BIT Numerical Mathematics (2007) 47: 259–275
DOI: 10.1007/s10543-007-0120-5
Published online: 16 February 2007 – © Springer 2007

FAST RUNGE–KUTTA METHODS FOR NONLINEAR CONVOLUTION SYSTEMS OF VOLTERRA INTEGRAL EQUATIONS*

G. CAPOBIANCO[1], D. CONTE[2], I. DEL PRETE[3] and E. RUSSO[3]

[1]*Dipartimento S.T.A.T., Università degli Studi del Molise, Contrada Fonte Lappone, I-86090 Pesche (IS), Italy. e-mail: giovanni.capobianco@unimol.it*

[2]*Dipartimento di Matematica e Informatica, Università di Salerno, via Ponte Don Melillo, I-84084 Fisciano (SA), Italy. e-mail: dajconte@unisa.it*

[3]*Dipartimento di Matematica e Applicazioni "R. Caccioppoli", Università degli Studi Napoli "Federico II", Via Cintia, Monte S. Angelo, I-80126 Napoli, Italy. e-mail: {ida.delprete, elvrusso}@unina.it*

Abstract.

In this paper fast implicit and explicit Runge–Kutta methods for systems of Volterra integral equations of Hammerstein type are constructed. The coefficients of the methods are expressed in terms of the values of the Laplace transform of the kernel. These methods have been suitably constructed in order to be implemented in an efficient way, thus leading to a very low computational cost both in time and in space. The order of convergence of the constructed methods is studied. The numerical experiments confirm the expected accuracy and computational cost.

AMS subject classification (2000): 65R20, 45D05, 44A35, 44A10.

Key words: Volterra integral equations of Hammerstein type, Volterra Runge–Kutta methods, convolution, fast numerical methods.

1 Introduction.

Volterra integral equations (VIEs) arising in mathematical modeling processes are often characterized by convolution kernels. The more general class of nonlinear convolution equations are the so-called Volterra integral equations of Hammerstein type:

$$(1.1)\quad y(t) = f(t) + \int_0^t k(t-\tau)g(\tau, y(\tau))d\tau, \quad t \in I := [0,T], \qquad y, f, g \in R^d, k \in R^{d\times d} \quad d \geq 1.$$

* Received June 5, 2006. Accepted December 5, 2006. Communicated by Uri Ascher.

We assume that the functions f and k are continous on I and g satisfies the Lipschitz condition with respect to y. This assumptions ensure the existence and the uniqueness of the solution of (1.1) [4, 16].

These equations are mathematical models of evolutionary phenomena incorporating memory and, directly or indirectly, they arise in many branches of science. For example VIEs of Hammerstein type directly arise in models of epidemic diffusion, in neurophysiology, in feedback control theory, in the study of the behaviour of nuclear reactors, wherease we can also arrive to equations of the form (1.1) by manipulating some special hyperbolic differential equations, by constructing, for example, transparent boundary conditions in wave diffusion problems in unbounded domains [10, 12, 13, 14] or by considering the semi-discretization in space of Volterra–Fredholm integral equations [3, 5].

It is known that the numerical treatment of VIEs has a very high computational cost, since, for each time step, we have to compute the history term (*lag term*). For a naive implementation of a classical numerical method the computational cost in time to calculate the solution of (1.1) over N_t time steps is $\mathcal{O}(d^2N_t^2)$, with a memory request of $\mathcal{O}(d^2N_t)$ space (see [4]). Even in the simpler case of scalar VIEs (i.e. $d = 1$), as the number of points N_t may be very large, it is important to seek for numerical methods which can pull down the computational cost (both in time and in space). As a matter of fact, in the literature several authors have been interested in the reduction of the computational cost of numerical methods for the scalar version of (1.1). For example a Volterra Runge–Kutta (VRK) method of order $p = 4$ requiring a computational effort of $\mathcal{O}(N_t(\log N_t)^2)$ in time and of $\mathcal{O}(N_t)$ in space was constructed in [11]. Moreover a class of fast collocation methods with a computational cost of $\mathcal{O}(N_t(\log N_t))$ operations and memory reqirement of $\mathcal{O}(\log N_t)$ was proposed in [6]. Such numerical methods are all based on fast algorithms for the lag term computation.

In this work we construct implicit and explicit fast VRK (FVRK) methods of generic order p for the system of equations (1.1) that require only $\mathcal{O}(d^2N_t(\log N_t))$ and $\mathcal{O}(d^2 \log N_t)$ of cost in time and space respectively. We observe that the pulling down of the cost in space from $\mathcal{O}(d^2N_t)$ to $\mathcal{O}(d^2 \log N_t)$ is not a less important goal than the reduction of the cost in time. As a matter of fact in many problems the number of points N_t may be very large, thus leading to difficulties in the memorization of the arrays. The coefficients of FVRK methods are expressed in terms of M values of the Laplace transform $K(s)$ of the kernel $k(t)$, so they are particularly suitable for all those problems where we only know $K(s)$ rather than the kernel itself (see [13, 14] and the related bibliography). Such methods are inspired on the scheme proposed in [14], also used in [6] for the construction of fast collocation methods.

In Section 2 we construct two classes of FVRK methods to approximate the solution of (1.1), namely, explicit FVRK methods of Pouzet type (FPVRK methods) and implicit FVRK methods of de Hoog and Weiss (FHVRK methods). In Section 3 we determine the order of convergence of the constructed methods and we prove that they keep the same order of the corresponding classical methods with a not very large number of points M. The calculation of the computational cost is presented in Section 4. Section 5 contains numerical results obtained on some real problems that confirm the expected performances of the FVRK methods in terms of accuracy and computational cost. In Section 6 some concluding remarks are reported.

2 Fast Runge–Kutta methods.

In order to construct FVRK methods for the equation (1.1) we shall refer to classical explicit extended VRK methods of Pouzet type (PVRK methods) and implicit VRK methods of de Hoog and Weiss [8] (HVRK methods).

Suppose that the given interval I is discretized by the uniform mesh

$$I_h = \{t_n := nh, n = 0, \ldots, N_t, h \geq 0, N_t h = T\}.$$

The equation (1.1) can be rewritten, by relating it to this mesh, as

$$y(t) = F_n(t) + \Phi_n(t) \quad t \in [t_n, T],$$

where

$$F_n(t) := f(t) + \int_0^{t_n} k(t-\tau) g(\tau, y(\tau)) d\tau$$

and

$$\Phi_n(t) := \int_{t_n}^{t} k(t-\tau) g(\tau, y(\tau)) d\tau$$

represent respectively the *lag term* and the *increment function*.

A VRK method is based on an approximation scheme for the increment function $\Phi_n(t)$, that will be called a *VRK formula* and denoted by $\bar{\Phi}_n(t)$, and on an approximation scheme, $\bar{F}_n(t)$, for the lag term, that will be called *lag term formula*.

The approximation of the equation in the mesh point t_{n+1} leads to the discrete method of the form

$$y_{n+1} = \bar{F}_n(t_n + h) + \bar{\Phi}_n(t_n + h) \quad n = 0, \ldots, N_t - 1. \tag{2.1}$$

Let us fix the vectors $c = (c_i)_{i=1}^m$, $b = (b_i)_{i=1}^m$ and the square matrix $A = (a_{is})_{i,s=1}^m$, determined by the "Butcher array" for ODEs

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} \tag{2.2}$$

and let us fix $c_{m+1} = 1$.

An explicit m-stage PVRK method applied to the equation (1.1) reads

$$y_{n+1} = \bar{F}_n(t_{n,m+1}) + \bar{\Phi}_n(t_{n,m+1}) \quad n = 0, \ldots, N_t - 1, \tag{2.3}$$

where the increment term is given by

$$\bar{\Phi}_n(t_{n,m+1}) = h\sum_{i=1}^{m} b_i k((1-c_i)h)g(t_{n,i},Y_{n,i}). \tag{2.4}$$

The stages $Y_{n,i}$ are explicitly computed through

$$Y_{n,i} = \bar{F}_n(t_{n,i}) + h\sum_{s=1}^{i-1} a_{is}k((c_i-c_s)h)g(t_{n,s},Y_{n,s}) \quad i=1,\dots,m \tag{2.5}$$

and the lag term is given by

$$\bar{F}_n(t_{n,i}) = f(t_{n,i}) + h\sum_{r=0}^{n-1}\sum_{s=1}^{m} b_s k(t_{n,i}-t_{r,s})g(t_{r,s},Y_{r,s}) \quad i=1,\dots,m+1. \tag{2.6}$$

An m-stage HVRK method, applied to equation (1.1) reads

$$y_{n+1} = Y_{n,m} \quad n=0,\dots,N_t-1 \tag{2.7}$$

with $Y_{n,i}$ determined by

$$Y_{n,i} = \bar{F}_n(t_{n,i}) + hc_i\sum_{l=1}^{m} b_l k(c_i(1-c_l)h)g\Big(t_n+c_ic_lh, \sum_{s=1}^{m} L_s(c_ic_l)Y_{n,s}\Big) \quad i=1,\dots,m, \tag{2.8}$$

where the lag terms $\bar{F}_n(t_{n,i})$ are given by (2.6) for $i=1,\dots,m$ and we remind that in this case is $c_m=1$.

A straightforward implementation of classical VRK method (either of Pouzet type or of de Hoog and Weiss) would require $\mathcal{O}(d^2N_t^2)$ operations and $\mathcal{O}(d^2N_t)$ memory for computing the numerical solution over N_t time steps. The reduction of the computational cost to $\mathcal{O}(d^2N_t\log N_t)$ in time and $\mathcal{O}(d^2\log N_t)$ in space will be obtained by taking into account of the peculiarity of the considered equation and by using the following inverse Laplace transform approximation formula [14]

$$u(t) = \frac{1}{2\pi i}\int_{\Gamma_l} U(\lambda)e^{t\lambda}d\lambda \approx \sum_{j=-N}^{N} \omega_j^{(l)}U\big(\lambda_j^{(l)}\big)e^{t\lambda_j^{(l)}} \quad t\in\bar{I}_l. \tag{2.9}$$

This formula locally approximates a function $u(t)$ by sums of exponentials on a sequence of fast growing intervals $\bar{I}_l = [B^{l-1}h+(c_1-1)h,(2B^l-1)h]$, covering $[h,T]$, where $B>1$ is an integer, Γ_l is a suitably chosen Talbot contour [17, 19] and the weights $\omega_j^{(l)}$ are obtained using the trapezoidal rule to a parametrization of the contour integral on Γ_l. The number $2N+1$ of quadrature points $\lambda_j^{(l)}$ chosen on Γ_l, is independent of l and it is much smaller than would be required for a uniform approximation on the whole interval I. What's more, by using an M nodes trapezoidal rule, the resulting error satisfies

$$\|E(t)\|_{t\in I_l} = O(e^{-c\sqrt{M}}), \tag{2.10}$$

where the positive constant c depends on the distance of the singularities of the Laplace transform $U(s)$ of the function $u(t)$ with respect to the Talbot contour.

REMARK 2.1. The use of the formula (2.9) for the inverse Laplace transform approximation is the core of the scheme proposed by Lubich and Shädle in [14]. As a matter of fact, by opportunely inserting the (2.9) in (2.6), we will be able to efficiently compute the lag term at each time step by exploiting its evaluations at the previous time steps (see Section 2.1).

REMARK 2.2. Note that the formula (2.9) can be applied when $t > 0$. An implicit PVRK method involves also evaluations of the kernel $k(t)$ with $t < 0$, thus motivating our choice of considering HVRK methods.

2.1 Fast computation of the lag terms.

As in [14] let L be the smallest integer for which $t_{n+1} < 2B^L h$ and for $l = 1, 2, \ldots, L-1$ determine the integer $q_l \geq 1$ such that $\tau_l = q_l B^L h$ satisfies $t_{n+1} - \tau_l \in [B^l h, (2B^l - 1)h]$, with $\tau_0 = t_n$ and $\tau_L = 0$. It is easy to verify that $[t_{n,i} - \tau_{l-1}, t_{n,i} - \tau_l] \subseteq \bar{I}_l$.

In order to use the formula (2.9) for the approximation of the kernel, we have to split the sum over r in (2.6) as

$$\bar{F}_n(t_{n,i}) = f(t_{n,i}) + h \sum_{l=1}^{L} \sum_{r=\frac{\tau_l}{h}}^{\frac{\tau_{l-1}}{h}-1} \sum_{s=1}^{m} b_s k(t_{n,i} - t_{r,s}) g(t_{r,s}, Y_{r,s}) \qquad i = 1, \ldots, m+1. \tag{2.11}$$

In this way for each fixed l the argument $t_{n,i} - t_{r,s}$ of the kernel $k(t)$ belongs to $[t_{n,i} - \tau_{l-1}, t_{n,i} - \tau_l] \subseteq \bar{I}_l$. Now we approximate each component $k_{\alpha\beta}(t)$ of the kernel $k(t)$ in (2.11) with the formula (2.9) evaluated in $t = t_{n,i} - t_{r,s}$,

$$k_{\alpha\beta}(t_{n,i} - t_{r,s}) \approx \sum_{j=-N_{\alpha\beta}}^{N_{\alpha\beta}} \omega_j{}^{(l,\alpha,\beta)} K_{\alpha,\beta}(\lambda_j{}^{(l,\alpha,\beta)}) e^{(t_{n,i} - t_{r,s})\lambda_j{}^{(l,\alpha,\beta)}}, \qquad \alpha, \beta = 1, \ldots, d, \tag{2.12}$$

where the points $\lambda_j{}^{(l,\alpha,\beta)}$ belong to a Talbot contour $\Gamma_{l,\alpha,\beta}$ associated to the complex function $K_{\alpha,\beta}(s)$. The number of points chosen on each contour and the quadrature weights also depend on α and β.

The approximation $\bar{F}_{n,i} = ({}^1\bar{F}_{n,i}, \ldots, {}^d\bar{F}_{n,i})^T$ of $\bar{F}(t_{n,i})$ is obtained by inserting (2.12) in (2.11)

$$\begin{aligned} {}^\alpha\bar{F}_{n,i} = f_\alpha(t_{n,i}) + \sum_{\beta=1}^{d} \sum_{l=1}^{L} \sum_{j=-N}^{N} \omega_j{}^{(l,\alpha,\beta)} K_{\alpha,\beta}\big(\lambda_j{}^{(l,\alpha,\beta)}\big) e^{(t_{n,i} - \tau_{l-1})\lambda_j{}^{(l,\alpha,\beta)}} \\ \times z_\beta\big(\tau_{l-1}, \tau_l, \lambda_j^{(l,\alpha,\beta)}\big) \quad i = 1, \ldots, m+1, \quad \alpha = 1, \ldots, d, \end{aligned} \tag{2.13}$$

where

$$(2.14)\quad z_\beta(\tau_{l-1},\tau_l,\lambda) := h \sum_{r=\frac{\tau_l}{h}}^{\frac{\tau_{l-1}}{h}-1} \sum_{s=1}^{m} b_s e^{(\tau_{l-1}-t_{r,s})\lambda} g_\beta(t_{r,s},Y_{r,s}) \quad \beta=1,\dots,d.$$

REMARK 2.3. The HVRK methods have $c_m = 1$, and thus the index i in (2.13) should arrive only up to $i = m$.

The implementation of the formula (2.13) by mean of the direct computation of (2.14) would still lead to a computational cost of $\mathcal{O}(d^2 N_t^2)$. The idea proposed in [14] to reduce the computational cost was based on a new organization in the computation of the function z at each time step, which could exploit its evaluations at the previous time steps. In order to reach the same goal we split the interval $[\tau_l, \tau_{l-1}]$ in subintervals $[\tau_l + t_k, \tau_{l-1} + t_{k+1}]$ of length h, we denote with $\mathbf{z_k} = z(\tau_l + t_k, \tau_l, \lambda)$ and we prove that the following one step formula for the evaluation the function z in the mesh points from τ_l to τ_{l-1} holds.

PROPOSITION 2.1. *Let $z(\tau_{l-1},\tau_l,\lambda)$ be given by (2.14), with $\tau_l = \bar{m}h$, $\tau_{l-1} = \tau_l + \bar{n}h$, then*

$$(2.15)\quad \begin{cases} \mathbf{z_{k+1}} = e^{\lambda h}\mathbf{z_k} + h \sum\limits_{s=1}^{m} b_s e^{(1-c_s)h\lambda} g(\tau_l + t_{k,s}, \tilde{Y}_{k,s}) & k = 0,\dots,\bar{n}-1 \\ \mathbf{z_0} = \mathbf{0}, \end{cases}$$

where $\tilde{Y}_{k,s} = Y_{\bar{m}+k,s}$ and $\mathbf{z_k} \in R^d$.

We can advance the values (2.15) of z by one step for all required values $\lambda_j^{(l)}$ on all Talbot contours in every time step $t_n \to t_{n+1}$, according to the scheme illustrated in [14]. So the computational cost is of $\mathcal{O}(d^2 N_t \log_B N_t)$ operations (see Section 3 for the detailed calculation). Note that the function z in (2.13) does not depend on i, so we have to evaluate it only one time at each step $t_n \to t_{n+1}$ independently on the number stages m. Moreover the computation of z_{k+1} through (2.15) only requires the value z_k of z at the previous step and the values of the stages $\tilde{Y}_{k,r}$, which represent an approximation of the exact solution at the point $\tau_l + t_{k,r} \in [\tau_l + t_k, \tau_l + t_{k+1}]$. So we do not need to keep in memory all the past values, thus leading to a memory requirement of $\mathcal{O}(d^2 \log_B N_t)$.

2.2 Determination of the approximate solution.

Once approximated the lag terms in the points $t_{n,i}$, the next step to follow is to solve the nonlinear system (2.5) or (2.8), after inserting the inverse Laplace transform approximation (2.9).

2.2.1 Explicit extended FPVRK methods.

Now we can use the approximations

$$(2.16)\quad k_{\alpha\beta}((c_i - c_s)h) \approx \sum_{j=-N_{\alpha\beta}}^{N_{\alpha\beta}} \omega_j{}^{(\alpha,\beta)} K_{\alpha\beta}\big(\lambda_j{}^{(\alpha,\beta)}\big) e^{(c_i-c_s)\lambda_j{}^{(\alpha,\beta)}} =: \Psi_{is}(\alpha,\beta),$$

where Ψ_{is} is a matrix of dimension d, the weigths ${\omega_j}^{(\alpha,\beta)}$ and the nodes ${\lambda_j}^{(\alpha,\beta)}$ correspond to a Talbot contour $\Gamma_{0,\alpha,\beta}$ associated to the interval $\bar{I}_0 = [0, h]$. As concerns the increment term (2.4) the evaluations of $k(t)$ are approximated by

$$k_{\alpha\beta}((1-c_i)h) \approx \sum_{j=-N_{\alpha\beta}}^{N_{\alpha\beta}} {\omega_j}^{(\alpha,\beta)} K_{\alpha\beta}\big({\lambda_j}^{(\alpha,\beta)}\big) e^{(1-c_i)h{\lambda_j}^{(\alpha,\beta)}} =: \Psi_i(\alpha,\beta),$$

where Ψ_i is a matrix of dimension d. Thus the stages $\bar{Y}_{n,i}$ can be computed explicitly through the formula

$$\bar{Y}_{n,i} = \bar{F}_n(t_{n,i}) + h\sum_{s=1}^{i-1} a_{is}\Psi_{is} g(t_{n,s}, \bar{Y}_{n,s}) \quad i = 1, \dots, m \tag{2.17}$$

obtained by inserting the approximation (2.16) in the formula (2.5). The approximate solution of (1.1) in the mesh points I_h is obtained by

$$y_{n+1} = \bar{F}_{n,m+1} + h\sum_{i=1}^{m} b_i \Psi_i g(t_{n,i}, \bar{Y}_{n,i}) \quad n = 0, \dots, N_t - 1. \tag{2.18}$$

2.2.2 Implicit FHVRK methods.

As before we can use the approximations

$$k_{\alpha\beta}(c_i(1-c_l)h) \approx \sum_{j=-N_{\alpha\beta}}^{N_{\alpha\beta}} {\omega_j}^{(\alpha,\beta)} K_{\alpha\beta}\big({\lambda_j}^{(\alpha,\beta)}\big) e^{c_i(1-c_l)h{\lambda_j}^{(\alpha,\beta)}} =: \Psi_{il}(\alpha,\beta), \tag{2.19}$$

where Ψ_{il} is a matrix of dimension d, the weigths ${\omega_j}^{(\alpha,\beta)}$ and the nodes ${\lambda_j}^{(\alpha,\beta)}$ correspond to a Talbot contour $\Gamma_{0,\alpha,\beta}$ associated to the interval $\bar{I}_0 = [0, h]$.

Thus the nonlinear system (2.8) becomes

$$\bar{Y}_{n,i} = \bar{F}_{n,i} + hc_i \sum_{l=1}^{m} b_l \Psi_{il} g\Big(t_n + c_i c_l h, \sum_{s=1}^{m} L_s(c_i c_l)\bar{Y}_{n,s}\Big), \tag{2.20}$$

and the approximate solution of (1.1) in the mesh points I_h is obtained by

$$\bar{y}_{n+1} = \bar{Y}_{n,m} \quad n = 0, \dots, N_t - 1. \tag{2.21}$$

Remark 2.4 (Linear case). In the case of linear VIEs the nonlinear systems (2.17) and (2.20) become linear, and they can be written in the form

$$(\mathbf{I} - h\mathbf{D})\bar{\mathbf{Y}}_n = \bar{\mathbf{F}}_n, \tag{2.22}$$

where $\bar{\mathbf{Y}}_n = (\bar{Y}_{n,1}^T, \dots, \bar{Y}_{n,m}^T)^T$, $\bar{\mathbf{F}}_n = (\bar{F}_{n,1}^T, \dots, \bar{F}_{n,m}^T)^T$, $\mathbf{I}$ denotes the identity matrix of order md and $\mathbf{D} = (D_{is})$ is a block square matrix of dimension md.

Each block is defined as

$$D_{is} = \begin{cases} a_{is}\Psi_{is} & \text{FPVRK methods} \\ c_i \sum_{l=1}^{m} b_l \Psi_{il} L_s(c_i c_l) & \text{FHVRK methods,} \end{cases} \tag{2.23}$$

where Ψ_{il} are given by (2.16) for FPVRK methods or by (2.19) for FHVRK methods.

3 Computational cost.

In this section we will give the calculation of the computational cost of the FVRK methods in function of the number N_t of mesh points, proving that it is of $\mathcal{O}(d^2 N_t \log N_t)$ operations and of $\mathcal{O}(d^2 \log N_t)$ memory requirement. We will only take into consideration the FHVRK methods, since for the FPVRK methods the same result can be obtained in a similar way.

In the subsequent computations m will represent the number of stages, L_t the total number of different Talbot contours, $N = \max_{\alpha,\beta=1,\dots,d} N_{\alpha,\beta}$, $M = 2N+1$ the maximum number of points chosen on the Talbot contours. A FHVRK method consists, for each time step t_n, $n = 0, \dots, N_t - 1$, in the following steps:

STEP 1 Evaluate the lag terms (2.13) for $i = 1, \dots, m+1$ and $\alpha = 1, \dots, d$ using all the values of the function z, already computed in the previous time steps.
We can observe that, by construction, the integer L in (2.13) satisfies

$$L \le L_t \le \log_B N_t.$$

As the formula (2.13) involves, for each i and α, a sum for $l = 1, \dots, L$, one for $j = -N, \dots, N$, and one for $\beta = 1, \dots, d$ the number of floating point operations $FLOP_{lag}$ for the lag terms computation is proportional to $d^2 mML$. Thus

$$FLOP_{lag} \le d^2 C_1 mM \log_B N_t.$$

STEP 2 Determine the approximate solution $\bar{y}_{n+1} = \bar{Y}_{n,m}$ by mean of (2.21). Thus we need to solve the nonlinear system of dimension md.
If we solve such system by an iterative method, the computational cost for the solution of the nonlinear system is of

$$FLOP_{sist} = C_2 k m^2 d^2$$

operations, where k is the number of iterations required by the iterative method.

STEP 3 Advance by one step the formula (2.15) for each l (on all Talbot contours) and for $j = -N, \dots, N$. This values will be used for the computation of the lag terms in the subsequent time steps. The formula (2.15) requires, for each l and for each j, the computation of a sum for $s = 1,\dots,m$, thus requiring a number of floating point operations $FLOP_{advance}$ proportional to $dmML_t$. Thus

$$FLOP_{advance} \le dC_3 mM \log_B N_t.$$

Thus the total number of floating point operations $FLOP_{tot}$ in function of the time steps N_t is given by

$$\begin{aligned} FLOP_{tot} &= N_t(FLOP_{lag} + FLOP_{sist} + FLOP_{advance}) \\ &\le (d^2C_1 + dC_3)mMN_t \log_B N_t + d^2C_2 km^2 N_t \end{aligned}$$

and then

$$FLOP_{tot} = \mathcal{O}(d^2 N_t \log_B N_t).$$

As regards the memory requirement for computing the solution of (1.1) over N_t time steps through (2.13), (2.15), (2.20), (2.21) we can observe that we have to store

1. $K_{\alpha,\beta}(\lambda_j^{(l,\alpha,\beta)})$, $\lambda_j^{(l,\alpha,\beta)}$, $\omega_j^{(l,\alpha,\beta)}$, $e^{h\lambda_j^{(l,\alpha,\beta)}}$, for $l = 1,\dots,L_t$ and for $j = -N,\dots,N$, $\alpha,\beta = 1,\dots,d$
2. $z(\tau_{l-1},\tau_l,\lambda_j^{(l)})$, for $l = 1,\dots,L_t$ and for $j = -N,\dots,N$,
3. $Y_{n,i}$, $i = 1,\dots,m$.

The total cost in space of the FVRK methods is less or equal to $Cd^2M \log_B N_t + md$, and thus is $\mathcal{O}(d^2 \log_B N_t)$.

4 Convergence analysis.

Let us denote by $\|\cdot\|_d$ a vector norm on R^d (when applied to a matrix it will denote a compatible matrix norm) and let us consider classical explicit PVRK methods and implicit HVRK methods of order p. The following theorem estabilishes the order of convergence of the corresponding FVRK methods.

THEOREM 4.1. *Let $\bar{e}_n = y(t_n) - \bar{y}_n$ be the error of the FVRK methods (explicit FPVRK or implicit FHVRK). If the function $g(\tau, y)$ is Lipschitz continuous with respect to y then*

$$\max_{1\le n\le N_t} \|\bar{e}_n\|_d = O(h^p) + O(e^{-c\sqrt{M}}). \tag{4.1}$$

PROOF. We prove the thesis for the implicit FHVRK methods, since in the case of explicit FPVRK methods the proof is similar. Let $e_n = y(t_n) - y_n$ and $\bar{e}_n = y(t_n) - \bar{y}_n$, $n = 1,\dots,N_t$ respectively denote the error of the classical HVRK method (2.6), (2.7), (2.8) and of the corresponding FHVRK method (2.13), (2.20), (2.21), and define $\epsilon_n = y_n - \bar{y}_n$.

According to our notation we have

$$\max_{1\le n\le N_t}\|\bar{e}_n\|_d \le \max_{1\le n\le N_t}\|e_n\|_d + \max_{1\le n\le N_t}\|\epsilon_n\|_d.$$

Since we are considering classical methods of order p, it follows that

$$\max_{1\le n\le N}\|e_n\|_d \le C_1 h^p$$

(see [4]).

Let Q be the Lipschitz constant of the function g with respect to y, i.e.

$$\|g(t,y) - g(t,\bar{y})\|_d \le Q\|y-\bar{y}\|_d$$

uniformly with respect to t and for all $y, \bar{y} \in R^d$.

Let us denote by $\eta_{n,i} = Y_{n,i} - \bar{Y}_{n,i} \in R^d$ the error committed in the approximation of the stages and let us define the matrix $M_{i,s}^{(n,k)} \in R^{d\times d}$ by

$$\big(M_{i,s}^{(n,k)}\big)_{\alpha,\beta} = \sum_{j=-N_{\alpha\beta}}^{N_{\alpha\beta}} \omega_j^{(l,\alpha,\beta)} K_{\alpha,\beta}\big(\lambda_j^{(l,\alpha,\beta)}\big) e^{(t_{n,i}-t_{k,s})\lambda_j^{(l,\alpha,\beta)}} \qquad \alpha,\beta = 1,\dots,d.$$

By substracting (2.20) from (2.8), and exploiting the Lipschitz condition on g we obtain

(4.2)
$$\|\eta_{n,i}\|_d \le h\sum_{s=1}^{m} d_{is}\|\eta_{n,s}\|_d + h\sum_{k=0}^{n-1}\sum_{s=1}^{m} q_{i,s}^{(n,k)}\|\eta_{k,s}\|_d + h\big\|E_{n,n}^{(i)}\big\|_d + h\sum_{l=1}^{L}\big\|E_{n,l}^{(i)}\big\|_d \qquad i = 1,\dots,m,$$

where $q_{i,s}^{(n,k)}$, d_{is} are scalar values and $E_{n,n}^{(i)}$, $E_{n,l}^{(i)}$ are vectors of dimension d, defined by:

$$\begin{aligned}
q_{i,s}^{(n,k)} &= Q|b_s|\big\|M_{i,s}^{(n,k)}\big\|_d \quad k = 0,\dots,n-1, \quad i,s = 1,\dots,m,\\
E_{n,l}^{(i)} &= \sum_{r=\frac{\tau_l}{h}}^{\frac{\tau_{l-1}}{h}-1}\sum_{s=1}^{m} b_s\big(k(t_{n,i}-t_{r,s}) - M_{i,s}^{(n,r)}\big)g(t_{r,s}, Y_{r,s}),\\
E_{n,n}^{(i)} &= c_i\sum_{l=1}^{m} b_l(k(c_i(1-c_l)h) - \Psi_{il})g\Big(t_{n,s}, \sum_{s=1}^{m} L_s(c_ic_l)Y_{n,s}\Big),\\
d_{is} &= c_iQ\sum_{l=1}^{m} b_l\|\Psi_{il}\|_d|L_s(c_ic_l)|.
\end{aligned}$$

By using (2.10) we obtain the discrete Gronwall inequality

$$\|\eta_n\|_\infty \le \frac{hC_3}{1-hC_2}\sum_{k=0}^{n-1}\|\eta_k\|_\infty + \frac{C_4}{1-hC_2}e^{-c\sqrt{M}}, \quad n = 0,\dots,N_t-1, \tag{4.3}$$

where $\eta_n = (\|\eta_{n,1}\|_d,\dots,\|\eta_{n,m}\|_d)^T$, $C_2 = \max_i\sum_{s=1}^{m} d_{is}$ and $C_3 = \max_{i,n,k}\sum_{s=1}^{m} q_{i,s}^{(n,k)}$.

Using the Gronwall result (Corollary 1.5.2 in [4]) for (4.3), it follows that $\|\eta_n\|_\infty \leq C_5 e^{-c\sqrt{M}}$ and hence, being $\epsilon_n = \eta_{n,m}$, we obtain

$$\max_{1 \leq n \leq N_t} \|\epsilon_n\|_d \leq C_5 e^{-c\sqrt{M}}.$$

□

Remark 4.1. An immediate consequence of Theorem 4.1 is that, by choosing M such that $e^{-c\sqrt{M}} \leq C_1 h^p$, then the FVRK methods keep the same order of the corresponding classical ones. Thus, since the error due to the approximation of the inverse Laplace transform decreases exponentially with M, it is sufficient to fix a not too high number of points on the Talbot contour in order to preserve the order of convergence of the classical methods.

It immediately follows from Theorem 4.1, Remark 4.1 and Theorem 5.3.5 in [4], that it is possible to achieve local superconvergence at the mesh points by opportunely choosing the parameters of the VRK methods:

Corollary 4.1. *Let f, $k \in C^{2m-v}$, with $v \in \{0,1,2\}$. Then it is possible to choose M such that the following statements hold:*

(i) If the nodes $\{c_i\}$ are the Radau II points for $(0,1]$*, then the FHVRK* (2.13), (2.15), (2.20), (2.21) *satisfies, for $v = 1$,*

$$\max_{1 \leq n \leq N_t} \|\bar{e}_n\|_d = O(h^{2m-1}).$$

(ii) If the nodes $\{c_i\}$ are the Lobatto points for $[0,1]$*, then the FHVRK method* (2.13), (2.15), (2.20), (2.21) *satisfies, for $v = 2$,*

$$\max_{1 \leq n \leq N_t} \|\bar{e}_n\|_d = O(h^{2m-2}).$$

(iii) Let the nodes $\{c_i\}$ are the m Gauss points for (0,1), $c_{m+1} = 1$. Furthermore, suppose to consider a modification of the FHVRK method given by (2.13), (2.15), (2.20) *for the computation of the stages, but with the approximate solution calculated by mean of* (2.18)*, then, for $v = 0$,*

$$\max_{1 \leq n \leq N_t} \|\bar{e}_n\|_d = O(h^{2m}).$$

5 Numerical results.

In the numerical experiments we tested the performances of the FVRK methods in terms of order of convergence and computational cost, in order to validate the theoretical results of Sections 3 and 4. In addition, we have comparated the performarces of the FVRK methods with that ones of a pair of extended explicit Bel'tyukov Runge–Kutta (EBVRK) formulas [18]. Our methods were implemented in MATLAB and the numerical experiment have been performed on different text examples.

For the first purpose we report the results obtained on the two test problems:

- the linear Volterra integral equation of renewal theory taken from [4, 9]:

$$y(t) = 1 - e^{-\lambda t}(1 + \lambda t) + \int_0^t \lambda^2 (t-\tau) e^{-\lambda(t-\tau)} y(\tau) d\tau, \quad t \in [0, 10], \tag{5.1}$$

  with $K(s) = (\frac{\lambda}{\lambda+s})^2$, $\lambda = 1/2$, and exact solution $y(t) = \frac{1}{4}(2\lambda t - 1 + e^{-2\lambda t})$;
- the nonlinear equation given in [11], arising in the analysis of neural networks with post inhibitory rebound:

  (5.2)
$$y(t) = 1 + \int_0^t (t-\tau)^3 (4 - t + \tau) e^{-t+\tau} \frac{y^4(\tau)}{1 + 2y^2(\tau) + 2y^4(\tau)} d\tau, \quad t \in [0, 10],$$

  with $K(s) = \frac{24s}{(1+s)^5}$ and reference solution $y(10) = 1.25995582337233$, obtained numerically by using different codes with very stringent tolerances;
- the linear system given in [4]:

  (5.3)
$$\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{259}{9} - 24\cos t - \frac{16}{9}\cos 3t \end{pmatrix} + \int_0^t \begin{pmatrix} 0 & -1 \\ -25 & 0 \end{pmatrix} \begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} dt,$$
$$t \in [0, 5],$$

  with $K_{11}(s) = K_{22}(s) = 0$, $K_{12}(s) = -1/s$, $K_{21}(s) = -25/s$ and exact solution:

$$y_1(t) = \sin(t) + \sin(3t)/3 + \sin(5t)/5$$
$$y_2(t) = \cos(t) + \cos(3t) + \cos(5t).$$

5.1 *Convergence.*

The following FVRK methods have been used, where p denotes the order of the method, according to Theorem 4.1 and Corollary 4.1:

IMPLICIT METHODS:

$L2$: 2-points Lobatto ($c_1 = 0$, $c_2 = 1$), $p = 2$;

$R3$: 3-points Radau II $\left(c_1 = \frac{4-\sqrt{6}}{10},\ c_2 = \frac{4+\sqrt{6}}{10},\ c_3 = 1\right)$, $p = 5$;

$G3$: 3-points Gauss $\left(c_1 = \frac{5-\sqrt{15}}{10},\ c_2 = 1/2,\ c_3 = \frac{5+\sqrt{15}}{10}\right)$, $p = 6$.

EXPLICIT METHODS:

$E3$: 3-points with Butcher array:

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 2/3 & 2/3 & 0 & 0 \\ 2/3 & 5/12 & 1/4 & 0 \\ \hline & 1/4 & -1/4 & 1 \end{array}, \quad p = 3;$$

$E4$: classical 4-points with Butcher array:

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}\ , \quad p = 4.$$

The number of correct significant digits cd at the end point is defined to be

$$cd := -\log_{10}\left(\|y(T) - y_{N_t}\|_d \,/\, \|y(T)\|_d\right).$$

In Figures 5.1–5.3 we report the value of cd obtained by the application of each method respectively to the Equations (5.1)–(5.3), with respect the number N_t of mesh points.

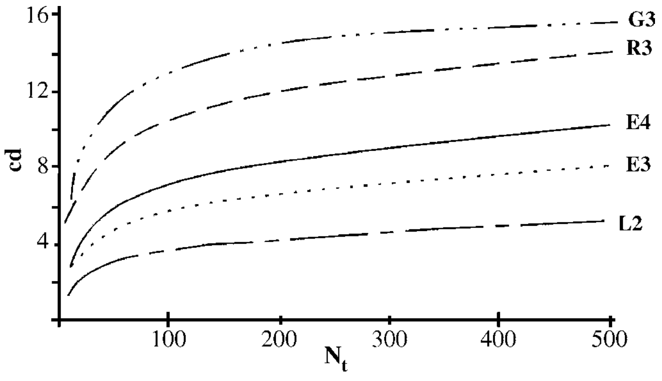


Figure 5.1: Number of correct significant digits for problem (5.1).

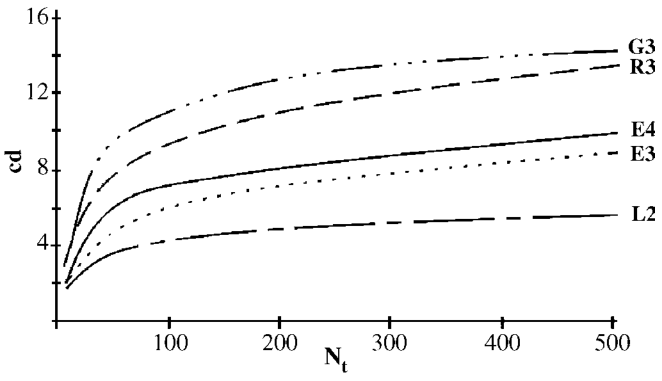


Figure 5.2: Number of correct significant digits for problem (5.2).

In Tables 5.1–5.3 we report the numerical results and we compute a numerical estimation of the order of the method with the formula $p(h) = \frac{cd(h)-cd(2h)}{\log_{10} 2}$ for a fixed h, which shows that our methods produce the expected order.

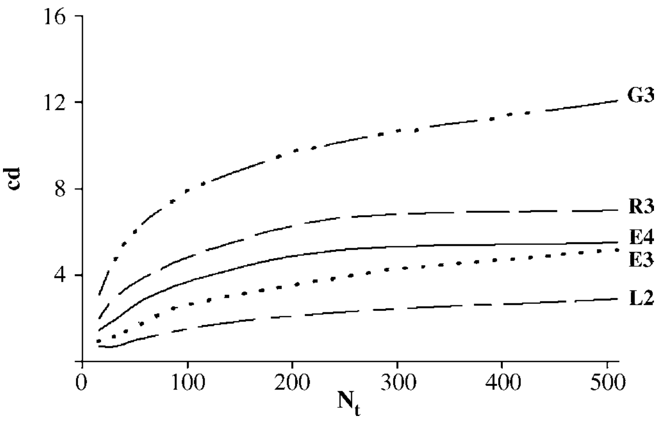


Figure 5.3: Number of correct significant digits for problem (5.3).

Table 5.1: Number of correct significant digits for problem (5.1) at $t = T = 10$.

method	$N_t = 16$	$N_t = 32$	$N_t = 64$	$N_t = 128$	$p(h = \frac{10}{128})$
$L2$	1,82	2,41	3,01	3,61	1,99
$E3$	3,23	4,07	4,94	5,83	2,96
$E4$	3,78	4,93	6,10	7,29	3,95
$R3$	6,18	7,66	9,16	10,67	5,02
$G3$	7,98	9,76	11,56	13,35	5,95

Table 5.2: Number of correct significant digits for problem (5.2) at $t = T = 10$.

method	$N_t = 16$	$N_t = 32$	$N_t = 64$	$N_t = 128$	$N_t = 256$	$p(h = \frac{10}{256})$
$L2$	2,82	3,27	3,84	4,44	5,04	1,99
$E3$	3,05	4,14	5,27	6,41	7,35	3,12
$E4$	3,38	4,96	6,51	7,20	8,26	3,52
$R3$	4,73	6,45	8,33	9,91	11,41	4,98
$G3$	6,35	8,62	9,26	11,53	13,17	5,45

Table 5.3: Number of correct significant digits for problem (5.3) at $t = T = 5$.

method	$N_t = 32$	$N_t = 64$	$N_t = 128$	$N_t = 256$	$p(h = \frac{5}{256})$
$L2$	0,72	1,13	1,71	2,30	1,99
$E3$	1,11	2,02	3,02	4,02	3,33
$E4$	1,97	2,99	4,07	5,20	3,74
$R3$	3	4	5,30	6,70	4,65
$G3$	4,86	6,66	8,46	10,27	5,99

5.2 Computational cost.

In order to verify that our methods have a computational cost of order $\mathcal{O}(N_t \log_B N_t)$ in Figure 5.4 we plot the cpu-time in seconds versus N_t, obtained by applying the R3 FVRK method to Equation (5.1) on an Intel Pentium 4/3,2 GHz. We also plot the lines corresponding to algorithms of order $\mathcal{O}(N_t)$ and $\mathcal{O}(N_t^2)$. The picture shows that our method perfectly follows the behaviour of the function $CN_t \log_B N_t$.

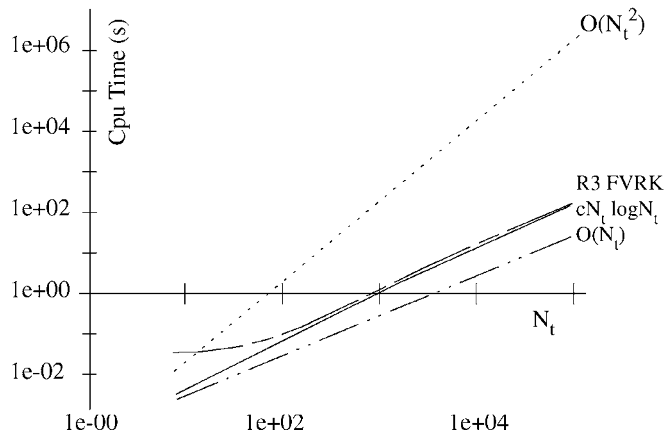


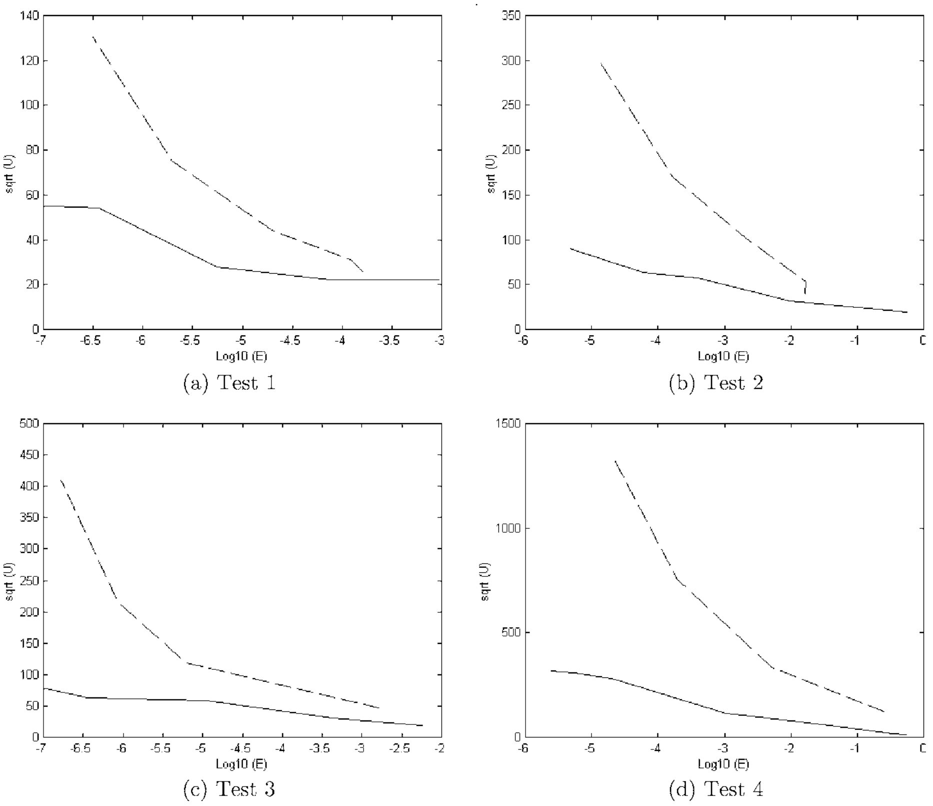
Figure 5.4: Computational cost.

In a second session of experiments we have compared our fast implementation of VRK methods with a well-chosen implementation of Bel'tyukov Runge–Kutta (EBVRK) methods. We have tested the efficiency, on equal accuracy, of the explicit *IV*-order FVRK (E4) and EBVRK methods on the four standard test problems of the convolution type specified in [2] and summarized in Table 5.4.

Table 5.4: A summary of the four test problems.

	$g(t)$	$k(t)$	T
1	$t^2 \exp(-t)/2$	$(t-\tau)^2 \exp(\tau - t) y(\tau)/2$	5
2	$1 + \sin^2(t)$	$-3\sin(t-\tau) y^2(\tau)$	5
3	1	$(t-\tau)^3 (4 - t + \tau) \exp(\tau - t) y^4(\tau)/(1 + 2y^2(\tau) + 2y^4(\tau))$	10
4	$\exp(-t)$	$\exp(\tau - t)(y(\tau) + \exp(-y(\tau)))$	40

In Figure 5.5 we compare the efficiency curves for the two methods. There we plot the curves of square root of u against the absolute end-point global error for the four convolution test problems of Table 5.4, where u is the number of kernel evaluations. For the method E4 the number of kernel evaluations we have considered is the sum of the evaluations of the Laplace transforms K and of



(a) Test 1 (b) Test 2

(c) Test 3 (d) Test 4

Figure 5.5: Efficiency curves of $u^{1/2}$ against the absolute end-point global error (E) for the four test problems in Table 5.4, where u is the number of kernel evaluations. Solid line FVRK E4, dashed-dot line EBVRK 4.

the exponentials in (2.13) and (2.16). The choice of $u^{1/2}$ was done because the number of kernel evaluations in [18] varies as N_t^2.

6 Concluding remarks.

In this work we constructed fast VRK method for the Equation (1.1) of order p that can be implemented with a computational cost of $\mathcal{O}(d^2 N_t \log N_t)$ operations and with a memory requiremet of $\mathcal{O}(d^2 \log N_t)$ to compute the numerical solution over N_t time steps. This was possible by exploiting the Laplace transform of the kernel and the fact that the equation is of convolution type. We proved that the convergence and stability properties depend on the number M of points chosen on the Talbot contour for the inverse Laplace transform approximation. In particular with a suitable choice of M, the order of convergence p is the same of the corresponding classical VRK method. The numerical experiments clearly confirm the theoretical expectations.

Acknowledgement.

We would like to thank Christian Lubich for suggesting us many helpful ideas and for fruitful discussions. We extend our thanks to the anonymous referee for his comments on the paper.

REFERENCES

1. A. Bellen, Z. Jackiewicz, R. Vermiglio, and M. Zennaro, *Stability analysis of Runge–Kutta methods for Volterra integral equations of second kind*, IMA J. Numer. Anal., 10 (1990), pp. 103–118.
2. J. G. Blom and 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), pp. 806–830.
3. H. Brunner and E. Messina, *Time-stepping methods for Volterra–Fredholm integral equations*, Rend. Mat. Appl., VII. Ser., 23 (2003), pp. 329–342.
4. H. Brunner and P. J. van der Houwen, *The numerical solution of Volterra equations*, CWI Monographs, vol. 3, North-Holland, Amsterdam, 1986.
5. A. Cardone, E. Messina, and E. Russo, *A fast iterative method for Volterra–Fredholm integral equations*, J. Comput. Appl. Math., 189(1–2) (2006), pp. 568–579.
6. D. Conte, I. Del Prete, *Fast collocation methods for Volterra integral equations of convolution type*, J. Comput. Appl. Math., 196(2) (2006), pp. 652–663.
7. M. R. Crisci, E. Russo, and A. Vecchio, *On the stability of the one-step exact collocation methods for the numerical solution of the second kind Volterra integral equation*, BIT, 29 (1989), pp. 258–269.
8. F. de Hoog, R. Weiss, *Implicit Runge–Kutta methods for second kind Volterra integral equations*, Numer. Math., 23 (1975), pp. 199–213.
9. Z. S. Deligonoul, S. Bilgen, *Solution of the Volterra equation of renewal theory with the Galerkin technique using cubic spplines*, J. Stat. Comput. Simulation, 20 (1984), pp. 37–45.
10. D. Givoli, *Numerical methods for problems in infinite domains*, Elsevier Science Publishers, Amsterdam, 1992.
11. E. Hairer, C. Lubich, and M. Schlichte, *Fast numerical solution of nonlinear Volterra convolution equations*, SIAM J. Sci. Stat. Comput., 6 (1985), pp. 532–541.
12. H. Han, L. Zhu, H. Brunner, and J. Ma, *The numerical solution of parabolic Volterra integro-differential equations on unbounded spatial domains*, Appl. Numer. Math., 55 (2005), pp. 83–99.
13. R. Hiptmair and A. Schädle, *Non-reflecting boundary conditions for Maxwell's equations*, Computing, 71(3) (2003), pp. 265–292.
14. C. Lubich and A. Schädle, *Fast convolution for non-reflecting boundary conditions*, Siam. J. Sci. Comput., 24 (2002), pp. 161–182.
15. C. Lubich, *Convolution quadrature and discretized operational calculus II*, Numer. Math., 52 (1988), pp. 413–425.
16. R. K. Miller, *Nonlinear Volterra Integral Equations*, W. A. Benjamin, Menlo Park, CA, 1971.
17. M. Rizzardi, *A modification of Talbot's method for the simultaneous approximation of several values of the inverse Laplace Transform*, ACM Trans. Math. Softw., 21(4) (1995), pp. 347–371.
18. P. W. Sharp and J. H. Verner: *Some extended explicit Bel'tyukov pairs for Volterra integral equations of the second kind*, SIAM J. Numer. Anal., 38(2) (2000), pp. 347–359.
19. A. Talbot, *The accurate numerical inversion of Laplace Transforms*, J. Inst. Math. Appl., 23 (1979), pp. 97–120.