Journal of Mathematical Modeling
Vol. 9, No. 3, 2021, pp. 361-373

JMM

Recent advances in the numerical solution of Volterra integral equations

Ali Abdi[†][*]

[†]*Faculty of Mathematical Sciences, University of Tabriz, Tabriz, Iran*

Email(s): a_abdi@tabrizu.ac.ir

Abstract. Natural Volterra Runge–Kutta methods and general linear methods are two large family of the methods which have recently attracted more attention in the numerical solution of Volterra integral equations. The purpose of the paper is the presentation of some recent advances in these methods. Also, implementation issues for these methods will be discussed.

Keywords: Volterra integral equations, general linear methods, natural Volterra Runge–Kutta methods, Nordsieck technique, implementation issues.
AMS Subject Classification 2010: 65R20.

1 Introduction

The modeling of certain physical phenomena in which a quantity varies in time and simultaneously depends on its past values can usually be modeled by a system of Volterra integral equations (VIEs). A classical system of VIEs of the second kind is therefore often assumed to be

$$y(t) = g(t) + \int_{t_0}^{t} K(t,s,y(s))\,ds, \qquad t \in I := [t_0, T]. \tag{1}$$

Here the *forcing function* $g : I \to \mathbb{R}^m$ and the *kernel* $K : \Delta \times \mathbb{R}^m \to \mathbb{R}^m$ with $\Delta := \{(t,s) : t_0 \le s \le t \le T\}$ are assumed to be sufficiently smooth so that system (1) has a unique solution y [18,37]. The unfortunate fact is that there is no general rule to solve analytically a given VIE and therefore one is forced to employ a suitable numerical method giving an approximation of the exact solution. Over the last few decades, numerous numerical methods have been introduced for the numerical solution of (1). Although, the convergence of the *global* or *spectral* methods (see, for instance, [13]), may be spectral, they are not widely used. Their main drawback is that they treat Volterra equations as the Fredholm ones; therefore the value of T must be determined in advance; moreover, to obtain the approximate solution, one needs to solve a large dense system

[*]Corresponding author. Invited paper.

Received: 22 October 2020 / Revised: 25 November 2020 / Accepted: 25 November 2020
DOI: 10.22124/jmm.2020.17997.1554

of (nonlinear) equations which becomes very costly, in particular for the large values of m. In contrast, discretization methods for the numerical solution of VIEs (1), such as , do not have such drawbacks and usually approximate the solution only at a finite number of points t_n of the t-variable.

Numerous numerical methods within different classes of the methods have been already introduced to approximate the solution of VIEs together with analyzing their properties: a discussion on the theory and the application of Runge–Kutta type methods [18]; an investigation of the convergence of some classes of Runge–Kutta methods [31,32]; developing general structure of order conditions for Volterra–Runge–Kutta methods [17]; analyzing the stability properties of Runge–Kutta type methods [30, 38]; collocation methods [24–26]; a very general class of Runge–Kutta methods [23]; and general linear methods (GLMs) [2, 6, 33]. Furthermore, some reliable and efficient variable-stepsize (VS) codes for the numerical solution of VIEs have been developed based on: predictor-corrector schemes [36]; one-step method of collocation type [16]; embedded pair of Runge–Kutta methods [32]; and two-step continuous methods [22]; direct quadrature methods based on the linear barycentric rational quadrature rule [3] (references are made to [9, 14, 15] for discussion of barycentric interpolation and its applications). Moreover, the author of the present paper, together with his co-authors, developed two more efficient VS codes for VIEs based on natural Volterra Runge–Kutta (VRK) method [8], and GLMs [5] which we intend to discuss about them in the present paper.

The rest of the paper is organized along the following lines. In Section 2, representation of natural VRK methods in a VS environment is given and implementation strategies utilized in Matlab code `nvrk4.m` which is based on natural VRK method of order four, is discussed. In Section 3, implementation of Nordsieck GLMs in a VS environment using Nordsieck technique is surveyed and the Matlab code `vglm4.m` which is based on an unconditionally zero-stable GLM of order four is discussed. Some numerical experiments confirming the efficiency of the codes `nvrk4.m` and `vglm4.m` are presented in Section 4. Finally, some concluding remarks are given in Section 5.

2 The VS code based on natural VRK method of order four

Natural VRK methods for the numerical solution of VIEs (1) were first introduced in [11] and further investigated in [12, 23]. The VS formulation of ν-stage natural VRK methods of order p on nonuniform grid

$$t_0 < t_1 < \cdots < t_N, \quad t_N \geq T,$$

with the stepsizes $h_n = t_{n+1} - t_n$, is defined by [8]

$$\begin{aligned} Y_i^{[n]} &= h_n \sum_{j=1}^{\mu} \alpha_{ij} K\big(t_n + d_{ij}h_n, t_n + e_{ij}h_n, \sum_{l=1}^{\nu} \beta_{ijl} Y_l^{[n]}\big) + \widetilde{F}_n(t_n + c_i h_n), \quad i = 1, 2, \ldots, \nu, \\ y_{n+1} &= \sum_{j=1}^{\nu} w_j Y_j^{[n]}, \end{aligned} \tag{2}$$

for $n = 0, 1, \dots, N-1$, with

$$\widetilde{F}_n(t) = g(t) + \sum_{\kappa=1}^{n} h_{\kappa-1} \sum_{j=1}^{\nu} v_j K\big(t, t_{\kappa-1} + \xi_j h_{\kappa-1}, u(t_{\kappa-1} + \xi_j h_{\kappa-1})\big). \tag{3}$$

Here, μ is a fixed integer, $\widetilde{F}_n(t)$ is an approximation of order at least p to the tail

$$F_n(t) = g(t) + \int_{t_0}^{t_n} K(t, s, y(s))\, ds, \tag{4}$$

and u is a natural continuous extensions of the numerical solution by the interpolation formula of degree $d \le p$

$$u(t_n + \theta h_n) = \sum_{j=1}^{\nu} w_j(\theta) Y_j^{[n]}, \quad n = 0, 1, \dots, N-1, \quad \theta \in [0, 1],$$

where $w_j(\theta)$ are polynomials of degree d with $\lfloor p/2 \rfloor \le d \le \min\{\nu - 1, p\}$.

The importance of the paper [23] is that the first examples of VRK methods of orders p and stage order $q = p$ with $p = 3$ and $p = 4$ have been constructed in this paper which are A– and V_0–stable. We recall that the VRK method is said to be A–stable if

$$\Big\{h\lambda \in \mathbb{C} : \mathrm{real}(h\lambda) \le 0, \;\; \mathrm{imag}(h\lambda) \le 0\Big\} \subseteq \Big\{h\lambda \in \mathbb{C} : \overline{y}_n(h\lambda) \to 0 \quad as \quad n \to 0\Big\},$$

in which $\{\overline{y}_n\}_{n=0}^{\infty} = \{\overline{y}_n(h\lambda)\}_{n=0}^{\infty}$ is the numerical solution obtained by application of the VRK method with the fixed stepsize h to the basic test equation

$$y(t) = 1 + \lambda \int_0^t y(s)\, ds, \quad t \ge 0, \quad \lambda \in \mathbb{C}.$$

Also, the VRK method is said to be V_0–stable if

$$\Big\{(h\lambda, h^2\eta) \in \mathbb{R}^2 : h\lambda < 0, \;\; h^2\eta < 0\Big\} \subseteq \Big\{(h\lambda, h^2\eta) \in \mathbb{R}^2 : \widehat{y}_n(h\lambda, h^2\eta) \to 0 \quad as \quad n \to 0\Big\},$$

in which $\{\widehat{y}_n\}_{n=0}^{\infty} = \{\overline{y}_n(h\lambda, h^2\eta)\}_{n=0}^{\infty}$ is the numerical solution obtained by application of the VRK method with the fixed stepsize h to the convolution test equation

$$y(t) = 1 + \int_0^t (\lambda + \eta(t-s))\, y(s)\, ds, \quad t \ge 0, \quad \lambda, \eta \in \mathbb{R}.$$

In [8], the author of the present paper together with G. Hojjati, Z. Jackiewicz, H. Mahdi, have developed an efficient MATLAB code, `nvrk4.m`, based on A– and V_0–stable natural VRK method of order and stage four constructed in [23]. In this section, we focus on this code and its implementation issues:

– Reducing the computational cost: At first, it should be noted that the main computational cost of the code is concerned to that of the lag term $\widetilde{F}_n(t)$. An elegant idea to compute an approximation to the tail $F_n(t)$ (4) has been introduced in [8] which reduces the computational cost of the code by more than %80. In this way, the tail is approximated by

$$\begin{aligned}\widetilde{F}_n(t) = g(t) &+ \sum_{\kappa=1}^{n^{(6)}} \sum_{j=1}^{4} v_{\kappa j}^{(6)} K\big(t, t_{6(\kappa-1)+2(j-1)}, u(t_{6(\kappa-1)+2(j-1)})\big) \\ &+ \sum_{\kappa=1}^{n^{(3)}} \sum_{j=1}^{4} v_{n^{(6)}j}^{(3)} K\big(t, t_{6n^{(6)}+j-1}, u(t_{6n^{(6)}+j-1})\big) \\ &+ \sum_{\kappa=1}^{n-\ell} h_{\ell+\kappa-1} \sum_{j=1}^{\nu} v_j K\big(t, t_{\ell+\kappa-1} + \xi_j h_{\ell+\kappa-1}, u(t_{\ell+\kappa-1} + \xi_j h_{\ell+\kappa-1})\big),\end{aligned} \tag{5}$$

where $\ell := 6n^{(6)} + 3n^{(3)}$. Here, the quadrature weights $v_{n^{(6)}j}^{(3)}$ and $v_{\kappa j}^{(6)}$ which correspond to quadrature rules of order greater or equal to $p = 4$, are complicated expressions in terms of $h_{6n^{(6)}}$, $h_{6n^{(6)}+1}$, $h_{6n^{(6)}+2}$ and h_r, h_{r+1}, h_{r+2}, h_{r+3}, h_{r+4}, h_{r+5}, with $r = 6(\kappa-1)$, respectively. The quadrature order conditions for the weights $v_{n^{(6)}j}^{(3)}$ and $v_{\kappa j}^{(6)}$ are given by

$$\begin{cases} \displaystyle\sum_{j=1}^{4} v_{n^{(6)}j}^{(3)} = \sum_{i=0}^{2} h_{6n^{(6)}+i}, \\ \displaystyle\sum_{j=2}^{4} v_{n^{(6)}j}^{(3)} \Big(\sum_{i=0}^{j-2} h_{6n^{(6)}+i}\Big)^{k-1} = \frac{1}{k}\Big(\sum_{i=0}^{2} h_{6n^{(6)}+i}\Big)^{k}, \quad k = 2,3,4, \end{cases} \tag{6}$$

and

$$\begin{cases} \displaystyle\sum_{j=1}^{4} v_{\kappa j}^{(6)} = \sum_{i=0}^{5} h_{r+i}, \\ \displaystyle\sum_{j=2}^{4} v_{\kappa j}^{(6)} \Big(\sum_{i=0}^{2j-3} h_{r+i}\Big)^{k-1} = \frac{1}{k}\Big(\sum_{i=0}^{5} h_{r+i}\Big)^{k}, \quad k = 2,3,4, \end{cases} \tag{7}$$

respectively.

– Initial stepsize: Automatic selection of initial stepsize is important to produce a reliable and effective code and may reduce the computational cost of the code. The utilized initial stepsize in the code `nvrk4.m` is as

$$h_0 = \min\Big\{0.01(T - t_0), \mathrm{tol}^{1/5}/\big\|K(t_0, t_0, g(t_0))\big\|\Big\},$$

with tol as a given tolerance. This strategy for the selection of h_0 is inspired by the idea from Gladwell et al. [27].

– Local error estimation: Local error of the method in the point t_{n+1} is defined by

$$LE_{n+1} = \Big| \int_{t_n}^{t_{n+1}} K(t_{n+1}, s, y_n(s))\,ds - h_n \sum_{j=1}^{\mu} \alpha_{\nu j} K\big(t_n + d_{\nu j} h_n, t_n + e_{\nu j} h_n, \sum_{l=1}^{\nu} \beta_{\nu j l} Y_l^{[n]}\big) \Big|,$$

with

$$y_n(t) = g(t) + \widetilde{F}_n(t) + \int_{t_n}^{t} K(t, s, y_n(s))\,ds, \quad t \in [t_n, t_{n+1}].$$

The local error LE_{n+1} is estimated as

$$\begin{aligned}\operatorname{est}(t_{n+1}) := \Big| & h_n \sum_{j=1}^{5} \overline{v}_j K\big(t_{n+1}, t_n + \overline{\xi}_j h_n, u(t_n + \overline{\xi}_j h_n)\big) \\ & - h_n \sum_{j=1}^{\mu} \alpha_{\nu j} K\big(t_n + d_{\nu j} h_n, t_n + e_{\nu j} h_n, \sum_{l=1}^{\nu} \beta_{\nu j l} Y_l^{[n]}\big) \Big|,\end{aligned}$$

in which the quadrature weights $\overline{v}_j$ are the weights for a five order quadrature rule and are given by

$$\sum_{j=1}^{5} \overline{v}_j \, \overline{\xi}_j^{k-1} = \frac{1}{k}, \quad k = 1, 2, \dots, 5.$$

The value of $\overline{\xi}_j$, $j = 1, 2, \dots, 5$, has been chosen as $(j-1)/4$.

– Stepsize changing strategy: the utilized stepsize controller in the code `nvrk4.m` is as follows: the current step will be accepted if $\mathrm{err}_{n+1} := \|\operatorname{est}(t_{n+1})\| \le \rho \|sc\|$, and then the new stepsize is computed by the formula

$$h_n = h_{n-1} \cdot \min\{facmax, r_{opt}\},$$

with r_{opt} given by

$$r_{opt} = \Big(\frac{fac \cdot \|sc\|}{\mathrm{err}_n} \Big)^{1/(p+1)},$$

for $n = 1$ and

$$r_{opt} = \Big(\frac{fac \cdot \|sc\|}{\mathrm{err}_n} \Big)^{k_I} \Big(\frac{fac \cdot \|sc\|}{\mathrm{err}_{n-1}} \Big)^{k_P},$$

for $n > 1$ which was suggested by Gustafsson et al. [29] (see also [19, 28]). The values for the parameters k_I and k_p have been chosen as $0.7/(p+1)$ and $-0.4/(p+1)$, respectively. Also, the components of the m-dimensional vector sc is defined by

$$sc_i = \mathrm{Atol}_i + \max\big\{ |(y_{n-1})_i, (y_n)_i| \big\} \mathrm{Rtol}_i, \quad i = 1, 2, \dots, m,$$

where Atol_i and Rtol_i are absolute and relative error tolerances corresponding to the ith component of the solution $y_i(t)$. Furthermore, the current step is rejected if $\mathrm{err}_{n+1} > \rho\|sc\|$ and then the computations are repeated with a new stepsize $\widetilde{h}_n$ given by

$$\widetilde{h}_n = h_n \cdot \min\{facmin, \widetilde{r}_{opt}\},$$

with $\widetilde{r}_{opt}$ as

$$\widetilde{r}_{opt} = \Big(\frac{fac \cdot \|sc\|}{\mathrm{err}_n}\Big)^{1/(p+1)}.$$

The values for the parameters has been considered as $\rho = 1.2$, $facmax = 2$, $facmin = 0.5$, and $fac = 0.9$.

3 The VS code based on GLM of order four

General linear methods for ODEs, as a comprehensive extension of these traditional methods, were introduced by Butcher [20] (see also [19,34]). This large family of the methods opened up the possibility of obtaining new methods which are neither traditional methods nor slight variations of them. The efficient GLM-based VS codes `dim18.m` [21], `dim13s.m` [35], and `irks14.m` [10] which are also variable order, have been introduced for nonstiff and stiff ODEs. Moreover, the more reliable VS code `SGLM4.m` [4] has been developed for stiff ODEs which is based on second derivative GLMs as an extension of GLMs (see [1,7]).

GLMs for VIEs were first introduced by Izzo et al. [33] and investigated more by Abdi et al. in [2,6]. In [33], Nordsieck GLMs of orders $p = 1, 2$ and stage order $q = p$ has been constructed in which the input and output vectors of the methods approximate the Nordsieck vector of order p. By investigating GLMs in general form, rather than Nordsieck form, in [6], methods of order p and stage order $q = p$ up to four have been constructed; in the constructed methods, methods of orders one and two were are A– and $V_0(\alpha)$–stable while the stability regions of the methods of orders three and four with respect to both the basic and convolution test equations are bounded which obtained by setting arbitrary values for free parameters in the coefficents of the methods. Therefore, Abdi in [2] constructed GLMs of orders $p = 3, 4$ and stage order $q = p$ with a large region of absolute stability by minimizing the objective function for the negative area of the region of absolute stability of the method. The VS mode of these methods by using the Nordsieck technique has been investigated in [5] which we focus on it in this section: Consider a nonuniform grid

$$t_0 < t_1 < \cdots < t_N, \quad t_N \geq \mathrm{T},$$

with the stepsizes $h_n = t_n - t_{n-1}$, $n = 1, 2, \ldots, N$. Assume that the input and output vectors $y^{[n-1]}$ and $y^{[n]}$ of the method respectively approximate the Nordsieck vectors

$$z(t_{n-1}, h_{n-1}) := \begin{bmatrix} y(t_{n-1}) \\ h_{n-1} y'(t_{n-1}) \\ \vdots \\ h_{n-1}^{p} y^{(p)}(t_{n-1}) \end{bmatrix}, \qquad z(t_n, h_n) := \begin{bmatrix} y(t_n) \\ h_n y'(t_n) \\ \vdots \\ h_n^{p} y^{(p)}(t_n) \end{bmatrix}.$$

Defining $t_{n-1,c} := [t_{n-1,1} \quad t_{n-1,2} \quad \cdots \quad t_{n-1,s}]^T$ with $t_{n-1,j} = t_{n-1} + c_j h_n$, $j = 1, 2, \ldots, s$, and the vectors $Y^{[n]}$, $F_h^{[n]}(t_{n-1,c})$, and $\Phi_h^{[n]}(t_{n-1,c})$ as

$$Y^{[n]} := \begin{bmatrix} Y_1^{[n]} \\ Y_2^{[n]} \\ \vdots \\ Y_s^{[n]} \end{bmatrix}, \quad F_{h_n}^{[n]}(t_{n-1,c}) \quad := \begin{bmatrix} F_{h_n}^{[n]}(t_{n-1,1}) \\ F_{h_n}^{[n]}(t_{n-1,2}) \\ \vdots \\ F_{h_n}^{[n]}(t_{n-1,s}) \end{bmatrix}, \quad \Phi_{h_n}^{[n]}(t_{n-1,c}) := \begin{bmatrix} \Phi_{h_n}^{[n]}(t_{n-1,1}) \\ \Phi_{h_n}^{[n]}(t_{n-1,2}) \\ \vdots \\ \Phi_{h_n}^{[n]}(t_{n-1,s}) \end{bmatrix},$$

with

$$F_{h_n}^{[n]}(t_{n-1,j}) = g(t_{n-1,j}) + \sum_{\nu=1}^{n-1} h_\nu \sum_{\ell=1}^{s} b_\ell K(t_{n-1,j}, t_{\nu-1,\ell}, Y_\ell^{[\nu]}), \tag{8}$$

and

$$\Phi_{h_n}^{[n]}(t_{n-1,j}) = h_n \sum_{\ell=1}^{s} w_{j,\ell} K(t_{n-1,j}, t_{n-1,\ell}, Y_\ell^{[n]}), \tag{9}$$

a Nordsieck GLM in a VS environment takes the form

$$\begin{aligned} Y^{[n]} &= (A \otimes I_m)\left(F_{h_n}^{[n]}(t_{n-1,c}) + \Phi_{h_n}^{[n]}(t_{n-1,c})\right) + (UD_n \otimes I_m)y^{[n-1]}, \\ y^{[n]} &= (B \otimes I_m)\left(F_{h_n}^{[n]}(t_{n-1,c}) + \Phi_{h_n}^{[n]}(t_{n-1,c})\right) + (VD_n \otimes I_m)y^{[n-1]}, \end{aligned} \tag{10}$$

with the diagonal rescaling matrix $D_n := D(\delta_n)$ defined by

$$D(\delta_n) := \mathrm{diag}\big(1, \delta_n, \delta_n^2, \ldots, \delta_n^p\big),$$

where δ_n is the ratio of sequential stepsizes at the step numbers $n-1$ and n and is defined by $\delta_n = h_n/h_{n-1}$. One should note that the zero-stability properties of the GLMs (10) are determined by the product matrix VD_n. The constructed methods in [2] are in general form that should be transformed to the Nordsieck form to use in a VS environment (see [5] for the transformation of the methods); although the method of order four in [2] has good stability properties, it is not zero-stable in VS environment [5] and therefore can not be utilized in a practical code. A method of order four in the Nordsieck form with good stability properties and zero-stable in a (fixed stepsize and) VS environment for any stepsize pattern has been constructed in [5] which has been utilized in the VS MATLAB code `vglm4.m` based on GLM of order four. In this section, we focus on this code and its implementation issues:

- Reducing the computational cost: In a similar way used in the code `nvrk4.m`, to reduce the total computational cost, the tail is approximated by

$$\begin{aligned} F_{h_n}^{[n]}(t_{n-1,j}) = g(t_{n-1,j}) &+ \sum_{\kappa=1}^{n^{(6)}} \sum_{i=1}^{4} b_{\kappa i}^{(6)} K\big(t_{n-1,j}, t_{6(\kappa-1)+2(i-1)}, y_{6(\kappa-1)+2(i-1)}\big) \\ &+ \sum_{\kappa=1}^{n^{(3)}} \sum_{i=1}^{4} b_{n^{(6)}i}^{(3)} K\big(t_{n-1,j}, t_{6n^{(6)}+i-1}, y_{6n^{(6)}+i-1}\big) \\ &+ \sum_{\kappa=1}^{n-1-\ell} h_{\ell+\kappa-1} \sum_{i=1}^{s} b_i K\big(t_{n-1,j}, t_{\ell+\kappa-1} + c_i h_{\ell+\kappa-1}, Y_i^{[\ell+\kappa]}\big), \end{aligned} \tag{11}$$

where $\ell := 6n^{(6)} + 3n^{(3)}$. The quadrature order conditions for obtaining the quadrature weights $b^{(3)}_{n^{(6)}j}$ and $b^{(6)}_{\kappa j}$ are the same as those for $v^{(3)}_{n^{(6)}j}$ and $v^{(6)}_{\kappa j}$ given by (6) and (7).

– Starting procedure: To begin the computations, at first the starting vector $y^{[0]}$ should be computed. To do this, we carry out one step of natural VRK method of order four with the abscissae vector $\widetilde{c} = [\widetilde{c}_1 \quad \widetilde{c}_2 \quad \widetilde{c}_3 \quad \widetilde{c}_4]^T$ which gives sufficient output information $\widetilde{Y}_i$, $i = 1, 2, 3, 4$, to compute $y^{[0]}$. Then the starting vector is computed by $y^{[0]} = (T^{-1} \otimes I_m)\, S_0$ in which $S_0 := \begin{bmatrix} y_0^T & \widetilde{Y}_1^T & \widetilde{Y}_2^T & \widetilde{Y}_3^T & \widetilde{Y}_4^T \end{bmatrix}^T$, and $T = [t_{i,j}]$ is a 5×5 matrix given by

$$t_{i,j} = \begin{cases} \delta_{1j}, & i = 1, \text{ or } j = 1, \\ \dfrac{\widetilde{c}_{i-1}^{\,j-1}}{(j-1)!}, & i, j \neq 1, \end{cases}$$

with δ_{ij} as the Kronecker delta.

– Local error estimation: Local error of the method in the point t_n is defined by

$$LE_n = \left| \int_{t_{n-1}}^{t_n} k(t_n, s, y_n(s))\, ds - h_n \sum_{\ell=1}^{s} w_{s,\ell} K(t_{n-1,s}, t_{n-1,\ell}, Y_\ell^{[n]}) \right|,$$

with

$$y_n(t) = g(t) + F_{h_n}^{[n]}(t) + \int_{t_{n-1}}^{t} k(t, s, y_n(s))\, ds, \quad t \in [t_{n-1}, t_n].$$

The local error LE_{n+1} is estimated as

$$\text{est}(t_{n+1}) := \left| h_n \sum_{j=1}^{6} \bar{b}_j k\big(t_n, t_{n-1} + \xi_j h_n, u(t_{n-1} + \xi_j h_n)\big) - h_n \sum_{\ell=1}^{s} w_{s,\ell} K(t_{n-1,s}, t_{n-1,\ell}, Y_\ell^{[n]}) \right|,$$

in which $u(t_{n-1}+\theta h_n)$ is the continuous extension of the numerical solution by the formula

$$u(t_{n-1}+\theta h_n) = \sum_{j=1}^{s} w_j(\theta) Y_j^{[n]} + w_{s+1}(\theta)\delta_n y^{[n-1]}(m+1 : 2m), \quad n = 1, 2, \ldots, N, \quad \theta \in [0, 1],$$

with $w_j(\theta)$ as polynomials of degree $d \leq 5$. These polynomials satisfy the linear system of equations

$$\sum_{j=1}^{s} w_j(\theta)\, c_j^k + \delta_{1k} w_{s+1}(\theta) = \theta^k, \quad k = 0, 1, \ldots, 5,$$

and the quadrature weights $\bar{b}_j$ are given by

$$\sum_{j=1}^{6} \bar{b}_j\, \xi_j^{k-1} = \frac{1}{k}, \quad k = 1, 2, \ldots, 6.$$

The value of ξ_j, $j = 1, 2, \ldots, 5$, has been chosen as $(j-1)/5$.

– Initial stepsize and stepsize changing strategy: The used strategy for the selection of the initial stepsize and designing the stepsize changing strategy, for `vglm4.m` are the same as those for `nvrk4.m` which are explained in the previous section.

4 Numerical experiments

In this section, we give some numerical results obtained by the codes `nvrk4.m` and `vglm4.m`. We performed numerical experiments on several linear and nonlinear problems which we report here the results obtained on the following problems:

– The linear VIEs

$$y(t) = e^t + \int_0^t 2\cos(t-s)y(s)ds, \quad t \in [0,2], \tag{12}$$

with the exact solution $(1+t)^2e^t$.

– The nonlinear system of VIE [37]

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} e^{-t} - \dfrac{t^2}{2} \\ t - \dfrac{t^4}{24} \end{bmatrix} + \int_0^t \begin{bmatrix} e^{s-t}y_1(s)^2 + y_2(s) \\ (t-s)\dfrac{y_2(s)^2}{1+y_1(s)^2} \end{bmatrix} ds, \quad t \in [0,2], \tag{13}$$

with the exact solution $y_1(t) = 1$ and $y_2(t) = t$.

Table 1: Numerical results for the problem (12).

tol	The code	ns	nrs	nge	nke	nje	imax	ge
10^{-4}	vglm4	17	0	86	2065	16	2	$4.45 \cdot 10^{-1}$
	nvrk4	21	0	85	2020	192	3	$1.05 \cdot 10^{-2}$
10^{-6}	vglm4	30	0	151	4345	16	2	$2.04 \cdot 10^{-2}$
	nvrk4	36	4	161	4672	624	4	$8.79 \cdot 10^{-4}$
10^{-8}	vglm4	66	0	331	12685	16	2	$7.20 \cdot 10^{-4}$
	nvrk4	82	4	345	14004	1280	3	$6.08 \cdot 10^{-5}$
10^{-10}	vglm4	163	1	821	51230	32	2	$2.07 \cdot 10^{-5}$
	nvrk4	201	0	805	56252	2224	3	$2.03 \cdot 10^{-6}$
10^{-12}	vglm4	394	1	1976	237480	32	2	$5.64 \cdot 10^{-7}$
	nvrk4	513	0	2053	302060	4048	3	$5.25 \cdot 10^{-8}$

Some cost statistics such as the number of steps, `ns`, the number of rejected steps, `nrs`, the number of kernel evaluations, `nke`, the number of Jacobian evaluations, `nje`, the maximum number of local Newton iterations on the whole steps, `imax`, and the global error at the endpoint of the interval of integration, `ge`, have been reported in Tables 1 and 2 for problems (12) and (13) and various values of tol = Atol = Rtol. Furthermore, Figure 1 displays stepsize and order patterns in which the symbols "$*$" stands for indicating the rejected steps in the code `nvrk4.m`. The results illustrate the efficiency and capability of the codes `nvrk4.m` and `vglm4.m` in solving linear and nonlinear (system of) VIEs.

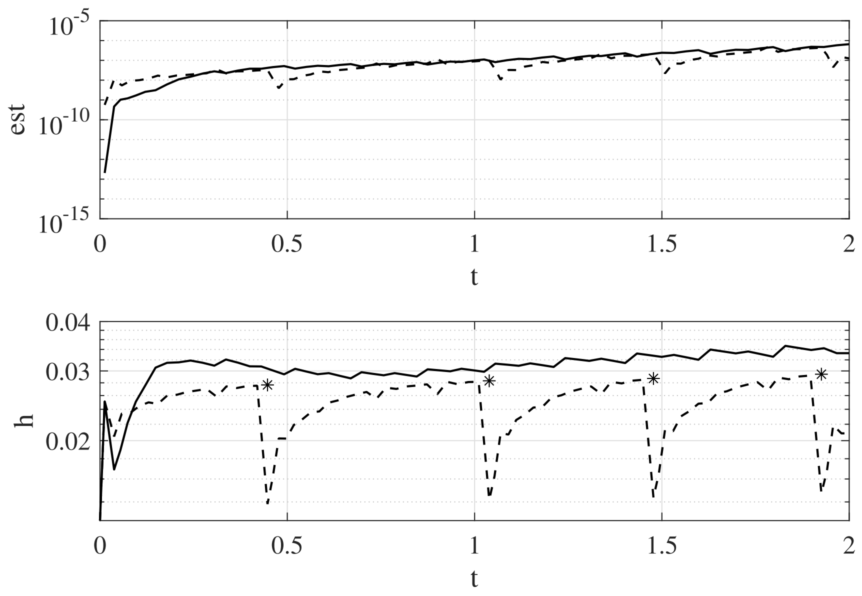

Figure 1: Local error estimate (top) and stepsize versus t (bottom) for the problem (12) with tol $= 10^{-8}$ (thick solid line for `vglm4` and dashed line for `nvrk4`).

Table 2: Numerical results for the problem (13).

tol	The code	*ns*	*nrs*	*nge*	*nke*	*nJe*	*imax*	*ge*
10^{-4}	`vglm4`	9	0	46	1020	32	3	$2.83 \cdot 10^{-4}$
	`nvrk4`	15	0	61	5396	176	248	$3.48 \cdot 10^{-4}$
10^{-6}	`vglm4`	15	1	81	1995	80	3	$1.35 \cdot 10^{-4}$
	`nvrk4`	22	0	89	2264	336	3	$4.95 \cdot 10^{-5}$
10^{-8}	`vglm4`	58	3	306	10920	128	3	$1.27 \cdot 10^{-5}$
	`nvrk4`	43	2	181	5640	720	3	$3.62 \cdot 10^{-6}$
10^{-10}	`vglm4`	134	4	691	37655	144	3	$4.45 \cdot 10^{-7}$
	`nvrk4`	92	2	377	16488	1504	3	$2.21 \cdot 10^{-7}$
10^{-12}	`vglm4`	255	3	1291	109725	112	3	$1.19 \cdot 10^{-8}$
	`nvrk4`	214	0	857	63736	3424	3	$7.48 \cdot 10^{-9}$

5 Conclusion

In this paper we have aimed at introducing the reader to the recent efficient numerical methods for VIEs and the elegant codes `nvrk4.m` and `vglm4` developed in recent years as the production of the natural VRK method and GLM of order four, respectively. The obtained numerical results by these practical codes confirm that they can be widely used to solve system of (nonlinear) VIEs.

References

[1] A. Abdi, *Construction of high-order quadratically stable second-derivative general linear methods for the numerical integration of stiff ODEs*, J. Comput. Appl. Math. **303** (2016) 218–228.

[2] A. Abdi, *General linear methods with large stability regions for Volterra integral equations*, Comp. Appl. Math. **38** (2019) 52:1–16.

[3] A. Abdi, J.P. Berrut, S.A. Hosseini, *Explicit methods based on barycentric rational interpolants for solving non-stiff Volterra integral equations*, submitted.

[4] A. Abdi, D. Conte, *Implementation of second derivative general linear methods*, Calcolo **57** (2020) 20:1–29.

[5] A. Abdi, D. Conte, *Implementation of general linear methods for Volterra integral equations*, J. Comput. Appl. Math. **386** (2021) 113261.

[6] A. Abdi, S. Fazeli, G. Hojjati, *Construction of efficient general linear methods for stiff Volterra integral equations*, J. Comput. Appl. Math. **292** (2016) 417–429.

[7] A. Abdi, G. Hojjati, *An extension of general linear methods*, Numer. Algorithms 57 (2011) 149–167

[8] A. Abdi, G. Hojjati, Z. Jackiewicz, H. Mahdi, *A new code for Volterra integral equations based on natural Runge–Kutta methods*, Appl. Numer. Math. **143** (2019) 35–50.

[9] A. Abdi, S.A. Hosseini, *The barycentric rational difference-quadrature scheme for systems of Volterra integro-differential equations*, SIAM J. Sci. Comput. **40** (2018) A1936–A1960.

[10] A. Abdi, Z. Jackiewicz, *Towards a code for nonstiff differential systems based on general linear methods with inherent Runge–Kutta stability*, Appl. Numer. Math. **136** (2019) 103–121.

[11] A. Bellen, Z. Jackiewicz, R. Vermiglio, M. Zennaro, *Natural continuous extensions of Runge–Kutta methods for Volterra integral equations of the second kind and their applications*, Math. Comput. **52** (1989) 49–63.

[12] A. Bellen, Z. Jackiewicz, R. Vermiglio, M. Zennaro, *Stability analysis of Runge–Kutta methods for Volterra integral equations of the second kind*, IMA J. Numer. Anal. **10** (1990) 103–118.

[13] M.I. Berenguer, A.I. Garralda-Guillem, M. Ruiz Galan, *An approximation method for solving systems of Volterra integro-differential equations*, Appl. Numer. Math. **67** (2013) 126–135.

[14] J.P. Berrut, *Rational functions for guaranteed and experimentally well-conditioned global interpolation*, Comput. Math. Appl. **15** (1988) 1–16.

[15] J.P. Berrut, L.N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Rev. **46** (2004) 501–517.

[16] J.G. Blom, H. Brunner, *The numerical solution of nonlinear Volterra integral equations of the second kind by collocation and iterated collocation methods*, SIAM J. Sci. Stat. Comput. **8** (1987) 806–830.

[17] H. Brunner, E. Hairer, S.P. Nørsett, *Runge–Kutta theory for Volterra integral equations of the second kind*, Math. Comput. **39** (1982) 147–163.

[18] H. Brunner, P.J. van der Houwen, *The Numerical Solution of Volterra Equations*, CWI monographs. North-Holland, Amsterdam, 1986.

[19] J.C. Butcher, *Numerical Methods for Ordinary Differential Equations*, Wiley, New York, 2016.

[20] J.C. Butcher, *On the convergence of numerical solutions to ordinary differential equations*, Math. Comput. **20** (1966) 1–10.

[21] J.C.Butcher, P. Chartier, Z. Jackiewicz, *Experiments with a variable-order type 1 DIMSIM code*, Numer. Algorithms **22** (1999) 237–261.

[22] G. Capobianco, D. Conte, B. Paternoster, *Construction and implementation of two-step continuous methods for Volterra integral equations*, Appl. Numer. Math. **119** (2017) 239–247.

[23] D. Conte, R. D'Ambrosio, G. Izzo, Z. Jackiewicz, *Natural Volterra Runge–Kutta methods*, Numer. Algorithms **65** (2014) 421–445.

[24] D. Conte, R. D'Ambrosio, B. Paternoster, *Two-step diagonally-implicit collocation based methods for Volterra Integral Equations*, Appl. Numer. Math. **62** (2012) 1312–1324.

[25] D. Conte, Z. Jackiewicz, B. Paternoster, *Two-step almost collocation methods for Volterra integral equations*, Appl. Math. Comput. **204** (2008) 839–853.

[26] D. Conte, B. Paternoster, *Multistep collocation methods for Volterra integral equations*, Appl. Numer. Math. **59** (2009) 1721–1736.

[27] I. Gladwell, L.F. Shampine, R.W. Brankin, *Automatic selection of the initial step size for an ODE solver*, J. Comput. Appl. Math. **18** (1987) 175–192.

[28] K. Gustafsson, *Control theoretic techniques for stepsize selection in explicit Runge–Kutta methods*, ACM T. Math. Software **17** (1991) 533–554.

[29] K. Gustafson, M. Lundh, and G. Söderlind, *A PI stepsize control for the numerical solution of ordinary differential equations*, BIT **28** (1988) 270–287.

[30] E. Hairer, C. Lubich, *On the stability of Volterra-Runge-Kutta methods*, SIAM J. Numer. Anal. **21** (1984) 123–135.

[31] E. Hairer, C. Lubich, S.P. Nørsett, *Order of convergence of one-step methods for Volterra integral equations of the second kind*, SIAM J. Numer. Anal. **20** (1983) 569–579.

[32] F.C. Hoppensteadt, Z. Jackiewicz, B. Zubik-Kowal, *Numerical solution of Volterra integral and integro-differential equations with rapidly vanishing convolution kernels*, BIT **47** (2007) 325–350.

[33] G. Izzo, Z. Jackiewicz, E. Messina, A. Vecchio, *General linear methods for Volterra integral equations*, J. Comput. Appl. Math. **234** (2010) 2768–2782.

[34] Z. Jackiewicz, *General Linear Methods for Ordinary Differential Equations*, John Wiley & Sons, 2009.

[35] Z. Jackiewicz, *Implementation of DIMSIMs for stiff differential systems*, Appl. Numer. Math. **42** (2002) 251–267.

[36] H.M. Jones, S. McKee, *Variable step size predictor–corrector schemes for second kind Volterra integral equations*, Math. Comput. **44** (1985) 391–404.

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

[38] P.J. van der Houwen, P.H.M. Wolkenfelt, C.T.H. Baker, *Convergence and stability analysis for modified Runge–Kutta methods in the numerical treatment of second-kind Volterra integral equations*, IMA J. Numer. Anal. **1** (1981) 303–328.