Name: Phan Tran Thanh Huy

ID: ITCSIU22056

Lab 7

Activity 1: First Derivative

Code:

```python
import math

# Given function and its exact derivative
def f(x):
    return math.sin(x)

def f_exact_derivative(x):
    return math.cos(x)

# Given parameters
xi = math.pi / 4
h = 0.1
xi_minus_1 = xi - h
xi_plus_1 = xi + h

# Approximate derivatives
forward_diff = (f(xi_plus_1) - f(xi)) / h
backward_diff = (f(xi) - f(xi_minus_1)) / h
central_diff = (f(xi_plus_1) - f(xi_minus_1)) / (2 * h)

# Exact derivative
true_value = f_exact_derivative(xi)

# Error calculations
def compute_errors(approx, true):
    abs_error = abs(approx - true)
    rel_error = abs_error / abs(true) * 100  # relative error in %
    return abs_error, rel_error

fd_abs, fd_rel = compute_errors(forward_diff, true_value)
bd_abs, bd_rel = compute_errors(backward_diff, true_value)
cd_abs, cd_rel = compute_errors(central_diff, true_value)

# Print results
print(f"At x = π/4 ≈ {xi:.5f}, with h = {h}")
print("\nApproximate Derivatives:")
print(f"  Forward Difference: {forward_diff:.6f}")
print(f"  Backward Difference: {backward_diff:.6f}")
print(f"  Central Difference: {central_diff:.6f}")
print(f"\nExact Derivative (cos(π/4)): {true_value:.6f}")

print("\nErrors:")
print(f"  FD - Absolute Error: {fd_abs:.6e}, Relative Error: {fd_rel:.2f}%")
print(f"  BD - Absolute Error: {bd_abs:.6e}, Relative Error: {bd_rel:.2f}%")
print(f"  CD - Absolute Error: {cd_abs:.6e}, Relative Error: {cd_rel:.2f}%")
```

Result:

```
At x = π/4 ≈ 0.78540, with h = 0.1

Approximate Derivatives:
  Forward Difference: 0.670603
  Backward Difference: 0.741255
  Central Difference: 0.705929

Exact Derivative (cos(π/4)): 0.707107

Errors:
  FD - Absolute Error: 3.650381e-02, Relative Error: 5.16%
  BD - Absolute Error: 3.414796e-02, Relative Error: 4.83%
  CD - Absolute Error: 1.177922e-03, Relative Error: 0.17%
```

Activity 2: Second Derivative with Central Difference

Code:

```python
import math

# Define the function
def f(x):
    return math.exp(-x**2)

# Exact second derivative of f''(x) = (4x^2 - 2) * e^(-x^2)
def f_double_prime_exact(x):
    return (4 * x**2 - 2) * math.exp(-x**2)

# Given parameters
x = 0.5
h = 0.1

# Central Difference Approximation for second derivative
f_x_plus = f(x + h)
f_x = f(x)
f_x_minus = f(x - h)

f_double_prime_approx = (f_x_plus - 2 * f_x + f_x_minus) / h**2

f_double_prime_true = f_double_prime_exact(x)

# Error calculations
abs_error = abs(f_double_prime_approx - f_double_prime_true)
rel_error = abs_error / abs(f_double_prime_true) * 100

# Print results
print(f"At x = {x}, with h = {h}")
print(f"\nApproximate f''(x) using Central Difference: {f_double_prime_approx:.6f}")
print(f"Exact f''(x): {f_double_prime_true:.6f}")
print(f"\nAbsolute Error: {abs_error:.6e}")
print(f"Relative Error: {rel_error:.2f}%")
```

Result:

```
PS E:\Homework\TMC\Lab7> & C:/Python312/python.exe e:/Homework/TMC/Lab7/p2.py
At x = 0.5, with h = 0.1

Approximate f''(x) using Central Difference: -0.778145
Exact f''(x): -0.778801

Absolute Error: 6.556725e-04
Relative Error: 0.08%
```

Activity 3: Derivative from Tabular Data (Discrete)

Code:

```python
# Given data
x_vals = [1.0, 1.1, 1.2, 1.3, 1.4]
f_vals = [2.71, 3.00, 3.32, 3.67, 4.05]
h = 0.1

# Central difference function for discrete data
def central_difference(x_data, f_data, h, target_x):
    try:
        i = x_data.index(target_x)
        if i == 0 or i == len(x_data) - 1:
            raise ValueError("Cannot compute central difference at the boundary point.")
        return (f_data[i + 1] - f_data[i - 1]) / (2 * h)
    except ValueError as e:
        return str(e)

# Points to evaluate
target_points = [1.1, 1.2, 1.3]

# Compute and display results
print("Central Difference Approximation:")
for x in target_points:
    result = central_difference(x_vals, f_vals, h, x)
    print(f"f'({x}) ≈ {result:.4f}" if isinstance(result, float) else f"Error at x = {x}: {result}")

```

Result:

```
PS E:\Homework\TMC\Lab7> & C:/Python312/python.exe e:/Homework/TMC/Lab7/p3.py
Central Difference Approximation:
f'(1.1) ≈ 3.0500
f'(1.2) ≈ 3.3500
f'(1.3) ≈ 3.6500
```

Activity 4: Error Behavior with Varying Step Size

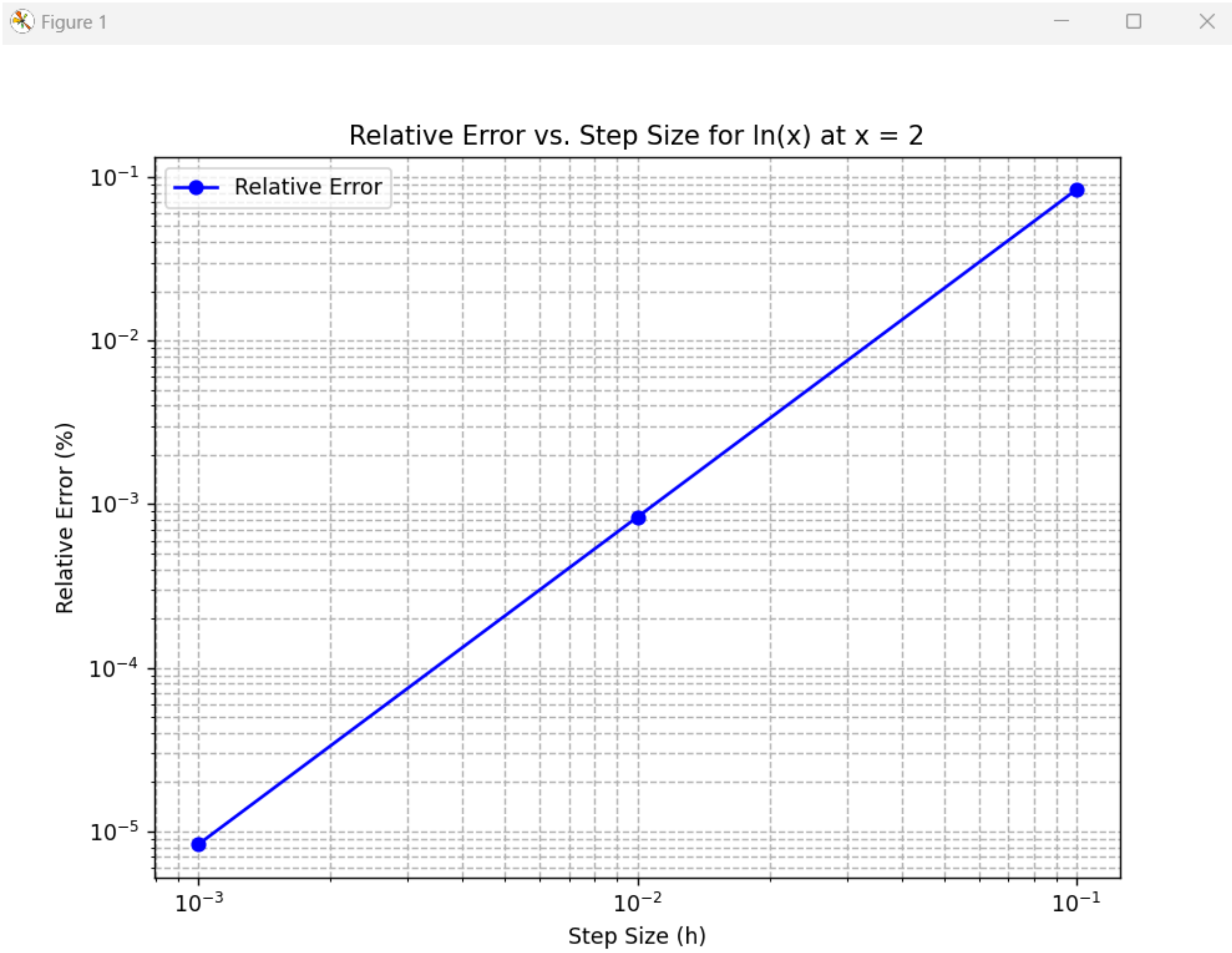
Code:

```python
import numpy as np
import matplotlib.pyplot as plt

x = 2.0
h_values = [0.1, 0.01, 0.001]
exact = 1 / x  # Exact derivative: 1/x
rel_errors = []

print("Error Behavior for ln(x) at x = 2 (Central Difference)")
for h in h_values:
    f_x_plus_h = np.log(x + h)
    f_x_minus_h = np.log(x - h)
    cd = (f_x_plus_h - f_x_minus_h) / (2 * h)
    abs_error = abs(exact - cd)
    rel_error = (abs_error / exact) * 100
    rel_errors.append(rel_error)
    print(f"h = {h}: Derivative = {cd:.9f}, Absolute Error = {abs_error:.9f}, Relative Error = {rel_error:.6f}%")

# Plotting
plt.figure(figsize=(8, 6))
plt.loglog(h_values, rel_errors, 'bo-', label='Relative Error')
plt.xlabel('Step Size (h)')
plt.ylabel('Relative Error (%)')
plt.title('Relative Error vs. Step Size for ln(x) at x = 2')
plt.grid(True, which="both", ls="--")
plt.legend()
plt.savefig('error_vs_h.png')  # Save the plot
plt.show()  # Display the plot
```

Result:

```
h = 0.1: Derivative = 0.500417293, Absolute Error = 0.000417293, Relative Error = 0.083459%
h = 0.01: Derivative = 0.500004167, Absolute Error = 0.000004167, Relative Error = 0.000833%
h = 0.001: Derivative = 0.500000042, Absolute Error = 0.000000042, Relative Error = 0.000008%
```

Plot:



Report & Discussion Questions

1. Which method is most accurate for estimating first derivative?

Answer: The Central Difference (CD) method is generally the most accurate among the three (Forward, Backward, Central). This is because:

- FD and BD are first-order accurate,
- CD is second-order accurate.

In Activity #1, the CD method gave an estimate of the derivative at $x=\pi/4$ that was closest to the exact value.

2. How does central difference compare to forward and backward?

Answer: The Central Difference uses information on both sides of the point x_i, while Forward and Backward use only one-sided information. This results in:

- Better accuracy with CD,
- Symmetric error cancellation in CD,
- FD/BD are more prone to bias and larger truncation error.

```
Errors:
  FD - Absolute Error: 3.650381e-02, Relative Error: 5.16%
  BD - Absolute Error: 3.414796e-02, Relative Error: 4.83%
  CD - Absolute Error: 1.177922e-03, Relative Error: 0.17%
```

3. What happens to error when h becomes very small?

Answer: As $h \to 0$:

- Truncation error decreases (good),
- But round-off error may increase (bad),
- There's an optimal range for h where total error is minimized.

We can observe it in activity #4.

4. Is numerical differentiation from tabular data reliable? When?

Answer: Yes, numerical differentiation is reliable from tabular data, if:

- The data is accurate and smooth,
- The step size h is uniform and reasonably small,

- Avoiding boundary points when using CD (or switch to FD/BD there).

In Activity #3, CD gave good derivative estimates from the discrete values of f(x). However:

- Noisy or sparse data reduces reliability,
- Interpolation or smoothing may be needed for better results.