ASSIGNEMET 02

1. **Write a Python script to implement the two-dimensional Steepest Descent Method for optimizing a function. Use the following specifications:**

- The objective function to minimize is $f(x,y) = x^2 - xy + y^2$
- The gradient of the function is $\nabla f(x,y) = [2x-y, -x+2y]$
- Initialize the starting point at (1, 1)
- Set the step size (α) to 0.1.
- Use a convergence tolerance of 1×10^{-6}
- Limit the number of iterations to 1000.

Tasks:

- Print the optimal point and the function value at that point after convergence.
- Plot the function and the path to optimal value for each case in a 2D contour.

Here are the steps for implementing the Steepest Descent Method with the given specifications:

1. Define the Objective Function and Gradient

- Objective Function:

- Define a function f(x, y) that calculates the value of f(x, y) = x^2 - xy + y^2.

- Gradient Function:

- Define a function ∇f(x, y) that calculates the gradient: [2x - y, -x + 2y].

2. Initialize Parameters

-Starting Point:

- Set initial values for x and y. For example, (x, y) = (1, 1) .

- Step Size (α)

- Define a step size alpha. For example, alpha = 0.1.

- Convergence Tolerance:

- Define a small value for convergence tolerance, e.g., 10^{-6}.

- Maximum Iterations:
  - Set a maximum number of iterations, e.g., 1000.

3. Steepest Descent Method Iteration

1. Start Loop:
   - Iterate up to the maximum number of iterations.
2. Compute Gradient:
   - Calculate the gradient at the current point (x, y) .
3. Update Point:
   - Update x and y using the formula:
     - $x_{new} = x - \text{alpha}\ \nabla f_x$
     - $y_{new} = y - \text{alpha}\ \nabla f_y$

4. Check Convergence:
   - Compute the distance between the new and old points:
     - If the distance norm(x_{new}- x, y_{new}- y) is less than the tolerance, break the loop. (linalg.norm([x_{new}- x, y_{new}- y]))

5. Update Current Point:
   - Set x = x_{new} and y = y_{new}.

6. Record Path:
   - Save the current point (x, y) to a list of path points.

4. Print Results

- Optimal Point:
  - Print the final coordinates of (x, y) after convergence.
- Function Value:
  - Compute and print the function value at the optimal point f(x, y).

5. Plotting

1. Prepare Grid:

- Create a grid of x and y values over a range.

Syntax;

```
x_vals = np.linspace(-2, 2, 400)  # Define the range and number of x values
y_vals = np.linspace(-2, 2, 400)  # Define the range and number of y values
X, Y = np.meshgrid(x_vals, y_vals)  # Create a 2D grid of x and y values
```

2. Calculate Function Values:

- Compute the function values f(x, y) for each point in the grid.

Syntax: `Z = f(X, Y)  # Compute the function values over the grid`

3. Create Plot:

Syntax:

```
plt.figure(figsize=(10, 6))
# Plot filled contour plot
contour = plt.contourf(X, Y, Z, levels=50, cmap='viridis', alpha=0.6)
plt.colorbar(contour, label='Function value')
```

4. Plot Path :

```
plt.plot(trajectory_x, trajectory_y, 'r-o', markersize=5, label='Steepest Descent Path')
```

2. You are analyzing the cooling of a cylindrical object using the lumped capacitance method. The object's initial temperature is 80°C, and it is placed in an environment with a constant ambient temperature of 20°C. You want to study how different heat transfer coefficients affect the temperature of the object over time.

Given :

- Initial temperature of the object: $\mathbf{T_0 = 80^0C}$

- Ambient temperature: $\mathbf{T_\infty = 20^0C}$

- Density of the object : $\boldsymbol{\rho} = \mathbf{1000\ kg}/\boldsymbol{m^3}$

- Heat capacity (C) of the object: $\mathbf{C = 2000\ J/K}$

- Surface area of the object: **A = 0.5 $\mathbf{m^2}$**

- Volume of the object: **V = 0.25 $\mathbf{m^3}$**

- Time to analyze: t = 0 to 120

You have the following list of heat transfer coefficients: **h = [5, 10, 15] w/$\mathbf{m^2}$ K**

Tasks:

1. Calculate the Temperature: For each heat transfer coefficient, calculate the temperature of the object when t = 0 to12o using the lumped capacitance formula: T(t) = $T_\infty + (T_0 - T_\infty)e^{\frac{-hAt}{\rho v C}}$

2. Print Results: For each heat transfer coefficient, print the calculated temperature of the object from 0 to 120 s.

3. Plot Results: Create a plot showing how the temperature of the object changes with different heat transfer coefficients. Customize the plot with the following attributes:

   - Use different colors and line styles for each heat transfer coefficient.

   - Add markers for each data point.

   - Include a legend to distinguish between different heat transfer coefficients.

   - Add axis labels, a title, and grid lines for better readability.

Hints: 1. Import Libraries

- Import numpy for numerical operations

- Import matplotlib.pyplot for plotting

2. Define Parameters

- Set the initial temperature (T0)

- Set the ambient temperature (T_∞)

- Define the surface area (A)

- Define the volume (V)

- Set the density (rho)

- Set the heat capacity (C)

- Create a time array ranging from 0 to 120 seconds with 1-second intervals

- Define a list of heat transfer coefficients (h)

3. Initialize Plot

- Create a new figure for plotting

- Define colors and line styles for different heat transfer coefficients

4. Loop Over Heat Transfer Coefficients

- For each heat transfer coefficient (h_i) in the list:

- Compute the temperature at each time step using the lumped capacitance formula

- Plot the temperature versus time with a specific color and line style

- Add markers and labels to distinguish different coefficients

5. Customize and Show Plot

- Add labels for the x-axis and y-axis

- Add a title to the plot

- Add a legend to identify different heat transfer coefficients

- Add grid lines for better readability

- Display the plot