European Journal of Mathematics and Statistics
Vol 5 | Issue 5 | October 2024
ISSN 2736-5484

RESEARCH ARTICLE

On the Numerical Solution of Linear and Non-Linear Dirichlet Boundary Value Problems of Ordinary Differential Equations: The Shooting Method

Odongo Benard Audi[1,*], Opiyo Richard Otieno[2], Owego Dancun Okeso[3], and Onyango Thomas Mboya[1]

ABSTRACT

In this article, a numerical technique called shooting which entails the solution of initial value problems with the single-step fourth-order Runge-Kutta method together with the iterative root finding secant method is formulated for use on both linear and non-linear boundary value problems of ordinary differential equations with Dirichlet boundary conditions. Two examples are illustrated. One, the solution of the linear case with its analytic counterpart is compared, and two, the non-linear case. Graphical outputs of the solutions from two MATHEMATICA codes are presented.

Keywords: BVPs, ODEs, Runge-Kutta method, Shooting technique.

Submitted: September 08, 2022

Published: October 16, 2024

10.24018/ejmath.2024.5.5.96

[1] *Department of Industrial and Engineering Mathematics, Technical University of Kenya, Kenya.*
[2] *Department of Pure and Applied Mathematics, Maseno University, Kenya.*
[3] *Department of Pure and Applied Mathematics, Technical University of Kenya, Kenya.*

* ***Corresponding Author:*** e-mail: benardodongo84@yahoo.com

1. Introduction

1.1. *Differential Equations*

Differential equations express the relationship between unknown function or functions and their derivatives. Usually, the unknown functions represent physical quantities (heat transfer in a medium, flow of a current in the electric circuit, the evolution of price options in mathematical finance, etc.,) and their derivatives represent their rates of change. This is one reason why we endeavor to study differential equations, they are the tools used in the world of mathematical modeling. The study of differential equations concentrates mainly on their solutions and the properties of these solutions. When a solution of a differential equation cannot be found by explicit formulae, the solution may be approximated by numerical methods.

1.2. *Classification*

The classification of differential equations is important because it helps to choose the type of solution method required for a particular differential equation. Classification is based on some of the many concepts, namely; ordinary or partial differential equations, linear or non-linear equations, homogeneous or heterogeneous equations, order of differential equations, and others.

A differential equation whose unknown function depends only on one independent variable is called an ordinary differential equation. A differential equation whose unknown function depends on more than one independent variable is called a partial differential equation.

In this article, our attention will be focused on ordinary differential equations.

2. Ordinary Differential Equations

Ordinary Differential Equation is in most cases abbreviated as ODE. The unknown function is called the dependent variable. If y is the dependent variable and x is the independent variable, then the relation

$$F(x, y, y', y'', \ldots, y^{(n-1)}, y^{(n)}) = 0 \tag{1}$$

is an ODE. Equation (1) is also called an nth order ODE.

2.1. Classification of ODEs

1) Order: The order of the differential equation is the highest derivative that appears in (1).

2) Linear or nonlinear ODEs: An nth order differential equation is said to be linear if it can be written in the form

$$a_n(x)\frac{d^n y}{dx^n} + a_{n-1}(x)\frac{d^{n-1} y}{dx^{n-1}} + \ldots + a_1(x)\frac{dy}{dx} + a_0(x)y = g(x)$$

that must satisfy two conditions:

- the dependent variable y and all its derivatives in the equation are of power one.
- all the coefficients $a_n(x)$, $a_{n-1}(x), \ldots, a_0(x)$ and the function $g(x)$ are either constants or depend only on the independent variable x.

If any of these two conditions is not satisfied then the ODE is nonlinear. Usually, explicit solutions of nonlinear ODEs are difficult to find.

3) Homogeneous and Nonhomogeneous ODEs: As in above, if the function $g(x) = 0$, the nth order ODE is said to be homogeneous, and if the function $g(x) \neq 0$ then the ODE is said to be nonhomogeneous or heterogeneous.

4) Initial and Boundary Conditions: The solution to an ODE contains arbitrary constants (constants of integration) as the process solution involves integration. A boundary condition expresses the behaviuor of a function on the boundary or border of its area of definition. An initial condition is like a boundary condition for the time direction. In most physical problems, boundary conditions describe how the system behaves on its boundaries and initial conditions that specify the state of the system for an initial time. An initial value problem of an ODE abbreviated IVP is an ODE whose solution is specified at only one given point in the domain of the equation. This condition is often called an initial condition. An ODE whose solution in an interval domain say $[a, b]$ where $a, b \in \mathbb{R}$ is specified at more than one point is called a Boundary Value Problem (BVP). The conditions in this case are called boundary conditions [1]. Boundary conditions in which only the solution is specified at boundary points are called Dirichlet boundary conditions. If the derivative of the solution is only specified at the boundary points, the boundary conditions are called Neumann boundary conditions. Robin boundary conditions are a mixture of both.

3. Solution Methods for ODEs

Explicit formulae solutions of ordinary differential equations are limited. At best there are only a few ordinary differential equations that can be solved analytically. There are many different methods in the literature today for analytical solutions of both IVPs and BVPs [2]. Examples of these methods include methods for first order ODEs such as linear equations solved by integrating factors, homogeneous equations, exact equations, equations in which variables are separable and Bernoulli type equations. For second and higher order ODEs, techniques such as variation of parameter method, use of complementary function and particular integral, Laplace transform are well developed. There are only few ODEs for which the solution is in terms of mathematical functions such as sine, cosine, logarithm, exponential, etc. Some simple second order linear differential equations can be solved using some special functions such as Bessel and Legendre. Beyond second order, the kinds of special functions needed to solve even simple linear differential equations become extremely complicated. Solutions to most practical problems involving ordinary differential equations require the use of numerical methods. Numerical solutions of IVPs of ODEs can be classified into two groups. One-step methods and multi-step methods. The one-step methods include Taylor's methods, Heun's methods, Euler's methods and Runge-Kutta methods. Linear multi-step methods include Implicit Euler's method, Adams-Bashforth method, Adams-Moulton method and Predictor-Corrector methods [3]. For BVP of ODEs there are methods such as the shooting method and finite difference methods that are suitable for both linear and nolinear BVPs. The shooting method is the subject of this article.

4. A Solution of BVPs of ODEs

4.1. Basics

Solution of the nonlinear equation $f(x) = 0$. A value or values of x which when substituted in $f(x)$ makes $f(x) = 0$ are called the roots of the equation $f(x) = 0$. Example, consider $f(x) = x^2 - 5x + 6 = 0$. We can factorize $f(x)$ and have $(x - 2)(x - 3) = 0$ which implies that $x = 2$ or $x = 3$. 2 and 3 are the roots of $f(x) = x^2 - 5x + 6 = 0$. In most cases, you may not factorize easily $f(x) = 0$ to find the roots. For example, suppose $f(x) = x^3 - 4x - 8 = 0$. Finding the roots of $f(x) = 0$ in this case requires other techniques other than factorization. It is for this reason that we discuss some of the numerical techniques that we may use to find the roots of $f(x) = 0$. But before we begin, let us learn the following concept. Consider an interval (a, b) on the real line $\mathbb{R}$ and a continuous function $f(x)$ on this interval. If $f(a)f(b) < 0$ (i.e., the function has opposite signs on the interval), then there exists (you can find) a number $c \in (a, b)$ such that $f(c) = 0$ (i.e., c is a root of $f(x) = 0$). Hence, this condition $f(a)f(b) < 0$ guarantees the existence of a root of $f(x) = 0$ on the interval (a, b). There are several numerical techniques that we can use to find the root of $f(x) = 0$. Some of them are; the bisection method, regula falsi method, Newton-Raphson method, secant method, etc. In this article, we shall learn how the secant method can be used to find a root of $f(x) = 0$. The secant method is:

$$x_{i+1} = \frac{x_{i-1}f(x_i) - x_i f(x_{i-1})}{f(x_i) - f(x_{i-1})}, \quad i = 1, 2, 3, \ldots$$

Graphical derivation of this method is left to the reader. Note that this method requires two initial guess values x_0 and x_1 for the root. As an example consider the nonlinear equation $f(x) = x^3 - 2x - 5 = 0$. We find a root of $f(x) = 0$ by the secant method. Note that $f(2) = 2^3 - 2(2) - 5 = -1 < 0$ and $f(3) = 3^3 - 2(3) - 5 = 16 > 0$. Thus, the root $x \in (2, 3)$. Let the initial approximation be $x_0 = 2$ and $x_1 = 3$. Then:

$$x_2 = \frac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)} = 2.05882$$

$$x_3 = \frac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)} = 2.08126$$

$$x_4 = \frac{x_2 f(x_3) - x_3 f(x_2)}{f(x_3) - f(x_2)} = 2.09483$$

$$x_5 = \frac{x_3 f(x_4) - x_4 f(x_3)}{f(x_4) - f(x_3)} = 2.09452,$$

$$x_6 = \frac{x_4 f(x_5) - x_5 f(x_4)}{f(x_5) - f(x_4)} = 2.094584$$

Hence the root is $x = 2.095$ to 3 decimal places. The question that may arise in this problem is when do you possibly stop the iteration and declare that the value of x you have computed is the root of the equation? One way out of this is that you work out the absolute difference between two consecutive iterates say $|x_{i+1} - x_i|$. If this value is less than a given tolerance you have set, then you stop the iteration and output your root. Otherwise, you continue the iteration till the tolerance is achieved.

4.2. The Shooting Method

Let us now relate the above information to the shooting method. This is a numerical method that can be used to find solutions of BVPs of ordinary differential equations. Given a BVP of the ordinary differential equation, first reduce it to a system of first order equations. The key task here is to solve the above equations as IVPs. One or more of the resulting IVPs (depending on the order of the BVP) will not have an initial value at the first boundary point. Let the value of this IVP at the first boundary point be say s_i [4]. If you can guess this value correctly then the solution you obtain for the IVP is the solution of the BVP. Let $y = y(x)$ be the solution of the IVP. Since the solution depends on the guessed value s_i, let $y(x, s_i)$ be the solution of the IVP. We generate a sequence $\{s_i\}$ such that:

$$\lim_{i \to \infty} y(x, s_i) = y(x)$$

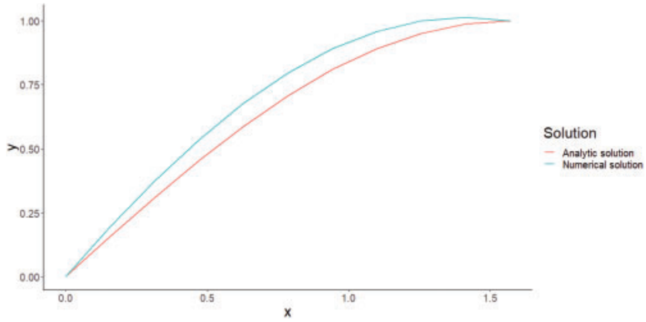

Fig. 1. Solution for case 1.

The solution of the IVP at the second boundary point say b depends on both b and s_i. If β is the value of the solution of the BVP at the second boundary point, then $(y(b, s_i) - \beta)$ is the error made. Suppose s is a value that makes $(y(b, s) - \beta)$ zero i.e., $y(b, s) - \beta = 0$. Clearly, this is a root finding problem. We can now invoke the use of secant method. The secant method approximation of $y(b, s) - \beta = 0$ will be:

$$s_{i+1} = s_i + \frac{[s_{i-1} - s_i]}{y(b, s_{i-1}) - y(b, s_i)}[\beta - y(b, s_i)]$$

5. Illustration Examples

In this section, we shall have two examples. The first one for a simple linear BVP whose analytic solution is readily obtained and the second example is a non-linear BVP whose solution is not readily obtained by analytical means. Each example is accompanied by a MATHEMATICA code for the solution.

1) Case 1: Given the linear BVP

$$\frac{d^2y}{dx^2} + y = 0, \quad \text{on} \quad 0 \leq x \leq 1 \quad \text{with} \quad y(0) = 0, \quad y(1) = 1$$

we determine the solution as stated above. By letting $dy/dx = z$ we can reduce the BVP to a system of two IVPs as below:

$$\frac{dy}{dx} = z, \quad y(0) = 0$$

$$\frac{dz}{dx} = -y, \quad z(0) = ?$$

If we choose (guess) a value for the solution z at 0, we can solve the two IVPs numerically by single-step 4th order Runge-Kutta method. The shooting method requires that we make two intelligent guesses of the initial values of the solution z and use an iterative procedure for root finding, in this case the secant method to obtain the appropriate value which we compare with the second boundary point of the BVP in this case $\beta = 1$. This is done in the MATHEMATICA code as shown in Appendix below.

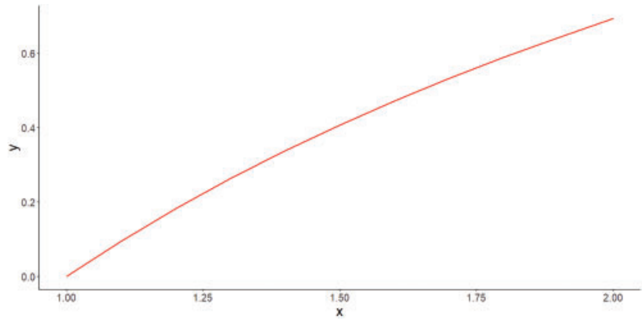

Fig. 2. Solution for case 2.

The analytic solution of the BVP can be shown to be $y = \sin(x)/\sin(1)$. The code also compares the two solutions (Fig. 1). For verification, note that $y(1) = y4[n] = 1.00$ agrees well with the program code.

2) Case 2: Consider the non-linear BVP:

$$\frac{d^2y}{dx^2} + \left(\frac{dy}{dx}\right)^2 + y = log_e x, \quad 1 \leq x \leq 2, \quad y(1) = 0, \quad y(2) = \log_e 2$$

The solution is shown in Fig. 2. The MATHEMATICA code is as in Appendix below. Also, note that $y(2) = N[Log[E, 2] = y4[n] = 0.693147$. Accurate to six decimal places.

6. Conclusion

The solution of the linear BVP using the shooting technique matches very well with its analytic solution. This motivated us to find the solution of the non-linear BVP. Solution techniques are numerous in the literature. The shooting method formulated in this article is one method that can be used to solve linear as well as non-linear boundary value problems of ordinary differential equations with Dirichlet boundary conditions.

Conflict of Interest

The authors state that there is no conflict of interest.

References

[1] Manyonge AW, Opiyo R, Kweyu D. Numerical solution of non-linear boundary-value problems of ordinary differential equations by the shooting technique. *J Inov Technol Educ*. 2017;4(1):29–36.
[2] Jain MK. *Numerical Solution of Differential Equations*. 2nd ed. New Delhi: Wiley Eastern Limited; 1984.
[3] Butcher JC. *Numerical Methods for Ordinary Differential Equations*. 2nd ed. England: John Wiley and Sons Ltd; 2008.
[4] Odongo B, Opiyo R, Manyonge A. Effects of inclination angle and free convection on velocity profile for a steady 2-dimensional magnetohydrodynamic fluid flow in an inclinedcylindrical pipe. *EJERS*. 2021;6(7):114.

Appendix: MATHEMATICA Code

```
tol = .0001;
f[x_, y_, z_] := z;
g[x_, y_, z_] := -y;
x0 = 0; xf = 1; y0 = 0; yf = 1; n = 10; za[1] = 1; z1[0] = za[1]; y1[0] = y0; x1[0] = x0;
step = (xf - x0) / n; h = N[step];
For[i = 0, i < n, i++,
  x1[i + 1] = x1[i] + h;
  k1y1 = f[x1[i], y1[i], z1[i]];
  k1z1 = g[x1[i], y1[i], z1[i]];
  k2y1 = f[x1[i] + 0.5 * h, y1[i] + k1y1 * h * 0.5, z1[i] + k1z1 * h * 0.5];
  k2z1 = g[x1[i] + 0.5 * h, y1[i] + k1y1 * h * 0.5, z1[i] + k1z1 * h * 0.5];
  k3y1 = f[x1[i] + 0.5 * h, y1[i] + k2y1 * h * 0.5, z1[i] + k2z1 * h * 0.5];
  k3z1 = g[x1[i] + 0.5 * h, y1[i] + k2z1 * h * 0.5, z1[i] + k2z1 * h * 0.5];
  k4y1 = f[x1[i] + h, y1[i] + k3y1, z1[i] + k3z1];
  k4z1 = g[x1[i] + h, y1[i] + k3y1, z1[i] + k3z1];
  y1[i + 1] = y1[i] + (k1y1 + 2 * k2y1 + 2 * k3y1 + k4y1) * h / 6;
  z1[i + 1] = z1[i] + (k1z1 + 2 * k2z1 + 2 * k3z1 + k4z1) * h / 6;];
Table[{x1[i], y1[i]}, {i, 0, n}];
plot1 = ListPlot[Table[{x1[i], y1[i]}, {i, 0, n}], Frame -> True, Joined → True];
za[2] = 4; z2[0] = za[2]; y2[0] = y0;
For[i = 0, i < n, i++,
  x1[i + 1] = x1[i] + h;
  k1y2 = f[x1[i], y2[i], z2[i]];
```

```
      k1z2 = g[x1[i], y2[i], z2[i]];
      k2y2 = f[x1[i] + 0.5 * h, y2[i] + k1y2 * h * 0.5, z2[i] + k1z2 * h * 0.5];
      k2z2 = g[x1[i] + 0.5 * h, y2[i] + k1y2 * h * 0.5, z2[i] + k1z2 * h * 0.5];
      k3y2 = f[x1[i] + 0.5 * h, y2[i] + k2y2 * h * 0.5, z2[i] + k2z2 * h * 0.5];
      k3z2 = g[x1[i] + 0.5 * h, y2[i] + k2y2 * h * 0.5, z2[i] + k2z2 * h * 0.5];
      k4y2 = f[x1[i] + h, y2[i] + k3y2, z2[i] + k3z2];
      k4z2 = g[x1[i] + h, y2[i] + k3y2, z2[i] + k3z2];
      y2[i + 1] = y2[i] + (k1y2 + 2 * k2y2 + 2 * k3y2 + k4y2) * h / 6;
      z2[i + 1] = z2[i] + (k1z2 + 2 * k2z2 + 2 * k3z2 + k4z2) * h / 6;];
   Table[{x1[i], y2[i]}, {i, 0, n}];
   plot2 = ListPlot[Table[{x1[i], y2[i]}, {i, 0, n}], Frame -> True, Joined → True];
   za[3] = za[2] + ((za[1] - za[2]) / (y1[n] - y2[n])) * (yf - y2[n]);
   z3[0] = za[3];
   y3[0] = y0;
   For[i = 0, i < n, i++,
      x1[i + 1] = x1[i] + h;
      k1y3 = f[x1[i], y3[i], z3[i]];
      k1z3 = g[x1[i], y3[i], z3[i]];
      k2y3 = f[x1[i] + 0.5 * h, y3[i] + k1y3 * h * 0.5, z3[i] + k1z3 * h * 0.5];
      k2z3 = g[x1[i] + 0.5 * h, y3[i] + k1y3 * h * 0.5, z3[i] + k1z3 * h * 0.5];
      k3y3 = f[x1[i] + 0.5 * h, y3[i] + k2y3 * h * 0.5, z3[i] + k2z3 * h * 0.5];
      k3z3 = g[x1[i] + 0.5 * h, y3[i] + k2y3 * h * 0.5, z3[i] + k2z3 * h * 0.5];
      k4y3 = f[x1[i] + h, y3[i] + k3y3, z3[i] + k3z3];
    k4z3 = g[x1[i] + h, y3[i] + k3y3, z3[i] + k3z3];
    y3[i + 1] = y3[i] + (k1y3 + 2 * k2y3 + 2 * k3y3 + k4y3) * h / 6;
    z3[i + 1] = z3[i] + (k1z3 + 2 * k2z3 + 2 * k3z3 + k4z3) * h / 6;];
  Table[{x1[i], y3[i]}, {i, 0, n}];
  plot3 = ListPlot[Table[{x1[i], y3[i]}, {i, 0, n}], Frame -> True, Joined → True];
  j = 3;
  Abs[yf - y3[n]];
  While[Abs[yf - y3[n]] > tol,
    za[j + 1] = za[j] + ((za[j - 1] - za[j]) / (y2[n] - y3[n])) * (yf - y3[n]);
    y4[0] = y0;
    z4[0] = za[j + 1];
    For[i = 0, i < n, i++,
     x1[i + 1] = x1[i] + h;
     k1y4 = f[x1[i], y4[i], z4[i]];
     k1z4 = g[x1[i], y4[i], z4[i]];
     k2y4 = f[x1[i] + 0.5 * h, y4[i] + k1y4 * h * 0.5, z4[i] + k1z4 * h * 0.5];
     k2z4 = g[x1[i] + 0.5 * h, y4[i] + k1y4 * h * 0.5, z4[i] + k1z4 * h * 0.5];
     k3y4 = f[x1[i] + 0.5 * h, y4[i] + k2y4 * h * 0.5, z4[i] + k2z4 * h * 0.5];
     k3z4 = g[x1[i] + 0.5 * h, y4[i] + k2y4 * h * 0.5, z4[i] + k2z4 * h * 0.5];
     k4y4 = f[x1[i] + h, y4[i] + k3y4, z4[i] + k3z4];
     k4z4 = g[x1[i] + h, y4[i] + k3y4, z4[i] + k3z4];
     y4[i + 1] = y4[i] + (k1y4 + 2 * k2y4 + 2 * k3y4 + k4y4) * h / 6;
     z4[i + 1] = z4[i] + (k1z4 + 2 * k2z4 + 2 * k3z4 + k4z4) * h / 6];
    y2[n] = y3[n];
    y3[n] = y4[n];
    j = j + 1];
  Abs[yf - y3[n]];
  j;
  Table[{x1[i], y4[i]}, {i, 0, n}];
  plot4 = ListPlot[Table[{x1[i], y4[i]}, {i, 0, n}], Frame -> True, Joined → True];
  plot5 = Plot[y[x] = Sin[x] / Sin[1], {x, 0, 1}, PlotStyle → RGBColor[0, 1, 0]];
  Show[plot4, plot5, PlotRange -> Automatic, Frame -> True, Joined → True]
```