International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.3, Issue No.3, pp : 250-252 1 March 2014

A Numerical Simulator for Solving Ordinary Differential Equations

B. K. Datta, N. Rahman & R. C. Bhowmik
Department of Mathematics, Pabna University of Science and Technology.
bimaldu@gmail.com, nl.nizhum@gmail.com
& rajcumath@yahoo.com

Abstract

Numerical differential equations studies the methods for finding numerical approximations to the solutions of differential equations, occur in many scientific disciplines, for instance in physics, chemistry, biology, and economics. Because of its various applications, this is often viewed as a discipline in and of itself. In this paper we develop a numerical simulator for solving ordinary differential equations (ODEs). This simulator is incorporated with a combination of Euler, modified Euler and Runge-Kutta second and fourth order methods.

1. Introduction

Very often the differential equations appearing in physical problems cannot be solved analytically. Thus it becomes imperative to discuss their solutions by numerical methods. In numerical method, we do not proceed in the hope of finding a relation between variables but we find the numerical values of the dependent variables for certain values of independent variables[i].

The process of finding a derivative is called differentiation. The modern development of differentiation is usually credited to Isaac Newton (1643–1727) and Gottfried Leibniz (1646–1716), who provided independent and unified approaches to differentiation and derivatives[ii]. In numerical analysis, numerical differentiation describes algorithms for estimating the derivative of a mathematical function or function subroutine using values of the function and perhaps other knowledge about the function. Isaac Newton (1642-1727), Joseph Raphson (1648-1715), Leonhard Euler (1707-1783), Joseph Lagrange (1736-1813), Carl David Tolme Runge (1856-1927), Martin Wilhelm Kutta (1867-1944), George Dantzig (1914-2005) and so on have developed the numerical differentiation.

Programming language C is very flexible and powerful. It was originally designed in the early 1970s. It allows us to maximum control with minimum command. It is recognized worldwide and used in a multitude of applications especially in Numerical solution of Differential Equations. Along with other numerous benefits, we have used programming language C in this paper.

The outline of this paper is as follows: Section 2 contains the description of the existing methods of numerical differential equations with methodology. In Section 3, we develop a numerical simulator, using the programming language C, which gives us the solution of an ordinary differential equation simultaneously regarding four popular existing methods namely Euler, modified Euler, Runge-Kutta second and fourth order. Moreover, the technique identifies the method that gives the best solution comparing with possible exact solution of the problem. We devote Section 4 to a trial of the simulator for a specific problem. Conclusions are given at the end at Section 5.

2. Existing methods

We give a brief description of the existing methods of numerical differential equations like Euler, modified Euler, Runge-Kutta second and fourth order in this section with their methodology.

2.1 Euler method

In mathematics and computational science, the Euler method is a first-order numerical procedure for solving ODEs with a given initial value. It is the most basic explicit method for numerical ODEs.

We consider the differential equation [iii]

$$y' = f(x, y) \tag{1}$$

with the initial condition $y(x_0) = y_0$ (2)

Suppose that we wish to solve the equation (1) with (2) for the value of y at $x_r = x_0 + rh$ $(r = 1, 2, 3, \ldots)$.

Integrating (1) with respect to x from y_0 to y_1 and x_0 to x_1, we get

$$y = y_0 + \int_{x_0}^{x_1} f(x, y)\, dx \tag{3}$$

Assuming that $f(x, y) = f(x_0, y_0)$ in $x_0 \le x \le x_1$, this gives Euler's formula

$$y_1 \approx y_0 + h\, f(x_0, y_0) \text{ [for } x - x_0 = h \text{]} \tag{4}$$

Similarly for the range $x_1 \le x \le x_2$, we have

$$y_1 = y_1 + \int_{x_1}^{x_2} f(x, y)\, dx$$

International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.3, Issue No.3, pp : 250-252 1 March 2014

Substituting $f(x_1, y_1)$ for $f(x, y)$ where $x_1 \le x \le x_2$, we have $y_2 \approx y_1 + h\, f(x_1, y_1)$ [since $x_2 - x_1 = h$]. And proceeding in this way, we obtain the general formula

$$y_{n+1} \approx y_n + h\, f(x_n, y_n),\ n = 0,1,2,\ldots$$

2.2 Modified Euler's method

Integrating (3) by means of trapezoidal rule to obtain

$$y_1 = y_0 + (h/2)[f(x_0, y_0) + f(x_1, y_1)] \quad (5)$$

We thus obtain the iterative formula

$$y_1^{(n+1)} = y_0 + (h/2)[f(x_0, y_0) + f(x_1, y_1^{(n)})];$$
$$n = 0,1,2,\ldots \quad (6)$$

Where $y_1^{(n)}$ is the nth approximation to y_1. The iterative formula (6) can be started by choosing y_1^0 from Euler's formula $y_1^0 = y_0 + h\, f(x_0, y_0)$

2.3 Runge-Kutta method (second order)

Substitute $y_1 = y_0 + h\, f(x_0, y_0)$ on the right side of equation (5), we obtain

$$y_1 = y_0 + (h/2)[f_0 + f(x_0 + h, y_0 + h\, f_0)] \quad (7)$$

Where $f_0 = f(x_0, y_0)$ and $x_1 = x_0 + h$.

Now set $k_1 = h\, f_0$ and $k_2 = h\, f(x_0 + h, y_0 + k_1)$ so that equation (7) becomes $y_1 = (1/2)[k_1 + k_2]$.

This is known as Runge-Kutta second order formula.

2.4 Runge-Kutta method (fourth order)[iv]

We mention the fourth order formulae defined by

$$y_1 = y_0 + W_1 k_1 + W_2 k_2 + W_3 k_3 + W_4 k_4 \quad (8)$$

Where

$$k_1 = h\, f(x_0, y_0)$$
$$k_2 = h\, f(x_0 + \alpha_0 h, y_0 + \beta_0 k_1)$$
$$k_3 = h\, f(x_0 + \alpha_1 h, y_0 + \beta_1 k_1 + \nu_1 k_1)$$
$$k_4 = h\, f(x_0 + \alpha_2 h, y_0 + \beta_2 k_1 + \nu_2 k_2 + \delta_1 k_3) \quad (9)$$

Where the parameters have to be determined by expanding both sides of (8) by Taylor's series and securing agreement of terms up to and including those containing h^4. The choice of the parameters is, again arbitrary and we have therefore several fourth order Runge-kutta formulae. If for example we set $\alpha_0 = \beta_0 = \alpha_1 = 1/2$, $\alpha_2 = 1$, $\beta_1 = (\sqrt{2} - 1)/2$, $\beta_2 = 0$, $\nu_1 = (\sqrt{2} - 1)/2$, $\nu_2 = -1/\sqrt{2}$, $\delta_1 = (\sqrt{2} + 1)/2$, $W_1 = W_4 = 1/6$, $W_2 = (\sqrt{2} - 1)/3\sqrt{2}$ and $W_3 = (\sqrt{2} + 1)/3\sqrt{2}$, We obtain the method of Gill, whereas the choice $\alpha_0 = \alpha_1 = 1/2$, $\beta_0 = \nu_1 = 1/2$, $\beta_1 = \beta_2 = \nu_2 = 0$, $\alpha_2 = \nu_1 = 1$, $W_1 = W_4 = 1/6$ and $W_2 = W_3 = 2/6$ leads to the fourth order Runge-Kutta formulae, whereas

$$k_1 = h\, f(x_0, y_0)$$
$$k_2 = h\, f(x_0 + h/2, y_0 + k_1/2)$$
$$k_3 = h\, f(x_0 + h/2, y_0 + k_2/2)$$
$$k_4 = h\, f(x_0 + h, y_0 + k_3)$$

Consequently, $y_1 = y_0 + (1/6)(k_1 + 2k_2 + k_3 + k_4)$

3. Algorithm[vi]

We here give the algorithm that can solve differential equation numerically regarding four methods- Euler, modified Euler, Runge-Kutta second and fourth order simultaneously. Moreover, it will ensure which method gives the best solution with comparing exact solution.

INPUT: function $f(x, y)$, initial condition (x_0, y_0), interval h, value of x, direct result r.

Step-1: Set $n = (x - x_0)/h$, $y_{00} = y_0$, $y_{oe} = y_0$, $y_{02} = 0$, $y_{04} = 0$, $y_1 = y_{00} + h\, f(x_0, y_{00})$

Step-2: Set $i = 1$

Step-3: While $i \le n$ repeat Step-4 to step-7

Step-4: Set $j = 1$

Step-5: While $j \le i$ repeat step-10

Step-6: Set $x_1 = x_0 + h$,

$y_{10} = y_{00} + (1/2)\, h\, f(x_0, y_0) + f(x_1, y_1)$, $y_1 = y_{10}$

Step-7: Set $y_{1e} = y_{0e} + h\, f(x_0, y_{0e})$,

$k_{11} = h\, f(x_0, y_{04})$,

$k_{22} = h\, f(x_0 + h/2, y_{04} + k_{11}/2)$,

$k_{33} = h\, f(x_0 + h/2, y_{04} + k_{22}/2)$,

$k_{44} = h\, f(x_0 + h, y_{04} + k_{33})$,

$y_1 r_4 = y_{04} + (k_{11} + 2k_{22} + 2k_{33} + k_{44})/6$,

$k_1 = h\, f(x_0, y_{02})$, $k_2 = h\, f(x_0 + h, y_{02} + k_1)$,

$y_1 r_2 = y_{02} + (k_1 + k_2)/2$, $y_{0e} = y_{1e}$, $y_{00} = y_{10}$, $y_{04} = y_1 r_4$, $y_{02} = y_1 r_2$, $x_0 = x_n$.

Step-8: Set

$a = |y_{1e} - r|$, $b = |y_{10} - r|$, $c = |y_1 r_2 - r|$,

$d = |y_1 r_4 - r|$

If ($a < b$ and $a < c$)
If ($a < d$)
y_{1e}
Else
$y_1 r_4$
If ($b < a$ and $b < c$)
If ($b < d$)
y_{10}
Else
$y_1 r_4$
If ($c < a$ and $c < b$)
If ($c < d$)
$y_1 r_2$
Else
$y_1 r_4$

Output: $y_{1e}, y_{10}, y_1 r_2$ and $y_1 r_4$ with message which method gives best solution.

4 Findings

We are now offering a trial of the simulator for the numerical differential equation $\frac{dy}{dx} = -y$, with condition $y(0) = 1$ to find $y(0.04)$.

4.1 Input

This is a program of differential equation.
Enter Equation:

$$-y = \frac{dy}{dx}$$

Enter value of h:
0.01
Enter value of x_0:
0
Enter value of y_0:
1
Enter target:
0.04
Type the exact value
0.9607894

4.2 Output

The required interpolated value of $y[0.0400]$ in Euler method: 0

The required interpolated value of $y[0.0400]$ in modified Euler method: 1

The required interpolated value of $y[0.0400]$ in Runge-Kutta method: 0.960790

The required interpolated value of $y[0.0400]$ in Runge-Kutta 4th order method: 0.960789

The Runge-Kutta 4th order method is the best.

5. Conclusion[v]

In this paper, we develop a simulator incorporated with the traditional Euler, modified Euler, Runge-Kutta second and fourth order methods for solving ordinary differential equations. It been observed in the paper that the result obtained according to our procedure is much nearer to the exact solution. We therefore, hope that this algorithm can solve differential equation problems numerically and can save our time and labour. Finally, we have seen that Runge-Kutta fourth order gives the best solution among these.

References

i. *A. Kaw, E.E. Kalu, Numerical Methods with Applications, Lalu.com, 2008.*

ii. *C. Edwards, Jr. The Historical Development of the Calculus, Springer-Verlag, 1997.*

iii. *M. Goyal, Computer-based Numerical & Statistical Techniques, Infinity Science Press LLC, New Delhi, India, 2007.*

iv. *S. S. Sastry, Introductory Methods of Numerical Analysis, Prentice-Hall India, 2005.*

v. *Bimal kumar Datta & M .Babul Hasan, 'A code for solving Non-Linear Programming Problems' , Dhaka Univ.J.Sci.59(1);25-31,2011(january).*

vi. *Bimal kumar Datta & M .Babul Hasan, 'A c*omputer oriented Lagrange Method for solving Non-Linear Programming Problems', *Dhaka Univ.J.Sci.59(1);*71-75,*2011(january).*