12.9.7.15 Presentation

G. Abhimanyu Koushik
EE24BTECH11024

January 8, 2025

1. Problem
2. Solution
   - Input Parameters
   - Laplace Transform properties
   - Equation solving
   - Computational Solution
3. Plot of the function
4. C Code
5. Python Code

Problem Statement

The population of a village increases continuously at the rate proportional to the number of its inhabitants present at any time. If the population of the village was 20,000 in 1999 and 25,000 in the year 2004, what will be the population of the village in 2009?

Input Parameters

Variable	Description
n	Order of given differential equation
a_i	Coeefficient of ith derivative of the function in the equation
c	constant in the equation
y^i	ith derivative of given function
$\mathbf{y}(t)$	$\begin{pmatrix} c \\ y(t) \\ y'(t) \\ \vdots \\ y^{n-1}(t) \end{pmatrix}$
h	stepsize, taken to be 0.001
$u(x)$	Unit step function
k_0	proportionality constant

Laplace Transform properties

Properties of Laplace tranform

$$\mathcal{L}\left(y'\right) = s\mathcal{L}\left(y\right) - y\left(0\right) \tag{3.1}$$

$$\mathcal{L}\left(1\right) = \frac{1}{s} \tag{3.2}$$

$$\mathcal{L}\left(cf\left(t\right)\right) = c\mathcal{L}\left(f\left(t\right)\right) \tag{3.3}$$

$$\mathcal{L}\left(f\left(t\right)\right) = F\left(s\right) \implies \mathcal{L}\left(e^{at}f\left(t\right)\right) = F\left(s-a\right) \tag{3.4}$$

Equation solving

Applying the properties to the given equation

$$y' = k_0 y \tag{3.5}$$

$$\mathcal{L}\left(y'\right) - \mathcal{L}\left(k_0 y\right) = 0 \tag{3.6}$$

$$s\mathcal{L}\left(y\right) - y\left(0\right) - k_0\mathcal{L}\left(y\right) = 0 \tag{3.7}$$

$$\mathcal{L}\left(y\right) = \frac{y\left(0\right)}{s - k_0} \tag{3.8}$$

$$y = y\left(0\right) e^{k_0 x} u\left(x\right) \tag{3.9}$$

Taking 1999 to be the initial year, we get $y\left(0\right) = 20000$ and $y\left(5\right) = 25000$

Substituting the initial conditions gives

$$y(0) = 20000 \tag{3.10}$$

$$y(5) = 25000 \tag{3.11}$$

$$20000e^{5k_0} = 25000 \tag{3.12}$$

$$e^{5k_0} = \frac{5}{4} \tag{3.13}$$

$$5k_0 = \ln\frac{5}{4} \tag{3.14}$$

$$k_0 = \frac{1}{5}\ln\frac{5}{4} \tag{3.15}$$

The theoritical solution is

$$f(x) = 20000e^{\frac{1}{5}\left(\ln\frac{5}{4}\right)x}u(x) \tag{3.16}$$

$$f(x) = 20000\left(\frac{5}{4}\right)^{\frac{x}{5}}u(x) \tag{3.17}$$

Substituting $x = 10$ gives 31250

Computational Solution

First we have to find the k value in the differential equation, for that

$$y(t+h) = y(t) + hy'(t) \tag{3.18}$$

$$y(t+h) = y(t) + hk_0 y(t) \tag{3.19}$$

$$y(t+2h) = y(t+h) + hk_0 y(t+h) \tag{3.20}$$

$$y(t+2h) = y(t) + hk_0 y(t) + hk_0 (y(t) + hk_0 y(t)) \tag{3.21}$$

$$y(t+2h) = (1+hk_0)^2 y(t) \tag{3.22}$$

Similarly

$$y(t+nh) = (1+hk_0)^n y(t) \tag{3.23}$$

Subtituting the initial condition and value of h gives

$$y(5) = (1 + 0.001k_0)^{5000} y(0) \tag{3.24}$$

$$25000 = 20000 (1 + 0.001k_0)^{5000} \tag{3.25}$$

$$\left(\frac{5}{4}\right)^{\frac{1}{5000}} - 1 = 0.001k_0 \tag{3.26}$$

$$k_0 = \frac{\left(\frac{5}{4}\right)^{\frac{1}{5000}} - 1}{0.001} \tag{3.27}$$

$$k_0 \approx 0.04462 \tag{3.28}$$

Consider the given linear differential equation

$$a_n y^n + a_{n-1} y^{n-1} + \cdots + a_1 y' + a_0 y + c = 0 \tag{3.29}$$

Then

$$y'(t) = \lim_{h \to 0} \frac{y(t+h) - y(t)}{h} \tag{3.30}$$

$$y(t+h) = y(t) + hy'(t) \tag{3.31}$$

Similarly

$$y^i(t+h) = y^i(t) + hy^{i+1}(t) \tag{3.32}$$

$$y^{n-1}(t+h) = y^{n-1}(t) + hy^n(t) \tag{3.33}$$

$$y^{n-1}(t+h) = y^{n-1}(t) + h\left(-\frac{a_{n-1}}{a_n} y^{n-1} - \frac{a_{n-2}}{a_n} y^{n-2} - \cdots - \frac{a_0}{a_n} y - \frac{c}{a_n}\right) \tag{3.34}$$

Where i ranges from 0 to $n-1$

$$\mathbf{y}(t+h) = \mathbf{y}(t) + \begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \\ -\frac{1}{a_n} & -\frac{a_0}{a_n} & -\frac{a_1}{a_n} & -\frac{a_2}{a_n} & \dots & -\frac{a_{n-2}}{a_n} & -\frac{a_{n-1}}{a_n} \end{pmatrix} (h\mathbf{y}(t)) \tag{3.35}$$

$$\mathbf{y}(t+h) = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & h & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & h & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & h \\ -\frac{h}{a_n} & -\frac{a_0 h}{a_n} & -\frac{a_1 h}{a_n} & -\frac{a_2 h}{a_n} & \dots & -\frac{a_{n-2} h}{a_n} & 1-\frac{a_{n-1} h}{a_n} \end{pmatrix} (\mathbf{y}(t)) \tag{3.36}$$

Discretizing the steps gives us

$$\mathbf{y}_{k+1} = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & h & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & h & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & h \\ -\frac{h}{a_n} & -\frac{a_0 h}{a_n} & -\frac{a_1 h}{a_n} & -\frac{a_2 h}{a_n} & \dots & -\frac{a_{n-2} h}{a_n} & 1 - \frac{a_{n-1} h}{a_n} \end{pmatrix} (\mathbf{y}_k) \tag{3.37}$$

where k ranges from 0 to number of data points with y_0^i being the given initial condition and vector $\mathbf{y}_0 = \begin{pmatrix} c \\ y(0) \\ y'(0) \\ \vdots \\ y^{n-1}(0) \end{pmatrix}$

For the given question

$$\mathbf{y}_{k+1} = \begin{pmatrix} 1 & 0 \\ -h & 1 + hk_0 \end{pmatrix} \mathbf{y}_k \tag{3.38}$$

Record the y_k for

$$x_k = lowerbound + kh \tag{3.39}$$

and then plot the graph. The result will be as given below.
The codes below verifies the obtained solution.

Plot of the function

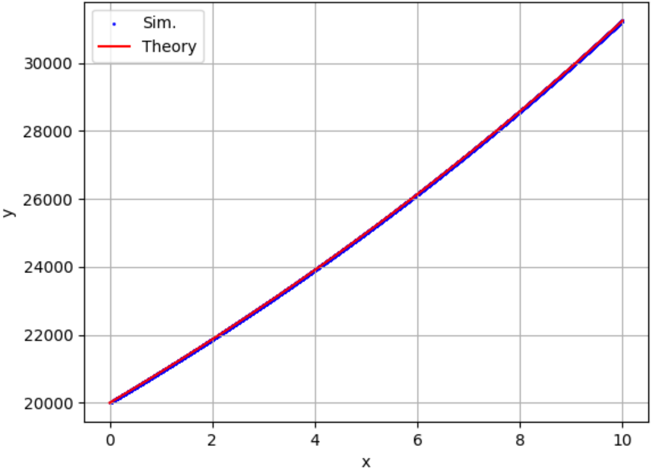

Figure: Function satisfying given differential equation

C Code

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "functions.h"
double** matrixgen(int order, double coefficients[order+2], double
    stepsize){
        double** outputmatrix = identity(order+1);
        for(int i=1; i<order; i++){
                outputmatrix[i][i+1] = stepsize;
        }
        outputmatrix[order][0] = -1/coefficients[0]*stepsize;
        for(int i=1; i<order+1; i++){
                outputmatrix[order][i] = (-coefficients[order+1-i]/
                    coefficients[0])*stepsize;
        }
        outputmatrix[order][order] += 1;
        return outputmatrix;}
```

```
double* recorddata(double lowerbound, double upperbound, int order,
    double coefficients[order+2], double initialconditions[order], double
    stepsize){
        double** vector_y = createMat(order+1,1);
        vector_y[0][0] = coefficients[order+1];
        for(int i=0;i<order;i++){
                vector_y[i+1][0] = initialconditions[i];
        }
        double** matrix = matrixgen(order, coefficients, stepsize);
        int no_datapoints = ((upperbound-lowerbound)/stepsize);
        double* yvalues = malloc(no_datapoints*sizeof(double));
        for(int i = 0; i<no_datapoints; i++){
                vector_y = Matmul(matrix,vector_y,order+1,order+1,1);
                yvalues[i] = vector_y[1][0];
        }
        return yvalues;
}
```

Python Code for Plotting

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

# Load the shared library
solver = ctypes.CDLL('./solver.so')

# Define the function signatures
solver.recorddata.restype = ctypes.POINTER(ctypes.c_double)
solver.recorddata.argtypes = [
    ctypes.c_double, # lowerbound
    ctypes.c_double, # upperbound
    ctypes.c_int, # order
    ctypes.POINTER(ctypes.c_double), # coefficients
    ctypes.POINTER(ctypes.c_double), # initialconditions
    ctypes.c_double # stepsize
]
```

```
# Define parameters
order =1
lowerbound = 0.0
upperbound = 10.0
stepsize = 0.001
y_0 = 20000
k = (((5/4)**(1/5000)) - 1)/0.001
coefficients = np.array([1.0, -k, 0.0], dtype=np.double)
initialconditions = np.array([y_0, k*y_0], dtype=np.double)

# Calculate the number of data points
no_datapoints = int((upperbound - lowerbound) / stepsize)

# Call the C function
results_ptr = solver.recorddata(
    ctypes.c_double(lowerbound),
    ctypes.c_double(upperbound),
    ctypes.c_int(order),
```

```
    ctypes.c_double(stepsize)
)
# Convert results back to a NumPy array
results = np.ctypeslib.as_array(results_ptr, shape=(no_datapoints,))
# Generate x-values for plotting
x_values = np.arange(lowerbound + stepsize, upperbound + stepsize,
    stepsize)
y_function = 20000*((5/4)**(x_values/5))
# Plot the data
plt.scatter(x_values, results, color='blue', s=1, label='Sim.')
plt.plot(x_values, y_function, color='red', label='Theory')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.savefig('../figs/fig.png')
plt.show()
```