Assignment 6

1. **Forward Difference Method** – Calculates the first and second derivatives using Newton's forward difference formula.

2. **Backward Difference Method** – Computes the first derivative using Newton's backward difference formula.

3. **Lagrange Interpolation Method** – Finds the derivative when the data points are not evenly spaced.

4. **Finding Extrema** – Identifies local maxima and minima of the given dataset.

Numerical differentiation is useful when we only have discrete data points and cannot obtain the derivative analytically.

The forward difference formula for the first derivative is given by:

$$f'(x) \approx \frac{-f(x+2h) + 4f(x+h) - 3f(x)}{2h}$$

This method approximates the derivative at a given point using values ahead of it. It is more accurate when the step size h is small.

```
def forward_difference_derivative(x, y, point, h):
    """Computes the first derivative using the Forward Difference Method"""
    idx = np.where(x == point)[0][0]  # Find index of the point
    if idx + 2 >= len(x):
        raise ValueError("Not enough points ahead for forward difference")
    return (-y[idx+2] + 4*y[idx+1] - 3*y[idx]) / (2*h)
```

Backward Difference Method

The backward difference formula for the first derivative is:

$$f'(x) \approx \frac{3f(x) - 4f(x-h) + f(x-2h)}{2h}$$

This method is useful when we do not have enough data points ahead of x and must use points behind it.

```
def backward_difference_derivative(x, y, point, h):

    """Computes the first derivative using the Backward Difference Method"""

    idx = np.where(x == point)[0][0]  # Find index of the point

    if idx < 2:

        raise ValueError("Not enough points behind for backward difference")

    return (3*y[idx] - 4*y[idx-1] + y[idx-2]) / (2*h)
```

3. Lagrange Interpolation for Unequally Spaced Data

When the data points are not evenly spaced, we cannot use the forward or backward difference formulas. Instead, we approximate the derivative using **Lagrange interpolation**.

The Lagrange polynomial P(x) is formed using the given data points, and its derivative is found analytically.

```
def lagrange_derivative(x, y, point):

    """Computes the derivative using Lagrange interpolation"""

    poly = lagrange(x, y)  # Generate Lagrange polynomial

    der_coeffs = np.polyder(poly.coef)  # Differentiate polynomial

    return np.polyval(der_coeffs, point)  # Evaluate derivative at the given point
```

To find **maxima or minima**, we analyze where the first derivative changes sign.

Algorithm:

1. Compute the first derivative using the **central difference method**:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

2. Identify where the derivative changes sign – this indicates a local extrema.

```
def find_extrema(x, y):
    """Finds local maxima or minima in a dataset"""
    derivatives = []

    # Compute central difference for interior points
    for i in range(1, len(x)-1):
        dx = (x[i+1] - x[i-1]) / 2
        dy = (y[i+1] - y[i-1]) / 2
        derivatives.append(dy/dx)

    extrema = []
    for i in range(len(derivatives)-1):
        if derivatives[i] * derivatives[i+1] < 0:  # Sign change
            x_extremum = (x[i+1] + x[i+2]) / 2  # Estimate extrema position
            extrema.append(x_extremum)

    return extrema
```

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange

# Sample data
x = np.array([0, 2, 4, 6, 8])
y = np.array([1, 4, 16, 36, 64])
h = x[1] - x[0]

Tabnine | Edit | Test | Explain | Document
def forward_difference_derivative(x, y, point, h):
    """Computes the first derivative using the Forward Difference Method"""
    idx = np.where(x == point)[0][0]  # Find index of the point
    if idx + 2 >= len(x):
        raise ValueError("Not enough points ahead for forward difference")
    return (-y[idx+2] + 4*y[idx+1] - 3*y[idx]) / (2*h)

# Forward Difference Method
d1_forward = forward_difference_derivative(x, y, 0, h)
print(f"Forward Difference, First Derivative at x=0: {d1_forward:.4f}")


Tabnine | Edit | Test | Explain | Document
def backward_difference_derivative(x, y, point, h):
    """Computes the first derivative using the Backward Difference Method"""
    idx = np.where(x == point)[0][0]  # Find index of the point
    if idx < 2:
        raise ValueError("Not enough points behind for backward difference")
    return (3*y[idx] - 4*y[idx-1] + y[idx-2]) / (2*h)

# Backward Difference Method
x_back = np.array([5, 6, 7, 8, 9])
y_back = np.array([10, 16, 26, 40, 58])
h_back = x_back[1] - x_back[0]


d1_backward = backward_difference_derivative(x_back, y_back, 9, h_back)
print(f"Backward Difference, First Derivative at x=9: {d1_backward:.4f}")
```

```
Tabnine | Edit | Test | Explain | Document
def lagrange_derivative(x, y, point):
    """Computes the derivative using Lagrange interpolation"""
    poly = lagrange(x, y)  # Generate Lagrange polynomial
    der_coeffs = np.polyder(poly.coef)  # Differentiate polynomial
    return np.polyval(der_coeffs, point)  # Evaluate derivative at the given point

# Lagrange Interpolation for Unequally Spaced Data
x_lagr = np.array([1, 2, 4, 7])
y_lagr = np.array([3, 6, 12, 21])
d1_lagrange = lagrange_derivative(x_lagr, y_lagr, 3)
print(f"Lagrange Interpolation, First Derivative at x=3: {d1_lagrange:.4f}")

Tabnine | Edit | Test | Explain | Document
def find_extrema(x, y):
    """Finds local maxima or minima in a dataset"""
    derivatives = []

    # Compute central difference for interior points
    for i in range(1, len(x)-1):
        dx = (x[i+1] - x[i-1]) / 2
        dy = (y[i+1] - y[i-1]) / 2
        derivatives.append(dy/dx)

    extrema = []
    for i in range(len(derivatives)-1):
        if derivatives[i] * derivatives[i+1] < 0:  # Sign change
            x_extremum = (x[i+1] + x[i+2]) / 2  # Estimate extrema position
            extrema.append(x_extremum)

    return extrema

# Finding Extrema
x_ext = np.array([2, 4, 6, 8, 10])
y_ext = np.array([5, 7, 8, 6, 3])
extrema = find_extrema(x_ext, y_ext)
print(f"Extrema found at x ≈ {[f'{x:.4f}' for x in extrema]}")
```

```
ktop/Study/as3web/a.py
Forward Difference, First Derivative at x=0: -0.7500
Backward Difference, First Derivative at x=9: 20.0000
Lagrange Interpolation, First Derivative at x=3: 3.0000
Extrema found at x ≈ ['5.0000']
```