Programming in Engineering (C++) (191158510)

Julius Mbaziira
(m7666913)

July 2, 2019

PROJECT DESCRIPTION

1 (ODE SOLVER - THE FOURTH ORDER RUNGE KUTTA NUMERICAL INTEGRATION METHOD)

Many physical systems in Engineering are modeled in form of differential equations. But most of them (especially, nonlinear differential equations) do not posses the analytical techniques to be solved. Hence, they are solved numerically. Here, an object-oriented numerical solver (**the fourth order Runge-Kutta method**) for a first-order ordinary differential equation is implemented. The following two problems are solved with stepsize (say, h) = 0.1: (**Example**: 14.2.3, *S. D. Rajan*, Object-Oriented Numerical Analysis (2018))

$$\frac{dy}{dx} = -2y, \quad 0 \le x \le 2.5 \quad y(x=0) = 2, \quad \text{and} \tag{1.1}$$

$$\frac{dy}{dx} = x^2 - 5x + 10, \quad 0 \le x \le 2 \quad y(x=0) = 1. \tag{1.2}$$

THE NUMERICAL SCHEME

The above ODE equations can be rewritten in the form;

$$\frac{dy}{dx} = f(x, y). \tag{1.3}$$

Hence, The fourth order Runge-Kutta numerical scheme/method takes the following form;

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{1.4}$$

where

$$k_1 = f\left(x_i, y_i\right) \tag{1.5}$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{hk_1}{2}\right) \tag{1.6}$$

$$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{hk_2}{2}\right) \tag{1.7}$$

$$k_4 = f\left(x_i + h, y_i + hk_3\right) \tag{1.8}$$

THE MAIN PROGRAM

The main program can be presented in two parts. The first part describes the variables associated with the two problems – the value of h (lines 24 and 33), the range of x values (lines 25 and 34) and the initial value of y (lines 25 and 34) for each problem. The problem to be solved is selected in line 17. In the second part, the rest of the required details are can be easily specified. The range of x values is divided into equal parts using the value of h (line 38) and the computed number of points is used to allocate the memory required to store the y and $dydx$ values at each point in line 39. The ODE solver object is defined in line 42 by invoking the overloaded constructor. The public member function **GoCompute** is called to obtain the solution (line 43), and the results are displayed in a tabular form (lines 45-57).

THE ODE SOLVER CLASS

In the "ODESolver" class, the "GoCompute" function reveals the where the 4^{th}-order Runge-Kutta method class is used in obtaining the solution (lines 32-35). The last argument in the Solve function – it is the name of the function where f_i is computed.

The member function "**UserFunctionEvaluation**" is where the function evaluation takes place –(x_i, y_i) come in as input and the y_i is updated with the computed f_i value.

THE RUNGE-KUTTA CLASS

The Runge-Kutta class has three public functions – the "constructor", "destructor" and the "Solve" function.

The **SOLVE** functionalities can be divided into three parts. In the first part, the function value at the initial point is computed (lines 22-27). The second part involves the loop where the solution at all the other points is computed. This involves computing the four k_i values given by Eqns. (1.5)-(1.8) – lines 32-53. For each k_i, the user function is called to obtain the $f_i(x_i, y_i)$ value.

In the final part, the recurrence formula, Eqn. (1.4), is used to compute the value at the next point (line 56), y_{i+1}, and preparations are made for computations at the next point.

THE VECTOR TEMPLATE AND THE ARRAYBASE CLASS

Given by *S. D. Rajan*, Object-Oriented Numerical Analysis (2018), are containers of the vector-matrix operations. Here, declarations of the functionalities of a "CVector" class (used in the code) are made to manipulate vectors.