Newton’s Method in Optimization

Newton’s Method (or the Newton–Raphson method) is a **second-order iterative optimization technique** used to find **stationary points** (minima, maxima, or saddle points) of a real-valued differentiable function $f(x)$.

It extends the 1D Newton–Raphson root-finding method to optimization problems by finding where the **gradient** (first derivative) becomes zero.

Objective:

We want to find x^* such that:

$$\nabla f(x^*) = 0$$

where

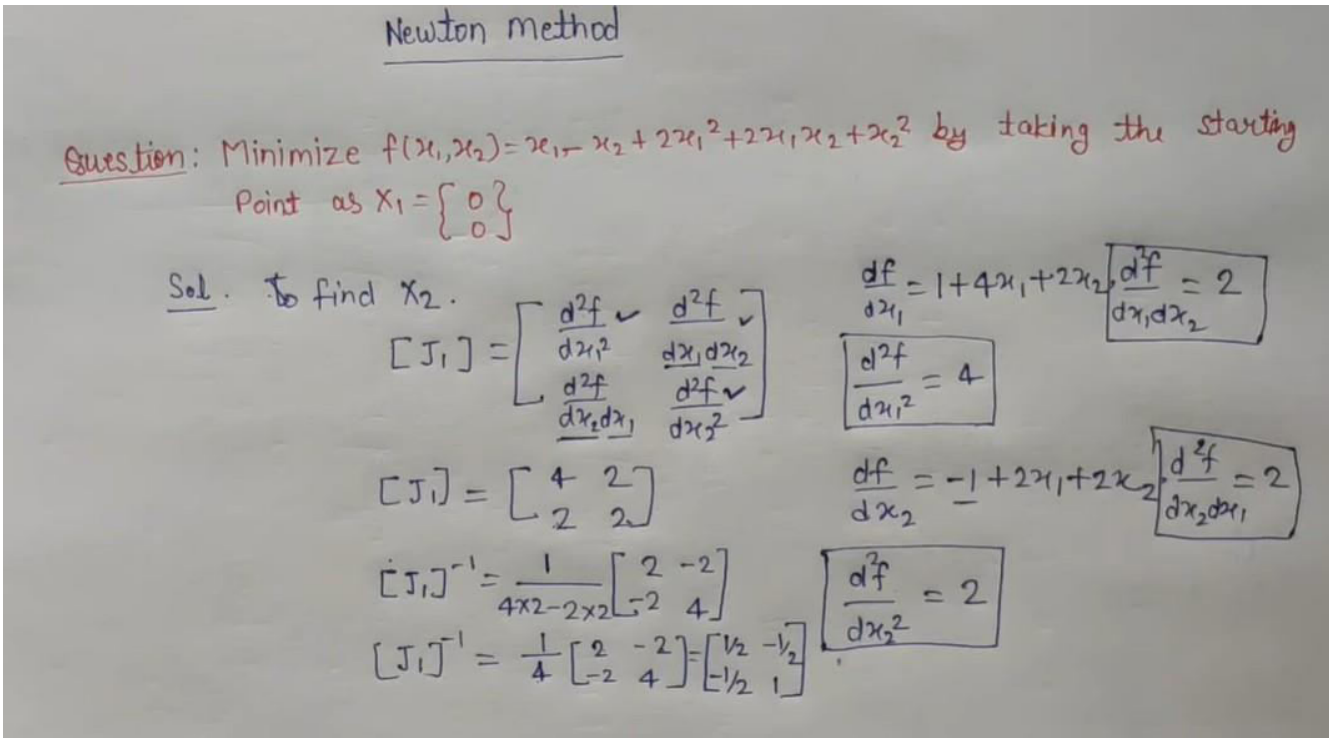
- $\nabla f(x)$ = gradient vector of $f(x)$,
- $\nabla^2 f(x)$ = Hessian matrix (matrix of second derivatives).

Algorithm

1. **Initialize:** Choose a starting point x_0.
2. **Compute gradient:** $g_k = \nabla f(x_k)$.
3. **Compute Hessian:** $H_k = \nabla^2 f(x_k)$.
4. **Compute search direction:** $d_k = -H_k^{-1} g_k$.
5. **Update:** $x_{k+1} = x_k + d_k$.
6. **Check convergence:** If $||g_{k+1}|| < \epsilon$, stop; else repeat.

Disadvantages of Newton’s Method

Disadvantage	**Explanation**
1. Requires Hessian computation	The Hessian ((n \times n) matrix) must be computed and inverted — expensive for large (n).
2. May not converge	If the Hessian is not positive definite (saddle point or maximum), the step can move away from minimum.
3. Sensitive to initial guess	Poor starting point can lead to divergence or convergence to the wrong stationary point.
4. High computational cost	Computing and inverting the Hessian costs (O(n^3)).
5. Not suitable for non-smooth functions	Requires continuous second derivatives.
6. Step may overshoot	If the step size is too large, the quadratic approximation fails — often a line search or damping factor is added.



Newton method

Question: Minimize $f(x_1,x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$ by taking the starting Point as $X_1 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$

Sol. To find X_2.

$$[J_1] = \begin{bmatrix} \frac{d^2f}{dx_1^2} ✓ & \frac{d^2f}{dx_1 dx_2} ✓ \\ \frac{d^2f}{dx_2 dx_1} & \frac{d^2f}{dx_2^2} ✓ \end{bmatrix}$$

$\frac{df}{dx_1} = 1 + 4x_1 + 2x_2$, $\boxed{\frac{d^2f}{dx_1 dx_2} = 2}$

$\boxed{\frac{d^2f}{dx_1^2} = 4}$

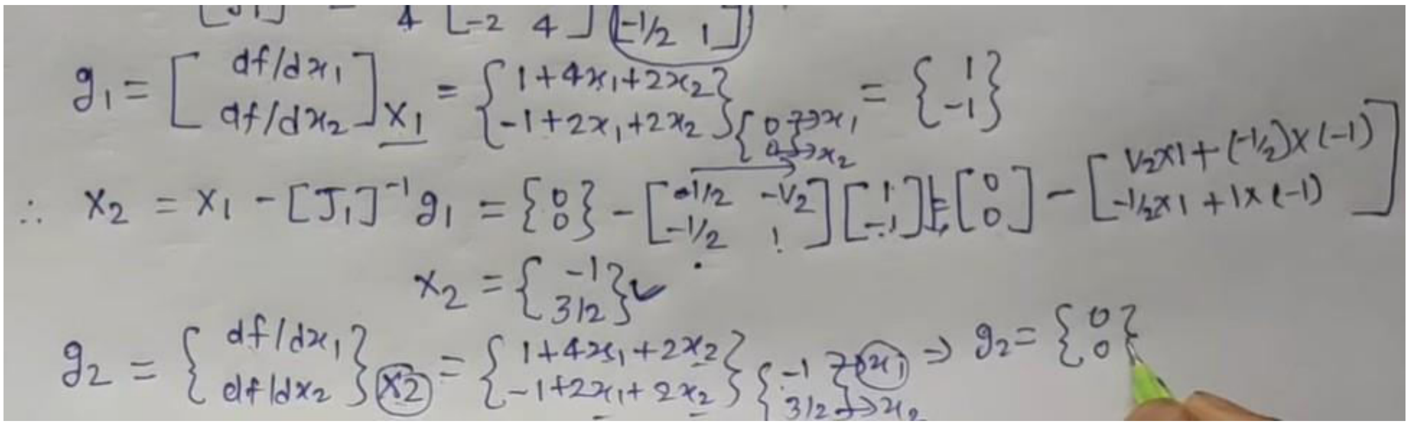
$\frac{df}{dx_2} = -1 + 2x_1 + 2x_2$, $\boxed{\frac{d^2f}{dx_2 dx_1} = 2}$

$\boxed{\frac{d^2f}{dx_2^2} = 2}$

$$[J_1] = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}$$

$$[J_1]^{-1} = \frac{1}{4\times2 - 2\times2}\begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix}$$

$$[J_1]^{-1} = \frac{1}{4}\begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix} = \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1 \end{bmatrix}$$

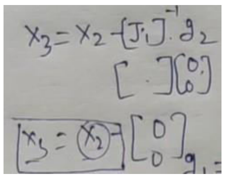


$$g_1 = \begin{bmatrix} df/dx_1 \\ df/dx_2 \end{bmatrix}_{X_1} = \begin{Bmatrix} 1 + 4x_1 + 2x_2 \\ -1 + 2x_1 + 2x_2 \end{Bmatrix}_{\begin{Bmatrix} 0 \to x_1 \\ 0 \to x_2 \end{Bmatrix}} = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix}$$

$$\therefore X_2 = X_1 - [J_1]^{-1} g_1 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} - \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1/2\times1 + (-1/2)\times(-1) \\ -1/2\times1 + 1\times(-1) \end{bmatrix}$$

$$X_2 = \begin{Bmatrix} -1 \\ 3/2 \end{Bmatrix} ✓$$

$$g_2 = \begin{Bmatrix} df/dx_1 \\ df/dx_2 \end{Bmatrix}_{\boxed{X_2}} = \begin{Bmatrix} 1 + 4x_1 + 2x_2 \\ -1 + 2x_1 + 2x_2 \end{Bmatrix}_{\begin{Bmatrix} -1 \to x_1 \\ 3/2 \to x_2 \end{Bmatrix}} \Rightarrow g_2 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$



$$X_3 = X_2 - [J_1]^{-1} \cdot g_2$$

$$[\;\cdot\;]\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

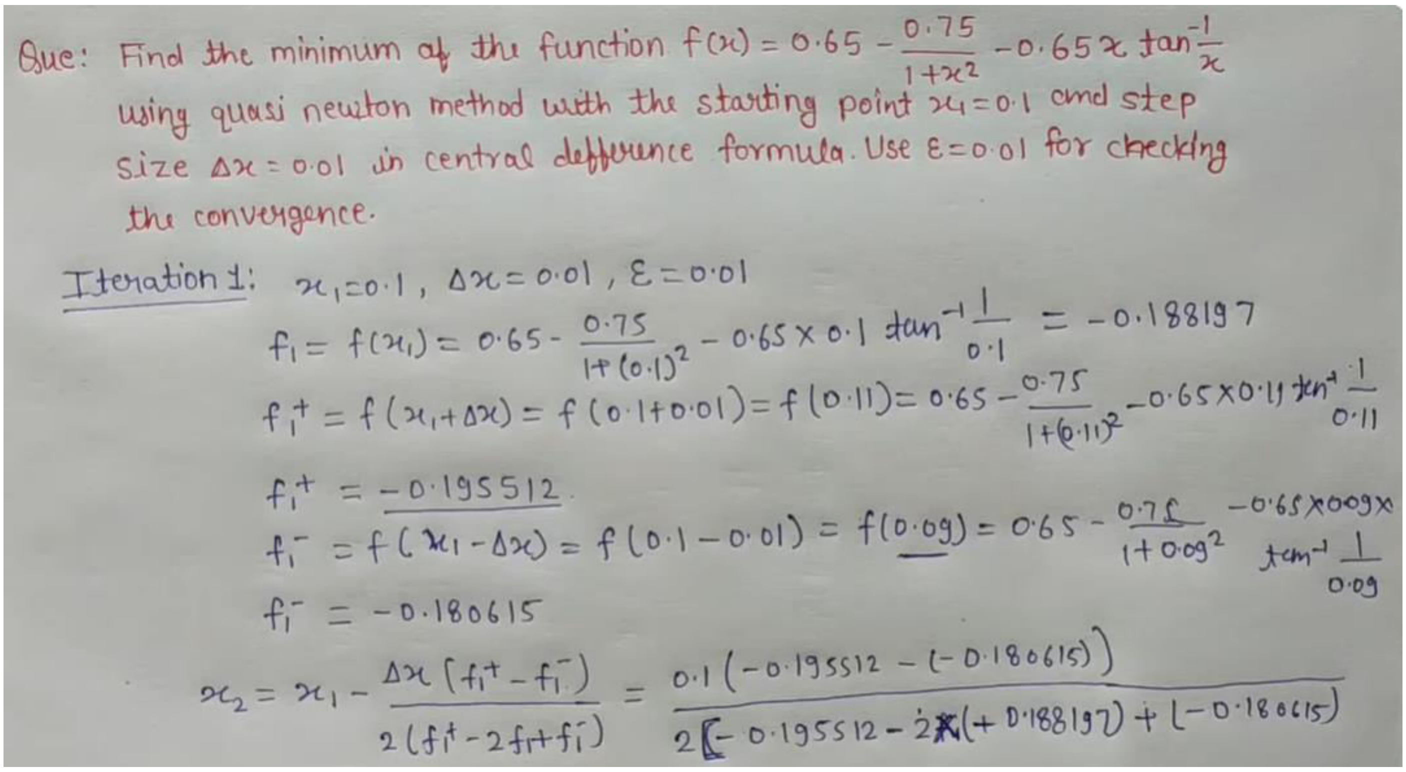
$$\boxed{X_3 = X_2} \begin{bmatrix} 0 \\ 0 \end{bmatrix}_{g_1 =}$$

Newton vs. Gradient Descent

Feature	Gradient Descent	Newton's Method
Uses	Gradient only	Gradient + Hessian
Step Direction	Negative gradient	Inverse Hessian × gradient
Convergence Rate	Linear	Quadratic (faster near optimum)
Cost per Iteration	Low	High (Hessian inversion)
Suitable for	Large-scale, simple problems	Smaller, well-behaved quadratic problems

Quasi Newton Method

- Quasi-Newton methods are optimization algorithms that find minima or maxima of functions by approximating the Hessian matrix of second derivatives.
- Unlike Newton's method, they avoid the computational cost of calculating the true Hessian, instead updating an approximation using gradient and position information from previous steps.
- This makes them more efficient for large-scale problems while still achieving superlinear convergence.



Que: Find the minimum of the function $f(x) = 0.65 - \frac{0.75}{1+x^2} - 0.65x\tan^{-1}\frac{1}{x}$ using quasi newton method with the starting point $x_1 = 0.1$ and step size $\Delta x = 0.01$ in central defference formula. Use $\varepsilon = 0.01$ for checking the convergence.

Iteration 1: $x_1 = 0.1$, $\Delta x = 0.01$, $\varepsilon = 0.01$

$$f_1 = f(x_1) = 0.65 - \frac{0.75}{1+(0.1)^2} - 0.65 \times 0.1 \tan^{-1}\frac{1}{0.1} = -0.188197$$

$$f_1^+ = f(x_1+\Delta x) = f(0.1+0.01) = f(0.11) = 0.65 - \frac{0.75}{1+(0.11)^2} - 0.65 \times 0.11 \tan^{-1}\frac{1}{0.11}$$

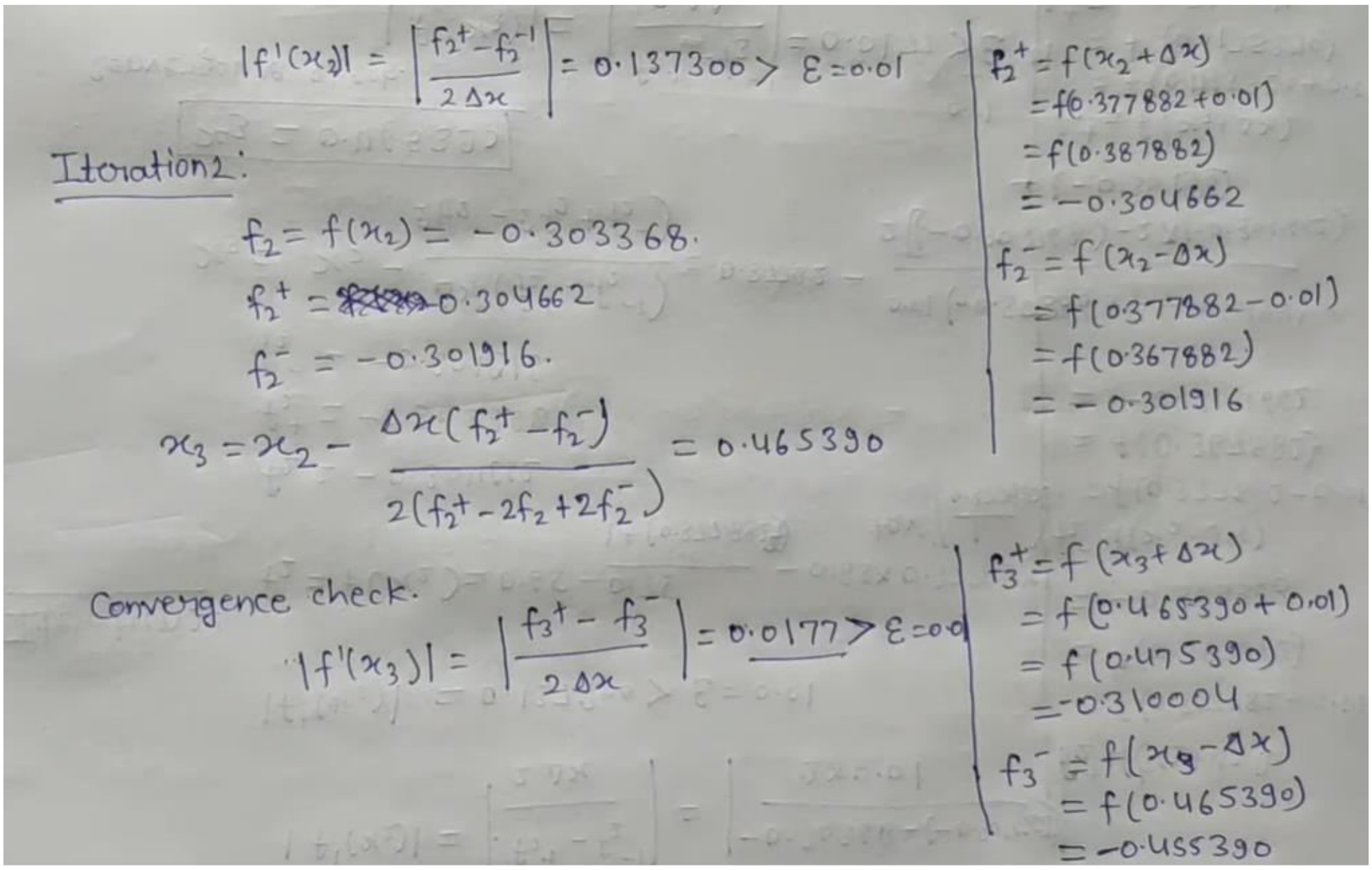
$$f_1^+ = -0.195512$$

$$f_1^- = f(x_1-\Delta x) = f(0.1-0.01) = f(0.09) = 0.65 - \frac{0.75}{1+0.09^2} - 0.65 \times 0.09 \times \tan^{-1}\frac{1}{0.09}$$

$$f_1^- = -0.180615$$

$$x_2 = x_1 - \frac{\Delta x\,(f_1^+ - f_1^-)}{2(f_1^+ - 2f_1 + f_1^-)} = \frac{0.1\left(-0.195512 - (-0.180615)\right)}{2[-0.195512 - 2(+0.188197) + (-0.180615)}$$

So, we have x_2 = 0.377882



$|f'(x_2)| = \left|\frac{f_2^+ - f_2^{-1}}{2\Delta x}\right| = 0.137300 > \varepsilon = 0.01$

$f_2^+ = f(x_2 + \Delta x)$
$= f(0.377882 + 0.01)$
$= f(0.387882)$
$= -0.304662$

$f_2^- = f(x_2 - \Delta x)$
$= f(0.377882 - 0.01)$
$= f(0.367882)$
$= -0.301916$

Iteration2:

$f_2 = f(x_2) = -0.303368.$

$f_2^+ = -0.304662$

$f_2^- = -0.301916.$

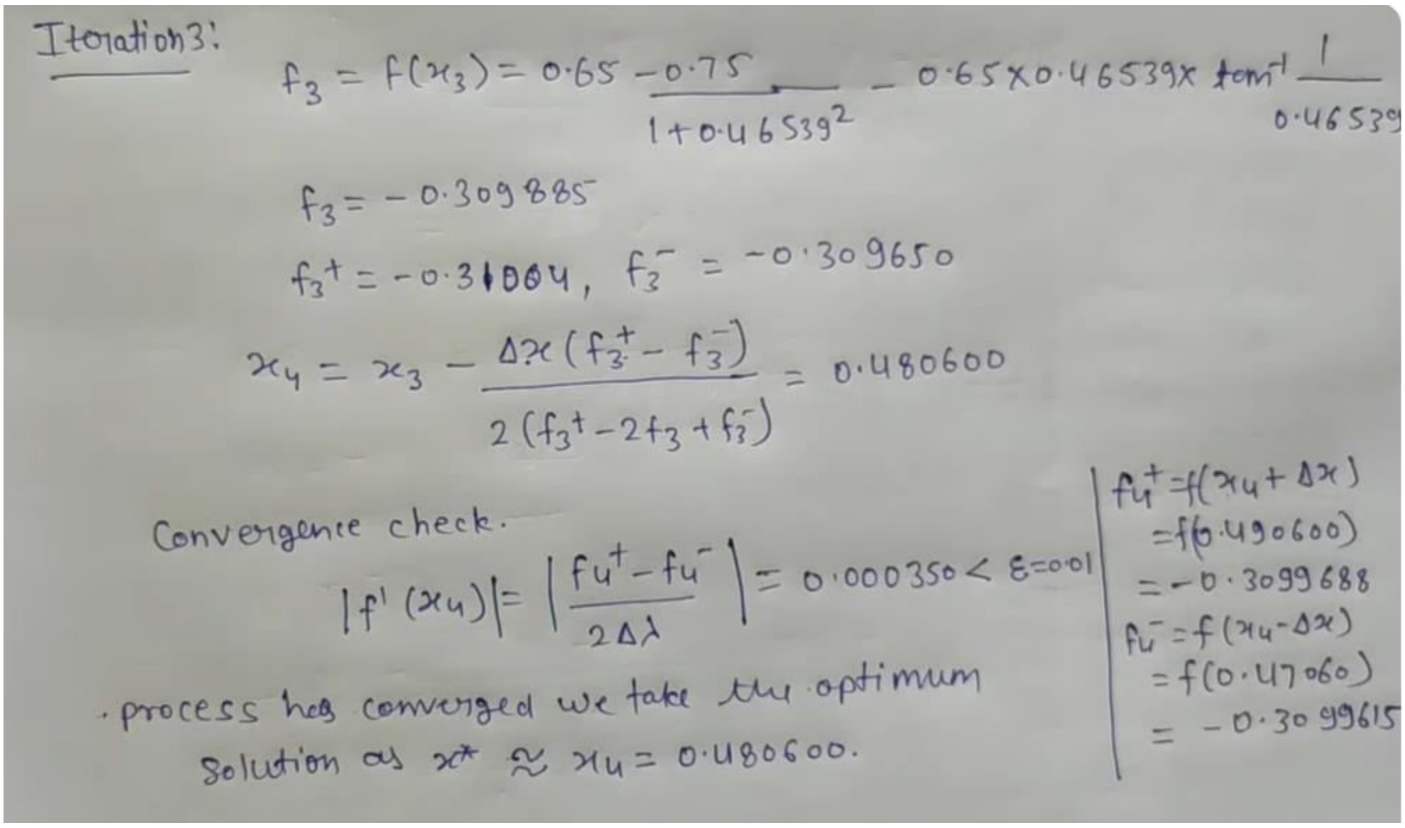
$$x_3 = x_2 - \frac{\Delta x(f_2^+ - f_2^-)}{2(f_2^+ - 2f_2 + 2f_2^-)} = 0.465390$$

Convergence check.

$|f'(x_3)| = \left|\frac{f_3^+ - f_3^-}{2\Delta x}\right| = 0.0177 > \varepsilon = 0.0$

$f_3^+ = f(x_3 + \Delta x)$
$= f(0.465390 + 0.01)$
$= f(0.475390)$
$= -0.310004$

$f_3^- = f(x_3 - \Delta x)$
$= f(0.465390)$
$= -0.455390$



Iteration3:

$f_3 = f(x_3) = 0.65 - \frac{0.75}{1 + 0.46539^2} - 0.65 \times 0.46539 \times \tan^{-1}\frac{1}{0.4653}$

$f_3 = -0.309885$

$f_3^+ = -0.31004, \quad f_3^- = -0.309650$

$$x_4 = x_3 - \frac{\Delta x(f_3^+ - f_3^-)}{2(f_3^+ - 2f_3 + f_3^-)} = 0.480600$$

Convergence check.

$|f'(x_4)| = \left|\frac{f_4^+ - f_4^-}{2\Delta\lambda}\right| = 0.000350 < \varepsilon = 0.01$

$f_4^+ = f(x_4 + \Delta x)$
$= f(0.490600)$
$= -0.3099688$

$f_4^- = f(x_4 - \Delta x)$
$= f(0.47060)$
$= -0.3099615$

· process has converged we take the optimum solution as $x^* \approx x_4 = 0.480600.$

Concept of Regression Tree

A Regression Tree is a type of Decision Tree used when the target (output) variable is continuous (not categorical).

It divides the data into smaller and smaller regions so that the value of the dependent variable (Y) within each region is as homogeneous as possible.

How It Works

1. Start with all the training data.
2. At each node, select the variable and the split point that minimizes the sum of squared errors (SSE) or variance within the resulting groups.
3. Repeat splitting recursively until a stopping condition is met (e.g., minimum number of samples, or no significant improvement).
4. The predicted value for each terminal node (leaf) is the mean of the target variable in that region.

Mathematical Criterion for Split

For a split based on variable X_j at split point s:

$$R_1(j,s) = \{X | X_j \le s\}, \quad R_2(j,s) = \{X | X_j > s\}$$

The cost function is:

$$C(j,s) = \sum_{x_i \in R_1(j,s)} (y_i - \bar{y}_{R_1})^2 + \sum_{x_i \in R_2(j,s)} (y_i - \bar{y}_{R_2})^2$$

We choose (j^*, s^*) that **minimizes** $C(j,s)$.

Dataset

We want to predict the **Sales (Y)** based on the **Advertising Spend (X)** (in ₹ lakh):

Observation	X (Advertising)	Y (Sales)
1	2	4
2	4	6
3	6	8
4	8	10
5	10	11

We will build a **simple regression tree** using one variable X.

Step 1: Find Possible Splits

The possible split points are midpoints between X values:

$$s = 3, 5, 7, 9$$

Step 2: For each split, compute SSE

We'll calculate for each split:

$$SSE = \sum(y_i - \bar{y}_{left})^2 + \sum(y_i - \bar{y}_{right})^2$$

Split 1: s = 3

Left ($X \le 3$): Y = [4] → mean = 4

Right ($X > 3$): Y = [6, 8, 10, 11] → mean = 8.75

SSE =

Left: $(4-4)^2 = 0$

Right: $(6-8.75)^2 + (8-8.75)^2 + (10-8.75)^2 + (11-8.75)^2$

= 7.56 + 0.56 + 1.56 + 5.06 = **14.74**

→ Total SSE = 14.74

Split 2: s = 5

Left ($X \leq 5$): Y = [4, 6] → mean = 5

Right ($X > 5$): Y = [8, 10, 11] → mean = 9.67

SSE =

Left: $(4-5)^2 + (6-5)^2 = 1 + 1 = 2$

Right: $(8-9.67)^2 + (10-9.67)^2 + (11-9.67)^2 = 2.78 + 0.11 + 1.78 = 4.67$

→ Total SSE = 2 + 4.67 = 6.67

Split 3: s = 7

Left ($X \leq 7$): Y = [4, 6, 8] → mean = 6

Right ($X > 7$): Y = [10, 11] → mean = 10.5

SSE =

Left: $(4-6)^2 + (6-6)^2 + (8-6)^2 = 4 + 0 + 4 = 8$

Right: $(10-10.5)^2 + (11-10.5)^2 = 0.25 + 0.25 = 0.5$

→ Total SSE = 8 + 0.5 = 8.5

Split 4: s = 9

Left ($X \leq 9$): Y = [4, 6, 8, 10] → mean = 7

Right ($X > 9$): Y = [11] → mean = 11

SSE =

Left: $(4-7)^2 + (6-7)^2 + (8-7)^2 + (10-7)^2 = 9 + 1 + 1 + 9 = 20$

Right: $(11-11)^2 = 0$

→ Total SSE = 20

Step 3: Choose the Best Split

Split	SSE	Decision
s = 3	14.74	—
s = 5	**6.67 (minimum)**	✅ Best
s = 7	8.5	—
s = 9	20	—

Hence, the **best split** is at **X = 5**.

Step 4: Interpret the Tree

- **If X ≤ 5:** predict Y = 5
- **If X > 5:** predict Y = 9.67

Step 5: Predicted values

X	Y (Actual)	Predicted Y
2	4	5
4	6	5
6	8	9.67
8	10	9.67
10	11	9.67

Final Regression Tree

```
         [X <= 5?]
           /   \
       Yes/     \No
         /       \
Predict Y=5   Predict Y=9.67
```

Mean Squared Error (MSE) in Regression Analysis

In regression analysis, the **Mean Squared Error (MSE)** is a commonly used metric to measure how well a regression model fits the data. It represents the **average of the squares of the errors** — that is, the average squared difference between the **actual (observed)** and the **predicted** values. It is used as a **loss function** in regression algorithms (e.g., linear regression, neural networks). During model training, the goal is often to **minimize the MSE**.

If the actual values are $y_1, y_2, \ldots, y_n$ and the corresponding predicted values from the regression model are $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$, then:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where:

- y_i = actual value
- $\hat{y}_i$ = predicted value
- n = number of data points

Interpretation

- **MSE = 0** means a perfect fit (predictions are exactly equal to actual values).
- A **smaller MSE** value indicates a better fit of the regression model to the data.
- A **larger MSE** value indicates poor predictive accuracy — the model's predictions deviate more from actual data.

Observation (i)	Actual Value y_i	Predicted Value $\hat{y}_i$	Error $y_i - \hat{y}_i$	Squared Error $(y_i - \hat{y}_i)^2$
1	10	12	-2	4
2	8	9	-1	1
3	12	11	1	1
4	14	13	1	1

$$\text{MSE} = \frac{4+1+1+1}{4} = \frac{7}{4} = 1.75$$

Root Mean Square Error (RMSE)	$\sqrt{\frac{1}{n}\sum(y_i - \hat{y}_i)^2}$	Square root of MSE; same units as the response variable.
R² (Coefficient of Determination)	$1 - \frac{\text{SSR}}{\text{SST}}$	Measures proportion of variance explained by the model.

Advantages of MSE

- Easy to compute and widely used.
- Useful in optimization algorithms (like gradient descent in machine learning).
- Penalizes large errors heavily due to squaring, which helps discourage outliers.

Limitations of MSE

- Sensitive to outliers because errors are squared.
- The value is not in the same units as the original data (since errors are squared).
- May not be intuitive for interpretation compared to MAE.

Usage in Model Evaluation

In regression analysis and machine learning:

- MSE helps compare different models — the one with the lowest MSE is usually preferred.
- It is commonly used as the loss function in training models like **Linear Regression**, **Ridge Regression**, and **Neural Networks**.

Numerical on multiple regression analysis

Problem

A company wants to predict **Sales (Y)** based on two independent variables:

- **X_1** = Advertising expenditure (in ₹ lakhs)
- **X_2** = Number of salespeople

The following data is collected:

Observation	X_1	X_2	Y (Sales)
1	1	2	2
2	2	1	3
3	3	4	6
4	4	3	7
5	5	5	11

We need to find the regression equation:

$$Y = b_0 + b_1X_1 + b_2X_2$$

STEP 1: Compute the required means

$$\bar{X}_1 = \frac{1+2+3+4+5}{5} = 3$$

$$\bar{X}_2 = \frac{2+1+4+3+5}{5} = 3$$

$$\bar{Y} = \frac{2+3+6+7+11}{5} = 5.8$$

STEP 2: Compute the deviations

Obs	X_1	X_2	Y	$(X_1-\bar{X}_1)$	$(X_2-\bar{X}_2)$	$(Y-\bar{Y})$
1	1	2	2	-2	-1	-3.8
2	2	1	3	-1	-2	-2.8
3	3	4	6	0	1	0.2
4	4	3	7	1	0	1.2
5	5	5	11	2	2	5.2

STEP 3: Compute the necessary sums

Quantity	Formula	Calculation	Value
S_{x1x1}	$\Sigma(X_1-\bar{X}_1)^2$	$(-2)^2 + (-1)^2 + 0^2 + 1^2 + 2^2$	10
S_{x2x2}	$\Sigma(X_2-\bar{X}_2)^2$	$(-1)^2 + (-2)^2 + 1^2 + 0^2 + 2^2$	10
S_{x1x2}	$\Sigma(X_1-\bar{X}_1)(X_2-\bar{X}_2)$	(-2)(-1) + (-1)(-2) + 0(1) + 1(0) + 2(2)	8
S_{x1y}	$\Sigma(X_1-\bar{X}_1)(Y-\bar{Y})$	(-2)(-3.8)+(-1)(-2.8)+0(0.2)+1(1.2)+2(5.2)	21.0
S_{x2y}	$\Sigma(X_2-\bar{X}_2)(Y-\bar{Y})$	(-1)(-3.8)+(-2)(-2.8)+1(0.2)+0(1.2)+2(5.2)	17.0

STEP 4: Formulas for b_1 and b_2

For two independent variables:

$$b_1 = \frac{S_{x2x2}S_{x1y} - S_{x1x2}S_{x2y}}{S_{x1x1}S_{x2x2} - S_{x1x2}^2}$$

$$b_2 = \frac{S_{x1x1}S_{x2y} - S_{x1x2}S_{x1y}}{S_{x1x1}S_{x2x2} - S_{x1x2}^2}$$

Compute the denominator:

$$D = S_{x1x1}S_{x2x2} - S_{x1x2}^2 = (10)(10) - (8)^2 = 100 - 64 = 36$$

Now compute numerators

$$\text{Numerator for } b_1 = (10)(21.0) - (8)(17.0) = 210 - 136 = 74$$

$$\text{Numerator for } b_2 = (10)(17.0) - (8)(21.0) = 170 - 168 = 2$$

Hence $b_1 = 74/36 = 2.056$, $b_2 = 2/36 = 0.056$

STEP 5: Compute the intercept b_0

$$b_0 = \bar{Y} - b_1\bar{X}_1 - b_2\bar{X}_2$$

Value of b_0 is = 5.8 – (2.056 *3) – (0.056 * 3) = = 5.8 – 6.336 = - 0.536

Final Regression Equation is $Y = -0.536 + 2.056 X_1 + 0.056 X_2$

Interpretation

- When both X_1 and X_2 = 0, predicted sales = **– 0.536** (the intercept).
- For every unit increase in **advertising expenditure (X_1)**, sales increase by 2.056 **units** (holding X_2 constant).
- For every additional **salesperson (X_2)**, sales increase by **0.667 units** (holding X_1 constant).