Problem 1: Gradient Descent for Polynomial Regression (15 Points)

Point Distribution

Criteria	Points	
Dataset Calculation (x, y pairs correctly computed)	3	Correctly computes y-values for x = {-1, 0, 1, 2}.
Correct Gradient Descent Iterations (3 iterations)	6	Computes three iterations of gradient descent correctly using the
Convergence Explanation	3	Clearly explains whether the algorithm will converge to correct val
General Convergence Analysis	3	Discusses if the algorithm converges irrespective of the starting v

Solution for Problem 1

1.1 Dataset Calculation (3 points)

The given function to be learned is:

$$y = 3x^3 + 4x^2 + 5x + 6$$

We aim to estimate the coefficients (a, b, c, d) in the polynomial regression model:

$$y = ax^3 + bx^2 + cx + d$$

Computing the Dataset Values

Using the given x values:

$$x = \{-1, 0, 1, 2\}$$

we compute (y) values:

(x)	(y) Computation	(y) Value
-1	$3(-1)^3 + 4(-1)^2 + 5(-1) + 6$	(-3 + 4 - 5 + 6 = 2)
0	$3(0)^3 + 4(0)^2 + 5(0) + 6$	(6)
1	$3(1)^3 + 4(1)^2 + 5(1) + 6$	(3 + 4 + 5 + 6 = 18)
2	$3(2)^3 + 4(2)^2 + 5(2) + 6$	(24 + 16 + 10 + 6 = 56)

Thus, the dataset is:

$$[(-1, 2), (0, 6), (1, 18), (2, 56)]$$

1.2. Correct Gradient Descent Iterations (6 points)

Define the Error Function

We use the **Mean Squared Error (MSE)** as the loss function:

$$\text{MSE} = \frac{1}{n} \sum (y - \hat{y})^2$$

where:

$$\hat{y} = ax^3 + bx^2 + cx + d$$

The partial derivatives for **gradient descent updates** are:

$$\frac{\partial E}{\partial a} = -\frac{2}{n}\sum x^3(y - \hat{y})$$
$$\frac{\partial E}{\partial b} = -\frac{2}{n}\sum x^2(y - \hat{y})$$
$$\frac{\partial E}{\partial c} = -\frac{2}{n}\sum x(y - \hat{y})$$
$$\frac{\partial E}{\partial d} = -\frac{2}{n}\sum (y - \hat{y})$$

Perform Gradient Descent Iterations

Initialization

- Set initial values: $a = 1, b = 1, c = 1, d = 1$
- Learning rate: $\alpha = 0.01$
- Number of iterations: **3**

Iteration 1: Compute Predictions and Update Parameters

Compute Predictions

Using:

$$\hat{y}_i = ax_i^3 + bx_i^2 + cx_i + d$$

x	y	$\hat{y}_{initial}$	**Error (** y **-** $\hat{y}$ **)**
-1	2	0	2
0	6	1	5
1	18	4	14
2	56	15	41

Compute Gradients

$$\frac{\partial E}{\partial a} = -\frac{2}{4}((-1)(2) + (0)(5) + (1)(14) + (8)(41)) = -170$$
$$\frac{\partial E}{\partial b} = -\frac{2}{4}((1)(2) + (0)(5) + (1)(14) + (4)(41)) = -90$$
$$\frac{\partial E}{\partial c} = -\frac{2}{4}((-1)(2) + (0)(5) + (1)(14) + (2)(41)) = -46$$
$$\frac{\partial E}{\partial d} = -\frac{2}{4}(2 + 5 + 14 + 41) = -31$$

Update Parameters

$a = a - \alpha \frac{\partial E}{\partial a} = 1 - 0.01(-170) = 2.7$

$b = b - \alpha \frac{\partial E}{\partial b} = 1 - 0.01(-90) = 1.9$

$c = c - \alpha \frac{\partial E}{\partial c} = 1 - 0.01(-46) = 1.46$

$d = d - \alpha \frac{\partial E}{\partial d} = 1 - 0.01(-31) = 1.31$

Iteration 2: Compute Predictions and Update Parameters

Compute Predictions

Using the updated parameters from Iteration 1:

$\hat{y}_i = a_1 x_i^3 + b_1 x_i^2 + c_1 x_i + d_1$

x	y	$\hat{y}$ (Updated)	Error (y - \hat{y})
-1	2	(2.7(-1)^3 + 1.9(-1)^2 + 1.46(-1) + 1.31)	(2 - (-2.7 + 1.9 - 1.46 + 1.31) = 2 - (-0.95) = 2.95)
0	6	(2.7(0)^3 + 1.9(0)^2 + 1.46(0) + 1.31)	(6 - 1.31 = 4.69)
1	18	(2.7(1)^3 + 1.9(1)^2 + 1.46(1) + 1.31)	(18 - (2.7 + 1.9 + 1.46 + 1.31) = 18 - 7.37 = 10.63)
2	56	(2.7(2)^3 + 1.9(2)^2 + 1.46(2) + 1.31)	(56 - (21.6 + 7.6 + 2.92 + 1.31) = 56 - 33.43 = 22.57)

Compute Gradients

$\frac{\partial E}{\partial a} = -\frac{2}{4}(-1(2.95) + 0(4.69) + 1(10.63) + 8(22.57))$

$= -\frac{2}{4}(-2.95 + 0 + 10.63 + 180.56) = -\frac{2}{4}(188.24) = -94.12$

$\frac{\partial E}{\partial b} = -\frac{2}{4}(1(2.95) + 0(4.69) + 1(10.63) + 4(22.57))$

$= -\frac{2}{4}(2.95 + 0 + 10.63 + 90.28) = -\frac{2}{4}(103.86) = -51.93$

$\frac{\partial E}{\partial c} = -\frac{2}{4}(-1(2.95) + 0(4.69) + 1(10.63) + 2(22.57))$

$= -\frac{2}{4}(-2.95 + 0 + 10.63 + 45.14) = -\frac{2}{4}(52.82) = -26.41$

$\frac{\partial E}{\partial d} = -\frac{2}{4}(2.95 + 4.69 + 10.63 + 22.57)$

$= -\frac{2}{4}(40.84) = -20.42$

Update Parameters

Using **learning rate** $\alpha = 0.01$:

$$a_2 = a_1 - \alpha \frac{\partial E}{\partial a} = 2.7 - 0.01(-94.12) = 3.64$$

$$b_2 = b_1 - \alpha \frac{\partial E}{\partial b} = 1.9 - 0.01(-51.93) = 2.42$$

$$c_2 = c_1 - \alpha \frac{\partial E}{\partial c} = 1.46 - 0.01(-26.41) = 1.73$$

$$d_2 = d_1 - \alpha \frac{\partial E}{\partial d} = 1.31 - 0.01(-20.42) = 1.51$$

Iteration 3: Compute Predictions and Update Parameters

Compute Predictions

Using updated values:

x	y	$\hat{y}$ **(Updated)**	**Error (y - \hat{y})**
-1	2	(3.64(-1)^3+2.42(-1)^2+1.73(-1)+1.51)	$2 - (-1.44) = 3.44$
0	6	$3.64(0)^3 + 2.42(0)^2 + 1.73(0) + 1.51$	$6 - 1.51 = 4.49$
1	18	$3.64(1)^3 + 2.42(1)^2 + 1.73(1) + 1.51$	(18 - (3.64 + 2.42 + 1.73 + 1.51) = 18 - 9.3 = 8.7)
2	56	$3.64(2)^3 + 2.42(2)^2 + 1.73(2) + 1.51$	$56 - 43.77 = 12.23$

Compute Gradients and Update Parameters

Following the same method, we get:

$$a_3 = 4.15, \quad b_3 = 2.72, \quad c_3 = 1.88, \quad d_3 = 1.66$$

1.3. Convergence Explanation (3 points)

1. The algorithm **converges gradually** because it minimizes MSE iteratively.
2. Proper learning rate ensures smooth updates.
3. **If the learning rate is too large,** the updates may oscillate and never converge.

1.4. Convergence Analysis (3 points)

Yes, this algorithm will eventually converge to the correct values. This is because the error surface for linear regression is convex (quadratic) even though we're fitting a cubic function. This is because we're minimizing the squared error.

Though the original main function is cubic in terms of x, the function is linear in terms of the coefficents and the squared error function will be quadratic in terms of the coefficents.

A convex error surface has only one global minimum and no local minima. With an appropriate learning rate, gradient descent on a convex surface will always converge to the global minimum, regardless of the starting point.

The algorithm will converge to the correct values irrespective of the starting point and examples, as long as we have enough linearly independent examples (we need at least 4 for a cubic function), the learning rate is appropriate (not too large to cause divergence), and the examples span enough of the input space to constrain all parameters. In this case, our four examples are sufficient to uniquely determine the four parameters of the cubic function.

Problem 2: Learning a Quadratic Function with Limited Models (15 Points)

Point Distribution

Criteria	Points	Description
Correct Solution Identification	10	Correctly determines whether the function (y = ax^2 + bx + c) can be learned using
Explanation of failure	4	Correctly explains why it is not possible mentioning non-Orthogonality
Methodology Explanation (If Yes)	1	If the function can be learned, provides a clear procedure for combining models.

Solution for Problem 2

2.0 Understanding the Given Models

The three regression algorithms provided each serve a specific purpose in learning different

mathematical relationships. A1 is specialized in learning quadratic functions of the form y = ax², which means it can fit data that follows a parabolic curve passing through the origin. This algorithm finds the optimal value of 'a' that minimizes the error between the predicted values and actual data points. A2 is designed to learn linear relationships of the form y = ax, representing straight lines passing through the origin. This algorithm determines the best slope 'a' to fit the data. A3 is the simplest of the three, learning only constant functions y = a, which represent horizontal lines. This means A3 finds the optimal fixed value that best fits the data regardless of the x-values.

2.1 Correct Solution Identification (10 points)

The goal is to determine whether we can **combine these models** to learn the general quadratic function:

$$y = ax^2 + bx + c$$

It is NOT possible to learn the complete function y = ax² + bx + c using any combination of these three algorithms.

2.2 Explanation of failure (4 points)

This identification stems from understanding the fundamental limitations of how regression algorithms work when fitting data. Each algorithm minimizes its error independently across the entire dataset, and there's no way to make them work together to isolate their respective components. The true solution would require simultaneous optimization of all three coefficients (a, b, and c), which cannot be achieved by combining these algorithms in any sequence or combination.

The direct learning of $y = ax^2 + bx + c$ using these algorithms is impossible with the standard polynomial basis $1, x, x^2$, but becomes possible with orthogonalization. Let's prove both parts:

Proof of failure with standard basis:
Consider $y = 3x^2 + 6x + 4$ with $x = -1, 0, 1$:
Data points: $(-1, 1), (0, 4), (1, 13)$

If we try to learn coefficients separately:

A1 attempts to fit ax²

A2 attempts to fit bx
A3 attempts to fit c

We get $y = 7x^2 + 6x + 6$ (incorrect) because the basis functions interfere with each other during learning.

Proof of success with orthogonal basis: Transform to orthogonal basis $[1, x, (x^2 - 2/3)]$:

These basis functions are orthogonal: $\int_{1_i\,j} = 0$ for i ≠ j Now each algorithm can learn its coefficient independently Transform back to get original coefficients (3,6,4)

2.3 Methodology Explanation (1 points)

The key to making this work is orthogonalization. Here's the detailed procedure:

Start with basis transformation. Transform the original function $y = ax^2 + bx + c$ to:
$y = \alpha[(x^2 - 2/3)] + \beta x + \gamma$, where $[1, x, (x^2 - 2/3)]$ are orthogonal.

For the learning process, use A1 to learn α directly (no interference), A2 to learn β directly (no interference) and A3 to learn γ directly (no interference).

Mathematical validation:

For $y = 3x^2 + 6x + 4$:
In orthogonal basis:
Learn coefficient of (x² - 2/3): α = 3
Learn coefficient of x: β = 6
Learn coefficient of 1: γ = 6

Transform back:
$\alpha[(x^2 - 2/3)] + \beta x + \gamma = 3x^2 + 6x + 4$

This works because the orthogonal basis ensures no interference between terms. Each algorithm learns exactly one coefficient and the transformation is reversible.

The solution requires understanding why direct learning fails (non-orthogonal basis interference), knowing how to orthogonalize basis functions, learning coefficients in orthogonal basis and transforming back to original basis. This is similar to techniques like Gram-Schmidt orthogonalization or SOR method from HW1, where orthogonalization ensures independent convergence of components.