LAB 1

Dr. Fatimah Alshahrani

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns  # Import seaborn
# Set the seaborn style
sns.set_theme(style="whitegrid")
```

This code plot the exact solution and the forward finite difference method:

```
# This defines the step size h for the numerical computation.
h = 0.1
# This creates a grid x from 0 to 2  with intervals of h.
x = np.arange(0, 2*np.pi, h)
# This computes the cosine function values for each point in the
                                    grid x.
y = np.cos(x)

# This calculates the forward differences of y 'vector' divided by
                                    h, which is an approximation of
                                    the derivative of cos(x).
forward_diff = np.diff(y)/h
# This defines a new grid x_diff that corresponds to the forward
                                    differences. It excludes the last
                                     point of x since np.diff reduces
                                     the array size by one.
x_diff = x[:-1:]
# This calculates the exact solution for the derivative of cos(x),
                                    which is -sin(x), evaluated at
                                    x_diff.
exact_solution = -np.sin(x_diff)

# This code block creates a plot showing both the finite difference
                                     approximation and the exact
                                    solution.
plt.figure(figsize = (12, 8))
plt.plot(x_diff, forward_diff, '--', \
         label = 'Finite difference approximation')
plt.plot(x_diff, exact_solution, \
         label = 'Exact solution')
plt.legend()
plt.show()

#Finally, this computes the maximum error between the numerical
                                    derivative and the exact solution
                                    , then prints the result.
max_error = max(abs(exact_solution - forward_diff))
print(max_error)
```

EXAMPLE 3 Page 40 implementation in Python

```
 import numpy as np

# Function and point of interest
x0 = 0.7

# Step sizes
h = 2.**-np.arange(1, 30)

# Finite difference approximation of the derivative
df = (np.cos(x0 + h) - np.cos(x0)) / h

# True value of the derivative
true_value = -np.sin(x0)

# Display the results with formatted output
print("k  | Approximation         | Ratio of errors | Relative
                                    differences")
print("---|----------------------|-----------------|
                                    ----------------------")
previous_approximation = None
previous_error = None
for k in range(1, len(h) + 1):
    approximation = df[k - 1]
    error = np.abs(approximation - true_value)
    ratio = np.abs(previous_error / error) if previous_error is not
                                        None else "   "
    relative_difference = np.abs((approximation -
                                        previous_approximation) /
                                        previous_approximation) if
                                        previous_approximation is not
                                        None else "   "
    # Format the output
    formatted_approximation = f"{approximation:.15f}"
    formatted_ratio = f"{ratio:.6f}" if isinstance(ratio, float)
                                        else ratio
    formatted_relative_difference = f"{relative_difference:.10f}"
                                        if isinstance(
                                        relative_difference, float)
                                        else relative_difference
    print(f"{k:<3}| {formatted_approximation:<21}| {formatted_ratio
                                        :<17}| {
                                        formatted_relative_difference
                                        :<20}")
    previous_error = error
    previous_approximation = approximation

# Print the true value of the derivative for reference
print(f"\nTrue value of the derivative: {true_value:.17f}")
```