Numerical Method for a Linear Volterra Integro-differential Equation with Cash-Karp Method

Ali Filiz[1]

[1]Department of Mathematics
Faculty of Science
Adnan Menderes University
09010 AYDIN-Turkey

ABSTRACT— ***In this paper a linear Volterra integro-differential equation is studied. Example of this question has been solved numerically using Cash-Karp method for ODE (Ordinary Differential Equation) parts and Newton-Cotes formulae (quadrature rules) for integral part. Finally, a new fifth order routine is used for the numerical solution of the linear Volterra integro-differential equation***.

Keywords— Volterra integro-differential (integral) equation, Cash-Karp method, Runge-Kutta Method, Truncation error, quadrature rule, fifth order.

1. INTRODUCTION

This paper presents some numerical methods for the solution of linear integro-differential equation namely; Runge-Kutta method of order five. Runge-Kutta method attempts to obtain greater accuracy. Numerical experiments were taken into consideration to determine the accuracy of the methods. An explicit Runge-Kutta method of order five is more accurate than the third orders and fourth orders Runge-Kutta methods (see [22, 23, 18]). In this paper, we shall only focus on the fifth order Runge-Kutta method. We will derive here the numerical solution of the linear Volterra integro-differential equation of the fifth-order Runge method using a following form:

$$u'(t) = F\left(t, u(t), \int_0^t K(t,s)u(s)ds\right), \qquad u(0) = u_0, \quad t \geq 0. \tag{1}$$

The numerical solution of the ordinary differential equation

$$u'(t) = f(t,u(t)), \quad t \geq 0; \qquad u(t_0) = u_0, \qquad \text{which take the form} \tag{2}$$

$$\tilde{u}_{n+1} = \tilde{u}_n + \sum_{i=1}^{s} b_i k_i, \quad \text{and} \quad k_i = h(t_n + c_i h, \tilde{u}_n + \sum_{j=1}^{s} a_{ij} k_j) \tag{3}$$

The methods listed on this page are each defined by its Butcher tableau, which puts the coefficients of the method in a table as follows:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} \tag{4}$$

2. EXPICIT METHODS

The explicit methods are those where the matrix $[a_{ij}]$ is lower triangular.

The third-order Runge-Kutta method: This method is a third order Runge-Kutta method for approximating the solution of the initial value problem (2) which evaluates the integrand, *f(t,u(t))*, three times per step.

0	0	0	0
1/2	1/2	0	0
1	−1	2	0
	1/6	2/3	1/6

$$\tilde{u}_{n+1} = \tilde{u}_n + \frac{1}{6}(k_1 + 4k_2 + k_3),$$

where

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

$$k_2 = hf(t_n + \frac{1}{2}h, \tilde{u}_n + \frac{1}{2}k_1),$$

$$k_3 = hf(t_n + h, \tilde{u}_n - k_1 + 2k_2),$$

Classical fourth-order Runge-Kutta method: The Butcher tableau corresponding to the Rk4 method

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

the equivalent corresponding equations defining the classical RK4 method:

$$\tilde{u}_{n+1} = \tilde{u}_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

where

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

$$k_2 = hf(t_n + \frac{1}{2}h, \tilde{u}_n + \frac{1}{2}k_1),$$

$$k_3 = hf(t_n + \frac{1}{2}h, \tilde{u}_n + \frac{1}{2}k_2),$$

$$k_4 = hf(t_n + h, \tilde{u}_n + k_3),$$

Recently, the fourth order classical Runge-Kutta method (RK4) has been adapted to the numerical solution of equation (1) by Filiz (see [18]).

3/8-rule fourth-order method: This method doesn't have as much notoriety as the "classical" method, but is just as classical because it was proposed in the same paper (see [21]).

0	0	0	0	0
1/3	1/3	0	0	0
2/3	−1/3	1	0	0
1	1	−1	1	0
	1/8	3/8	3/8	1/8

$$\tilde{u}_{n+1} = \tilde{u}_n + \frac{1}{8}(k_1 + 3k_2 + 3k_3 + k_4),$$

where

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

$$k_2 = hf(t_n + \frac{1}{3}h, \tilde{u}_n + \frac{1}{3}k_1),$$

$$k_3 = hf(t_n + \frac{2}{3}h, \tilde{u}_n - \frac{1}{3}k_1 + k_2),$$

$$k_4 = hf(t_n + h, \tilde{u}_n + k_1 - k_2 + k_3),$$

3. EMBEDDED METHODS

The embedded methods are designed to produce an estimate of the local truncation error of a single Runge-Kutta step, and as result, allow controlling the error with adaptive stepsize. This is done by having two methods in the tableau, one with order p and one with order p-1.

3.1 Bogacki–Shampine

The Bogacki–Shampine method is a method for the numerical solution of ordinary differential equations, which was proposed by Przemyslaw Bogacki and Lawrence F. Shampine in 1989 ([12]). The Bogacki–Shampine method has two methods of orders 3 and 2. Its extended Butcher Tableau is:

0				
1/2	1/2			
3/4	0	3/4		
1	2/9	1/3	4/9	
	2/9	1/3	4/9	0
	7/24	1/4	1/3	1/8

The first row of *b* coefficients gives the third-order accurate solution, and the second row has order two.

Following the standard notation, the differential equation to be solved is $u'(t) = f(t, u(t))$. Furthermore, $\tilde{u}(t_n)$ or $\tilde{u}_n$ denotes the numerical solution at time t_n and h is the step size, defined by $h = t_{k+1} - t_k$ or $h = (t_n - t_0)/n$. Then, one step of the Bogacki–Shampine method is given by:

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

$$k_2 = hf\left(t_n + \frac{1}{2}h, \tilde{u}_n + \frac{1}{2}k_1\right),$$

$$k_3 = hf\left(t_n + \frac{3}{4}h, \tilde{u}_n + \frac{3}{4}k_2\right),$$

$$\tilde{u}_{n+1} = \tilde{u}_n + \frac{2}{9}k_1 + \frac{1}{3}k_2 + \frac{4}{9}k_3$$

$$k_4 = hf(t_n + h, \tilde{u}_{n+1}) = hf\left(t_n + h, \tilde{u}_n + \frac{2}{9}k_1 + \frac{1}{3}k_2 + \frac{4}{9}k_3\right),$$

$$(\tilde{u}_{n+1})^* = \tilde{u}_n + \frac{7}{24}k_1 + \frac{1}{4}k_2 + \frac{1}{3}k_3 + \frac{1}{8}k_4,$$

Here, $(\tilde{u}_{n+1})^*$ is a second-order approximation to the exact solution. The method for calculating $\tilde{u}_{n+1}$ is due to [13]. On the other hand, $\tilde{u}_{n+1}$ is a third-order approximation, so the difference between $\tilde{u}_{n+1}$ and $(\tilde{u}_{n+1})^*$ can be used to adapt the step size.

3.2 Dorman-Prince Method

In numerical analysis, the **Dormand–Prince method,** or DOPRI method, is an explicit method for solving ordinary differential equations ([14]). The method is a member of the Runge–Kutta family of ODE solvers. More specifically, it uses six function evaluations to calculate fourth- and fifth-order accurate solutions. This error estimate is very convenient for adaptive stepsize integration algorithms. Other similar integration methods are Fehlberg (RKF) and Cash–Karp (RKCK).

Dormand and Prince choose the coefficients of their method to minimize the error of the fifth-order solution. This is the main difference with the Fehlberg method, which was constructed so that the fourth-order solution has a small error. For

this reason, the Dormand–Prince method is more suitable when the higher-order solution is used to continue the integration, a practice known as local extrapolation [14]. The extended tableau for the Dormand–Prince method is:

0							
1/5	1/5						
3/10	3/40	9/40					
4/5	44/45	−56/15	32/9				
8/9	19372/6561	−25360/2187	64448/6561	−212/729			
1	9017/3168	−355/33	46732/5247	49/176	−5103/18656		
1	35/384	0	500/1113	125/192	−2187/6784	11/84	
	5179/57600	0	7571/16695	393/640	−92097/339200	187/2100	1/40
	35/384	0	500/1113	125/192	−2187/6784	11/84	0

The first row of *b* coefficients gives the fourth-order accurate solution, and the second row has order five.

3.3 Runge–Kutta–Fehlberg method

Runge–Kutta–Fehlberg method (RKF5) is an algorithm in numerical analysis for the numerical solution of ordinary differential equations. The novelty of Fehlberg's method is that it is an embedded method from the Runge-Kutta family, meaning that identical function evaluations are used in conjunction with each other to create methods of varying order and similar error constants. The method presented in Fehlberg's 1969 paper has been dubbed the **RKF45** method, and is a method of order O(h^4) with an error estimator of order O(h^5) ([16]). Its extended Butcher Tableau is:

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	
	25/216	0	1408/2565	2197/4104	−1/5	0
	16/135	0	6656/12825	28561/56430	−9/50	2/55

The first row of *b* coefficients gives the fourth-order accurate solution, and the second row has order five. The fifth order Runge–Kutta–Fehlberg (RKF5) has been adapted to the numerical solution of equation (1) by Filiz (see [19]).

3.4 Cash-Karp Method

In numerical analysis, the Runge-Kutta-Cash–Karp method (RKCK) is a method for solving ordinary differential equations (ODEs) (17). The method is a member of the Runge–Kutta family of ODE solvers. More specifically, it uses six function evaluations to calculate fourth- and fifth-order accurate solutions. The difference between these solutions is then taken to be the error of the (fourth order) solution. This error estimate is very convenient for adaptive stepsize integration algorithms. Other similar integration methods are Fehlberg (RKF) and Dormand–Prince (RKDP). Cash and Karp have modified Fehlberg's original idea. The extended tableau for the Cash–Karp method is:

(5)

0						
1/5	1/5					
3/10	3/40	9/40				
3/5	3/10	−9/10	6/5			
1	−11/54	5/2	− 0/27	35/27		
7/8	1631/55296	175/512	575/13824	44275/110592	253/4096	
	37/378	0	250/621	125/594	0	512/1771
	2825/27648	0	18575/48384	13525/55296	277/14336	1/4

The first row of *b* coefficients at the bottom of the table gives the fifth-order accurate method, and the second row gives the fourth-order accurate method. An integral equation is an equation in which the function to be determined appears under the integral sign. If we consider linear integral equations, that is, equations in which no nonlinear functions of the unknown function are involved. Equation (1) can be solved numerically using various methods. In this paper $u(t_n)$ will denote the exact value of *u(t)* at $t_n = t_0 + nh$. We shall use $\tilde{u}(t_n)$ or $\tilde{u}_n$ to denote an approximate value of u at t_n. However, in this paper we will define fifth order numerical methods for (1). Since the integral cannot be determined explicitly, it may be approximated using familiar numerical integration methods. The Newton-Cotes integration formulas, which include the trapezoidal rule and Simpson's 1/3, are well suited here since they use nodes which were given in [8] and [1].

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, 7, 13,18] and [3, 8].

4. THE NUMERICAL SOLUTION OF INTEGRO-DIFFERENTIAL EQUATIONS

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

$$\tilde{z}(t_n) := h\sum_{j=0}^{n} \omega_{n,j} K(t_n, t_j)\, \tilde{u}(t_j) \approx \int_{t_0}^{t_n} K(t,s)\, u(s)\, ds. \tag{6}$$

Stepping from $\tilde{u}_n$ with step-size h to obtain $\tilde{u}_{n+1}$, the RK4 method as applied to this problem (1) in [7, 18]. 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 $[t_n, t_{n+1}]$. Any Runge-Kutta method is uniquely identified by its Butcher Tableau in (5). The embedded pair proposed by Fehlberg [16].

The Runge-Kutta-Cash-Karp Method (denoted RKCK) is one way to try to resolve this problem. 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. The Runge-Kutta-Cash-Karp can also be adapted to the numerical solution of (1). Stepping from $\tilde{u}_n$ with step-size h to obtain $\tilde{u}_{n+1}$, the RKCK method as applied to this problem may be written as:

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

$$\tilde{u}^a_{n+1/5} = \tilde{u}_n + \frac{1}{5}k_1,$$

$$k_2 = hF(t_{n+1/5}, \tilde{u}^a_{n+1/5}, \tilde{z}_{n+1/5}),$$

$$k_2 = hF\left(t_{n+1/5}, \tilde{u}^a_{n+1/5}, \tilde{z}_n + \frac{h}{10}\left[\tilde{u}_n + \tilde{u}^a_{n+1/5} \right] \right),$$

$$\tilde{u}^b_{n+3/10} = \tilde{u}_n + \frac{3}{40}k_1 + \frac{9}{40}k_2,$$

$$k_3 = hF(t_{n+3/10}, \tilde{u}^b_{n+3/10}, \tilde{z}_{n+3/10}\ ),$$

$$k_3 = hF\left(t_{n+3/10}, \tilde{u}^b_{n+3/10}, \tilde{z}_n + \frac{3h}{20}\left[\tilde{u}_n + \tilde{u}^b_{n+3/10} \right] \right),$$

$$\tilde{u}^c_{n+3/5} = \tilde{u}_n + \frac{3}{10}k_1 - \frac{9}{10}k_2 + \frac{6}{5}k_3,$$

$$k_4 = hF(t_{n+3/5}, \tilde{u}^c_{n+3/5}, \tilde{z}_{n+3/5}\ ),$$

$$k_4 = hF\left(t_{n+3/5}, \tilde{u}^c_{n+3/5}, \tilde{z}_n + \frac{3h}{10}\left[\tilde{u}_n + \tilde{u}^c_{n+3/5} \right] \right), \quad (7)$$

$$\tilde{u}^d_{n+1} = \tilde{u}_n - \frac{11}{54}k_1 + \frac{5}{2}k_2 - \frac{70}{27}k_3 + \frac{35}{27}k_4,$$

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

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

$$\tilde{u}^e_{n+7/8} = \tilde{u}_n + \frac{1631}{55296}k_1 + \frac{175}{512}k_2 + \frac{575}{13824}k_3 + \frac{44275}{110592}k_4 + \frac{253}{4096}k_5,$$

$$k_6 = hF(t_{n+7/8}, \tilde{u}^e_{n+7/8}, \tilde{z}_{n+7/8}\ ),$$

$$k_6 = hF\left(t_{n+7/8}, \tilde{u}^e_{n+7/8}, \tilde{z}_n + \frac{7h}{16}\left[\tilde{u}_n + \tilde{u}^e_{n+7/8} \right] \right),$$

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

$$\tilde{u}_{n+1} = \tilde{u}_n + \frac{2825}{27648}k_1 + \frac{18575}{48384}k_3 + \frac{13525}{55296}k_4 + \frac{277}{14336}k_5 + \frac{1}{4}k_6, \quad (8)$$

where the four function values k_1, k_3, k_4 and k_5 are used. A better value for the solution is determined using a Runge-Kutta Method of order 5:

$$\tilde{u}_{n+1} = \tilde{u}_n + \frac{37}{378}k_1 + \frac{250}{621}k_3 + \frac{125}{594}k_4 + \frac{512}{1771}k_6. \quad (9)$$

In this example, the trapezoidal rule is used to approximate $\tilde{z}(t_n) \approx \int_0^{t_n} k(t-s)u(s)\,ds$ on $[t_n, t_{n+1/5}]$, $[t_n, t_{n+3/10}]$, $[t_n, t_{n+3/5}]$, $[t_n, t_{n+1}]$, $[t_n, t_{n+7/8}]$ 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 [7, 8]) may be used on $[t_0, t_n]$.

In order to get **fifth** order accuracy the integral term must be evaluated more accurately on $[t_n, t_{n+1/5}]$, $[t_n, t_{n+3/10}]$, $[t_n, t_{n+3/5}]$, $[t_n, t_{n+1}]$, $[t_n, t_{n+7/8}]$ in calculating, k_2, k_3, k_4, k_5 and k_6, as shown in (10)-(14) below.

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

z(1)=0

u(1)=u0

If n=1

z(n+1)= z(n) + h(u(n) + u(n+1)) /2

elseif n==2

z(n+1)= z(n-1) + h(u(n-1) +4 u(n) + u(n+1)) /3

elseif n==3

z(n+1)= z(n-2) +3h (u(n-2) +3 u(n-1) +3u(n) + u(n+1)) / 8

elseif n==4

z(n+1)= z(n-3) +2h (7 u(n-3) +32 u(n-2) +12u(n-1) + 32 u(n) +7 u(n+1)) / 45

elseif n==5

z(n+1)= z(n-4) +5h (19 u(n-4) +75 u(n-3) +50 u(n-2)+50 u(n-1) + 75 u(n) +19 u(n+1)) / 288

elseif n==6

z(n+1)= z(n-5) + h (41u(n-5)+ 216 u(n-4) +27 u(n-3) +272 u(n-2)+27 u(n-1) + 216 u(n) +41 u(n+1)) / 140

elseif n==7

z(n+1)= z(n-6) + 7h (751u(n-6)+3577u(n-5)+ 1323 u(n-4) +2989 u(n-3) +2989 u(n-2)+1323 u(n-1) +…

3577 u(n) +7511 u(n+1)) / 17280

elseif n==8

z(n+1)= z(n-3) + 2h (7 u(n-3) +32 u(n-2) +12u(n-1) + 32 u(n) +7 u(n+1)) / 45

elseif mod(n,4)==0

z(n+1)= z(n-3) + 2h (7 u(n-3) +32 u(n-2) +12u(n-1) + 32 u(n) +7 u(n+1)) / 45

elseif mod(n,4)==1

z(n+1)= z(n-3) + 2h (7 u(n-3) +32 u(n-2) +12u(n-1) + 32 u(n) +7 u(n+1)) / 45

elseif mod(n,4)==2

z(n+1)= z(n-3) + 2h (7 u(n-3) +32 u(n-2) +12u(n-1) + 32 u(n) +7 u(n+1)) / 45

elseif mod(n,4)==3

z(n+1)= z(n-3) + 2h (7 u(n-3) +32 u(n-2) +12u(n-1) + 32 u(n) +7 u(n+1)) / 45

else

z(n+1)= z(n-3) + 2h (7 u(n-3) +32 u(n-2) +12u(n-1) + 32 u(n) +7 u(n+1)) / 45

If we interpolating on $\tilde{u}_{n-2}, \tilde{u}_{n-1}, \tilde{u}_n, \tilde{u}_{n+1/5}$ (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, 1/5 gives

$$u(t) = \frac{1}{h^3}\left[-\frac{5}{22}t\,(t-\frac{h}{5})\,(t+h)u_{-2} + \frac{5}{6}t\,(t-\frac{h}{5})(t+2h)u_{-1} - \frac{5}{2}(t+h)(t+2h)(t-\frac{h}{5})u_0 + \frac{125}{66}t(t+h)(t+2h)u_{1/5}\right].$$

If we integrate the expression between 0 and h/5, we get

$$(10)\quad \int_0^{h/5} u(s)ds \approx h\left(\frac{11}{120}u_{1/5} + \frac{1}{3000}u_{-2} - \frac{7}{3000}u_{-1} + \frac{331}{3000}u_0\right).$$

Similarly, we can find t= t=-2, -1, 0, 3/10

$$(11)\quad \int_0^{3h/10} u(s)ds \approx h\left(\frac{57}{352}u_{3/8} + \frac{9}{4096}u_{-2} - \frac{315}{22528}u_{-1} + \frac{921}{496}u_0\right),$$

and find t= t=-2, -1, 0, 3/5

$$(12)\quad \int_0^{3h/5} u(s)ds \approx h\left(\frac{114}{325}u_{12/13} + \frac{72}{2197}u_{-2} - \frac{9216}{54925}u_{-1} + \frac{1554}{2197}u_0\right),$$

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

$$(13)\quad \int_0^{h} u(s)ds \approx h\left(\frac{3}{8}u_1 + \frac{1}{24}u_{-2} - \frac{5}{24}u_{-1} + \frac{19}{24}u_0\right),$$

and finally find t= t=-2, -1, 0, 7/8

$$(14)\quad \int_0^{7h/8} u(s)ds \approx h\left(\frac{5}{24}u_{1/2} + \frac{1}{192}u_{-2} - \frac{1}{32}u_{-1} + \frac{61}{192}u_0\right).$$

when n<3 the first two approximations can be found by formula (7). Therefore the Runge-Kutta-Crash-Karp formulae become (for starting values, we can use (16) and (17)).

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

$$\tilde{u}^a_{n+1/4} = \tilde{u}_n + \frac{1}{5}k_1,$$

$$k_2 = hF(t_{n+1/5}, \tilde{u}^a_{n+1/5}, \tilde{z}_{n+1/5}\ ),$$

$$k_2 = hF\left(t_{n+1/5}, \tilde{u}^a_{n+1/5}, \tilde{z}_n + h\left[\frac{9}{80}\tilde{u}^a_{n+1/5} + \frac{1}{1536}\tilde{u}_{n-2} - \frac{17}{3840}\tilde{u}_{n-1} + \frac{217}{1536}\tilde{u}_n\right]\right),$$

$$\tilde{u}^b_{n+3/10} = \tilde{u}_n + \frac{3}{40}k_1 + \frac{9}{40}k_2,$$

$$k_3 = hF(t_{n+3/10}, \tilde{u}^b_{n+3/10}, \tilde{z}_{n+3/10}\ ),$$

$$k_3 = hF\left(t_{n+3/10}, \tilde{u}^b_{n+3/10}, \tilde{z}_n + h\left[\frac{57}{352}\tilde{u}^b_{n+3/10} + \frac{9}{4096}\tilde{u}_{n-2} - \frac{315}{22528}\tilde{u}_{n-1} + \frac{921}{4096}\tilde{u}_n\right]\right),$$

$$\tilde{u}^c_{n+3/5} = \tilde{u}_n + \frac{3}{10}k_1 - \frac{9}{10}k_2 + \frac{6}{5}k_3,$$

$$k_4 = hF(t_{n+3/5}, \tilde{u}^c_{n+3/5}, \tilde{z}_{n+3/5}\ ),$$

$$k_4 = hF\left(t_{n+3/5}, \tilde{u}^c_{n+3/5}, \tilde{z}_n + h\left[\frac{114}{325}\tilde{u}^c_{n+3/5} + \frac{721}{2197}\tilde{u}_{n-2} - \frac{9216}{54925}\tilde{u}_{n-1} + \frac{1554}{2197}\tilde{u}_n\right]\right),$$

$$\tilde{u}^d_{n+1} = \tilde{u}_n - \frac{1}{54}k_1 + \frac{5}{2}k_2 - \frac{70}{27}k_3 + \frac{35}{27}k_4,$$

$$(15)\quad k_5 = hF(t_{n+1}, \tilde{u}^d_{n+1}, \tilde{z}_{n+1}),$$

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

$$\tilde{u}^e_{n+7/8} = \tilde{u}_n + \frac{1631}{55296}k_1 + \frac{175}{512}k_2 + \frac{575}{13824}k_3 + \frac{44275}{110592}k_4 + \frac{253}{4096}k_5,$$

$$k_6 = hF(t_{n+7/8}, \tilde{u}^e_{n+7/8}, \tilde{z}_{n+7/8}\ ),$$

$$k_6 = hF\left(t_{n+7/8}, \tilde{u}^e_{n+7/8}, \tilde{z}_n + h\left[\frac{5}{24}\tilde{u}^e_{n+7/8} + \frac{1}{192}\tilde{u}_{n-2} - \frac{1}{32}\tilde{u}_{n-1} + \frac{61}{192}\tilde{u}_n\right]\right),$$

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

$$(16) \qquad \tilde{u}_{n+1} = \tilde{u}_n + \frac{2825}{27648}k_1 + \frac{18575}{48384}k_3 + \frac{13525}{55296}k_4 + \frac{277}{14336}k_5 + \frac{1}{4}k_6,$$

In (17), this technique consistent of using the RKCK Method with local truncation error of order five,

$$(17) \qquad \tilde{u}_{n+1} = \tilde{u}_n + \frac{37}{378}k_1 + \frac{250}{621}k_3 + \frac{125}{594}k_4 + \frac{512}{1771}k_6.$$

We can construct an algorithm similar to the sixth, seventh, eighth and tenth-order Runge-Kutta formulae and we can repeat Example 3.1 using this new method. Table 3 shows the fifth-order accuracy obtained with this formula.

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

$$(18) \qquad u'(t) = \gamma + \alpha u(t) + \beta \int_{t_0}^{t} e^{-\delta(t-s)} u(s)\, ds, \quad t \geq 0; \qquad u(t_0) = u_0.$$

Equation (18) can be solved analytically with Laplace Transform using symbolic programming. In (18):

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, Table 2 and Table 3, where ***error=|true value – approximate value|***. Unless otherwise indicated, in this paper, error means absolute error.

Table 1: Errors in the Solutions (18) for the third-order Runge-Kutta method
$\gamma = 0,\ \alpha = 0,\ \beta = -1,\ \delta = 0\ \ u_0 = 0,\ t_0 = 0,\ t_{\max} = 1)$ gives $O(h^3)$.

	h=0.0250	**h=0.0125**	**h=0.00625**
t	**Errror1**	**Errror2**	**Errror3**
0.1	5.3647e-09	7.4154e-10	9.7125e-11
0.2	2.3628e-08	3.0936e-09	3.9545e-10
0.3	5.4434e-08	7.0098e-09	8.8905e-10
0.4	9.7172e-08	1.2412e-08	1.5680e-09
0.5	1.5099e-07	1.9191e-08	2.4187e-09
0.6	2.1480e-07	2.7211e-08	3.4239e-09
0.7	2.8730e-07	3.6307e-08	4.5629e-09
0.8	3.6701e-07	4.6291e-08	5.8122e-09
0.9	4.5224e-07	5.6953e-08	7.1454e-09
1.0	5.4120e-07	6.8067e-08	8.5342e-09

Table 1 is consistent with the property that the order of the error is $O(h^3)$.

Table 2: Errors in the Solutions (18) for the fourth order RK 3/8-rule

($\mu=1,\ \lambda=-1,\ \delta=0,\ \ u_0=0,\ t_0=0,\ t_{\max}=1$) gives $O(h^4)$.

	h=0.0250	**h=0.0125**	**h=0.00625**
t	**Error1**	**Error 2**	**Error3**
0.1	7.3484e-10	3.3686e-11	1.7235e-12
0.2	1.0657e-09	5.4461e-11	3.0259e-12
0.3	1.3753e-09	7.4015e-11	4.2551e-12
0.4	1.6552e-09	9.1811e-11	5.3775e-12
0.5	1.8973e-09	1.0734e-10	6.3615e-12
0.6	2.0941e-09	1.2014e-10	7.1770e-12
0.7	2.2388e-09	1.2976e-10	7.7981e-12
0.8	2.3255e-09	1.3585e-10	8.2002e-12
0.9	2.3491e-09	1.3808e-10	8.3635e-12
1.0	2.3057e-09	1.3619e-10	8.2722e-12

Table 2 is consistent with the property that the order of the error is $O(h^3)$.

Table 3: Errors in the Solutions (18) for RKCK Method

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 $O(h^5)$.

t	***Error1 with h=0.0125***		***Error2 with h=0.00625***		***Error3 with h=0.003125***	
	Method A	Method B	Method A	Method B	Method A	Method B
0.1	3.0410e-09	2.4070e-11	1.8994e-10	7.5163e-13	1.1867e-11	2.3495e-14
0.2	3.0009e-09	2.3731e-11	1.8726e-10	7.4032e-13	1.1694e-11	2.3120e-14
0.3	2.9309e-09	2.3138e-11	1.8271e-10	7.2098e-13	1.1404e-11	2.2538e-14
0.4	2.8315e-09	2.2297e-11	1.7634e-10	6.9394e-13	1.1001e-11	2.1705e-14
0.5	2.7039e-09	2.1216e-11	1.6820e-10	6.5947e-13	1.0487e-11	2.0706e-14
0.6	2.5492e-09	1.9907e-11	1.5839e-10	6.1784e-13	9.8699e-12	1.9207e-14
0.7	2.3691e-09	1.8384e-11	1.4699e-10	5.6954e-13	9.1529e-12	1.7542e-14
0.8	2.1654e-09	1.6663e-11	1.3412e-10	5.1537e-13	8.3447e-12	1.5987e-14
0.9	1.9400e-09	1.4764e-11	1.1992e-10	4.5552e-13	7.4529e-12	1.3878e-14
1.0	1.6952e-09	1.2706e-11	1.0451e-10	3.9080e-13	6.4864e-12	1.1768e-14

Table 3 is consistent with the property that the order of the errors are $O(h^4)$ and $O(h^5)$.

4. CONCLUSION

In this paper, some numerical methods for the solution of linear Volterra integro-differential equation have been developed. Runge-Kutta method is good choice to get more accurate and more efficient solutions for solving the specified linear Volterra integro-differential equation. The approximated solution converges faster to exact solution and the order of classical Runge Kutta method is five.

In fact, after this numerical calculation we were expecting order of $O(h^5)$. In view, it seems to be true because of the truncation error for the fifth order Runge-Kutta-Crash-Karp (RKCK) and Boole's rule is $O(h^5)$. Thus, we found the expected $O(h^5)$. Numerical order of convergence is also calculated:

$$Ord = \frac{\ln(Error_1) - \ln(Error_2)}{\ln(2)}.$$

We expected that Ord=5. Obtained theoretical results are confirmed by numerical experiments.

5. REFERENCES

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

[2] Baker, C. T. H., G A Bochorov, A Filiz, N J Ford, C A H Paul, F A Rihan, A Tang, R M Thomas, H Tian and D R Wille., Numerical Modelling By Retarded Functional Differential Equations, Numerical analysis report, Manchester Centre for Computational Mathematics, No:335, ISSN 1360-1725, 1998.

[3] Baker, C. T. H., G A Bochorov, A Filiz, N J Ford, C A H Paul, F A Rihan, A Tang, R M Thomas, H Tian and D R Wille., Numerical modelling by delay and Volterra functional differential equations, In: Computer Mathematics and its Applications- Advances and Developments (1994-2005), Editor: Elias A. Lipitakis, LEA Publishers, Athens, Greece, 2006.

[4] Bellman, R., A survey of the theory of the boundedness stability and asymptotic behaviour of solutions of linear and non-linear differential and difference equations, Washington, 1949.

[5] Cooke, K. L., Functional differential equations close to differential equation, Amer. Math. Soc., 72, 285-288, 1966.

[6] Filiz, A., On the solution of Volterra and Lotka-Volterra type equations, LMS supported One Day Meeting in Delayed Differential Equation (Liverpool, UK), 12th March 2000.

[7] Filiz, A., Numerical solution of some Volterra integral equations, PhD Thesis, University of Manchester, 2000.

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

[9] Volterra, V. (1931)., Le¸cons sur la Th´eorie Math´ematique de la Lutte Pour la Vie, Gauthier-Villars, Paris, 1931.

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

[11] Volterra, V., 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, pp. 288-293, 1957.

[12] Bogacki, Przemyslaw and Shampine, Lawrence F., A 3(2) pair of Runge–Kutta formulas, *Applied Mathematics Letters*, **2** (4), pp. 321–325, 1989.

[13] Ralston, Anthony, *A First Course in Numerical Analysis*, New York: McGraw-Hill, 1965.

[14] Dormand, J. R.; Prince, P. J., A family of embedded Runge-Kutta formulae, *Journal of Computational and Applied Mathematics*, **6** (1): 19–26, 1980.

[15] Hairer, Ernst; Nørsett, Syvert Paul; Wanner, Gerhard, *Solving ordinary differential equations I: Nonstiff problems*, Berlin, New York: Springer-Verlag, 2008.

[16] Erwin Fehlberg, *Low-order classical Runge-Kutta formulas with step size control and their application to some heat transfer problems*, NASA Technical Report 315, 1969.

[17] Cash, J. R. and A. H. Karp., A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides, *ACM Transactions on Mathematical Software*, **16**: 201-222, 1990.

[18] Filiz, A., Fourth-Order Robust Numerical Method for Integro-differential Equations, Asian Journal of Fuzzy and Applied Mathematics, 2013, Vol 1, pp. 28-33.

[19] Filiz, A., Numerical Solution of Linear Volterra Integro-differential Equation using Runge-Kutta-Fehlberg Method, Applied and Computational Mathematics (ACM), submitted.

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

[21] Kutta, W., Beitrag zur näherungsweisen Integration totaler Differentialgleichungen, Z. Math. Phys., 435-453, 1901.

[22] Henrici, P., Discrete variable methods in ordinary differential equation, John Wiley and Sons, New York, 1962.

[23] Abraham, O. and G. Bolarin, On error estimation in Runge-Kutta Methods, Leonardo J. Sci., 2011 ijs.academicdirect.org / A18/ 001_010.htm.