최적화개론 Project1

산업공학과 2020067356 장윤수

CONTENTS

1. Problem1 and solution

Problem 1

Problem 1

$$f(x_1, x_2) = 1 + 2x_1 + 3(x_1^2 + x_2^2) + 4x_1x_2$$

$$f(x_1, x_2) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 3 & 3 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 16 & 23 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \pi^2$$

$$f(x, y) = 3(x^2 + y^2) + 4xy + 5x + 6y + 7$$

Problem 1

1. Show that the functions are convex.
2. Obtain the optimal solution to the above problems using CVX
3. Obtain the optimal solution and optimal cost for each function by your hand
4. Use Python to design the gradient descent algorithm and find the optimal solution and optimal value. Also, obtain the convergence plot
5. Use Python to design the steepest (optimal step size) gradient descent and find the optimal solution and optimal value. Also, obtain the convergence plot
6. Let x^* be the optimal solution obtained by 2. Let $x^*_{Gradient}$ be the optimal solution obtained by 2. Let $x^*_{SteepestGradient}$ be the optimal solution obtained by 2. Plot the convergence plot of $\|x^* - x^*_{Gradient}\|$, $\|x^* - x^*_{SteepestGradient}\|$
7. Discuss the convergence speed in the convergence plots in 5 and 6

11

First Equation

$$f(x_1, x_2) = 1 + 2x_1 + 3(x_1^2 + x_2^2) + 4x_1x_2$$

- Step 1: Convexity Check
  - The Hessian matrix is H = [6, 4; 4, 6].
  - Its eigenvalues are 10 and 2, both positive.

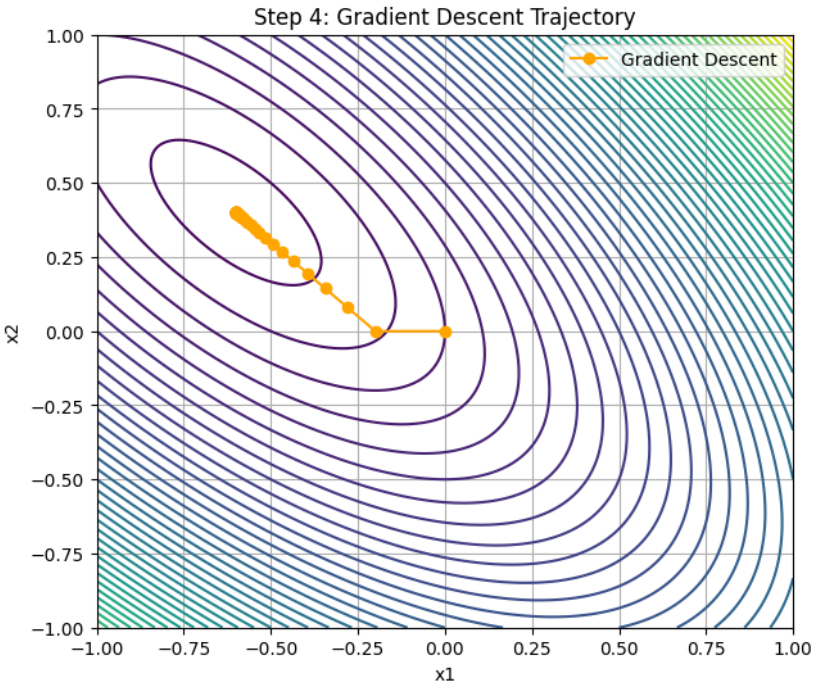
- Step 2: Optimal Solution Using CVX
  - Rewritten as: $f(x) = (1/2)x^T Qx + c^T x + r$
  - Where Q = [6, 4; 4, 6]/2, c = [2, 0], r = 1
  - Result: $x^* \approx [-0.6000, 0.4000]$ and $f(x^*) \approx 0.4000$

- Step 3: Hand Calculation
  - Result: $x_1 = -3/5$, $x_2 = 2/5$, $f(x^*) = 2/5 = 0.4$

First Equation

$$f(x_1, x_2) = 1 + 2x_1 + 3(x_1^2 + x_2^2) + 4x_1x_2$$

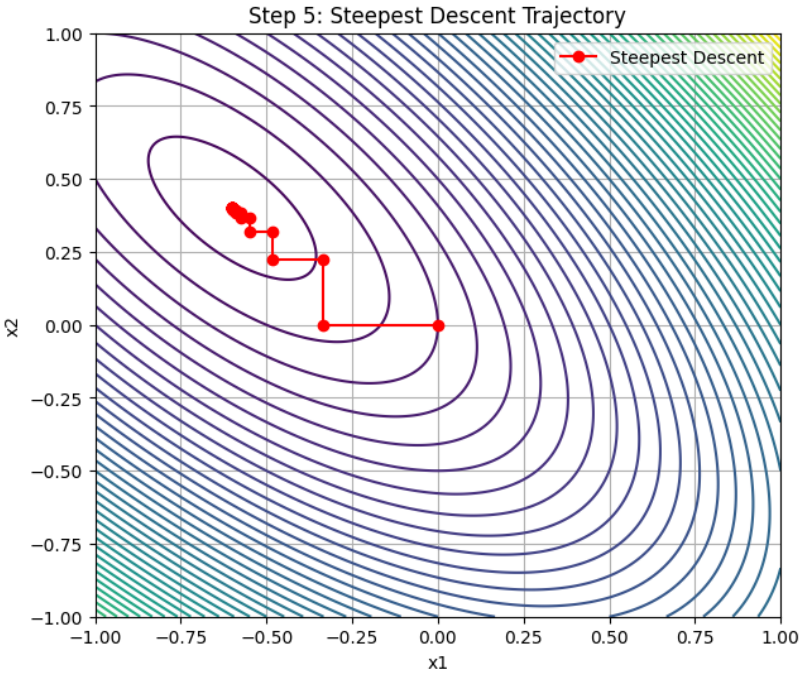
- Step 4: Gradient Descent
  - Start at x_0 = [0, 0], learning rate = 0.1
  - Result: x* ≈ [-0.599997, 0.399997], f(x*) ≈ 0.4000



First Equation

$$f(x_1, x_2) = 1 + 2x_1 + 3(x_1^2 + x_2^2) + 4x_1x_2$$

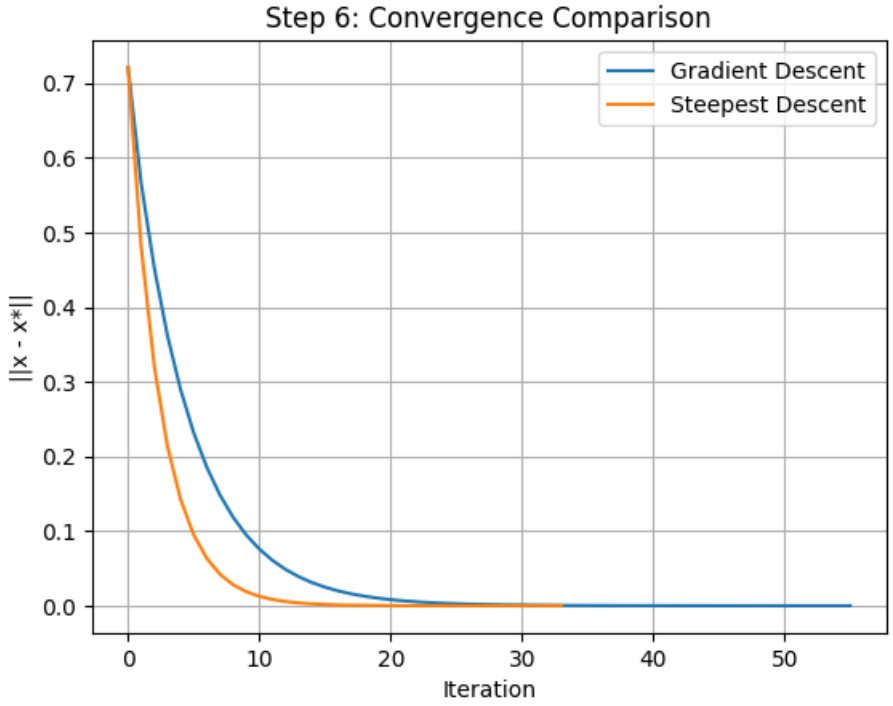
- Step 5: Steepest Descent (Optimal Step Size)
  - Optimal α is computed at each step.
  - Result: x* ≈ [-0.599999, 0.399999] f(x*) ≈ 0.4000



First Equation

$$f(x_1, x_2) = 1 + 2x_1 + 3(x_1^2 + x_2^2) + 4x_1x_2$$

- Step 6: Convergence Comparison
  - Plotted ||x(k) - x*|| for both methods.
  - Steepest descent converges faster than fixed step gradient descent.



First Equation

$$f(x_1, x_2) = 1 + 2x_1 + 3(x_1^2 + x_2^2) + 4x_1x_2$$

- Step 7: Convergence Speed Discussion
  - Fixed step GD converges slowly.
  - Steepest descent uses optimal step size and converges faster and more stably.

Second Equation

$$f(x_1, x_2) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 3 & 3 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 16 & 23 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \pi^2$$

- Step 1: Convexity Check
  - The Hessian matrix is H = [6, 4; 4, 6].
  - Its eigenvalues are 10 and 2, both positive.

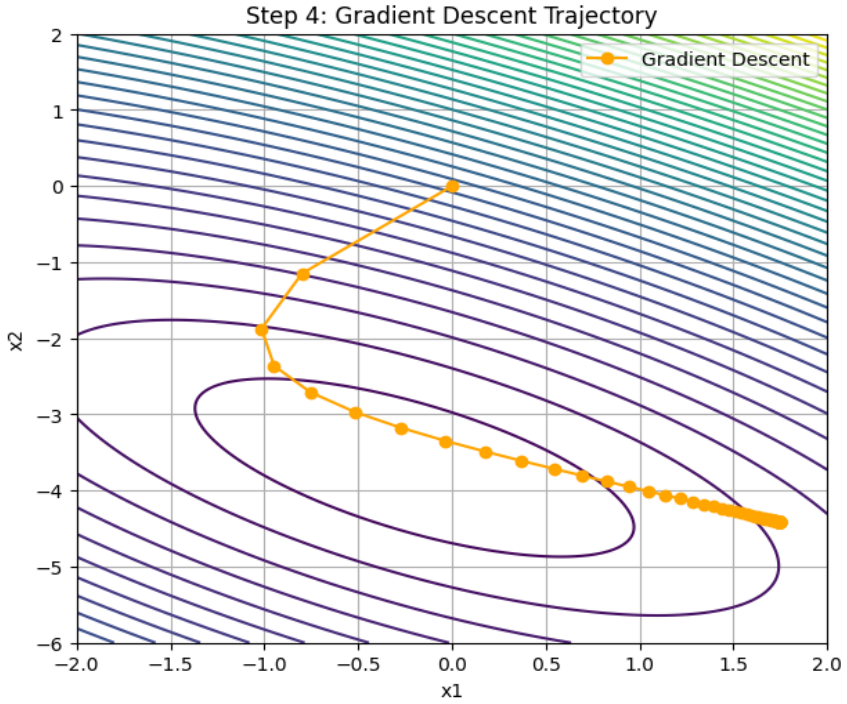
- Step 2: Optimal Solution Using CVX
  - Symmetrized Q: [3, 2; 2, 3]
  - Optimal solution: x* ≈ [-0.2000, -3.7000]
  - Optimal value: f(x*) ≈ -34.2804

- Step 3: Hand Calculation
  - Gradient set to 0 → solve manually.
  - $x_1 = -1/5$, $x_2 = -37/10$
  - $f(x^*) = -883/20 + \pi^2 \approx -34.15$

Second Equation

$$f(x_1, x_2) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 3 & 3 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 16 & 23 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \pi^2$$

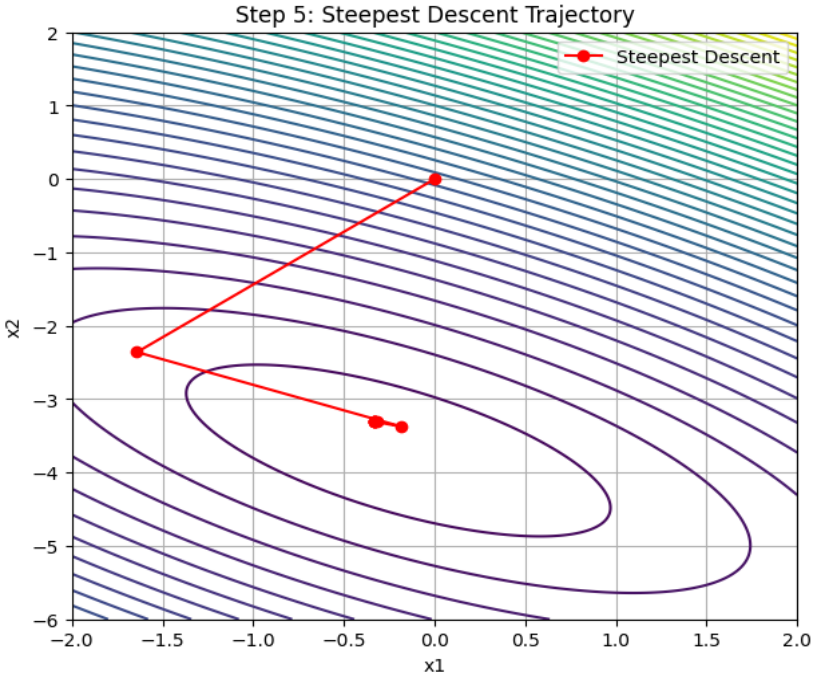
- Step 4: Gradient Descent
  - Initial point: [0, 0], α = 0.05
  - Result: Approx. x* ≈ [1.75, -4.42], f(x*) ≈ -26.92



Second Equation

$$f(x_1, x_2) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 3 & 3 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 16 & 23 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \pi^2$$

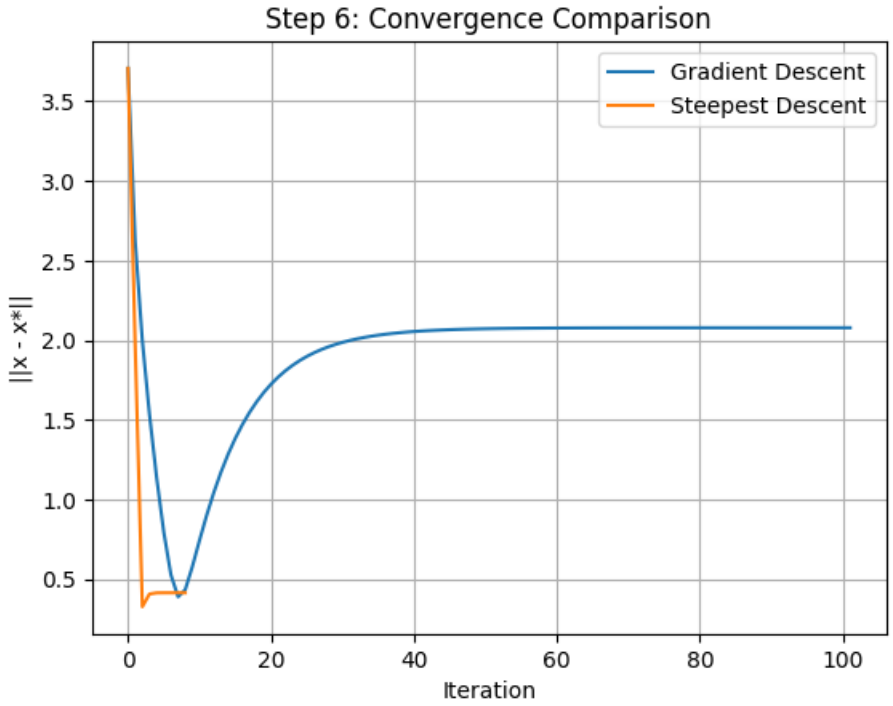
- Step 5: Steepest Descent (Optimal Step Size)
  - Optimal α is computed at each step.
  - Result: x* ≈ [-0.3312, -3.3070] f(x*) ≈ -33.97



Second Equation

$$f(x_1, x_2) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 3 & 3 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 16 & 23 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \pi^2$$

- Step 6: Convergence Comparison
  - Plotted ||x(k) - x*|| for GD and Steepeset GD.
  - Steepest descent converges faster and closer to CVX solution.



Second Equation

$$f(x_1, x_2) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 3 & 3 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 16 & 23 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \pi^2$$

- Step 7: Convergence Speed Discussion
  - Steepest GD reaches near-optimal faster.
  - Fixed-step GD slower and less accurate.

Third Equation

$$f(x, y) = 3(x^2 + y^2) + 4xy + 5x + 6y + 7$$

- Step 1: Convexity Check
  - The Hessian matrix is H = [6, 4; 4, 6].
  - Its eigenvalues are 10 and 2, both positive.

- Step 2: Optimal Solution Using CVX
  - Rewrote the function in the quadratic form $f(x) = \frac{1}{2} x^t Q x + c^t x + r$
  - Q = [6, 4; 4, 6] / 2, c = [5, 6], r = 7
  - Optimal solution: $x^* \approx [-0.7, -0.5]$
  - Optimal value: $f(x^*) \approx -1.6$

Third Equation

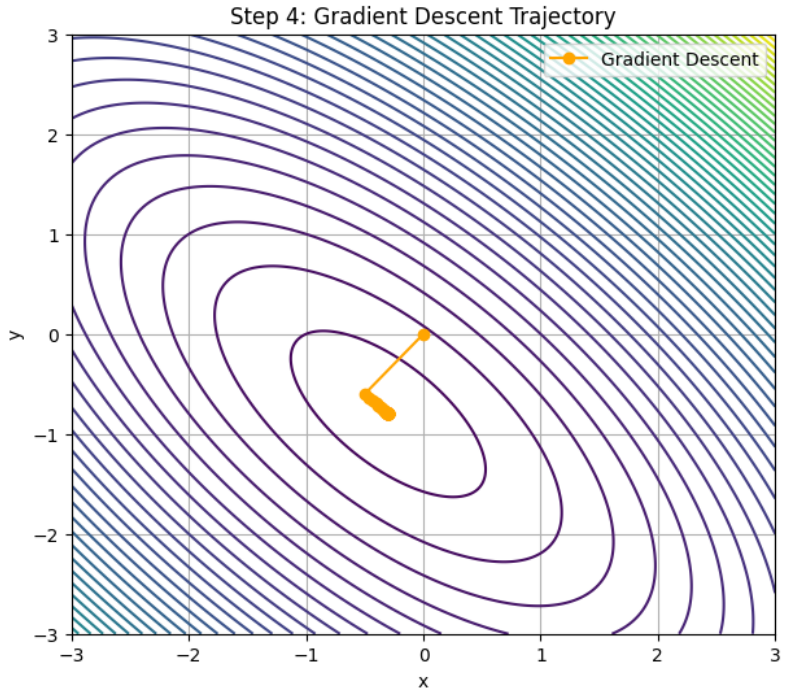
$$f(x, y) = 3(x^2 + y^2) + 4xy + 5x + 6y + 7$$

- Step 3: Hand Calculation
  - solved the gradient equation: ∇f(x, y) = [6x + 4y + 5, 6y + 4x + 6] = 0
  - x = -7/10, y = -1/2
  - f(x*) = -8/5 = -1.6
  - This matches the result from CVX

Third Equation

$$f(x, y) = 3(x^2 + y^2) + 4xy + 5x + 6y + 7$$

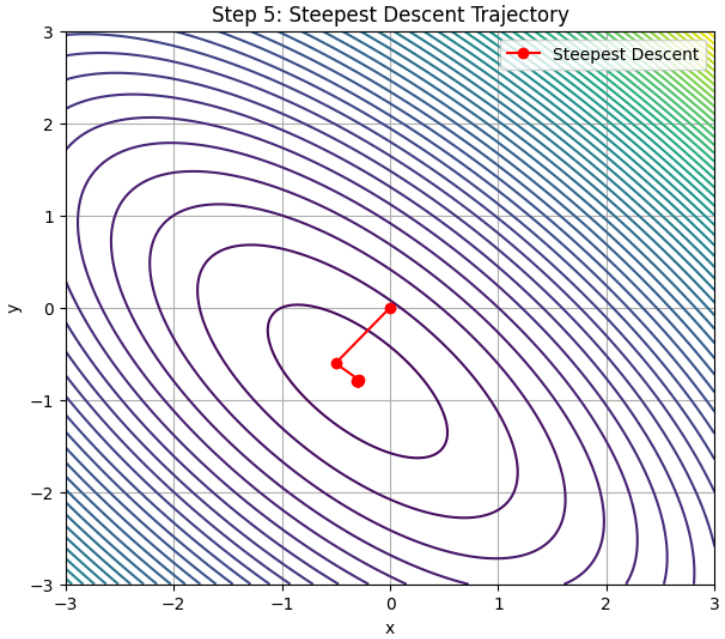
- Step 4: Gradient Descent
  - Initial point: [0, 0], Learning rate: 0.1
  - Result: Approx. x ≈ [-0.699997, -0.499998], f(x) ≈ -1.5999999



Third Equation

$$f(x, y) = 3(x^2 + y^2) + 4xy + 5x + 6y + 7$$

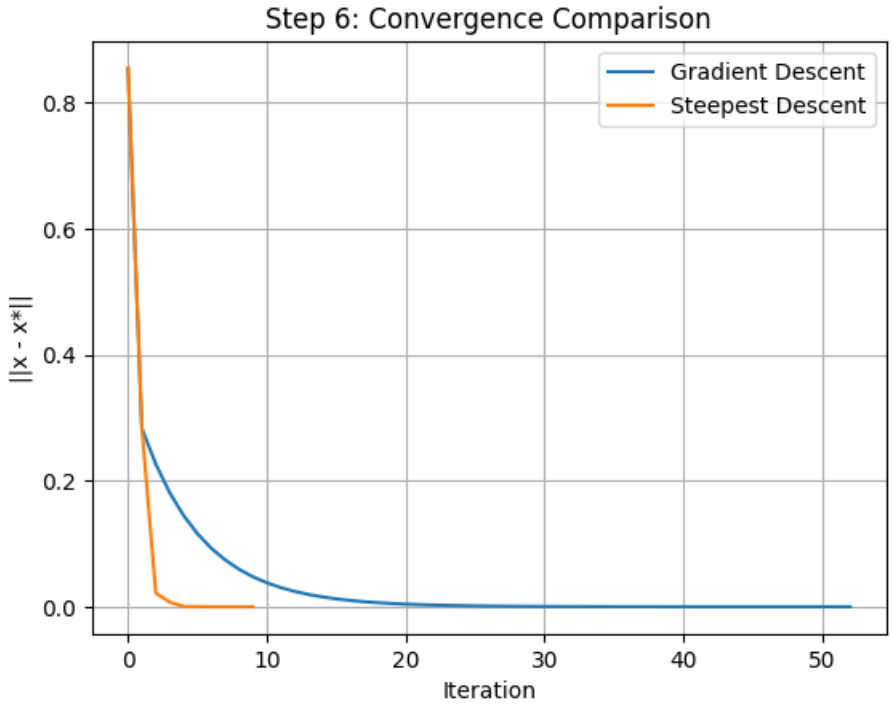
- Step 5: Steepest Descent (Optimal Step Size)
  - Optimal α is computed at each step.
  - Result: x ≈ [-0.7, -0.5] f(x) ≈ -1.6
  - Steepest Descent converged faster and with higher precision than fixed-step Gradient Descent.



Third Equation

$$f(x, y) = 3(x^2 + y^2) + 4xy + 5x + 6y + 7$$

- Step 6: Convergence Comparison
  - Plotted ||x(k) - x*|| for GD and Steepeset GD.
  - Steepest descent converges faster than GD.



Third Equation

$$f(x, y) = 3(x^2 + y^2) + 4xy + 5x + 6y + 7$$

- Step 7: Convergence Speed Discussion
  - Gradient Descent is sensitive to the learning rate.
  - Steepest Descent chooses the best step size automatically. And it converges faster and more reliably.
  - Both trajectory and convergence plots confirm this result.

2. Problem2 and solution

Problem 2

Problem 2

Consider the quadratic programming

$$f(x) = \frac{1}{2} x^\top Q x - b^\top x$$

$$\nabla f(x) = Qx - b$$

$$Q = Q^\top > 0, \; b \in \mathbb{R}^n$$

Use "randn" in numpy to generate a 1000 x 1000 matrix A. Then obtain Q by

$$Q = AA^\top + \rho I$$

Here, rho is any positive number such that Q is positive definite.
Use "randn" in numpy to generate 1000 x 1 vector b

Problem 2

1. Show that the function is convex.
2. Obtain the optimal solution and optimal cost using CVX
3. Use Python to design the gradient descent algorithm and find the optimal solution and optimal value. Also, obtain the convergence plot
4. Use Python to design the steepest (optimal step size) gradient descent and find the optimal solution and optimal value. Also, obtain the convergence plot
5. Use Python to design the Neterov-2 algorithm and find the optimal solution and optimal value. Also, obtain the convergence plot
6. Let x^* be the optimal solution obtained by 2. Let $x^*_{Gradient}$ be the optimal solution obtained by 3. Let $x^*_{SteepestGradient}$ be the optimal solution obtained by 4. Let $x^*_{Nestrov-2}$ be the optimal solution obtained by the Nesterov-2 in 5. Plot the convergence plot of

$$\|x^* - x^*_{Gradient}\|, \quad \|x^* - x^*_{SteepestGradient}\| \quad \|x^* - x_{Nesterov-2}\|$$

7. Discuss the convergence speed in the convergence plots in 6

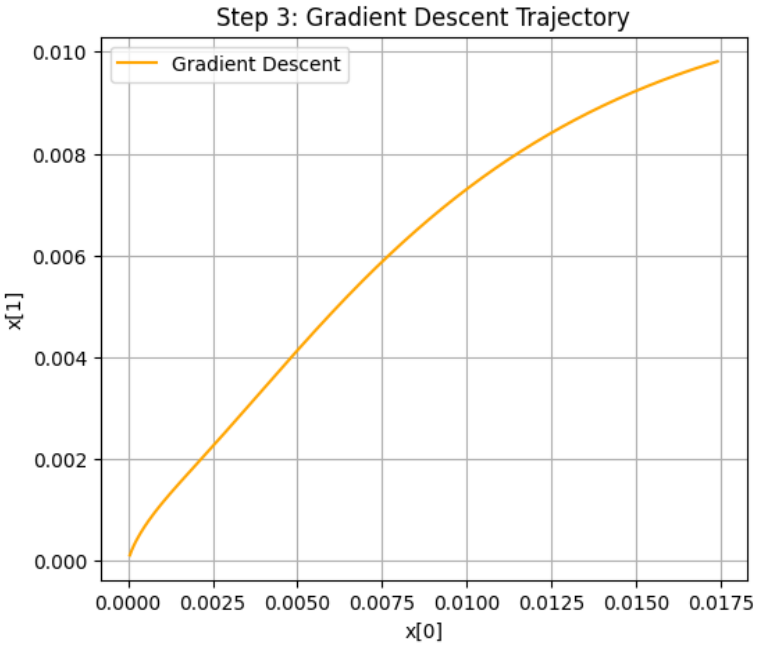
Problem 2

- Step 1: Convexity Check
  - The matrix Q is symmetric ($Q^T = Q$).
  - All eigenvalues of Q are positive.
  - Therefore, Q is positive definite and f(x) is strictly convex.
  - The problem has a unique global minimum.

- Step 2: Optimal Solution Using CVX
  - Optimal cost: $f(x^*) \approx -12.607$
  - First 5 values of x^* : [0.0303, -0.0002, -0.0020, 0.0165, 0.1205]
  - CVX result is accurate and reliable.

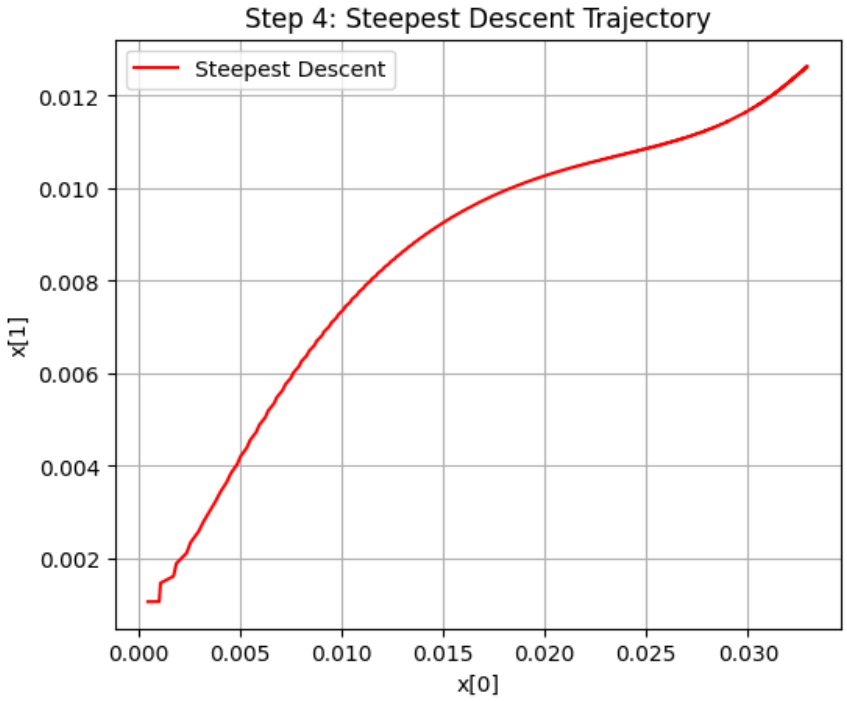
Problem 2

- Step 3: Gradient Descent with fixed step size
  - Initialized x_0 as zero vector, step size α = 1e-4 with Update rule: $x \leftarrow x - \alpha(Qx - b)$
  - After 500 iterations:
    - » Final cost $f(x) \approx -4.511$
    - » $x^* \approx [0.0174, 0.0098, 0.0106, -0.0006, -0.0067]$
  - Slow convergence due to small fixed step size.



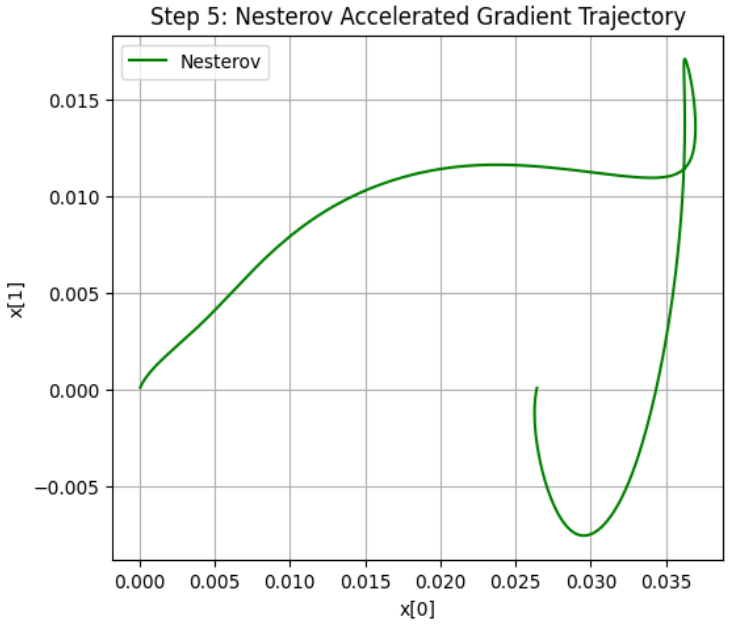
Problem 2

- Step 4: Steepest Descent (Optimal Step Size)
  - At each iteration, step size α is computed by $\alpha = (g^t g) / (g^t Q g)$
  - Final cost: $f(x) \approx -8.875$
  - $x^* \approx [0.0329, 0.0126, 0.0138, -0.0032, 0.0237]$
  - Faster convergence than standard Gradient Descent.



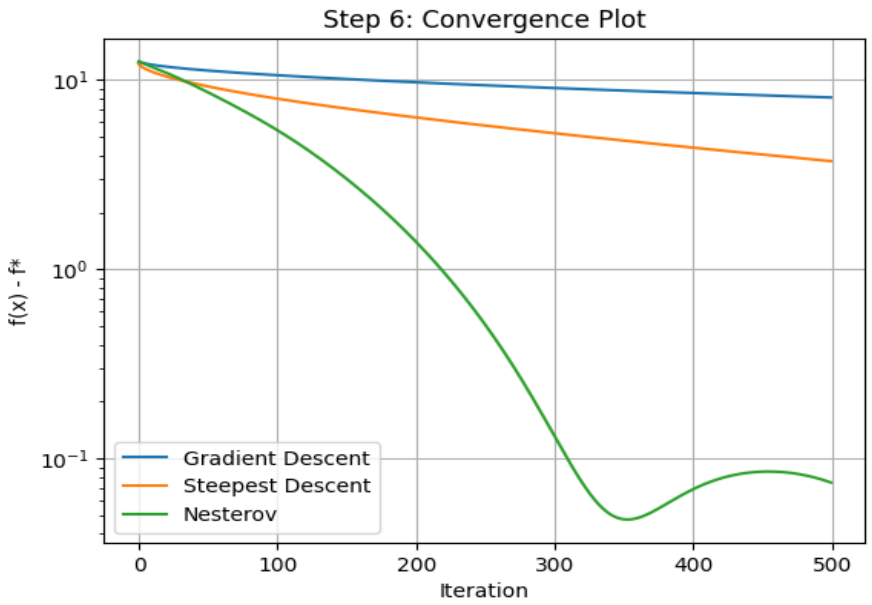
Problem 2

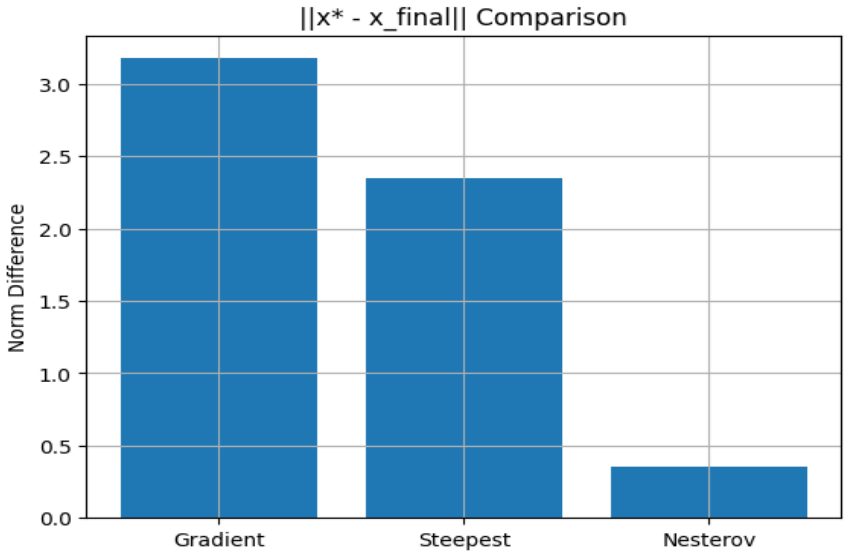
- Step 5: Nesterov Accelerated Gradient
  - Uses momentum
    - » $x_{k+1} = y_k - \alpha \nabla f(y_k)$
    - » $y_{k+1} = x_{k+1} + ((t_k - 1)/t_{k+1})(x_{k+1} - x_k)$
  - Final cost: $f(x) \approx -12.533$
  - $x^* \approx [0.0264, 0.0001, -0.0063, 0.0229, 0.1314]$
  - Fastest convergence among all methods.



Problem 2

- Step 6: Convergence Comparison
  - Gradient Descent: largest error
  - Steepest Descent: smaller error
  - Nesterov: smallest error
  - Nesterov method is closest to the true solution.





Problem 2

- Step 7: Convergence Speed Summary
  - Gradient Descent is slow due to fixed step size.
  - Steepest Descent is faster with optimal step size.
  - Nesterov is the fastest due to momentum acceleration.
  - Nesterov shows the best performance in both speed and accuracy.

3. Problem3 and solution

Problem 3

Problem 3

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

1. Find the optimal solution using the gradient descent method with the initial condition that you choose

2. Try the gradient descent with different initial conditions
   - For 1 and 2, you need to use Python
   - Discuss the convergence speeed with different initial conditions
   - Does the convergence depend on the initial condition?

Equation

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

- Step 1: Gradient Descent with Initial Condition
  - I applied gradient descent to the Rosenbrock function
    - » $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$
  - The global minimum is at (1, 1) and initial point: [0.0, 0.0].
  - Converged successfully after several iterations.
  - A small learning rate ensures convergence.
  - However, the function's curved valley slows down the process.

Equation

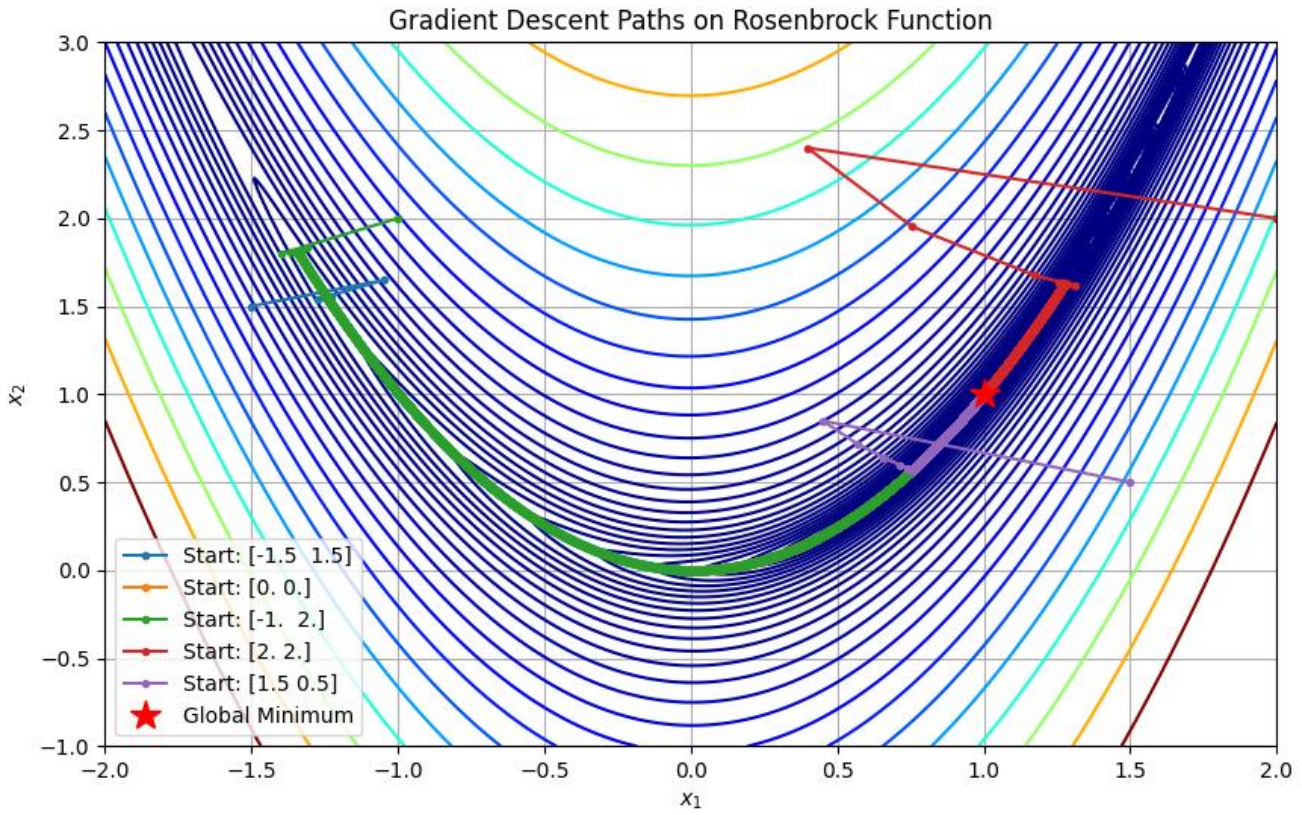
$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

- Step 2: Gradient Descent with Different Initial Conditions
  - Tested five initial points
    - » [-1.5, 1.5], [0.0, 0.0], [-1.0, 2.0], [2.0, 2.0], [1.5, 0.5]
  - Each starting point showed different behavior.
  - [0.0, 0.0] and [1.5, 0.5] converged relatively fast.
  - [-1.5, 1.5] and [-1.0, 2.0] took more time.
  - [2.0, 2.0] converged slowly due to flat region.

- Conclusion
  - Convergence is highly sensitive to the initial condition.
  - The Rosenbrock function has a narrow, curved valley.
  - Visualization confirms the importance of proper initialization.

Equation

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$



4. Problem4 and solution

Problem 4

Problem 4

$$\min_{x \in \mathbb{R}^n} c^\top x$$

$$\text{subject to } Ax \leq b$$

$$x_i \geq 0, \ i = 1, \ldots, n$$

note that

$$c \in \mathbb{R}^n, \ A \in \mathbb{R}^{p \times n} \ (p \leq n), \ b \in \mathbb{R}^p$$

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Problem 4

Problem 4

- Use MATLAB or Python to generate random c, A, b data (e.g. for MATLAB)
  - c = rand(10,1), A = randn(8,10)*3+0.2, b = -5 + (5+5)*rand(8,1)
  - Use linprog in MATLAB or Python to find the optimal cost and optimal solution with the random data given above
- Use MATLAB or Python to generate random c, A, b data (e.g. for MATLAB)
  - c = rand(100,1), A = randn(55,100)*5-1.2, b = -1 + (1+1)*rand(55,1)
  - Use linprog in MATLAB or Python to find the optimal cost and optimal solution with the random data given above
- Discuss the location of the optimal solution in terms of the constrains

$$Ax \leq b$$
$$x_i \geq 0, \ i = 1, \ldots, n$$

17

Problem 4

- Step 1: Small-Scale Linear Programming (n = 10, p = 8)
  - I randomly generated data.
    - » c $\in \mathbb{R}^{10}$ (uniform random values between 0 and 1)
    - » A $\in \mathbb{R}^{8x10}$ (normally distributed, scaled by 3 and shifted by 0.2)
    - » b $\in \mathbb{R}^{8}$ (uniformly random in [-5, 5])
  - The optimal solution using linprog.
    - » x* = [0.1388, 0, 0, ..., 0]
    - » Optimal cost $c^{T}x^{*}$ = 0.0761
  - Most variables are zero and only one is non-zero.
  - Solution lies on the vertex where multiple constraints are active.

Problem 4

- Step 2: Large-Scale Linear Programming (n = 100, p = 55)
  - I generated data.
    - » $c \in \mathbb{R}^{100}$
    - » $A \in \mathbb{R}^{55x100}$
    - » $b \in \mathbb{R}^{55}$
    - » $A \sim \mathcal{N}(0, 1)$, scaled by 5 and shifted by -1.2
    - » b ~ Uniform(-1, 1)
  - Most components of x* are zero
  - Optimal cost $c^T x^* \approx 0.0433$
  - Solution is sparse and lies at a vertex of the feasible region.

```
Optimal solution x*: [0.         0.         0.01809753 0.         0.04834609 0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.04001913 0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.06937607 0.         0.         0.02770573
 0.         0.04067056 0.         0.06837704 0.         0.0120592
 0.         0.         0.         0.05245286 0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.00943971 0.         0.         0.         0.
 0.         0.         0.         0.         0.03275683 0.
 0.         0.         0.         0.         0.07218219 0.05351896
 0.         0.         0.         0.         0.         0.
 0.         0.01756716 0.07183468 0.         0.         0.
 0.         0.         0.         0.        ]
Optimal cost c^T x*: 0.043303737158779644
```

Problem 4

- Step 3: Location of the Optimal Solution
  - In linear programming, the optimal solution lies on the boundary of the feasible region.
  - Due to non-negativity constraints ($x_i \geq 0$), many variables become zero.
  - The solution is determined by the intersection of active constraints.

5. Problem5 and solution

Problem 5

Problem 5

$$\min_{x \in \mathbb{R}^n} f(x)$$

$$\text{where } f(x) = \|Ax - b\|_Q^2 = (Ax - b)^\top Q(Ax - b)$$

$A \in \mathbb{R}$

$b \in \mathbb{R}^n$

$Q = Q$

Problem 5

1. Show that the problem, is convex
2. Find the optimal solution to this problem by your hand
3. Let n=100, m=50, use Python to generate the Gaussian random data of A, b, Q. Then find the optimal solution via the gradient descent method and the solution of 2. Also use CVX to find the optimal solution
4. Let n=50, m=100 , use Python to generate the Gaussian random data of A, b, Q. Then find the optimal solution via the gradient descent method and the solution of 2. Also use CVX to find the optimal solution.
5. Discuss the solutions of 3 and 4. Provide your discussion in terms of the matrix inverse and rank condition

Problem 5

Problem 5

1. Show that the problem, is convex

2. Find the optimal solution to this problem by your hand

3. Let n=100, m=50, use Python to generate the Gaussian random data of A, b, Q. Then find the optimal solution via the gradient descent method and the solution of 2. Also use CVX to find the optimal solution

4. Let n=50, m=100 , use Python to generate the Gaussian random data of A, b, Q. Then find the optimal solution via the gradient descent method and the solution of 2. Also use CVX to find the optimal solution.

5. Discuss the solutions of 3 and 4. Provide your discussion in terms of the matrix inverse and rank condition

Problem 5

- Step 1: Convexity of the Problem
  - Q is symmetric and positive definite, so the Hessian $H = A^T Q A$ is also positive semidefinite.
  - The function $f(x) = (Ax - b)^T Q (Ax - b)$ is convex.
  - A has full column rank, then $H \succ 0 \rightarrow$ strictly convex.

- Step 2: Closed-form Solution
  - The gradient is $\nabla f(x) = 2 A^T Q (Ax - b)$.
  - Setting this to 0 gives $A^T Q A x = A^T Q b$.
  - The solution is $x^* = (A^T Q A)^{(-1)} A^T Q b$.

Problem 5

- Step 3: Case 1 – n=100, m=50
  - Convexity check: False
  - Analytical x*: [-0.7772, 0.5154, 0.1842, -0.4622, 0.4988]
  - GD result: NaN
  - CVX x*: [0.1296, -0.1178, 0.0004, 0.0193, -0.1651]
  - Difference ||x(ana) – x(cvx)|| ≈ 8.37
    - » Ill-conditioned matrix causes instability.

- Step 4: Case 2 – n=50, m=100
  - Convexity check: True
  - Analytical x*: [-0.1099, 0.2264, -0.0149, -0.0188, 0.0255]
  - GD result: NaN
  - CVX x*: [-0.1099, 0.2264, -0.0149, -0.0188, 0.0255]
  - Difference ||x(ana) – x(cvx)|| ≈ 8e-14
    - » Solutions match. Therefore, matrix is well-conditioned.

Problem 5

- Step 5: Discussion
  - Case 1 (n > m): $A^T Q A$ may not be full-rank
    - » Unstable inverse.
  - Case 2 (m > n): $A^T Q A$ is likely invertible
    - » Stable solution.
  - CVX provides robust results even when the matrix is ill-conditioned.