Applied and Computational Mathematics
2014; 3(1): 9-14
Published online February 20, 2014 (http://www.sciencepublishinggroup.com/j/acm)
doi: 10.11648/j.acm.20140301.12

Numerical solution of linear Volterra integro-differential equation using Runge-Kutta-Fehlberg method

Ali Filiz

Department of Mathematics, Adnan Menderes University, 09010 AYDIN-TURKEY

Email address:
afiliz@adu.edu.tr

To cite this article:
Ali Filiz. Numerical Solution of Linear Volterra Integro-Differential Equation using Runge-Kutta-Fehlberg Method. *Applied and Computational Mathematics.* Vol. 3, No. 1, 2014, pp. 9-14. doi: 10.11648/j.acm.20140301.12

Abstract: In this paper a new fourth and fifth-order numerical solution of linear Volterra integro-differential equation is discussed. One popular technique that uses here for error control is called the Runge-Kutta-Fehlberg method for Ordinary Differential Equation (ODE) part and Newton-Cotes formulae for integral parts.

Keywords: A fourth and Fifth-Order Accuracy, Lagrange Polynomial Interpolating, Newton-Cotes Formulas, Runge-Kutta Methods, Linear Volterra Integro-Differential Equation

1. Introduction

Mathematical modeling of real-life problems usually results in functional equations, like ordinary or partial differential equations, and integral and integro-differential equations. Many mathematical formulations of physical phenomena contain integro-differential equations; these equations arise in many fields like physics, astronomy, potential theory, fluid dynamics, biological models, and chemical kinetics. Moreover, equations containing partial integro-differential equations arise in fluid dynamics, viscoelasticity, engineering, mathematical biology, financial mathematics and other disciplines. A functional equation in which the unknown function appears in the form of it is a derivative as well as under the integral sign is called an integro-differential equation (see [10, 11, 14, 15, 16]). Integro-differential equations; are usually difficult to solve analytically and numerically; so, it is required to obtain an efficient approximate solution.

In this paper we will consider the linear Volterra integro-differential equation of the form (see [2, 4, 7, 8])

$$u'(t) = F(t,u(t), \int_{t_0}^{t} k(t-s)u(s)ds),\ u(t_0) = u_0,\ t \geq t_0. \quad (1)$$

Equation (1) can be solved numerically using various methods (see [6, 9, 10, 11, 12]). In this paper $u(t_n)$ will denote the exact value of u at $t_n = t_0 + nh$. We shall use $\tilde{u}(t_n)$ or $\tilde{u}_n$ to denote a numerical solution u of at t_n. Since the integral cannot be determined explicitly, it may be approximated using familiar numerical integration methods. The Newton-Cotes integration formulae, which include the 2-point closed Newton-Cotes formula is called the trapezoidal rule, the 3-point rule is known as Simpson's 1/3 rule, the 4-point closed rule is Simpson's 3/8 rule, the 5-point closed rule is Boole's rule (Bode's rule), Weddle's rule, higher rules include the 6-point, 7-point and 8-point are well suited here since they use nodes which were given in [1, 10, 13,17] and [4, 12].

2. The Numerical Treatment of Integro-Differential Equations

In general formulas for the numerical solution of integro-differential equations rely on formulas for the underlying Ordinary Differential Equation (ODE), combined with auxiliary quadrature rules approximation of

$$\tilde{z}(t_n) := h\sum_{j=0}^{n} \omega_{n,j} k(t_n - t_j)\ \tilde{u}(t_j) \approx \int_{t_0}^{t_n} k(t-s)\ u(s)\ ds. \quad (2)$$

For equation (1), we will adapt the Runge-Kutta-Fehlberg Method (see in [3, 10, 11, 13]) and method of convergence $O(h^4)$ and $O(h^5)$ respectively.

Of course, we have defined approximations $\tilde{z}(t_n)$ in terms of appropriate quadrature rules that reflect the underlying ODE method. The combinations of formulas can be chosen on the basis of order of convergence. The first

involves adapting Runge-Kutta methods. Thus, we will require to approximate integral terms in (2) at selected values of t in $\tilde{u}(t_n)$. Equation (1) can be solved using various methods. In this paper we shall focus on fourth and fifth-order numerical method for equation (1). The integral term may be approximated using familiar numerical integration methods. The Newton-Cotes integration formulae, which include left and right rectangle rules, the trapezoidal rule, Simpson's 1/3 rule and Simpson's 3/8 rule are well suited here since they used nodes which were previously calculated [10, 11]:

$$\int_{t_0}^{t_n} k(t-s)\, u(s)\, ds \approx h\sum_{j=0}^{n} \omega_{n,j} k(t_n - t_j)\, \tilde{u}(t_j),$$

where $\omega_{n,j}$ are the appropriate coefficients for the composite integration schemes chosen. A combination of integration method may be used.

3. A New Numerical Routine for Linear Integro-Differential Equation

Now consider the non-dimensional problem (1). In order to solve (1) numerically, we purpose the use of the Runge-Kutta methods familiar to most mathematicians. We consider methods which approximate the solution the initial value problem (IVP) in equation (1) at time $t_n = t_0 + nh$, n=0, 1, 2, 3… where $h = t_n - t_{n-1}$ is the constant nodal step-size and, in the Example 3.1,

$$F(t, u(t), \int_{t_0}^{t} k(t-s)\, u(s)\, ds) = \gamma + \alpha u(t) + \beta \int_{0}^{t} k(t-s)\, u(s)\, ds.$$

For example, the explicit Euler method approximates the solution to Example 3.1 at t_{n+1}

$$\tilde{u}_{n+1} = \tilde{u}_n + h \left(\gamma + \alpha u(t) + \beta \int_{t_0}^{t_n} e^{-\delta(t-s)}\, u(s)\, ds \right).$$

The explicit finite difference method given in [11] as applied to equation (1) easily extended to more accurate predictor-corrector method. The predictor step uses $(\tilde{u}_{n+1} = \tilde{u}_n + h\left(F(t_n, \tilde{u}_n, \tilde{z}(t_n))\right))$ to obtain $\tilde{u}_{n+1}^{\kappa}$, which is followed by the corrector step, which uses higher order trapezoidal method

$$\tilde{u}_{n+1} = \tilde{u}_n + h\left(\frac{1}{2}F\left(t_n, \tilde{u}_n, \tilde{z}(t_n)\right) + \frac{1}{2}F\left(t_{n+1}, \tilde{u}_{n+1}^{\kappa}, \tilde{z}(t_{n+1})\right)\right). \quad (3)$$

This procedure is sometimes referred to as modified Euler method (second order Runge-Kutta-RK2) and is one order magnitude more accurate than the explicit Euler method.

The fourth order classical Runge-Kutta method (RK4) can also be adapted to the numerical solution of equation (1). Stepping from $\tilde{u}_n$ with step-size h to obtain $\tilde{u}_{n+1}$, the RK4 method as applied to this problem in [10, 11].

The fifth-order Runge-Kutta-Fehlberg and sixth order Runge-Kutta-Verner methods [3] may be used but not readily, since the intranodal evaluation points are uniformly spaced. Consequently, the integrals needed during the intermediate calculations to step from t_n to t_{n+1} may require the trapezoidal rule or Lagrange polynomial interpolating integration on a non-uniform partition $[t_n, t_{n+1}]$.

Other high-order finite difference methods which may be used here include the Adam-Basforth multistep methods. One such fourth order method is described in [3]. It uses the RK4 method to obtain the starting values u_0, u_1, u_2. There after, the method uses the fourth order explicit Adam-Basforth method as a predictor and fourth-order implicit Adam-Moulton method corrector to step from t_n to t_{n+1}.

Runge-Kutta methods are methods for the numerical solution of the ODE as follows

$$u'(t) = f(t, u(t)), \quad u(t_0) = u_0,$$

which takes from

$$\tilde{u}_{n+1} = \tilde{u}_n + \sum_{i=1}^{s} b_i k_i,$$

$$k_i = hf(t_n + c_i h, \tilde{u}_n + \sum_{j=1}^{s} a_{ij} k_j),$$

where

$$k_1 = hf(t_n, \tilde{u}_n),$$

$$k_2 = hf(t_n + c_2 h, \tilde{u}_n + a_{21} k_1),$$

$$k_3 = hf(t_n + c_3 h, \tilde{u}_n + a_{31} k_1 + a_{32} k_2),$$

…..

$$k_s = hf(t_n + c_s h, \tilde{u}_n + a_{s1} k_1 + a_{s2} k_2 + \cdots + a_{ss} k_s).$$

The methods mentioned in this paper are defines by its Butcher tableau, which puts the coefficients of the method in the following table

c_1	a_{11}	a_{12}	…	a_{1s}
c_2	a_{21}	a_{22}	…	a_{2s}
⋮	⋮	⋮	⋱	⋮
c_s	a_{s1}	a_{s2}	…	a_{ss}
	b_1	b_2	…	b_s

To specify a particular method, are needs to provide the integer s and the coefficients a_{ij} (for $1 \le j < i \le s$), b_i and c_i (for $i = 2, 3, 4, \cdots, s$).

The matrix $[a_{ij}]$ is called the Runge-Kutta matrix, while

b_i and c_i are known as the weights and nodes. The explicit methods are these where the matrix is lower triangular. The embedded pair proposed by Fehlberg [3]

0						
1/4	1/4					
3/8	3/32	9/32				
12/13	1932/2197	−7200/2197	7296/2197			
1	439/216	−8	3680/513	−845/4104		
1/2	-8/27	2	−3544/2565	1859/4104	−11/40	
	16/135	0	6656/12825	28561/56430	−9/50	2/55
	25/216	0	1408/2565	2197/4104	−1/5	0

The first row of coefficients at the bottom of the table gives the fifth-order accurate method, and the second row gives the fourth-order accurate method.

The Runge-Kutta-Fehlberg Method (denoted RKF) is one way to try to resolve equation (1). It has a procedure to determine if the proper step size h is being used. At each step, two different approximations for the solution are made and compared. If the two answers are in close agreement, the approximation is accepted. If the two answers do not agree to specified accuracy, the step size is reduced. If the answers agree to more significant digits than required, the step size is increased. Each step requires the use of the following six values from k_1 to k_6. Runge-Kutta-Fehlberg method (RKF) can also be adapted to the numerical solution of (1). Stepping from $\widetilde{u}_n$ with step-size h to obtain $\widetilde{u}_{n+1}$, the RKF method as applied to this problem may be written as:

$$k_1 = hF(t_n, \widetilde{u}_n, \widetilde{z}(t_n)),$$

$$\widetilde{u}^a_{n+1/4} = \widetilde{u}_n + \frac{1}{4}k_1, \quad k_2 = hF(t_{n+1/4}, \widetilde{u}^a_{n+1/4}, \widetilde{z}_{n+1/4}),$$

$$k_2 = hF\left(t_{n+1/4}, \widetilde{u}^a_{n+1/4}, \widetilde{z}_n + \frac{h}{8}\left[\widetilde{u}_n + \widetilde{u}^a_{n+1/4}\right]\right),$$

$$\widetilde{u}^b_{n+3/8} = \widetilde{u}_n + \frac{3}{32}k_1 + \frac{9}{32}k_2,$$

$$k_3 = hF(t_{n+3/8}, \widetilde{u}^b_{n+3/8}, \widetilde{z}_{n+3/8}),$$

$$k_3 = hF\left(t_{n+3/8}, \widetilde{u}^b_{n+3/8}, \widetilde{z}_n + \frac{3h}{16}\left[\widetilde{u}_n + \widetilde{u}^b_{n+3/8}\right]\right),$$

$$\widetilde{u}^c_{n+12/13} = \widetilde{u}_n + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3,$$

$$k_4 = hF(t_{n+12/13}, \widetilde{u}^c_{n+12/13}, \widetilde{z}_{n+12/13}),$$

$$k_4 = hF\left(t_{n+12/13}, \widetilde{u}^c_{n+12/13}, \widetilde{z}_n + \frac{12h}{26}\left[\widetilde{u}_n + \widetilde{u}^c_{n+12/13}\right]\right), \quad (4)$$

$$\widetilde{u}^d_{n+1} = \widetilde{u}_n + \frac{439}{216}k_1 - 8k_2 - \frac{3680}{513}k_3 - \frac{845}{4104}k_4,$$

$$k_5 = hF(t_{n+1}, \widetilde{u}^d_{n+1}, \widetilde{z}_{n+1}),$$

$$k_5 = hF\left(t_{n+1}, \widetilde{u}^d_{n+1}, \widetilde{z}_n + \frac{h}{2}\left[\widetilde{u}_n + \widetilde{u}^d_{n+1}\right]\right),$$

$$\widetilde{u}^e_{n+1/2} = \widetilde{u}_n + \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5,$$

$$k_6 = hF(t_{n+1/2}, \widetilde{u}^e_{n+1/2}, \widetilde{z}_{n+1/2}),$$

$$k_6 = hF\left(t_{n+1/2}, \widetilde{u}^e_{n+1/2}, \widetilde{z}_n + \frac{h}{4}\left[\widetilde{u}_n + \widetilde{u}^e_{n+1/2}\right]\right),$$

Then an approximation to the solution of the equation (1) is made using a Runge-Kutta method of order 4:

$$\widetilde{u}_{n+1} = \widetilde{u}_n + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5, \quad (5)$$

where the four function values k_1, k_3, k_4 and k_5 are used. Notice that, k_2 is not used in formula (5). A better value for the solution is determined using a Runge-Kutta Method of order 5:

$$\widetilde{u}_{n+1} = \widetilde{u}_n + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6. \quad (6)$$

In this example, the trapezoidal rule is used to approximate $\widetilde{z}(t_n) \approx \int_{t_0}^{t_n} k(t-s)\,u(s)\,ds$ on $[t_n, t_{n+1/4}]$, $[t_n, t_{n+3/8}]$, $[t_n, t_{n+12/13}]$, $[t_n, t_{n+1}]$, $[t_n, t_{n+1/2}]$ in calculating, k_2, k_3, k_4, k_5 and k_6 respectively. If desired, the trapezoidal rule may be used on $[t_0, t_n]$ (gives second order accuracy); the trapezoidal rule and Simpson's 1/3 rule (giving third order accuracy, see [10, 11]) may be used on $[t_0, t_n]$.

In order to get fifth order accuracy the integral term must be evaluated more accurately on $[t_n, t_{n+1/4}]$, $[t_n, t_{n+3/8}]$, $[t_n, t_{n+12/13}]$, $[t_n, t_{n+1}]$, $[t_n, t_{n+1/2}]$ in calculating, k_2, k_3, k_4, k_5 and k_6, as shown in (7), (8), (9), (10), (11) below.

The 5-point extended closed rule is Boole's method may be devised on $[t_0, t_n]$ as following pseudocode:

Algorithm 1. Boole's method on $[t_0, t_n]$.

```
Set z(1) to zero
Set u(1) to u0
For all n Compute z(n+1)
If (n==1) then
Set  z(n+1) to z(n) + h(u(n) + u(n+1))/2
{the trapezoidal rule}
elseif (n==2) then
Set z(n+1)to z(n-1) + h(u(n-1) + 4u(n) + u(n+1))/3
{Simpson's 1/3 rule}
elseif (n==3) then
Set z(n+1) to z(n-2) +3h(u(n-2) + 3u(n-1) + 3u(n) +
u(n+1))/8
{Simpson's 3/8 rule}
elseif (n==4) then
Set z(n+1) to z(n-3) + 2h(7 u(n-3) + 32u(n-2) + 12u(n-1)
+ 32u(n) + 7u(n+1))/45 {Boole's rule}
```

```
elseif (n==5) then
Set z(n+1) to z(n-4) + 5h(19u(n-4) + 75u(n-3) + 50u(n-2)
+ 50u(n-1) + 75u(n) + 19u(n+1))/288
elseif (n==6) then
Set z(n+1) to z(n-5) + h(41u(n-5)+ 216 u(n-4) + 27u(n-3)
+ 272u(n-2) + 27 u(n-1) + 216u(n) + 41u(n+1))/140
elseif (n==7) then
Set z(n+1) to z(n-6) + 7h(751u(n-6) + 3577u(n-5) +
1323u(n-4) + 2989u(n-3) + 2989u(n-2) + 1323u(n-1) + …
+ 3577u(n) + 7511u(n+1))/17280
elseif (n==8) then
Set z(n+1) to z(n-3) + 2h(7u(n-3) + 32 u(n-2) + 12u(n-1)
+ 32u(n) + 7u(n+1))/45
elseif (mod(n,4)==0) then
Set z(n+1) to z(n-3) + 2h(7 u(n-3) + 32u(n-2) +
12u(n-1) + 32u(n) + 7u(n+1))/45
elseif (mod(n,4)==1) then
Set z(n+1) to z(n-3) + 2h(7u(n-3) + 32u(n-2) + 12u(n-1)
+ 32u(n) + 7u(n+1))/45
elseif (mod(n,4)==2) then
Set z(n+1) to z(n-3) + 2h(7u(n-3) + 32u(n-2) + 12u(n-1)
+ 32u(n) +7u(n+1))/45
elseif (mod(n,4)==3) then
Compute z(n+1)= z(n-3) + 2h(7u(n-3) + 32u(n-2) +
12u(n-1) + 32u(n) + 7u(n+1))/45
else
Set z(n+1) to z(n-3) + 2h(7u(n-3) + 32u(n-2) + 12u(n-1)
+ 32u(n) + 7u(n+1))/45
end if
return n; u(n) {Returns summation of z(n)}
```

If we interpolating on $\widetilde{u}_{n-2}, \widetilde{u}_{n-1}, \widetilde{u}_n, \widetilde{u}_{n+1/6}$ (special formulae required for the first two steps, for example we can use (5) and (6)) Lagrange's formula for points t=-2, -1, 0, 1/4 gives

$$u(t)=\frac{1}{h^3}\begin{bmatrix} -\frac{2}{9}t\ (t-\frac{h}{4})\ (t+h)u_{-2}+\frac{4}{5}t\ (t-\frac{h}{4})(t+2h)u_{-1}- \\ 2(t+h)(t+2h)(t-\frac{h}{4})u_0+\frac{64}{45}t(t+h)(t+2h)u_{1/4} \end{bmatrix}.$$

If we integrate the expression between 0 and h/4, we get

$$\int_0^{h/4} u(s)ds \approx h\left(\frac{9}{80}u_{1/4}+\frac{1}{1536}u_{-2}-\frac{17}{3840}u_{-1}+\frac{217}{1536}u_0\right). \quad (7)$$

Similarly, we can find t=-2, -1, 0, 3/8

$$\int_0^{3h/8} u(s)ds \approx h\left(\frac{57}{352}u_{3/8}+\frac{9}{4096}u_{-2}-\frac{315}{22528}u_{-1}+\frac{921}{496}u_0\right), \quad (8)$$

and find t=-2, -1, 0, 12/13

$$\int_0^{12h/13} u(s)ds \approx h\left(\frac{114}{325}u_{12/13}+\frac{72}{2197}u_{-2}-\frac{9216}{54925}u_{-1}+\frac{1554}{2197}u_0\right) \quad (9)$$

and find t=-2, -1, 0, 1

$$\int_0^{h} u(s)ds \approx h\left(\frac{3}{8}u_1+\frac{1}{24}u_{-2}-\frac{5}{24}u_{-1}+\frac{19}{24}u_0\right), \quad (10)$$

and finally find t=-2, -1, 0, 1/2

$$\int_0^{h/2} u(s)ds \approx h\left(\frac{5}{24}u_{1/2}+\frac{1}{192}u_{-2}-\frac{1}{32}u_{-1}+\frac{61}{192}u_0\right). \quad (11)$$

When n<3 the first two approximations can be found by formula (4)

Therefore the Runge-Kutta-Fehlberg formulae become $n \geq 3$ (for starting values we can use equation (5) and (6))

$$k_1 = hF(t_n, \widetilde{u}_n, \widetilde{z}(t_n)\ ),$$

$$\widetilde{u}^a_{n+1/4} = \widetilde{u}_n + \frac{1}{4}k_1,$$

$$k_2 = hF(t_{n+1/4}, \widetilde{u}^a_{n+1/4}, \widetilde{z}_{n+1/4}\ ),$$

$$k_2 = hF\left(t_{n+1/4}, \widetilde{u}^a_{n+1/4}, \widetilde{z}_n + h\begin{bmatrix}\frac{9}{80}\widetilde{u}^a_{n+1/4}+\frac{1}{1536}\widetilde{u}_{n-2}- \\ \frac{17}{3840}\widetilde{u}_{n-1}+\frac{217}{1536}\widetilde{u}_n\end{bmatrix}\right),$$

$$\widetilde{u}^b_{n+3/8} = \widetilde{u}_n + \frac{3}{32}k_1 + \frac{9}{32}k_2,$$

$$k_3 = hF(t_{n+3/8}, \widetilde{u}^b_{n+3/8}, \widetilde{z}_{n+3/8}\ ),$$

$$k_3 = hF\left(t_{n+3/8}, \widetilde{u}^b_{n+3/8}, \widetilde{z}_n + h\begin{bmatrix}\frac{57}{352}\widetilde{u}^b_{n+3/8}+\frac{9}{4096}\widetilde{u}_{n-2}- \\ \frac{315}{22528}\widetilde{u}_{n-1}+\frac{921}{4096}\widetilde{u}_n\end{bmatrix}\right),$$

$$\widetilde{u}^c_{n+12/13} = \widetilde{u}_n + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3,$$

$$k_4 = hF(t_{n+12/13}, \widetilde{u}^c_{n+12/13}, \widetilde{z}_{n+12/13}\ ),$$

$$k_4 = hF\left(t_{n+12/13}, \widetilde{u}^c_{n+12/13}, \widetilde{z}_n + h\begin{bmatrix}\frac{114}{325}\widetilde{u}^c_{n+12/13}+\frac{721}{2197}\widetilde{u}_{n-2}- \\ \frac{9216}{54925}\widetilde{u}_{n-1}+\frac{1554}{2197}\widetilde{u}_n\end{bmatrix}\right),$$

$$\widetilde{u}^d_{n+1} = \widetilde{u}_n + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4,$$

$$k_5 = hF(t_{n+1}, \widetilde{u}^d_{n+1}, \widetilde{z}_{n+1}), \quad (12)$$

$$k_5 = hF\left(t_{n+1}, \widetilde{u}^d_{n+1}, \widetilde{z}_n + h\begin{bmatrix}\frac{3}{8}\widetilde{u}^d_{n+1}+\frac{1}{24}\widetilde{u}_{n-2}- \\ \frac{5}{24}\widetilde{u}_{n-1}+\frac{19}{24}\widetilde{u}_n\end{bmatrix}\right),$$

$$\widetilde{u}^{e}_{n+1/2} = \widetilde{u}_n - \frac{8}{27}k_1 + 2k_2 + \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5,$$

$$k_6 = hF(t_{n+1/2}, \widetilde{u}^{e}_{n+1/2}, \widetilde{z}_{n+1/2}\ ),$$

$$k_6 = hF\left(t_{n+1/2}, \widetilde{u}^{e}_{n+1/2}, \widetilde{z}_n + h\begin{bmatrix}\frac{5}{24}\widetilde{u}^{e}_{n+1/2} + \frac{1}{192}\widetilde{u}_{n-2} - \\ \frac{1}{32}\widetilde{u}_{n-1} + \frac{61}{192}\widetilde{u}_n\end{bmatrix}\right),$$

to estimate the local error in a Runge-Kutta Method of order four given by

$$\widetilde{u}_{n+1} = \widetilde{u}_n + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5,$$

In formula (12), this technique consistent of using a Runge-Kutta Method with local truncation error of order five,

$$\widetilde{u}_{n+1} = \widetilde{u}_n + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6.$$

We can construct an algorithm similar to the sixth, seventh and eighth-order Runge-Kutta formulae and we can repeat Example 3.1 using this new method.

Table 1 show the fourth and fifth order accuracy obtained with this formula. In Example 3.1, we have used Runge-Kutta-Fehlberg methods and numerical quadrature rules, such that trapezoidal rule, the 3-point rule is known as Simpson's 1/3 rule, the 4-point closed rule is Simpson's 3/8 rule, the 5-point closed rule is Boole's rule (Bode's rule), Weddle's rule, higher rules include the 6-point, 7-point and 8-point and their combinations.

Example 3.1: Consider a first order Linear Volterra integro-differential equation of the form

$$u'(t) = \gamma + \alpha u(t) + \beta\int_{t_0}^{t} e^{-\delta(t-s)}u(s)\,ds,\ \ t \geq 0; \qquad u(t_0) = u_0. \tag{13}$$

This equation (13) has analytical solution

$$\begin{aligned}u(t) = &-\frac{\delta\gamma}{\beta+\alpha\delta} + (-e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}-\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\alpha\beta u_0 + \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}+\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\alpha\beta u_0 - \\ &2e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}-\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\beta\gamma + \\ &2e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}+\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^{2^2}}\right)}\beta\gamma - \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}-\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\alpha^2\ \delta u_0 + \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}+\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\alpha^2\ \delta u_0 - \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}-\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\beta\ \delta u_0 + \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}+\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\beta\ \delta u_0 - \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}-\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\alpha\ \gamma\ \delta + \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}+\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\alpha\ \gamma\ \delta - \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}-\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}u_0\ \alpha\ \delta^2 + \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}+\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\alpha\ \delta^2 u_0 - \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}-\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\gamma\ \delta^2 + \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}+\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\gamma\ \delta^2\ + \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}-\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}u_0\ \beta\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2} + \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}+\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}u_0\ \beta\sqrt{\alpha^2+4\beta+2\ \alpha\ \delta+\delta^2} + \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}-\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}u_0\ \alpha\ \delta\sqrt{\alpha^2+4\beta+2\ \alpha\ \delta+\delta^2} + \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}+\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}u_0\ \alpha\ \delta\sqrt{\alpha^2+4\beta+2\ \alpha\ \delta+\delta^2} + \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}-\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\gamma\ \delta\sqrt{\alpha^2+4\beta+2\ \alpha\ \delta+\delta^2} + \\ &e^{t\left(\frac{\alpha}{2}-\frac{\delta}{2}+\frac{1}{2}\sqrt{\alpha^2+4\beta+2\alpha\delta+\delta^2}\right)}\gamma\ \delta\sqrt{\alpha^2+4\beta+2\ \alpha\ \delta+\delta^2}\ \ )\ / \\ &(\ 2\ (\beta+\alpha\ \delta)\sqrt{\alpha^2+4\beta+2\ \alpha\ \delta+\delta^2}\ )\end{aligned} \tag{14}$$

In solution (14);

Case (i): If we choose $\gamma = 0,\ \alpha = 0,\ \beta = -1,\ \delta = 0$, we obtain $u(t) = u_0\ \cos(t)$.

Case (ii): If we choose $\gamma = 1,\ \alpha = 0,\ \beta = -1,\ \delta = 0$, we obtain $u(t) = u_0\ \cos(t) + \sin(t)$.

Case (iii): If we choose $\gamma = 1,\ \alpha = 0,\ \beta = 1,\ \delta = 0$, we obtain $u(t) = u_0\ \cosh(t) + \sinh(t)$.

Case (iv): If we choose $\gamma = 0,\ \alpha = 0,\ \beta = 1,\ \delta = 1$, we obtain $u(t) = u_0\dfrac{e^{-t/2}}{3}\left(3\cos\left(\dfrac{\sqrt{3}}{2}t\right) + \sqrt{3}\sin\left(\dfrac{\sqrt{3}}{2}t\right)\right)$.

The errors found are given Table 1, where *error*=|*true value* – *approximate value*|. Unless otherwise indicated, in this paper, error means absolute error. Table 1 is consistent with the property that the order of the errors are $O(h^4)$ and $O(h^5)$.

4. Conclusion

After above calculation we are expecting order of $O(h^4)$ and $O(h^5)$. In view, it seems to be true because of the truncation error for Runge-Kutta-Fehlberg and Boole's rule are $O(h^4)$ and $O(h^5)$. Numerical order of convergence is also calculated:

$$Ord = \frac{\ln(Error1) - \ln(Error2)}{\ln(2)}$$

We expected that Ord=4 and Ord=5. Obtained theoretical results are confirmed by numerical experiment.

Table 1. *Errors in the Solutions (13) for RKF Method*

t	Error1 with h=0.0250		Error2 with h=0.0125		Error3 with h=0.00625	
	Method A	Method B	Method A	Method B	Method A	Method B
0.1	4.8696e-08	7.7084e-10	3.0410e-09	2.4070e-11	1.8994e-10	7.5163e-13
0.2	4.8142e-08	7.6159e-10	3.0009e-09	2.3731e-11	1.8726e-10	7.4030e-13
0.3	4.7108e-08	7.4415e-10	2.9308e-09	2.3137e-11	1.8271e-10	7.2092e-13
0.4	4.5603e-08	7.1872e-10	2.8315e-09	2.2295e-11	1.7634e-10	6.9383e-13
0.5	4.3643e-08	6.8557e-10	2.7039e-09	2.1213e-11	1.6820e-10	6.5936e-13
0.6	4.1247e-08	6.4505e-10	2.5492e-09	1.9903e-11	1.5839e-10	6.1773e-13
0.7	3.8439e-08	5.9760e-10	2.3691e-09	1.8379e-11	1.4699e-10	5.6932e-13
0.8	3.5248e-08	5.4373e-10	2.1653e-09	1.6657e-11	1.3412e-10	5.1514e-13
0.9	3.1705e-08	4.8402e-10	1.9400e-09	1.4756e-11	1.1992e-10	4.5519e-13
1.0	2.7845e-08	4.1910e-10	1.6952e-09	1.2697e-11	1.0451e-10	3.9047e-13

A: ($\gamma = 0,\ \alpha = 0,\ \beta = -1,\ \delta = 0\ \ u_0 = 0,\ t_0 = 0,\ t_{\max} = 1$) gives $O(h^4)$.

B: ($\gamma = 1,\ \alpha = 0,\ \beta = -1,\ \delta = 0\ \ u_0 = 0,\ t_0 = 0,\ t_{\max} = 1$) gives $O(h^5)$.

The fifth order Runge-Kutta-Fehlberg Method (RKF) and numerical quadrature rules (gives error $O(h^4)$ and $O(h^5)$).

References

[1] M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions: with Formulas, Graphs and Mathematical Tables, New York: Dover, 1972, pp. 885–887.

[2] A. Asanov, Uniqueness of the solution of systems of convolution-type Volterra integral equations of the first kind, In: Inverse problems for differential equations of the mathematical physics (Russian), Novasibirsk: Akad. Nauk SSSR Sibirsk. Otdel. Vychil. Tsentr, 1978, Vol 155, pp. 2–34.

[3] R. L. Burden and J. D. Faires, Numerical Analysis, New York: Brooks/Cole Publishing Company, USA, 1997, ch.5.

[4] C. T. H. Baker, The Numerical Treatment of Integral Equations, Clarendon Press; Oxford University Press, 1977.

[5] C. T. H. Baker, G. A. Bochorov, A. Filiz, N. J. Ford, C. A. H. Paul, F. A. Rihan, A. Tang, R. M. Thomas, H. Tian, D. R. Wille "Numerical Modelling by Retarded Functional Differential Equations," Numerical Analysis Report, Manchester Center for Computational Mathematics, No:335, ISS 130-1725,1998.

[6] C. T. H. Baker, G. A. Bochorov, A. Filiz, N. J. Ford, C. A. H. Paul, F. A. Rihan, A. Tang, R. M. Thomas, H. Tian, D. R. Wille "Numerical Modelling by Delay and Volterra Functional Differential Equations," Numerical Analysis Report, In: Computer Mathematics and its Aplications-Advances & Developments (1994-2005), Elias A. Lipitakis (Editor), LEA Publishers, Athens, Greece, 2006, pp. 233-256.

[7] R. Bellman, A Survey of the Theory of the Boundedness Stability and Asymptotic Behaviour of Solutions of Linear and Non-linear differential and difference equations, Washington, D. C., 1949.

[8] K. L. Cooke, "Functional differential equations close to cifferential equation," Amer. Math. Soc., 1966, Vol.72, pp. 285-288.

[9] A. Filiz, "On the solution of Volterra and Lotka-Volterra Type Equations," LMS supported One Day Meeting in Delayed Differential equation (Liverpool, UK), 12th March 2000.

[10] A. Filiz, "Numerical Solution of Some Volterra Integral Equations," PhD Thesis, The University of Manchester, 2000.

[11] A. Filiz, "Fourth-order robust numerical method for integro-differential equations," Asian Journal of Fuzzy and Applied Mathematics, 2013, Vol. 1 I, pp. 28-33.

[12] P. Linz, Analytical and Numerical Methods for Volterra Equations, SIAM, Philadelphia, 1985.

[13] C. W. Ueberhuber, Numerical Computation 2: Methods, Software and analysis, Berlin: Springer-Verlag, 1997.

[14] V. Volterra, Leçons Sur la Theorie Mathematique de la Lutte Pour La Vie, Gauthier-villars, Paris, 1931.

[15] V. Volterra, Theory of Functional and of Integro-Differential Equations. Dover, New York, 1959.

[16] V. Volterra, "Sulle Equazioni Integro-differenziali Della Teoria Dell'elastica," Atti Della Reale Accademia dei Lincei 18 (1909), Reprinted in Vito Volterra, Opera Mathematiche; Memorie e Note, Vol. 3, Accademia dei Lincei Rome, 1957.

[17] Wolfram MathWorld, Newton-Cotes Formulas, http://mathworld.wolfram.com/Newton-CotesFormulas.html