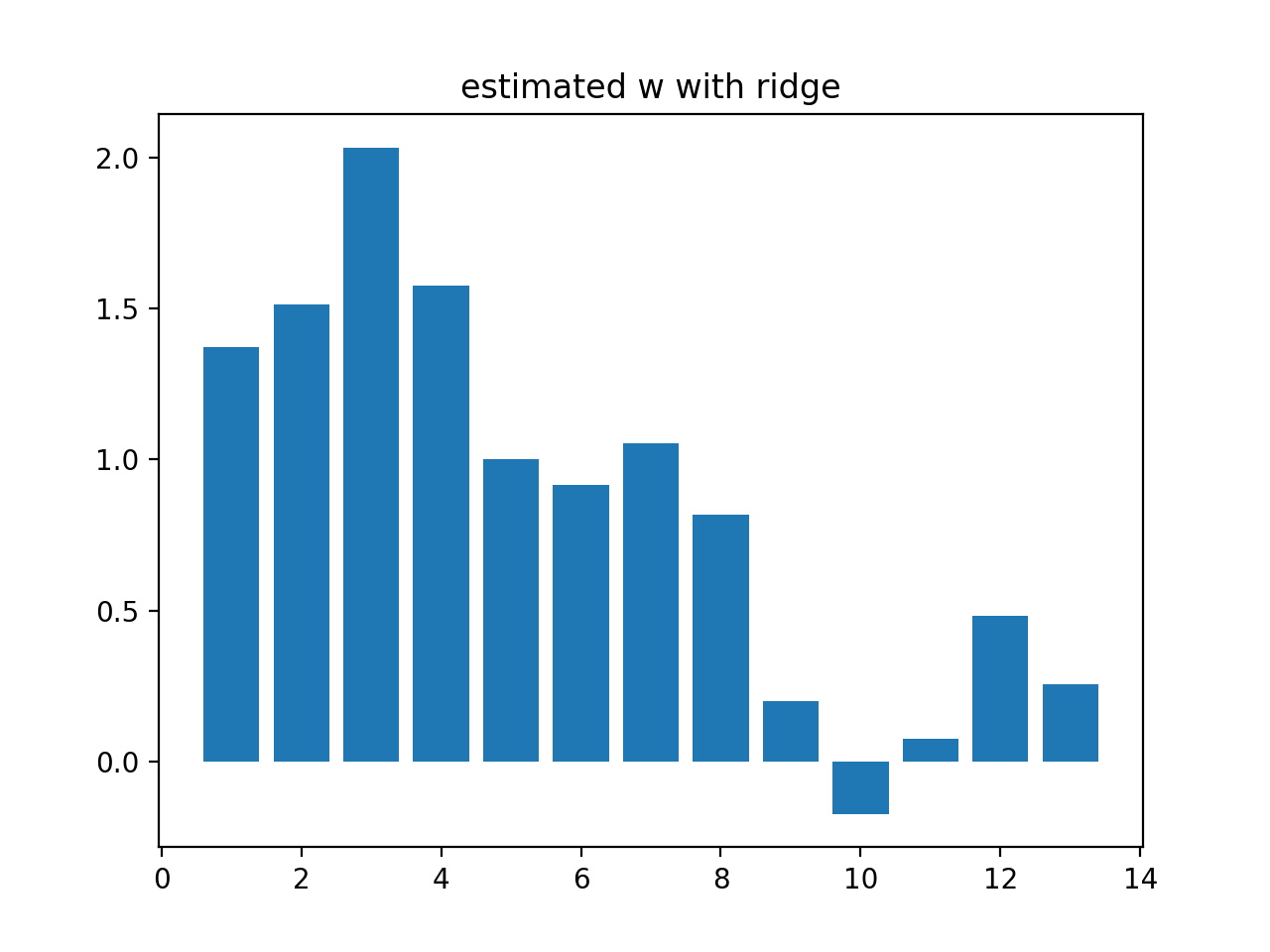
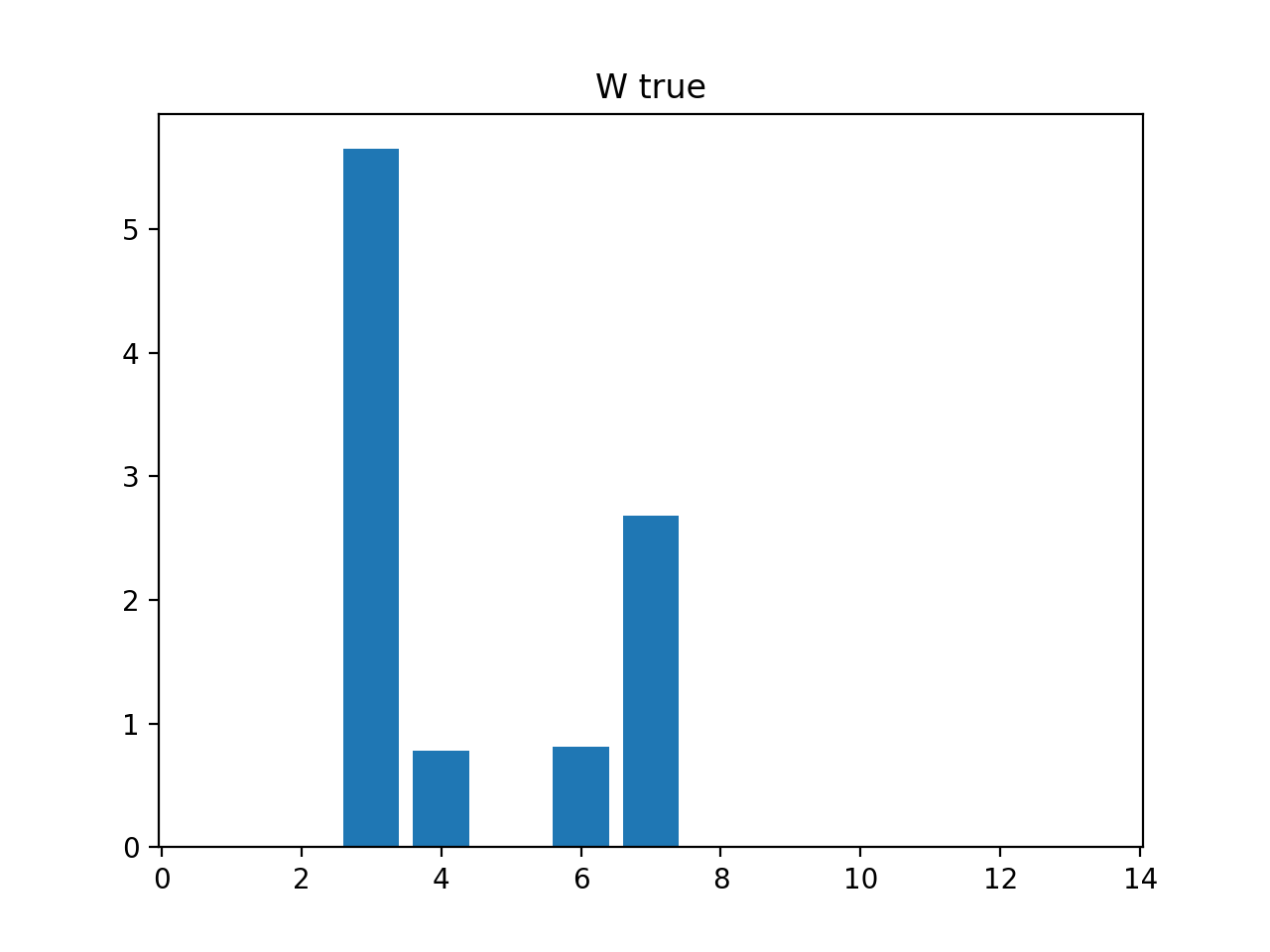
Insert phi equation

Using sklearn, we applied LASSO with the following [0.01, 0.1, 1, 10]. When is sufficiently small, there are still nonzero terms because the norm term, [ℓ](https://en.wikipedia.org/wiki/%E2%84%93" \o "ℓ)1 , is weighted less by it. Thus, as increases from 0.01 to 0.1, we see that the model becomes more sparse and closer to the true weight.





On the other hand, as becomes larger than 1, the resulting weights are smaller, but still sparse. This is due to , [ℓ](https://en.wikipedia.org/wiki/%E2%84%93)1 dominating the objective model and therefore, restricting the resulting model. Plotting the components of resulting weight vectors at different we can assume that the best estimate is when is set at 0.01.



We see that LASSO produces a better estimate of the true curve when compared to ridge regression.