

9-9.6-17

Harshil Rathan Y - EE24BTECH11064

January 9, 2025

Question

Find the equation of a curve passing through the point $(0, 2)$ given that the sum of the coordinates of any point on the curve exceeds the magnitude of the slope of the tangent to the curve at that point by 5

Equation

$$x + y = \frac{dy}{dx} + 5 \quad (1)$$

$$\frac{dy}{dx} - y = x - 5 \quad (2)$$

Theoretical Solution

Comparing 2 with $\frac{dy}{dx} + Py = Q$,

$$P = -1 \text{ and } Q = x - 5 \quad (3)$$

I.F is 4

$$e^{\int P dx} = e^{-x} \quad (4)$$

Solution is 5

$$y \cdot e^{-x} = \int (x - 5) e^{-x} dx + c \quad (5)$$

On applying product rule

$$\int A \cdot B dx = A \int B dx - \int \frac{d}{dx}(A) \left(\int B dx \right) dx \quad (6)$$

Theoretical Solution

$$ye^{-x} = -(x - 5)e^{-x} + \int e^{-x} dx + c \quad (7)$$

On Simplyfying

$$x + y = 4 + ce^x \quad (8)$$

To find c, put $x=0$ and $y=2$ in 8

$$c = -2 \quad (9)$$

The curve equation is

$$x + y = 4 - 2e^x \quad (10)$$

Solution by Laplace Transform

$$\frac{dy}{dx} = x + y - 5 \quad (11)$$

Applying the Laplace transform to both sides:

$$\mathcal{L} \left\{ \frac{dy}{dx} \right\} = \mathcal{L} \{x + (y - 5)\} \quad (12)$$

$$sY(s) - y(0) = \frac{1}{s^2} + Y(s) - \frac{5}{s} \quad (13)$$

$$Y(s) = \frac{\frac{1}{s^2} - \frac{5}{s} + y(0)}{s - 1} \quad (14)$$

$$Y(s) = \frac{1}{s^2(s - 1)} - \frac{1}{s(s - 1)} \cdot 5 + \frac{y(0)}{s - 1} \quad (15)$$

Solution by Laplace Transform

Inverse Laplace Transform of Each Term

$$\frac{1}{s^2(s-1)} = \frac{A}{s} + \frac{B}{s^2} + \frac{C}{s-1} \quad (16)$$

$$1 = A s (s-1) + B (s-1) + C s^2 \quad (17)$$

we get

$$\frac{1}{s^2(s-1)} = -\frac{1}{s} - \frac{1}{s^2} + \frac{1}{s-1} \quad (18)$$

Taking the inverse Laplace transform

$$\mathcal{L}^{-1} \left\{ \frac{1}{s} - \frac{1}{s-1} \right\} = -1 - x + e^x \quad (19)$$

Solution by Laplace Transform

Inverse Laplace of $\frac{5}{s(s-1)}$

$$\frac{5}{s(s-1)} = \frac{A}{s} + \frac{B}{s-1} \quad (20)$$

$$5 = A(s-1) + Bs \quad (21)$$

Solving gives $A = 5$ and $B = -5$

$$\frac{5}{s(s-1)} = \frac{5}{s} - \frac{5}{s-1} \quad (22)$$

Taking the inverse Laplace transform

$$\mathcal{L}^{-1} \left\{ \frac{5}{s} - \frac{5}{s-1} \right\} = 5 - 5e^x \quad (23)$$

Solution by Laplace Transform

Inverse Laplace of $\frac{y(0)}{s-1}$

$$\mathcal{L}^{-1} \left\{ \frac{y(0)}{s-1} \right\} = y(0)e^x \quad (24)$$

combining the results from all parts, we have the solution for $y(x)$ The general solution too this differential equation is

$$y(x) = 4 - x + (y(0) - 4)e^x \quad (25)$$

$$y(x) = 4 - x + ce^x \quad (26)$$

To find c , put $x=0$ and $y=2$ in 26

$$c = -2 \quad (27)$$

The curve is

$$x + y = 4 - 2e^x \quad (28)$$

Verification

Now let's verify the solution computationally from the definition of $\frac{dy}{dx}$

$$y_{n+1} = y_n + \frac{dy}{dx} \cdot h \quad (29)$$

From the differential equation given,

$$\frac{dy}{dx} = x + y - 5 \quad (30)$$

Substituting 30 in 29

$$y_{n+1} = y_n + (x_n + y_n - 5) \cdot h \quad (31)$$

C Code - Eulers Method

```
#include <stdio.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

// Function to calculate dy/dx for the differential equation
float dy_dx(float x, float y) {
    return x + y - 5; // Differential equation dy/dx = x + y - 5
}

// Function to calculate points using Euler's method
void points(float x_0, float y_0, float x_end, float h, float *
    x_points, float *y_points, int steps) {
    float x_n = x_0;
    float y_n = y_0;

    for (int i = 0; i < steps; i++) {
        x_points[i] = x_n; // Store current x value
        y_points[i] = y_n; // Store current y value
    }
}
```

C Code - Eulers Method

```
// Calculate the next y using Euler's method
y_n = y_n + h * dy_dx(x_n, y_n);
x_n = x_n + h; // Move to the next x value
}
// Main function
int main() {
    float x_0 = 0.0; // Initial condition for x
    float y_0 = 2.0; // Initial condition for y
    float x_end = 1.0; // Final value of x
    float step_size = 0.001; // Step size for Euler's method
    int steps = (int)((x_end - x_0) / step_size) + 1;
    // Allocate memory for arrays to store points
    float *x_points = (float *)malloc(steps * sizeof(float));
    float *y_points = (float *)malloc(steps * sizeof(float));
    if (x_points == NULL || y_points == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }
}
```

C Code - Eulers Method

```
// Call the points function
points(x_0, y_0, x_end, step_size, x_points, y_points, steps)
    ;

// Print the calculated points (optional, for debugging
    purposes)
printf("x\t\tty\n");
for (int i = 0; i < steps; i++) {
    printf("%f\t%f\n", x_points[i], y_points[i]);
}

// Free allocated memory
free(x_points);
free(y_points);

return 0;
}
```

Python Code

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

# Load the shared library
lib = ctypes.CDLL("./c.so")

# Define the function signature for points
lib.points.argtypes = [
    ctypes.c_float, # x_0
    ctypes.c_float, # y_0
    ctypes.c_float, # x_end
    ctypes.c_float, # h
    np.ctypeslib.ndpointer(dtype=np.float32, ndim=1), # x_points
    np.ctypeslib.ndpointer(dtype=np.float32, ndim=1), # y_points
    ctypes.c_int # stepsclass 12 differential equations
]
```

Python Code

```
# Parameters for simulation
x_0 = 0.0 # Initial condition for x
y_0 = 2.0 # Initial condition for y
x_end = 1.0 # Final value of x
step_size = 0.001 # Reduced step size for higher accuracy
steps = int((x_end - x_0) / step_size) + 1

# Create numpy arrays to hold the points
x_points = np.zeros(steps, dtype=np.float32)
y_points = np.zeros(steps, dtype=np.float32)

# Call the points function from the C shared library
lib.points(x_0, y_0, x_end, step_size, x_points, y_points, steps)

# Define the theoretical solution with C = -2
def theoretical_solution(x):
    return (-x + 4 - 2* np.exp(x)) # C = -2
```

```
# Generate theoretical values for y
x_theory = np.linspace(x_0, x_end, 1000)
y_theory = theoretical_solution(x_theory)

# Plot the results
plt.figure(figsize=(10, 6))

# Plot Euler's method results
plt.plot(x_points, y_points, 'ro-', markersize=2, linewidth=4,
         label="sim")

# Plot the theoretical solution
plt.plot(x_theory, y_theory, 'b-', linewidth=2, label="theory")
```



```
1 # Add labels, title, grid, and legend
2 plt.xlabel("x")
3 plt.ylabel("y")
4 plt.grid(True, linestyle="--")
5 plt.legend()
6
7 # Display the plot
8 plt.show()
```

Plot

