Proceedings of IAM, V.4, N.2, 2015, pp.149-174

SOLVING INITIAL VALUE ORDINARY DIFFERENTIAL EQUATIONS BY MONTE CARLO METHOD

Muhammad Naveed Akhtar[1], Muhammad Hanif Durad[2], Asad Ahmed[3]

[1, 2] Department of Computer and Information Science (DCIS),
Pakistan Institute of Engineering & Applied Sciences (PIEAS)
[3] Pakistan Atomic Energy Commission, Islamabad, Pakistan
e-mail: naveed@ pieas.edu.pk, hanif@pieas.edu.pk, asadahmed@ pieas.edu.pk

Abstract. The objective of this paper is to perform a computational analysis of an existing Monte Carlo based algorithm to solve initial value problem of ordinary differential equations (ODEs). Firstly the problems associated with the existing algorithm have been rectified by suggesting a new elaborate algorithm. Then the new algorithm has been applied to solve different types of ODEs including simple, explicit coupled, implicit and coupled system of first order ODEs. Furthermore the same has also been implemented to known physical systems such as Van der Pol equation and SIR epidemic model. The limitations of proposed algorithm have also been identified by applying Lipschitz continuity check for an exemplary ODE. Finally it has been demonstrated that it still very difficult to propose a computationally efficient algorithm to solve ODEs with considerable accuracy using Monte Carlo method.

Keywords: Monte Carlo integration, ordinary differential equation (ODE), implicate ODE, Lorenz problem, Van der Pol Equation, SIR Model.

AMS Subject Classification: 65C05, 78M31.

1. Introduction

Differential equations play a prominent role in engineering, physics, economics, and other disciplines. An ordinary differential equation (ODE) is a differential equation in which the unknown variable is a function of a single independent variable. The traditional methods used to solve Initial Value Problem (IVP) ODEs are Euler's method, backward Euler's method, Runge-Kutta (RK) methods, multi-step method, and multi-value methods [6] etc. Although these methods can get different variation in their results, they are based on classical mathematical theories.

Traditionally Monte Carlo (MC) methods have been used to solve partial differential equations (PDEs) but the idea to solve the ODEs was suggested by Wei Zhong and Zhou Tian [12]. The idea presented in this paper would have been a great theoretical break through if it had worked efficiently with lower computational complexity. Unfortunately their method had serious limitations to be presented in next section. Similar ideas are also available in literature but they have not been documented as a paper, at least according to our information. A possible reason may be higher associated computational cost.

In this research paper an effort had made to present a more accurate and elaborate general MC algorithm to solve ODEs whereas the algorithm in [12] lacks such ability. The proposed algorithm was applied to various types of system of ODEs; the results obtained were considerably accurate.

In order to explain the algorithm and its results, the paper is divided into following sections. In section 2, related work is discussed and in section 3 the extended concepts of Monte Carlo integration to be used in next section are elaborated. These concepts are usually not discussed in elementary texts. In section 5 Computational analysis of the proposed MC algorithm has been carried out to show that an efficient algorithm is still awaited in scientific community. Finally in section 6, the paper has been concluded.

2. Related work

While writing this paper we were lead by eye catcher idea presented by Wei Zhong and Zhou Tian [12]. Last year an attempt was made to solve SIR epidemic model using their idea, but it was in vain to catch their thought. It is believed the method suggested by Wei Zhong and Zhou Tian [12] has serious limitations as below:

1. The output results are always zero, when the initial condition vector or resultant vectors of any intermediate iteration step are zero, it is clear from next iteration output equation as below.
$$\vec{Y}(j) = \vec{Y}(j-1) \times \left(1 - \frac{S}{N}\right). \tag{1}$$
2. Another problem with above relation is, the output results are always zero when $S = N$ which is valid situation, the results shouldn't be zero.
3. The value of judgment factor used in [2] is given by equation below:
$$k = \frac{f\left(\vec{X}(j-i), \vec{Y}(j-i)\right)}{\vec{Y}(j-i)} \times \Delta\vec{x}. \tag{2}$$
It has two serious problems as:
   a. If $\vec{Y}(j-i)$ becomes zero, the value of k is undefined.
   b. If $\vec{Y}(j-i)$ is very small, the value of k needs to be guaranteed less than 1 which requires resizing the value of $\Delta\vec{x}$. It may be an additional computational overhead.

The algorithm suggested in this paper overcomes all these difficulties associated with [12]. The output results are only zero when in fact solution is zero; judgment factor is never undefined and doesn't require resizing in all iteration steps.

In this paper the methods to generate random numbers haven't been discussed, the interested readers can refer to [3, 5]. It has been observed that the generation methods can only affect the precision of the results; those should not have significant impact on accuracy as it is usually true for all MC methods. It may be

noted that, for all examples discussed in this paper uniform random generator has been used.

In the next section the fundamental concepts of MC integration upon which the method for solving ODEs is based, has been reviewed. This however differs from many text books, which only discuss the cases, where the function values are nonnegative [10]. Probably the same concepts has been used by [12], however they were unable to describe it explicitly.

3. Monte Carlo Integration

Consider a function to be integrated as shown in Figure 1.

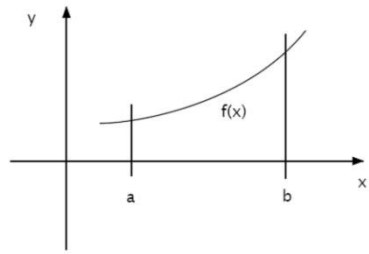

Figure 1. A simple function to be integrated

The integral is just the area under the curve. The width of the interval $(b-a)$ times the average value of the function is also the value of the integral, that is:

$$I=\int_a^b f(x)dx=(b-a)f_{average}=(b-a)\langle f\rangle. \tag{3}$$

So if we had some independent way of calculating the average value of the integrand, then the integral could be evaluated. That is where the random numbers can be used. Imagine that we have a list of random numbers, x_i uniformly distributed between a and b. To calculate the function average, we simply evaluate $f(x)$ at each of the randomly selected points, and divide by the number of points:

$$\langle f\rangle_N=\frac{1}{N}\sum_{i=1}^{N}f(x_i). \tag{4}$$

As the number of points used in calculating the average increases, $\langle f\rangle_N$ approaches the true average value $\langle f\rangle$. Therefore, as a numerical approximation could be written as:

$$\int_a^b f(x)dx=\frac{(b-a)}{N}\sum_{i=1}^{N}f(x_i). \tag{5}$$

Alternatively, we can look at this so-called Monte Carlo integration method in the following way:

To integrate the function $f(x)$ over the interval $[a,b]$ we can:

1. Find some value M such that $f(x) < M$ over the interval $[a,b]$
2. Select a random number x from a uniform distribution over the interval $[a,b]$
3. Select a random number y from a uniform distribution over the interval $[0,M]$
4. Determine if $y > f(x)$ or $y \le f(x)$
5. Repeat this process N times, keeping track of the number of times $y \le f(x)$ or under the curve (= successes); call the total number of successes S.

$$\frac{S}{N} = \frac{\text{Area under curve}}{\text{Total area inside rectangle}} = \frac{\int_a^b f(x)dx}{M(b-a)}. \tag{6}$$

The rectangle mentioned in above equitation is shown in Figure 2.

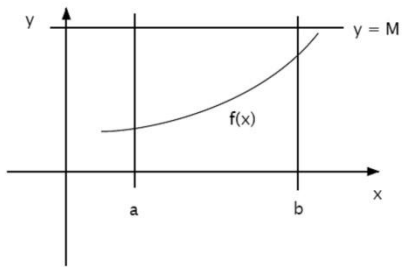

Figure 2. A bounded function below M

After a number of trials, the value of the integral could be calculated from the above formula

$$\int_a^b f(x)dx = M(b-a)\frac{S}{N}. \tag{7}$$

Think about throwing darts and counting the number of darts that land in the area representing the integral. The above method will only works if everywhere over the range of integration the integrand is greater than or equal to zero. Suppose, in fact, that the function $f(x)$ was not always greater than zero in the interval $[a,b]$ as shown in Figure 3. The Monte Carlo integration method can be modified to handle such cases, i.e., fix the problem with $f(x)$ possibly being less than zero as follows.

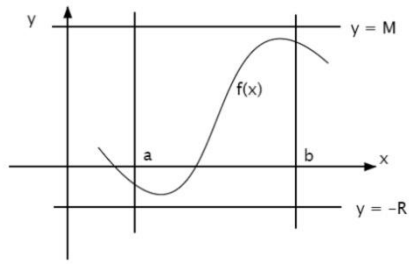

Figure 3. A function bounded below and above

To integrate the function $f(x)$ over the interval $[a,b]$ we can:

1. Find some value M such that $f(x) < M$ over the interval $[a,b]$
2. Find some R such that $f(x) > -R$ over the interval $[a,b]$
3. Select a random number x from a uniform distribution over the interval $[a,b]$
4. Select a random number y from a uniform distribution over the interval $[-R,M]$
5. Determine if $y > f(x)$ or $y \le f(x)$
6. Repeat this process N times, keeping track of the number of times $y < f(x)$ or under the curve (= successes); call the total number of successes S.

The estimated probability of success is then

$$\frac{S}{N} = \frac{\text{Area under curve}}{\text{Total area inside rectangle}} = \frac{\int_a^b f(x)}{(M+R)(b-a)}$$

$$\int_a^b f(x)dx = (M+R)(b-a)\frac{S}{N}. \tag{8}$$

This must now be corrected for the fact that the line $y = -R$ has been used as the baseline for the integral instead of the line $y = 0$. This is accomplished by subtracting the rectangular area $R(b-a)$. The final integral is then:

$$\int_a^b f(x)dx = (M+R)(b-a)\frac{S}{N} - R(b-a). \tag{9}$$

According to our information most of the text books discussing Monte Carlo integration methods talk about the former cases where integrand function has positive upper bound, the latter case has not been discussed explicitly. However while implementing (9) it was observed that the combined use of M and R affected the precision of the final results. So for the negative bound we just use (7) as:

$$\int_a^b f(x)dx = -R(b-a)\frac{S}{N}. \tag{10}$$

In these simulations the random number were just scaled in $[-R,0]$. The advantage of using (7) and (10) separately and then adding the results improves the precision of final result.

4. Numerical Examples

In the following section different types of ODEs including simple, explicit coupled, implicit and coupled system of first order ODEs has been solved. While solving these examples a typical value of step size = 0.001 and the random number count = 100,000 were used.

4.1. Simple ODEs

We know how to solve integration problems using MC methods [8]. We desire to be able to solve the differential equation which is same as estimate functionals of the function that solves a given equation. Traditional solution is to convert them into integral equations and apply the MC integration rules to them. Consider a simple example to find the value of $f(4)$, given the differential equation and boundary condition:

$$\frac{d}{dx} f(x) = 2e^x,\ f(0) = 4. \tag{11}$$

It can be integrated from 0 (*the known value*) to the desired value to get:

$$\int_0^4 \frac{d}{dx} f(x)dx = \int_0^4 2e^x dx$$

$$f(4) - f(0) = 2\int_0^4 e^x dx$$

$$f(4) = 4 + 2\int_0^4 e^x dx. \tag{12}$$

The definite integral on right side of (12) may be evaluated using any crude or improved MC integration schemes with suitable probability distributions functions (PDFs) [3, 5].

4.2. Explicit Coupled ODEs

In case of coupled equation set, the principles are same, but more care is needed in these:

1. Putting in multiple boundary conditions.
2. Keeping up with multiple sampled variables (each equation will have one)
3. Most tricky is realizing and adapting to changing limits on the integrals (after the first).
4. Much more difficult to optimize the choice of the probability PDFs used.

Consider another example of second order differential equation to find the value of $f(2)$, given the differential equation and boundary condition:

$$\frac{d^2}{dx^2} f(x) = x^4,\ f(0) = 1,\ f'(0) = 2. \tag{13}$$

In order to make it fit the category, this equation can be re-written as the linked set:

$$\frac{d}{dx} g(x) = x^4,\ g(0) = 2\,. \tag{14}$$

$$\frac{d}{dx} f(x) = g(x),\ f(0) = 1\,. \tag{15}$$

Applying the method to the (15) first, it is transformed it into an integral equation for the value at $x = 2$:

$$f(2) = 1 + \int_0^2 g(x)dx\,. \tag{16}$$

Using MC integration approximation, we get:

$$f(2) = 1 + \frac{g(\hat{x})}{\pi(\hat{x})} \equiv w_f \quad where\ \ 0 < \hat{x} < 2\,. \tag{17}$$

How to get the $g(\hat{x})$? The answer is, it can be estimated from the other equation (14). Applying this method to the equation (14), first, it is transformed it into an integral equation for the value at $x = \hat{x}$

$$g(\hat{x}) = 2 + \int_0^{\hat{x}} u^4 du \cong 2 + \frac{\hat{u}^4}{\pi(\hat{u})} \equiv w_g \quad where\ \ 0 < \hat{u} < \hat{x}\,. \tag{18}$$

The resulting procedure is as:

1. Choose a value of $\hat{x} \in (0,4)$ using $\pi_{\hat{x}}(\hat{x})$
2. Choose a value of $\hat{u} \in (0,\hat{x})$ using $\pi_{\hat{u}}(\hat{u})$
3. Score:

$$f(2) \cong w_f \equiv 1 + \frac{2 + \dfrac{\hat{u}^4}{\pi(\hat{u})}}{\pi(\hat{x})} = 1 + \frac{w_g}{\pi(\hat{x})}\,. \tag{19}$$

Equation (13) was solved using the above procedure and the results are given in Table 1. The exact value of $f(2) = 7.13333$ is up to five decimal places. From the Table 1 it can be observed that the accuracy of the results improves with increase in the count of random numbers (N) used in simulation.

Table 1. Results of various trials for equation (13)

N	**100**	**500**	**1000**	**10000**
$f(2)$ - Trial 1	6.9949	7.2939	7.2440	7.1278
$f(2)$ - Trial 2	7.2821	7.2179	6.9121	7.1201
$f(2)$ - Trial 3	6.5576	6.8734	7.2528	7.1490
Average	6.9448	7.1284	7.1363	7.1323

4.3. Implicit ODEs

In the following section the proposed algorithm to solve implicit function ODEs has been presented, which are generally solved by numerical techniques, and then its implementation for various types of ODEs has been discussed.

4.3.1. Monte Carlo based proposed algorithm

The major problem in case of implicit functions is initially estimating the value of M and R. Here a general algorithm for implicit ODEs is proposed, then it is suggested how to predict initial values of M and R.

Algorithm 1 Gillespie's direct method

1: Initialize S, I, R and set $t \leftarrow 0$, $t_f \leftarrow$ target time.
2: $a_1 \leftarrow \alpha SI$
3: $a_2 \leftarrow \beta I$
4: $a_0 \leftarrow a_1 + a_2$
5: **while** $t \leq t_f$ and ($S > 0$ or $I > 0$) **do**
6: Generate two uniform random numbers U_1 and U_2.
7: $\Delta t \leftarrow -ln(\frac{U_1}{a_0})$
8: $t \leftarrow t + \Delta t$
9: **if** $U_2 < \frac{a_1}{a_0}$ **then**
10: $S \leftarrow S - 1$
11: $I \leftarrow I + 1$
12: **else**
13: $I \leftarrow I - 1$
14: $R \leftarrow R + 1$
15: **end if**
16: **end while**

Depending upon the initial conditions of ODE the values of M and R needs to be justified accordingly. If values of M and R for $F(X,Y)$ are not known then these values can be predicted by initializing M and R to zero, executing the Algorithm 2, and between lines 6 and 7 setting the values of M and R equal to the maximum and minimum value of $JF(judgement\ factor)$ respectively.

4.3.2. Single Implicit ODEs

In this section three exemplary first order implicit ODES along with their analytical solution has been discussed.

Algorithm 2 Proposed algorithm for solving ODEs

1: Consider the differential equations in form of vector $\frac{dY}{dX}$ as multi-variable function $F(X,Y)$.
2: Find M and R for the $F(X,Y)$.
3: Generate two group of random numbers, 1^{st} Group ranging [0 **to** M] and 2^{nd} [$-R$ **to** 0] each having N numbers.
4: Initialize $X \leftarrow X_0$, $Y \leftarrow Y_0$, $X_f \leftarrow$ Final limit of integration
5: **while** $X \leq X_f$ **do**
6: **JF** $\Leftarrow F(X,Y)$
7: **if JF** ≥ 0 **then**
8: **S** $\Leftarrow$ Count of random numbers $<$ **JF**
9: $Y_{k+1} \Leftarrow Y_k + M\frac{\mathbf{S}}{N}\Delta X$
10: **else**
11: **S** $\Leftarrow$ Count of random numbers $>$ **JF**
12: $Y_{k+1} \Leftarrow Y_k - R\frac{\mathbf{S}}{N}\Delta X$
13: **end if**
14: $X \Leftarrow X + \Delta X$
15: **end while**

Example 1: The example of first order implicit stiff ODE has been taken from [1] to demonstrate forward Euler's method as:

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}\ ; y(0) = 0\ . \tag{20}$$

The analytical solution of this equation is as given below:

$$y = 3 - 0.998e^{-1000t} - 2.002e^{-t}\ . \tag{21}$$

Monte Carlo based proposed algorithm was applied to equation (20) and the results are shown in Figure 4 and listed in Table 2.

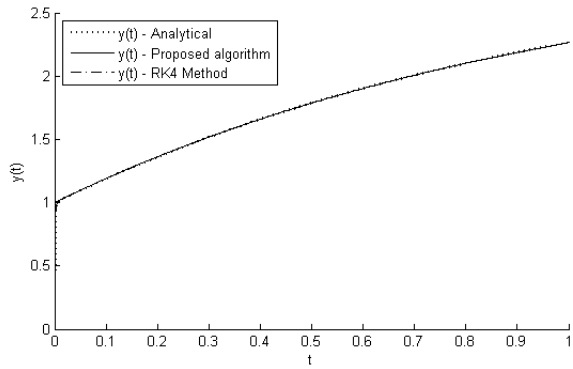

Figure 4. Comparison of numerical and exact solutions of equation (20)

From Figure 4 and the Table 2 It is clear that the results are very consistent with analytical solution. While close view of error in Figure 5 indicates that results having error of order 10^{-4} is acceptable to some extent but still RK4 method is more accurate.

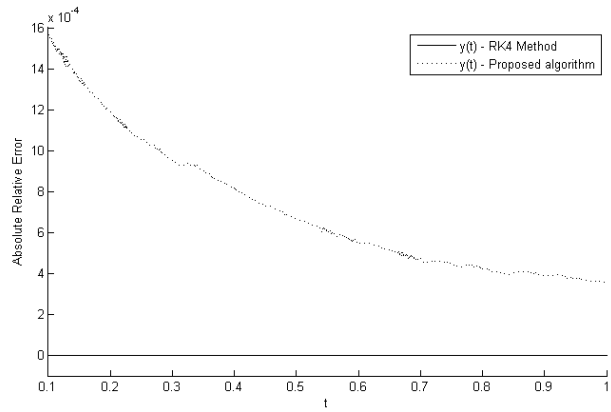

Figure 5. Error comparison of proposed and RK4 methods for equation (20)

Table 2. Results of two sample trials for equation (20)

t	0.0	0.2	0.4	0.8	1.0
Analytical Solution Results					
$y(t)$	0	1.3609	1.6580	2.1004	2.2635
Proposed Algorithm Result after 1st attempt					
$y(t)$	0	1.3623	1.6592	2.1011	2.2639
Proposed Algorithm Result after 2nd attempt					
$y(t)$	0	1.3624	1.6593	2.1012	2.2642

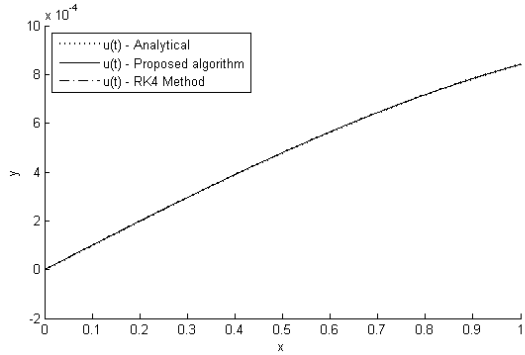

Figure 6. Comparison of numerical and exact solutions of equation (22)

Example 2: Another example of first order implicit stiff ODE has been taken from [11] to demonstrate the usefulness of the backward Euler's method as:

$$u' = -1000u + \sin(t)\,; u(0) = -1/100001\,. \tag{22}$$

This has a smooth solution as:

$$u = \frac{1000\sin(t) + \cos(t)}{1000001}\,. \tag{23}$$

Again Monte Carlo based proposed algorithm was applied to equation (22) and the results are shown in Figure 6 and listed in Table 3.

Table 3. Results of two sample trials for equation (22)

t	0.0	0.2	0.4	0.8	1.0
Analytical Solution Results					
$u(t)$	-1e-6	1.977e-4	3.885e-4	7.166e-4	8.409e-4
Proposed Algorithm Result after 1st attempt					
$u(t)$	-1e-6	1.986e-4	3.894e-4	7.173e-4	8.414e-4
Proposed Algorithm Result after 2nd attempt					
$u(t)$	-1e-6	1.986e-4	3.894e-4	7.173e-4	8.414e-4

From Figure 6 and the Table 3 it can be observed the results are very consistent with analytical solution.

Example 3: This is 3^{rd} example of single stiff ODE taken from [6] to illustrate both forward and backward Euler's method as:

$$y' = -100y + 100t + 101\,;\; y(0) = 1\,. \tag{24}$$

The analytical solution of this equation is as given below:

$$y(t) = 1 + t\,. \tag{25}$$

Again Monte Carlo based proposed algorithm was applied to equation (24) and the results are shown in Figure 7 and listed in Table 4.

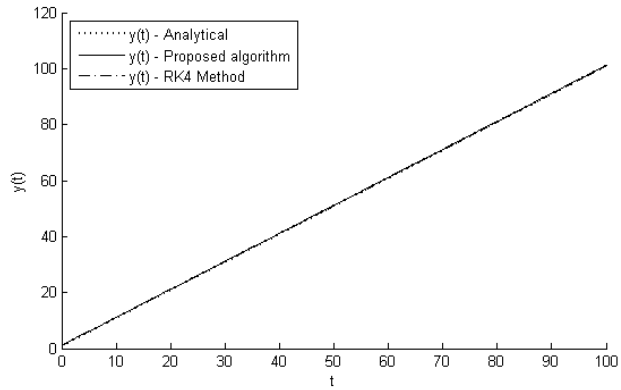

Figure 7. Comparison of numerical and exact solutions of equation (24)

From Figure 7 and the Table 4 it can be seen that results are very reliable as compared to analytical solution.

Table 4. Results of two sample trials for equation (24)

t	**0**	**20**	**40**	**80**	**100**
Analytical Solution Results					
$y(t)$	0	21	41	81	101.00
Proposed Algorithm Result after 1st attempt					
$y(t)$	0	21.01	41.01	81.01	101.01
Proposed Algorithm Result after 2nd attempt					
$y(t)$	0	21.03	41.03	81.03	101.03

Hence in this section it has been demonstrated that the proposed algorithm can equivalently be used instead of both forward and backward Euler's methods. So it depicts that the proposed algorithm does not suffer instability problem associated with forward Euler's method.

4.3.3. Two Coupled First Order ODEs

The example is a set of two coupled ODEs taken from [12] as below:

$$\frac{dy}{dx} = y + 2z\,;\, y(0) = 1.0\,. \tag{26}$$

$$\frac{dz}{dx} = 4y + 3z \; ; z(0) = 0.0 \; . \tag{27}$$

The analytical solutions are:

$$y(x) = \frac{1}{3} \times (e^{5x} + 2e^{-x}) \; . \tag{28}$$

$$z(x) = \frac{2}{3} \times (e^{5x} - e^{-x}) \; . \tag{29}$$

Monte Carlo based proposed algorithm was applied to equation (26) and (27) and the results are shown in Figure 8 and listed in Table 5.

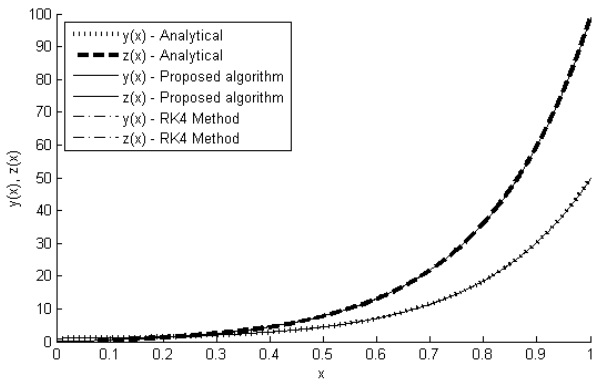

Figure 8. Comparison of numerical and exact solutions of equation (26) and (27)

Table 5. Results of two sample trials for equation (26) and (27)

x	**0.0**	**0.2**	**0.4**	**0.8**	**1.0**
Analytical Solution Results					
$y(x)$	1	1.4519	2.9099	18.4989	49.7163
$z(x)$	0	1.2664	4.4792	36.0992	98.6969
Proposed Algorithm Result after 1st attempt					
$y(x)$	1	1.4568	2.9321	18.7541	50.3846
$z(x)$	0	1.2789	4.5246	36.6118	99.0358
Proposed Algorithm Result after 2nd attempt					
$y(x)$	1	1.4614	2.9322	18.5024	49.0638
$z(x)$	0	1.2902	4.5294	36.1098	97.3981

From Figure 8 and the Table 5 it is clear that results are very reliable as compared to analytical solution.

4.3.4. Implicit 2nd order ODE

The example is taken from [2] with analytical solution provided as below:

$$yy'' - y'^2 = 0 \; ; y(0) = 1 \; , y'(0) = 2 \; . \tag{30}$$

The analytical solution is:

$$y(x) = e^{2x}. \tag{31}$$

Monte Carlo based proposed algorithm was applied to equation (30) and the results are shown in Figure 9 and listed in Table 6.

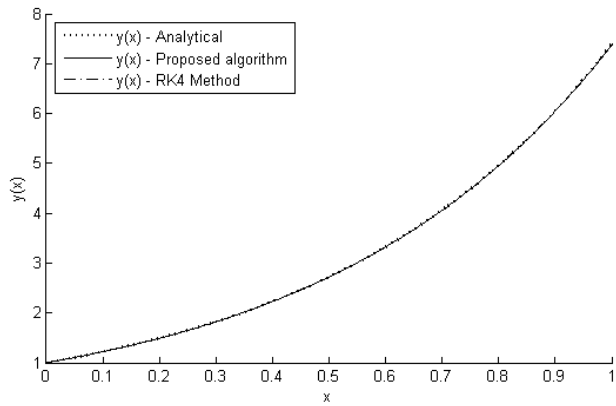

Figure 9. Comparison of numerical and analytical solutions of equation (30)

Table 6. Results of two sample trials for equation (30)

x	**0**	**0.2**	**0.4**	**0.8**	**1.0**
Analytical Solution Results					
$y(x)$	1	1.4918	2.2255	4.9530	7.3891
Proposed Algorithm Result after 1st attempt					
$y(x)$	1	1.4884	2.2169	4.9223	7.3425
Proposed Algorithm Result after 2nd attempt					
$y(x)$	1	1.4958	2.2314	4.9524	7.3831

From Figure 9 and the Table 6 above it can be observed that the results are very consistent with analytical solution.

4.3.5. Three Coupled first order ODEs

In this section two examples of three coupled first order ODEs along with their analytical or numerical solution are presented.

Example 1: The first example has been taken from [4], is for solving the homogeneous, linear systems with constant coefficients. This is an initial value problem defined as:

$$\begin{aligned} x_1' &= 8x_1 - 5x_2 + 10x_3 \quad ; x_1(0) = 2 \\ x_2' &= 2x_1 + x_2 + 2x_3 \quad ; x_2(0) = 2 \\ x_3' &= -4x_1 + 4x_2 - 6x_3 \quad ; x_3(0) = 2. \end{aligned} \tag{32}$$

The analytical solutions are:

$$x_1(t) = 6e^{-2t} - 4e^{3t}$$
$$x_2(t) = 6e^{2t} - 4e^{3t} \quad (33)$$
$$x_3(t) = -6e^{-2t} + 3e^{2t}.$$

Monte Carlo based proposed algorithm was applied to set of equations (32) and the results are shown in Figure 10 and listed in Table 7.

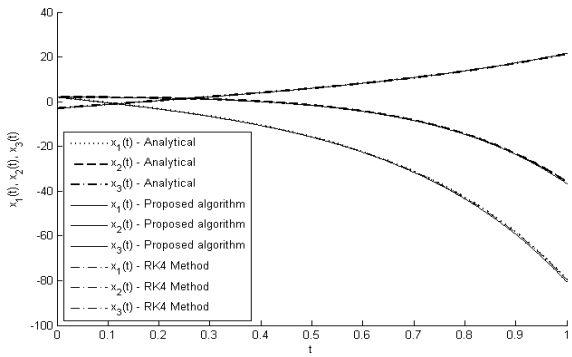

Figure 10. Comparison of numerical and exact solutions of equation (32)

From Figure 10 and the Table 7 it can be seen that the results are very reliable as compared to analytical solution. While having a precise view in Figure 11 we see that results are acceptable to some extent but still RK4 method is more accurate.

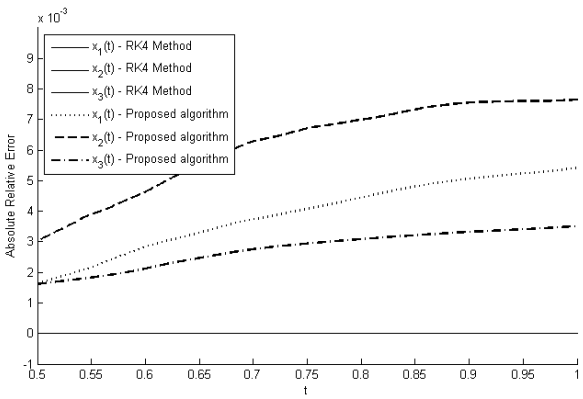

Figure 11. Error comparison of proposed and RK4 methods for equation (32)

Table 7. Results of two sample trials for equation (32)

t	**0.0**	**0.2**	**0.4**	**0.8**	**1.0**
Analytical Solution Results					
$x_1(t)$	2	-3.2666	-10.5845	-42.8813	-79.5301
$x_2(t)$	2	1.6625	0.0728	-14.3745	-36.0078
$x_3(t)$	-3	0.4536	3.9806	13.6477	21.3552

Proposed Algorithm Result after 1st attempt					
$x_1(t)$	2	-3.2708	-10.5806	-42.9768	-79.6706
$x_2(t)$	2	1.6563	0.0643	-14.4306	-36.0964
$x_3(t)$	-3	0.4624	3.9870	13.6640	21.3798
Proposed Algorithm Result after 2nd attempt					
$x_1(t)$	2	3.2786	-10.6364	-43.1725	-79.9850
$x_2(t)$	2	1.6559	0.0473	-14.5422	-36.2975
$x_3(t)$	-3	0.4571	3.9873	13.7005	21.4350

Example 2: The second example has been taken from [9], explaining the reliable Rosenbrock methods used to solve a stiff ODE system. This is an initial value problem defined as:

$$\begin{aligned} y_1' &= -0.013y_1 - 1000x_1y_2 && ; y_1(0) = 1 \\ y_2' &= -2500y_2y_3 && ; y_2(0) = 1 \\ y_3' &= -0.013y_1 - 1000y_1y_3 - 2500y_2y_3 && ; y_3(0) = 2. \end{aligned} \tag{34}$$

No analytical solutions were available, the built in routine ode45 (4th, 5th order RK methods) available from MATLAB was used for comparison. Monte Carlo based proposed algorithm was applied to set of equations (34) and the results are shown in Figure 12 and listed in Table 8.

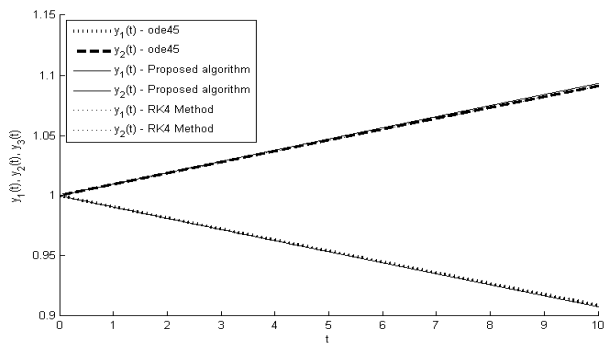

Figure 12. Comparison of numerical solutions of equation (34)

Table 8. Results of two sample trials for equation (34)

t	**0**	**2**	**4**	**8**	**10**
Analytical Solution Results					
$y_1(t)$	1	0.9815	0.9632	0.9270	0.9092
$y_2(t)$	1	1.0185	1.0368	-14.3745	1.0908
$y_3(t)$	0	-3.86e-6	-4.16e-6	-3.29e-6	-2.71e-6
Proposed Algorithm Result after 1st attempt					
$y_1(t)$	1	0.9815	0.9631	0.9270	0.9093
$y_2(t)$	1	1.0186	1.0370	1.0732	1.0911
$y_3(t)$	0	-3.61e-6	-3.52e-6	-3.34e-6	-3.25e-6

Proposed Algorithm Result after 2nd attempt					
$y_1(t)$	1	0.9815	0.9633	0.9272	0.9095
$y_2(t)$	1	1.0185	1.0368	1.0728	1.093
$y_3(t)$	0	-3.63e-6	-3.53e-6	-3.35e-6	-3.26e-6

From Figure 12 and the Table 8 it can be seen that the results are very reliable as compared to ode45 solution

4.3.6. Practical Systems

In this section three exemplary practical systems involving along with their numerical solution has been presented.

Van der Pol Equation: The Van der Pol equation is a model of an electronic circuit that arose back in the days of vacuum tubes [1].

$$y'' + \mu(y^2 - 1)y' + y = 0\,;\mu = 1\,,\, y(0) = 2\,,\, y'(0) = 1\,. \tag{35}$$

Monte Carlo based proposed algorithm was applied to equation (35) with given initial conditions and the results are depicted in Figure 13, and listed in Table 9.

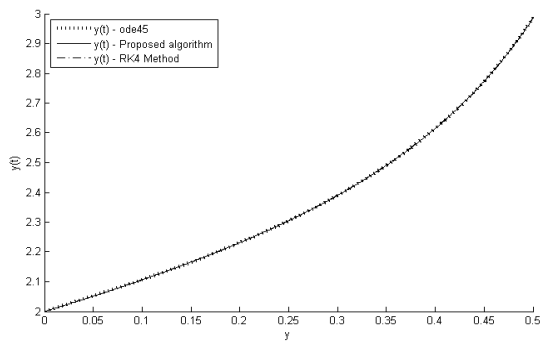

Figure 13. Comparison of numerical solutions of equation (35)

Table 9. Results of two sample trials for equation (35)

t	**0.0**	**0.1**	**0.2**	**0.4**	**0.5**
ODE45 Results					
$y(t)$	2	2.1061	2.2308	2.6138	2.9854
Proposed Algorithm Result after 1st attempt					
$y(t)$	2	2.1065	2.2315	2.6137	2.9831
Proposed Algorithm Result after 2nd attempt					
$y(t)$	2	2.1062	2.2309	2.6153	2.9865

From Figure 13 and the Table 9 above it can be observed that the results are very reliable as compared to ode45 solution.

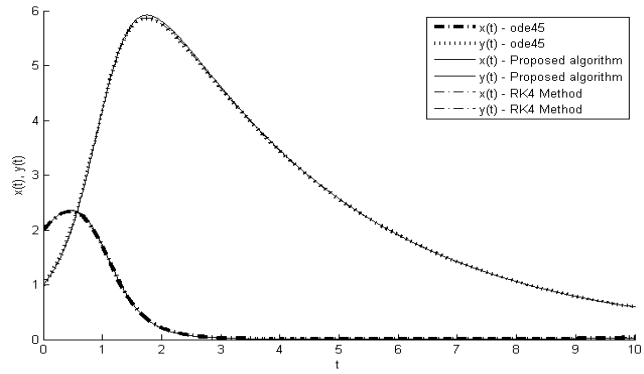

Figure 14. Comparison of numerical solutions of equation (36)

Predator – prey model: Consider the Lotka-Volterra predator-prey [1] with initial conditions as below:

$$\begin{aligned} x'(t) &= ax - bxy \quad ; a = 1.2 \quad , b = 0.6 \quad , x(0) = 2 \\ y'(t) &= -cy + dxy \; ; c = 0.8 \quad , d = 0.3 \quad , y(0) = 1. \end{aligned} \tag{36}$$

Here $x(t)$ and $y(t)$ are the prey and predator population sizes respectively at time t, and a, b, c, d are biologically determined parameters. Monte Carlo based proposed algorithm was applied to equation (36) with given initial conditions and the results are depicted in Figure 14, and listed in Table 10.

From Figure 14 and the Table 10 it is clear that the results are very reliable as compared to ode45 solution.

Table 10. Results of two sample trials for equation (36)

t	**0**	**2**	**4**	**8**	**10**
ODE45 Results					
$x(t)$	2	0.1828	0.0089	0.0098	0.0378
$y(t)$	1	5.7078	3.3796	1.0091	0.5964
Proposed Algorithm Result after 1st attempt					
$x(t)$	2	0.1826	0.0074	0.0081	0.0334
$y(t)$	1	5.7657	3.3868	1.0071	0.5884
Proposed Algorithm Result after 2nd attempt					
$x(t)$	2	0.1811	0.0088	0.0092	0.0337
$y(t)$	1	5.7624	3.4129	1.0181	0.6032

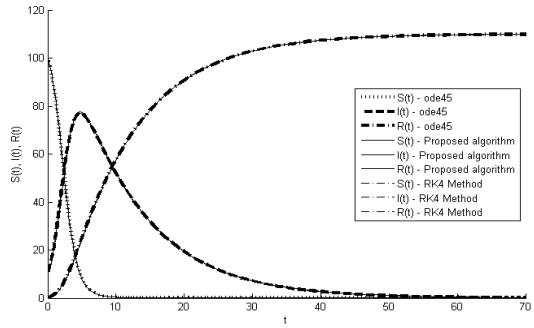

Figure 15. Comparison of numerical solutions of equation (37)

The SIR Epidemic model: The SIR model for epidemic dispersion [7] given by the following equations with initial conditions described as.

$$\begin{aligned} S'(t) &= -\beta SI + \gamma R \quad ; S(0) = 100 \\ I'(t) &= -\alpha I + \beta SI \quad ; I(0) = 10 \\ R'(t) &= -\gamma R + \alpha I \quad ; R(0) = 0 \\ \alpha &= 0.1, \beta = 0.01, \gamma = 0. \end{aligned} \tag{37}$$

Monte Carlo based proposed algorithm to equation (37) with given initial conditions and the results are depicted in Figure 15, and listed in Table 11.

Table 11. Results of two sample trials for equation (37)

t	**0**	**1**	**10**	**20**	**70**
ODE45 Results					
$S(t)$	100	84.496	0.307	0.012	0.002
$I(t)$	10	24.701	51.865	19.431	0.130
$R(t)$	0	1.803	57.827	90.557	109.868
Proposed Algorithm Result after 1st attempt					
$S(t)$	100	84.006	0.238	0.015	0.007
$I(t)$	10	22.498	51.986	19.211	0.137
$R(t)$	0	1.496	57.539	90.648	109.846
Proposed Algorithm Result after 2nd attempt					
$S(t)$	100	84.071	0.297	0.012	0.006
$I(t)$	10	24.208	52.468	19.519	0.158
$R(t)$	0	1.721	57.555	90.578	109.915

From Figure 15 and the Table 11 it can be seen that the results are very reliable as compared to ode45 solution.

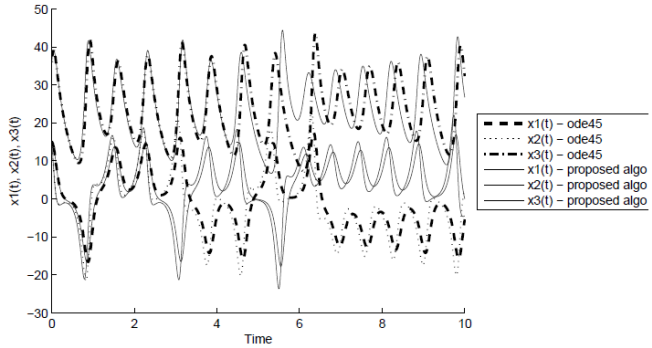

Figure 16. Comparison of numerical solutions of equation (38)

4.3.7. Unsolvable Implicit ODEs

In this section two exemplary problems are discussed.

Lorenz Problem: The ODEs of Lorenz problem used for weather prediction belong to a class referred to as chaotic; they produce wildly different results when their initial conditions are changed infinitesimally. In other words, accurate weather prediction depends crucially on the accuracy of the measurements of initial conditions. It is known to have solutions that are potentially poorly conditioned. The set of ODEs used in our simulation has been taken from [2] are as:

$$\begin{aligned} x_1' &= 10(x_2 - x_1) && ; x_1(0) = 15 \\ x_2' &= x_1(28 - x_1) - x_2 && ; x_2(0) = 15 \\ x_3' &= x_1 x_2 - \frac{8}{3} x_2 && ; x_3(0) = 36. \end{aligned} \tag{38}$$

Monte Carlo based proposed algorithm was applied to equation (38) with given initial conditions and the results are depicted in Figure 16, and listed in Table 12.

Table 12. Results of two sample trials for equation (38)

t	**0**	**2**	**4**	**8**	**10**
Analytical Solution Results					
$x_1(t)$	15	3.5803	-4.1650	-3.4768	-5.4116
$x_2(t)$	15	5.5524	0.2592	-4.0746	2.2911
$x_3(t)$	36	15.5830	28.3352	19.7510	32.3058
Proposed Algorithm Result after 1st attempt					
$x_1(t)$	15	2.9842	8.4022	-1.6157	-11.8506
$x_2(t)$	15	4.7407	3.4533	-2.8473	-8.1210
$x_3(t)$	36	14.9478	32.3746	17.7672	35.1122
Proposed Algorithm Result after 2nd attempt					
$x_1(t)$	15	3.1827	4.3548	7.2789	0.0427
$x_2(t)$	15	5.1893	0.5714	11.3518	-4.4594
$x_3(t)$	36	14.2940	27.7014	17.9530	26.7454

From Figure 16, it can be observed that the results are unreliable as compared to ode45 solution because the proposed algorithm behaves like Euler's method (similarities to be discussed in preceding section) with constant slope diverging away from the solution. The Table 12 lists two sample trials for set of equations (38) which illustrate the inaccuracy of results we obtain.

Lipschitz Continuity Check: Here an exemplary problem has been discussed to check the lipschitz continuity where the proposed algorithm has been used solve a very simple ODE as:

$$y' = y^2 \; ; \; y(0) = 1 \quad \forall \; t \; in \; [0, 2] \, . \tag{39}$$

The analytical solution is:

$$y(t) = \frac{1}{1-t} \, . \tag{40}$$

This solution goes to infinity as t approaches 1 as shown in Figure 17. Let us consider the solution in the interval $0 \le t \le 2, 10 \le y \le 10$.It can be seen that Lipschitz constant=20, but solution is not guaranteed on the entire interval $[0, 2]$. Furthermore Euler's Method and the proposed algorithms have been applied to equation (39), the results are shown in Figure 17,

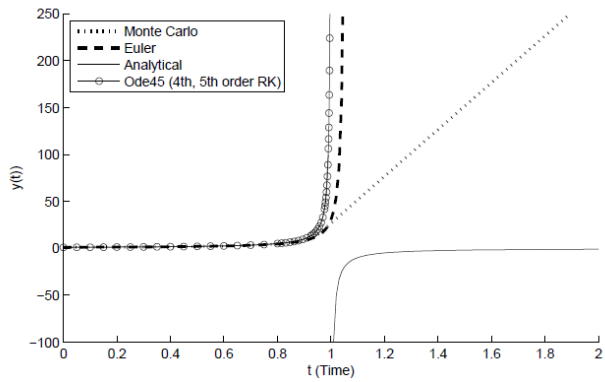

Figure 17. Comparison of numerical and analytical solutions of equation (39)

It can be observed soon the Euler's method blows as function slope increases rapidly at $t = 1$. But the proposed algorithm is not affected because it didn't detect the singularity at $t = 1$, which is a usual attribute of MC methods. The ratio of S to N has been shown in Figure 18.

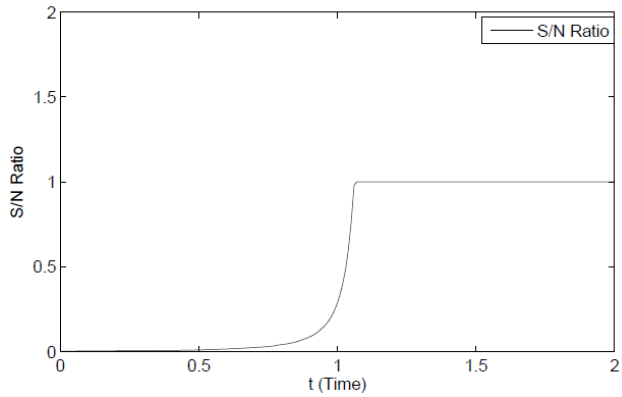

Figure 18. Plot of S/N ration in the interval $0 \le t \le 2$

We observe at time $t = 1, S = N,$ the next iteration solution Y_{k+1} becomes.

$$Y_{k+1} = Y_k + M\Delta X \ . \tag{41}$$

The function values increase with M being a constant slope. This trend is very clear from Figure 18, again it a characteristic of MC method, it is predicting a solution even if it inaccurate.

5. Computational analysis

As stated earlier while trying to implement [12] for SIR model, the associated problems were rectified giving rise to a theoretically sound algorithm 2, however it needs an in-depth analysis of the proposed algorithm. The proposed relations for the output are:

$$Y_{k+1} = Y_k + M\frac{\mathbf{S}}{N}\Delta X \ . \tag{42}$$

or

$$Y_{k+1} = Y_k - R\frac{\mathbf{S}}{N}\Delta X \ . \tag{43}$$

However if values of M and R for $F(X,Y)$ are not known then these values can be predicted as stated in section 4.3.1. Now consider the standard Euler's method relation:

$$Y_{k+1} = Y_k + F(X,Y)\times\Delta X \ . \tag{44}$$

As in Euler's method the function $F(X,Y)$ is replaced with $M\frac{\mathbf{S}}{N}$ *or* $-R\frac{\mathbf{S}}{N}$.Now if $S = N$ then $F(X,Y) = M$ *or* $-R$. Hence the algorithm is kind of Euler's method approximation where M or R are constants is substituted for slope of the function.

Furthermore computational aspects of the proposed algorithm have been analyzed by comparing it to some other standard ODE solvers. The proposed

algorithm has been analyzed using a number of set of ODEs; however the results presented here are for the system of ODEs given by equation (26) and (27), as given in [12] :

5.1. Variation of Random Number Count (N)

In all Monte Carlo simulation it is traditional to observe the effect of change in N, the following subsection presents its impact on computational time and mean square error. Here the step size has been fixed at 0.001.

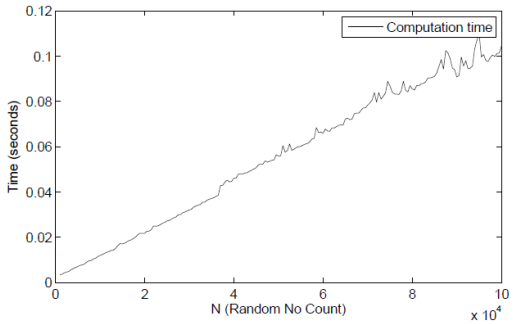

Figure 19. Random number count versus computational time

5.1.1. Computational time

Here in the following diagram the quantity of random numbers used has been varied and the execution time of the algorithm has been measured, the results are shown in Figure 19. It is clear as the random number count used in simulation is increased; the computational time is also increased.

5.1.2. Mean Square Error

Here in the following diagram the quantity of random numbers used has been varied and the mean square error has been computed for the proposed algorithm, the results are shown in Figure 20.

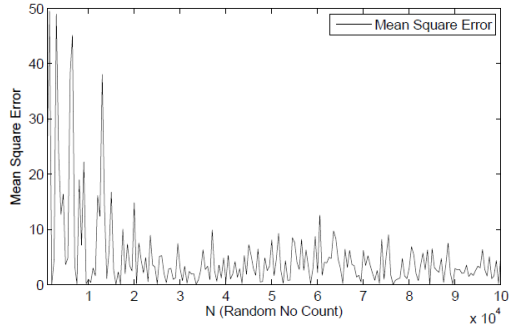

Figure 20. Random number count versus mean square error

It is clear as the random number count used in simulation is increased the mean square is decreased. This is also a normal trait of all MC methods.

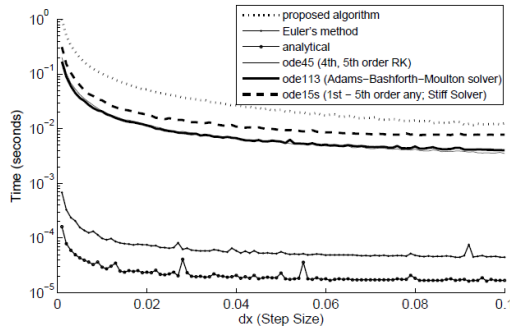

Figure 21. Step Size versus computational time

5.2. Change of step size

The step size is very important parameter for measuring the performance of ODEs algorithm. Smaller step size means slower computation, but perhaps the right precision and vice versa. In the following section its effects on computational time and mean square have been analyzed. Here a fix value of N = 100,000 is used;

5.2.1. Computational time

Here in the following diagram the step sizes used have been varied and the execution time of various ODE algorithms has been measured, the results are shown in Figure 21.

It can be seen that large step size means faster computation, but perhaps not the right precision. However the proposed algorithms have much higher computational time than even the most basic Euler's Method algorithm.

5.2.2. Mean Square Error

Here in the following diagram the step sizes used have been varied and the mean square error has been computed for the proposed algorithm, the results are shown in Figure 22.

It is general observation that increasing the step size may in fact make errors worse. It can be seen mean square error of the proposed algorithm follows same pattern as basic Euler's method.

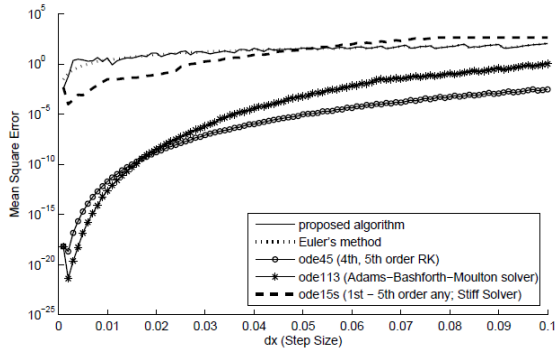

Figure 22. Step size versus mean square error

5.3. Accuracy and stability

Most of the ODEs solved in this paper are stiff usually difficult to solve by using forward Euler's method, in such cases backward Euler's method is used. Furthermore it was also compared to Rosenbrock method which is considered to more effective for stiff ODEs, where other solvers are not useful. The results had considerable accuracy as compared to Rosenbrock methods. It has been observed that the accuracy of the method depends upon the quantity of random numbers generated. The algorithms has always been found stable due to bounding value of M and R.

6. Conclusion

The idea presented in this paper for Monte Carlo based method to solve initial value problem of ODEs is mathematically based on natural extension of Monte Carlo integration. It is mathematically sound and lucid as compared to [12]. The scope of this paper is much wider as it discusses a large class of ODEs as compared to [12]. The limitation of the said algorithm has clearly been identified. The normalcies associated with proposed algorithm are as:

1. The algorithm is less sensitive to step size however same step size was used in all examples for comparison with analytical and numerical solutions.
2. Higher value of N will give higher accuracy and precision of results
3. In the last example of Lipschitz continuity check the two points can be concluded in this regard.
   a. The subtle peaks and troughs will be lost by using proposed algorithm.
   b. The algorithm never fails to predict solution, even the results are wrong.

   These are normal features of Monte Carlo based algorithms.
4. The algorithm is computationally more expensive than all other ODE solvers.

The implementation of this algorithm is computationally more expensive than other available ODE solvers, but still there are a few open questions listed as:

1. Would it be possible to use suggested algorithm to simulate the system of coupled ODEs in much higher dimensions, by using parallel random number generators?
2. In case of parallel implementation whether communication or computational factor will dominate, needs further investigation.

It may be noted that we come across the idea of this paper from simulation to theory, while implementing [12]. In future our intention is strive to find the answers to these open questions. In our opinion, at present computational community needs to wait for an efficient Monte Carlo algorithm for solving ODEs, which may be possible in near future with emerging processor architectures.

References

1. Chapra S.C., Applied Numerical Methods with MatLab for Engineers and Scientists, NY: McGraw-Hill, 2008, pp.322-329.
2. Cheney E., Kincaid D., Numerical mathematics and computing, Cengage Learing, 2012.
3. Dunn W.L., Shultis J.K., Exploring Monte Carlo Methods, Elsevier, 2011.
4. John P., Ordinary differential equations using MATLAB, Pearson Education India, 2009.
5. Kalos M.H., Whitlock P.A., Monte carlo methods, John Wiley & Sons, 2008.
6. Michael T.H., Scientific computing: an introductory survey, The McGraw Hill Companies Inc., New York, 2002.
7. Nagle R.K., Saff E.B., Snider A.D., Fundamentals of differential equations.: Pearson/Addison-Wesley, 2008.
8. Pevey D.R.E., Ronald E., Pevey Home Page. [cited 2012 August 2012]; Available from: http://web.utk.edu/~rpevey/.
9. Preiss W., et al., Numerical Recipes in C++; The Art of Scientific Computing., Cambridge University Press, Cambridge, 2002.
10. Random numbers and monte carlo methods. [cited 2006 July 2006]; Available from: http://chaos.swarthmore.edu/courses/Phys50L_2006 /Matlab/Lab_2_2006.pdf.
11. Sewell G., The numerical solution of ordinary and partial differential equations, Vol.75, John Wiley & Sons, 2005.
12. Zhong W., Z. Tian, Solving initial value problem of ordinary differential equations by Monte Carlo method in Multimedia Technology (ICMT), International Conference on. 2011. IEEE.

Başlanğıc sərhəd şərtli adi diferensial tənliyin Monte-Karlo üsulu ilə həlli

M.N. Akhtar, M.H. Durad, A. Ahmed

XÜLASƏ

Məqalədə adi diferensial tənliyin həlli üçün Monte-Karlo üsuluna əsaslanan alqoritm təklif edilir. Bu alqoritm birinci tərtib sadə, aşkar, qeyri-aşkar adi diferensial tənliklərə və tənliklər sisteminə tətbiq edilir. Sonra bu üsul Van der Pol və SİR epidemik və fiziki sistemlərə də tətbiq edilir. Göstərilir ki, Monte-Karlo üsulu ilə adi diferensial tənliklərin tələb edilən dəqiqliklə həlli ciddi çətinliklərlə bağlıdır.

Açar sözlər: Monte-Karlo inteqrallanması, adi diferensial tənliklər, Lorens problemi, Van der Pol tənliyi, SİR modeli.

Решение обыкновенных дифференциальных уравнений с начальными условиями методом Монте-Карло

М.Н. Актар, М.Х. Дурад, А. Ахмед

РЕЗЮМЕ

В статье предлагается алгоритм базирующийся на методе Монте-Карло для решения обыкновенных дифференциальных уравнений. Этот алгоритм применяется к решению обыкновенных дифференциальных уравнений первого порядка простого, явного и неявного типа и систем уравнений. Далее решается физическое уравнение Ван дер Пола и эпидемическое СИР уравнение. Показывается что, решение обыкновенных дифференциальных уравнений с требуемой точностью представляется достаточно трудным.

Ключевые слова: интегрирование Монте-Карло, обыкновенные дифференциальные уравнения, Проблема Лоренца, Уравнение Ван дер Пол, СИР уравнение.