IOSR Journal of Mathematics (IOSR-JM)
e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 14, Issue 1 Ver. I (Jan. - Feb. 2018), PP 49-61
www.iosrjournals.org

Comparative Analysis of Linear two point Boundary Value Problems via Shooting and Finite Difference Methods

S. Markus[1], I. O. Ibrahim[2] And B.M. Abba[3]
[1,2,3] *Department of Mathematical Sciences, University of Maiduguri, Borno State, Nigeria.*
Corresponding Author: S. Markus

Abstract: *This paper investigates a reduction from boundary value problems to initial value problems. Shooting and Finite difference methods were used in the numerical solutions of two-point boundary value problems. Problems with and without exact solutions were considered. It showed the rate, efficiency and efficacy of convergence for two point Boundary Value Problems. The difficulty in providing exact solutions to two point boundary value problems via analytical methods necessitated the study. It was observed that the shooting method provides a better result than the finite difference method in solving two point boundary value problems (ODEs) with Dirichlet boundary conditions. The study further revealed other integrating factors adopted.*

Keywords: *BVP, Shooting method, Finite difference method, MATLAB, MAPLE, Euler's method, Trapezoidal method, Classical RK4,Embedded RK23.*

Date of Submission: 05-01-2018 Date of acceptance: 22-01-2018

I. Introduction

Real life situation in engineering can be modelled into differential equations with initial or boundary value conditions. While it is easy to provide a solution to well-posed Initial Value Problems (IVPs), not all Boundary Value Problems (BVPs) can be solved analytically. This gave rise tonumerical solution which includes the shooting and finite difference methods. These methods have gained popular, due to their applications in resolving problems in science and engineering. Discussions in this context will be centered on two point boundary value problems (TPBVPs).

1.1 Two Point Boundary Value Problems

The Boundary value Problems (BVPs) is an Ordinary Differential Equation whose values or derivative are known usually at the end points (or boundary of some domain of interest). Let f be a continuously differentiable function. The Two Point Boundary Value Problem (TPBVP) with dirichlet boundary condition is defined by

$$x'' = f(t, x, x'), \qquad x(a) = A, \qquad x(b) = B$$

Gear [11] discussed the direct numerical solution of the mth order differential system

$$x^m = f\left(t, x, x', \dots, x^{(m-1)}\right) \tag{1.1}$$

With the tacit assumption that f is continuous in the independent variable t and satisfies a Lipschitz condition of order one with respect to *x* and each of

$$x^{(i)} = \frac{d^i x}{dt^i} \qquad i = 1\,(1)m - 1 \tag{1.2}$$

Rutishauser[21] examined the direct solution of 1.2 and its equivalent first order IVP defined.

$$x' = f(t, x), \quad x(a) = x_0$$

Where $x = (x_1, x_2, \dots, x_m)^T x_0 = (\eta_1, \eta_2, \dots, \eta_m)^T$ (1.3)

and Rutishauser concluded that the choice of approach depends on the particular problem at hand.

If the conditions in (1.3) are given at more than one point, we have a boundary value problem, which is well treated in [15], [17-19]discussed the theory of the direct finite difference method for second order Initial Value Problem.

$$x'' = f\left(t, x, x'\right), \quad x(a), x'(a) \ given \tag{1.4}$$

The application of the conventional numerical methods demand for the reduction of (1.4) to a set of first order IVP

$$x' = z, \qquad x(a) = x_0$$

$$z' = f(t, x, z), \qquad z(a) = z_0 \tag{1.5}$$

For the numerical solution of (1.4) by finite differences, we can either reduce (1.4) to a system of two first order equations and then apply standard methods for the first order systems. Although Kayode [16] analyzed that the reduction of higher order differential equation to systems of first order equations leads to serious computational burden as wastage in computer time. While this is true, methods which solve higher differential equations directly without reduction to systems of first order equation most times gives a larger error than those methods that include reduction to first order equation. These methods are not without their own limitation.

[11],[13-15]developed explicit Runge-Kutta Nystrom methods for the numerical solution of (1.4).

Dormand [6] proposed two classes, RKM2 and RKM4 of embedded Runge-Kutta formulas.

$$x'' = f(t, x) \qquad x(a),\ x'(a)\ \ given \tag{1.6}$$

The efficiency of the various type of Runge-Kutta formulas has greatly improved since the idea of embedding was introduced to obtain local truncation errors ([2],[3] and [10]).

Shampine [22] in the code STEP, made provision to deal the discontinuities. Enright [9] gave an overview of numerical methods IPBVPs and it identified strategies and approaches that could significantly improve existing method based on some useful codes.

Enright [8] modified the RK methods of [7] to handle discontinuities. Some of the codes in existence include.

i. Multiple Shooting:- Methods (BVPSOL, [12])
ii. Collocation method (PASVA3, [20])
iii. Finite Difference Method (COLSYS, [5])

The listed three categories of methods for BVPs share a common structure using a modified Newton iteration to obtain numerical approximation which is determined by a discrete solution *x*, satisfying the non-linear equations. The finite difference method and Collocation method schemes are stressing compared to the shooting methods.

The RKM45 derived by Bogacki and Shampine is significantly more efficient than the Fehlberg and Dormand-Prince pairs which has been shown to give better result than the Runge-Kutta Method 4 (see[1]).

RKM45 uses more iterations for certain problems as the step size is selected automatically. But however, the RK4 method may produce a better result if the right step size is used to implement it (see[4]).

Theorem 1

Any linear nth order Ordinary Differential Equation

$$a_n \frac{d^n x}{dt^n} + a_{n-1} \frac{d^{n-1} x}{dt^{n-1}} + \cdots + a_0 x = f(t)$$

Can be reduced to a linear ordinary differential equation.

Theorem 2

Any differential equation of order n,

$$x^{(n)} = f\left(t, x, x'', \ldots, x^{(n-1)}\right), \quad x(a), x''(a), \ldots, x^{(n-1)}(a)\ \ given$$

May be written as a set of *n* first order IVP by defining a new family of known functions

$$x^{(i-1)} = x_i, \qquad for\ i = 1,2,\ldots,n \qquad x_i(a)\ \ given$$

The n-dimensional system of 1st order coupled differential equations is then

$$\begin{aligned} x'_1 &= x_2, & x_1(a)\ \ given \\ x'_2 &= x_3, & x_2(a)\ \ given \\ &\vdots & \vdots \\ x'_{n-1} &= x_n, & x_{n-1}(a)\ \ given \end{aligned}$$

$$x'_n = f(t, x_1, x_2, \ldots, x_n)$$

More compactly in vector form notation

$$\mathrm{x}' = \mathrm{f}(\mathrm{t}, \mathrm{x}) \quad \text{where} \quad \mathrm{x} = (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n)$$

Corollary1*:* the second order IVP define by

$$\mathrm{x}'' = \mathrm{f}\left(\mathrm{t}, \mathrm{x}, \mathrm{x}'\right), \qquad \mathrm{x}(\mathrm{a}), \mathrm{x}'(\mathrm{a})\ \ \text{given}$$

may be written as a set of two first order IVP in the form

$$\begin{aligned} \mathrm{x}'_1 &= \mathrm{x}_2, & \mathrm{x}_1(\mathrm{a})\ \ \text{given} \\ \mathrm{x}'_2 &= \mathrm{f}(\mathrm{t}, \mathrm{x}_1, \mathrm{x}_2), & \mathrm{x}_2(\mathrm{a})\ \ \text{given} \end{aligned}$$

1.2Ode Solver

ODE23:

Integrating the IVP using embedded Runge-Kutta method 2 and 3, we simply use ode23 inbuilt function on matlab. This method is also called the Bogacki-Shampine method and its butcher array/tableau is given by

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{3}{4}$	0	$\frac{3}{4}$		
1	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	
	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	0
	$\frac{7}{24}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{8}$

1.3 Existence of Solution to BVPs

Problem of existence and uniqueness of a solution is more subtle than initial value problems. *x(a) = A* and *x(b) = B* as conditions guarantee that a solution to the problem exists and should be checked before any numerical scheme is applied. Otherwise, a list of meaningless output may guarantee, the existence and uniqueness of the solution of the BVP which is much more difficult than IVPs.

However, if the BVP is linear, we shall give the theorem that guarantees its uniqueness and existence.

Theorem 3

Assume that $f(t,x,x^{'})$ *i*s continuous on the region

$\{R = f(t,x,x^{'})\colon\ a \le t \le b,\ \ -\infty \le x \le \infty,\ \ -\infty \le x^{'} \le \infty\}$ and that

$\frac{\partial f}{\partial x} = f\left(t,x,x^{'}\right) and\ \frac{\partial f}{\partial x} = f_{x}\left(t,x,x^{'}\right)$are continuous on R.

If there exist a constant $M > 0$, for which the partial derivative $f_{x} and\ f\ x^{'}$ satisfy

i. $f_{x}\left(t,x,x^{'}\right) > 0 \qquad \forall\ \left(t,x,x^{'}\right) \epsilon\ R$

ii. $|\ f_{x^{'}}\left(t,x,x^{'}\right)\ | \le M \qquad \forall\ \left(t,x,x^{'}\right) \epsilon R$

Then the BVP $x'' = f(t,x,x^{'})$*with x(a) = A and x(b) = B* has a unique *s*olution

x = x(t) for $a \le t \le b$

Corollary 2: with the assumption that the BVP is linear i.e

$f\ (t,x,x^{'}) = p(t)x^{'} + q(t)x + r(t)$and that the function *f* and its partial derivatives that

$\frac{\partial f}{\partial x} = q(t)\ and\ \frac{\partial f}{\partial x^{'}} = p(t)$are continuous on R if there exist a constant *M* for which $p(t)$ and $q(t)$ satisfy $q(t) > 0\ \forall\ t \in [a,b]$ and

$$p\ (t) \le M = \max_{a<t<b}\{|p(t)|\}$$

Then the linear BVP has a unique solution.

Note that we may use the lotkin relation to determine the range for constant *M* (see [23]).

1.4 Writing BVP in Form of a Set of IVPs

The theorem below guides us on how to write a two point boundary value problems as a set of two 2nd order IVPs and we go further to establish the existence of a unique solution to a linear two point boundary value problems.

Theorem 4

The BVP defines by the linear 2nd order differential equation

$$x^{''} = p\ (t)x^{'} + q(t)x + r(t) \qquad a \le t \le b$$

$x(a) = A \qquad x(b) = B$ *A, B are constant*

It can be written as a set of IVP define by 2nd order differential equation

$u^{''} = p(t)u^{'} + q(t)u + r(t) \quad for\ a \le t \le b$ with the initial condition

$u(a) = A$ *and* $u'(a) = 0$and
$v'' = p(t)v' + q(t)v$*for* $a \leq t \leq b$ *with the initial condition*
$v(a) = 0$ *and* $v'(a) = 1$
And if $v(b) \neq 0$, then the BVP has a unique solution of the form

$$x(t) = u(t) + \frac{B - u(b)}{v(b)} v(t)$$

Proof
The linear combination of the solution to each of the 2nd order equation

$$x(t) = u(t) + cv(t) \qquad (*)$$

is a solution.
$x'' = p_{(t)}x' + q_{(t)}x + r_{(t)}$ as seen by the computation

$$x''(t) = u''(t) + cv''(t)$$
$$= p(t)u'(t) + q(t)u(t) + r(t) + cp(t)v'(t) + Cq(t)v(t)$$
$$= p(t)[u'(t) + Cv'(t)] + q(t)[u(t) + cv(t)] + r(t)$$
$$x''(t) = p(t)x'(t) + q(t)x(t) + r(t)$$

The solution (*) takes the boundary values.

$$x(a) = u(a) + cv(a) = A + 0 = A$$
$$x(b) = u(b) + cv(b)$$

Imposing the boundary condition $x(b) = B$ gives
$c = \frac{B - u(b)}{v(b)}$
Therefore, $v(b) \neq 0$, the unique solution (*) is

$$x(t) = u(t) + \frac{B - u(b)}{v(b)} v(t)$$

Corollary *3*: theBVPdefines by the linear 2nd order difference equation.

$$x'' = p(t)x' + q(t)x + r(t) \qquad a \leq t \leq b$$
$$x(a) = A \qquad x(b) = B$$

Can be written as a set of two set of IVP define by 2nd order differential equations.
$u'' = p(t)u' + q(t)u + r(t)$ for $a \leq t \leq b$with the initial condition.
$u(b) = B$ and $u'(b) = 0$ And
$v'' = p(t)v' + q(t)v$ for $a \leq t \leq b$ with the initial condition

$$v(b) = 0 \text{ and } v'(b) = 1$$

Remark*:* if $q(t) > 0$this rules out the troublesome solution$v(t) = 0$, so that (*) is of the form of the required solution.

1.5 Convergence of the Method

We define error as the difference between the numerical solution w_i *and theoretical solution* x_i*i.e.*

$$e_h(t_i) = x_h(t_i) - w_h(t_i) = x_i - w_i$$

Definition1.1
The numerical method defines by a mesh operator $e_h$is called convergent in the maximum norm when

$$\lim_{h \to o} ||e_h||_\infty = 0.$$

If $||e_h||_\infty = 0(h^p)$with some number $p \geq 1$, then we say that the convergence is of order p.

II. Material and Method

2.1 Shooting Method

The shooting method is a method for solving a BVP by reducing it to the solution of an Initial Value Problem (IVP) beginning the solution at one end of the BVP and shoot to the other end with an Initial Value solver until the boundary condition at the other end converges to its correct value.
We define the two point boundary value problem of a second order ordinary differential equation as follows:

$$x''(t) = f(t, x(t), x'(t)) \qquad (2.1)$$
$$x(t_0) = x_0 \qquad x(t_1) = x_1$$

Let $x(t, \propto)$ denote the solution of the IVP defined by

$$x''(t) = f(t, x(t), x'(t)) \qquad (2.2)$$
$$x(t_0) = x_0 x'(t_0) = \propto$$

We define the function $f(\propto)$ as the difference between $x(t, \propto)$ and the specified boundary value x_1.

$$f(\propto) = x(t, \propto) - x_1$$

If f has a root $\propto$, then obviously, the solution $x(t, \propto)$ of the corresponding (2.2) is also a solution of (2.1)
On the other hand, if the (2.1) has a solution $x(t)$, then $x(t)$ is also a unique solution of the (2.2) of the IVP where $\propto = x'(t_0)$ thus $\propto$is a root of f. We may use the Newton's method to find the roots.
We have Shooting method for linear and non-linear problems. And also shooting method for Dirichlet and non-Dirichlet problems.
However, the general procedure for implementing the linear shooting method is as follows

1. Linearize he problem and reduce the BVP to an IVP.
2. Discretize the problem into desired number of points.
3. Solve the IVP with an integrator. We may choose to use Explicit Eulers method, Trapezoidal Method, Runge-Kutta method etc.
4. Iterate to find the solution.

i.e. for a problem with boundary condition $x_0 = 0$ and $x_1 = 1$. The only boundary condition at $t = 0$ is $x_1(0) = 0$, then we need to guess the value of $x_0(0)$ and then use a predictor-corrector algonthm to shoot to the ther end of the domain and see if this guess satisfies the boundary $x_2(1) = 1$

2.2 Finite Difference Method on Linear Problems

Consider a second order linear equation

$$\frac{d^2x}{dt^2} + p(t)\frac{dx}{dt} + q(t)x = r(t) \qquad a \le t \le b\ x(a) = A \qquad x(b) = B$$

the general procedure for implementing the finite method is as follows :

1. We divide the interval $[a\ b]$ into N+1 equal intervals, each of length h,

we may do this with N + 1 = $\frac{b-a}{h}$ and $t_i = t_0 + ih$, We denote a by t_0, b by t_{N+1}

2. We replace the differential equation by an appropriate finite difference equation by replacing the derivatives by central difference quotient i.e

$$\frac{1}{h^2}(x_{i+1} - 2x_i + x_{i-1}) + \frac{1}{2h}\big(p(t_i)(x_{i+1} - x_{i-1})\big) + q(t_i)x_i = r(t_i) \qquad (2.3)$$

On multiplying through by h^2 and collecting like terms together, we obtain

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = h^2 r(t_i) \qquad (2.4)$$

Where, $a_i = 1 - \frac{h}{2}p(t_i)$, $b_i = -2 + h^2 q(t_i)$, $c_i = 1 + \frac{h}{2}p(t_i)x_0 = A \qquad x_{N+1} = B$

3. Insert the boundary conditions. We have,

$$\begin{aligned} a_1 A + b_1 x_1 + c_1 x_2 &= h^2 r(t_1) \\ a_2 x_1 + b_2 x_2 + c_2 x_3 &= h^2 r(t_2) \\ a_{N-1}x_{N-2} + b_{N-1}x_{N-1} + c_{N-1}x_N &= h^2 r(t_{N-1}) \\ a_N x_{N-1} + b_N x_N + c_N B &= h^2 r(t_N) \end{aligned} \qquad (2.5)$$

4. solve the obtained linear system of equation obtained in 3 above

$$Hx = S \qquad (2.6)$$

Where $x = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}$ $S = \begin{pmatrix} S_1 \\ \vdots \\ S_N \end{pmatrix} = h^2 \begin{pmatrix} r(t_1) \\ \vdots \\ r(t_N) \end{pmatrix} - \begin{pmatrix} a_1 A \\ \vdots \\ c_N B \end{pmatrix}$

and $H = \begin{pmatrix} b_1 & c_1 & 0 & 0 \cdots 0 & 00 \\ a_2 & b_2 & c_2 & 0\ \ 0 & 00 \\ 0 & a_3 & b_3 & c_3 \ldots 0 & 00 \\ & ::: & & :: & :: \\ & \cdots & & \cdot\cdot & \cdot\cdot \\ & 000 & 0 & a_{N-1} & c_{N-1}c_{N-1} \\ & 000 & 0 & 0 & a_N\ \ b_N \end{pmatrix}$

H is a tri-diagonal matrix whose elements are known.

5. Tri-diagonal Solver by the use of Crout Algorithm for LU

Suppose we adopt the LU decomposition approach in providing solution to (2.6)
$Hx = S$, let $H = LU$, So that$LUx = S$, $Ux = L^{-1}S$
Where L^{-1} is the inverse of a lower triangular matrix, which is easily computed as described in chapter two and x is obtained by forward or backward substitutions.

$$\begin{pmatrix} b_1 & c_1 & 0\cdots & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 \\ 0 & a_3 & b_3 & 0 & 0 \\ & ::: & & : & : \\ & \cdots & & . & . \\ & 000 & & b_{N-1} & c_{N-1} \\ & 000 & & a_N & c_N \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0\cdots & 0 & 0 \\ a'_2 & 1 & 0 & 0 & 0 \\ 0 & a'_3 & 1 & 0 & 0 \\ & :: & & : & : \\ & .. & & . & . \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & a'_N & 1 \end{pmatrix} \begin{pmatrix} b'_1 & c_1 & 0\ldots & 0 & 0 \\ 0 & b'_2 & c_2 & 0 & 0 \\ 0 & 0 & b'_3 & 0 & 0 \\ : & : & :: & & : \\ . & . & .. & & . \\ 0 & 0 & b'_{N-1} & & c_{N-1} \\ 0 & 0 & 0 & & b'_N \end{pmatrix}$$

In other words, only two diagonals held to be change from the matrix multiplication and the order of the direct LU decomposition. It easy to derive

$$b'_1 = b_1, \qquad a'_1 = \frac{a_1}{b_1}$$

$$b'_i = b_i - a'_i c_{i-1} \text{from } a'_i c_{i-1} + b'_i = b_i$$

$$a'_{i+1} = \frac{a_{i+1}}{b_i} \text{ from } a'_{i+1} b'_{i-1} = a_{i+1}$$

The decomposition is called the Crout factorization we can have the following Pseudo-code

For $i = 2, N, a_{ii} = (a_i | b_{i-1}), b_{ii} = b_i - a_i c_{i-1}$

End.

Once we have the LU factorization, we can easily derive the formula for forward and backward substitution for solving $Hx = S$

The forward substitution

$x_1 = d_1$ For $i = 2, N$ $x_i = d_i - a_i x_{i-1}$

End

The backward substitution

$j_N = \frac{x_N}{b_N}$ For $i = N-1, 1 j_i = \frac{x_i - c_i x_{i+1}}{b_i}$

End.

The entire process of the solving the tri-diagonal system of equations described above is called the Chasing method.

Note We may normalize the interval $[a, b]$ i.e. to $[0, 1]$ by using $t = (b-a)u + a$ before implementing the step 2 in the above.

III. Analysis and Results

3.1 Problem 1

3.1.1 Shooting Method Solution to Problem 1 Using Trapezoidal Method as Integrator

Consider the linear problem

$$\frac{d^2x}{dt^2} - \left(1 - \frac{t}{5}\right)x = t \qquad 1 \le t \le 3 \qquad x(1) = 2 \qquad x(3) = -1$$

Let $x = x_1$ $x'_1 = x_2$ $x'_2 = x''_1 = t + \left(1 - \frac{t}{5}\right)x_i$

Thus we have system of two equations with two unknown x_1 and x_2

$x'_1 = x_2, \quad x'_2 = t + \left(1 - \frac{t}{5}\right)x_1$ $x_1 = 2x_2 = x'_1 = ?$ where t is known

We guess for x'_2 using the slope $x'(1) x_2 = x'_1 = x'(1) = \frac{x(3) - x(1)}{3-1} = \frac{-1-2}{3-1} = -1.5$

Integrating using Trapezoidal method defined by

$x_{1,n+1} = x_{1,n} + \frac{h}{2}(f_{1,n} + f_{1,n+1})$

Firstly, we compute using Explicit Euler's method to resolve the implicitness encountered using Trapezoidal method.

The explicit Euler's method is given by

$$x^*_{1,n+1} = x_{1,n} + hf_{1,n}$$

$= x_{1,n} + hx'_{1,n}$ since $f = x'$, and $x'_1 = x_2$

$= x_{1,n} + hx_{2,n}$ (3.1)

$x^*_{2,n+1} = x_{2,n} + hf_{2,n}$ $= x_{2,n} + hx'_{2,n}$ where $f = x'$

$= x_{2,n} + h(t_n + (1 - \frac{t_n}{5})x_{1,n})$ (3.2)

The Trapezoidal Method is described by

$$x_{1,n+1} = x_{1,n} + \frac{h}{2}(f_{1,n} + f_{1,n+1}) = x_{1,n} + \frac{h}{2}\left(x_{2,n} + x^*_{2,n+1}\right) \quad (3.3)$$

$$x_{2,n+1} = x_{2,n} + \frac{h}{2}(f_{2,n} + f_{2,n+1}) = x_{2,n} + \frac{h}{2}\left(x'_{2,n} + x'_{2,n+1}\right)$$

Substituting for $x'_{2,n}$ and $x'_{2,n+1}$we have,

$$x_{2,n+1} = x_{2,n} + \frac{h}{2}\left(t_n + x_{1,n} + t_{n+1} + x_{1,n+1} - \left(\frac{t_n x_{1,n} + t_{n+1} x_{1,n+1}}{5}\right)\right) \quad (3.4)$$

For $n = 0$ we substitute $x_{1,0} = x(t_0) = x(1) = 2,\ h = 0.2,$

$t_0 = 1$ $x_{2,0} = \propto_1 = -1.5$ in equ (3.1), (3.2), (3.3), (3.4).

Repeating the procedure for $n = 1,2,\ldots,9$ using the guessed value of $x_{2,0} = x'_1 = x'(1) = \propto_1 = -1.5$ gives the result summarized in the third column of table 1 and we observe that $x_{2,10} = x(t_{10}) = x_1(3) = 4.811$ which does not satisfy the boundary condition at the other end.

For this reason, we choose another guessed value of $x_{2,0} = x'_1 = x'(1) = \propto_1 = -3.0$ and repeat the iteration procedure above for $n = 0,1,2,\ldots,9$ and this gives the result summarized in the fourth column of table 1 and we observe that we obtained $x_{2,10} = x(t_{10}) = x_2(3) = 0.4853$ which does not also satisfy the boundary condition at the other end.

Thus, we interpolate to obtain

$$x'(1) = x'_1(1) + \frac{\propto_2 - \propto_1}{x_2(3) - x_1(3)}[x(3) - x_1(3)] = -1.5 + \frac{-3.0 - (-1.5)}{0.4853 - 4.811}[-1.0 - 4.811] = -3.5$$

Once again, we repeat the iteration procedure for $n = 0,1,2,\ldots,9$ using the interpolated value of $x_{2,0} = x'_1 = x'(1) = -3.5$ as guessed value and the result summarized in the fifth column of table 1, we observed that $x_{2,10} = x(t_{10}) = x(3) = -1.000$ and this satisfies the boundary condition at the other end as required.

Table 1:*Shooting method solution to problem 1 using Trapezoidal Method as integrator*

n	t_n	$x'(1) = \propto_1 = -1.5$		$x'(1) = \propto_2 = -3.0$		$x'(1) = -3.5$	
		$x_{1,n}$	$x_{2,n}$	$x_{1,n}$	$x_{2,n}$	$x_{1,n}$	$x_{2,n}$
0	1	2.000	-1.500	2.000	-3.000	2.000	-3.500
1	1.2	1.1752	-0.987	1.452	-2.510	1.560	-3.001
2	1.4	1.1062	-0.552	0.974	-2.896	0.852	-2.561
3	1.6	1.1218	-0.110	0.437	-2.855	-0.380	-2.174
4	1.8	1.625	0.594	0.328	-1.252	-0.104	-1.876
5	2.0	1.803	1.186	0.118	-0.844	-0.443	-1.521
6	2.2	2.105	1.832	-0.007	-0.417	-0.712	-1.167
7	2.4	2.542	2.542	-0.045	0.040	-1.908	-0.794
8	2.6	3.128	3.324	0.013	0.539	-1.026	-0.391
9	2.8	3.880	4.185	0.175	1.087	-1.060	0.054
10	3.0	4.8110	5.1280	0.4530	1.6930	-1.0001	0.5470

3.1.2 Shooting Method Solution to Problem 1 Using Embedded Rk23 Methodas Integrator

Supposing we decided to Integrate the IVP using Embedded Runge-Kutta Method 2 and 3 (ode23 on MATLAB). It's also called the Bogacki-Shampine method and its butcher is given by ODE23 under 1.3 above. We shall adopt the use of MATLAB inbuilt function ode23 for faster computation.

Firstly, we create an m-file function defined by

```
Function dx=dxsys(t,x)
dx=[x(2); t(1)+(1-t(1)/5)*x(1)];
```

On the command window; we type the following command

```
>> #1st shooting(1st guessed value x'(1) = −1.5)
>> [t, x] = ode23(@dxsys, [1,3], [2,-1.5]);
>> x1 = x(length(x))
X1 = 4.7859
>>p1ot(t, x)
>>ho1d
>>text(3,-1,'*')
>>#2nd shooting(2nd guessed value, x'(1) = −3.0
>>[t,x] = ode23(@dxsys,[1,3],[2 -3.0]);
>>x2 = x(length(x))
x2 = 0.4354
>>plot(x, y)
>>hold
```

Interpolated value

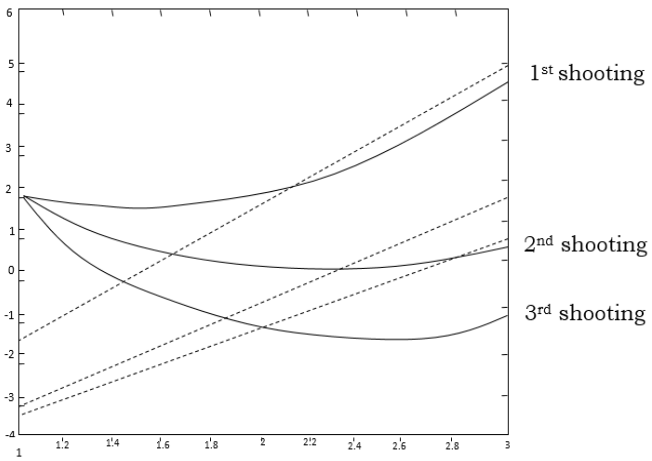
$$x'(1) = x_1^{'}(1) + \frac{\propto_2 - \propto_1}{x_2(3) - x_1(3)}[x(3) - x_1(3)]$$
$$= -1.5 + \frac{-3.0 - (-1.5)}{0.4354 - 4.7850}[-1.0 - 4.7859]$$
$$= -3.495$$

>> #3rd shooting (interpolated value $x'(1) = -3.495$)
>> [t, x] = ode23(@dxsys,[1,3],[2.-3.5])
>> x3 = x(length(x))
X3 = -1.0001
>>plot(t,x)
>>hold
>>text (3,1,'*')
The following was produced

Table 2:*Shooting method solution to solve problem 1 (linear) using embedded RK23 (ode23) method as integrator*

$x^1(l) = \alpha_1$ = -1.5			x'(l) =α_2= -3.0			x'(l) = -3.4950		
t_n	$x_{1,n}$	$x_{2,n}$	t_n	$x_{1,n}$	$x_{2,n}$	t_n	x_1,n	x_2,n
1.0000	2.0000	-1.5000	1.0000	2.0000	-3.0000	1.0000	2.0000	-3.4950
1.0462	1.9335	-1.3806	1.0533	1.8437	-2.8638	1.0458	1.8427	-3.3782
1.2462	1.7083	-0.8718	1.2533	1.3190	-2.3906	1.2458	1.2145	-2.9141
1.4462	1.5846	-0.3631	1.4533	0.8843	-1.9600	1.4458	0.6730	-2.5075
1.6462	1.5640	0.1624	1.6533	0.5333	-1.5522	1.6458	0.2088	-2.1380
1.8143	1.6300	0.6282	1.8533	0.2629	-1.1506	1.8458	-0.1836	-1.7876
2.0143	1.8141	1.2222	2.0533	0.0734	-0.7401	2.0458	-0.5066	-1.4411
2.2143	2.1223	1.8707	2.1838	-0.0050	-0.4614	2.2458	-0.7596	-1.0848
2.4143	2.5665	2.5838	2.3142	-0.0464	-0.1701	2.4458	-0.9392	-0.7064
2.6143	3.1604	3.3696	2.4221	-0.0512	0.0825	2.6458	-1.0400	-0.2950
2.8143	3.9194	4.2343	2.5284	-0.0287	0.3432	2.8458	-1.0545	0.1590
3.0000	4.7859	5.1109	2.6101	0.0078	0.5528	3.0000	-1 .0007	0.5436
			2.6919	0.0619	0.7709			
			2.7959	0.1570	1.0612			
			2.9344	0.3321	1.4721			
			3.0000	0.4354	1.6770			

Figure 1: Graph showing shooting method solution to problem 1



Comparing the two tables, we observe that the both method provides good and acceptable solution. The difference between the approximation provided by Trapezoidal Method and Bogacki-Shampine Method (ode23) is subtle (-1.0001 is closer to -1.0000 than -1.0007). It is easier to write and implement a solution using Bogacki-.Shampine Method on MATLAB since it has inbuilt function called ode23. Ode23 gives less smooth figures than ode45.

However, for a linear problem, using Bogacki-Shampine Method may be consider as a waste of resources since Trapezoidal Method gives us a better result with an acceptable error tolerance. In solving a non-linear problem, the use of ode23 and ode45 cannot be over-emphasized.

The consequence of having $x = c_1x_1 + c_2\,x_2$as a solution enable us to be able to calculate the value of x by taking the proper combination of the values obtained from the earlier calculations.

At the left end of table 1 $t = 1,\ x = c_1(2) + c_2(2) = 2$

At the right end of table 1 $t = 3,\ x = c_1(4.811) + c_2(0.453) = -1.0$

Solving simultaneously yield $c_1 = -0.3334$, $c_2 = 1.3334$

Which implies that $x = -0.3334(1.803) + 1.3334(0.118) = -0.443$

as shown on the table 1.

3.1.3 Finite Difference Method Solution to Problem 1

Consider the linear problem

$\frac{d^2x}{dt^2} - \left(1 - \frac{t}{5}\right)x = t \qquad x(1) = 2 \qquad x(3) = 1$

Using h=0.2, We discretize [1,3] into $N + 1 = \frac{3-1}{0.2} = 10 points$

This implies $i = 0, \ldots 10,$

$$x_0 = x(t_0) = x(1) = 2\ and\ x_{10} = x(t_{10}) = x(3) = -1$$

Is given while we compute for $x_1\ to\ x_9$

Replacing the derivative by the central difference quotient yield

$$\frac{x_{i+1} - 2x_i + x_{i-1}}{h^2} - \left(1 - \frac{t_i}{5}\right)x_i = t_i$$

Multiplying through byh^2, we have

$$x_{i+1} - 2x_i + x_{i-1} - h^2\left(1 - \frac{t_i}{5}\right)x_i = h^2t_i$$

Substituting $h = 0.2$, we obtained

$x_{i-1} - (2.04 - 0.008t_i)x_i + x_{i+1} = 0.04t_i$ (3.5)

Where $i = 1, \ldots, 9$ with boundary condition written as$x_0 = 2 \qquad x_{10} = -1$

Inserting the boundary condition reduces (4.5) to a system of equation in the matrix form

$$\begin{pmatrix} -2.0304 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2.0288 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2.0312 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2.0256 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2.0249 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2.0224 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2.0208 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2.0192 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2.0176 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{pmatrix} = \begin{pmatrix} -1.952 \\ 0.056 \\ 0.064 \\ 0.072 \\ 0.08 \\ 0.088 \\ 0.096 \\ 0.104 \\ 1.112 \end{pmatrix}$$

H X S

We use the LU algorithm described in chapter three to solve the system of equation above by first decomposing H into L and U.

We simply do this using the Matlab program

>>H= [-2.0304 1 0 0 0 0 0 0 0;1-2.0288 1 0 0 0 0 0 0; 1-2.031210... -2.01 76]

>>[LUP] = 1u(H)

>>S=[-1.952 0.0056 0.0064 0.0072 0.008 0.088 0.096 0.104 1.112]

>>d=p*S
>>g= L\d
>>x = u\g

$$>> x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{pmatrix} = \begin{pmatrix} 1.3506 \\ 0.7903 \\ 0.3087 \\ -0.0993 \\ -0.4379 \\ -0.7069 \\ -0.9035 \\ -1.0230 \\ -1.0579 \end{pmatrix}$$

Table 3: Comparison of the Solutions to Problem 1

T	Finite difference method solution (x)	Shooting method solution (x)
1.0	2.000	2.000
1.2	1.351	1.348
1.4	0.792	0.787
1.6	0.311	0.305
1.8	-0.097	-0.104
2.0	-0.436	-0.443
2.2	-0.705	-0.712
2.4	-0.903	-0.908
2.6	-1.022	-1.026
2.8	-1.058	-1.060
3.0	-1.000	-1.000

Clearly from the above table, the result from the shooting method are more accurate than that of the finite difference method. This may not be clearly seen since problem 1 has no analytical solution to compare with. However, it is pertinent to note that the accuracy of the shooting method is also dependent upon the integrator used. This is shown in the solution to problem 2.

PROBLEM 2

3.2.1 Shooting Method on Problem 2 Using Classical Rk4 as Integrator

$x'' = -\frac{2}{t}x' + \frac{2}{t^2}x + \frac{\sin(\square lnt)}{t^2}$ $for\ 1 \le t \le 2, x(1) = 1$ and $x(2) = 2$

Has an exact solution $x = c_1 t + \frac{c_2}{t^2} - \frac{3}{10}\sin(lnt) - \frac{1}{10}\cos(lnt)$,

Where $c_2 = \frac{1}{70}(8 - 12\sin(ln2) - 4\cos(ln2))and\ c_1 = \frac{11}{10} - c_2$

We reduce the BVP to an IVP and then write the second-order differential equation obtained as two first-order differential equation as follows:

Applying the linear shooting method described by theorem 4 requires approximating the solutions to the initial-value problems

$u'' = -\frac{2}{t}u' + \frac{2}{t^2}u + \frac{\sin(\square lnt)}{t^2}$ $for\ 1 \le t \le 2, u(1) = 1$ $and\ u'(1) = 0$(3.6a)

And $v'' = -\frac{2}{t}v' + \frac{2}{t^2}v$ $for\ 1 \le t \le 2, v(1) = 0$ and $v'(1) = 1$(3.6b)

The first second-order equation i.e. (3.6a) is written as a system of two first order differential equations as follows:

Let $u = u_1$ $u' = u'_1 = u_2 u'' = u''_1 = u'_2$

to give

$$u'_2 = -\frac{2}{t}u_2 + \frac{2}{t^2}u_1 + \frac{\sin(\square lnt)}{t^2}$$

$$\left.\begin{array}{l} u'_1 = u_2 \\ u_1(1) = 1 \ \ and \ \ \ u_2(1) = 0 \end{array}\right\} \quad (3.7a)$$

The 2nd order equation i.e. (3.6b) is written as a system of two first- order differential equations as follows:

Let $v = v_1 v' = v'_1 = v_2 v'' = v''_1 = v'_2$

So that

$$v'_2 = -\frac{2}{t}v_2 + \frac{2}{t^2}v_1 \qquad \left.\begin{array}{l} v'_1 = v_2 \\ v_1(1) = 0 \quad and v_2(1) = 1 \end{array}\right\} \text{(3.7b)}$$

Discretizing the problem
Supposing we chose $h = 0.1$

$$0.1 = \frac{2-1}{N}$$

$$N = \frac{1}{0.1} = 10$$

And we have $N + 1 = 11$ points

$$i = 0,1,2,\dots N,$$

$$x(t_0) = x(1) = 2, \qquad t_n = a + nh$$

$$t_0 = 1, \quad t_1 = 1 + 0.2, \quad t_2 = 1 + 2(0.2), \dots, \quad t_N = a + Nh$$

$$i = 0, \quad i = 1, \quad i = 2, \dots \quad i = N$$

Integrating using classical RKM of order 4,
We will use classical RKM of order 4 within Maple to solve both 3.7a and 3.7b
To solve 3.7a, enter the following command
>sys1:=D(u1)(t)=u2(t), D(u2)(t) = -2*u2(t)/t+2*u1(t)/t^2+sin(ln(t))/t^2 ;
>init1 :=u1(1)=1,u2(1)=0
We denote the solution to 3.7a by g1. The Runge-Kuttta method of order4 is invoked with the command
>gl:=dsolve({sysl,init1},numeric,
method=classical[rk4l,{u1(t),u2(t)},stepsize=0.1);
To solve 3.7b, enter the following command
> sys2: D(vl)(t)=v2(t), D(v2)(t)=-2*v2(t)/ t+2*vl (t)/ t^2
> init2: =v1(1)=0, v2(1)=1;
We denote the solution to 3.7b by g2. The Runge-Kutta method of order4 is invoked with the command
>g2:=dsolve({sys2,init2),numeric,method=classical[rk4],{vl(t),v2(t)},
Stepsize=0.1);
To combine solution to 3.7a and 3.7b
We form $x(t) = u(t) + \frac{2-u(2)}{v(2)}v(t)$ using the following maple code.
>c:=(2-rhs(gl(2)[2]))/rhs(g2(2)[2]);
>for i from 1 to 10 do
>x:=1+0.1*i;
>w[i]:=rhs(gl(x)[2])+c*rhs(g2(x)[2]);
>end.
This gives the results presented in the fifth column of table 4. The value as $u_{1.i}$ approximates $u(t_i)$, the value of $v_{1.i}$ approximates $v(t_i)$and w_i. Note that
$w_i = u_{1.i} + c^* v_{1.i}$ where $c^* = \frac{2-1.46472815}{0.58332538} = 0.9176213968$

Table 4: Shooting method solution to problem 2 using classical RK 4 as integrator

i	t_i	$u_{1.i}$	$v_{1.i}$	$w_{1.i}$
0	1.0	1.00000000	0.00000000	1.00000000
1	1.1	1.00896058	0.09117986	1.09262916
2	1.2	1.03245472	0.16851175	1.18708471
3	1.3	1.06674375	0.23608704	1.28338227
4	1.4	1.10928795	0.29659067	1.38144589
5	1.5	1.15830000	0.35184379	1.48115939
6	1.6	1.21248372	0.40311695	1.58239245
7	1.7	1.27087454	0.45131840	1.68501396
8	1.8	1.33273851	0.49711137	1.78889854
9	1.9	1.39750618	0.54098928	2.89392951
10	2.0	1.46472815	0.58332538	2.00000000

Read section 3.2.3 for comments on tabulated result.

When do we Say the Shooting Method Fails?

On implementing the shooting method, there can be round-off error problems hidden in the technique. If $u(t)$ rapidly increases as t goes from a to b, then

$u_{1.i} \approx u(b)$ will be large. Should B be small in magnitude compared to $u_{1.N}$, the term $(B - u_{1.N})/v_{1,N}$ will be approximately $-u_{1.N}/v_{1.N}$, so the approximations

$$x(t_i) \approx w_i \approx u_{1.i} - \left(\frac{B - u_{1.N}}{v_{1.N}}\right) v_{1.i} \approx u_{1.i} - \left(\frac{u_{1.N}}{v_{1.N}}\right) v_{1.i}$$

Allow the possibility of a loss of significant digits due to cancellation. This is considered a failure of the shooting method in providing solution to the problem.

Remedying the Shooting Method Failure

Since $u_{1.i}$, is an approximation to $u(t_i)$, the behavior of u can be easily monitored and $u_{1.i}$ increases rapidly from a to b, the shooting method can be employed in the other direction. That is, solving instead of the Initial Value Problem

$u'' = p(t)u' + q(t)u + r(t)$ for $a \le t \le b$ where

$u(b) = B \; and \; u'(b) = 0$and

$v'' = p(t)v' + q(t)v$ for $a \le t \le b$ where

$$v(b) = 0 \;\; and \;\; v'(b) = 1$$

If the reverse shooting technique still gives cancellation of significant digits and if increased precision does not yield greater accuracy, other techniques may be employed.

3.2.2 Finite Difference Method Solution to Problem 2

Consider the linear problem 2

$x'' = -\frac{2}{t}x' + \frac{2}{t^2}x + \frac{\sin(\ln t)}{t^2}$ $for\ 1 \le t \le 2$, x(1) = 1 and x(2) = 2

3.2.3 COMMENTS ON SOLUTION TO PROBLEM 2

Table 6: Comparing Finite Difference Method and Shooting Method Solution to Problem 2

		Analytic solution	Shooting method solution		Finite difference method solution	
i	t_i	$x(t_i)$	$w_{1,i}$	$\lvert x(t_i) - w_{1,i}\rvert$	w_i	$\lvert x(t_i) - w_i\rvert$
0	1.0	1.00000000	1.00000000		1.00000000	
1	1.1	1.09262930	1.09262916	1.34×10^{-7}	1.09260052	2.88×10^{-5}
2	1.2	1.18708484	1.18708471	1.39×10^{-7}	1.18704313	4.17×10^{-5}
3	1.3	1.28338236	1.28338227	9.78×10^{-8}	1.28333687	4.55×10^{-5}
4	1.4	1.38144595	1.38144589	6.02×10^{-8}	1.38140205	4.39×10^{-5}
5	1.5	1.48115942	1.48115939	3.06×10^{-8}	1.48112026	3.92×10^{-5}
6	1.6	1.58239246	1.58239245	1.08×10^{-8}	1.58235990	3.26×10^{-5}
7	1.7	1.68501396	1.68501397	1.43×10^{-8}	1.68498902	2.49×10^{-5}
8	1.8	1.78889853	1.78889854	1.05×10^{-8}	1.78888175	1.68×10^{-5}
9	1.9	1.89392951	1.89392954	3.41×10^{-8}	1.89392110	8.41×10^{-6}
10	2.0	2.00000000	2.00000000		2.00000000	

Note that the results from the finite difference method are considerable less accurate than those obtained using the shooting method. This is because the shooting method used involves a Runge-Kutta technique with error of order $O(h^4)$ whereas the difference method used here has error of order $O(h^2)$.

To obtain a difference method with greater accuracy, we can proceed in a number of ways. Using fifth-order Taylor series for approximating $x''(t_i)$ and $x'(t_i)$ result in an error term involving h^4.

i.e.

Central difference approximation for 4th order, 1st derivative,

$$\left|\frac{dx}{dt}\right|_i = \frac{-x_{i+2} + 8x_{i+1} - 8x_{i-1} + x_{i-2}}{12h} + O(h^4)$$

Central difference approximation for4th order, 2nd derivative

$$\left|\frac{d^2x}{dt^2}\right|_i = \frac{-x_{i+2} + 16x_{i+1} - 30x_i + 16x_{i-1} - x_{i-2}}{12h^2} + O(h^4)$$

Using this approximation leads to difficulty at $i = 0 \; and \; i = N$. Moreover, the resulting solution of equations is not in tri-diagonal form and the solution to the system requires many more calculations. Instead of obtaining a difference method with a higher-order error term in this manner, it is generally more satisfactory to consider a reduction in the step size.

IV. Conclusion

The shooting and finite difference methods provide solutions to BVP which has no exact or unique solution. The accuracy of the shooting method is dependent upon the integrator adopted. When a preferable step size is used, the shooting method which utilized the Trapezoidal Method and the Runge-kutta Method as integrators provide us with a better approximation compared to the solution obtain when an embedded Runge-kutta method is used as an integrator.

However, the embedded Runge-kutta provide us with an approximation to solution that has a good error tolerance i.e convergence is guaranteed. It is easier to write and execute a solution to BVP using the embedded Runge-kutta formula on most computer software since the step size is automatically computed and there exist an in-built function for it on most numerical software (MATLAB and MAPLE). Although this iteration may take longer computational time and the memory size of the computer as when compared with the classical Runge-kutta method.

Generally, the shooting method gives cancellation of significant digits and if increased precision does not yield greater accuracy, we employ the finite difference method which utilizes higher order central difference scheme with a reduction in the step size. This also have its own limitation as it leads to heavy computation.

References

[1]. Bogacki P. and Shampine L"A 3(2) pair of Runge-Kutta formulas". Appl. Math. . Vol. 2 (1989), pages 321-325.
[2]. Burrage, K. "Special Family of Runge-Kutta Methods," (1978), Ell 18 22-41
[3]. Butcher J. C. "The Numerical Analysis of Ordira y Differential Equations, Runge-Kutta and General Linear Methods" Wiley, Chichester, New York. (1987),
[4]. Butcher, J. C. "A Multistep Generalization of R-K Method 3 with Four or Five stages", Journal of tlw ACM 14, (1967), 84-99.
[5]. Causon M. and Mingham G. "Introduction to Finite Difference Methods for partial Differential Equations Ventus publishing" (2010), Aps ISBN 978-87-7681-642-1.
[6]. Dormand J and Prince P "A Family of Embedded Runge-Kutta formulae". J. Comput. Appl. Math., 6,(1980),:19-26.
[7]. Dormand J and Prince P "High order embedd d Runge-Kutta formulae". J. Comput. Appl. Math., 7,(1986):67-75,
[8]. Enright H "A New Error Control for IVP." ODE conference held at Sandia National Lab., Alnuquerque, New Mexico, USA July 1986.
[9]. Enright H, Hull T and Lindberg B "Comparing numerical methods for stiff systems of Ordinary Differential Equation" vol. 15(1975) pages 10-48.
[10]. Fehlberg E. "Low Order Classical R-K Formulas with s 3p size control and their application to some heat transfer problems" ASA Technical Report No. 315, George C. Marshall Space Flight Center, Hunville. (1969),
[11]. Gear C "The Numerical Integration of ODEs." Mathematics of Computation 21. (1967). 146-156.
[12]. Hans G and Karl J "A multiple shooting Algorithm for direct solution of optimal control problems" Institution of Mathematics University of Bonn, 5300 Bonn Federal Republic of Germany 2-6 July. (1984)
[13]. Harrer H "A study on 3rd order Runge-kutta Techniques for solving practical problems. (2013)
[14]. Houwen V "On the internal stability of Explicit m-stage Runge-Kutta methods for Large m-values. Journal of applied mathematics and mechanics. (1980)
[15]. Henrici P Discrete Variable Methods in ODEs, New York John Wiley & Sons. (1962)
[16]. Kayode S "An Efficient Zero Stable Numerical Method For 4th Order Equation", International Journal of Mathematics and Mathernatics Science, Hindawi Publishing Corp. (2009),
[17]. Keller H "Numerical Method for Two Point E boundary Value Problem", Grimm Blaisdell, Waltham, Mass, (1968) pp 111-129.
[18]. Keller H "Numerical Method for Two Point Boundary Value Problem", Regional Conference in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia. (1976)
[19]. Lambert D Computational Methods in ODEs, New York: John Wiley. (1973)
[20]. Lentini M and Pereyra V "PASVA3: An Adaptive finite Difference Program for First Order, Nonlinear Ordinary Boundary Problems," Lecture Notes in Computer Science 76 (B. Childs, ed.), New York: Springer Verlag.(1979)
[21]. Rutishauser H "Lecture Note on Numerical Mathematics, Library of Congress. (1976)
[22]. Shampine and Gordons "Advanced system of modelling and simulation with Block Diagram Languages. (1975)
[23]. Lotkin M "On the Accuracy of R-K Methods." MTAC 5, (1951), 118-132.

S. Markus"Comparative Analysis of Linear two point Boundary Value Problems via Shooting and Finite Difference Methods." IOSR Journal of Mathematics (IOSR-JM) 14.1 (2018): 49-61.