Jurnal Teknologi

Full Paper

HISTORY TRACKING ABILITY OF HYBRID SECOND AND FOURTH ORDERS RUNGE-KUTTA IN SOLVING DELAY DIFFERENTIAL EQUATIONS

Rui Sih Lim, Rohanin Ahmad*, Su Hoe Yeak

Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia

*Corresponding author
rohanin@utm.my

Article history
Received
31 October 2016
Received in revised form
8 June 2017
Accepted
10 August 2017

Graphical abstract

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \\ & b_1^* & \cdots & b_s^* \end{array} = \begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array}$$

Abstract

This paper presents numerical solution for Delay Differential Equations systems to identify frequent discontinuities which occur after and sometimes before the initial solution. The Runge-Kutta methods have been chosen because they are well-established methods and can be modified to handle discontinuities by means of mapping of past values. The state system of the problem is first discretized before the method is applied to find the solution. Our objective is to develop a scheme for solving delay differential equations using hybrid second and fourth order of Runge-Kutta methods. The results have been compared with the result from Matlab routine dde23 which used second and third order of Runge-Kutta methods. Our numerical scheme is able to successfully handle discontinuities in the system and produces results with acceptable error.

Keywords: Delay Differential Equations, Discontinuities, Runge-Kutta method, Matlab routine dde23

Abstrak

Kertas kerja ini mengutarakan penyelesaian berangka untuk sistem Persamaan Terbitan Tertunda untuk mengatasi ketakselanjaran yang kerap berlaku sebelum dan kadangkala selepas penyelesaian awal. Kaedah Runge-Kutta telah dipilih kerana ia merupakan kaedah yang berkesan dan boleh diubah suai untuk menangani ketakselanjaran daripada pemetaan nilai-nilai yang lepas. Sistem keadaan untuk masalah ini telah dijadikan diskret terlebih dahulu sebelum keadah Runge-Kutta digunakan untuk mendapatkan penyelesaian. Objektif kami adalah membangunkan satu skim yang mampu untuk menyelesaikan persamaan terbitan tertunda dengan menggunakan Runge-Kutta peringkat kedua dan keempat terhibrid. Untuk penesahan, keputusan telah dibandingkan dengan Matlab routine dde23 yang telah siselesaikan menggunakan Runge-Kutta peringkat kedua dan ketiga. Skim berangka kami ini terbukti mampu menangani ketakselanjaran sistem dengan jayanya dan menghasilkan keputusan dengan ralat yang boleh diterima.

Kata kunci: Persamaan Pembezaan Tertunda, Ketakselanjaran, Kaedah Runge-Kutta, Matlab routine dde23

1.0 INTRODUCTION

Delay differential equations (DDEs) are used to describe many phenomena of physical interests subjected to time delay. There exist large classes of dynamics systems, among which some are very important. In the initial value problem of ordinary differential equations (ODEs), the state variable of the system is dependent solely on the current time. To make the model more consistent with the real system, it is necessary to modify this equation by including past values of state variables [1]. Here, we focus only on systems of delay differential equations in the form of

$$\begin{aligned} y'\ t\ &= f\ t, y\ t \\ y'\ t\ &= f\ t, y\ t\ , y\ t-\tau_1\ , \ldots, y\ t-\tau_i\ , \ldots \end{aligned} \tag{1}$$

where the quantities $\tau_i, i = 1,2,\ldots,m$ are positive constants and $y\ t\ = S(t)$ are given history/past values; the history/past values will make the model more consistent. Most delay differential equations are nonlinear systems, such as predator-prey systems [2], continuously-stirred tank reactors [3], tracking component-wise continuity [4] and infectious disease model from [5], which are the focus of this study.

This paper presents the use of the fourth-order Runge-Kutta (RK4) method for solving the DDEs. We developed a method based on RK4 that is able to generate acceptable output. For validation, this output had been compared to numerical results obtained from dde23 in Matlab, based on second and third orders Runge-Kutta; we expect the values of our output naturally should be as accurate as the output of dde23, if not better.

2.0 METHODOLOGY

In deterministic setting, the most suitable form of mathematical model used to describe a real-life phenomenon whose evolution over time is administrated by the history of the state variables is by constructing formulation by using DDEs.

By considering the following general form of DDE with time lag $\tau > 0$

$$\begin{aligned} y'\ t\ &= f\ t, y\ t\ , y(t-\tau)\ ,\ t \in [-\tau, T] \\ y(t)\ &= S(t) \qquad\qquad\qquad ,\ t \in [-\tau, 0] \end{aligned} \tag{2}$$

where $S(t)$ corresponds to the initial function of the interval $-\tau \le t \le 0$ with time delay τ as positive constants delay or time-dependent delay, $\tau(t)$. Here, only the constant delay shall be considered. The presence of the initial function $S(t)$ in system (2) may contribute to discontinuities in the derivatives and affect the numerical treatment. These discontinuities may be present at times both before and after the initial point. Some models have histories with discontinuities in low-order derivative. These changes often cause discontinuities in low-order derivatives of the solution. Discontinuities will cause inaccuracy and inefficiency of numerical methods in solving DDEs making the exact solution of DDEs hard to find. The Runge-Kutta methods are attractive because they can be easily modified to handle the discontinuities by using mappings and easier to apply than other popular numerical methods.

In this study, the hybrid second order Runge-Kutta (RK2) and fourth order Runge-Kutta (RK4) methods had been used to calculate the node-points. RK2 was used to solve the problem with step-size, $\frac{h}{2}$; whereas RK4 was used to solve the problem with step-size, h.

The following is the general RK4 method:

$$\begin{aligned} y(t+h) &\approx y(t) + \frac{1}{6}(F_1 + 2F_2 + 2F_3 + F_4) \\ F_1 &= hf\ t, y \\ F_2 &= hf\left(t + \frac{1}{2}, y + \frac{1}{2}F_1\right) \\ F_3 &= hf\left(t + \frac{1}{2}, y + \frac{1}{2}F_2\right) \\ F_4 &= hf\ t + h, y + F_3 \end{aligned} \tag{3}$$

where $y'\ t\ = f\ t, y\ t$ with h being the step size.

The RK2 method, or also called as Heun's method is given as follows:

$$\begin{aligned} y(t+h) &\approx y(t) + \frac{1}{2}(k_1 + k_2) \\ k_1 &= hf\ t, y \\ k_2 &= hf\ t + h, y + k_1 \end{aligned}$$

By applying $h = \frac{1}{2}h$ in RK2, the formulae become:

$$y(t+\frac{1}{2}h) \approx y(t)+\frac{1}{2}(k_1+k_2)$$
$$k_1 = \frac{1}{2}hf\ t,y$$
$$k_2 = \frac{1}{2}hf\left(t+\frac{1}{2}h, y+k_1\right)$$
$$k_2 = \frac{1}{2}hf\left(t+\frac{1}{2}h, y+\frac{1}{2}F_1\right)$$

Applying $hf\ t,y\ = F_1$ and $hf\left(t+\frac{1}{2}h, y+\frac{1}{2}F_1\right) = F_2$, produces

$$y(t+\frac{1}{2}h) \approx y(t)+\frac{1}{2}(\frac{1}{2}F_1+\frac{1}{2}F_2)$$
$$y(t+\frac{1}{2}h) \approx y(t)+\frac{1}{4}(F_1+F_2) \quad (4)$$

Thus, the buildup of error from calculation can be minimized, since there is no extra calculation in RK2, by using only the values calculated from RK4.

RK for DDEs Algorithm

Step 0: Set the history/past values and initial value which are given.

Step 1: Identify the time delays τ_i by using Greatest Common Divisor (GCD). The biggest time delay will be set as the "Step".

Step 2: Solve the discretized state system in Equation (2) by using Equations (3) and (4).

Step 3: Set $t = t+h$ and go to Step 2.

Figure 1 shows the flow chart of the algorithm.

The Runge-Kutta methods mentioned are all explicit recipes for computations of y_{n+1} given y_n and have the ability to evaluate the equation or system. For reasons of efficiency, the longest delay τ_i shall be set as "Step", and then h shall be set as "node"; hence this will yield the specified accuracy. Normally, the shortest delay τ_i will be set as h, all the needed values of the solution in the span of the step will be stored automatically, which are easy to recall when $t = t+h$ is updated. Some solvers tried to compute the solution at the end of the step, although the values were still not known. Furthermore, they simply used any step size that appeared to be appropriate and iterate to evaluate the implicit formula that arises. Our method avoids this and the problem of discontinuity.

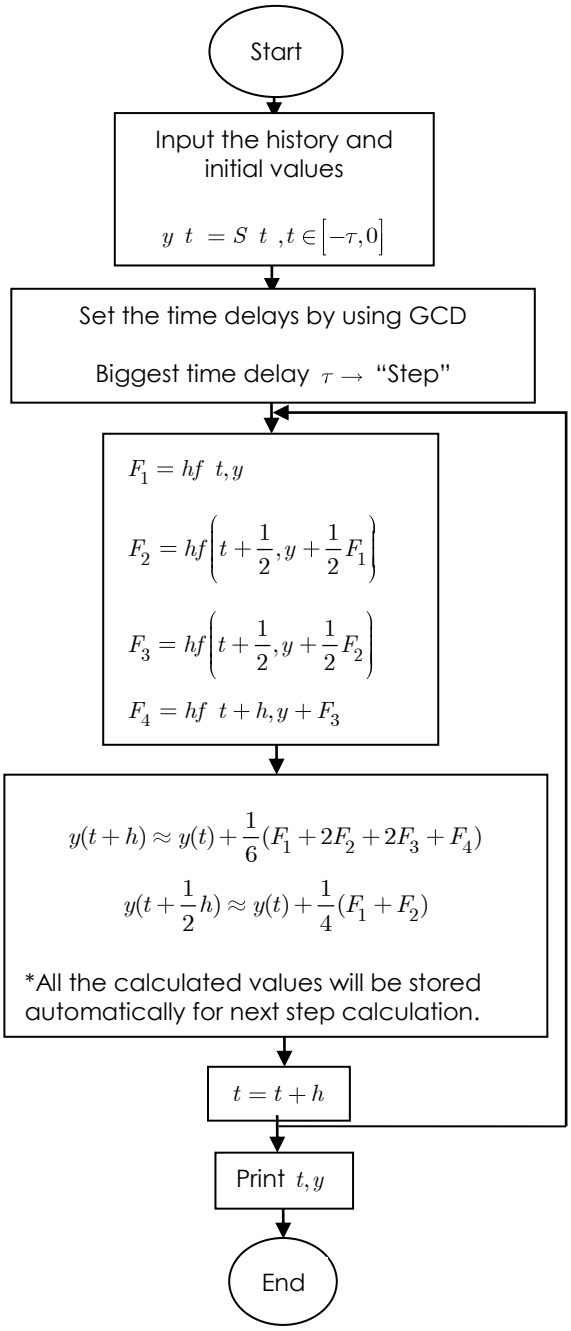

Figure 1 Flow Chart of RK for DDEs Algorithm

3.0 RESULTS AND DISCUSSION

For this study, problems from the literature had been adopted to show solution of DDEs with dde23 from Matlab and our method using hybrid RK2 and RK4. Consequently, their outputs were compared.

3.1 Example 1

For a start, Example 3 (Tracking component-wise solution continuity) from [4] had been used. The system is as follows:

$$y_1'\ t\ = y_1(t-1)$$
$$y_2'\ t\ = y_1\ t-1\ + y_2(t-0.2)$$
$$y_3'\ t\ = y_2(t)$$

to be solved over the interval [0,5] with history $y_1\ t\ = 1,\ y_2\ t\ = 1,\ y_3\ t\ = 1$ for $t \le 0$.

In the numerical computations, the following parameters were used: the number of steps, steps $= 0,\ 1,\ 2,\ 3$, where steps $= 0$ is history; the number of nodes, node $= 1,\ 2,\ 3,\ldots,\ 11$; $h =$ shortest time delay $\tau = 0.2$, and $\frac{h}{2} = 0.1$.

Table 1 Output Values of dde23

Time	y_1	y_2	y_3
0.0000	1.0000	1.0000	1.0000
0.2000	1.2000	1.4000	1.2400
0.4000	1.4000	1.8400	1.5627
0.6000	1.6000	2.3627	1.9815
0.8000	1.8000	2.9815	2.5141
1.0000	2.0000	3.7141	3.1816
⋮	⋮	⋮	⋮
4.2000	12.1810	85.5398	89.7943
4.4000	13.6441	102.6364	108.5615
4.6000	15.2830	123.0421	131.0693
4.8000	17.1187	147.3851	158.0406
5.0000	19.1749	176.4120	190.3353

Table 1 shows the output from dde23. The first column represents the time, the following columns are the output of y_1, y_2, and y_3 consecutively. This format is the same for all the tables in this paper. As shown in column one, the time had been increased by $h = 0.2$ and the increment of time was linear; no continuity occurred hence dde23 was indeed able to produce the solution. Next, the same linear problem was solved by using our numerical scheme. The results are tabulated in Table 2.

Table 2 Output Values of hybrid RK2 and RK4

Time	y_1	y_2	y_3
0.0000	1.0000	1.0000	1.0000
0.2000	1.2000	1.4000	1.2400
0.4000	1.4000	1.8400	1.5627
0.6000	1.6000	2.3627	1.9815
0.8000	1.8000	2.9815	2.5141
1.0000	2.0000	3.7142	3.1816
⋮	⋮	⋮	⋮
4.2000	12.1811	85.5517	89.8036
4.4000	13.6442	102.6514	108.5734
4.6000	15.2832	123.0609	131.0845
4.8000	17.1189	147.4087	158.0600
5.0000	19.1752	176.4414	190.3599

As shown in the table, the first column recorded linear increment of time. The values in the second to the fifth rows are exactly the same as the values from the application of dde23. This signifies that our coding of RK2 and RK4 was correct.

The step-size was retained to be the same. The value of $\frac{h}{2} = 0.1$ was stored automatically, making it easier to be recalled.

Table 3 Errors between dde23 and hybrid RK2 and RK4

Time	y_1	y_2	y_3
0.0000	0.0000	0.0000	0.0000
0.2000	0.0000	0.0000	0.0000
0.4000	0.0000	0.0000	0.0000
0.6000	0.0000	0.0000	0.0000
0.8000	0.0000	0.0000	0.0000
1.0000	0.0000	0.0001	0.0000
⋮	⋮	⋮	⋮
4.2000	0.0001	0.0119	0.0093
4.4000	0.0001	0.0150	0.0119
4.6000	0.0002	0.0188	0.0152
4.8000	0.0002	0.0236	0.0194
5.0000	0.0003	0.0294	0.0245

Table 3 represents the errors between dde23 and hybrid RK2 and RK4. By computing the difference between the output values from Table 1 and Table 2, we found that the maximum error was 0.0294. This means that the hybrid RK2 and RK4 are acceptable to use. Notice that the error start materializing from $t = 1$ onwards. As shown in Table 3, it is obvious that the values of y_2 contributed to the largest error.

The following Example 2 is an example where the increment of time is nonlinear; this is an instance when discontinuities may occur.

3.2 Example 2

Example 2 was tested as an infectious disease model from [5]. The system is given as

$$y_1'\ t\ = -y_1\ t\ y_2\ t-1\ + y_2\ t-10$$
$$y_2'\ t\ = y_1\ t\ y_2\ t-1\ - y_2(t)$$
$$y_3'\ t\ = y_2\ t\ - y_2(t-10)$$

to be solved over the interval [0, 30] with history $y_1\ t\ =5,\ y_2\ t\ =0.1,\ y_3\ t\ =1$ for $t \leq 0$ shortest time delay $\tau = 1.0$, and $\frac{h}{2} = 0.1$. Again, the problem was solved using both dde23 from Matlab and hybrid RK2 and RK4 from our scheme. Table 4 below shows the output from the application of dde23.

Table 4 Output Values of dde23

Time	y_1	y_2	y_3
0.0000	5.0000	0.1000	1.0000
0.0200	4.9920	0.1079	1.0001
0.1200	4.9523	0.1450	1.0028
0.3287	4.8706	0.2102	1.0192
0.5838	4.7732	0.2714	1.0554
0.7919	4.6955	0.3094	1.0952
1.0000	4.6194	0.3387	1.1419
⋮	⋮	⋮	⋮
3.0000	2.2250	1.3302	2.5448
⋮	⋮	⋮	⋮
10.0000	0.3325	0.0389	5.7286
⋮	⋮	⋮	⋮
11.0000	0.5538	0.0274	5.5187
⋮	⋮	⋮	⋮
12.0000	1.0755	0.0259	4.9986
⋮	⋮	⋮	⋮
20.0000	0.1700	0.8729	5.0571
⋮	⋮	⋮	⋮
28.6888	3.3104	0.0225	2.7671
28.9945	3.7665	0.0288	2.3047
29.2890	4.1642	0.0371	1.8987
29.5786	4.5000	0.0483	1.5518
29.7893	4.7053	0.0588	1.3359
30.0000	4.8765	0.0718	1.1516

As can be seen clearly from the table, the increment of time did not follow any rule, hence we could not tract past values of the system at specific time. This is where discontinuities may occur. For instance, we could not get the past value at $t = 0.8000$ because dde23 would not allow it. This was due to users' inability to influence the step size in dde23.

In contrast, our scheme allowed us to determine the number of nodes as needed, by varying the step size. Hence all values could be read, passed, and stored automatically. In Table 5, we tabulate the result from applying our method. The increment of time remains linear with $h = 0.2$, and $\frac{h}{2} = 0.1$.

Table 5 Output Values of hybrid RK2 and RK4

Time	y_1	y_2	y_3
0.0000	5.0000	0.1000	1.0000
0.1000	4.9602	0.1378	1.0020
0.2000	4.9208	0.1718	1.0074
0.3000	4.8818	0.2020	1.0162
0.4000	4.8432	0.2291	1.0277
0.5000	4.8049	0.2531	1.0419
0.6000	4.7671	0.2746	1.0583
0.7000	4.7296	0.2937	1.0768
0.8000	4.6925	0.3106	1.0970
0.9000	4.6557	0.3255	1.1188
1.0000	4.6193	0.3386	1.1420
⋮	⋮	⋮	⋮
3.0000	2.2256	1.3301	2.5443
⋮	⋮	⋮	⋮
10.0000	0.3329	0.0392	5.7280
⋮	⋮	⋮	⋮
11.0000	0.5539	0.0277	5.5184
⋮	⋮	⋮	⋮
12.0000	1.0751	0.0262	4.9987
⋮	⋮	⋮	⋮
20.0000	0.1707	0.864	5.0652
⋮	⋮	⋮	⋮
29.0000	3.7824	0.0295	2.2881
29.1000	3.9216	0.0321	2.1463
29.2000	4.0552	0.035	2.0098
29.3000	4.1824	0.0383	1.8793
29.4000	4.3031	0.0419	1.755
29.5000	4.4166	0.0459	1.6375
29.6000	4.5228	0.0503	1.5269
29.7000	4.6215	0.0552	1.4233
29.8000	4.7125	0.0607	1.3268
29.9000	4.7959	0.0667	1.2373
30.0000	4.8717	0.0734	1.1549

Clearly, the increment of time was controlled by the user through the choice of the step size, hence we could tract past values at any instance we desired. Assume that we require getting the past value at $t = 0.8500$ which is not in Table 5, therefore all we need to do is to change the step size to $h = 0.1$, and $\frac{h}{2} = 0.05$ which we definitely cannot do with dde23.

The errors between the result from dde23 and our scheme are shown in Table 6.

Table 6 Errors between dde23 and hybrid RK2 and RK4

Time	y_1	y_2	y_3
0.0000	0.0000	0.0000	0.0000
1.0000	0.0000	0.0001	0.0001
2.0000	0.0003	0.0002	0.0001
3.0000	0.0006	0.0001	0.0005
10.0000	0.0004	0.0003	0.0006
11.0000	0.0001	0.0003	0.0003
12.0000	0.0004	0.0003	0.0001
20.0000	0.0007	0.0089	0.0081
30.0000	0.0048	0.0016	0.0033

Performance-wise, our method is just as good as an approximator as dde23. This is evident from the errors tabulated in Table 6, where the maximum error was 0.0089. The entries in Table 6 are the only nodes comparable from both methods.

3.3 Example 3

Example 3 was employed as a dynamic model for a continuously-stirred tank reactor, in which the chemical reaction $A \rightarrow B$ would occur from [3]. The system is given as

$$x_1'(t) = -2x_1(t) + \frac{1}{10}\left(1 - x_1(t)\right)\exp\left[\frac{20x_2(t)}{20 + x_2(t)}\right] + x_1(t-2.0)$$

$$x_2'(t) = -\frac{5}{2}x_2(t) + \frac{4}{5}\left(1 - x_1(t)\right)\exp\left[\frac{20x_2(t)}{20 + x_2(t)}\right] + x_2(t-2.0)$$

to be solved over the interval [0, 6] with history $x_1(t) = 1,\ x_2(t) = 1$ for $t \le 0$ where $x_1(t)$ is the concentration of A at time t (dimensionless), $x_2(t)$ is the temperature of the reactor at the time t (dimensionless). Since there was just one time delay of $\tau = 2.0$, hence we set $h = 0.1$, and $\frac{h}{2} = 0.05$. The problem was solved using both dde23 from Matlab, and hybrid RK2 and RK4 from our scheme again. Table 7 below shows the output from the application of dde23.

Table 7 Output Values of dde23

Time	x_1	x_2
0.0000	1.0000	1.0000
0.0533	0.9497	0.9277
0.1603	0.8652	0.8201
0.2831	0.7896	0.7401
0.4241	0.7244	0.6853
0.5882	0.6696	0.6518
0.7848	0.6251	0.6360
1.0332	0.5902	0.6346
1.3697	0.5649	0.6444
1.6849	0.5538	0.6558
2.0000	0.5485	0.6650
2.1756	0.5350	0.6536
2.3511	0.5082	0.6294
2.5441	0.4747	0.6059
2.7953	0.4343	0.5889
3.1248	0.3938	0.5876
3.5624	0.3609	0.6054
4.0000	0.3452	0.6298
4.2316	0.3398	0.6399
4.4632	0.3308	0.6434
4.8471	0.3091	0.6449
5.2568	0.2849	0.6533
5.6284	0.2677	0.6699
6.0000	0.2564	0.6915

As shown in Table 7, the increment of time was non-linearly. Hence, the past values could not be tracked at specific instances, circumstances of which resulted in discontinuities.

Table 8 Output Values of hybrid RK2 and RK4

Time	x_1	x_2
0.0000	1.0000	1.0000
0.0500	0.9528	0.9321
0.1000	0.9104	0.8755
0.1500	0.8726	0.8291
0.2000	0.8386	0.7904
0.2500	0.8082	0.7587
0.3000	0.7808	0.7323
0.3500	0.7564	0.7108
0.4000	0.7343	0.6931
0.4500	0.7146	0.6789
0.5000	0.6968	0.6673
0.5500	0.6809	0.6582
0.6000	0.6666	0.6509
0.6500	0.6538	0.6454
0.7000	0.6423	0.6412
0.7500	0.6320	0.6383
0.8000	0.6227	0.6363
0.8500	0.6144	0.6351
0.9000	0.6069	0.6345
0.9500	0.6002	0.6346
1.0000	0.5942	0.6350
⋮	⋮	⋮
2.0000	0.5488	0.6653
⋮	⋮	⋮
3.0000	0.4077	0.5866
⋮	⋮	⋮
4.0000	0.3464	0.6312
⋮	⋮	⋮
5.0000	0.2997	0.6466
⋮	⋮	⋮
6.0000	0.2571	0.6925

Table 8 contains the output values from our scheme. Clearly, the increment of time was controlled by us through the choice of the step size; hence we could track past values at any necessary instance.

The errors between the result from dde23 and our scheme are summarized in Table 9.

Table 9 Errors between dde23 and hybrid RK2 and RK4

Time	x_1	x_2
0.0000	0.0000	0.0000
2.0000	0.0003	0.0003
4.0000	0.0012	0.0014
6.0000	0.0007	0.0010

The maximum error was 0.0014, which signifies that our scheme can deliver acceptable solutions, comparable to the ones from dde23. The entries in

Table 9 are only the nodes comparable from both methods.

4.0 CONCLUSION

Overcoming the discontinuities in DDE systems caused by the initial function $S(t)$ in system (2) is the main concern of this study. We have successfully developed a scheme based on hybrid RK2 and RK4 for such problems. The success of the scheme is due to the allowance for modification in RK methods for overcoming the discontinuities specifically at the point of determination of the step size. To verify our effort, we have compared our results with results from dde23. We have employed three cases of DDEs; one was linear with no possibility of discontinuity, and the other two were examples of nonlinear problems, where discontinuities would most probably occur. Numerical results indicate that our method, based on hybrid RK2 and RK4, has performed quite well compared to dde23 from Matlab. The first and second cases consisted of two different time delays, and third case only consisted one time delay; τ_i with the second one leading to possibility of discontinuities. We found that dde23 was not able to overcome the problem due its inability to track past histories, while ours excelled. Our method has the upper hand because it has the facility to tract past history at any instance required by varying its step size. In this study, we only used the fixed time delay τ ; for future work we intend to use the variable time delay τ with combination of the Lagrange interpolation.

Acknowledgement

The work is funded by Ministry of Higher Education Malaysia and Universiti Teknologi Malaysia through the grant RUG 14H58.

References

[1] Bellen, A. and Zennaro, M. 2003. *Numerical Methods for Delay Differential Equations.* Oxford: Oxford Science Publication.
[2] R. Xu and L. Chen. 2000. Persistence and Stability for a Two-species Ratio-dependent Predator-prey System with Time Delay in A Two-patch Environment. *Comput. Math. Applic.* 40(4-5): 577-588.
[3] Y. Y. Cao and P. M. Frank. 2000. Analysis and Synthesis of Nonlinear Time-delay Systems via Fuzzy Control Approach. *IEEE Trans. Fuzzy Syst.* 8(2): 200-211.
[4] D. R. Willé and C. T. H. Baker. 1992. DELSOL – A Numerical Code for the Solution of Systems of Delay-differential Equations. *Appl. Numer. Math.* 9: 223-234.
[5] E. Hairer, S. P. Norsett, and G. Wanner, 1987. *Solving Ordinary Differential Equations I.* Springer-Verlag, Berlin.
[6] Christopher, T. H. B., and Evelyn, B. 2000. Numerical Analysis of Explicit One-step Methods for Stochastic Delay Differential Equations. *LMS J. Comput. Math.* 3: 315-335.
[7] Kloeden, P. and Schropp, J. 2003. Runge-kutta Methods for Monotone Differential and Stochastic Equations. *Proc. Appl. Math. Mesh.* 3: 565-566.
[8] Küchler, U. and Platen, E. 2000. Strong Discrete Time Approximation of Stochastic Differential Equations with Time Delay. *Mathematics and Computers in Simulation*. 54(1-3): 189-205.
[9] Platen, E. 1999. An Introduction to Numerical Methods for Stochastic Differential Equations. *Acta Numeric.* 8: 197-246.
[10] Kloeden, P. E. and Platen, E. 1999. Numerical Solution of Stochastic Differential Equations. *Appl. Math.* Springer. 23: Third corrected printing.
[11] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brain P. Flannery. 1992. *Numerical Recipes in C.* 2nd ed. The Art of Scientific Computing. New York, USA: Cambridge University Press.
[12] Sergio Pissanatzky. 2004. *Vectors, Matrices, and C++ Code.* United States: Scientific Controls.
[13] Gennadii A. Bocharov, and Fathalla A. Rihan. 2000. Numerical Modelling in Biosciences Using Delay Differential Equations. *J. Comput. Appl. Math.* 125(1-2): 183-199.
[14] Orlov, Y., Belkoura, L., Richard, J. P., and Dambrine, M. 2002. On Identifiability of Linear Time-delay Systems. *IEEE Trans. on Automatic Control.* 47(8): 1319-1324.
[15] Lilianne Denis-Vidal, Carine Jauberthie, and Ghislaine Joly-Blanchard. 2006. Identifiability of a Nonlinear Delayed-Differential Aerospace Model. *IEEE Trans. on Automatic Control*. 51(1): 154-158.