12-9-6-2

EE23BTECH11060 - SRUTHI BIJILI *

Question

Find the general equation of

$$\frac{dy}{dx} + 3y = e^{-2x} \tag{1}$$

Theoretical solution

Laplace transform of the derivative $\frac{dy}{dx}$ is given by

$$\mathcal{L}\left\{\frac{dy}{dx}\right\} = sY(s) - y(0) \tag{2}$$

where Y(s) is the laplace transform of y(x),and y(0) is the initial condition. taking laplace transform on both sides of the equation:

$$\mathcal{L}\left\{\frac{dy}{dx} + 3y\right\} = \mathcal{L}\left\{e^{-2x}\right\} \tag{3}$$

$$\implies \mathcal{L}\left\{\frac{dy}{dx}\right\} + \mathcal{L}\left\{y\right\} = \mathcal{L}\left\{e^{-2x}\right\} \tag{4}$$

$$\mathcal{L}\left\{e^{-2x}\right\} = \frac{1}{s+2} \tag{5}$$

$$\implies sY(s) - y(0) + 3Y(s) = \frac{1}{s+2} \tag{6}$$

Theoretical solution

rearranging the equation

$$Y(s) = \frac{1}{(s+2)(s+3)} + \frac{y(0)}{s+3} \tag{7}$$

$$Y(s) = \frac{1}{(s+2)} - \frac{1}{(s+3)} + \frac{y(0)}{s+3} \tag{8}$$

Taking the inverse laplace transform of each term:

$$\mathcal{L}^{-1}\{Y(s)\} = \mathcal{L}^{-1}\left\{\frac{1}{s+2}\right\} + \mathcal{L}^{-1}\left\{\frac{1}{s+3}\right\} \tag{9}$$

thus,the solution by laplace transform is

$$y(x) = e^{-2x} - e^{-3x} + y(0)\,e^{-3x} \tag{10}$$

$$\implies y(x) = e^{-2x} + (y(0) - 1)\,e^{-3x} \tag{11}$$

$$\tag{12}$$

Method of finite differences

The derivative of f(x) can be written as

$$\frac{df}{dx} = \frac{f(x+h) - f(x)}{h} \tag{13}$$

$$\implies f(x+h) = f(x) + h \cdot \frac{df}{dx} \tag{14}$$

from the above question

$$\frac{dy}{dx} + 3y = e^{-2x} \tag{15}$$

$$\implies \frac{dy}{dx} = e^{-2x} - 3y \tag{16}$$

$$\implies y(x+h) = y(x) + h\left(e^{-2x} - 3y\right) \tag{17}$$

for $x \in [x_0, x_n]$ divide into equal parts by difference h
Let us assume that $x_0 = 0, y_0 = 1$
Let $x_1 = x_0 + h$ then

$$y_1 = y_0 + h\left(e^{-2x_0} - 3y_0\right) \tag{18}$$

Method of finite differences

To obtain the graph repeat the process until sufficient points to plot the graph and the general equation will be

$$x_{n+1} = x_n + h \tag{19}$$

$$y_{n+1} = y_n + h\left(e^{-2x_n} - 3y_n\right) \tag{20}$$

The curve generalised using the method of finite differences for the given question taking $x_0 = 0, y_0 = 1,\ h = 0.01$ and running iterations for 100 times

Simulation

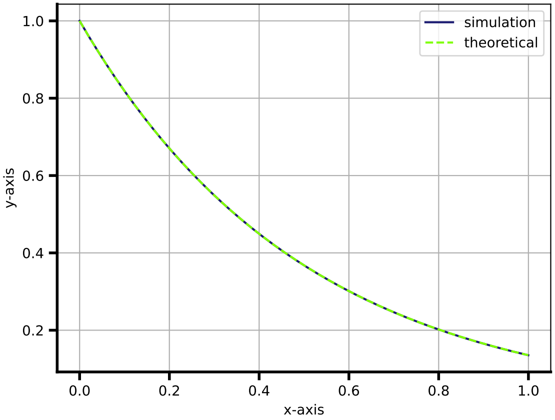

Figure:

C-Code

```c
#include<stdio.h>
#include<math.h>
double h=0.0002;
float dydx(float x, float y){
        return (-3*y+exp(-2*x));
}
void vals(float *x,float *y,int n){
        for(int i=0;i<n;i++){
                *y+=dydx(*x,*y)*h;
                *x+=h;
        }
}
```

Python Code

```
import math
import ctypes
import matplotlib.pyplot as plt

# Loading the .so file
lib = ctypes.CDLL('./funcs.so')

# Definations of return types and argument types in C code
lib.vals.argtypes = [ctypes.POINTER(ctypes.c_float),
↪  ctypes.POINTER(ctypes.c_float), ctypes.c_int]
lib.vals.restype = None

lib.dydx.argtypes = [ctypes.c_float, ctypes.c_float]
lib.dydx.restype = ctypes.c_float

# Set the initial values and parameters
x = ctypes.c_float(0.0)
y = ctypes.c_float(1.0)
```

frametitlePython Code

```
n = 5000
h = 0.0002
# Creating arrays to store the results for plotting
x_vals = []
y_vals = []
theory_values = []
# Generate values using the C function
for i in range(n):
    x_vals.append(float(x.value))
    y_vals.append(float(y.value))
# Compute theoretical values for y using the known solution of
↪   the ODE
    theory_y = math.exp(-2*x.value)
    theory_values.append(theory_y)
    # Call the C function to update x and y
    lib.vals(ctypes.byref(x), ctypes.byref(y), 1)
```

Python Code

```
#plotting
sim_line, = plt.plot(x_vals, y_vals, label="simulation",
↪  color='midnightblue')
theory_line, = plt.plot(x_vals, theory_values,
↪  label="theoretical", color='chartreuse', linestyle='--')
plt.xlabel("x-axis")
plt.ylabel("y-axis")
# Customize axis spines for thick black axes
ax = plt.gca()  # Get the current axes
ax.spines['bottom'].set_color('black')  # Bottom axis
ax.spines['bottom'].set_linewidth(2)    # Set thickness
ax.spines['left'].set_color('black')    # Left axis
ax.spines['left'].set_linewidth(2)      # Set thickness
# Customize tick parameters for thicker black ticks
ax.tick_params(axis='both', colors='black', width=2, length=6)
plt.legend(handles=[sim_line, theory_line])
plt.grid(True)
plt.show()
```