Lietuvos matematikos rinkinys
Proc. of the Lithuanian Mathematical Society, Ser. A
Vol. 66, 2025, pages 16–29
https://doi.org/10.15388/LMR.2025.44494

Stability analysis of an implicit and explicit numerical method for Volterra integro-differential equations with kernel $K(x, y(t), t)$

Justin Steven Calder Prentice

University of Johannesburg, Johannesburg, South Africa

Corresponding author: jpmsro@mathsophical.com

Received April 1, 2025; published December 21, 2025

Abstract. We present implicit and explicit versions of a numerical algorithm for solving a Volterra integro-differential equation. These algorithms are an extension of our previous work, and cater for a kernel of general form. We use an appropriate test equation to study the stability of both algorithms. We prove that the implicit method is unconditionally stable in the third quadrant. We determine the stability region in the third quadrant for the explicit method numerically. The explicit method has a bounded region close to the origin. We perform several calculations to demonstrate our results.

Keywords: Volterra; stability; integro-differential; numerical

AMS Subject Classification: 65R20, 65G99

1 Introduction

Many techniques exist for solving Volterra integro-differential equations (IDEs), such as Adomian decomposition [5], Laplace decomposition [4], Galerkin methods [13], Haar functions [14], homotopy perturbation [12] and more [10, 3, 20, 9, 2, 18, 19, 15, 11], including Runge-Kutta methods [1, 6].

Recently, we described a numerical method for solving the Volterra integro-differential equation

$$y^{(n)}(x) = f(x, y) + \int_{x_0}^{x} K dt, \quad x > x_0 \tag{1}$$

with various specific forms for the kernel K [16]. In this paper, a sequel to [16], we consider the more general kernel

$$K = K\big(x, y(t), t\big).$$

We modify our previous algorithm appropriately, and we consider both implicit and explicit forms of the resulting algorithm. Our methods are straightforward and easy to program, and they are easily combined with Richardson extrapolation for strict error control (we have satisfied tolerances of 10^{-12} in our examples). Our attention will primarily be focused on the stability of these algorithms. Although this work forms part of a larger project, which will be discussed elsewhere, we believe that our study of stability here is sufficiently interesting to be shared with the community.

2 Algorithm

To begin with, we consider the case of $n = 1$ in (1). We describe the more general case in Appendix. We partition the interval of interest, denoted $[x_0, x_N]$, by means of the *equispaced* nodes

$$x_0 < x_1 < x_2 < \cdots < x_N. \tag{2}$$

The spacing between the nodes – the *stepsize* – is denoted h.

Using the initial value

$$y(x_0) = y_0,$$

we compute the solution at x_1 via

$$\begin{aligned} y_1 &= y_0 + hf(x_1, y_1) + h\int_{x_0}^{x_1} K(x_1, y(t), t)dt \\ &= y_0 + hf(x_1, y_1) + h\left(\frac{h}{2}\right)\big(K(x_1, y_0, x_0) + K(x_1, y_1, x_1)\big), \end{aligned}$$

and the solution at x_2 via

$$\begin{aligned} y_2 = {}& y_1 + hf(x_2, y_2) + h\int\limits_{x_0}^{x_2} K\big(x_2, y(t), t\big)dt \\ &= y_1 + hf(x_2, y_2) \\ &\quad + \frac{h^2}{2}\big(K(x_2, y_0, x_0) + 2K(x_2, y_1, x_1) + K(x_2, y_2, x_2)\big). \end{aligned}$$

In both cases, the integral has been approximated using the composite Trapezium Rule with values for t obtained from the nodes. It must be appreciated that, because the kernel is dependent on x_i – the upper limit of the integral – the integral must be calculated in its entirety at each iteration. Hence, there will be an increasing number of terms in the Trapezium approximation as the iteration proceeds.

In general we have

$$\begin{aligned} y_{i+1} &= y_i + hf(x_{i+1}, y_{i+1}) + h\int_{x_0}^{x_{i+1}} K\big(x_{i+1}, y(t), t\big)dt \\ &= y_i + hf(x_{i+1}, y_{i+1}) \\ &\quad + \frac{h^2}{2}\Bigg(\sum_{j=0}^{i+1} 2K(x_{i+1}, y_j, x_j) - K(x_{i+1}, y_0, x_0) - K(x_{i+1}, y_{i+1}, x_{i+1})\Bigg). \end{aligned} \tag{3}$$

The presence of y_{i+1} on both sides identifies this as an *implicit* algorithm. In Appendix we provide an insight into how this equation can be solved for y_{i+1} using Newton's Method.

The *explicit* form of (3) is given by

$$\begin{aligned} y_{i+1} &= y_i + hf(x_i, y_i) \\ &\quad + \frac{h^2}{2}\Bigg(\sum_{j=0}^{i} 2K(x_i, y_j, x_j) - K(x_i, y_0, x_0) - K(x_i, y_i, x_i)\Bigg). \end{aligned} \tag{4}$$

3 Stability

The test equation we use here is

$$\begin{aligned} y'(x) &= \lambda(y-1) + \gamma\int_0^x y(t)dt \\ y(0) &= 2 \end{aligned} \tag{5}$$

with solution

$$\begin{aligned} y(x) &= e^{m_1 x} + e^{m_2 x} \\ m_1 &= \frac{\lambda - \sqrt{\lambda^2 + 4\gamma}}{2}, \quad m_2 = \frac{\lambda + \sqrt{\lambda^2 + 4\gamma}}{2} \end{aligned}$$

when m_1 and m_2 are real ($\lambda^2 + 4\gamma \geqslant 0$), and

$$y(x) = 2e^{\frac{\lambda x}{2}} \cos\left(\frac{\sqrt{|\lambda^2 + 4\gamma|}}{2}x\right),$$

when m_1 and m_2 are complex ($\lambda^2 + 4\gamma < 0$). This test is equation is similar to that used elsewhere [7, 8].

In (5), λ is intended to represent $\partial f/\partial y$ and γ is intended to represent $\partial K/\partial y$. When $\lambda > 0$ and/or $\gamma > 0$, at least one of m_1 and m_2 will also be greater than zero, ensuring that the solution $y(x)$ is an increasing function of x. Note that if $\gamma > 0$, m_1 and m_2 cannot be complex (thus obviating the oscillatory solution), and if $\lambda > 0$ and m_1 and m_2 are complex, then the oscillatory solution has exponentially increasing amplitude. In all of these cases, $|y(x)| \to \infty$ as $x \to \infty$. Only when $\lambda \leqslant 0$ and $\gamma \leqslant 0$

do we have the possibility that $|y(x)|$ decreases as $x \to \infty$. We acknowledge that when $\lambda = \gamma = 0$, we have the constant solution $y(x) = 2$.

The property of *stability* requires that the numerical solution to the test equation qualitatively mimics the test solution in the sense $|y(x)|$ decreases as $x \to \infty$. This means that, at the very least, $|y_i| < y(0) = 2$ for all i, when $\lambda < 0$ and $\gamma \leqslant 0$, and $|y_i| = 2$ when $\lambda = \gamma = 0$.

3.1 Implicit case

The *stability function* for (3) is obtained by applying (3) to the test equation:

$$\begin{aligned} y_1 &= y_0 + h\big(\lambda(y_1 - 1)\big) + \left(\frac{\gamma h^2}{2}\right)(y_0 + y_1) \\ &= 2 + zy_1 - z + \frac{w}{2}(2 + y_1) \\ \Rightarrow y_1 &= \frac{2 - z + w}{1 - z - \frac{w}{2}} = 2\left(\frac{2 - z + w}{2 - 2z - w}\right) \\ &\equiv P_1 = P_1(z, w), \end{aligned} \tag{6}$$

where we have defined $z \equiv h\lambda$, $w \equiv h^2\gamma$ and P_1 is the stability function at x_1 – i.e. the numerical solution of the test equation at x_1. We note that, in the third quadrant,

$$P_1 = 2\left(\frac{2 + |z| - |w|}{2 + 2|z| + |w|}\right) \quad \Rightarrow \quad |P_1| \leqslant 2. \tag{7}$$

It can be shown that

$$\begin{aligned} P_i = P_i(z, w) \equiv y_i &= 2\left(\frac{P_{i-1} - z + \frac{w}{2}\left(P_0 + 2P_{i-1} + \sum_{j=1}^{i-2} 2P_j\right)}{2 - 2z - w}\right) \\ &= 2\left(\frac{P_{i-1} + wP_{i-1} + T_{i-1}}{2 - 2z - w}\right), \end{aligned} \tag{8}$$

for $i \geqslant 2$. This recursion requires the definition $P_0 \equiv y_0 = 2$, and we define

$$\begin{aligned} T_{i-1} &\equiv -z + \frac{w}{2}\left(P_0 + \sum_{j=1}^{i-2} 2P_j\right) \\ T_1 &\equiv -z + \frac{wP_0}{2}. \end{aligned}$$

3.1.1 Stability analysis for the third quadrant

For stability to persist in the numerical solution, we must demand

$$\big|P_i(z, w)\big| \leqslant 2$$

for all i, whenever $\lambda < 0$ and $\gamma \leqslant 0$, and $|P_i| = 2$ when $\lambda = \gamma = 0$. For a given stepsize h, this is implies that $(z, w) = (h\lambda, h^2\gamma)$ is located in the third quadrant in the $z - w$ system.

We define the *stability region* S in the third quadrant as

$$S \equiv \{(z,w) \mid z<0,\ w \leqslant 0 \text{ and } |P_i(z,w)| < 2 \text{ for all } i\} \cup \{(0,0)\}.$$

The inclusion of $(z,w)=(0,0)$ caters for the case $\lambda=\gamma=0$.

To investigate stability in the third quadrant, we consider, for $i \geqslant 2$,

$$P_i = 2\left(\frac{P_{i-1}+wP_{i-1}+T_{i-1}}{2-2z-w}\right) = \frac{P_{i-1}+wP_{i-1}+T_{i-1}}{1-z-\frac{w}{2}}$$

$$T_i \equiv -z + \frac{w}{2}\left(P_0 + \sum_{j=1}^{i-1} 2P_j\right) = T_{i-1} + wP_{i-1}$$

which give the system

$$\begin{aligned}
\begin{bmatrix} P_i \\ T_i \end{bmatrix} &= \begin{bmatrix} \frac{1+w}{1-z-\frac{w}{2}} & \frac{1}{1-z-\frac{w}{2}} \\ w & 1 \end{bmatrix} \begin{bmatrix} P_{i-1} \\ T_{i-1} \end{bmatrix} \\
\Rightarrow \quad \begin{bmatrix} P_i \\ T_i \end{bmatrix} &= \begin{bmatrix} \frac{1+w}{1-z-\frac{w}{2}} & \frac{1}{1-z-\frac{w}{2}} \\ w & 1 \end{bmatrix} \begin{bmatrix} P_{i-1} \\ T_{i-1} \end{bmatrix} \\
&\equiv M \begin{bmatrix} P_{i-1} \\ T_{i-1} \end{bmatrix} = M^{i-1} \begin{bmatrix} P_1 \\ T_1 \end{bmatrix}.
\end{aligned} \tag{9}$$

The eigenvalues of the transfer matrix M are

$$\phi = \frac{4+w-2z \pm \sqrt{4z^2+w^2-4wz+16w}}{2(2-w-2z)}.$$

We consider four cases:

(1) $z<0$, $w<0$:

If $4z^2+w^2-4wz+16w<0$, we have

$$\phi\phi^* = |\phi|^2 = \frac{2}{2-w-2z} = \frac{2}{2+|w|+2|z|} < 1$$

in the third quadrant. If $4z^2+w^2-4wz+16w \geqslant 0$, then

$$\begin{aligned}
\sqrt{4z^2+w^2-4wz+16w} &= \sqrt{4|z|^2+|w|^2-4|w||z|-16|w|} \\
&< \sqrt{4|z|^2+|w|^2} \leqslant \sqrt{(2|z|+|w|)^2} = 2|z|+|w| \\
\Rightarrow \quad |\phi| &\leqslant \frac{4+|w|+2|z|+|\sqrt{4z^2+w^2-4wz+16w}|}{2(2+|w|+2|z|)} \\
&< \frac{4+|w|+2|z|+|2z|+|w|}{4+2|w|+4|z|} = 1
\end{aligned}$$

in the third quadrant. Since $|\phi| < 1$, we have

$$\lim_{i\to\infty}\begin{bmatrix} P_i \\ T_i \end{bmatrix} = \lim_{i\to\infty} M^{i-1}\begin{bmatrix} P_1 \\ T_1 \end{bmatrix} = 0.$$

This, with (7), implies that P_i does not increase in magnitude as $i \to \infty$, so that

$$\left|P_i(z,w)\right| \leqslant \left|P_1(z,w)\right| \leqslant \left|P_0(z,w)\right| = 2$$

when (z,w) is in the third quadrant.

(2) $z = 0$, $w < 0$:

If $w^2 + 16w < 0$, we have

$$\phi\phi^* = |\phi|^2 = \frac{2}{2-w} = \frac{2}{2+|w|} < 1$$

in the third quadrant. If $w^2 + 16w \geqslant 0$, then

$$\begin{aligned} \sqrt{w^2+16w} &= \sqrt{w^2-16|w|} < w \\ \Rightarrow \quad |\phi| &\leqslant \frac{4+|w|+|\sqrt{w^2+16w}|}{2(2+|w|)} \\ &< \frac{4+|w|+|w|}{2(2+|w|)} = 1 \end{aligned}$$

in the third quadrant. Since $|\phi| < 1$, we have the same result as in **(1)**.

(3) $z = 0$, $w = 0$:

When $z = w = 0$, we have $\lambda = \gamma = 0$ and

$$M = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad T_1 = 0, \quad P_1 = 2,$$

so that, for $i \geqslant 2$,

$$\begin{bmatrix} P_i \\ T_i \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^{i-1}\begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix},$$

i.e. $|P_i| = 2$ when $\lambda = \gamma = 0$.

(4) $z < 0$, $w = 0$:

Here, we have, with $T_1 = |z|$,

$$M = \begin{bmatrix} \frac{1}{1+|z|} & \frac{1}{1+|z|} \\ 0 & 1 \end{bmatrix} \quad \Rightarrow \quad M^{i-1} = \begin{bmatrix} \left(\frac{1}{1+|z|}\right)^{i-1} & \sum_{j=1}^{i-1}\left(\frac{1}{1+|z|}\right)^j \\ 0 & 1 \end{bmatrix}$$

$$\begin{aligned} \Rightarrow \quad \lim_{i\to\infty}\begin{bmatrix} P_i \\ T_i \end{bmatrix} &= \lim_{i\to\infty} M^{i-1}\begin{bmatrix} P_1 \\ T_1 \end{bmatrix} = \lim_{i\to\infty}\begin{bmatrix} \left(\frac{1}{1+|z|}\right)^{i-1} & \sum_{j=1}^{i-1}\left(\frac{1}{1+|z|}\right)^j \\ 0 & 1 \end{bmatrix}\begin{bmatrix} P_1 \\ |z| \end{bmatrix} \\ &= \lim_{i\to\infty}\begin{bmatrix} P_1\left(\frac{1}{1+|z|}\right)^{i-1} + |z|\sum_{j=1}^{i-1}\left(\frac{1}{1+|z|}\right)^j \\ |z| \end{bmatrix} = \begin{bmatrix} 1 \\ |z| \end{bmatrix}. \end{aligned} \tag{10}$$

When $z < 0$, $w = 0$ the test eqation becomes $y' = \lambda(y-1)$, which, with $y(0) = 2$, has solution $y(x) = e^{\lambda x} + 1$. Thus, $\lim_{i\to\infty} y(x) = 1$, consistent with (10).

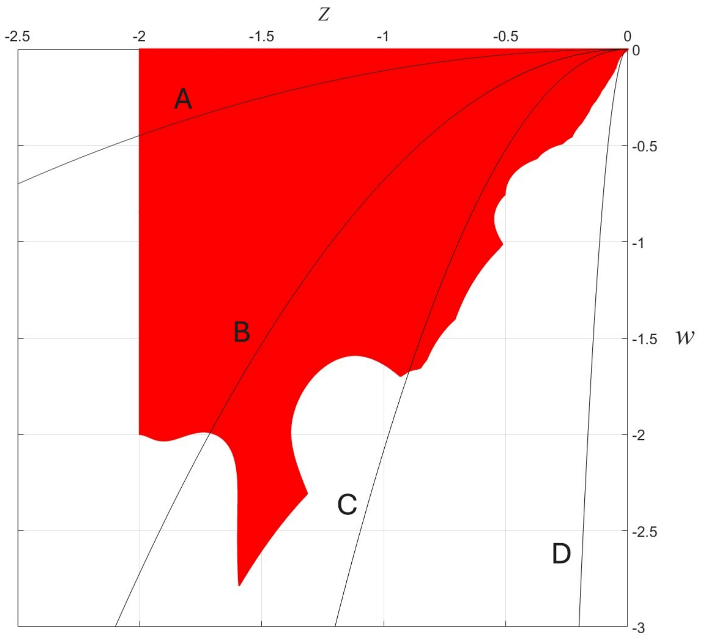

Fig. 1. Stability region (red) for the explicit algorithm. Various h-paths are also shown.

3.2 Explicit case

Applying the explicit algorithm to the test equation yields

$$\begin{aligned} P_0 &= 2 \\ P_1(z,w) &= z+2 \\ P_2(z,w) &= z^2 + \frac{zw}{2} + 2z + 2w + 2 \\ P_i(z,w) &\equiv 2\left(\frac{(1+z+\frac{w}{2})P_{i-1} - z - w + w\sum_{j=0}^{i-2} P_j}{2}\right), \quad i \geqslant 3. \end{aligned}$$

We choose a grid of 10^7 points (z,w) in the third quadrant and we compute $|P_i(z,w)|$ for each of these points, for $i \in [1,10^6]$. We determine, for each (z,w), whether or not $|P_i(z,w)| > 2$ for any $i \in [1,10^6]$. If not, we consider the point (z,w) to be stable. This is simply a brute-force grid search. We are able to construct a region R_E of stability for the explicit method, and we show this region in Fig. 1. Also shown are curves, which we call *h-paths*, for various values of (λ,γ). The h-path for (λ,γ) is the locus of points on the diagram for $(h\lambda,h^2\gamma)$, where $h \in [0,\infty]$. The h-path for (λ,γ) always passes through (λ,γ) – corresponding to $h=1$ – and always terminates at the origin. It is clear that three of the h-paths shown (A, B and C) intersect R_E quite obviously; one of them (D), however, intersects R_E only near the origin. These h-paths indicate the effect of reducing h on the location of the point $(h\lambda,h^2\gamma)$. Recall, for stability, it is necessary to choose h such that $(h\lambda,h^2\gamma)$ lies within the stability region, and the h-path gives a clear idea of the range of values of h required to achieve that. For the h-path D that intersects R_E only near the origin, a very small value of h will be needed, particularly if λ and/or γ are strongly negative. For the other three h-paths, relatively larger values for h could be tolerated to achieve stability.

4 Error control

We assume the algorithms have a first-order error. Consequently, we can use Richardson extrapolation to achieve solutions of higher-order. We will not discuss this here, and the reader is referred to our previous work for the necessary detail [16, 17]. In [16] we estimate the number of nodes needed for various tolerances when using a third-order solution. In this paper, we make similar estimates (in the next section) with reference to a fourth-order solution. In the notation of [16], we use $Y_i^4 - Y_i^5$ instead of $Y_i^3 - Y_i^5$ for error control. Again, the reader is referred to these previous papers for relevant detail.

5 Examples

We consider the same examples as used in [8], modified to suit our test equation. The parameters λ and γ are easily identified. The quantities N_1 and N_2 in Table 1 refer to the number of nodes (N in (2)) needed to achieve tolerances of $\varepsilon = 10^{-6}$ and $\varepsilon = 10^{-12}$, respectively, using the Richardson process described above. The stepsize can be found from $h = 10/(N-1)$. The implicit algorithm yielded stable solutions for all the examples.

Results for the explicit algorithm are shown in Table 2. Here, N indicates the least number of nodes required for a stable solution, h_s is the corresponding stepsize and $(z, w) = (h_s\lambda, h_s^2\gamma)$.

Table 1. Examples 1–3, implicit algorithm.

#	IDE	x	N_1	N_2
1	$y' = -100(y-1) - 0.1\int_0^x y dt$	$[0, 10]$	1158	36606
2	$y' = -14(y-1) - 15\int_0^x y dt$	$[0, 10]$	207	6519
3	$y' = -0.1(y-1) - 650\int_0^x y dt$	$[0, 10]$	10044	317613

Table 2. Explicit algorithm.

#	N	h_s	(z, w)
1	505	0.0198	$(-1.98, 4 \times 10^{-5})$
2	72	0.1408	$(1.97, 0.3)$
3	9501	0.0011	$(-1.1 \times 10^{-5}, 7.8 \times 10^{-4})$

Note the very small stepsize needed for #3. This case corresponds to the h-path D in Fig. 1 that is close to the vertical axis, and intersects the stability region only near the origin.

6 Conclusion

We have extended earlier work on the numerical solution of Volterra integro-differential equations by providing implicit and explicit versions of an Euler-type algorithm for a kernel of general form. We have conducted a stability analysis. We have found that the implicit method is unconditionally stable stable in the third quadrant of the stability space, while the explicit method is stable over a relatively small region near the origin. Numerical examples with a suitable test equation support these findings.

7 Disclosure statement

The authors do not work for, consult, own shares in or receive funding from any company or organization that would benefit from this article, and have disclosed no relevant affiliations beyond their academic appointment.

Appendix

Justification for the test equation

If $\pi(x)$ is a function that has $\pi(x_0)=y(x_0)$ then, with $D(x)\equiv y(x)-\pi(x)$, we may write

$$\begin{aligned} f(x,y) &= f\big(x_0+\delta_x, D(x)+\pi(x)\big) \\ &= f\big(x_0,\pi(x)\big)+f_y\big(x_0,\pi(x)\big)y+R_1(x) \\ &= f_y\big(x_0,\pi(x_0)\big)y+R_2(x) \end{aligned}$$

for suitable δ_x and $R_2(x)$. Also, we may write

$$K(x,y,t)=K_y\big(x_0,\pi(x_0),x_0\big)y+R_3(x,y,t)$$

for suitable $R_3(x,y,t)$. These give

$$\begin{aligned} y' &= f_y\big(x_0,\pi(x_0)\big)y+R_2(x)+\int\limits_{x_0}^{x}\big(K_y(x_0,\pi(x_0),x_0)y+R_3(x,y,t)\big)dt \\ &= f_y\big(x_0,\pi(x_0)\big)y+K_y\big(x_0,\pi(x_0),x_0\big)\int\limits_{x_0}^{x}ydt+\Bigg(\int\limits_{x_0}^{x}R_3(x,y,t)dt+R_2(x)\Bigg) \\ &\equiv \lambda(y-\alpha)+\gamma\int\limits_{x_0}^{x}ydt+G(x), \end{aligned}$$

where $G(x)\equiv\int_{x_0}^{x}R_3(x,y,t)dt+R_2(x)+\alpha\lambda$, and λ and γ have been implicitly defined. Hence,

$$y''-\lambda y'-\gamma y=G'(x).$$

This equation has the general solution

$$y(x)=c_1e^{m_1x}+c_2e^{m_2x}+y_p(x),$$

where $y_p(x)$ is a particular solution. We see that the complementary solution is explicitly present, and it is the qualitative behaviour of this solution in the third quadrant that is relevant to our study of stability in this paper. Of course, the complementary solution is the solution to

$$y' = \lambda(y-\alpha) + \gamma \int\limits_{x_0}^{x} y dt$$

and so this equation appears to be a useful candidate for a test equation. We have used $x_0 = 0$ in the test equation earlier; this is merely a translation along the x-axis and does not affect the outcome of the analysis. Also, we have found it convenient to impose the initial value $y(0) = 2$ and to demand $c_1 = c_2 = 1$, which is achieved by setting $\alpha = 1$. The parameter α does not affect our analysis – it simply leads to a term $-\alpha z$ that is absorbed into the definition of T_i, and it does not influence the eigenvalues of the transfer matrix M.

Systems

When $n = 2$ in (1), we have the system

$$\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} y_2 \\ f(x,y_1) + \int\limits_{x_0}^{x} K(x,y_1(t),t)dt \end{bmatrix}$$

and when $n = 3$, we have

$$\begin{bmatrix} y_1' \\ y_2' \\ y_3' \end{bmatrix} = \begin{bmatrix} y_2 \\ y_3 \\ f(x,y_1) + \int\limits_{x_0}^{x} K(x,y_1(t),t)dt \end{bmatrix},$$

The initial values $y_1(x_0)$ and $y_2(x_0) = y_1'(x_0)$ must be specified for the first system, and $y_1(x_0), y_2(x_0) = y_1'(x_0)$ and $y_3(x_0) = y_2''(x_0)$ must be specified for the second system. Of course, analogous expressions exist for $n > 3$.

The general first-order system has the form

$$\begin{bmatrix} y_1' \\ \vdots \\ y_m' \end{bmatrix} = \begin{bmatrix} f_1(x,y_1,\dots,y_m) + \int\limits_{x_0}^{x} K_1(x,y_1,\dots,y_m)dt \\ \vdots \\ f_m(x,y_1,\dots,y_m) + \int\limits_{x_0}^{x} K_m(x,y_1,\dots,y_m)dt \end{bmatrix}$$

$$\Rightarrow \quad \begin{bmatrix} y_{1,i+1} \\ \vdots \\ y_{m,i+1} \end{bmatrix} = \begin{bmatrix} y_{1,i} + hG_1 \\ \vdots \\ y_{m,i} + hG_m \end{bmatrix},$$

where the forms of G_1 and G_m can be inferred. For this system, we can find the Jacobians

$$J_f \equiv \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \cdots & \frac{\partial f_1}{\partial y_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial y_1} & \cdots & \frac{\partial f_m}{\partial y_m} \end{bmatrix} \quad \text{and} \quad J_K \equiv \begin{bmatrix} \frac{\partial K_1}{\partial y_1} & \cdots & \frac{\partial f K_1}{\partial y_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial K_m}{\partial y_1} & \cdots & \frac{\partial K_m}{\partial y_m} \end{bmatrix}.$$

Let $\{\lambda_1, \dots, \lambda_m\}$ denote the eigenvalues of J_f, and $\{\gamma_1, \dots, \gamma_m\}$ denote the eigenvalues of J_K. We form the pairs (λ_j, γ_j). If $\lambda_j \leqslant 0$ and $\gamma_j \leqslant 0$, or $\text{Re}(\lambda_j) \leqslant 0$ and $\text{Re}(\gamma_j) \leqslant 0$ if the eigenvalues are complex, we write

$$(z, w) = (h\lambda_j, h^2\gamma_j) = \Big(- h\sqrt{\lambda_j^* \lambda_j}, -h^2 \sqrt{\gamma_j^* \gamma_j} \Big).$$

Newton's Method

To solve (3) for y_{i+1} we can employ Newton's Method in the form

$$y_{i+1}^{k+1} = y_{i+1}^k - \frac{F(y_{i+1}^k)}{F'(y_{i+1}^k)},$$

where

$$F(y_{i+1}) \equiv y_{i+1} - y_i - hf(x_{i+1}, y_{i+1}) - \frac{h^2}{2} \Bigg(\sum_{j=0}^{i+1} 2K(x_{i+1}, y_j, x_j) - K(x_{i+1}, y_0, x_0) - K(x_{i+1}, y_{i+1}, x_{i+1}) \Bigg)$$

and

$$F'(y_{i+1}) = \frac{dF(y_{i+1})}{dy_{i+1}} = 1 - h\frac{df(x_{i+1}, y_{i+1})}{dy_{i+1}} - \frac{h^2}{2} \frac{dK(x_{i+1}, y_{i+1}, x_{i+1})}{dy_{i+1}},$$

with initial value $y_{i+1}^0 = y_i$. We would generally expect second-order convergence for this method.

However, if we use

$$y_{i+1}^{k+1} = y_{i+1}^k - \frac{F(y_{i+1}^k)}{F'(y_{i+1}^k)} - \frac{F^2(y_{i+1}^k)F''(y_{i+1}^k)}{2(F'(y_{i+1}^k))^3}$$

where

$$F''(y_{i+1}) = \frac{d}{dy_{i+1}} \frac{dF(y_{i+1})}{dy_{i+1}},$$

we may expect third-order convergence, which could improve the efficiency of the algorithm. Indeed, this is the approach we used in our calculations, subject to the convergence condition

$$\left| \frac{F(y_{i+1}^k)}{F'(y_{i+1}^k)} + \frac{F^2(y_{i+1}^k)F''(y_{i+1}^k)}{2(F'(y_{i+1}^k))^3} \right| < 10^{-14}.$$

Richardson Extrapolation

The Eulerian character of our algorithm, together with the use of the Trapezium Rule, ensures that we cannot expect an error better than first-order. However, this is quite acceptable, since we can deploy Richardson extrapolation to achieve higher-order approximations from first-order results. We have provided detail regarding Richardson extrapolation elsewhere [17], and we simply state here the process we use to construct solutions of order as high as five.

Let $y_i(h)$ denote the solution obtained at x_i using a stepsize h (i.e. the nodes in (2)). Let $y_i(h/2)$ denote the solution obtained at x_i using a stepsize $h/2$. Such a computation uses the equispaced nodes

$$x_0 < x_{1/2} < x_1 < x_{3/2} < x_2 < \cdots < x_{N-1/2} < x_N$$

where each intermediate node $x_{i-1/2}$ is located midway between x_{i-1} and x_i. We can similarly obtain the solutions $y_i(h/4), y_i(h/8)$ and $y_i(h/16)$, using appropriate node distributions. Now, we form the linear combinations

$$\begin{aligned}
Y_i^2 &= -y_i(h) + 2y_i(h/2),\\
Y_i^3 &= \frac{y_i(h)}{3} - 2y_i(h/2) + \frac{8y_i(h/4)}{3},\\
Y_i^4 &= -\frac{y_i(h)}{21} + \frac{2y_i(h/2)}{3} - \frac{8y_i(h/4)}{3} + \frac{64y_i(h/8)}{21},\\
Y_i^5 &= \frac{y_i(h)}{315} - \frac{2y_i(h/2)}{21} + \frac{8y_i(h/4)}{9} - \frac{64y_i(h/8)}{21} + \frac{1024y_i(h/16)}{315},
\end{aligned}$$

which yield 2nd-, 3rd-, 4th- and 5th-order solutions, respectively, at x_i. We are interested in the 4th-order solution. If we assume the 4th- and 5th-order solutions have error terms at x_i of the form

$$\begin{aligned}
&K_4^i h^4 + \ldots\\
&K_5^i h^5 + \ldots,
\end{aligned}$$

respectively, then

$$\begin{aligned}
Y_i^4 - Y_i^5 &= K_4^i h^4 + \cdots - \left(K_5^i h^5 + \cdots\right)\\
&\approx K_4^i h^4
\end{aligned}$$

for suitably small h. Since Y_i^4 and Y_i^5 are known, we have

$$K_4^i = \frac{Y_i^4 - Y_i^5}{h^4}$$

as a good estimate for the error coefficient K_4^i. Consequently, a suitable stepsize for a desired accuracy ε is found from

$$h_i = \sigma\left(\frac{\varepsilon}{|K_4^i|}\right)^{1/3}$$

where the *safety factor* σ is $\sigma \sim 0.85$. Naturally, such a value for the stepsize is computed at each x_i, and the smallest such value is the one chosen, denoted $\underset{\sim}{h}$. This chosen value is then used to rerun the algorithm, with the resulting output satisfying the specified tolerance ε. If we wish to control relative error, we compute

$$h_i = \sigma \left(\frac{\varepsilon \max\{1, |y_i|\}}{|K_4|} \right)^{1/3}$$

at each x_i and, as before, take the smallest such value and rerun the algorithm.

The number of nodes N is determined from

$$N = \frac{x_N - x_0}{\widetilde{h}} + 1$$

where $x_N - x_0$ is the lengtth of the interval of integration. In our examples we used $x_N - x_0 = 10$.

References

[1] A.F. AL-Shimmary, A.K. Hussain, S.K. Radhi. Numerical solution of Volterra integro-differential equation using 6th order Runge-Kutta method. *J. Phys.: Conf. Ser.*, **1818**:3219–3225, 2021. https://doi.org/10.1088/1742-6596/1818/1/012183.

[2] A. Arikoglu, I. Ozkol. Solutions of integral and integro-differential equation systems by using differential transform method. *Comput. Math. Appl.*, **56**:2411–2417, 2008. https://doi.org/10.1016/j.camwa.2008.05.017.

[3] A. Avudainayagam, C. Vani. Wavelet Galerkin method for integro-differential equations. *Appl. Math. Comput.*, **32**:247–254, 2000. https://doi.org/10.1016/S0168-9274(99)00026-4.

[4] D. Bahuguna, A. Ujlayan, D.N. Pandey. A comparative study of numerical methods for solving an integro-differential equation. *Comput. Math. Appl.*, **57**:1485–1493, 2009. https://doi.org/10.1016/j.camwa.2008.10.097.

[5] J. Biazar, E. Babolian, R. Islam. Solution of a system of Volterra integral equations of the first kind by Adomian method. *Appl. Math. Comput.*, **139**:249–258, 2003. https://doi.org/10.1016/S0096-3003(02)00173-X.

[6] H. Brunner, E. Hairer, S. P. Norsett. Runge-Kutta theory for Volterra integral equations of the second kind. *Math. Comput.*, **39**:147–163, 1982. https://doi.org/10.2307/2007625.

[7] H. Brunner, P.J. Houwen. *The Numerical Solution of Volterra Equations.* North-Holland, New York, 1986. ISBN 9780444700735.

[8] M.R. Crisci, E. Russo, A. Vecchio. Stability analysis of the de Hoog and Weiss implicit Runge-Kutta methods for the Volterra integral and integrodifferential equations. *J. Comput. Appl. Math.*, **29**:329–341, 1990. https://doi.org/10.1016/0377-0427(90)90015-R.

[9] L.M. Delves, J.L. Mohamed. *Computational Methods for Integral Equations.* Cambridge University Press, Cambridge, 1985. ISBN 9780511569609.

[10] H. Sadeghi Goghary, Sh. Javadi, E. Babolian. Restarted Adomian method for system of nonlinear Volterra integral equations. *Appl. Math. Comput.*, **161**:745–751, 2005. https://doi.org/10.1016/j.amc.2003.12.072.

[11] A.A. Hamoud, N.M. Mohammed, K.P. Ghadle, S.L. Dhondge. Solving integro-differential equations by using numerical techniques. *Int. J. Appl. Eng. Res.*, **14**:3219–3225, 2019.

[12] J. He. Homotopy perturbation technique. *Comput. Method Appl. Math.*, **178**:257–262, 1999. https://doi.org/10.1016/S0045-7825(99)00018-3.

[13] K. Maleknejad, M. Tavassoli Kajani. Solving linear integro-differential equation system by Galerkin methods with hybrid functions. *Appl. Math. Comput.*, **159**:603–612, 2004. https://doi.org/10.1016/j.amc.2003.10.046.

[14] K. Maleknejad, F. Mirzaee, S. Abbasbandy. Solving linear integro-differential equations system by using rationalized Haar functions method. *Appl. Math. Comput.*, **155**:317–328, 2004. https://doi.org/10.1016/S0096-3003(03)00778-1.

[15] J. Manafianheris. Solving the integro-differential equations using the modified Laplace Adomian decomposition method. *J. Math. Ext.*, **6**:41–55, 2012. https://www.ijmex.com/index.php/ijmex/article/view/96.

[16] J.S.C. Prentice. An Euler-type method for Volterra integro-differential equations. arXiv.org (Cornell University Library), 2023. https://doi.org/10.48550/arXiv.2306.02547.

[17] J.S.C. Prentice. Evaluating a double integral using Euler's method and Richardson extrapolation. *Liet. matem. rink. Proc. of LMS, Ser. A*, pp. 39–52, 2024. https://doi.org/10.15388/LMD.2024.38091.

[18] M. Taghipour, H. Aminikhah. Pell collocation method for solving the nonlinear time–fractional partial integro-differential equation with a weakly singular kernel. *J. Funct. Spaces*, 2022. https://doi.org/10.1155/2022/8063888.

[19] Siraj ul Islam, I. Aziz, A.S. Al-Fhaid. An improved method based on Haar wavelets for numerical solution of nonlinear integral and integro-differential equations of first and higher orders. *J. Comput. Appl. Math.*, **260**:449–469, 2014. https://doi.org/10.1016/j.cam.2013.10.024.

[20] E. Yusufoglu. An efficient algorithm for solving integro-differential equations system. *Appl. Math. Comput.*, **192**:51–55, 2007. https://doi.org/10.1016/j.amc.2007.02.134.

REZIUMĖ

Netiesioginio ir eksplicitinio skaitmeninio Volteros integralinių diferencialinių lygčių su branduoliu $K(x, y(t), t)$ stabilumo analizė

J.S.C. Prentice

Pateikiame netiesioginę ir tiesioginę skaitmeninio algoritmo versijas Volteros integro-diferencialinei lygčiai spręsti. Šie algoritmai yra mūsų ankstesnio darbo išplėtimas ir pritaikytas bendros formos branduoliui. Abiejų algoritmų stabilumui ištirti naudojame tinkamą bandymo lygtį. Įrodome, kad netiesioginis metodas yra besąlygiškai stabilus trečiajame kvadrante. Skaitmeniškai nustatome tiesioginio metodo stabilumo sritį trečiajame kvadrante. Tiesioginis metodas turi apribotą sritį, artimą koordinačių pradžiai. Atliekame keletą skaičiavimų, kad pademonstruotume savo rezultatus.

Raktiniai žodžiai: Volterra; stabilumas; integro-diferencialas; skaitmeninis