CMO 2024 Assignment-2

PonkailashRaj

October 20, 2024

1 Problem 1

1.1 Part 1

The equation $Ax = b$ has a unique solution if and only if A is invertible. Since A is a PD matrix, it is invertible. Hence, the equation $Ax = b$ has a unique solution.
Condider the quadratic objective function $f(x) = \frac{1}{2}x^T Ax - b^T x$. The gradient of the function is given by $\nabla f(x) = Ax - b$. The Hessian of the function is given by $\nabla^2 f(x) = A$. Since A is a PD matrix, the Hessian is positive definite. So x is a minimizer of the function $f(x)$ if and only if $\nabla f(x) = 0$. Hence, the solution of the equation $Ax = b$ is the minimizer of the function $f(x)$.

1.2 Part 2

Optimum $x*$: [2,3,5,8,3]
Number of iterations: 5

1.3 Part 3

$$\min_x \|Ax - b\|^2$$

This is equivalent to minimizing the following quadratic function:

$$\min_x f(x) = \min_x \frac{1}{2} \|Ax - b\|^2$$

Expanding this:

$$f(x) = \frac{1}{2}(Ax - b)^T (Ax - b)$$

This simplifies to:

$$f(x) = \frac{1}{2}\left(x^T A^T Ax - 2b^T Ax + b^T b\right)$$

Thus, the minimization problem becomes:

$$\min_x \left(\frac{1}{2}x^T A^T Ax - b^T Ax + \frac{1}{2}b^T b\right)$$

We will find the gradient and Hessians of the function $f(x)$ to find the minimizer of the function.

$$\nabla_x f(x) = A^T Ax - A^T b$$

$$Hessian(f(x)) = A^T A$$

$$x^T A^T Ax = (Ax)^T (Ax) = \|Ax\|^2 \geq 0$$
$$(A^T A)^T = A^T A$$

Hence $A^T A$ is a PSD matrix. So the function $f(x)$ is convex and clearly continuous, so exactly one minima exists. So the local minima is the global minima.

Existence and Uniqueness of Solutions

The minimization problem has a **unique solution** if and only if the matrix $A^T A$ is **invertible**.

$$A^T Ax = A^T b$$

This is a system of linear equations in x. If $A^T A$ is invertible, the solution is:

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

This x^* is the **unique solution** that minimizes the error $\|Ax - b\|$.

1.4 Part 4

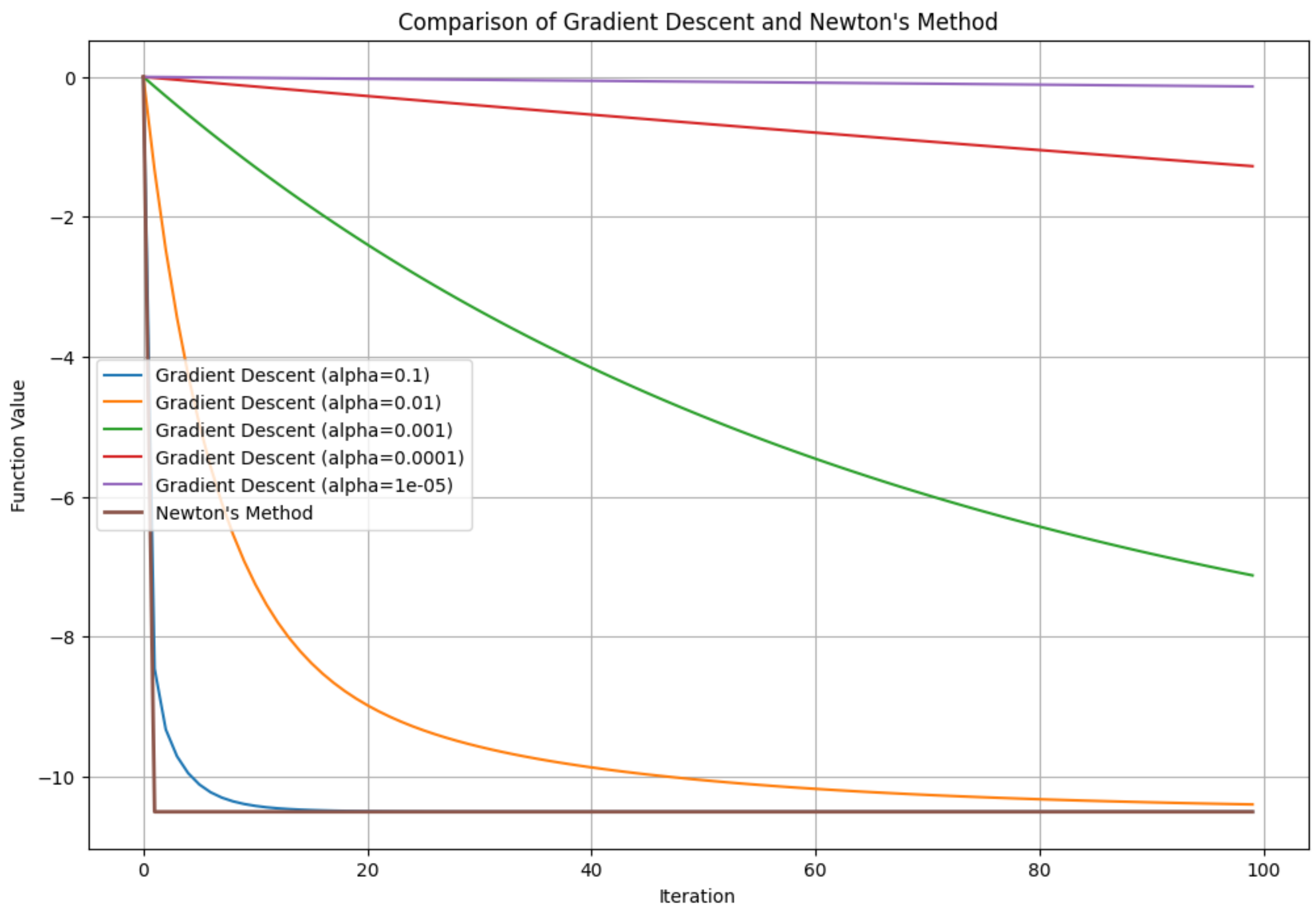
Optimum x*: [2,3,9,6,1]
Number of iterations: 1

2 Problem 2

2.1 Part 1,2

Final x (Gradient Descent, alpha=0.1): [1,1,0.99997344,1,1]
Final f(x) (Gradient Descent, alpha=0.1): -10.499999999564501
Final x (Gradient Descent, alpha=0.01): [0.86738044,0.86738044,0.63396766,0.99991981,0.99929483]
Final f(x) (Gradient Descent, alpha=0.01): -10.395021655806312
Final x (Gradient Descent, alpha=0.001): [0.1814332,0.1814332,0.09520785,0.59508352,0.50463553]
Final f(x) (Gradient Descent, alpha=0.001): -7.122103657993106
Final x (Gradient Descent, alpha=0.0001): [0.01980329,0.01980329,0.00995066,0.08610585,0.06762903]
Final f(x) (Gradient Descent, alpha=0.0001): -1.275408673359756
Final x (Gradient Descent, alpha=1e-05): [0.00199802,0.00199802,0.00099951,0.00896002,0.0069758]
Final f(x) (Gradient Descent, alpha=1e-05): -0.13655376884458403
Final x (Newton): [1,1,1,1,1]
Final f(x) (Newton): -10.5

Graph using linear scale.



2.2 Part 3

Initial x1: [0 0 0 0 0]
Final x1: [1, 1, 1, 1, 1]
Final f(x1): -10.5

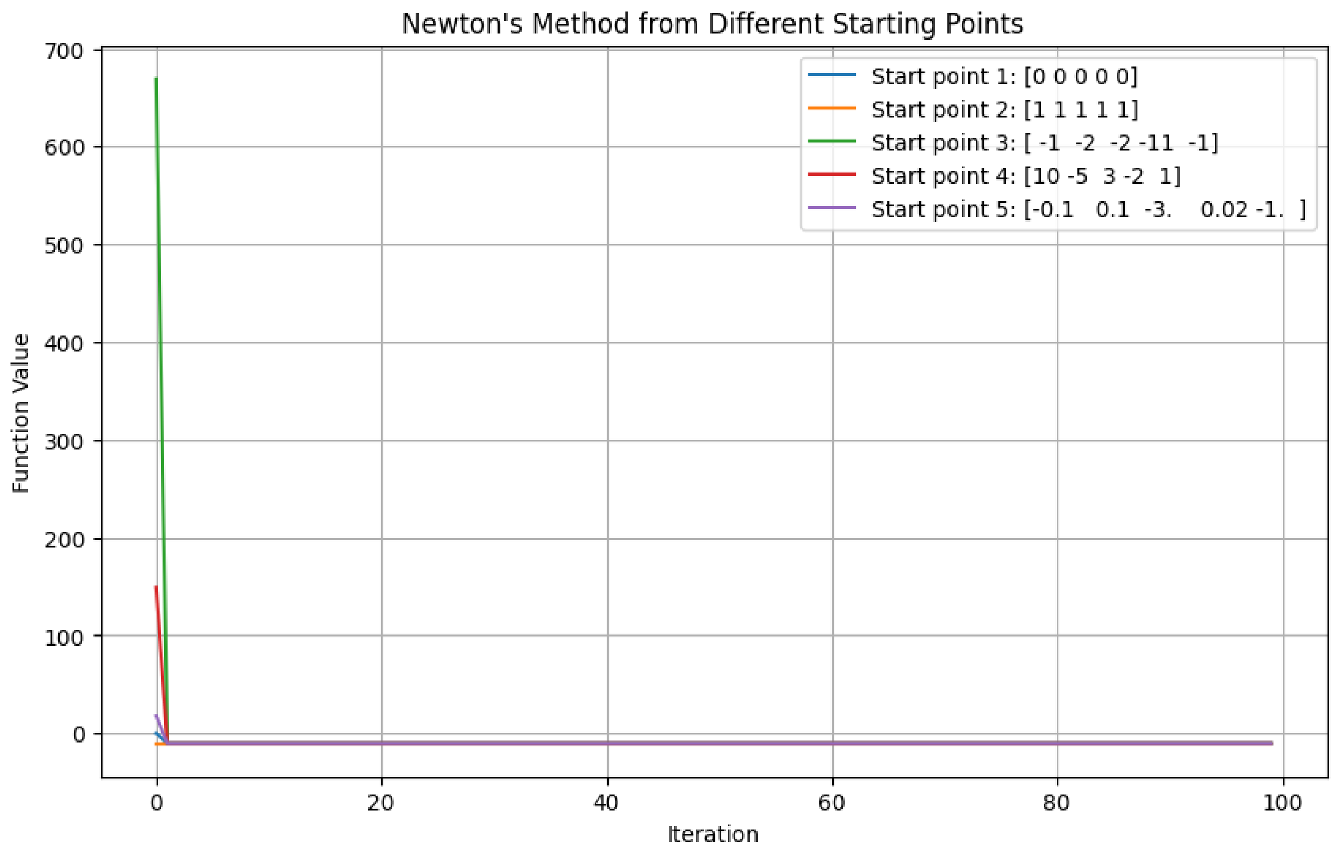
Initial x2: [1 1 1 1 1]
Final x2: [1, 1, 1, 1, 1]
Final f(x2): -10.5

Initial x3: [-1, -2, -2,-11 ,-1]
Final x3: [1, 1, 1, 1, 1]
Final f(x3): -10.5

Initial x4: [10,-5,3,-2,1]
Final x4: [1, 1, 1, 1, 1]
Final f(x4): -10.5

Initial x5: [-0.1,0.1,-3,0.02,-1]
Final x5: [1, 1, 1, 1, 1]
Final f(x5): -10.5

Graph using linear scale.



2.3 Part 4

The algorithm converges in exactly one iterarion for all initial points, so the function might be quadratic
Consider a general quadratic function in n-dimensions of the form:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$$

where:

- $\mathbf{x}$ is an n-dimensional vector,
- Q is an $n \times n$ symmetric matrix (the Hessian of $f(\mathbf{x})$),
- $\mathbf{b}$ is an n-dimensional vector,
- c is a scalar constant.

The gradient (first derivative) of $f(\mathbf{x})$ is:

$$\nabla f(\mathbf{x}) = Q\mathbf{x} + \mathbf{b}$$

The Hessian (second derivative) is the constant matrix Q:

$$\nabla^2 f(\mathbf{x}) = Q$$

which is positive definite so the function is convex, thus putting gradient zero will give the global minima.
Newton's method updates the vector $\mathbf{x}_n$ using the formula:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \left(\nabla^2 f(\mathbf{x}_n)\right)^{-1} \nabla f(\mathbf{x}_n)$$

Substitute the gradient $\nabla f(\mathbf{x}_n) = Q\mathbf{x}_n + \mathbf{b}$ and the Hessian $\nabla^2 f(\mathbf{x}_n) = Q$ into the update formula:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - Q^{-1}\left(Q\mathbf{x}_n + \mathbf{b}\right)$$

$$\mathbf{x}_{n+1} = -Q^{-1}\mathbf{b}$$

The update formula gives $\mathbf{x}_{n+1} = -Q^{-1}\mathbf{b}$, which is the **exact minimizer** of the quadratic function, as solving $\nabla f(\mathbf{x}) = 0$ yields:

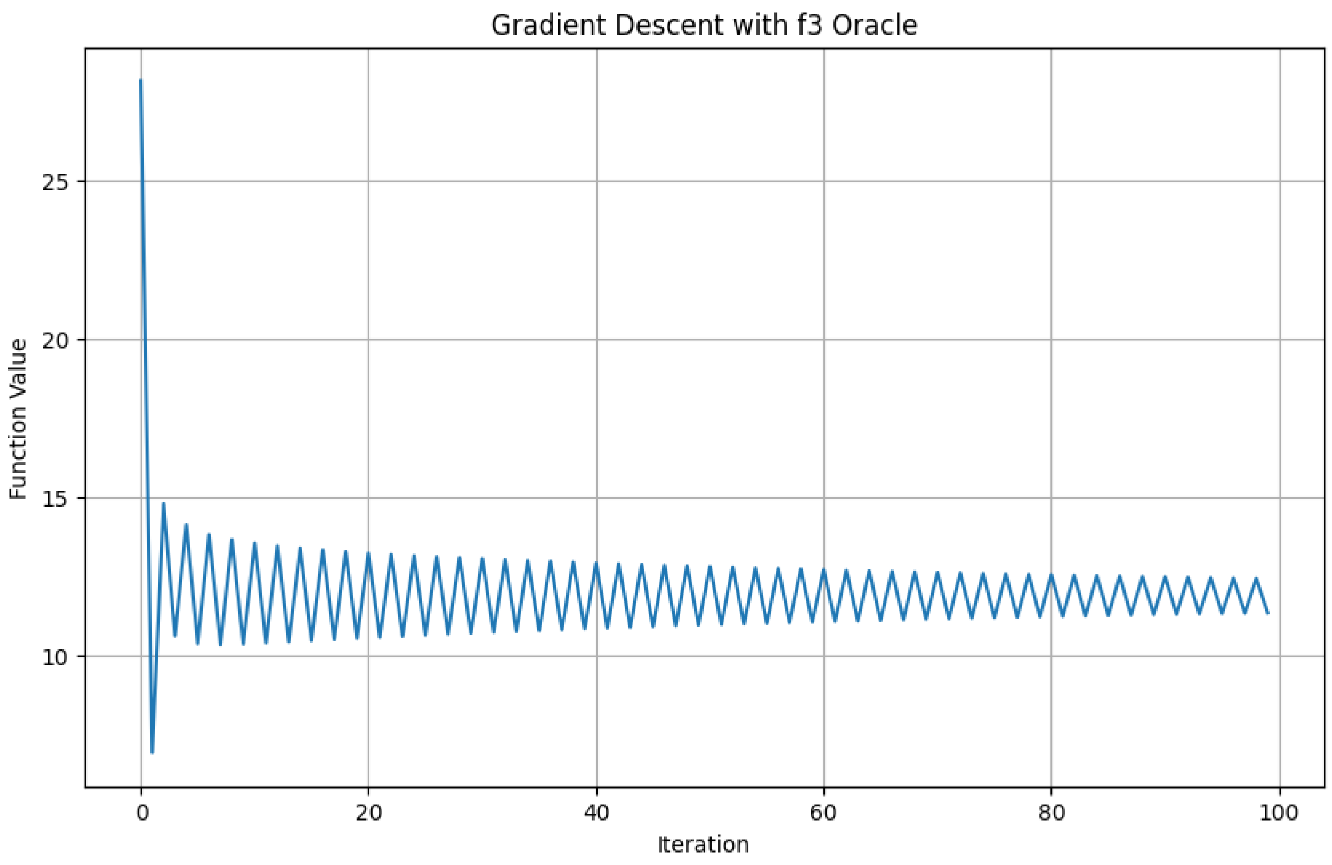
$$Q\mathbf{x} + \mathbf{b} = 0 \quad \Rightarrow \quad \mathbf{x} = -Q^{-1}\mathbf{b}$$

Thus, Newton's method converges to the exact solution in one iteration for any initial guess $\mathbf{x}_0$, since it directly computes the minimizer of the quadratic function.

3 Problem 3

3.1 Part 1

Initial x: [1,1,1,1,1]
Final x: [-4.79441025e-01 ,2.17408789e-17,3.19757550e-05,-4.79441025e-01,-3.44407938e-01]
Final f(x): 11.359296199638138
Best f(x): 6.944269753252721



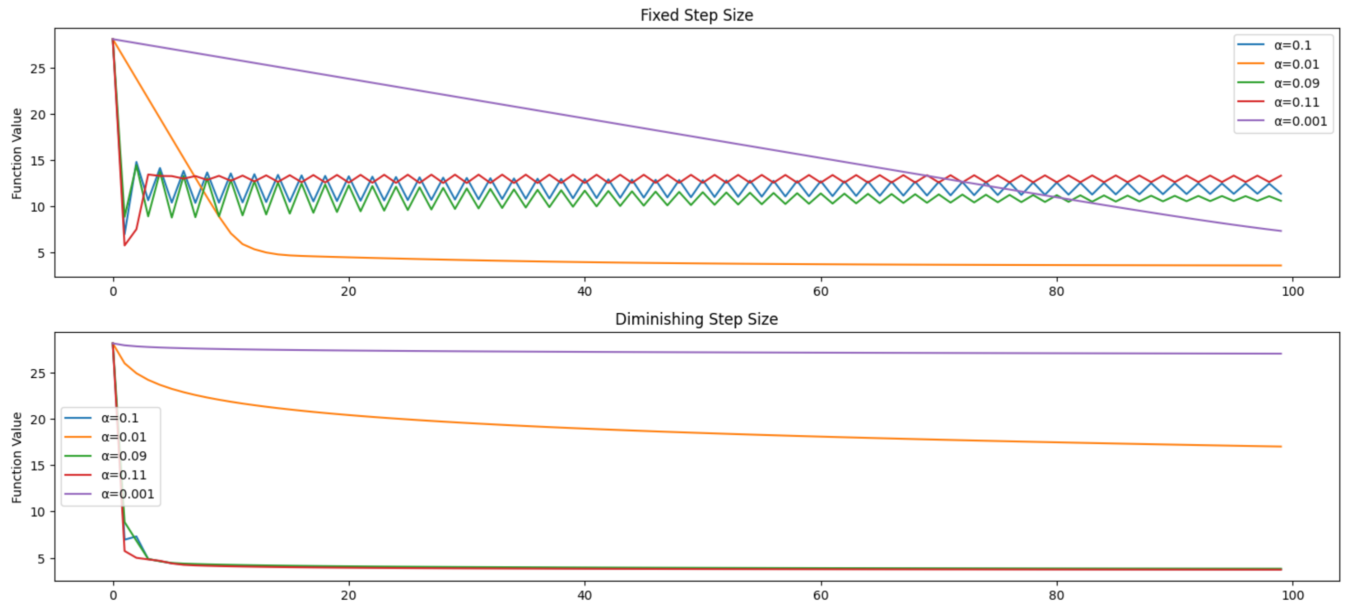
3.2 Part 2

Possible Reasons for Oscillations: Step-size too large: When the step-size is large, gradient descent can overshoot the minimum and oscillate around the true minimum instead of converging smoothly.
Taking a smaller step size can work.
Decreasing the step size after each iteration can work

I ran the algorithm with the same initial point but with step sizes = [0.1, 0.01, 0.09, 0.11,0.001]. Fixed and diminishing step size.

For step sizes close to 0.1 i.e 0.09 and 0.11 it still oscillates.(for fixed step size).

And for diminishing step size, it starts to oscilate and them it starts to converge but for some step sizes the convergence is slow.
Note that converging here means converging to some local minima. In diminishing step size, $\alpha = 0.001, 0.01$ converges to different local mininimas.



3.3 Part 3

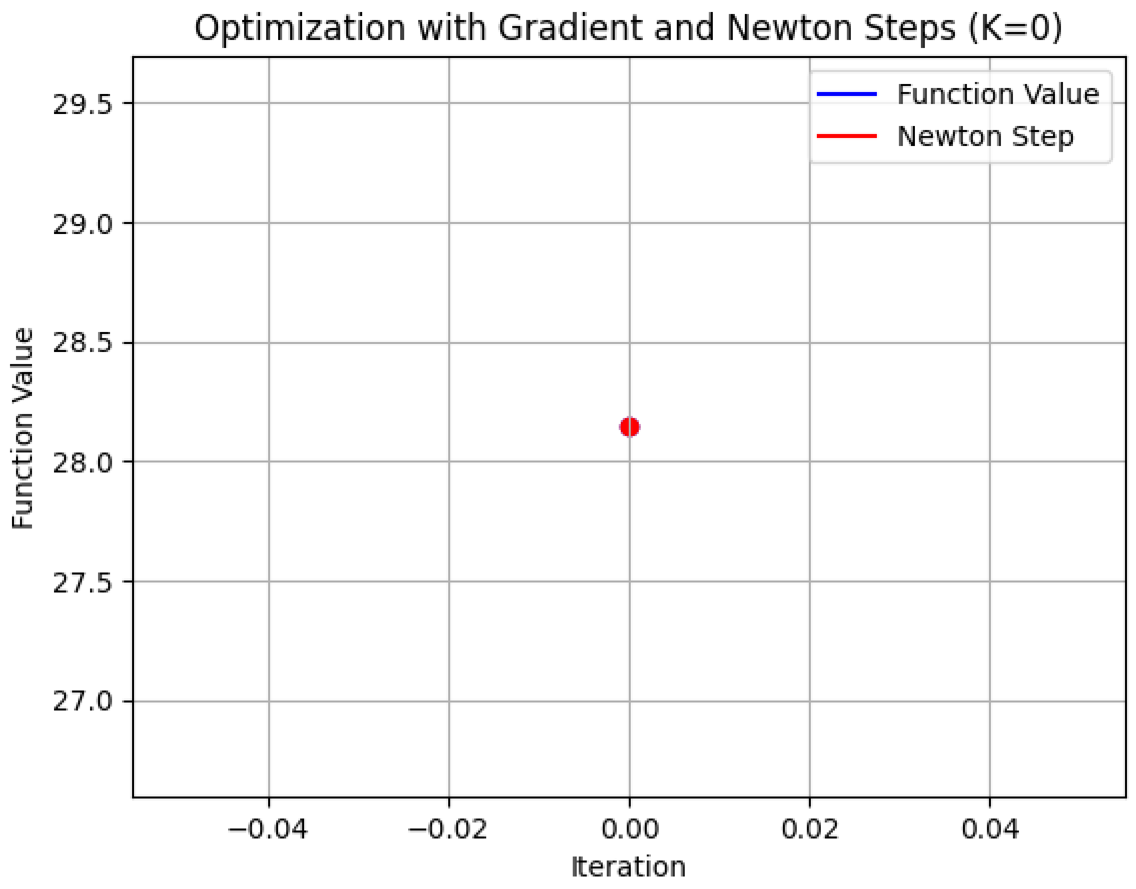
First 10 function values:
Iteration 0: 28.145078800949115
Iteration 1: inf
Iteration 2: inf
Iteration 3: nan
Iteration 4: nan
Iteration 5: nan
Iteration 6: nan
Iteration 7: nan
Iteration 8: nan
Iteration 9: nan

Final x: [nan,nan,-2.28224557e-17,nan,nan]
Final f(x): nan

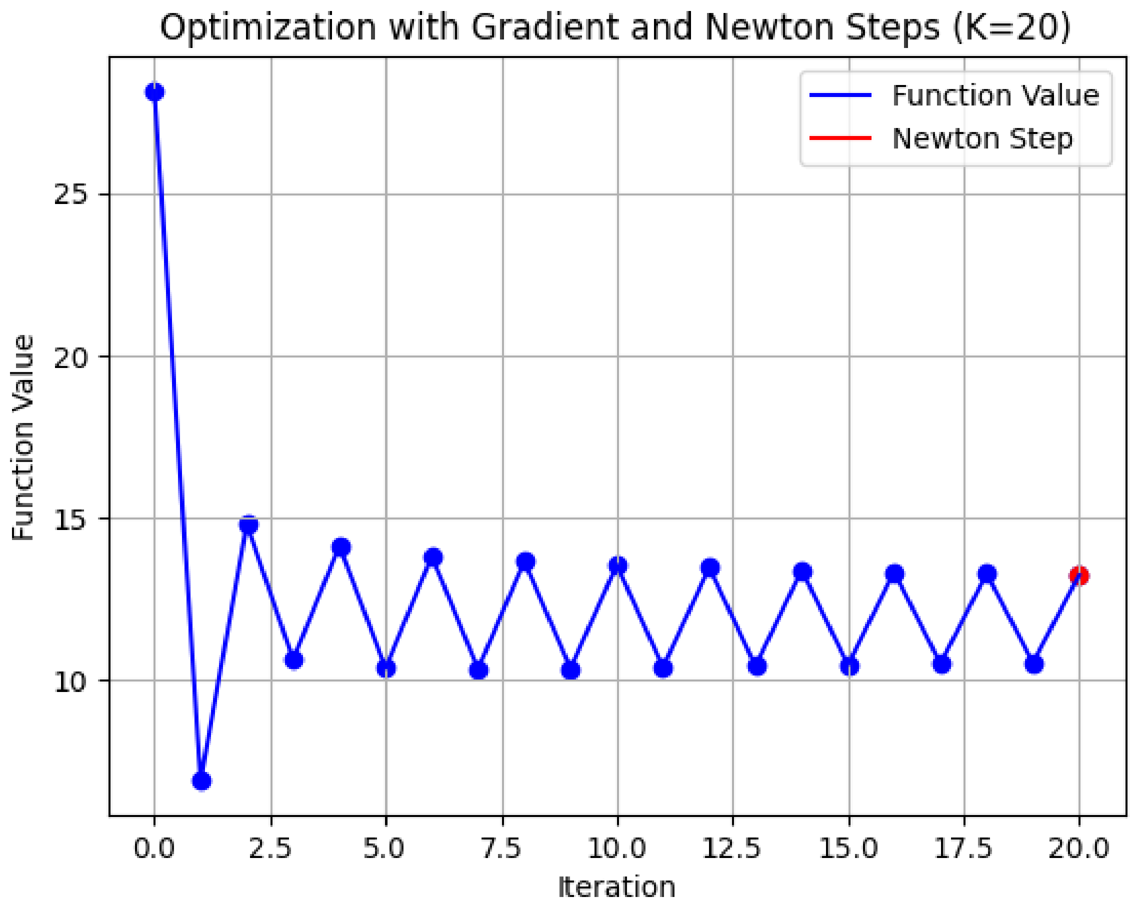
The algorithm does not converge. The algorithm gives an output for 0th iteration, after that $Hess^{-1}\nabla_f$ contains $-\infty$. And after an iteration we encounter an undefined operation and get **nan**.The reason is that the initial guess is too bad so that the algorithm doesnt converge.

3.4 Part 4

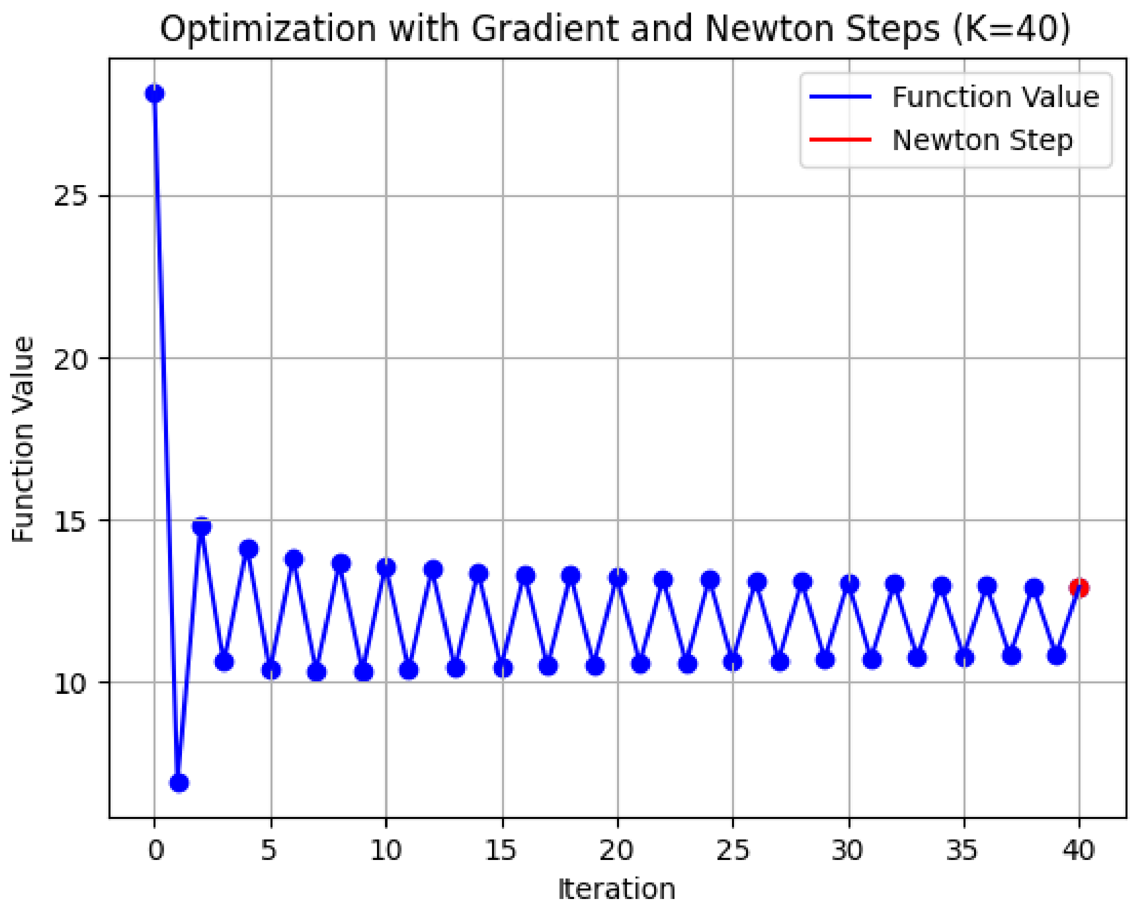
Let the cost of gradient update be 1 and cost of newton update = 25.



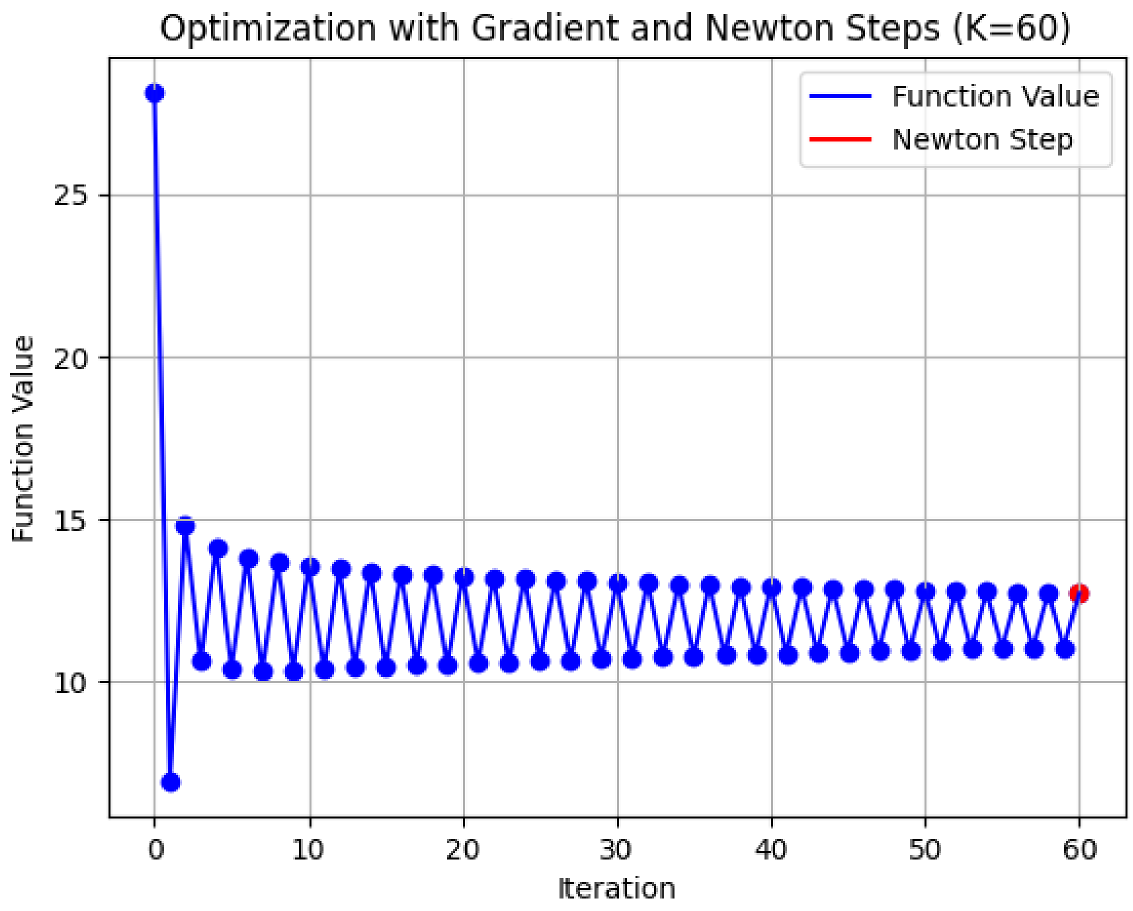
```
K=0
Cost: 2500
Final Function Value: nan
Final x: [nan,nan,-2.28224557e-17,nan,nan]
```



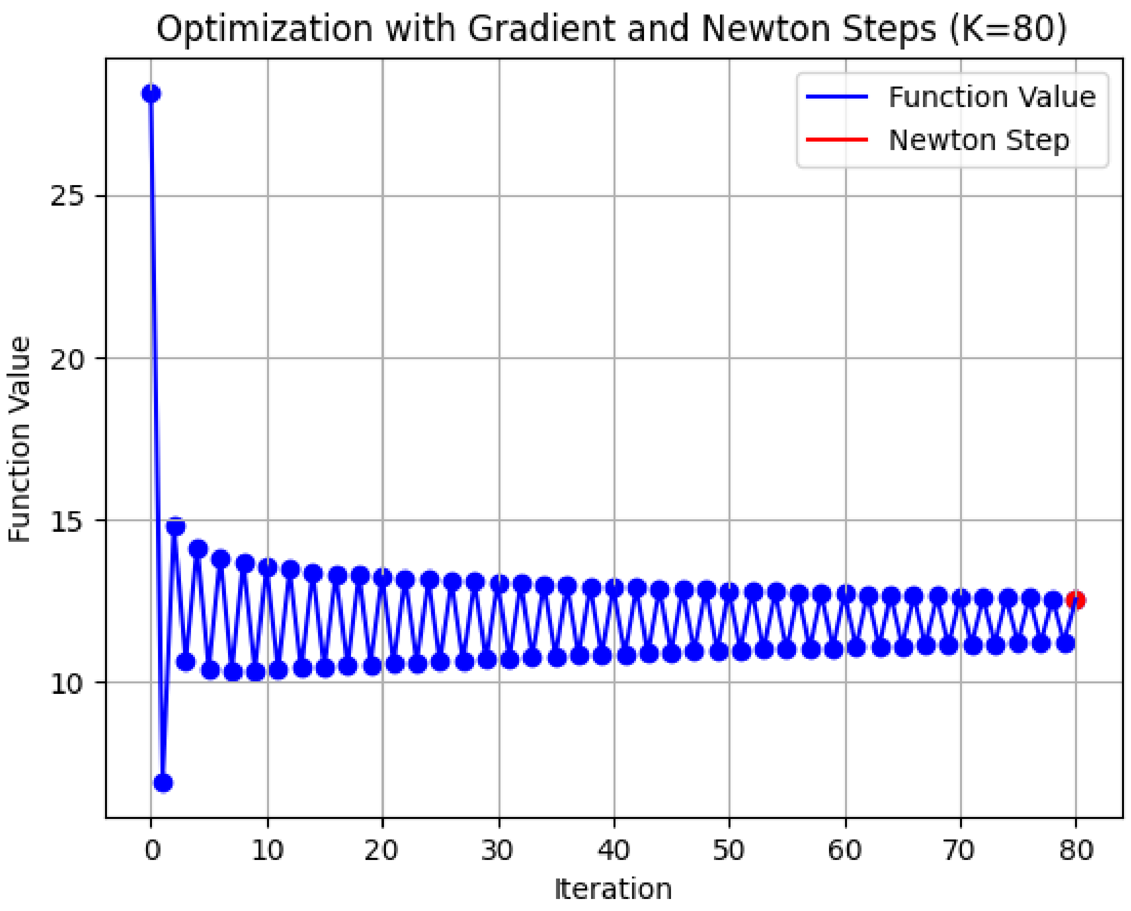
```
K=20
Cost: 2020
Final Function Value: nan
Final x: [nan,2.01933979e-18,-1.28553951e-17,nan,nan]
```



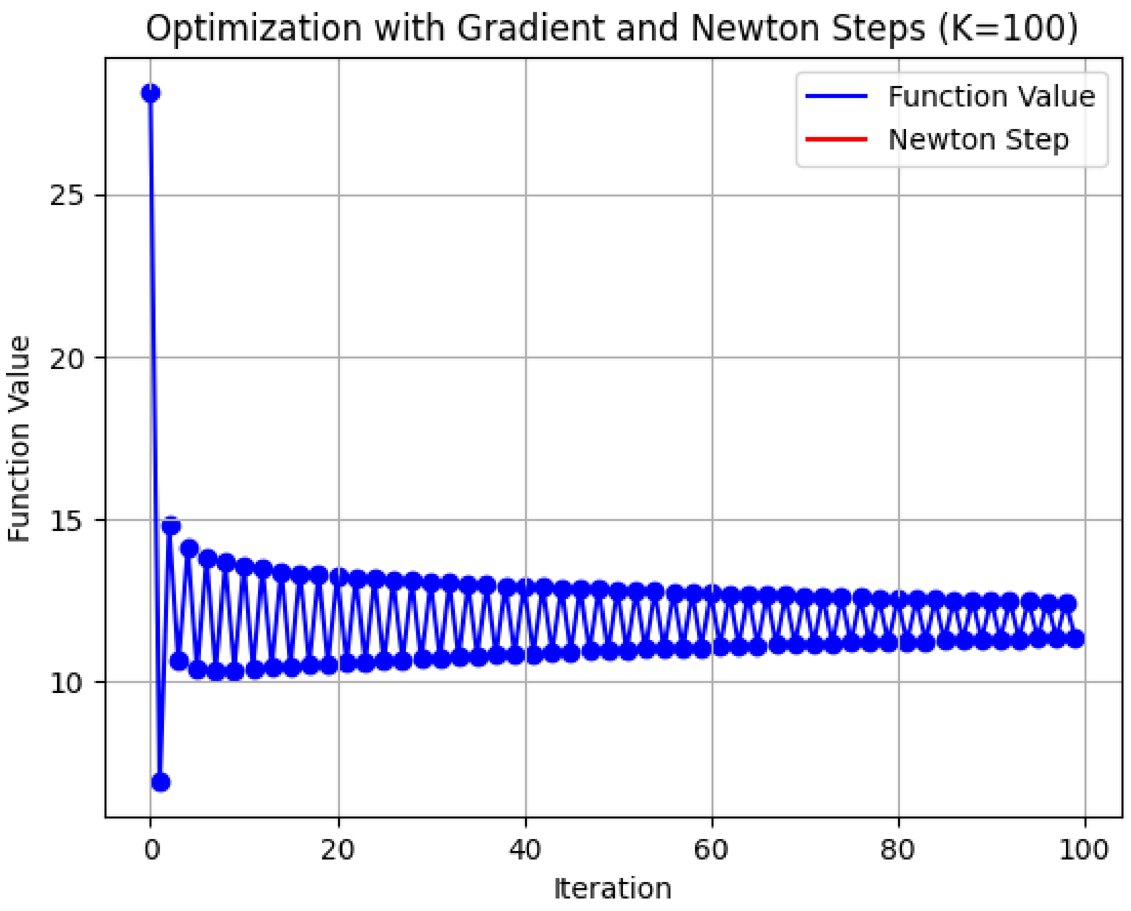
```
K=40
Cost: 1540
Final Function Value: nan
Final x: [nan,9.32644155e-18,4.62018356e-17,nan,nan]
```



```
K=60
Cost: 1060
Final Function Value: nan
Final x: [nan,5.08753358e-18,5.22293585e-18,nan,nan]
```



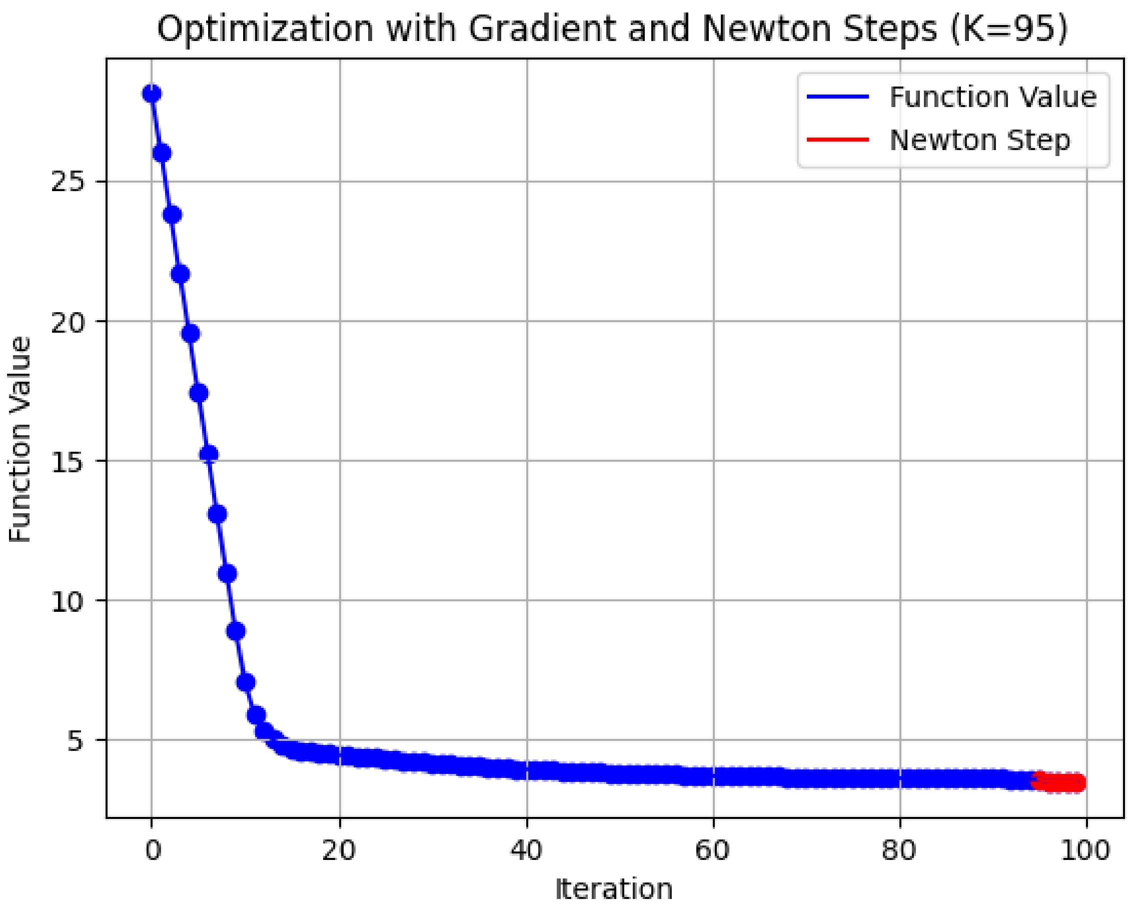
```
K: 80
Cost: 580
Final Function Value: nan
Final x: [nan,2.17408789e-17,5.69220851e-18,nan,nan]
```



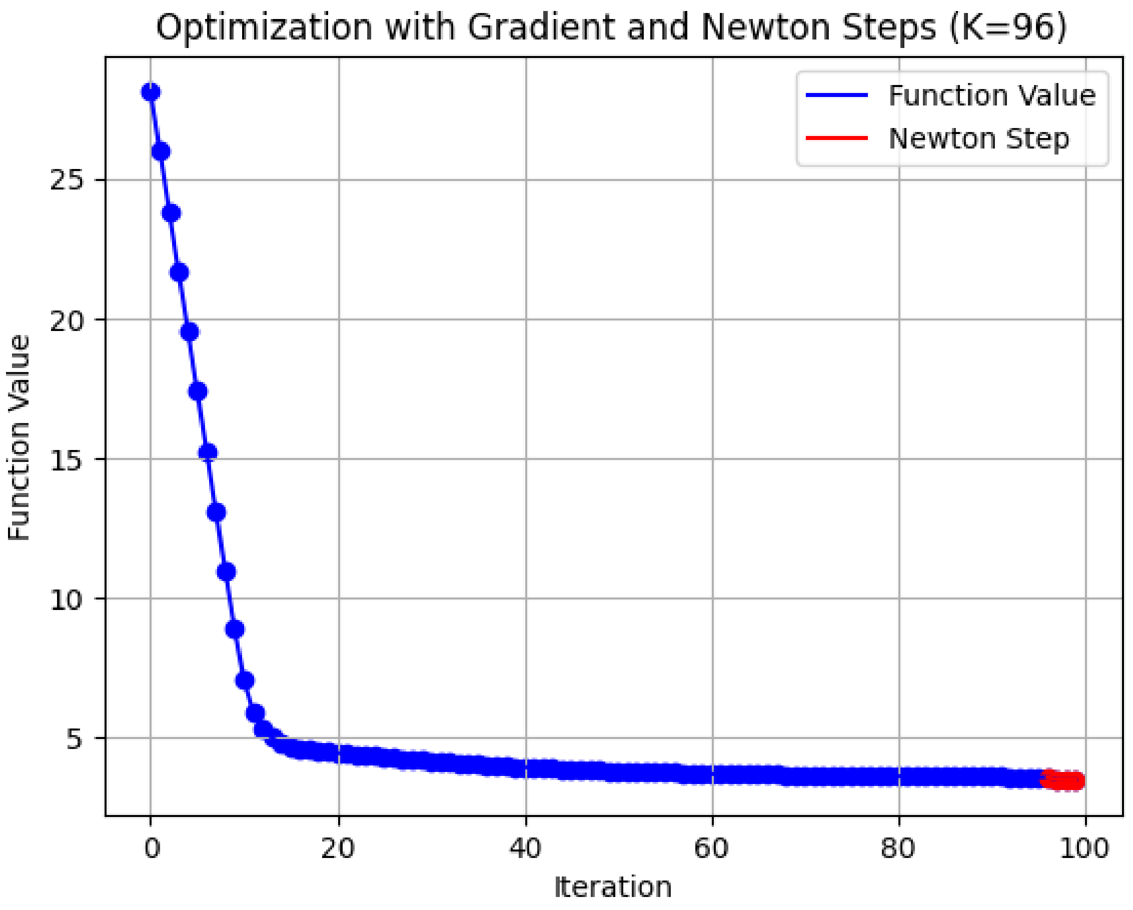
```
K: 100
Cost: 100
Final Function Value: 11.359296199638138
Final x: [-4.79441025e-01, 2.17408789e-17, 3.19757550e-05, -4.79441025e-01, -3.44407938e-01]
```

For any K the execution of one Hessian inverse blows up the values of x.

Now we will decrease the step size to 0.01
We will plot for high values of k, to get minimum cost.

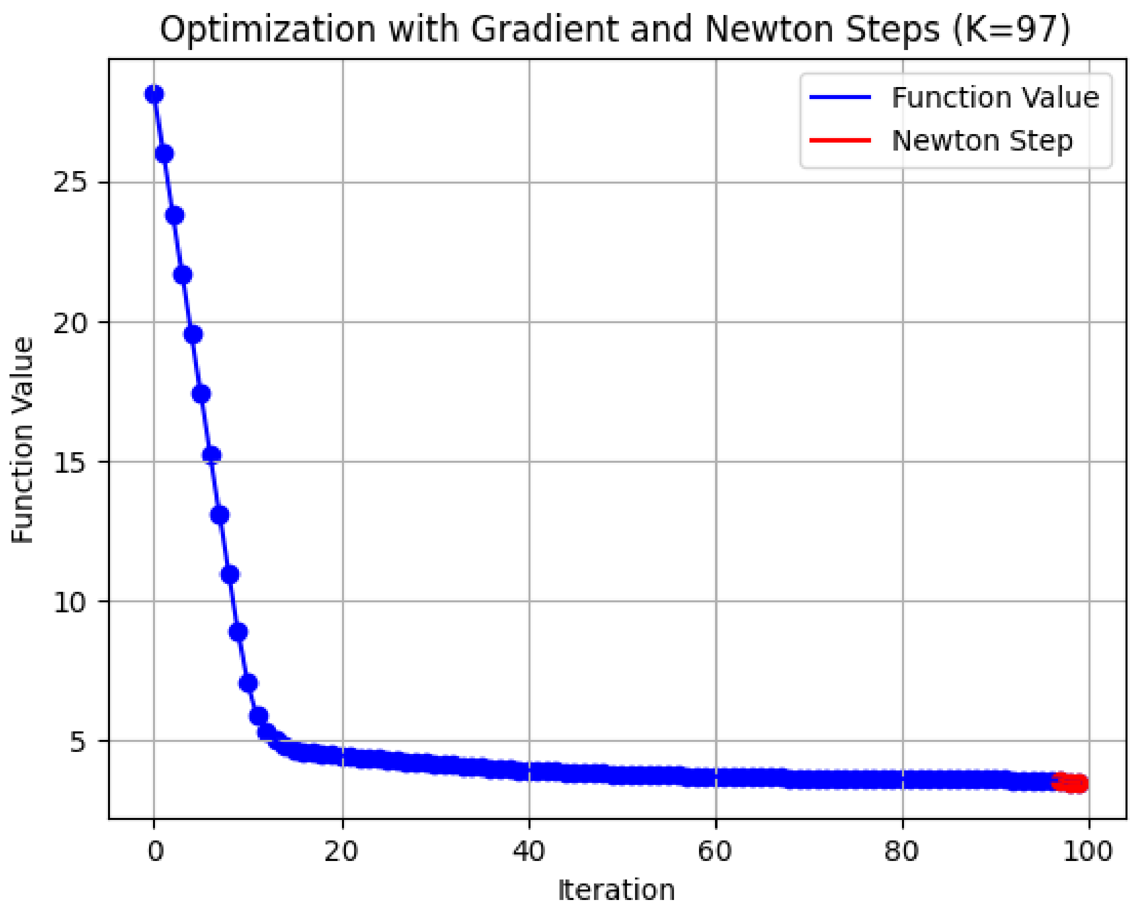


K: 95
Cost: 220
Final Function Value: 3.4657359027997265
Final x: [-1.12433801e-18 ,8.94646618e-19, -8.07730361e-18, -1.12433801e-18, 3.19769361e-18]
Gradient of f at x : [0, 0, 0, 0, 0]

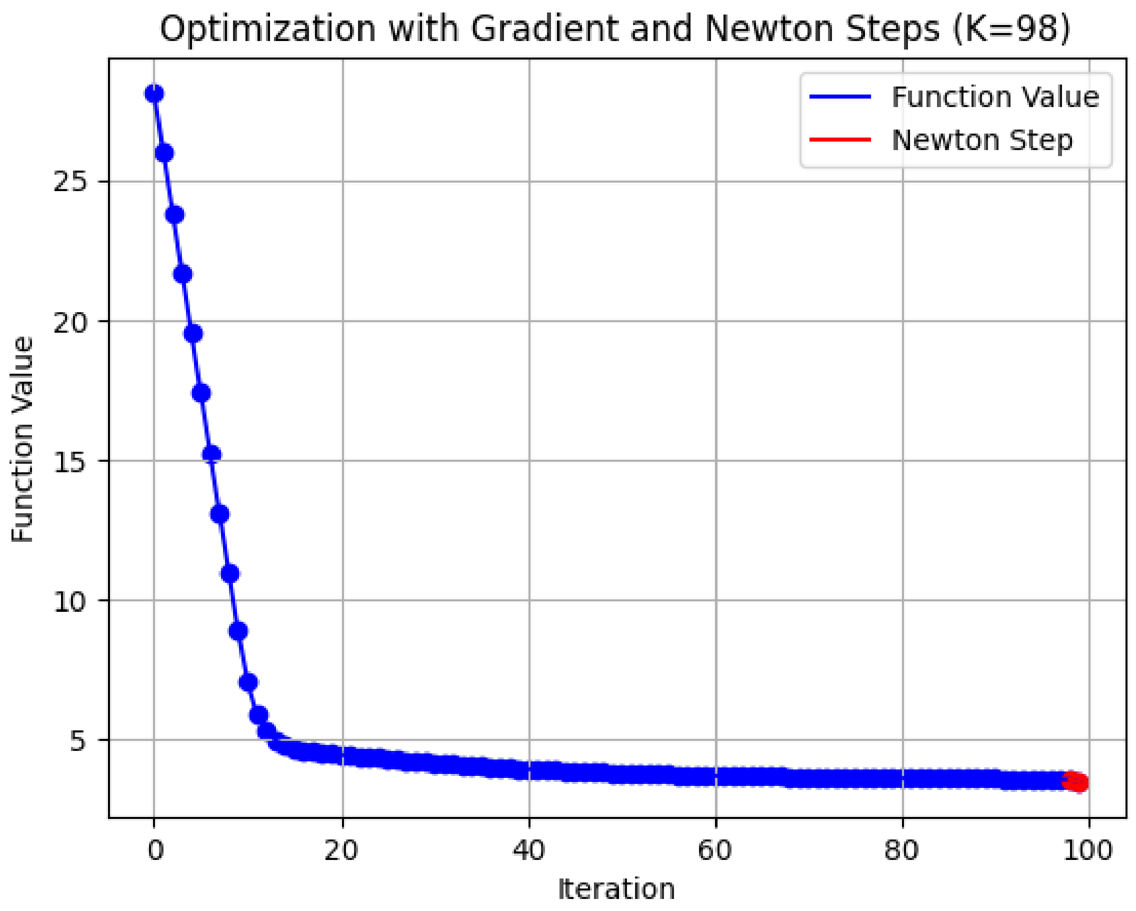


K: 96

```
Cost: 196
Final Function Value: 3.4657359027997265
Final x: [-1.12433801e-18 ,2.16849067e-18, 1.75911818e-17, -1.12433801e-18,3.19769361e-18]
Gradient of f at x = [0, 0, 0, 0, 0]
```



```
K: 97
Cost: 172
Final Function Value: 3.4657359091356623
Final x: [-1.12433801e-18,-5.44807448e-18,-9.50957439e-13,-1.12433801e-18,3.19769361e-18]
Gradient of f at x=[ 0.00000000e+00,0.00000000e+00,-9.50961532e-13,0.00000000e+00,0.00000000e+00]
```



K: 98
Cost: 148
Final Function Value: 3.467175767157041
Final x: [-1.12433801e-18,3.15347874e-12,1.03154665e-04,-1.12433801e-18,3.19769361e-18]
Gradient of f at x = [0.00000000e+00,1.26139099e-11,1.03154665e-04,0.00000000e+00,0.00000000e+00]

For this step size minimum cost is attained when K=96

4 Problem-4 :

4.1 Part 1

We approximate the Hessian matrix H_k as a scalar multiple of the identity matrix, $H_k = \gamma_k I$. The secant condition for Quasi-Newton methods is given by:

$$H_{k+1} s_k = y_k$$

where $s_k = x_{k+1} - x_k$ is the change in the iterate, and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ is the change in the gradient. Substituting $H_{k+1} = \gamma_{k+1} I$ into the secant condition, we get:

$$\gamma_{k+1} s_k = y_k$$

To solve for γ_{k+1}, multiply both sides by s_k^T:

$$\gamma_{k+1} s_k^T s_k = s_k^T y_k$$

Thus, the update for γ_{k+1} is:

$$\gamma_{k+1} = \frac{s_k^T y_k}{s_k^T s_k}$$

Instead if we multiply both sides by y_k^T:

$$y_k^T y_k = \gamma_{k+1} y_k^T s_k$$

Thus, the update for γ_{k+1} is:

$$\gamma_{k+1} = \frac{y_k^T y_k}{y_k^T s_k}$$

4.2 Part 2

```
x* (GD, step=0.01): [0.86738044 0.86738044 0.63396766 0.99991981 0.99929483]
x* (GD, step=0.1): [1. 1. 0.99997344 1. 1. ]
x* (GD, step=0.5): [ 1.00000000e+00 1.00000000e+00 1.00000000e+00 -2.55155207e+54 -6.22301528e+39]
x* (QN Scalar 1): [1. 1. 1. 1. 1.]
x* (QN Scalar 2): [1. 1. 1. 1. 1.]
```