Omkar Maurya

1 Gradient descent

1.1 Gradient descent is one of the oldest algorithms (dating back to Cauchy), yet simple and popular in many scientific communities even today...

Algorithm 1 Gradient Descent Procedure with Constant Step Length

article amsmath **Inputs:**

- Starting point x_0
- Tolerance level τ
- Step length η

Outputs:

- The final iterate x_k

Procedure:

1. **Initialization:** Set $k = 0$.
2. **Iteration Condition:** While the 2-norm of the gradient $\|\nabla f(x_k)\|_2$ is greater than the tolerance level τ, continue with the iterations.
3. **Update Rule:** Update the current iterate using the gradient descent update rule: $x_{k+1} = x_k - \eta \nabla f(x_k)$.
4. **Iteration Count:** Increment the iteration counter: $k = k + 1$.
5. **Output:** Once the iteration condition is no longer satisfied ($\|\nabla f(x_k)\|_2 \leq \tau$), output the final iterate x_k.

Procedure:

1. **Initialization:** Set $k = 0$.
2. **Iteration Condition:** While the 2-norm of the gradient $\|\nabla f(x_k)\|_2$ is greater than the tolerance level τ, continue with the iterations.
3. **Update Rule:** Update the current iterate using the gradient descent update rule: $x_{k+1} = x_k - \eta \nabla f(x_k)$.
4. **Iteration Count:** Increment the iteration counter: $k = k + 1$.
5. **Output:** Once the iteration condition is no longer satisfied ($\|\nabla f(x_k)\|_2 \leq \tau$), output the final iterate x_k.

1.2 With the starting point x = (1.5, 1.5) and = 0.001, we aim to analyze the behavior of the algorithm 2 for different tolerance values. We set = 10p where p = 1, 2, . . . , 13. For each , record the final minimizer, objective function value at termination, and the number of iterations required for convergence in a tabular form. Generate a plot

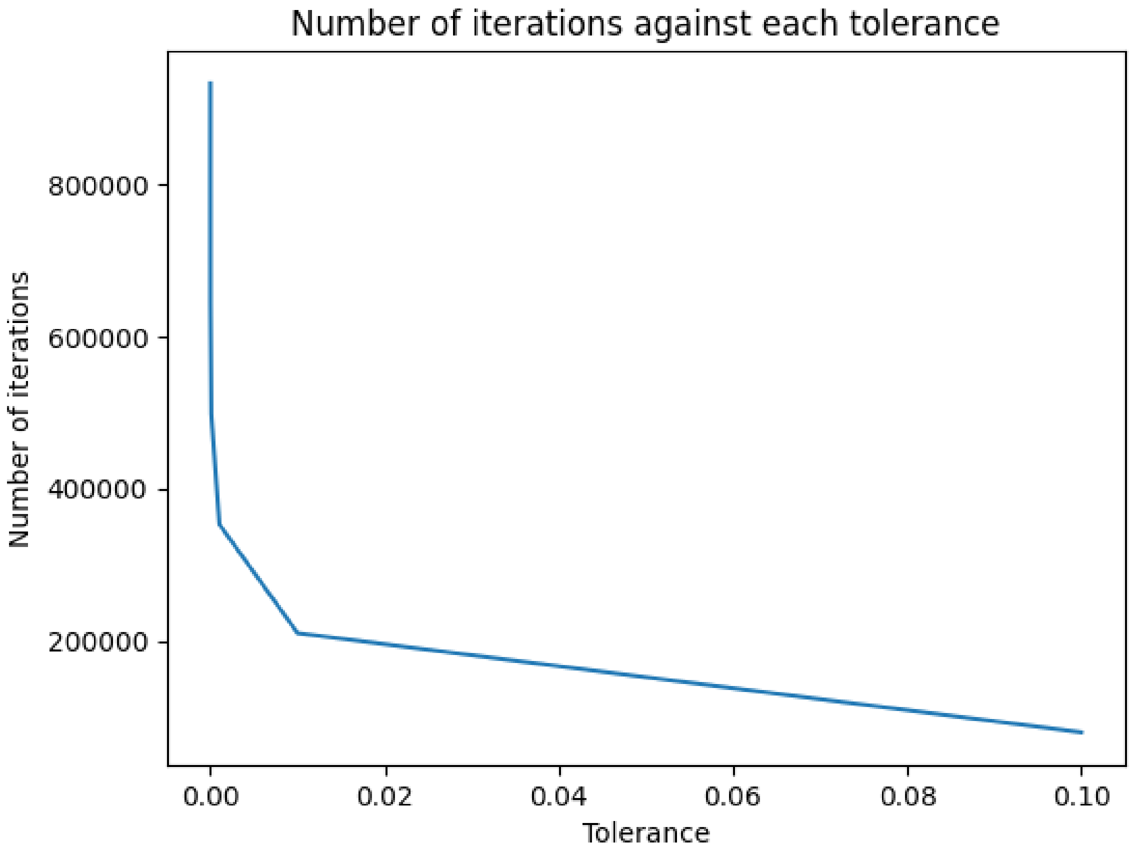

Figure 6: contour plot.

1.3 Comment on the observations. Comment about the minimizers and objective function values obtained for different choices of tolerance values.

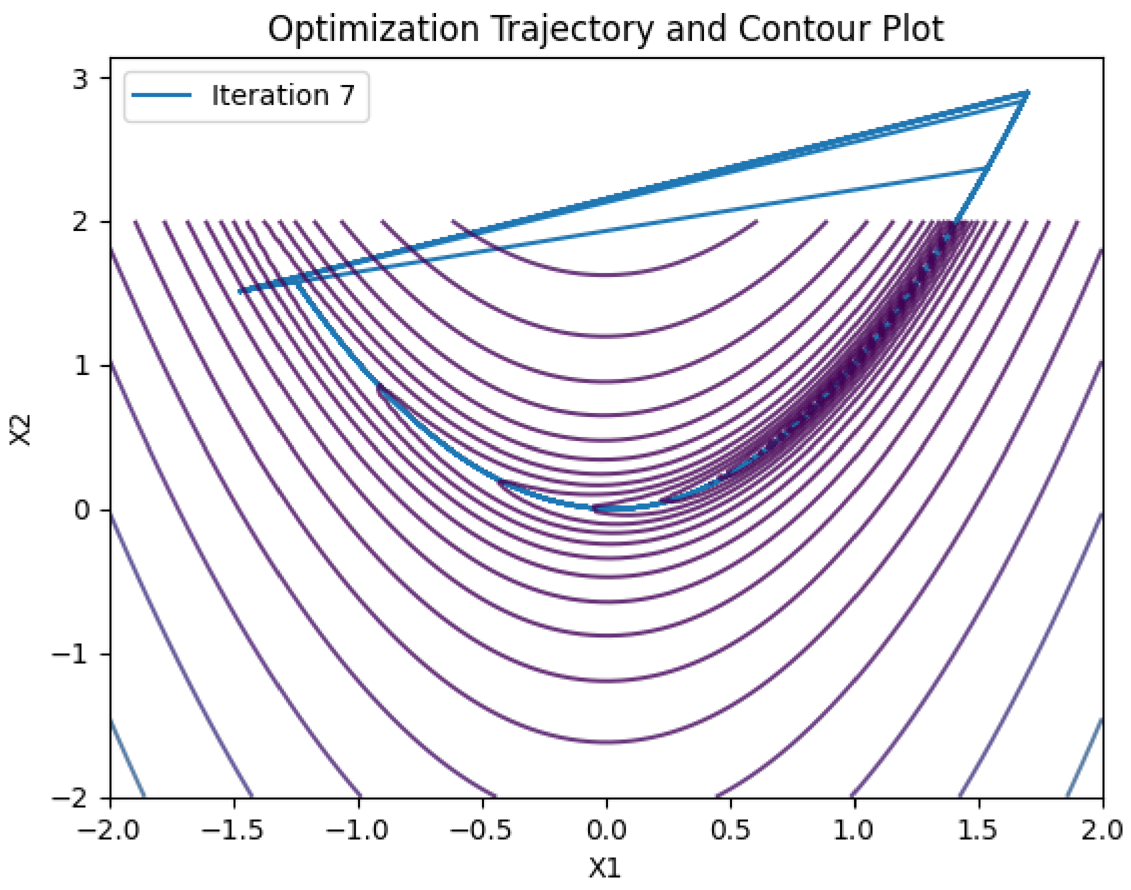

Figure 7: contour plot.

Observations:

1. **Convergence Speed:** As the tolerance decreases (i.e., as you move from left to right on the plot), the number of iterations required for convergence generally increases. This is because a smaller tolerance requires the algorithm to converge to a more precise solution, necessitating more iterations.

2. **Effectiveness of the Algorithm:** The gradient descent algorithm appears effective for a range of tolerance values, as it converges to a solution for each tested case. However, the rate of convergence varies, and it might take more iterations to reach convergence for smaller tolerance values.

3. **Minimizers and Objective Function Values:** The minimizers (values of x_1 and x_2) and objective function values obtained for different tolerance values are expected to be similar but more precise as the tolerance decreases. Lower tolerance values result in a more accurate approximation of the minimum of the objective function.

4. **Trade-off between Precision and Computational Cost:** Choosing a smaller

tolerance provides a more accurate solution but comes at the cost of increased computational effort (more iterations). There's a trade-off between the precision of the solution and the computational resources required.

5. **Behavior for Extremely Small Tolerance:** As you approach extremely small tolerance values, the algorithm might exhibit diminishing returns in terms of improvement in the objective function value. The computational cost may significantly increase without a proportional gain in precision.

1.4. What may be the shortcomings of this algorithm and suggests a possible solution to deal with it ?

1. **Fixed Step Size:** Using a fixed step size η may lead to slow convergence or divergence. If the step size is too small, convergence may be slow, and if it's too large, the algorithm might oscillate or even diverge.

   Solution: Implement techniques such as line search or adaptive step sizes. Line search dynamically adjusts the step size in each iteration based on a search along the gradient direction.

2. **Sensitivity to Initial Guess:** The algorithm's convergence may depend on the choice of the initial point. In some cases, it might converge to different local minima or saddle points based on the initial values.

   Solution: Use multiple initial points or employ optimization techniques that are less sensitive to the choice of the starting point.

3. **High Computational Cost:** The algorithm may be computationally expensive, especially for high-dimensional problems or large datasets, as it requires computing gradients in each iteration.

   Solution: Implement stochastic gradient descent (SGD) or mini-batch gradient descent to reduce the computational cost.

4. **Convergence to a Local Minimum:** Gradient descent is prone to getting stuck in local minima, especially for non-convex functions.

   Solution: Use optimization techniques that explore multiple paths or leverage advanced optimization algorithms like stochastic gradient descent with momentum, Nesterov accelerated gradient, or second-order methods like L-BFGS.

5. **Ill-Conditioned Problems:** For ill-conditioned problems, where the Hessian matrix varies widely, the algorithm may perform poorly.

   Solution: Use preconditioning techniques or more advanced optimization algorithms that can handle ill-conditioned problems.

6. **Noisy Objective Functions:** In the presence of noise, the algorithm might get misled by fluctuations in the gradient.

Solution: Incorporate techniques like gradient smoothing or implement optimization methods robust to noise, such as stochastic gradient descent.

Addressing these shortcomings may involve incorporating adaptive strategies for step size, leveraging advanced optimization techniques, considering multiple initial points, and optimizing for specific problem characteristics. The choice of the solution depends on the nature of the optimization problem at hand.

2 Newton's method

2.1 Newton's method, inspired by Isaac Newton's work, is a classic optimization approach. It iteratively updates the current solution by calculating a direction to descend and an optimal step size.

The general form of the function is:

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

Substituting $b = 80$ and $a = 2$, the specific function becomes:

$$f(x_1, x_2) = (3 - x_1)^2 + 80(x_2 - x_1^2)^2$$

Applying the first derivative condition, the system of equations is:

$$\frac{\partial}{\partial x_1} f(x_1, x_2) = 2x_1 - 6 + 80x_1^3 - 80x_1x_2 = 0$$
$$\frac{\partial}{\partial x_2} f(x_1, x_2) = 40x_2 - 40x_1^2 = 0$$

Solving the system, we find $x_1 = 3$ and $x_2 = 9$.
The second partial derivative with respect to x_1 twice is given by:

$$\frac{\partial^2}{\partial x_1^2} f(x_1, x_2) = 160(3x_1^2 - x_2 + 1)$$

$$\begin{bmatrix} 240x_1^2 - 80x_2 + 2 & -80x_1 \\ -80x_1 & 40 \end{bmatrix}$$

At the point $(3, 9)$, the Hessian matrix becomes:

$$\nabla^2 f(3, 9) = \begin{bmatrix} 1442 & -240 \\ -240 & 40 \end{bmatrix}$$

The eigenvalues of $\nabla^2 f(3, 9)$ are both positive, indicating that the matrix is positive definite. Therefore, the function $f(x_1, x_2)$ will have a minimum value at the point $(3, 9)$. Here's the LaTeX code: "'latex article amsmath The Hessian matrix $\nabla^2 f(x_1, x_2)$ is given by:

$$\nabla^2 f(x_1, x_2) = \begin{bmatrix} 240x_1^2 - 80x_2 + 2 & -80x_1 \\ -80x_1 & 40 \end{bmatrix}$$

At the point $(3,9)$, the Hessian matrix becomes:

$$\nabla^2 f(3,9) = \begin{bmatrix} 1442 & -240 \\ -240 & 40 \end{bmatrix}$$

The eigenvalues of $\nabla^2 f(3,9)$ are both positive, indicating that the matrix is positive definite. Therefore, the function $f(x_1, x_2)$ will have a minimum value at the point $x_1 = 3, x_2 = 9$.

1. After solving the first derivative $\nabla f(x_1, x_2) = 0$, we get $x_1 = 3,\ x_2 = 9$, i.e., the minimizer is unique.

2. Since the minimizer is unique and the second derivative is positive, the minimizer is a global minimum.

3. The figure below shows that the function $f(x_1, x_2)$ is convex, as it exhibits upward convexity.

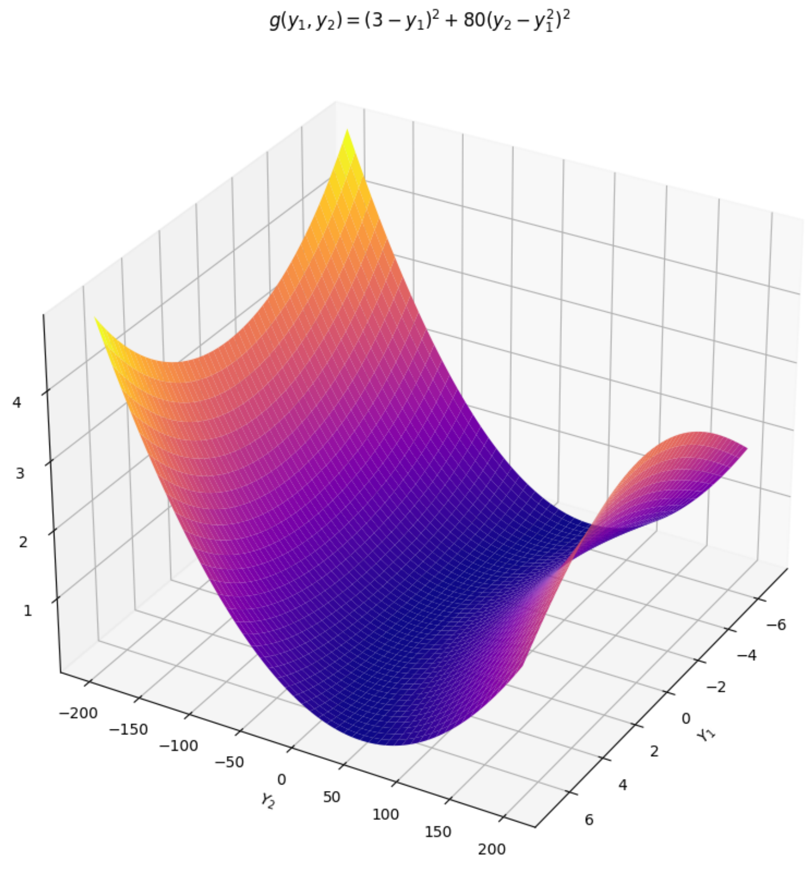

Figure 8: Convex

2.2 3. With the starting point x = (1.5, 1.5), we aim to analyze the behavior of the algorithm 3 for different tolerance values. We set $= 10(p)wherep = 1, 2, ..., 20.Foreach$

2.3 Observations

1. **Gradient Descent Optimization:** The code is implementing a gradient descent optimization algorithm to minimize the function $g(y_1, y_2)$.

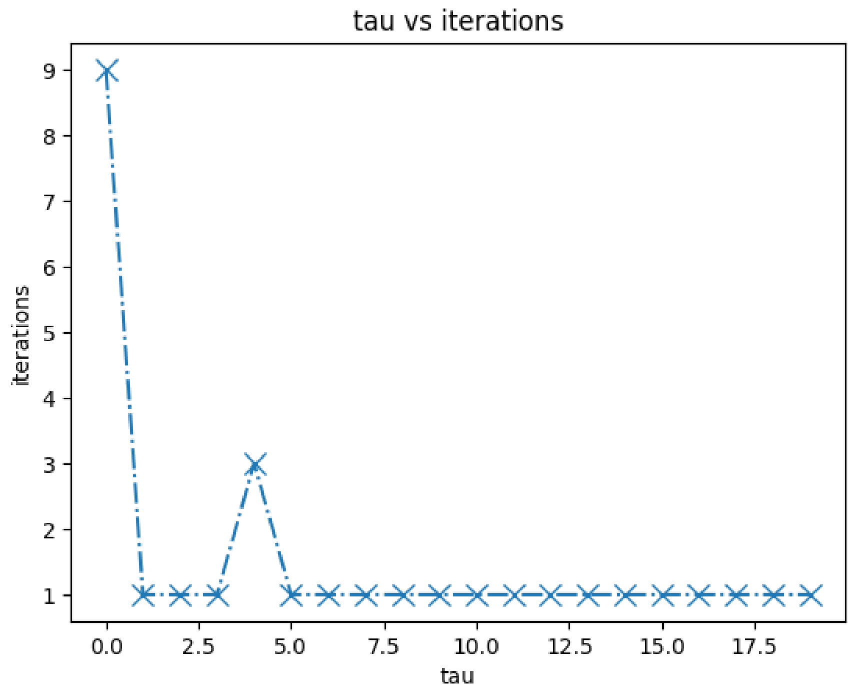

Figure 9: Convex

2. **Objective Function:** The objective function $g(y_1, y_2)$ is a non-linear function with a global minimum. It is the Rosenbrock function, commonly used to test optimization algorithms.

3. **Initial Point:** The optimization starts from the initial point $[-1.5, 1.5]$.

4. **Precision and Step Size (τ):** The precision (τ) is varied from 10^{-1} to 10^{-20} in the optimization loop. A smaller τ implies a more accurate solution but may increase the number of iterations.

5. **Convergence Criterion:** The optimization continues until the norm of the gradient becomes smaller than the current τ. This ensures that the algorithm stops when it reaches a sufficiently small gradient.

6. **Iterations:** The number of iterations required for convergence is recorded for each precision.

7. **Printed Output:** The code prints information at each iteration, including the precision, τ, final values of y_1 and y_2, the number of iterations, and the value of $g(y_1, y_2)$ at the optimal point.

8. **Plotting:** A plot is generated to show the relationship between the precision (τ) and the number of iterations. This helps visualize how the choice of precision influences the convergence behavior.