Assigment 6

```
import numpy as np

import matplotlib.pyplot as plt

from scipy.interpolate import lagrange


def forward_difference_derivative(x, y, point, h):

    """

    Calculate first derivative using Newton's Forward Difference Formula

    """

    # Find the index of the point

    idx = np.where(x == point)[0][0]


    if idx + 3 >= len(x):

        raise ValueError("Not enough points ahead for forward difference")


    # First derivative using forward difference formula

    d1 = (-y[idx+2] + 4*y[idx+1] - 3*y[idx]) / (2*h)

    return d1


def forward_second_derivative(x, y, point, h):

    """

    Calculate second derivative using Newton's Forward Difference Formula

    """

    idx = np.where(x == point)[0][0]


    if idx + 3 >= len(x):

        raise ValueError("Not enough points ahead for forward difference")
```

```python
    # Second derivative using forward difference formula
    d2 = (y[idx+2] - 2*y[idx+1] + y[idx]) / (h**2)
    return d2

def backward_difference_derivative(x, y, point, h):
    """
    Calculate first derivative using Newton's Backward Difference Formula
    """
    idx = np.where(x == point)[0][0]

    if idx < 2:
        raise ValueError("Not enough points behind for backward difference")

    # First derivative using backward difference formula
    d1 = (3*y[idx] - 4*y[idx-1] + y[idx-2]) / (2*h)
    return d1

def lagrange_derivative(x, y, point):
    """
    Calculate derivative using Lagrange interpolation
    """
    # Get Lagrange polynomial
    poly = lagrange(x, y)
    # Get derivative coefficients
    der_coeffs = np.polyder(poly.coef)
    # Evaluate derivative at point
    return np.polyval(der_coeffs, point)
```

```
def find_extrema(x, y):
    """
    Find maxima or minima in tabulated data
    """
    # Use central difference for interior points
    derivatives = []
    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)

    # Find where derivative changes sign
    extrema = []
    for i in range(len(derivatives)-1):
        if derivatives[i] * derivatives[i+1] < 0:
            # Linear interpolation to find more precise x value
            x_extremum = (x[i+1] + x[i+2])/2
            extrema.append(x_extremum)

    return extrema

def plot_data_with_derivatives(x, y, points, derivatives, title):
    """
    Plot data points and derivatives
    """
    plt.figure(figsize=(10, 6))
```

```
    # Plot original data
    plt.scatter(x, y, color='blue', label='Data points')
    plt.plot(x, y, 'b--', alpha=0.5)

    # Plot derivative vectors
    for point, deriv in zip(points, derivatives):
        # Plot derivative vector
        plt.quiver(point, np.interp(point, x, y), 1, deriv,
                  angles='xy', scale_units='xy', scale=2,
                  color='red', label='Derivative')

    plt.xlabel('x')
    plt.ylabel('y')
    plt.title(title)
    plt.legend()
    plt.grid(True)
    plt.show()

# Task 1: First Derivative Using Forward Difference
print("Task 1: First Derivative Using Forward Difference")
x1 = np.array([0, 2, 4, 6, 8])
y1 = np.array([1, 4, 16, 36, 64])
h1 = x1[1] - x1[0]
d1_forward = forward_difference_derivative(x1, y1, 0, h1)
print(f"First derivative at x=0: {d1_forward:.4f}")
plot_data_with_derivatives(x1, y1, [0], [d1_forward], "Forward Difference First Derivative")
```

```
# Task 2: Second Derivative Using Forward Difference

print("\nTask 2: Second Derivative Using Forward Difference")

d2_forward = forward_second_derivative(x1, y1, 0, h1)

print(f"Second derivative at x=0: {d2_forward:.4f}")


# Task 3: First Derivative Using Backward Difference

print("\nTask 3: First Derivative Using Backward Difference")

x3 = np.array([5, 6, 7, 8, 9])

y3 = np.array([10, 16, 26, 40, 58])

h3 = x3[1] - x3[0]

d1_backward = backward_difference_derivative(x3, y3, 9, h3)

print(f"First derivative at x=9: {d1_backward:.4f}")

plot_data_with_derivatives(x3, y3, [9], [d1_backward], "Backward Difference First Derivative")


# Task 4: Derivative Using Unequally Spaced Values

print("\nTask 4: Derivative Using Unequally Spaced Values")

x4 = np.array([1, 2, 4, 7])

y4 = np.array([3, 6, 12, 21])

d1_lagrange = lagrange_derivative(x4, y4, 3)

print(f"First derivative at x=3: {d1_lagrange:.4f}")

plot_data_with_derivatives(x4, y4, [3], [d1_lagrange], "Lagrange Interpolation Derivative")


# Task 5: Maxima or Minima

print("\nTask 5: Maxima or Minima")

x5 = np.array([2, 4, 6, 8, 10])

y5 = np.array([5, 7, 8, 6, 3])

extrema = find_extrema(x5, y5)
```

```
print(f"Extrema found at x ≈ {[f'{x:.4f}' for x in extrema]}")


plt.figure(figsize=(10, 6))
plt.scatter(x5, y5, color='blue', label='Data points')
plt.plot(x5, y5, 'b--', alpha=0.5)
for x_ext in extrema:
    plt.axvline(x=x_ext, color='r', linestyle='--', alpha=0.5)
    plt.scatter(x_ext, np.interp(x_ext, x5, y5), color='red', label='Extremum')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Function Extrema')
plt.legend()
plt.grid(True)
plt.show()
```