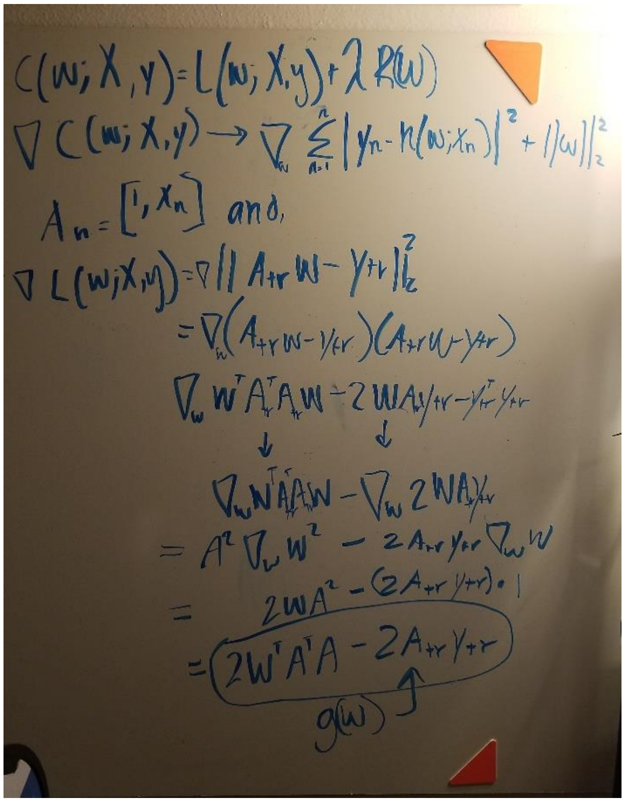
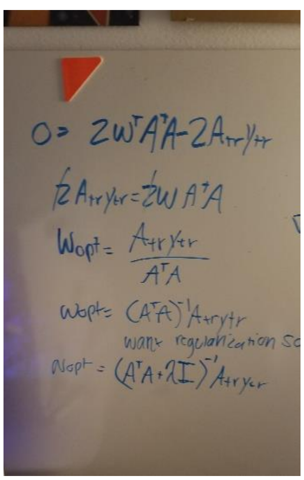
AI EXP : HW3 : Linear Regression
Alex Lamarche

Problem 1: Setting up the functions

1.A - Scripts included or check the GitHub link here : https://github.com/ajl3545/AI_HW3
1.B – Derivation of g(w) the gradient function



1.C Derivation of g(Wopt) the gradient function whose value is: g(w) = 0



1.D – Gradient Algorithm presentation and effects of α and τ:

```
def GD_SOLVER(X, y, p, l, step):

    parameters = []; parameters.append(p)
    costs = []

    i = 1

    while (True):

        # Most recent W
        w = parameters[-1]

        (mtr) = REG_MET(X, y, w, l)
        costs.append(mtr)

        g = gradient(X,w,y,l)

        # Terminate
        if (np.linalg.norm(g)**2 < pow(10,-8)):
            print("W len = " + str(len(parameters)))
            print("iters len = " + str(i))
            return (parameters,costs)

        # Descend gradient
        parameters.append(w - step*g)

        i+=1

def gradient(X,w,y,l):
    A = []
    for row in X:
        A.append(np.append(1,row))
    A_tr = np.array(A)
    A_trans = np.transpose(A_tr)

    d = np.dot(A_trans,A_tr)
    ident = l*np.identity(len(d))

    return (2 * np.dot(d+ident,w)) - (2 * np.dot(A_trans,y))
```

Discussion – α represents the ***step*** that must be taken towards the minimal value of the computed gradient. Each of the w values changes with the gradient descent iterations - towards an optimal set of parameters. By subtracting the previous set of w parameters by the current gradient "direction," computed by gradient(W,x,y,z), the algorithm gets closer to a solution that minimizes the cost function. Step is a multiplier that is used to get closer to the minimum of the gradient function. Increasing the step could lead to overfitting and would overshoot on the gradient.

τ, a threshold per se, is used to compute how close the parameters get us to an optimal regression line. First, we step towards the minimum, then we check how close we are. τ ensures that we are close to the minimal value. If the summation of the squares of the parameters (the cost at that parameter iteration) is lower than τ, then that means the most recent w parameters have- as closely as possible and within reason – calculated a minimal cost.

ignore the print statements

Problem 2 – Linear Regression (true function is affine)

2.A https://github.com/ajl3545/AI_HW3/blob/main/problem2.py or check out included file

2.B – Present the results of CF_SOLVER on training and testing data and show MSE for each:

Training data:

```
Wopt =
[0.9999999999999989, 2.0000000000000018, 2.999999999999999, 4.0, 4.999999999999999, 5.0, 3.999999999999999, 2.9999999999999996, 1.9999999999999998, 1.0000000000000022]
Opt Cost (mtr) = 3.3695454009416875e-28
Opt MSE = 3.7439393343796526e-29
```

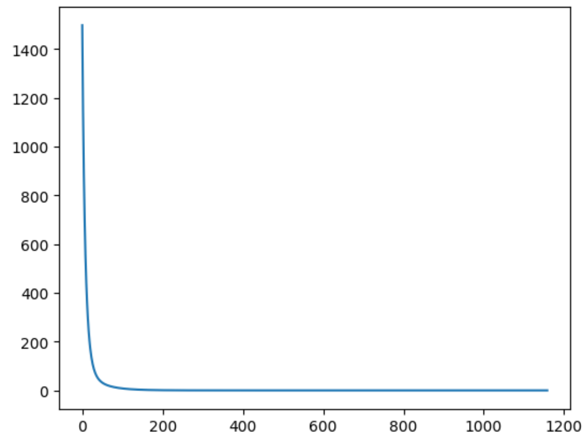
Testing data:

```
Wopt =
[1.0, 1.9999999999999996, 3.0000000000000004, 3.999999999999999, 4.999999999999997, 4.9999999999999964, 3.9999999999999987, 2.9999999999999996, 2.0, 1.0000000000000009]
Opt Cost (mtr) = 2.9228405874086454e-27
Opt MSE = 3.247600652676273e-28
```

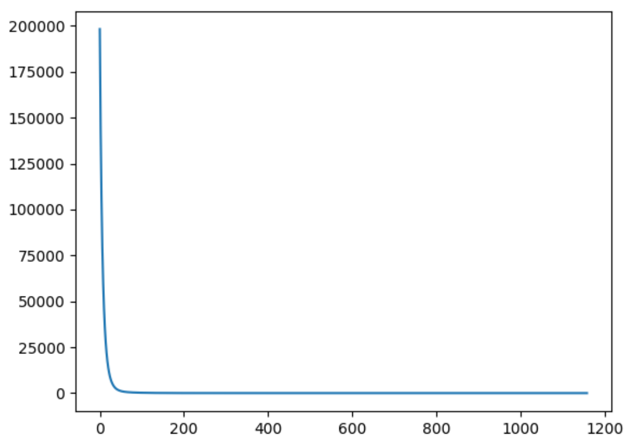
The Results: barely a difference, with the Wopt values nearly being the same. The testing data however, had more data and took less time to run. The training data took nearly 10x more iterations to come up with a result.

(cont.)

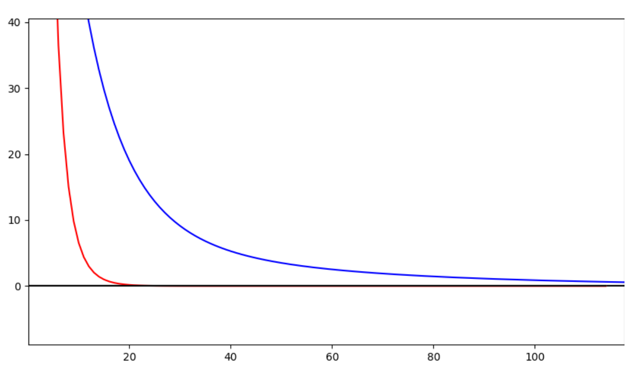
2.C – **Figure 1**. Regression objective(y) versus GD iteration (x). As the iteration progresses along the x axis, the cost metric reduces – which is expected since we want the cost to be minimal.



2.D – **Figure 2.** Euclidean norm (y) versus the iteration (x). Total # of iterations = 1158 on training data:

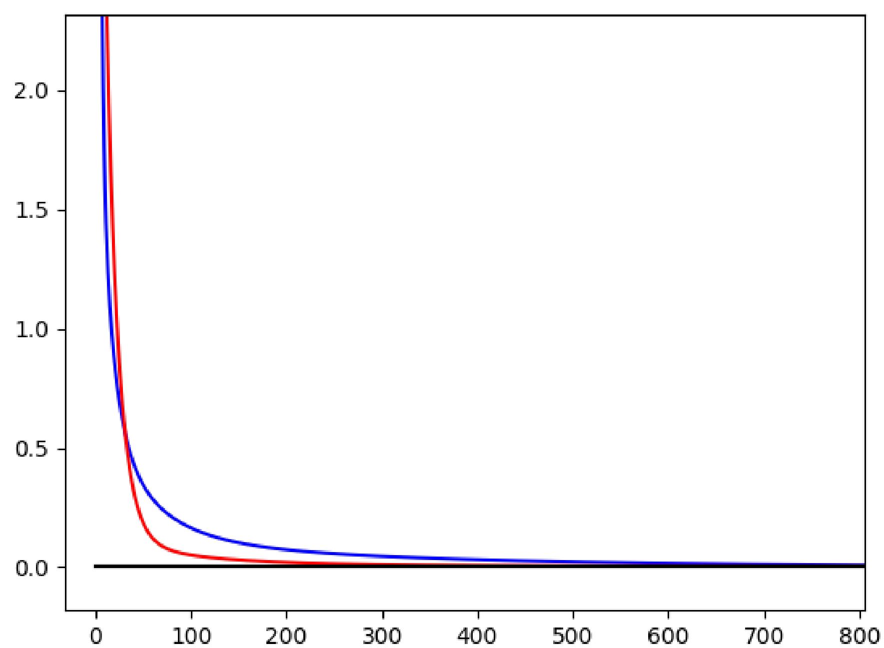


2.E – **Figure 3.** The metric lines (black) for both testing and training overlap since they are so small. Plotted are the MSE values for training (blue), testing, (red), and the benchmark MSE values (black) calculated by CF_SOLVER. Both MSE's converge close enough to the benchmarks to consider calculation successful.



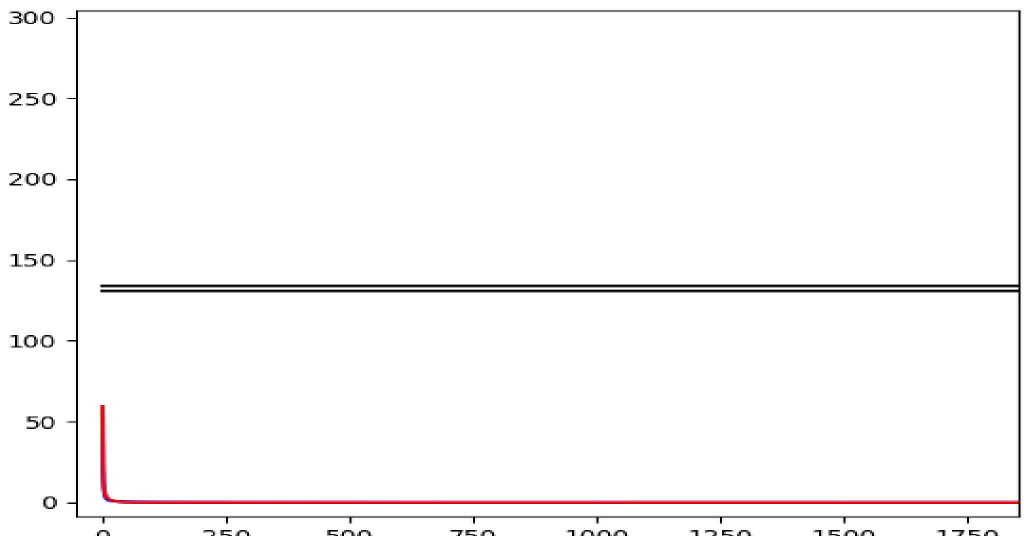
2.F – The benchmarks were meaningful since they predict a successful linear regression. Since the benchmark values aim to be at zero, the MSE lines must align with the benchmark as the iterations grow along the X axis. The MSE's are both aligning with the benchmark.

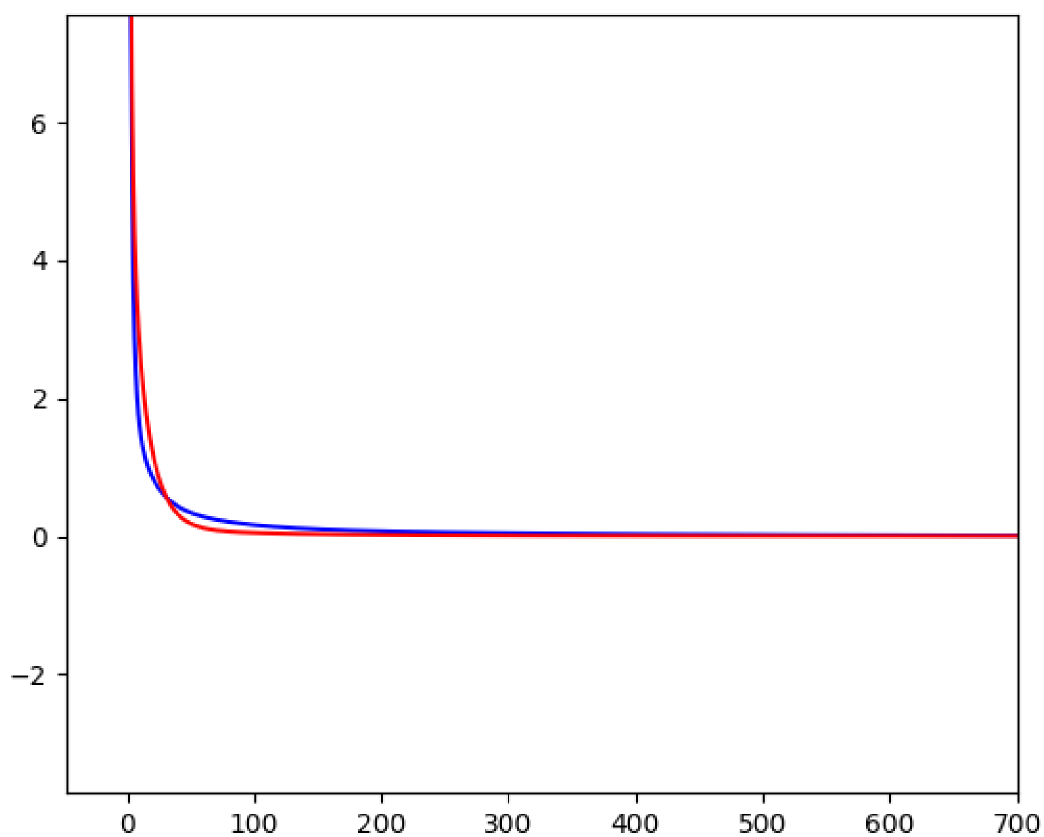
2.G – **Figure 4.** There isn't even 10 lines in the testing data to compare with… I still managed to run the plots. It seems that when the dataset is lower, more iterations are needed to converge to a minimum. Less data means more effort to calculate a relationship between variables. When there is more data, the linear relationship becomes more evident more quickly:



(cont.)

2.H – **Figure 5.** With lambda = 2. There were nearly 5k iterations on the training data and over 20k iterations on the testing data. I assume that with a large enough lambda, the data overfits or overshoots? The benchmark data is out of whack and doesn't represent a proper value at all. Affecting the regularization skews the accuracy:

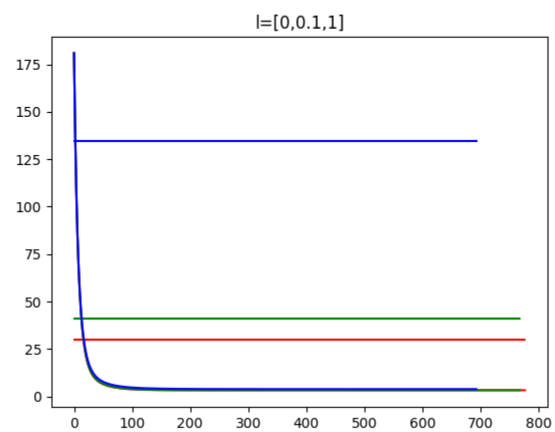


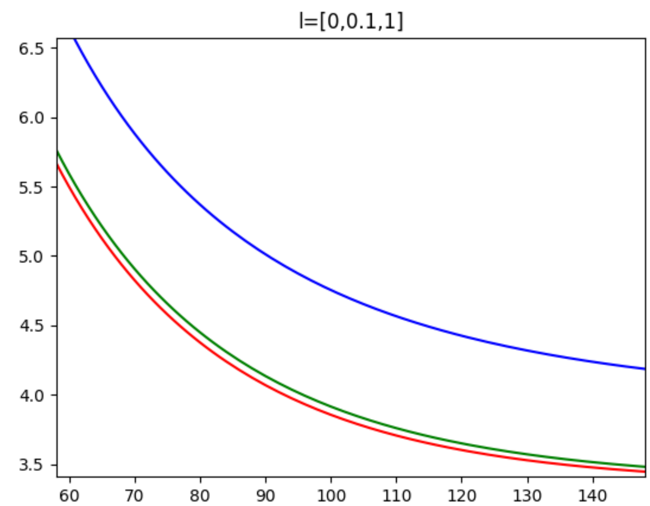


Problem 3 - Linear Regression (true function is affine + noise)

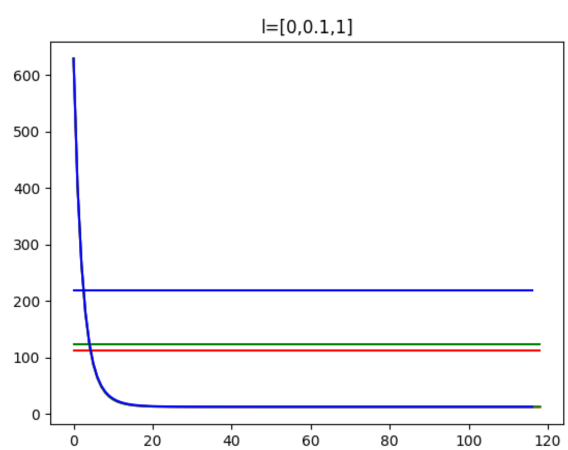
*All of the following graphs
Red: l=0
Green: l=0.1
Blue: l=1

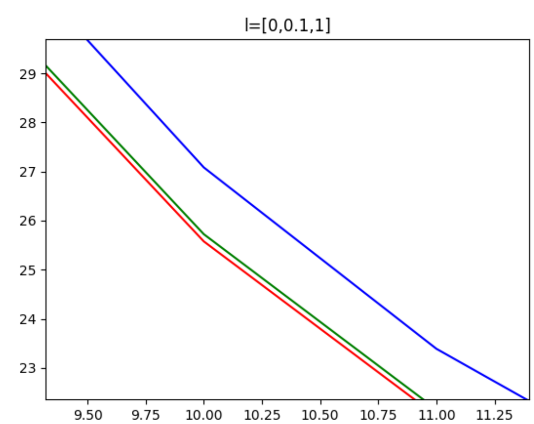
3.A – Check out the GitHub link : https://github.com/ajl3545/AI_HW3/blob/main/problem3.py or see attached files problem3.py.
3.B – **Figure 6-7** For l = [0,0.1,1] in the training data:



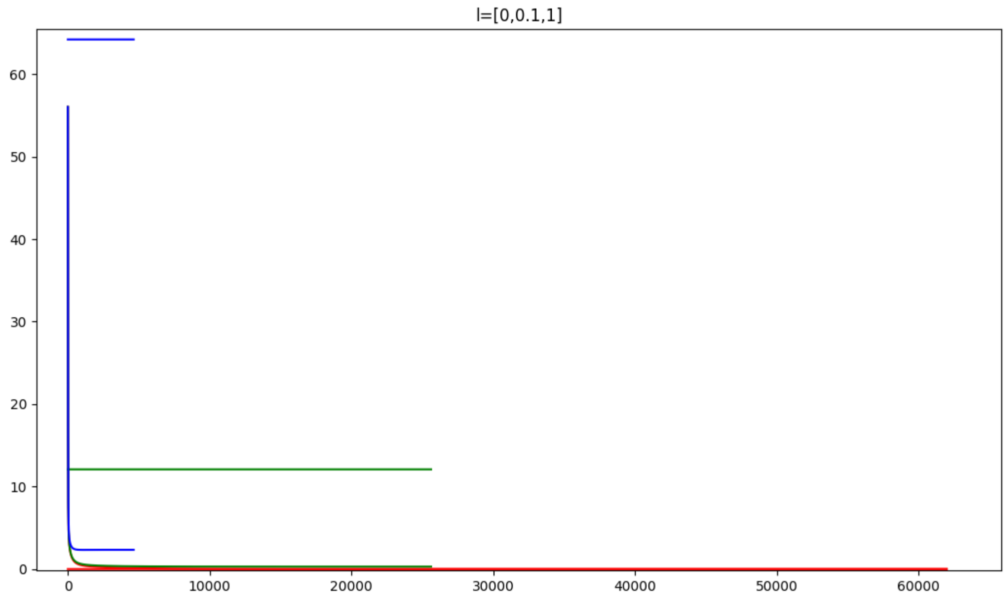


test:





3.C – **Figure 8.** For fewer data. Not sure I did this correctly



I don't know what the remaining questions for problem 3 are asking…

I tried my best. I still need to understand the concepts clearly. I can't even zoom into the graphs properly at this point.

DATA

The following is a MSE converging on 0 error over the course of several GD iterations given a random set of parameters and training data from clean data and lambda=0 and step = 0.01:

CF_SOLVER DATA START

```
Wopt =
[0.9999999999999989, 2.0000000000000018, 2.999999999999999, 4.0, 4.999999999999999,
5.0, 3.999999999999999, 2.9999999999999996, 1.9999999999999998, 1.0000000000000022]

Scientific notation. These values are nearly 0 (pretty much zero)
Opt Cost (mtr) = 3.3695454009416875e-28
Opt MSE = 3.7439393343796526e-29
```

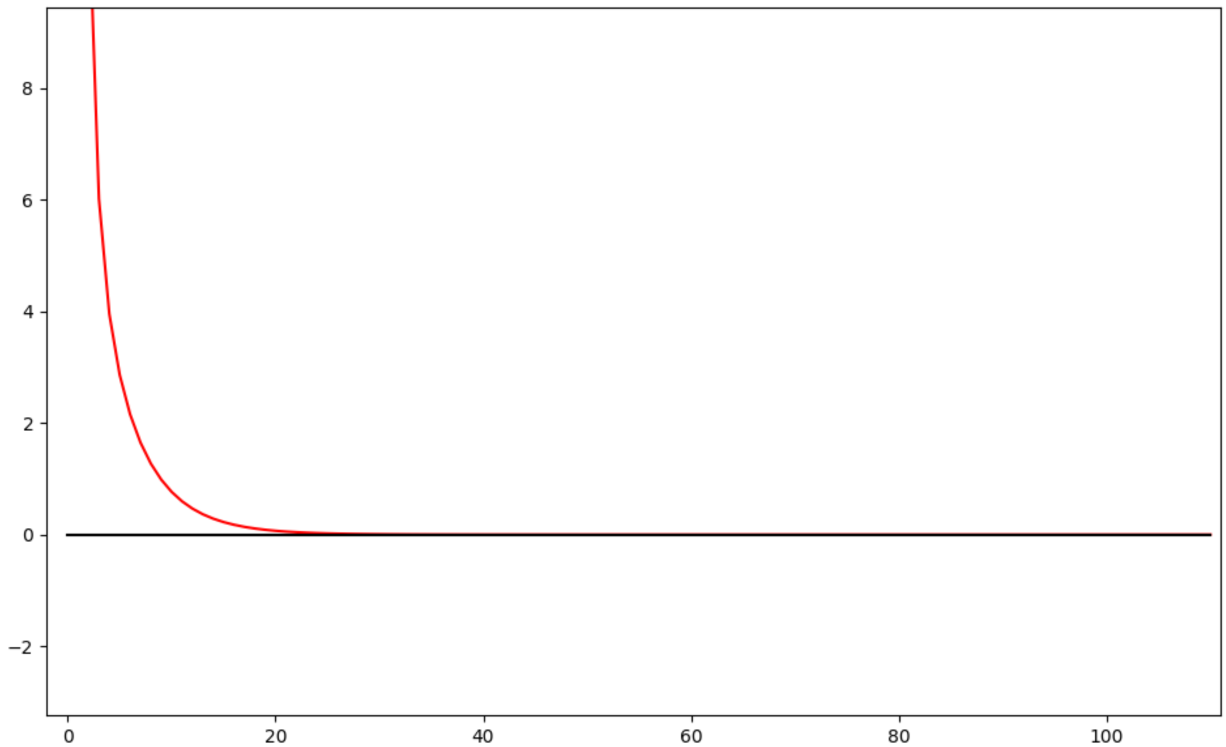
CF_SOLVER DATA END

```
GD_SOLVER DATA START
W len = 111
iters len = 111


MSE 0 = 166.4035357461
MSE 1 = 32.2847049312
MSE 2 = 11.4500481246
MSE 3 = 6.0198087551
MSE 4 = 3.9534427416
MSE 5 = 2.8626801898
MSE 6 = 2.1549692721
MSE 7 = 1.6486129376
MSE 8 = 1.2711225328
MSE 9 = 0.9846285416
MSE 10 = 0.7652286020
MSE 11 = 0.5962770496
MSE 12 = 0.4656485482
MSE 13 = 0.3643217362
MSE 14 = 0.2855073769
MSE 15 = 0.2240578514
MSE 16 = 0.1760480677
MSE 17 = 0.1384709846
MSE 18 = 0.1090134325
MSE 19 = 0.0858895519
MSE 20 = 0.0677161522
MSE 21 = 0.0534188664
MSE 22 = 0.0421610999
MSE 23 = 0.0332899662
MSE 24 = 0.0262949577
MSE 25 = 0.0207762204
MSE 26 = 0.0164201081
MSE 27 = 0.0129802817
MSE 28 = 0.0102630512
MSE 29 = 0.0081159775
MSE 30 = 0.0064189887
MSE 31 = 0.0050774403
MSE 32 = 0.0040166850
MSE 33 = 0.0031778171
MSE 34 = 0.0025143317
MSE 35 = 0.0019895003
MSE 36 = 0.0015743059
MSE 37 = 0.0012458175
MSE 38 = 0.0009859092
MSE 39 = 0.0007802506
MSE 40 = 0.0006175097
MSE 41 = 0.0004887245
MSE 42 = 0.0003868063
MSE 43 = 0.0003061475
MSE 44 = 0.0002423117
MSE 45 = 0.0001917891
MSE 46 = 0.0001518022
MSE 47 = 0.0001201534
MSE 48 = 0.0000951038
```

```
MSE 49 = 0.0000752771
MSE 50 = 0.0000595840
MSE 51 = 0.0000471628
MSE 52 = 0.0000373311
MSE 53 = 0.0000295490
MSE 54 = 0.0000233893
MSE 55 = 0.0000185137
MSE 56 = 0.0000146544
MSE 57 = 0.0000115997
MSE 58 = 0.0000091817
MSE 59 = 0.0000072678
MSE 60 = 0.0000057528
MSE 61 = 0.0000045537
MSE 62 = 0.0000036045
MSE 63 = 0.0000028531
MSE 64 = 0.0000022584
MSE 65 = 0.0000017876
MSE 66 = 0.0000014150
MSE 67 = 0.0000011201
MSE 68 = 0.0000008866
MSE 69 = 0.0000007018
MSE 70 = 0.0000005555
MSE 71 = 0.0000004397
MSE 72 = 0.0000003481
MSE 73 = 0.0000002755
MSE 74 = 0.0000002181
MSE 75 = 0.0000001726
MSE 76 = 0.0000001366
MSE 77 = 0.0000001082
MSE 78 = 0.0000000856
MSE 79 = 0.0000000678
MSE 80 = 0.0000000536
MSE 81 = 0.0000000425
MSE 82 = 0.0000000336
MSE 83 = 0.0000000266
MSE 84 = 0.0000000211
MSE 85 = 0.0000000167
MSE 86 = 0.0000000132
MSE 87 = 0.0000000104
MSE 88 = 0.0000000083
MSE 89 = 0.0000000065
MSE 90 = 0.0000000052
MSE 91 = 0.0000000041
MSE 92 = 0.0000000032
MSE 93 = 0.0000000026
MSE 94 = 0.0000000020
MSE 95 = 0.0000000016
MSE 96 = 0.0000000013
MSE 97 = 0.0000000010
MSE 98 = 0.0000000008
MSE 99 = 0.0000000006
MSE 100 = 0.0000000005
MSE 101 = 0.0000000004
MSE 102 = 0.0000000003
MSE 103 = 0.0000000002
MSE 104 = 0.0000000002
MSE 105 = 0.0000000002
MSE 106 = 0.0000000001
MSE 107 = 0.0000000001
MSE 108 = 0.0000000001
MSE 109 = 0.0000000001
MSE 110 = 0.0000000000
GD_SOLVER DATA END
```

GRAPH on NEXT PAGE



8
6
4
2
0
−2
0
20
40
60
80
100