1 Experiment 1

On fig.1 the left quadfunc1 with parameters:

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$b = (-0.5, 0.5)$$

On fig.1 the right quadfunc2 func with parameters:

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 1.5 \end{pmatrix}$$

$$b = (0.2, 0.4)$$

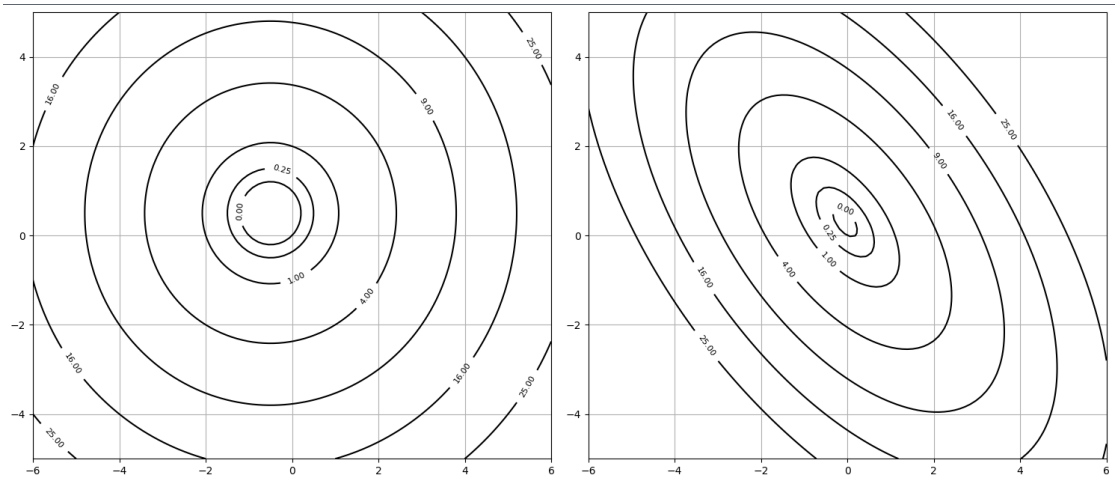

fig.1

With start points [-3.5, 3.5], [0.2, 0.8], [-2, -1] gradient descent with Wolfe linear search is taken to find the minimum point of quadfunc1 and quadfunc2 as showed in fig.2 and fig.3.

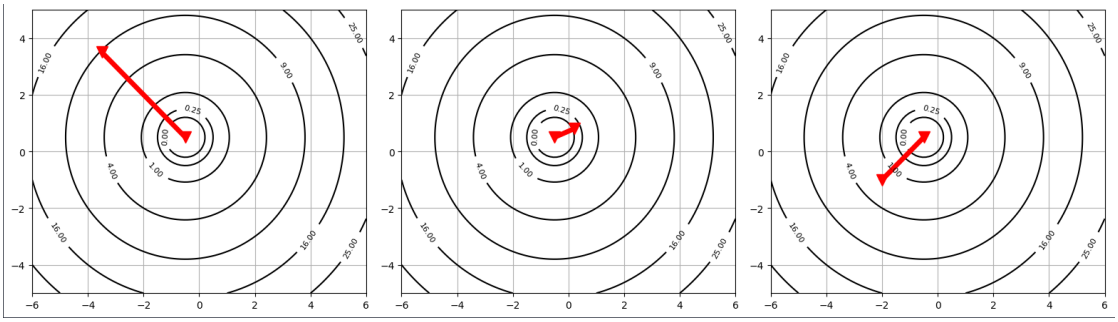

fig.2

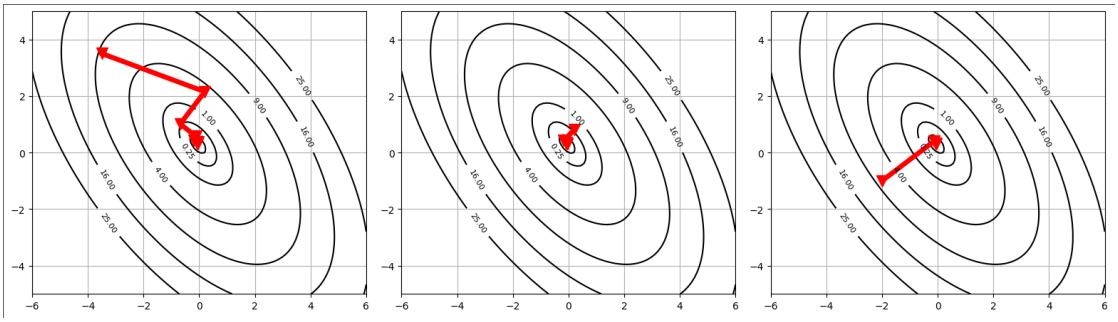

fig.3

With start points [-3.5, 3.5], [0.2, 0.8], [-2, -1] gradient descent with Armijo linear search is taken to find the minimum point of quadfunc1 and quadfunc2 as showed in fig.4 and fig.5.

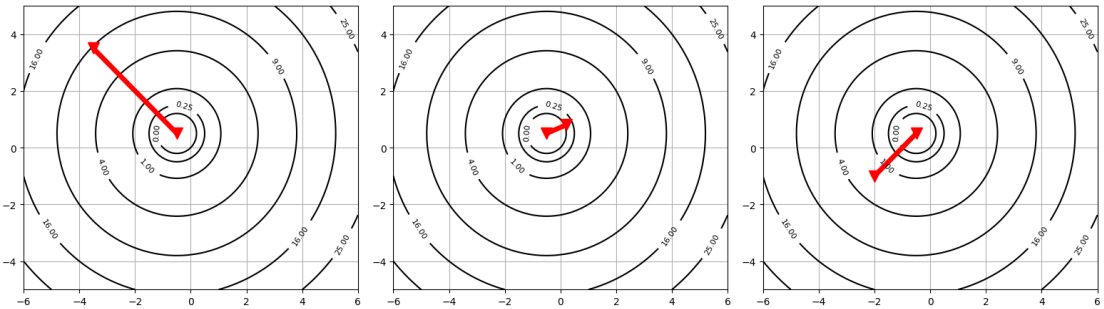

fig.4

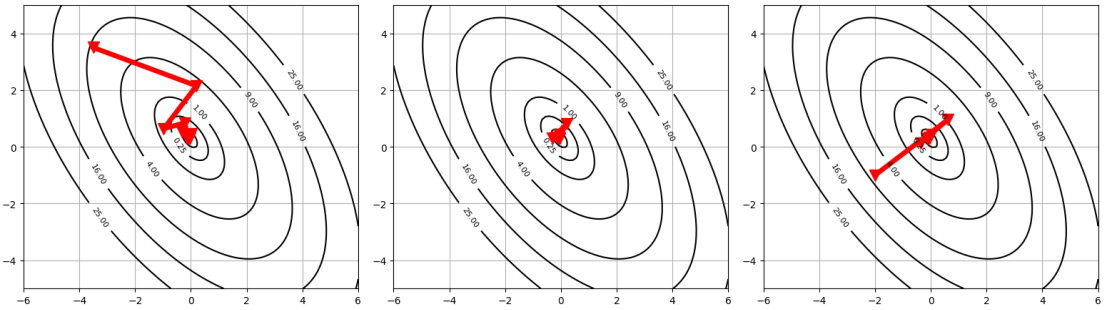

fig.5

With start points [-3.5, 3.5], [0.2, 0.8], [-2, -1] gradient descent with Constan linear search is taken to find the minimum point of quadfunc1 and quadfunc2 as showed in fig.6 and fig.7.

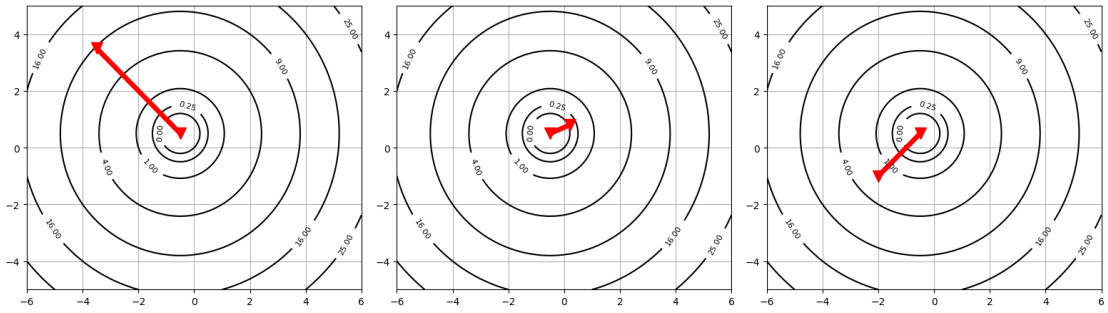

fig.6

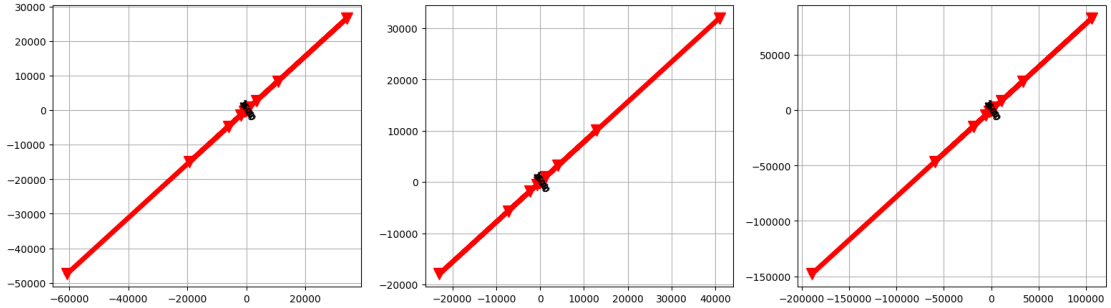

fig.7

For all these three linear search in the function quadfunc1 the trajetories are very similar, it's very eazy for all these 3 methods to find the minumum point. But in the function quadfunc2, they behave very differently. Wolfe takes less steps than others and Armijo sometimes pass trough the minimum point and needs to take back. For Constant method in hte quadfunc2 it even can't converghe from the any of the 3 startpoints.

2 Experiment 2

Settings of experiment in this part:

- randomseed: 45
- number of dimensions: [10, 100, 1000]
- value of condition: [100, 200,..., 1900, 2000]. In total 20.
- for every n, 10 funcs are randomly generated by k.
- for every generated function, 5 start points are ramdomly chosen and iteration for the funcion is their mean iterations.
- the curve with deep color is the average curve of all launches with an n.

In fig.8 Wolfe is used.

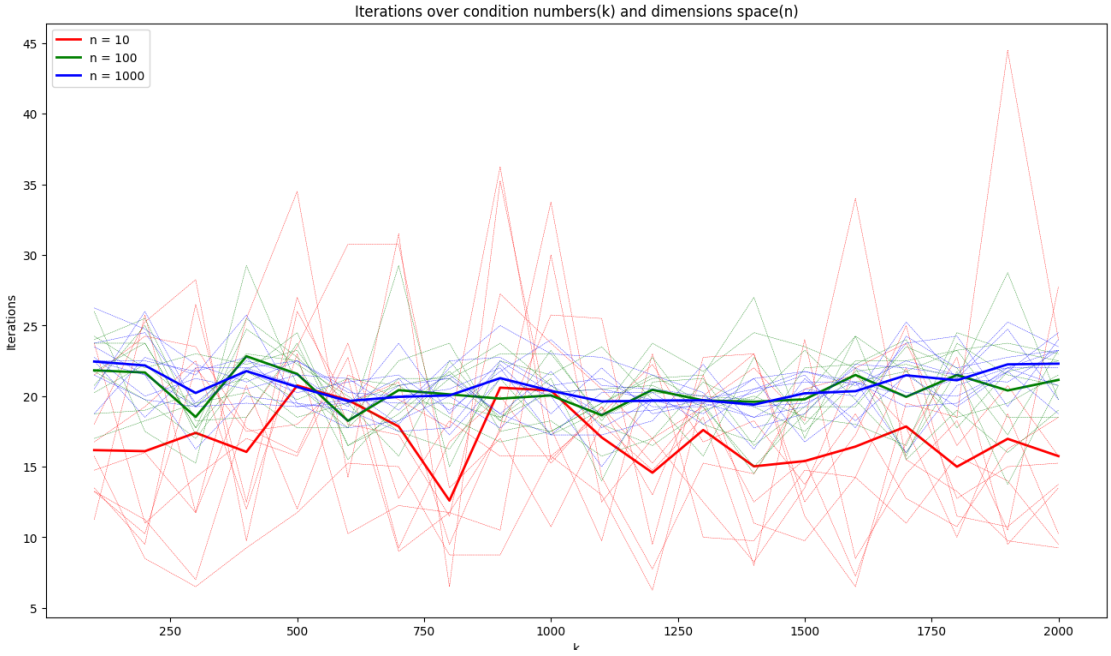

fig.8

In fig.8 Armijo is used.

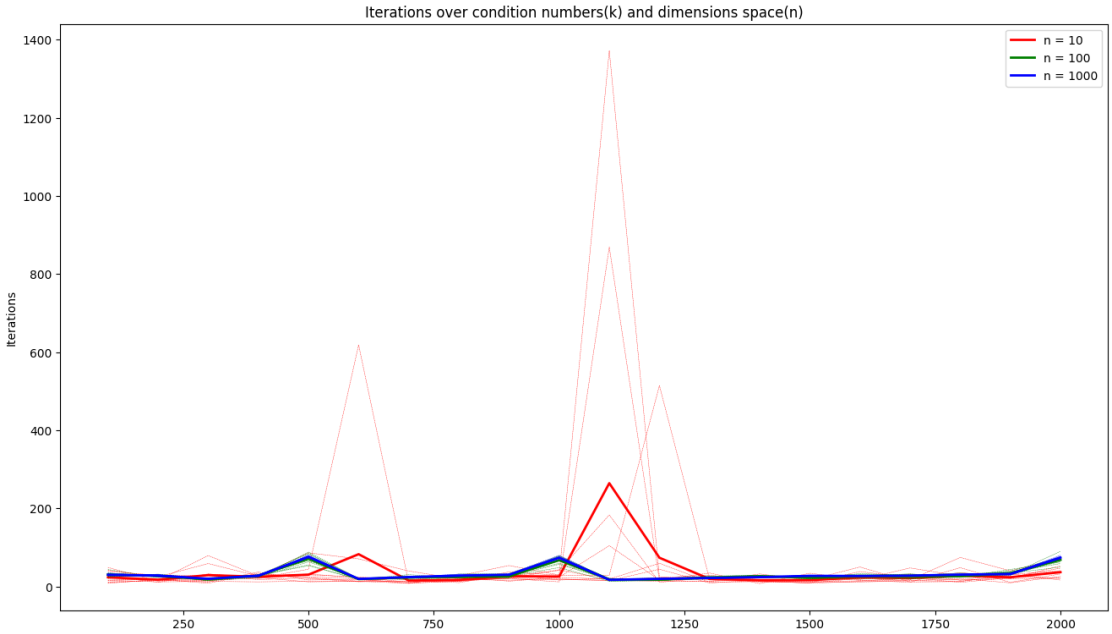

fig.9

From the results of fig.8 and fig.9 we can see that k hardly affect the iterations of optimization no matter whant kind of linears search method we use. n doese not affect the iterations much at n from 100 to 1000 but when n = 10, the iterartions needed are more likely to be fewer. And it's very clear that in fig.8 the descending proccess is much more stable than in fig.9 which shows the advantage of Wolfe linear search.

3 Experiment3

In this experiment 3 real datasets ['w8a', 'gisette', 'real-sim'] are used to do the logistic regression. For every dataset, gradient descent and newto methods with 3 types of linear search methods are taken. Function value and grandient norm are taken as metric y-axis to judge their performence.

In fig.10 and fig.11 are results of gradient descent with Wolfe for w8a.

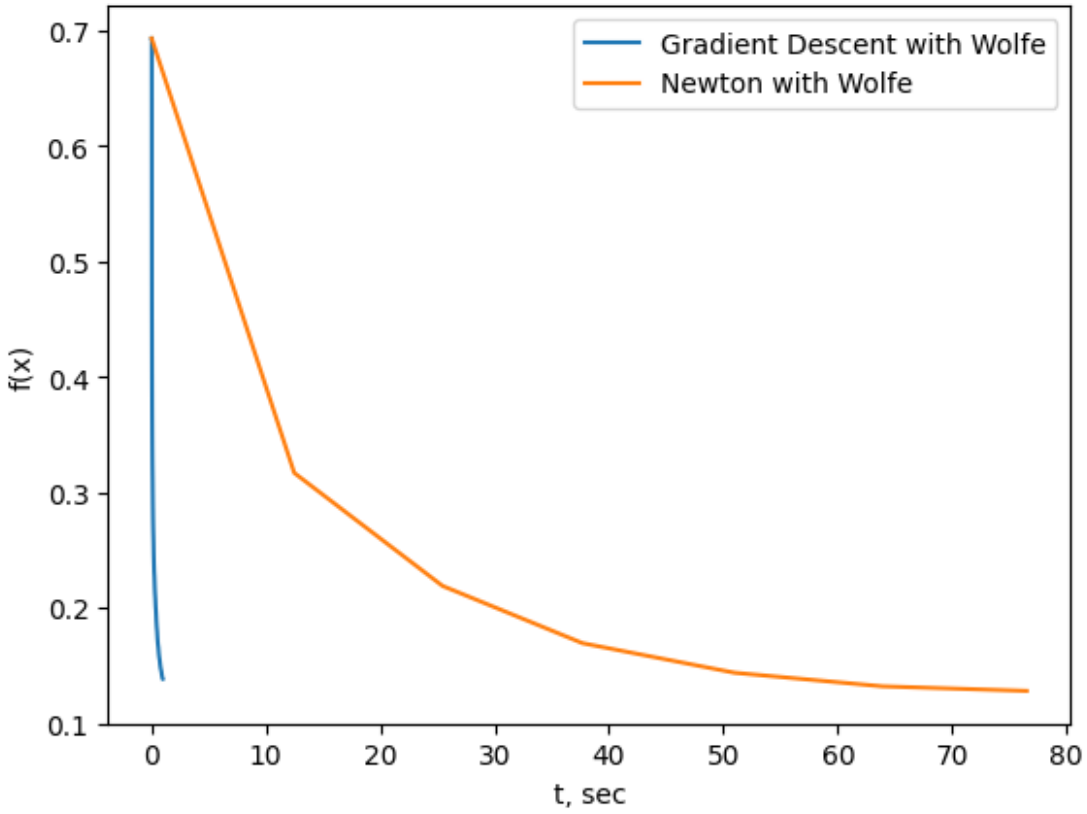

fig.10

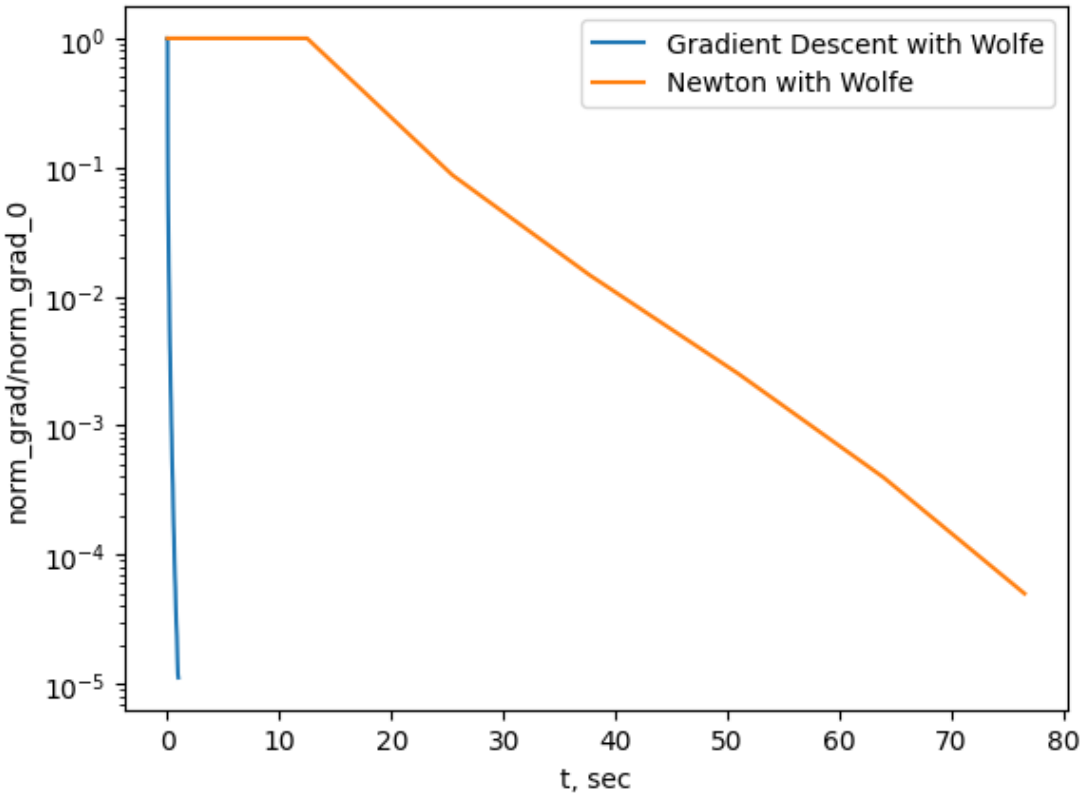

fig.11

In fig.12 and fig.13 are results of gradient descent with Armijo for w8a.

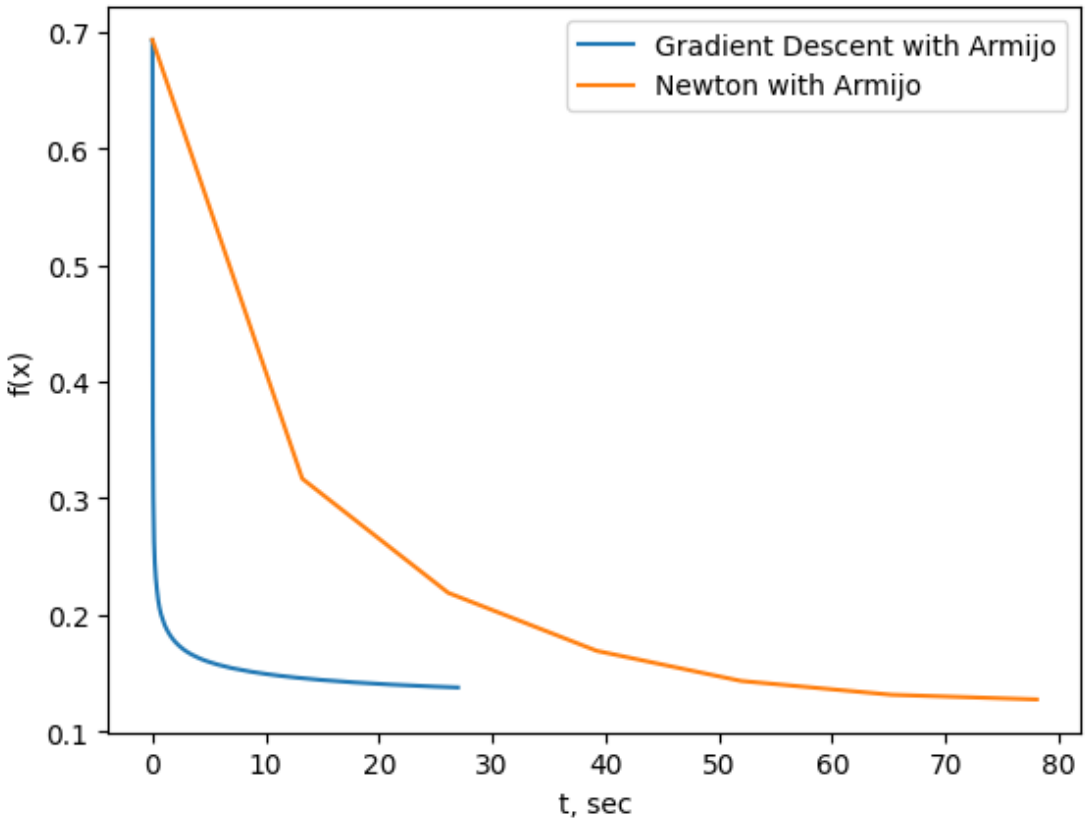
Gradient Descent with Armijo
Newton with Armijo
f(x)
t, sec
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50
60
70
80

fig.12

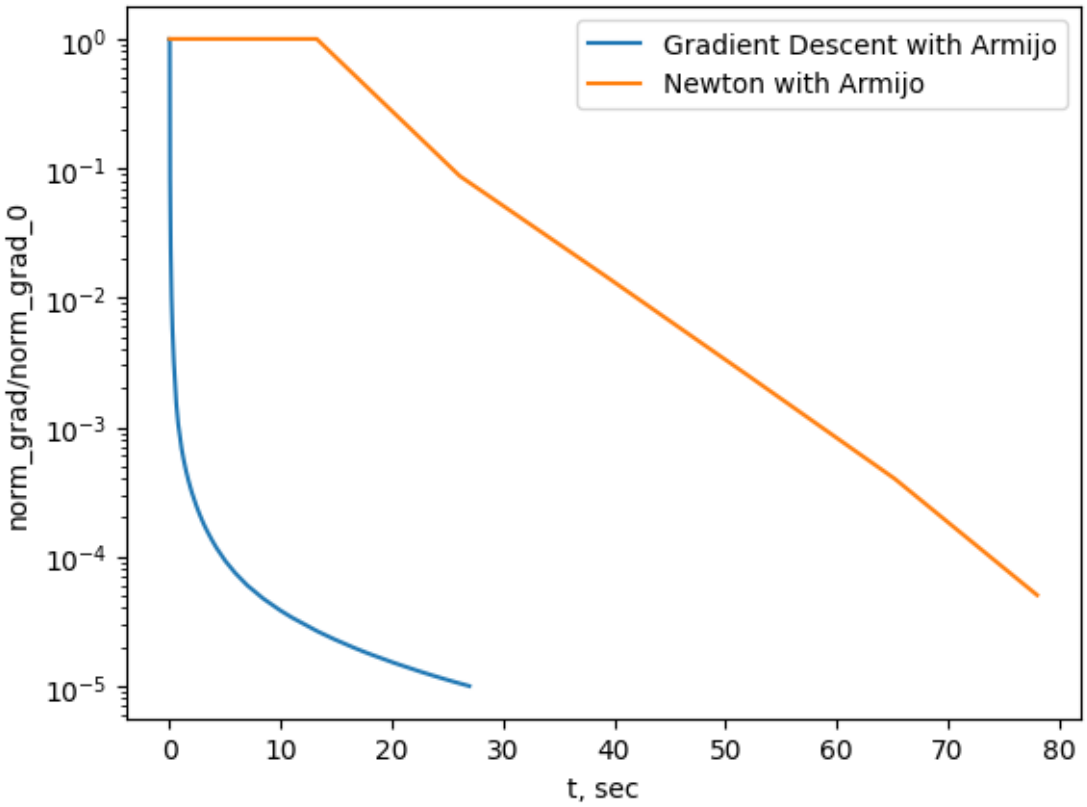

fig.13

In fig.14 and fig.15 are results of gradient descent with Constant for w8a.

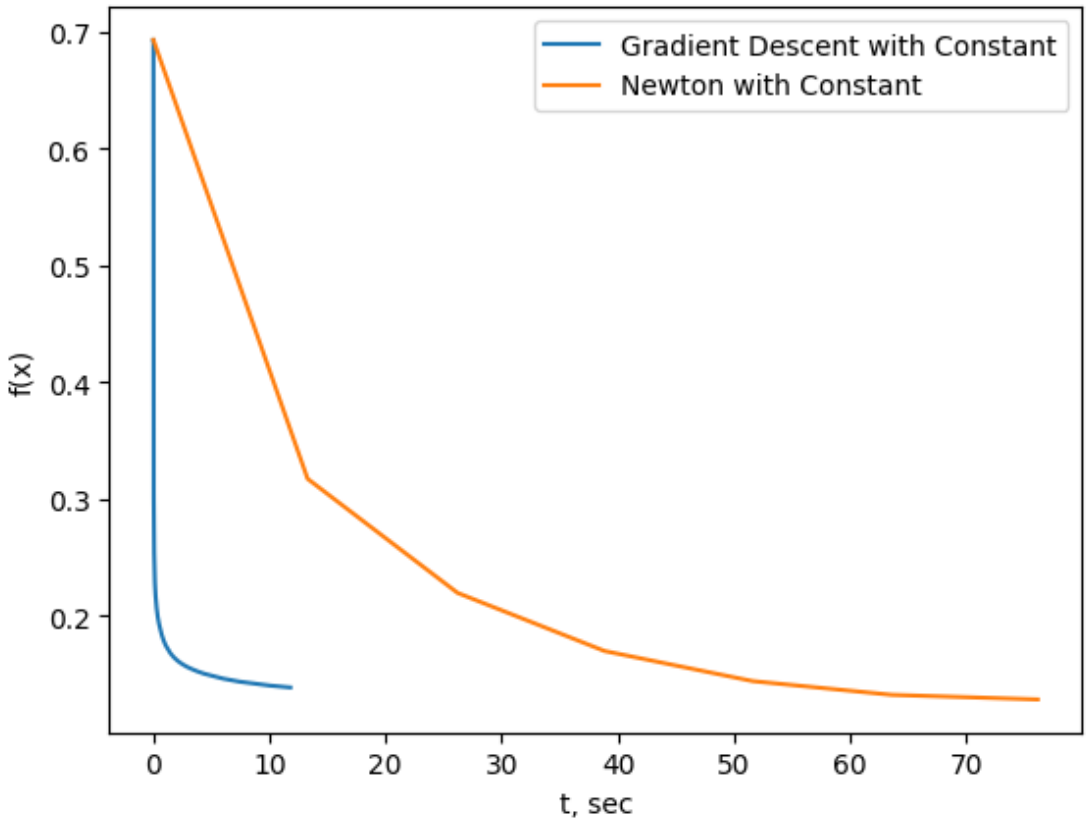
Gradient Descent with Constant
Newton with Constant
f(x)
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50
60
70
t, sec

fig.14

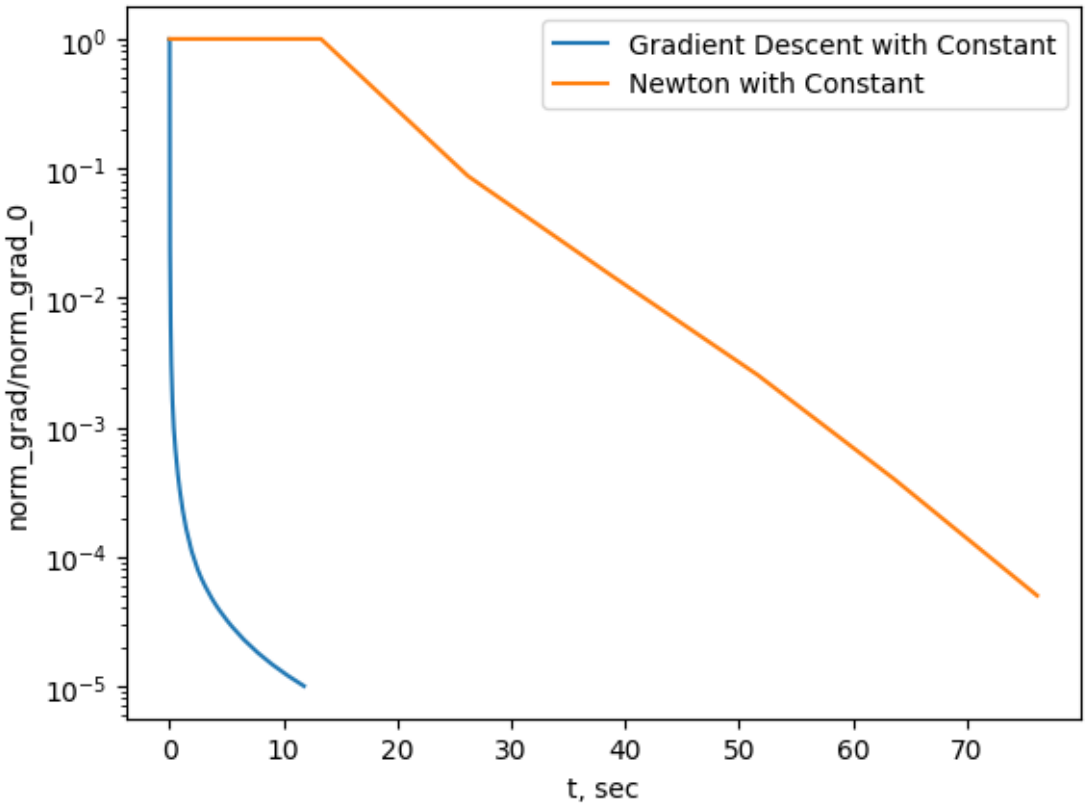

fig.15

In the part we can see all methods converge to the minimum point and the time taken by the gradient descent is much shoter than newton. Wolfe is still the best linear search method with gradient descent.

In fig.16 and fig.17 are results of gradient descent with Wolfe for gisette.

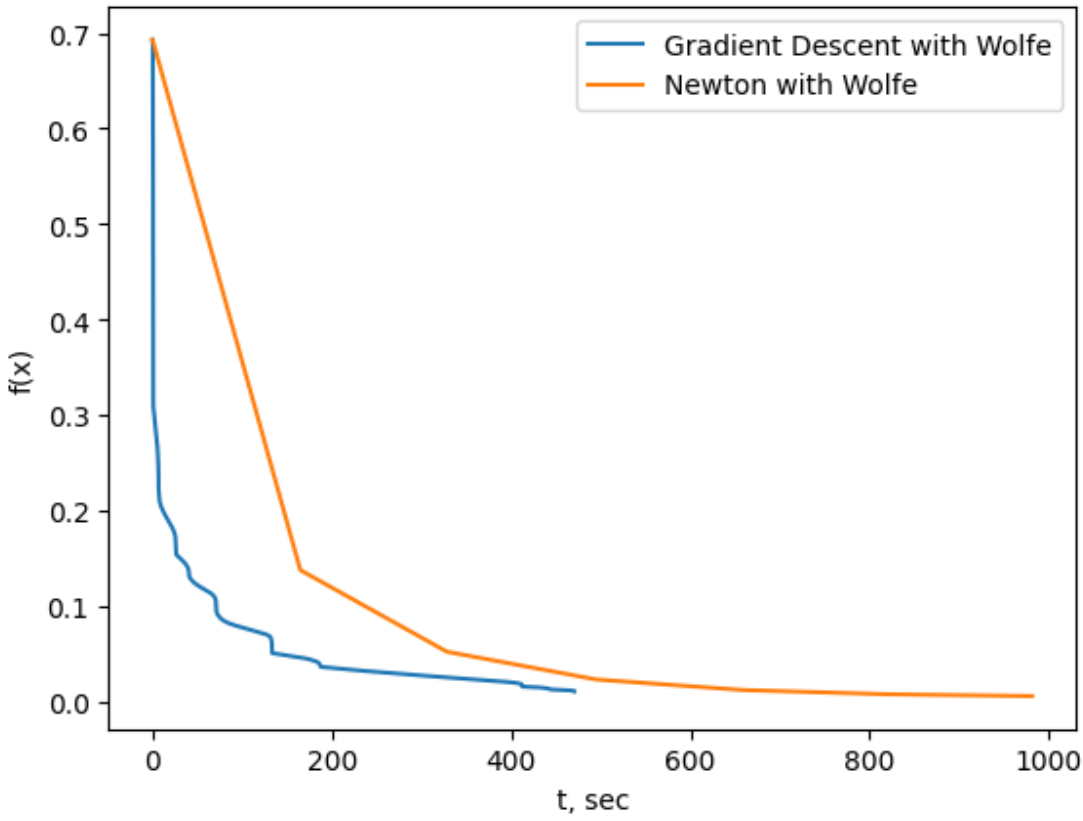
Gradient Descent with Wolfe
Newton with Wolfe
f(x)
t, sec
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000

fig.16

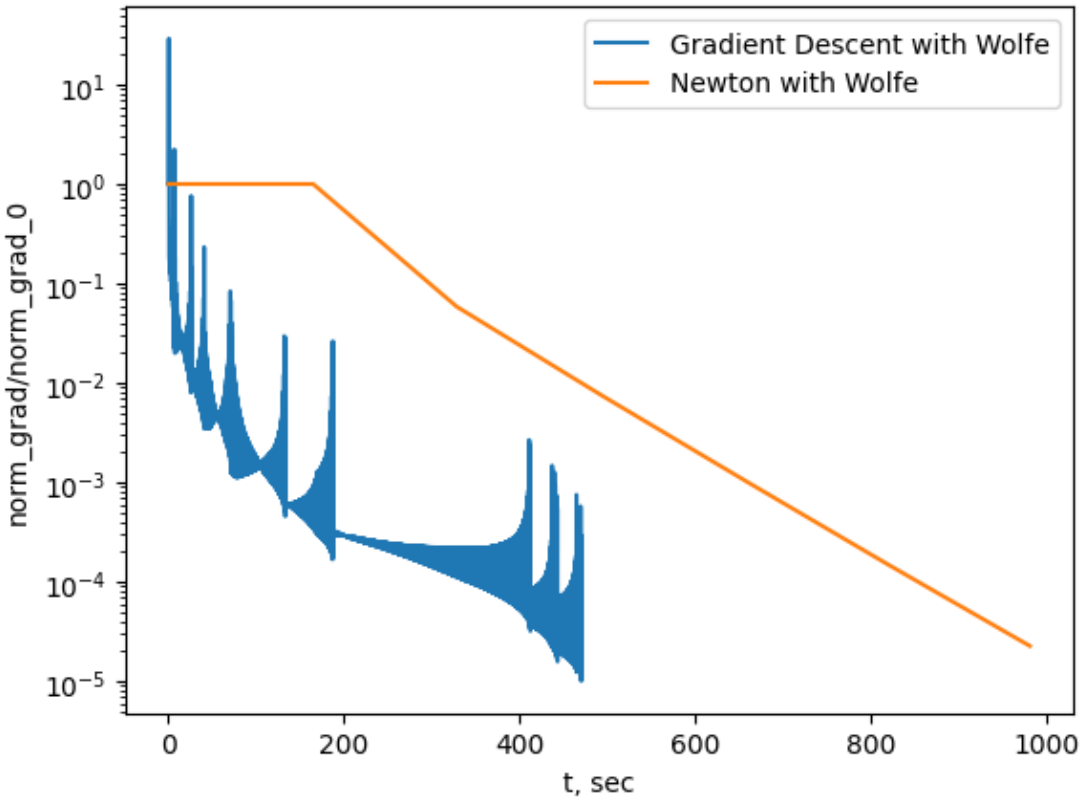

fig.17

In fig.18 and fig.19 are results of gradient descent with Armijo for gisette.

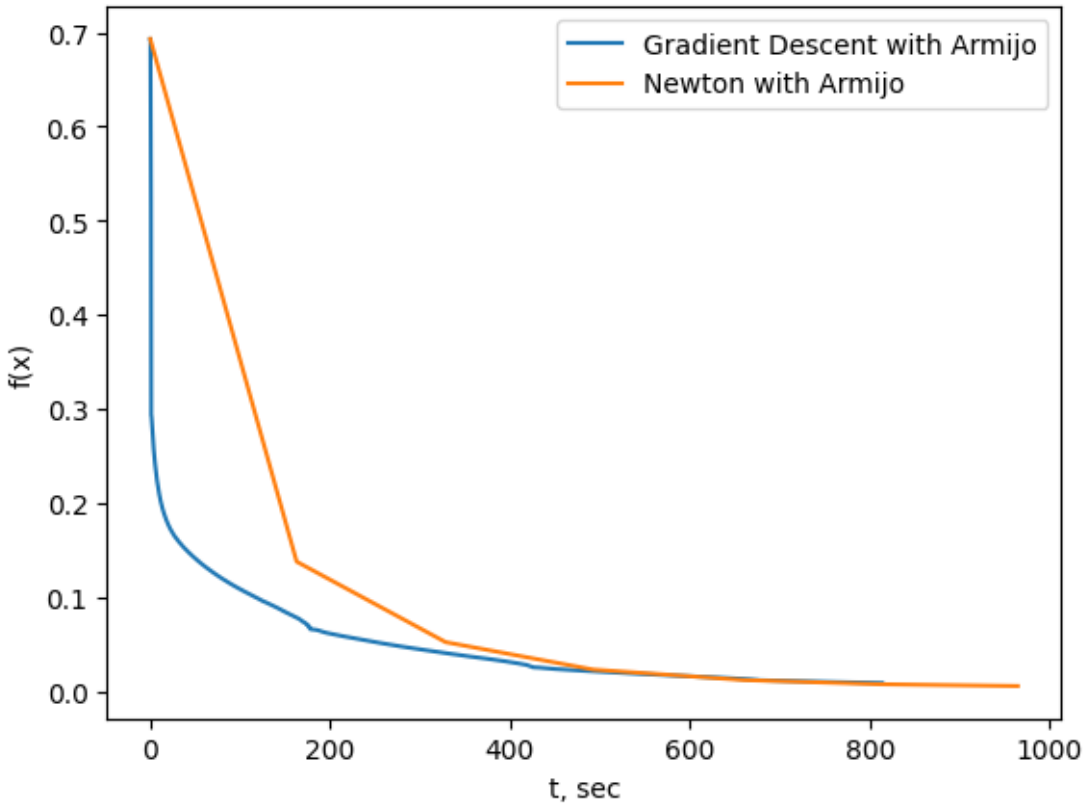
Gradient Descent with Armijo
Newton with Armijo
f(x)
t, sec
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000

fig.18

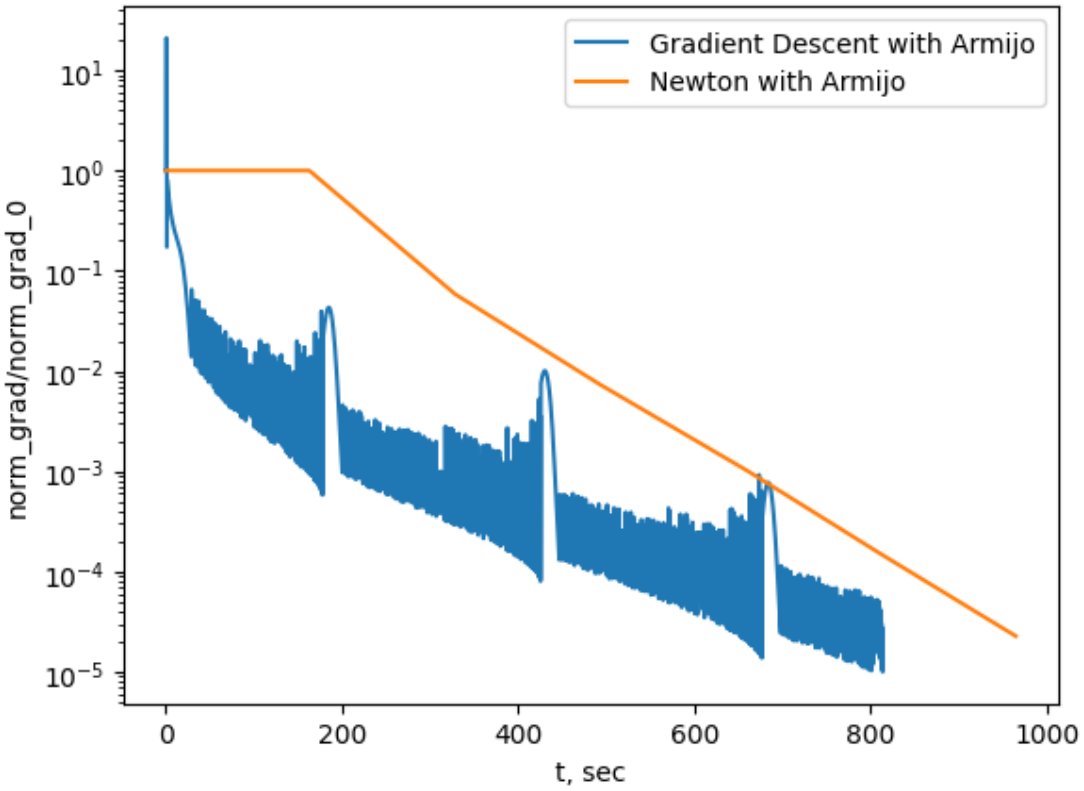

fig.19

In fig.20 and fig.21 are results of gradient descent with Constant for gisette.

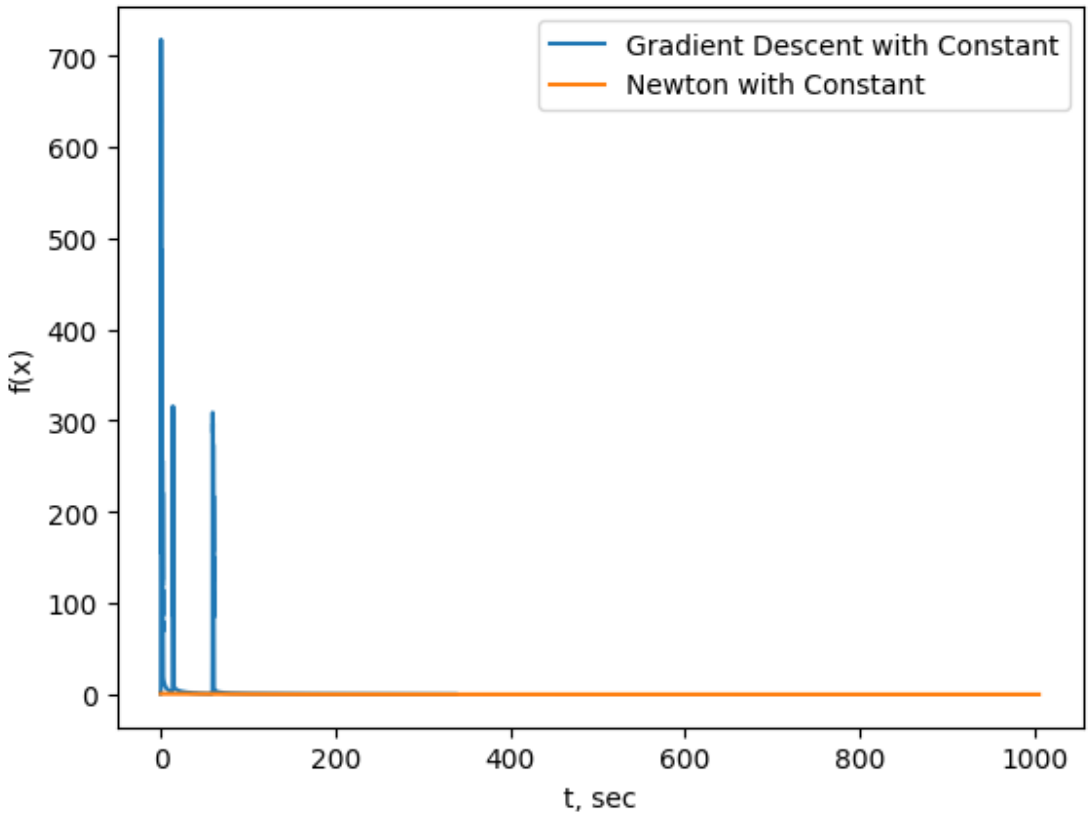
Gradient Descent with Constant
Newton with Constant
f(x)
700
600
500
400
300
200
100
0
0
200
400
600
800
1000
t, sec

fig.20

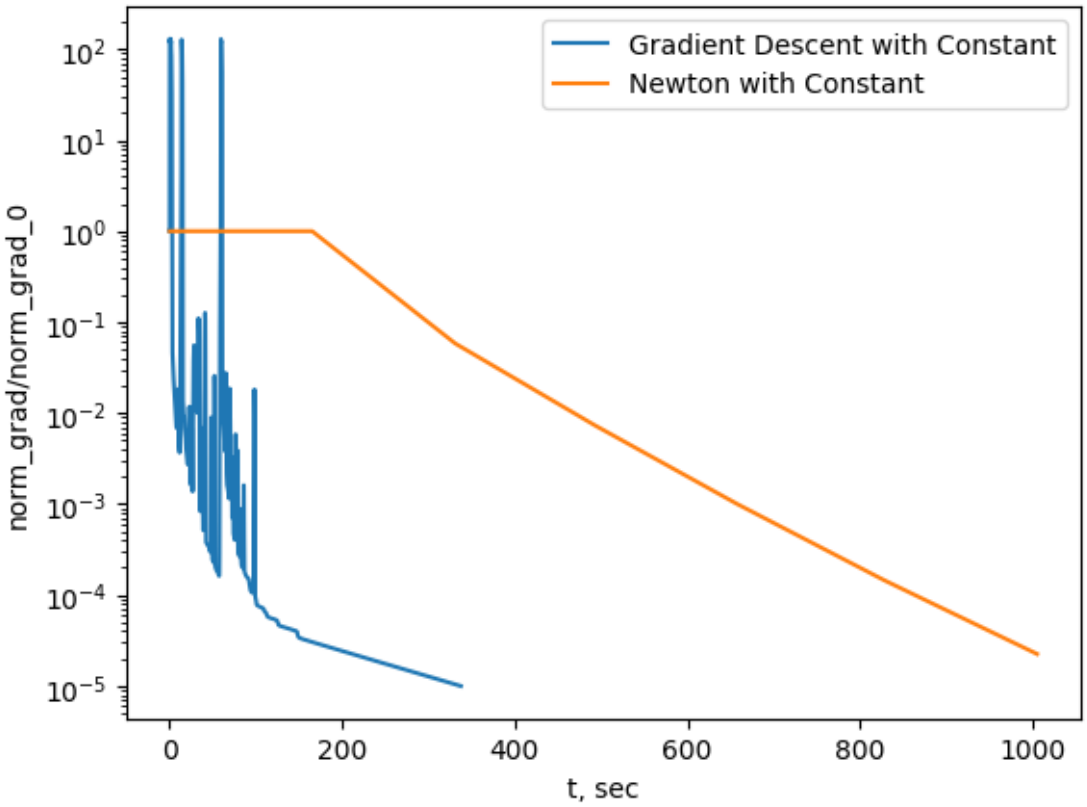

fig.21

In this part, form fig17,19,21 we can see that the gradient norm oscillates violently at a lot of points when using gradient descent, which show s the dataset is steep. But curve of newton is without this problem, it's still smooth.

In fig.22 and fig.23 are results of gradient descent with Wolfe for real-sim.

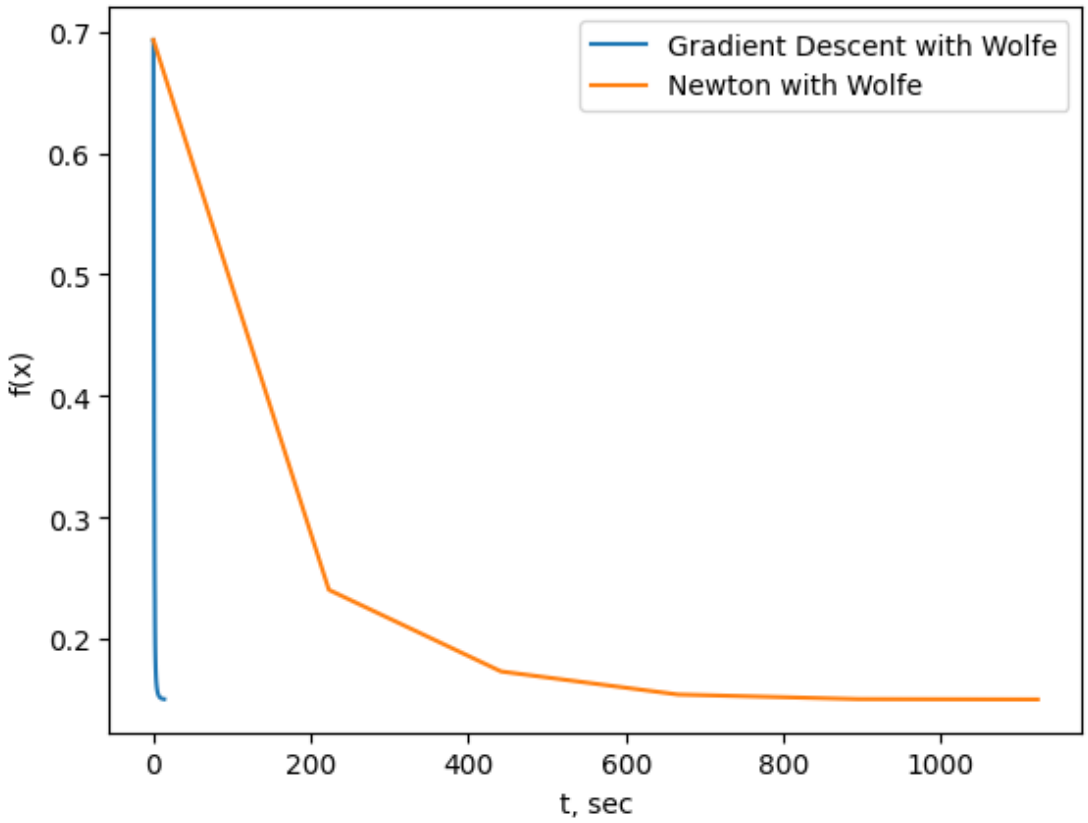
Gradient Descent with Wolfe
Newton with Wolfe
f(x)
0.7
0.6
0.5
0.4
0.3
0.2
0
200
400
600
800
1000
t, sec

fig.22

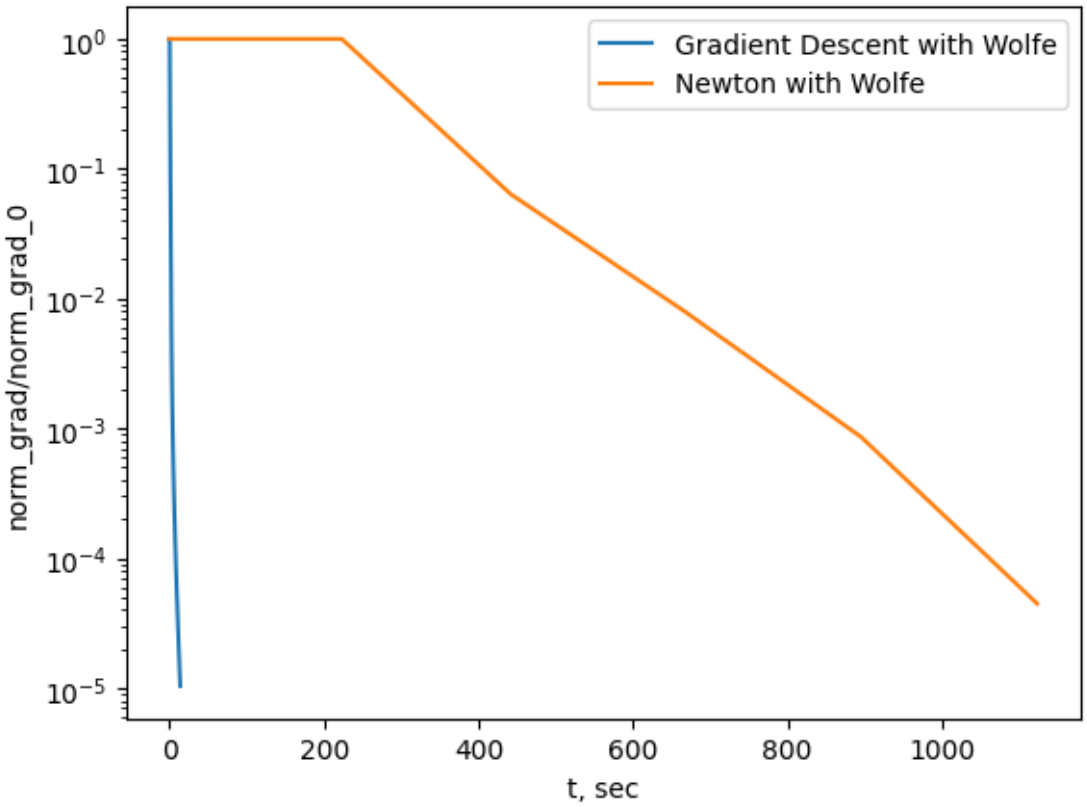

fig.23

In fig.24 and fig.25 are results of gradient descent with Armijo for real-sim.

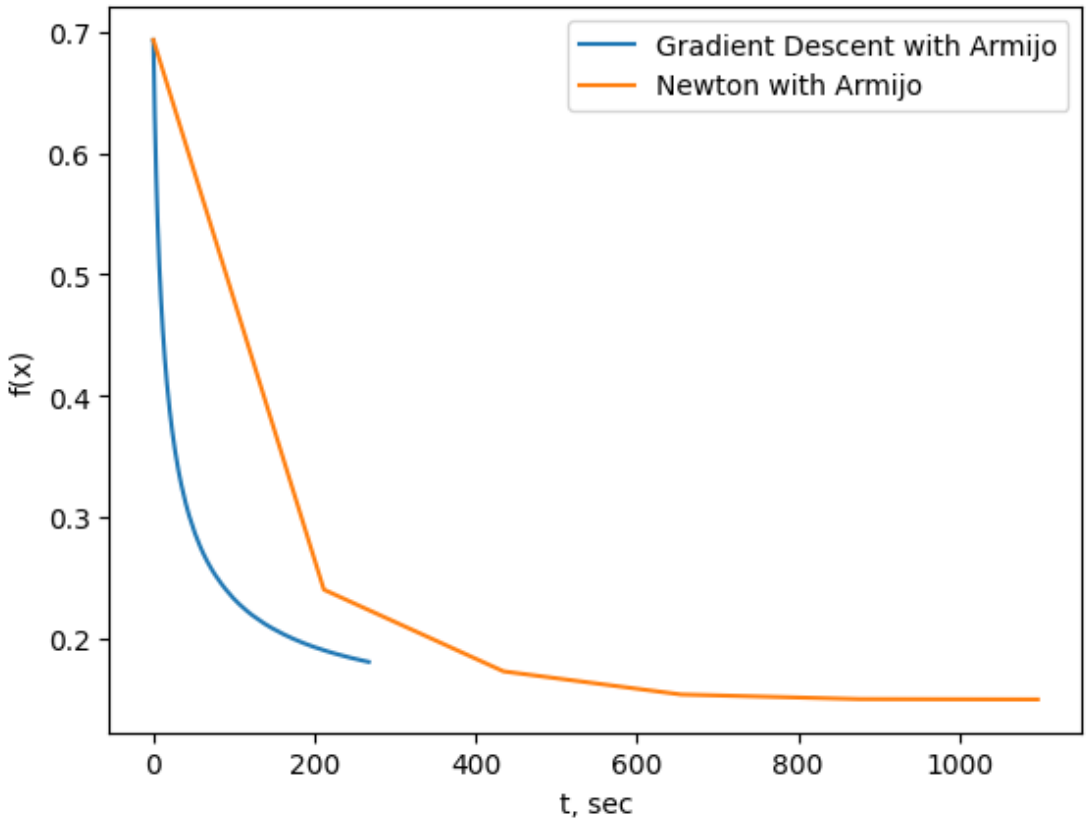
Gradient Descent with Armijo
Newton with Armijo
f(x)
0.7
0.6
0.5
0.4
0.3
0.2
0
200
400
600
800
1000
t, sec

fig.24

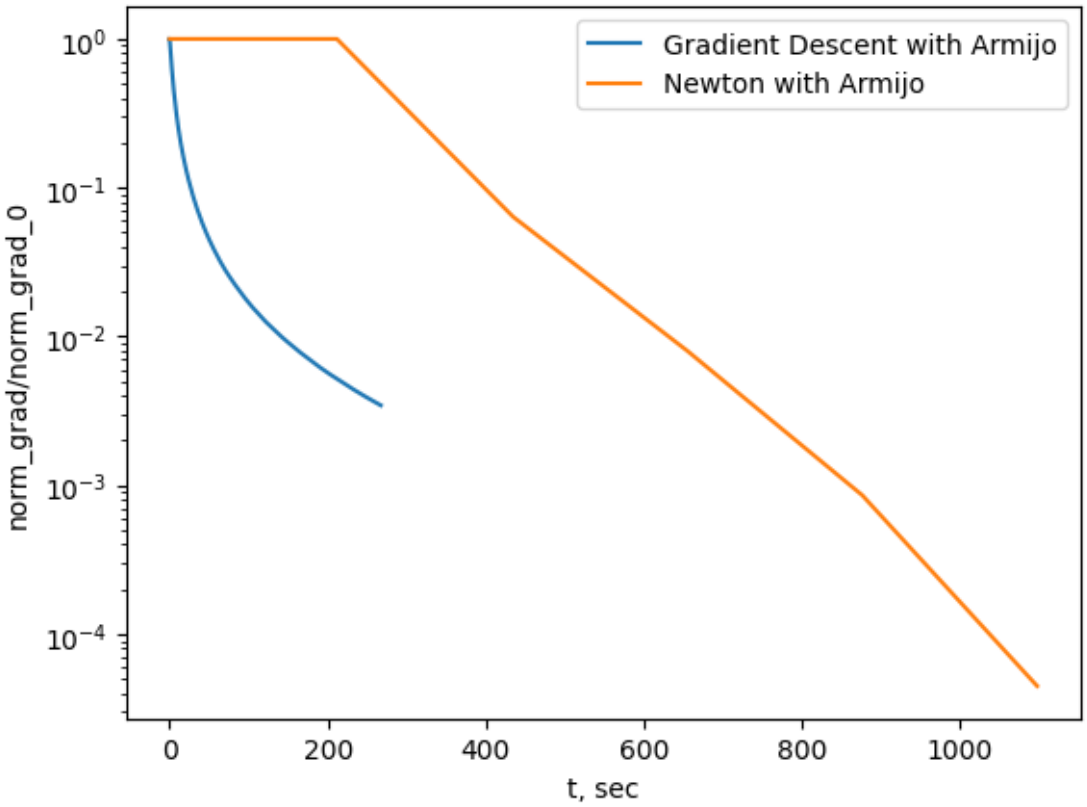

fig.25

In fig.26 and fig.27 are results of gradient descent with Constant for real-sim.

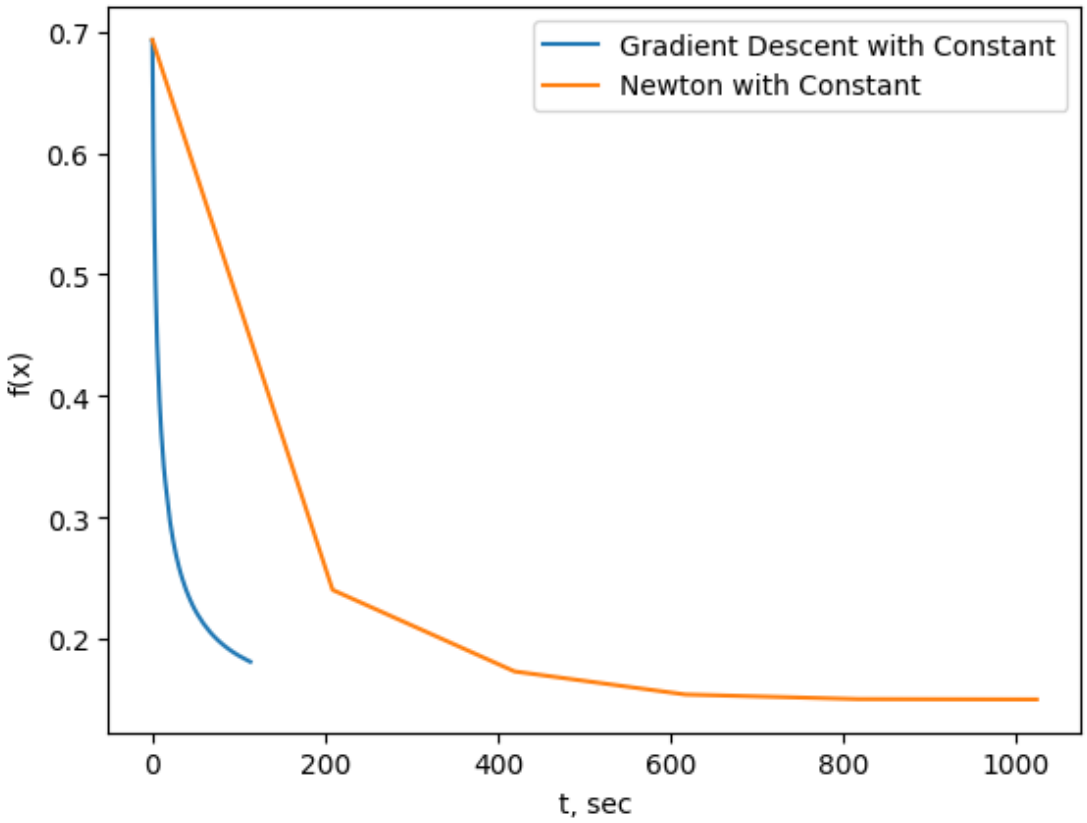
Gradient Descent with Constant
Newton with Constant
f(x)
0.7
0.6
0.5
0.4
0.3
0.2
0
200
400
600
800
1000
t, sec

fig.26

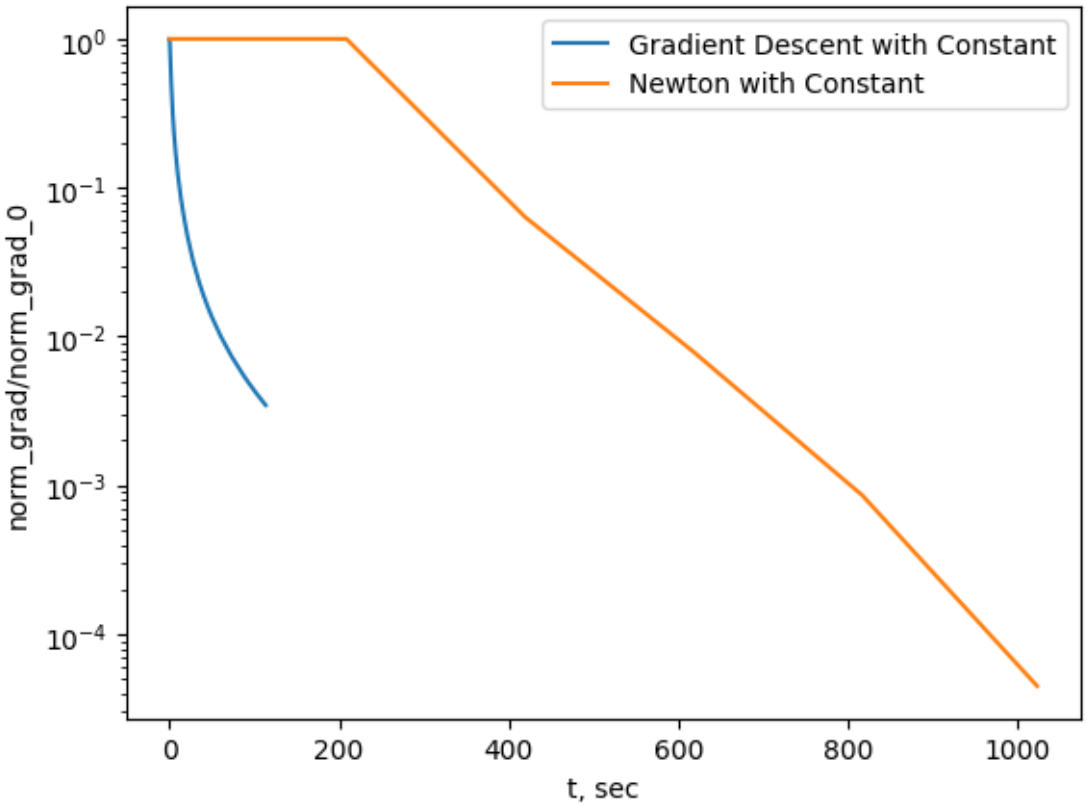

fig.27

In this part, newton still takes more time but still can converge until the tolerance. Gradient with Wolfe also converges to the tolerance and is still very fast. But for other methods cobinations it's hard to get to the tolerance.

Complexicy:

- newton space(O(hessian)): $O(n^2)$.
- gradient descent space(O(mat_A)): $O(kn)$
- newton iteration(O(solve_LU=b)+O(cholesky_factor)): $O(n^2) + O(n^3)$
- gradient descent iteration(O(Ax)): $O(kn)$

In general, especially in machine learning, the dimensions of the data are very large, so the gradient descent method is mostly used. Among them, Wolfe's performance is the best in most cases. In addition, when it is difficult to converge to the minimum value in some data spaces and if the spatial dimension is not high, we can also choose Newton's optimization method.

4 Experiment4

In this part I've randomly generated 2 functions. One is quadratic function and another is logistic regression. The random seed is 46.

For logistic function:

- dimension: 20
- samples: 10000
- regcoef: 1/samples
- y_metric $= \frac{||\nabla f(x)||_2^2}{||\nabla f(x_0)||_2^2}$

For quadratic function:

- dimension: 20
- y_metric $= \frac{f(x)-f(x^*)}{f(x^*)}$

For every descending curve 4 randomly chosen start points are used, and the curve is actually the average of all the 4 curves to show general cases.

$f(x^*)$ is the min value of the function and for quadratic function it is calculated as:

$$f(x^*) = -0.5x^T A^{-1} x$$

In fig.28,fig.29 and fig.30 the Armijo, Wolfe and Constant line search methods are used respectively to optimize the generated logistic regression task.

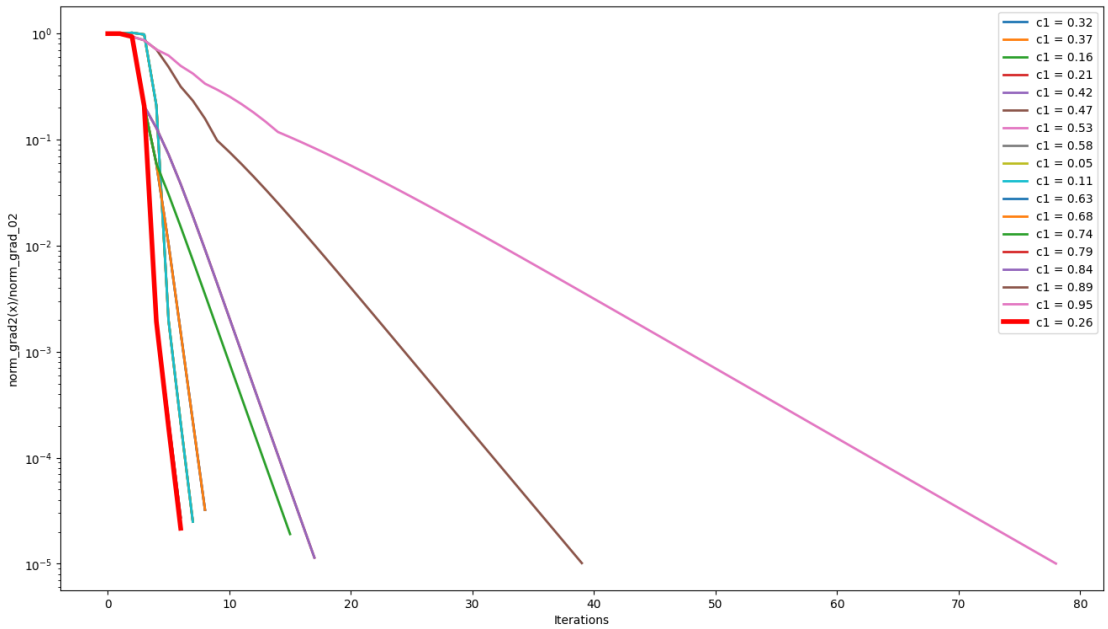

fig.28

In the fig.28 are descending curves with differenct c1 values for Armijo.

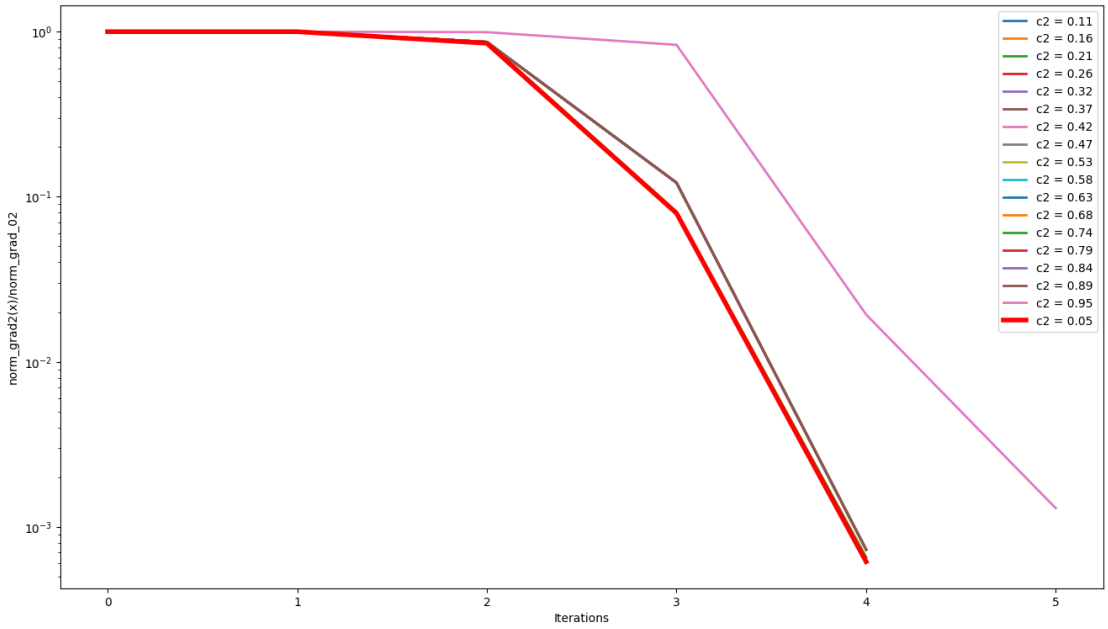

fig.29

In the fig.29 are descending curves with differenct c2 values for Wolfe and c1 is set to 0.0001.

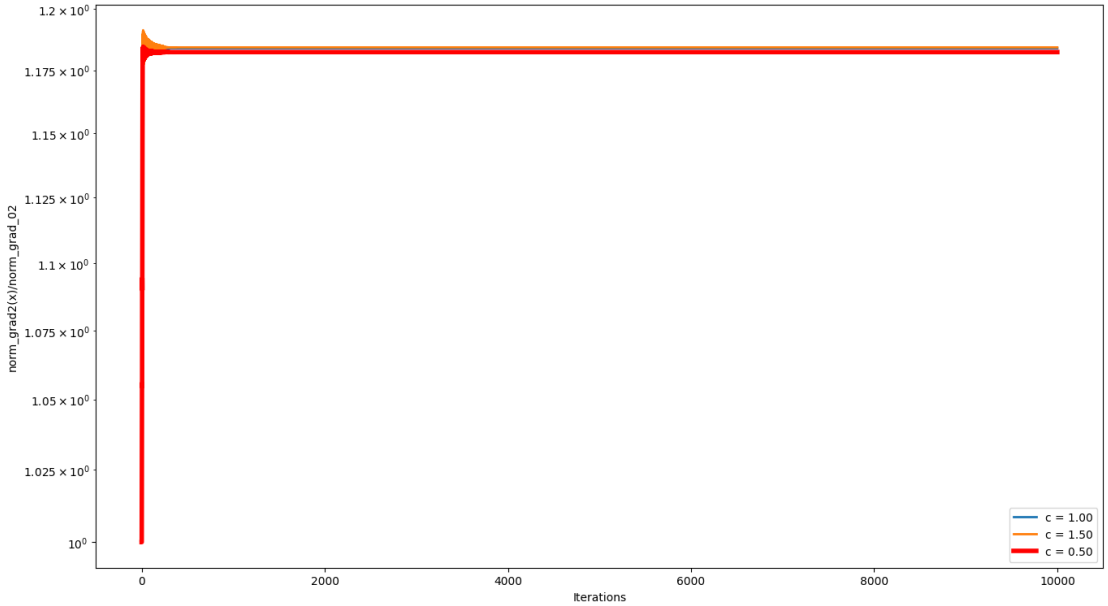

fig.30

In the fig.30 are descending curves with differenct c values for Constant.

- For Armijo, best c1 is 0.26.
- For Wolfe, best c2 is 0.05.
- For Constant, none for the curves converge.

In fig.31,fig.32 and fig.33 the Armijo, Wolfe and Constant line search methods are used respectively to optimize the generated quadratic function.

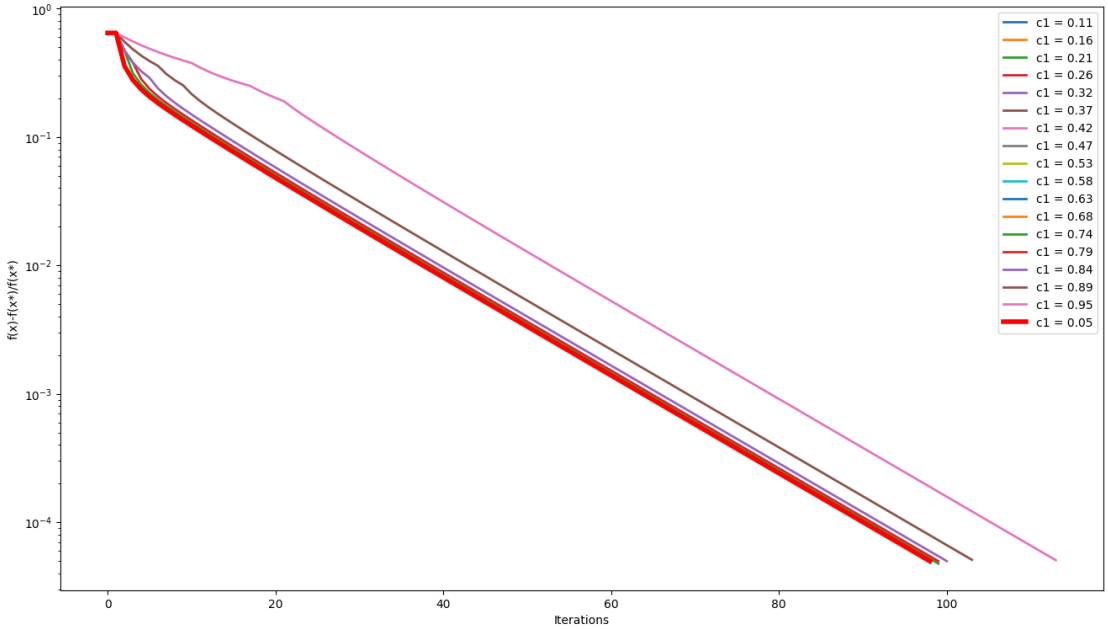

fig.31

In the fig.31 are descending curves with differenct c1 values for Armijo.

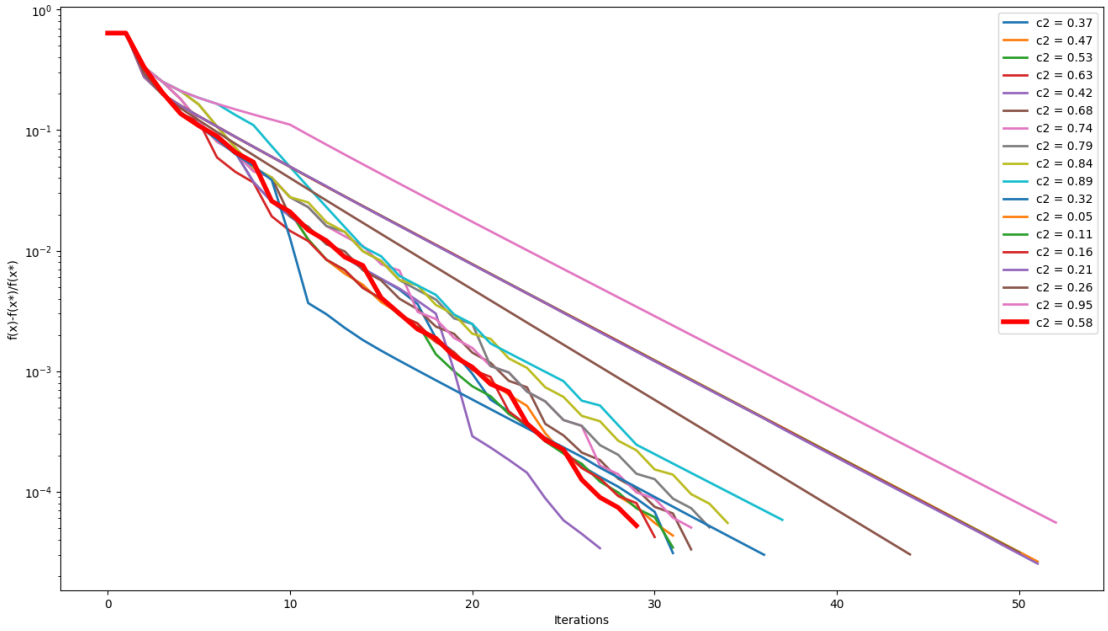

fig.32

In the fig.32 are descending curves with differenct c2 values for Wolfe and c1 is set to 0.0001.

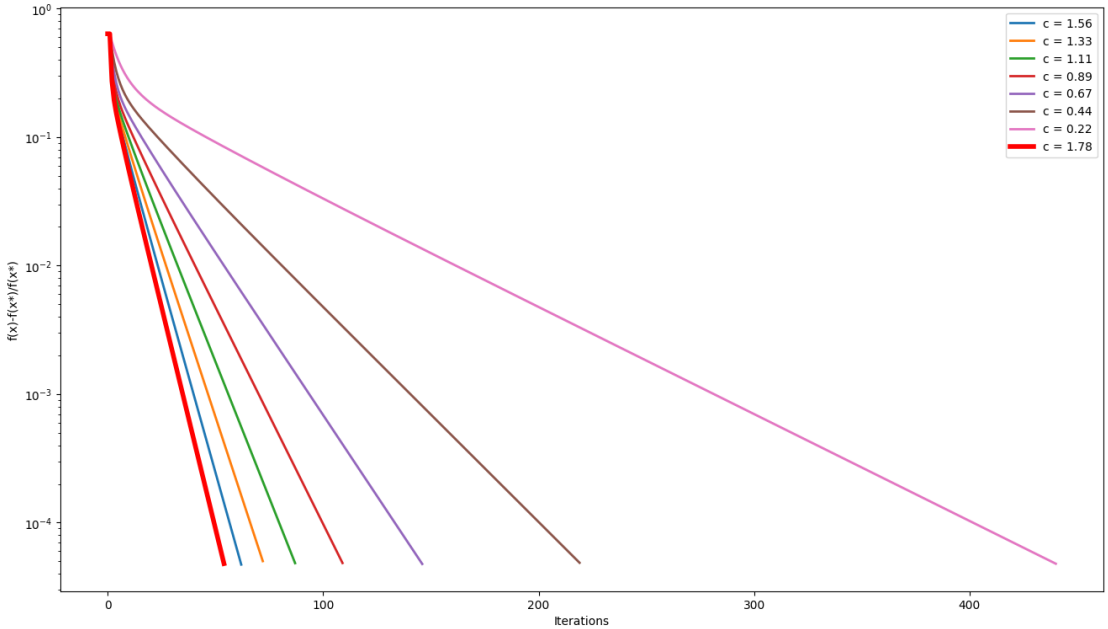

fig.33

In the fig.33 are descending curves with differenct c values for Constant.

- For Armijo, best c1 is 0.05.

- For Wolfe, best c2 is 0.058.
- For Constant, best c is 1.78.

5 Experiment5

In this part all of the parameters are as in the section Experiment4 execpt here we will use new's optimization method instead of gradient descent.

In fig.34,fig.35 and fig.36 the Armijo, Wolfe and Constant line search methods are used respectively to optimize the generated logistic regression task.

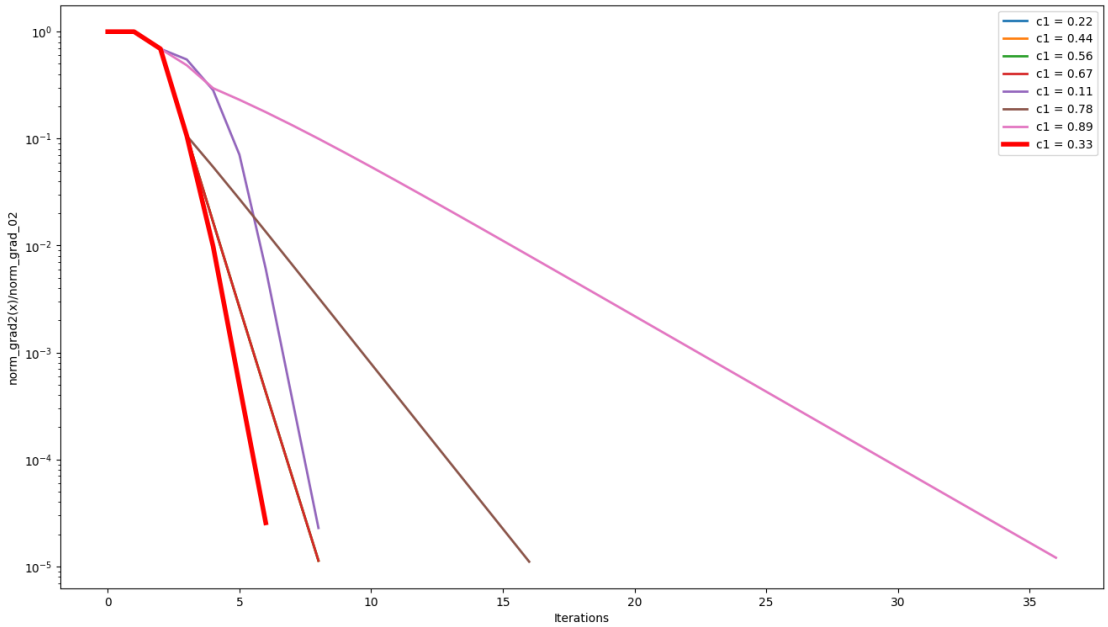

fig.34

In the fig.34 are descending curves with differenct c1 values for Armijo.

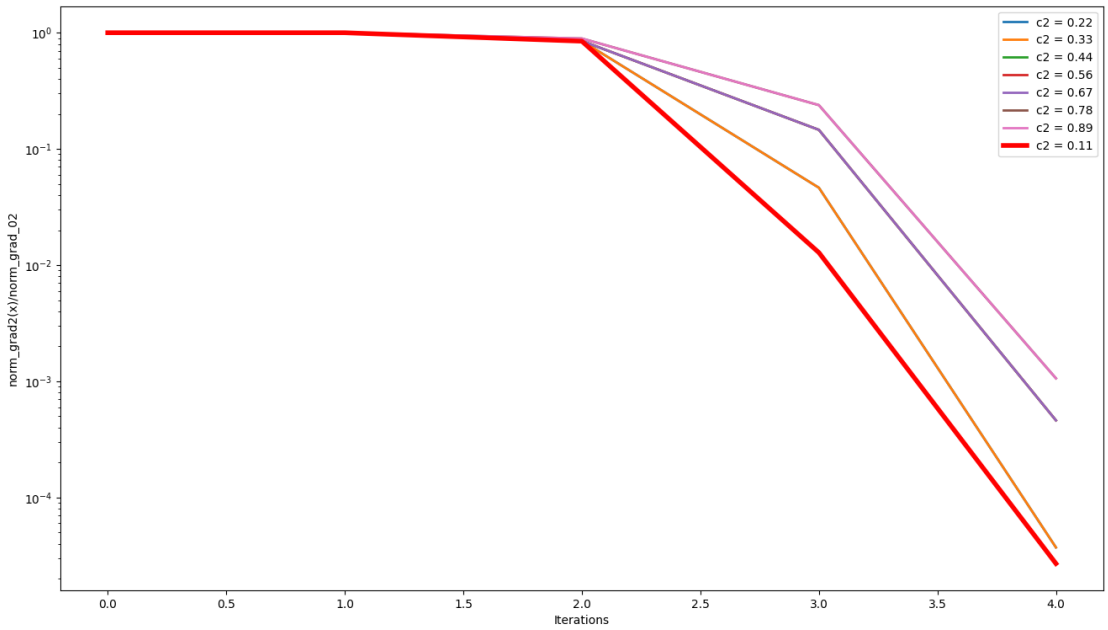

fig.35

In the fig.35 are descending curves with differenct c2 values for Wolfe and c1 is set to 0.0001.

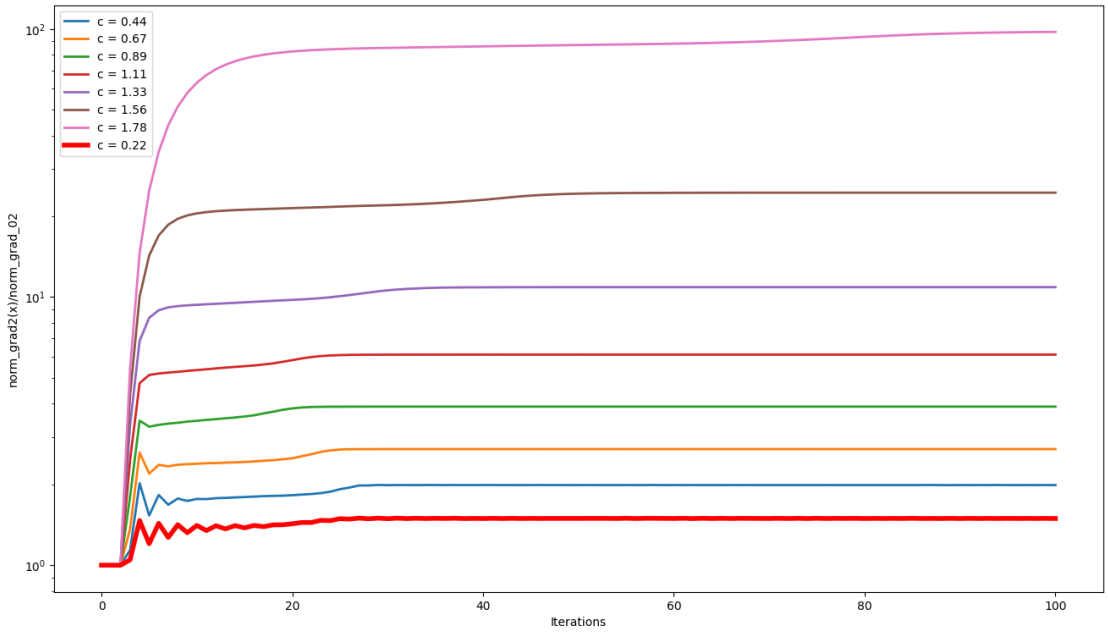

fig.36

In the fig.36 are descending curves with differenct c values for Constant.

- For Armijo, best c1 is 0.33.

- For Wolfe, best c2 is 0.11.
- For Constant, none for the curves converge.

In fig.37,fig.38 and fig.39 the Armijo, Wolfe and Constant line search methods are used respectively to optimize the generated quadratic function.

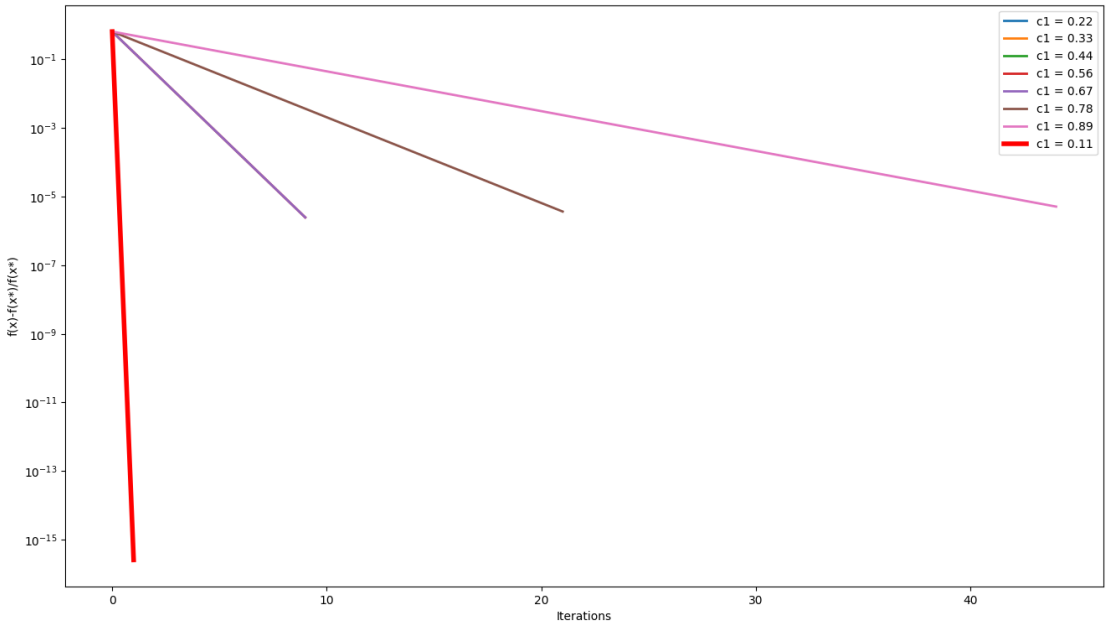

fig.37

In the fig.37 are descending curves with differenct c1 values for Armijo.

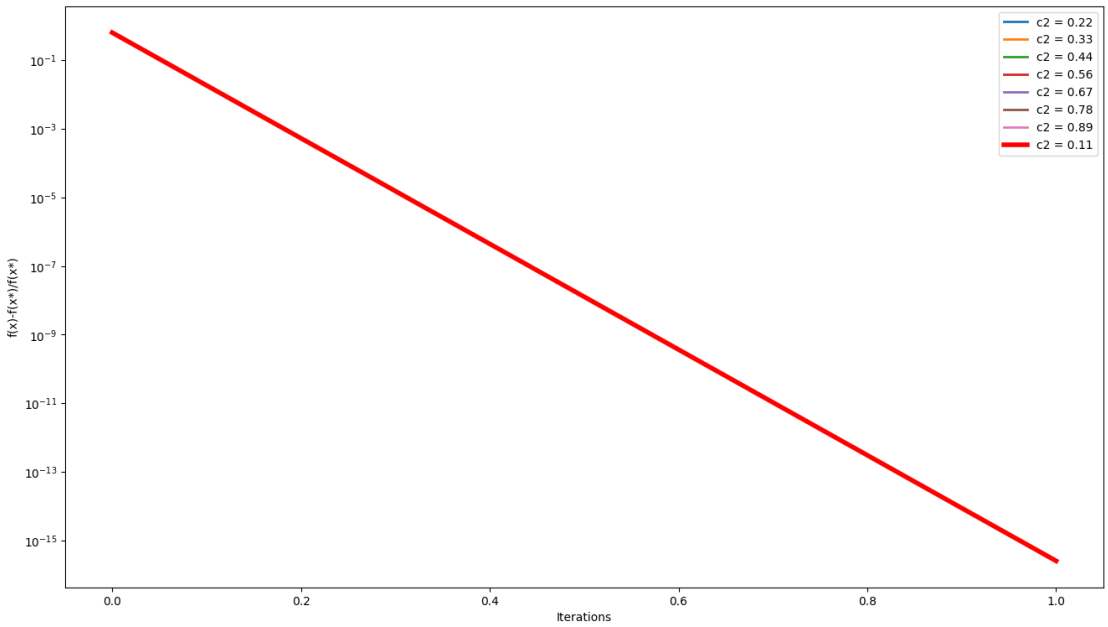

fig.38

In the fig.38 are descending curves with differenct c2 values for Wolfe and c1 is set to 0.0001.

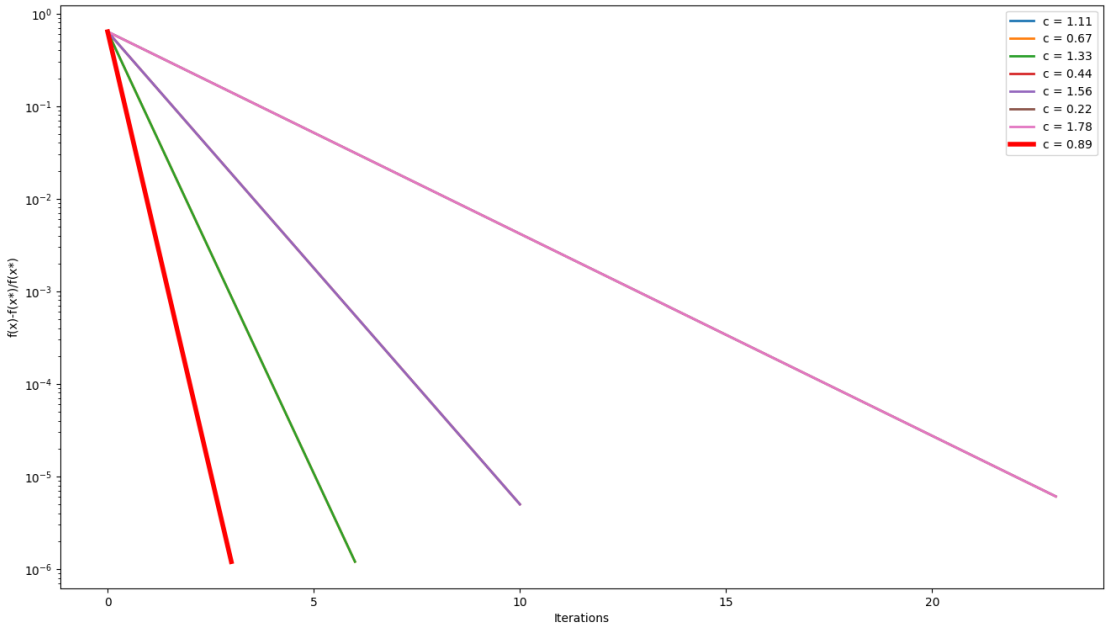

fig.39

In the fig.39 are descending curves with differenct c values for Constant.

- For Armijo, best c1 is 11.

- For Wolfe, all curves have same figure and get to the minumum value within 1 iteration.
- For Constant, best c is 0.89.