[Ibrahim et. al., Vol.6 (Iss.1): January, 2018]
(Received: Dec 28, 2017 - Accepted: Jan 18, 2018)
ISSN- 2350-0530(O), ISSN- 2394-3629(P)
DOI: 10.29121/granthaalayah.v6.i1.2018.1591

INTERNATIONAL JOURNAL OF RESEARCH – GRANTHAALAYAH
A knowledge Repository

Science

ON SHOOTING AND FINITE DIFFERENCE METHODS FOR NON-LINEAR TWO POINT BOUNDARY VALUE PROBLEMS

Ibrahim I. O. [1], Markus S. [2]
[1, 2] Department of Mathematical Sciences, University of Maiduguri, Borno State, Nigeria

Abstract

The paper investigates the efficacy of non-linear two point boundary value problems via shooting and finite difference methods. It was observed that the shooting method provides better result as when compared to the finite difference methods with dirichlet boundary conditions. It was observed that the accuracy of the shooting method is dependent upon the integrator adopted.

Keywords: BVP; MATLAB; MAPLE; Finite Difference Method; Shooting Method; RK45; RK4; Secant Method; Newton's Method.

Cite This Article: Ibrahim I. O., and Markus S.. (2018). "ON SHOOTING AND FINITE DIFFERENCE METHODS FOR NON-LINEAR TWO POINT BOUNDARY VALUE PROBLEMS." *International Journal of Research - Granthaalayah*, 6(1), 23-35. https://doi.org/10.29121/granthaalayah.v6.i1.2018.1591.

1. Introduction

In the West, only the method defined by Euler [9] is called the Euler method, more precisely, Euler's forward method. The method defined by Henrici [8] is usually called the midpoint method and is known as the Runge method [7] or modified Euler method, which is considered as the oldest method of Runge–Kutta method characterized by the property that each step involves a multiplicity of evaluations of the right-hand side function which is sometimes called Heun's second-order method if it is predicted by Euler's forward method, and it is called the trapezoidal rule. An iterative solution of the trapezoidal rule is also considered (see [6]).

Although the Euler method [9] is not recommended in actual computation, it serves as a model for theoretical considerations and facilitates comparison with more complicated methods after Euler publishes his method.

Arieh [5] proved Cauchy convergence with the use of implicit Euler's method.

it is always better to obtain an exact solution for the given differential equations but, due to some complications like time consumption and more manual operations, it is not possible to find analytical solution for such mathematical problems. Therefore, it is necessary to approximate

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
(Received: Dec 28, 2017 - Accepted: Jan 18, 2018) DOI: 10.5281/zenodo.1162064

(numerical) solutions. There are so many numerical methods solving available for such differential equation such as power series method, pointwise method, Taylor's method, Picard's method, Euler's method, Improved Euler's method, Modified Euler's method, Runge-kutta second and fourth order method, predictor corrector method etc. Runge-kutta techniques have become very popular and efficient tool for computational purposes [1-4] due to many real time application problems are solved effectively. Popular RK algorithms are adopted to solve differential equations efficiently that are equivalent to approximate the exact solutions by matching 'n' terms of the Taylor series expansion. Harrer [10] introduced Explicit Euler, Predictor-corrector and fourth-order Runge-kutta algorithms which are used for simulating cellulose neural works.

2. Materials and Methods

2.1. Shooting Method for Non-Linear Problems

Consider

$$x''(a) = f\big(t, x(t), x'(t)\big) \tag{2.1}$$
$$a \le t \le b$$
$$x(a) = A \qquad x(b)=B$$

Where $f\big(t, x(t), x'(t)\big)$ Is not linear in x andx'. Assume a given boundary value problem has a unique solution $x(t)$. we will approximate the solution $x_{(t)}$ by solving a sequence of IVPs

$$x''(a) = f(t, x(t), x'(t)) \tag{2.2}$$
$$a \le t \le b$$
$$x(a) = A \qquad x'(a) = s_k$$

Where s_k is a real number. Let $x(t, s_k)$ be solution of the IVP (2.2), we want to have a sequence $\{s_k\}$ so that

$$\lim_{k\to\infty} x(t, s_k) = x(t)$$

One of the choices for s_0 is

$$s_0 = x'(a) \approx \frac{x(b)- x(a)}{b-a} = \frac{B-A}{b-a}$$

How to choose s_k for $k \ge 1$?

Consider s such that

$$x(b, s) - B = 0 \tag{2.3}$$

That is s is a solution of the equation. Observe that the equation $x(b, s) - B = 0$ is a non-linear equation in one-variable. We can then use either Secant method or Newton method for non-linear equation of the form

$$s_k = s_{k-1} - \frac{g(s_{k-1})(s_{k-1}-s_k)}{g(s_{k-1})- g(s_{k-2})}$$

For s_0, the Newton's method computes

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
DOI: 10.5281/zenodo.1162064

$$s_k = s_{k-1} - \frac{g(s_{k-1})}{g'(s_{k-1})}$$

These methods can be used here to solve the non-linear (2.3)

Use the **secant method** to approximate the solution of $x(b,s) - B = 0$

$$s_k = s_{k-1} - \frac{[x(b, s_{k-1}) - B][s_{k-1} - s_{k-2}]}{x(b, s_{k-1}) - x(b, s_{k-2})}$$

Note that $x(b, S_{k-1})$ is the last element in the array x. Note that also that we will need two initial choices s_0 and s_1 to compute s_2 in other to continue the iterations.

Use **Newton's method** to approximate the solution of $x(b,s) - B = 0$

$$s_k = s_{k-1} - \frac{x(b, s_{k-1}) - B}{\frac{dx}{ds}(b, S_{k-1})}$$

Again $x(b, S_{k-1})$ is the last element in the array x, we do not know $x(t)$ explicitly, how we can determine $\frac{dx}{dt}(b, S_{k-1})$?

Let $x(t,s)$ be the solution of the IVP (2.2). Then from (2.2), we have

$$x''(t,s) = f\left(t, x_{(t,s)}, x'_{(t,s)}\right) \qquad a \le t \le b$$
$$x(a,s) = A \qquad x'(a,s) = s \tag{2.4}$$

By differentiating both sides of the equation w.r.t. s, we have

$$\frac{\partial x''(t,s)}{\partial s} = \frac{\partial f(t, x(t,s), x'(t,s))}{\partial s}$$
$$= f_t t_s + f_x \frac{\partial x(t,s)}{\partial s} + f_{x'} \frac{\partial x'(t,s)}{\partial s}$$

Since t and s are independent, $t_s = 0$, hence,

$$\frac{\partial x''(t,s)}{\partial s} = f_x \frac{\partial x(t,s)}{\partial s} + f_{x'} \frac{\partial x'(t,s)}{\partial s}$$

For $a \le t \le b$, the initial conditions are

$$\frac{\partial x(a,s)}{\partial s} = \frac{d(A)}{ds} = 0$$
$$\frac{\partial x'(a,s)}{\partial s} = \frac{d(s)}{ds} = 1$$

defines $z(t,s) = \frac{dx(t,s)}{ds}$, since

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
DOI: 10.5281/zenodo.1162064

$$\frac{\partial^3 x(t,s)}{\partial t^2 \partial s} = \frac{\partial}{\partial s}\left(\frac{\partial^2 x(t,s)}{\partial t^2}\right) = \frac{\partial}{\partial s}(x''(t,s))$$

We denote

$z''(t,s) = \frac{\partial}{\partial s}(x''(t,s))$
Then the IVP (2.4) becomes
$z''(t,s) = f_x z(t,s) + f_{x'} z'(t,s) \qquad a \le t \le b$ (2.5)

$z(a,s) = 0 \qquad z'(a,s) = 1$

We can update our S_k using the information from z (t, s) as follows

$S_k = S_{k-1} - \frac{x(b,S_{k-1}) - B}{z(b,S_{k-1})}$

2.2. Finite Difference Method on Non-Linear Problems

The difference method for the general non-linear boundary value problem

$x'' = f(t,x,x').$ for $a \le t \le b,$ where
$x(a) = A$ and $x(b) = B$

Is similar to the method applied to linear problems in last section. However, the system of equations will not be linear, so an iterative process is required to solve it.

As in the linear case, we divide $[a,b]$ into $(N+1)$ equal subintervals whose endpoints are at $x_i = a + ih$ for $i = 0,1,\dots,N+1$.

Assuming that the exact solution has a bounded fourth derivative allows us to replace $x''(t_i)$ and $x'(t_i)$ in each of the equations by the appropriate centered-difference formula to obtain. For each $i = 1,2,\dots,N,$

$$\frac{x(t_{i+1}) - 2x(t_i) + x(t_{i-1})}{h^2} = f\left(t_i, x(t_i), \frac{x(t_{i+1}) - x(t_{i-1})}{2h} - \frac{h^2}{6}x'''(\eta_i)\right) + \frac{h^2}{12}x^{(4)}(\xi_i)$$

For some (ξ_i) and (η_i) in the interval (t_{i-1}, t_{i+1}).

The difference method results when the error terms are deleted and the boundary conditions are added. This produces the $N \times N$ non-linear system.

$$2W_1 - w_2 + h^2 f\left(t_1, w_1, \frac{w_2 - A}{2h}\right) - A = 0$$
$$-w_1 + 2w_2 - w_3 + h^2 f\left(t_2, w_2, \frac{w_3 - w_1}{2h}\right) = 0$$
$$\vdots$$

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
DOI: 10.5281/zenodo.1162064

$$-w_{N-2} + 2w_{N-1} - w_N + h^2 f\left(t_{N-1}, w_{N-1,} \frac{w_N - w_{N-2}}{2h}\right) = 0$$
$$-w_{N-1} + 2w_N + h^2 f\left(t_N, w_{N,} \frac{B - w_{N-1}}{2h}\right) - B = 0$$

To approximate the solution to this system, we use Newton's method for non-linear systems, as discussed in section 2.1. A sequence of iterates.

$\left\{(w_1^{(k)}, w_2^{(k)}, \dots, w_N^{(k)})^t\right\}$ is generated that converges to the solution of system, provided that the initial approximation $(w_1^{(0)}, w_2^{(0)}, \dots, w_N^{(0)})^t$ is amply close to the true solution, $(w_1, w_2, \dots, w_N)^t$).

Newton's method for nonlinear systems requires solving, at each iteration, an $N \times N$ linear system involving the Jacobian matrix. In our case, the Jacobian matrix is tri-diagonal, and Crout factorization can be applied. The initial approximations to $w_i^{(0)}$ to w_i for each

$i = 1,2,\dots,N$, are obtained by passing a straight line through (a, A) and (b, B) and evaluating at t_i.

Since a good initial approximation may be required, an upper bound for k should be specified and, if exceeded, a new initial approximation or a reduction in step size is considered.
The program was be used to solve the Nonlinear Finite Difference method on Maple as used in section 3.2.

3. Results and Analysis

Non – linear problem

$$x'' = \frac{1}{8}(32 + 2t^3 - xx') \qquad 1 \le t \le 3 \qquad (3.1)$$
$$x(1) = 17 \qquad x(3) = \frac{43}{3}$$

Has an exact solution

$$x(t) = t^2 + \frac{16}{t}$$

The program can be used to solve the Nonlinear Finite Difference method on Maple as shown below. h = 0.1, to the nonlinear boundary-value problem.

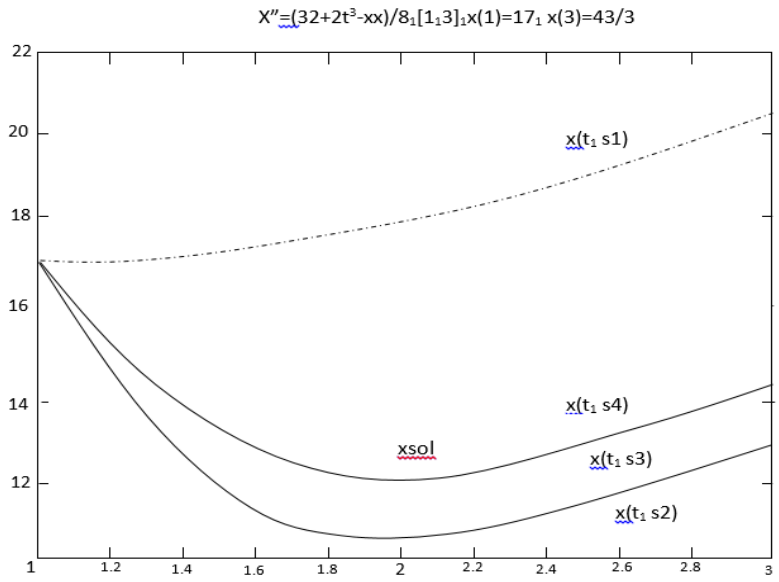
3.1. Shooting Method Solution to Non-Linear Using Rk4 as Integrator

Suppose we want to obtain a better solution for (3.1), we shall consider more digits (i.e of$|x(t_i) - w_{1,i}| \le 10^{-5}$), using Runge Kutta 4, we obtain fourth initial starting value of S(4) = S_3 = -14.3000203 and when used in our fourth iteration (as in appendix) gives the following table.

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
DOI: 10.5281/zenodo.1162064

Table 1: Shooting method solution on on-linear using RK4 as integrator

t_i	Analytic solution	Shooting method solution on non-linear problem		t_i	Shooting method solution on nor linear problem		
	$x(t_i)$	$w_{1,i}$	$\|x(t_i) - w_{1,i}\|$		$x(t_i)$	$w_{1,i}$	$\|x(t_i) - w_{1,i}\|$
1.0	17.000000	17.000000		2.0	12.000000	12.000023	2.30×10^{-5}
1.1	15.755455	15.755495	4.06×10^{-5}	2.1	12.029048	12.029066	2.88×10^{-5}
1.2	14.773333	14.773389	5.60×10^{-5}	2.2	12.112727	12.112741	4.17×10^{-5}
1.3	13.997692	13.997752	5.94×10^{-5}	2.3	12.246522	12.246532	4.55×10^{-5}
1.4	13.388571	13.388629	5.71×10^{-5}	2.4	12.426667	12.426673	4.39×10^{-5}
1.5	12.916667	12.916719	5.23×10^{-5}	2.5	12.650000	12.650004	3.92×10^{-5}
1.6	12.560000	12.560046	4.64×10^{-5}	2.6	12.913845	12.913847	3.26×10^{-5}
1.7	12.301765	12.301805	4.02×10^{-5}	2.7	13.215926	13.215924	2.49×10^{-5}
1.8	12.128889	12.128923	3.14×10^{-5}	2.8	13.554286	13.554284	1.68×10^{-5}
1.9	12.031053	12.031081	2.84×10^{-5}	2.9	13.927241	13.927236	8.41×10^{-5}
				3.0	14.333333	14.333327	



Graph 2: showing Figure shooting method solution to non-linear

Comparing our result of table we observe that RK4 gives a better approximation to the problem than the embedded RK4 since RK4 produces a smaller error (of$|x(t_i) - w_{1,i}| \leq 10^{-5}$) when compared to the actual solution.

3.2. Finite Difference Method Solution to Non – Linear Problems

Suppose we want to obtain a solution of$|x(t_i) - w_{1,i}| \leq 10^{-3}$, for (3.1) via the non – linear

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
DOI: 10.5281/zenodo.1162064

Table 2: Finite Difference Method Solution On Non-Linear

t_i	Analytic solution	finite difference method solution		t_i	finite difference method solution		
	$x(t_i)$	$w_{1,i}$	$\|x(t_i) - w_{1,i}\|$		$x(t_i)$	$w_{1,i}$	$\|x(t_i) - w_{1,i}\|$
1.0	17.000000	17.000000		2.0	12.000000	11.997915	2.085×10^{-3}
1.1	15.755455	15.754503	9.520×10^{-4}	2.1	12.029048	12.027142	1.905×10^{-3}
1.2	14.773333	14.771740	1.594×10^{-3}	2.2	12.112727	12.111020	1.707×10^{-3}
1.3	13.997692	13.995677	2.015×10^{-3}	2.3	12.246522	12.245025	1.497×10^{-3}
1.4	13.388571	13.386297	2.275×10^{-3}	2.4	12.426667	12.425388	1.278×10^{-3}
1.5	12.916667	12.914252	2.414×10^{-3}	2.5	12.650000	12.648944	1.056×10^{-3}
1.6	12.560000	12.557538	2.462×10^{-3}	2.6	12.913845	12.913013	8.335×10^{-4}
1.7	12.301765	12.299326	2.438×10^{-3}	2.7	13.215926	13.215312	6.142×10^{-4}
1.8	12.128889	12.126529	2.360×10^{-3}	2.8	13.554286	13.553885	4.006×10^{-4}
1.9	12.031053	12.028814	2.239×10^{-3}	2.9	13.927241	13.927046	1.953×10^{-4}
				3.0	14.333333	14.333333	

Finite difference method with four iterations produces the result in the table below.

Table 3: Comparing finite difference method and shooting method solution to Non-Linear

t_i	**Analytic solution**	**finite difference method solution**		**shooting method solution**	
	$x(t_i)$	$w_{1,i}$	$\|x(t_i) - w_{1,i}\|$	w_i	$\|x(t_i) - w_i\|$
1.0	17.000000	17.000000		17.000000	
.1.1	15.755455	15.754503	9.520×10^{-4}	15.755495	4.06×10^{-5}
1.2	14.773333	14.771740	1.594×10^{-3}	14.773389	5.60×10^{-5}
1.3	13.997692	13.995677	2.015×10^{-3}	13.997752	5.94×10^{-5}
1.4	13.388571	13.386297	2.275×10^{-3}	13.388629	5.71×10^{-5}
1.5	12.916667	12.914252	2.414×10^{-3}	12.916719	5.23×10^{-5}
1.6	12.560000	12.557538	2.462×10^{-4}	12.560046	4.64×10^{-5}
1.7	12.301765	12.299326	2.438×10^{-3}	12.301805	4.02×10^{-5}
1.8	12.128889	12.126529	2.360×10^{-3}	12.128923	3.14×10^{-5}
1.9	12.031053	12.028814	2.239×10^{-3}	12.031081	2.84×10^{-5}
2.0	12.000000	11.997915	2.085×10^{-3}	12.000023	2.30×10^{-5}
2.1	12.029048	12.027142	1.905×10^{-3}	12.029066	2.88×10^{-5}
2.2	12.112727	12.111020	1.707×10^{-3}	12.112741	4.17×10^{-5}
2.3	12.246522	12.245025	1.497×10^{-3}	12.246532	4.55×10^{-5}
2.4	12.426667	12.425388	1.278×10^{-3}	12.426673	4.39×10^{-5}
2.5	12.650000	12.648944	1.056×10^{-3}	12.650004	3.92×10^{-5}
2.6	12.913845	12.913013	8.335×10^{-4}	12.913847	3.26×10^{-5}

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
DOI: 10.5281/zenodo.1162064

2.7	13.215926	13.215312	6.142×10^{-4}	13.215924	2.49×10^{-5}
2.8	13.554286	13.553885	4.006×10^{-4}	13.554284	1.68×10^{-5}
2.9	13.927241	13.927046	1.953×10^{-4}	13.927236	8.41×10^{-5}
3.0	14.333333	14.333333		14.333327	

4. Conclusion

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 1st derivative, 4th order

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

Central difference approximation for 2nd derivative, 4th order

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

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.

APPENDIX A

Table 4: First Shot Using the First Guessed Value of -1.333 in Providing Solution to Non-Linear Using Integrator RK45

t	x	x^1	z	z'	t	x	x'	z	z'
	$U_1(1)$	$U_1(1)$	$U_3(1)$	$U_4(1)$		$U_1(1)$	$U_1(1)$	$U_3(1)$	$U_4(1)$
1.0000	17.0000	-1.3333	0	1.0000	1.7069	17.2393	1.4504	0.3655	0.2125
1.0001	16.9999	-1.3330	0.0001	0.9999	1.7569	17.3144	1.5531	0.3754	0.1874
1.0001	16.9999	-1.3326	0.0001	0.9998	1.8069	17.3945	1.6501	0.3842	0.1645
1.0002	16.9998	-1.3323	0.0002	0.9997	1.8569	17.4794	1.7421	0.3919	0.1437
1.0002	16.9997	-1.3319	0.0002	0.9996	1.9069	17.5687	1.8298	0.3986	0.1246
1.0005	16.9994	-1.3301	0.0005	0.9990	1.9569	17.6623	1.9138	0.4044	0.1071
1.0007	16.9991	-1.3284	0.0007	0.9985	2.0069	17.7600	1.9948	0.4094	0.0912
1.0010	16.9987	-1.3266	0.0010	0.9980	2.0569	17.8617	2.0732	0.4136	0.0766
1.0012	16.9984	-1.3248	0.0012	0.9974	2.1069	17.9673	2.1496	0.4171	0.0633
1.0025	16.9967	-1.3159	0.0025	0.9948	2.1569	18.0767	2.2244	0.4199	0.0512

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
DOI: 10.5281/zenodo.1162064

1.0037	16.9951	-1.3071	0.0037	0.9921	2.2069	18.1897	2.2980	0.4222	0.0401
1.0050	16.9935	-1.2983	0.0049	0.9895	2.2569	18.3064	2.3708	0.4239	0.0299
1.0062	16.9918	-1.2895	0.0062	0.9869	2.3069	18.4268	2.4430	0.4252	0.0206
1.0125	16.9839	-1.2459	0.0123	0.9738	2.3569	18.5507	2.5151	0.4260	0.0121
1.0188	16.9762	-1.2028	0.0184	0.9609	2.4069	18.6783	2.5872	0.4264	0.0044
1.0251	16.9688	-1.1602	0.0244	0.9482	2.4569	18.8095	2.6597	0.4265	-0.0027
1.0313	16.9616	-1.1182	0.0303	0.9357	2.5069	18.9443	2.7327	0.4262	-0.0092
1.0627	16.9297	-0.9161	0.0588	0.8757	2.5569	19.0827	2.8065	0.4256	-0.0151
1.0941	16.9040	-0.7262	0.0854	0.8196	2.6069	19.2249	2.8813	0.4247	-0.0205
1.1255	16.8840	-0.5477	0.1103	0.7673	2.6569	19.3709	2.9571	0.4235	-0.0255
1.1569	16.8695	-0.3797	0.1336	0.7183	2.7069	19.5207	3.0342	0.4221	-0.0300
1.2069	16.8567	-0.1323	0.1677	0.6467	2.7569	19.6743	3.1127	0.4205	-0.0342
1.2569	16.8558	0.0930	0.1984	0.5820	2.8069	19.8320	3.1926	0.4187	-0.0380
1.3069	16.8657	0.2985	0.2260	0.5236	2.8569	19.9936	3.2741	0.4167	-0.0415
1.3569	16.8854	0.4865	0.2508	0.4706	2.9069	20.1594	3.3573	0.4146	-0.0447
1.4069	16.9141	0.6589	0.2731	0.4225	2.9569	20.3294	3.4423	0.4123	-0.0476
1.4569	16.9511	0.8174	0.2931	0.3789	2.9677	20.3665	3.4608	0.4117	-0.0482
1.5069	16.9956	0.9635	0.3111	0.3391	2.9785	20.4039	3.4794	0.4112	-0.0488
1.5569	17.0472	1.0987	0.3271	0.3030	2.9892	20.4414	3.4980	0.4107	-0.0494
1.6069	17.1053	1.2241	0.3414	0.2700	3.0000	20.4792	3.5168	0.4102	-0.0500
1.6569	17.1695	1.3410	0.3542	0.2399					

APPENDIX B

Table 5: Second shot using the second starting value of -16.3174 in providing solution to non-linear using integrator RK45

t	*x*	*x′*	*z*	*z′*	*t*	*x*	*x′*	*z*	*z′*
	$\mathbf{u}_i(2)$	$\mathbf{u}_1(2)$	$\mathbf{u}_3(2)$	$\mathbf{u}_4(2)$		$\mathbf{u}_i(2)$	$\mathbf{u}_1(2)$	$\mathbf{u}_3(2)$	$\mathbf{u}_4(2)$
1.0000	17.0000	-16.3174	0	1.0000	1.7069	11.3030	-2.9441	0.4305	0.3918
10001	16.9992	-16.3155	0.0001	0.9999	1.7569	11.1673	-2.4886	0.4496	0.3724
1.0001	16.9984	-16.3135	0.0001	0.9998	1.8069	11.0537	-2.0600	0.4677	0.3537
1.0002	16.9975	-16.3116	0.0002	0.9997	1.8569	10.9609	-1.6555	0.4850	0.3356
1.0002	16.9967	-16.3096	0.0002	0.9996	1.9069	10.8878	-1.2723	0.5013	0.3178
1.0005	16.9926	-16.2999	0.0005	0.9990	1.9569	10.8333	-0.9083	0.5168	0.3003
1.0007	16.9885	-16.2901	0.0007	0.9985	2.0069	10.7966	-0.5614	0.5313	0.2829
1.0010	16.9844	-16.2803	0.0010	0.9980	2.0569	10.7769	-0.2299	0.5451	0.2658
1.0012	16.9804	-16.2706	0.0012	0.9974	2.1069	10.7734	0.0876	0.5579	0.2487
1.0025	16.9599	-16.2220	0.0025	0.9948	2.1569	10.7855	0.3925	0.5699	0.2317
1.0037	16.9396	-16.1735	0.0037	0.9922	2.2069	10.8125	0.6859	0.5811	0.2147
1.0050	16.9193	-16.1253	0.0049	0.9895	2.2569	10.8539	0.9689	0.5914	0.1977
1.0062	16.8991	-16.0772	0.0062	0.9869	2.3069	10.9092	1.2422	0.6008	0.1807
1.0125	16.7989	-15.8394	0.0123	0.9741	2.3569	10.9780	1.5067	0.6095	0.1637
1.0188	16.7002	-15.6059	0.0184	0.9615	2.4069	11.0597	1.7630	0.6172	0.1467
1.0251	16.6029	-15.3766	0.0244	0.9493	2.4569	11.1541	2.0117	0.6241	0.1298
1.0313	16.5070	-15.1515	0.0303	0.9374	2.5069	11.2608	2.2534	0.6302	0.1130
1.0627	16.0483	-14.0834	0.0589	0.8819	2.5569	11.3794	2.4886	0.6354	0.0962
1.0941	15.6217	-13.1036	0.0858	0.8325	2.6069	11.5096	2.7175	0.6398	0.0795
1.1255	15.2246	-12.2018	0.1112	0.7883	2.6569	11.6510	2.9406	0.6434	0.0630
1.1569	14.8547	-11.3692	0.1353	0.7487	2.7069	11.8035	3.1582	0.6461	0.0467
1.2069	14.3169	-10.1678	0.1714	0.6933	2.7569	11.9668	3.3706	0.6481	0.0306

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
DOI: 10.5281/zenodo.1162064

1.2569	13.8357	-9.0978	0.2048	0.6458	2.8069	12.1405	3.5780	0.6492	0.0148
1.3069	13.4053	-8.1387	0.2360	0.6045	2.8569	12.3245	3.7806	0.6495	-0.0006
1.3569	13.0203	-7.2735	0.2653	0.5682	2.9069	12.5185	3.9786	0.6491	-0.0157
1.4069	12.6766	-6.4884	0.2929	0.5359	2.9569	12.7223	4.1722	0.6480	-0.0304
1.4569	12.3703	-5.7723	0.3190	0.5068	2.9677	12.7674	4.2133	0.6476	-0.0336
1.5069	12.0984	-5.1157	0.3436	0.4803	2.9785	12.8130	4.2542	0.6473	-0.0366
1.5569	11.8579	-4.5107	0.3670	0.4560	2.9892	12.8590	4.2949	0.6468	-0.0397
1.6069	11.6466	-3.9507	0.3893	0.4333	3.0000	12.9054	4.3355	0.6464	-0.0427
1.6569	11.4622	-3.4301	0.4104	0.4120					

APPENDIX C

Table 6: Third shot using the third starting value of -14.1085 in providing solutions to non-linear using integrator RK45

t	x	x'	z	z'	t	x	x'	z	z'
	$u_1(3)$	$u_1(3)$	$u_3(3)$	$u_4(3)$		$u_1(3)$	$u_1(3)$	$u_3(3)$	$u_4(3)$
1.0000	17.0000	-14.1085	0	1.0000	1.7069	12.2416	-2.1164	0.4195	0.3581
1.0001	16.9993	-14.1067	0.0001	0.9999	1.7569	12.1462	-1.7059	0.4368	0.3368
1.0001	16.9986	-14.1050	0.0001	0.9998	1.8069	12.0706	-1.3208	0.4532	0.3163
1.0002	16.9979	-14.1033	0.0002	0.9997	1.8569	12.0137	-0.9583	0.4685	0.2965
1.0002	16.9972	-14.1016	0.0002	0.9996	1.9069	11.9745	-0.6161	0.4828	0.2773
1.0005	16.9936	-14.0930	0.0005	0.9990	1.9569	11.9518	-0.2921	0.4962	0.2587
1.0007	16.9901	-14.0844	0.0007	0.9985	2.0069	11.9450	0.0155	0.5087	0.2405
1.0010	16.9865	-14.0758	0.0010	0.9980	2.0569	11.9531	0.3082	0.5203	0.2227
1.0012	16.9830	-14.0673	0.0012	0.9974	2.1069	11.9756	0.5874	0.5310	0.2052
1.0025	16.9654	-14.0245	0.0025	0.9948	2.1569	12.0117	0.8545	0.5408	0.1880
1.0037	16.9478	-13.9819	0.0037	0.9922	2.2069	12.0608	1.1104	0.5498	0.1712
1.0050	16.9303	-13.9394	0.0049	0.9895	2.2569	12.1225	1.3561	0.5579	0.1546
1.0062	16.9128	-13.8971	0.0062	0.9869	2.3069	12.1963	1.5925	0.5652	0.1383
1.0125	16.8262	-13.6877	0.0123	0.9740	2.3569	12.2817	1.8205	0.5718	0.1223
1.0188	16.7409	-13.4820	0.0184	0.9614	2.4069	12.3782	2.0406	0.5775	0.1065
1.0251	16.6568	-13.2798	0.0244	0.9492	2.4569	12.4856	2.2536	0.5824	0.0911
1.0313	16.5741	-13.0811	0.0303	0.9371	2.5069	12.6035	2.4599	0.5866	0.0759
1.0627	16.1784	-12.1364	0.0589	0.8809	2.5569	12.7315	2.6601	0.5900	0.0611
1.0941	15.8111	-11.2668	0.0857	0.8306	2.6069	12.8694	2.8546	0.5927	0.0466
1.1255	15.4701	-10.4638	0.1111	0.7852	2.6569	13.0169	3.0439	0.5947	0.0324
1.1569	15.1534	-9.7204	0.1351	0.7441	2.7069	13.1737	3.2284	0.5959	0.0187
1.2069	14.6948	-8.6440	0.1708	0.6862	2.7569	13.3396	3.4083	0.5965	0.0053
1.2569	14.2870	-7.6821	0.2038	0.6360	2.8069	13.5144	3.5840	0.5965	0.0076
1.3069	13.9250	-6.8175	0.2345	0.5918	2.8569	3.6980	3.7558	0.5958	-0.0201
1.3569	13.6040	-6.0357	0.2631	0.5526	2.9069	13.8900	3.9240	0.5945	-0.0321
1.4069	13.3202	-5.3251	0.2898	0.5174	2.9569	14.0903	4.0887	0.5926	-0.0437
1.4569	13.0704	-4.6764	0.3149	0.4855	2.9677	14.1345	4.1238	0.5921	-0.0461
1.5069	12.8517	-4.0814	0.3384	0.4563	2.9785	14.1791	4.1587	0.5916	-0.0485
1.5569	12.6615	-3.5332	0.3606	0.4294	2.9892	14.2240	4.1934	0.5911	-0.0509
1.6069	12.4977	-3.0261	0.3814	0.4042	3.0000	14.2693	4.2280	0.5905	-0.0530
1.6569	12.3583	-2.5552	0.4010	0.3805					

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
DOI: 10.5281/zenodo.1162064

APPENDIX D

Table 7: Fourth shot using the Fourth starting value of -14.0001 in providing solution to non-linear using integrator RK45

t	*x*	*x′*	*z*	*z′*	*t*	*x*	*x′*	*z*	*z′*
	$u_1(4)$	$u_1(4)$	$u_3(4)$	$u_4(4)$		$u_1(4)$	$u_1(4)$	$u_3(4)$	$u_4(4)$
1.0000	17.0000	-14.0001	0	1.0000	1.7069	12.2871	-2.0776	0.4189	0.3566
1.0001	16.9993	-13.9984	0.0001	0.9999	1.7569	12.1935	-1.6695	0.4362	0.3351
1.0001	16.9986	-13.9967	0.0001	0.9998	1.8569	12.0645	-0.9263	0.4677	0.2947
1.0002	16.9979	-13.9950	0.0002	0.9997	1.9069	12.0267	-0.5861	0.4820	0.2755
1.0002	16.9972	-13.9933	0.0002	0.9996	1.9569	12.0056	-0.2642	0.4953	0.2568
1.0005	16.9937	-13.9847	0.0005	0.9990	2.0069	12.0000	0.0414	0.5076	0.2386
1.0007	16.9902	-13.9762	0.0007	0.9985	2.0569	12.0094	0.3322	0.5191	0.2207
1.0010	16.9867	-13.9677	0.0010	0.9980	2.1069	12.0330	0.6096	0.5297	0.2033
1.0012	16.9831	-13.9592	0.0012	0.9974	2.1569	12.0702	0.8747	0.5395	0.1861
1.0025	16.9656	-13.9167	0.0025	0.9948	2.2069	12.1203	1.1288	0.5483	0.1693
1.0037	16.9482	-13.8744	0.0037	0.9922	2.2569	12.1829	1.3727	0.5564	0.1527
1.0050	16.9308	-13.8322	0.0049	0.9895	2.3069	12.2575	1.6074	0.5636	0.1365
1.0062	16.9134	-13.7901	0.0062	0.9869	2.3569	12.3435	1.8336	0.5700	0.1205
1.0125	16.8275	-13.5822	0.0123	0.9740	2.4069	12.4407	2.0521	0.5757	0.1048
1.0188	16.7429	-13.3778	0.0184	0.9614	2.4569	12.5486	2.2633	0.5805	0.0894
1.0251	16.6595	-13.1769	0.0244	0.9491	2.5069	12.6669	2.4680	0.5846	0.0744
1.0313	16.5773	-12.9795	0.0303	0.9371	2.5569	12.7953	2.6666	0.5880	0.0596
1.0627	16.1847	-12.0410	0.0589	0.8809	2.6069	12.9335	2.8596	0.5906	0.0452
1.0941	15.8204	-11.1768	0.0857	0.8305	2.6569	13.0812	3.0474	0.5925	0.0312
1.1255	15.4822	-10.3787	0.1111	0.7850	1.8069	12.1197	-1.2866	0.4525	0.3145
1.1569	15.1681	-9.6397	0.1351	0.7439	2.7069	13.2381	3.2303	0.5937	0.0176
1.2069	14.7133	-8.5697	0.1708	0.6859	2.7569	13.4041	3.4088	0.5943	0.0043
1.2569	14.3091	-7.6133	0.2038	0.6355	2.8069	13.5789	3.5831	0.5941	-0.0085
1.3069	13.9504	-6.7534	0.2344	0.5912	2.8569	13.7624	3.7536	0.5934	-0.0208
1.3569	13.6325	-5.9758	0.2630	0.5519	2.9069	13.9542	3.9205	0.5921	-0.0327
1.4069	13.3516	-5.2691	0.2897	0.5165	2.9569	14.1544	4.0840	0.5902	-0.0441
1.4569	13.1045	-4.6239	0.3147	0.4845	2.9677	14.1985	4.1187	0.5897	-0.0465
1.5069	12.8884	-4.0320	0.3382	0.4552	2.9785	14.2430	4.1534	0.5891	-0.0489
1.5569	12.7006	-3.4867	0.3603	0.4281	2.9892	14.2879	4.1879	0.5886	-0.0512
1.6069	12.5390	-2.9824	0.3810	0.4028	3.0000	14.3332	4.2222	0.5880	-0.0530
1.6569	12.4017	-2.5140	0.4006	0.3791					

Programming

$u'_1 = u_2$ *

$u'_2 = \frac{1}{8}(32 + 2t^3 - u_1u_2)$ **

$u'_3 = u_4$ ***

$u'_3 = -\frac{1}{8}(u_2u_3 - u_1u_4)$ ****

$u_1(1) = 17 \quad u_1(1) = u_2(1) = s_k where\ s_o = -1.333\ as\ calculated\ in\ step\ 1$

$u_3(1) = 0 \quad u_4(1) = 1$

ISSN- 2350-0530(O), ISSN- 2394-3629(P)
DOI: 10.5281/zenodo.1162064

Solving using Non – linear shooting method on maple

We attempt to solve * and ** using the classical R-K 4 in maple
>Sys1:=D(u1)(x) – u2(x), Du2(x) = (32+2*x^3-u1(x))/8
>a:=1; b:=3; N: = 20; h:=(b - a)/N; alpha: = 17; beta:=43/3
>init1: = (beta-alpha) / (b-a);
>g1:=dsolve({sys1,init1}, numeric, method = classical [rk4],{u1(x),},stepsize-h);
Now we solve *** and **** using the following command
>sys2:=D(u3)(x)-u4(x), Du4(x)*u3(x)*u4(x)) /8
>a:=1; b:=3; N:20; h:=(b-a)/N; alpha:= 43/3
>sk:=(beta-alpha) / (b-a);
>init1:=u1(1)=17, u2(1)=sk,u3(1)=0,u4(1)=1;
>g2:=dsolve({sys2,init1},numeric method = classical [rk4],{u1(x),u2(x)u3(x),u4(x)},stepsize-h);
>u1(x),u2(x),u3(x),u4(x)
Gives the result in table 1

Solving using Non – Linear finite difference method on maple

>with(linalg)
>a:=1; b:=3; alpha: =17; beta: =43/3; N:=19; h:=(b-a)/(N+1);
>w:=vector(19,0);
>for 1 from 1 to N do
>t[i]:a+i*h;
>w[i]: = alpha +i*(beta - alpha) / (b-a)*h;
>od;
>A:=matrix(19,19,0)
>>u:=vector(19,0);
>f:=(t,x,xp)->-xp/8;
>fxp:=(t,x,xp)->-x/8
The nonzero entries of H and the right hand side, u of the linear system are generated as follows:
>H[1,1]:=2+h*h*evalf(fx(t[1],w[1],(w[2]-alpha)/(2*h)));
>H[1,2]:=-1+h*evalf(fxp(t[1],w[1],(w[2]-alpha)/(2*h)))/2;
>for 1 from 2 to N – 1 do
>H[i,i-1]:=-1-h*evalf(fxp(t[i],w[i],(w[i+1]-w[i-1])/(2*h)))/2;
>H[i,i+1]:=-1+h*evalf(fxp(t[i],w[i],(w[i+1]-w[i-1])/(2*h)))/2;
>H[i, i]:=-2+h*h*evalf(fx(t[i],w[i],(w[i+1]-w[i-1])/(2*h)));
>u[i]:=-(-w[i-1]+2*w[i]-w[i+1]+h*h*evalf(f(t[i],w[i],(w[i+1]-w[i+1]/(2*h))));
>end;
This gives table 2

References

[1] Williams H Press, Brian P Flannery, Saul A Tuekolsky and Williams T Vetterling. (1986). Numerical Recipes. “THE ART OF SCIENTIFIC COMPUTING” Cambridge University Press, New York.
[2] John C Butcher. (2003). “THE NUMERICAL ANALYSIS OF ORDINARY DIFFERENTIAL EQUATIONS”. John Wiley & Sons, UK.

[3] John C Butcher. (1987). "THE NUMERICAL ANALYSIS OF ORDINARY DIFFERENTIAL EQUATIONS". Runge-Kutta and general linear methods. . John Wiley & Sons, UK.
[4] Mahinder K Jain, satteluri R Iyengar and Rajendra Kumar Jain. (2010). "NUMERICAL METHODS FOR SCIENTIFIC AND ENGINEERING COMPUTATION". 5th edition. New Age International Publishers, Ne Delhi, India.
[5] Arieh Lserles. (1996). "A FIRST COURSE IN THE NUMERICAL ANALYSIS OF DIFFERENTIAL EQUATIONS", Cambridge University Press.
[6] James Murray Watt. (1976). "MODERN NUMERICAL METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS", clarendon press.
[7] Runge C. (1895). "EXPLICIT 2ND ORDER RUNGE-KUTTA METHOD" Martin Luther University, Halle-Wittenberg, Halle.
[8] Henrici Peter. (1962). "DISCRETE VARIABLE METHODS IN ODES", New York John Wiley & Sons.
[9] Leonhard Euler. (1913). "INSTITUTIONAL CALCULI INTEGRALS" primum omnia series prims.
[10] Hubert Harrer, A schuler and E Amelunxen. (1990). "COMPARISON OF DIFFERENT NUMERICAL INTEGRATIONS FOR SIMULATING CELLULOSE NEURAL NETWORKS". In: proceeding of the IEEE International Workshop on cellular neural networks and their Applications, Budapest, p. 151-159.

*Corresponding author.
E-mail address: ibrahimismailaomeiza@ gmail.com