Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 878497, 11 pages
doi:10.1155/2012/878497

Research Article

On the Computation of Blow-Up Solutions for Nonlinear Volterra Integrodifferential Equations

P. G. Dlamini and M. Khumalo

Department of Mathematics, University of Johannesburg, Cnr Siemert & Beit Streets, Doornfontein 2028, South Africa

Correspondence should be addressed to M. Khumalo, mkhumalo@uj.ac.za

Received 7 February 2012; Revised 19 March 2012; Accepted 20 March 2012

Academic Editor: Kuppalapalle Vajravelu

We make use of an adaptive numerical method to compute blow-up solutions for nonlinear ordinary Volterra integrodifferential equations (VIDEs). The method is based on the implicit midpoint method and the implicit Euler method and is named the implicit midpoint-implicit Euler (IMIE) method and was used to compute blow-up solutions in semilinear ODEs and parabolic PDEs in our earlier work. We demonstrate that the method produces superior results to the adaptive PECE-implicit Euler (PECE-IE) method and the MATLAB solver of comparable order just as it did in our previous contribution. We use quadrature rules to approximate the integral in the VIDE and demonstrate that the choice of quadrature rule has a significant effect on the blow-up time computed. In cases where the problem contains a convolution kernel with a singularity we use convolution quadrature.

1. Introduction

Many solutions of differential equations modeling physical problems blow-up in finite time. the phenomenon of blow-up is said to have occurred at $T_b < \infty$ if the solution of the differential equation becomes unbounded at $t = T_b$. The blow-up time often represents an important change in the properties of the model and it is important that it should be accurately reproduced by a numerical computation.

Much work in the study of blow-up has been on reaction-diffusion equations. The reaction-diffusion equation is a semilinear parabolic equation of the form

$$\begin{aligned} u_t(t,x) - \Delta u(t,x) &= f(t,x,u), \quad t > 0,\ x \in \Omega \subset \mathbb{R}^d, \\ u(0,x) &= u_0(x) \geq 0, \quad x \in \Omega, \\ u(t,x) &= 0, \quad t > 0,\ x \in \partial\Omega, \end{aligned} \tag{1.1}$$

where $\Delta u(t,x)$ is referred to as the diffusion term and $f(t,x,u)$ as the reaction term. It is well known from blow-up theories that for sufficiently large initial function $u_0(x)$ the solution of (1.1) will blow-up in finite time.

Most methods for solving evolution equations lose accuracy as the solution becomes large (approaches blow-up) and hence adaptive methods are used. Bandle and Brunner [1] present a survey of the theory and the numerical analysis of blow-up solutions for quasilinear reaction-diffusion equations of the form (1.1). Budd et al. [2] proposed the use of moving mesh partial differential equation methods (MMPDE) for solving (1.1). A more recent study of the MMPDE is presented in [3] and the references therein.

Berger and Kohn [4] used a mesh refinement strategy which is able compute the solution accurately over the entire physical interval even as the solution grows in magnitude. Stuart and Floater [5] showed that fixed step methods, both explicit and implicit fail to reproduce blow-up time for a scalar ODE. They also examined variable step methods. They used time stepping strategies which are based on a rescaling of the time variable in the underlying differential equation. They also apply these ideas to a PDE.

In some applications, for example, in population dynamics, the reaction term is nonlocal [6]. For example, $f(t,x,u) = \int_0^t k(t-s)u^p(s,x)ds$ so that (1.1) becomes

$$u_t(t,x) - u_{xx}(t,x) = \int_0^t k(t-s)u^p(s,x)ds, \quad t > 0,\ x \in (0,1). \tag{1.2}$$

In this paper, we consider the Volterra integrodifferential equation (VIDE),

$$y'(t) = \lambda y(t) + \int_0^t k(t-s)g\big(y(s)\big)ds, \quad t > 0,\ y(0) = y_0 > 0 \tag{1.3}$$

with $\lambda < 0$ and $k(t-s)$ is the kernel.

Recently Ma [7] has studied the finite-time blow-up theory and blow-up conditions for nonlinear ordinary VIDEs of the form (1.3) and nonlinear partial VIDEs. Assuming

(a) $k(t)$ is an integrable positive function such that $\lim_{t\to\infty} K(t) = \infty$, where $K(t) = \int_0^t k(t)ds$;

(b) $g(t)$ is nonnegative, nondecreasing and continuous for $t > 0$, $g \equiv 0$ for $t \le 0$, and $\lim_{y\to\infty}(g(y)/y) = \infty$,

he proves that the solution of the VIDE (1.3) blows up if and only if

$$\int_\alpha^\infty \left[\frac{s}{g(s)}\right]^{1/\beta} < \infty, \tag{1.4}$$

for any $\alpha > 0$, where $\beta > 0$.

The earliest author to look at blow-up solutions of parabolic equations with nonlinear memory is Bellout [8]. The theory of blow-up for nonlocal reaction-diffusion problems is given by authors such as [9, 10]. Ma et al. [11] implement the moving mesh methods to solve reaction-diffusion equations with nonlocal nonlinear terms such as (1.2) which blows-up in finite time.

2. Description of Methods Used

We use quadrature rules to approximate the integral in (1.3) and solve the resulting system of ODEs of the form:

$$y' = f(t, y), \quad t \geq 0. \tag{2.1}$$

We use the adaptive PECE-implicit Euler (PECE-IE) and adaptive implicit midpoint-implicit Euler (IMIE) methods introduced in Dlamini and Khumalo [12] to solve the resulting system of ODEs and compare the results with matlab ODE solvers ode45 and ode15s. As the blow-up time is approached changes occur on an increasingly smaller time scales and so to be more accurate in obtaining the blow-up time, variable stepsize methods are used. We select a mesh

$$\{t_i : 0 = t_0 < t_1 < t_2 < \cdots\}, \quad i = 0, 1, 2, 3, \ldots \tag{2.2}$$

with $t_i - t_{i-1} = h_i$.

In our procedures we specify the acceptable error per step and if it is not met the procedure adjusts the step size so that each step introduces an error that is not more than the acceptable error. At each step we solve the problem using two different algorithms, giving two different solutions, say $S1$ and $S2$. We approximate the local truncation error by computing the difference between the two solutions, that is, $|S1 - S2|$. If the error is smaller than the specified acceptable error, we move to the next step and accept $S2$ as the solution for the current step. Otherwise, we set a new stepsize and redo the current step.

2.1. Adaptive PECE-Implicit Euler Method (PECE-IE)

This method is based on the implicit Euler method. We compute $S1$ using a predictor-corrector method in which y_p is obtained using explicit Euler's method so that we have

$$S1 = y_{i+1} = y_i + h_i f(t_{i+1}, y_p). \tag{2.3}$$

To compute $S2$ we use Newton's method as the solver to deal with the implicit nature of the implicit Euler method.

2.2. Adaptive Implicit Midpoint-Implicit Euler Method (IMIE)

We use the implicit Euler method with Newton's method as the solver to get $S1$. To compute $S2$ we use implicit midpoint method given by

$$y_{i+1} = y_i + h_i f\left(t_i + \frac{h_i}{2}, \frac{y_i + y_{i+1}}{2}\right). \tag{2.4}$$

First we compute y_{i+1} using the implicit Euler method and substitute in (2.4) to get $S2$ that is,

$$y_p = y_i + h_i f\left(t_{i+1}, y_{i+1}\right), \tag{2.5}$$

then,

$$S2 = y_{i+1} = y_i + h_i f\left(t_i + \frac{h_i}{2}, \frac{y_i + y_p}{2}\right). \tag{2.6}$$

2.3. Matlab Solvers (ode45 and ode15s)

ode45 is a one-step Matlab solver that is based on an explicit Runge-Kutta (4, 5) scheme. It varies the size of the step of the independent variable in order to meet the accuracy specified. On the other hand, ode15s is a multistep Matlab variable order solver based on implicit methods. We use these solvers to compare the PECE-IE and IMIE methods.

3. Numerical Computation

We consider

$$y'(t) = \lambda y(t) + \int_0^t k(t-s)y^p(s)ds, \quad t > 0,\ y(0) = y_0 > 0 \tag{3.1}$$

with $\lambda < 0$ and $k(t-s)$ is the kernel.

We solve (3.1) by first approximating the integral using a quadrature rule and then solve the resulting system of ODEs using the methods mentioned above. We use the repeated trapezoidal rule and block-by-block-method (see [13, 14] and the references therein). The effect of the quadrature rule in the computation of the blow-up time will be investigated. In cases where the convolution kernel has a singularity we use convolution quadrature discussed in Lubich [15–17].

3.1. Using Repeated Trapezoidal Rule

We let

$$\begin{aligned} t_0 &= 0, \\ t_{i+1} &= t_i + h_i, \quad i = 0, 1, 2, 3, \ldots, n, \end{aligned} \tag{3.2}$$

$$y'(t_i) = \lambda y(t_i) + \sum_{j=1}^{i} \int_{t_{j-1}}^{t_j} k(t_i - s)y^p(s)ds, \quad i = 1, 2, 3, \ldots, n. \tag{3.3}$$

The approximation of every integral in (3.3) by repeated trapezoidal rule will yield the following system (with $y_i = y(t_i)$ and $k_{i,j} = k(t_i, t_j)$)

$$\begin{aligned}
y_1' &= \lambda y_1 + \frac{h_1}{2}\left[k_{1,0}y_0^p + k_{1,1}y_1^p\right],\\
y_2' &= \lambda y_2 + \frac{h_2}{2}\left[k_{2,0}y_0^p + k_{2,1}y_1^p + k_{2,2}y_2^p\right],\\
y_3' &= \lambda y_3 + \frac{h_3}{2}\left[k_{3,0}y_0^p + k_{3,1}y_1^p + k_{3,2}y_2^p + k_{3,3}y_3^p\right],\\
&\vdots\\
y_n' &= \lambda y_n + \frac{h_n}{2}\left[k_{n,0}y_0^p + k_{n,1}y_1^p + \cdots + k_{n,n-1}y_{n-1}^p + k_{n,n}y_n^p\right].
\end{aligned} \tag{3.4}$$

3.2. Using Block-by-Block Method

In this method we use Simpson's rule and a quadratic interpolation to approximate the integral in (3.1). We rewrite (3.1) in the form

$$y'(t_i) = \lambda y(t_i) + \int_0^{t_i} k(t_i - s)y^p(s)ds, \quad t_i > 0,\ y(0) = y_0 > 0, \tag{3.5}$$

$$\begin{aligned}
t_0 &= 0,\\
t_{i+1} &= t_i + h_i, \quad i = 0, 1, 2, 3, \ldots, n.
\end{aligned} \tag{3.6}$$

The approximation of (3.5) using the block-by-block method yield the following system:

$$\begin{aligned}
y_1' &= \lambda y_1 + \frac{5h_1}{12}k_{1,0}y_0^p + \frac{2h_1}{3}k_{1,1}y_1^p - \frac{h_1}{12}k_{1,2}y_2^p,\\
y_2' &= \lambda y_2 + \frac{h_2}{3}\left[k_{2,0}y_0^p + 4k_{2,1}y_1^p + k_{2,2}y_2^p\right],\\
&\vdots\\
y_{2n+1}' &= \lambda y_{2n+1} + \frac{h_{2n+1}}{3}\left[k_{2n+1,0}y_0^p + 4k_{2n+1,1}y_1^p + \cdots + k_{2n+1,2n}y_{2n}^p\right]\\
&\quad + \frac{5h_{2n+1}}{12}k_{2n+1,2n}y_{2n}^p + \frac{2h_{2n+1}}{3}k_{2n+1,2n+1}y_{2n+1}^p - \frac{h_{2n+1}}{12}k_{2n+1,2n+2}y_{2n+2}^p,\\
y_{2n+2}' &= \lambda y_{2n+2} + \frac{h_{2n+2}}{3}\left[k_{2n+2,0}y_0^p + 4k_{2n+2,1}y_1^p + \cdots + 4k_{2n+2,2n+1}y_{2n+1}^p + k_{2n+2,2n+2}y_{2n+2}^p\right].
\end{aligned} \tag{3.7}$$

3.3. Using Convolution Quadrature

In problems where the convolution kernel has a singularity, closed Newton's cotes formulas fail in the computation of the solution because of the singularity. To overcome this problem we use convolution quadrature. An introduction to convolution quadrature is found in Lubich [15, 16] and he also gives a review of all aspects of convolution quadrature in [17].

We rewrite (3.1) in the form

$$y'(t_i) = \lambda y(t_i) + \int_0^{t_i} k(t_i - s) y^p(s) ds, \quad t_i > 0,\ y(0) = y_0 > 0, \tag{3.8}$$

$$\begin{aligned} t_0 &= 0, \\ t_{i+1} &= t_i + h_i, \quad i = 0, 1, 2, 3, \ldots, n. \end{aligned} \tag{3.9}$$

Here the integral

$$\int_0^{t_i} k(t_i - s) y^p(s) ds \tag{3.10}$$

in (3.8) is approximated by convolution quadrature with a step size $h > 0$,

$$\sum_{j=0}^{i} w_j y^p(t_j), \quad i = 1, 2, 3, \ldots, n, \tag{3.11}$$

where the convolution quadrature weights w_j are the coefficients of the generating power series

$$\sum_{j=0}^{\infty} w_j \xi^j = F\left(\frac{\delta(\xi)}{h}\right). \tag{3.12}$$

Here, $F(s)$ is the Laplace transform of the convolution kernel $k(t)$ and $\delta(\xi)$ is the quotient of the generating polynomials of a linear multistep method. We will use a convolution quadrature formula based on the order 2 BDF method, that is, we use $\delta(\xi) = (1 - \xi) + (1/2)(1 - \xi)^2$, (see Lubich [17]). The approximation of (3.8) by the convolution quadrature method yields the following system:

$$\begin{aligned} y_1' &= \lambda y_1 + w_0 y_0^p + w_1 y_1^p, \\ y_2' &= \lambda y_2 + w_0 y_0^p + w_1 y_1^p + w_2 y_2^p, \\ y_3' &= \lambda y_3 + w_0 y_0^p + w_1 y_1^p + w_2 y_2^p + w_3 y_3^p, \\ &\vdots \\ y_n' &= \lambda y_n + w_0 y_0^p + w_1 y_1^p + \cdots + w_{n-1} y_{n-1}^p + w_n y_n^p. \end{aligned} \tag{3.13}$$

4. Numerical Examples

In this section we compute examples of (3.1).

Table 1: Blow-up time for (4.1) by repeated trapezoidal rule.

p	PECE-IE	IMIE	ode45	ode15s
2	2.058257	2.064995	2.065002	2.064995
2.5	1.253812	1.259185	1.259187	1.259179
3	0.764148	0.768275	0.768277	0.768271

Table 2: Blow-up time for (4.1) by block-by-block method.

p	PECE-IE	IMIE	ode45	ode15s
2	1.893982	1.899968	1.899970	1.899967
2.5	1.216445	1.221289	1.221291	1.221284
3	0.663921	0.667393	0.667395	0.667393

Example 4.1.

$$y'(t) = \lambda y(t) + \int_0^t y^p(s)ds, \quad t > 0,\ y(0) = y_0 > 0, \tag{4.1}$$

with $\lambda = -1$, $y(0) = y_0 = 3$ and $p = 2, 2.5, 3$.

Tables 1 and 2 show the blow-up results by the repeated trapezoidal rule and the block-by-block method, respectively, and Figure 1 shows the solution of (4.1) computed by the repeated trapezoidal rule.

Example 4.2.

$$y'(t) = \lambda y(t) + \int_0^t e^{s-t} y^p(s)ds, \quad t > 0,\ y(0) = y_0 > 0, \tag{4.2}$$

with $\lambda = -1$, $y(0) = y_0 = 3$ and $p = 2, 2.5, 3$.

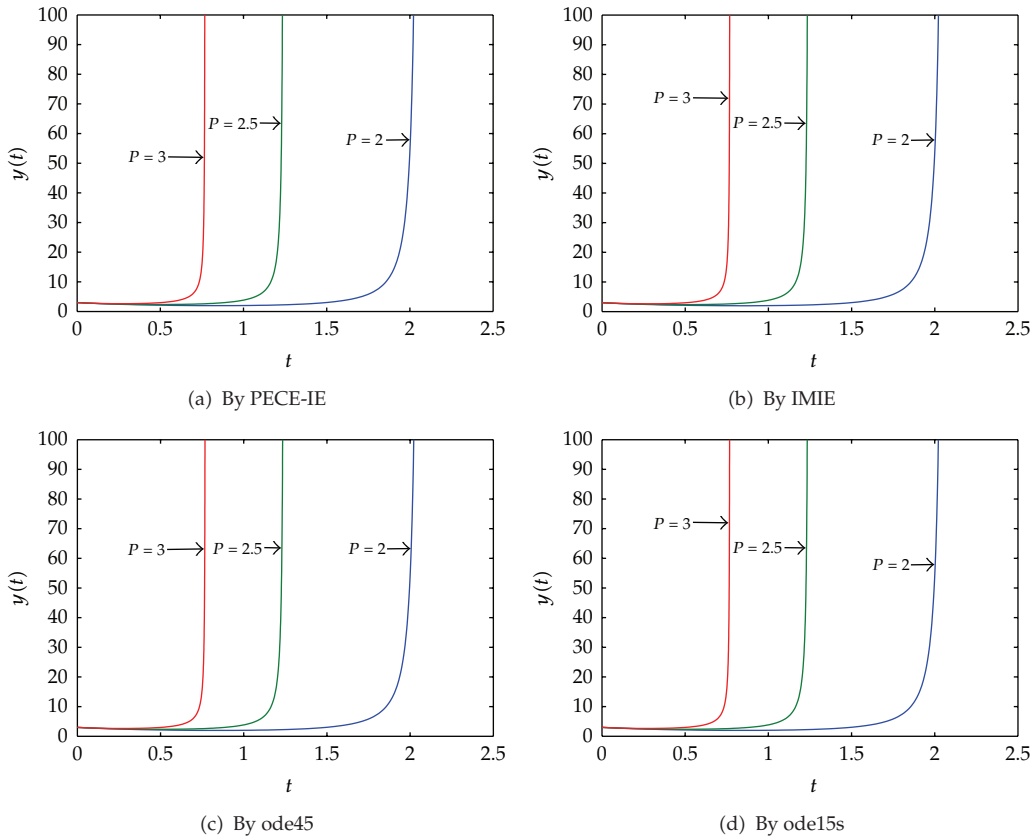
Tables 3 and 4 show the blow-up results by repeated trapezoidal rule and the block-by-block method, respectively, and Figure 2 shows the solution of (4.2) computed by trapezoidal rule.

Example 4.3.

$$y'(t) = \lambda y + \int_0^t (t-s)^{-\alpha} y^p(s)ds, \quad t > 0,\ y(0) = y_0 > 0, \tag{4.3}$$

with $\lambda = -1, \alpha = 2/3, y(0) = y_0 = 1$ and $p = 2, 2.5, 3$.

We use same parameters as in Ma [7] and compare the results.



(a) By PECE-IE (b) By IMIE

(c) By ode45 (d) By ode15s

Figure 1: Numerical solution of (4.1) by repeated trapezoidal rule.

Table 3: Blow-up time for (4.2) by repeated trapezoidal rule.

p	PECE-IE	IMIE	ode45	ode15s
2	3.141813	3.157533	3.157539	3.157510
2.5	1.583900	1.592792	1.592800	1.592728
3	0.878594	0.884238	0.884239	0.884229

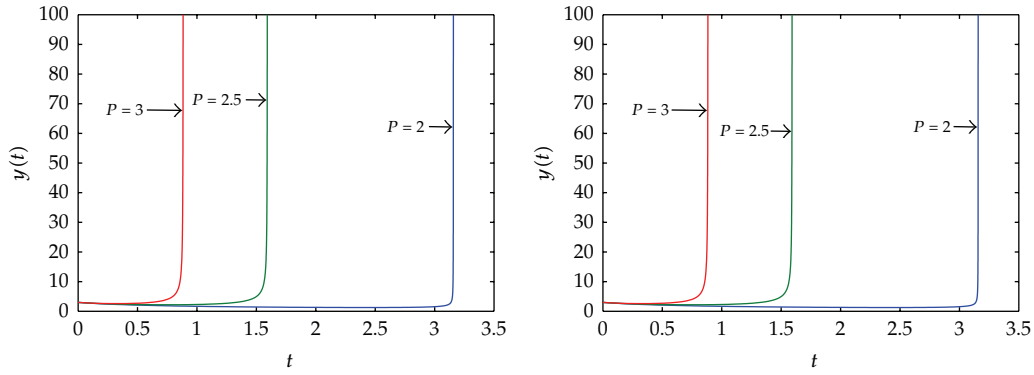
Equation (4.3) has a convolution kernel with a singularity and hence we use convolution quadrature to solve it. We have

$$\sum_{j=0}^{\infty} w_j \xi^j = F\left(\frac{\delta(\xi)}{h}\right) = \Gamma\left(\frac{1}{3}\right)\left[\frac{3}{2h} - \frac{2}{h}\xi + \frac{1}{2h}\xi^2\right]^{-1/3}, \tag{4.4}$$

where w_j is given by the generating power series (4.4), with $F(s) = \Gamma(1/3)s^{-1/3}$ and $\delta(\xi) = (1-\xi) + (1/2)(1-\xi)^2$. $\Gamma(1/3) = 2.6789385347$.

Table 4: Blow-up time for (4.2) by block-by-block method.

p	PECE-IE	IMIE	ode45	ode15s
2	2.769946	2.779028	2.779033	2.779019
2.5	1.596293	1.600180	1.600182	1.600169
3	0.960631	0.964731	0.964732	0.964730



(a) By PECE-IE

(b) By IMIE

(c) By ode45

(d) By ode15s

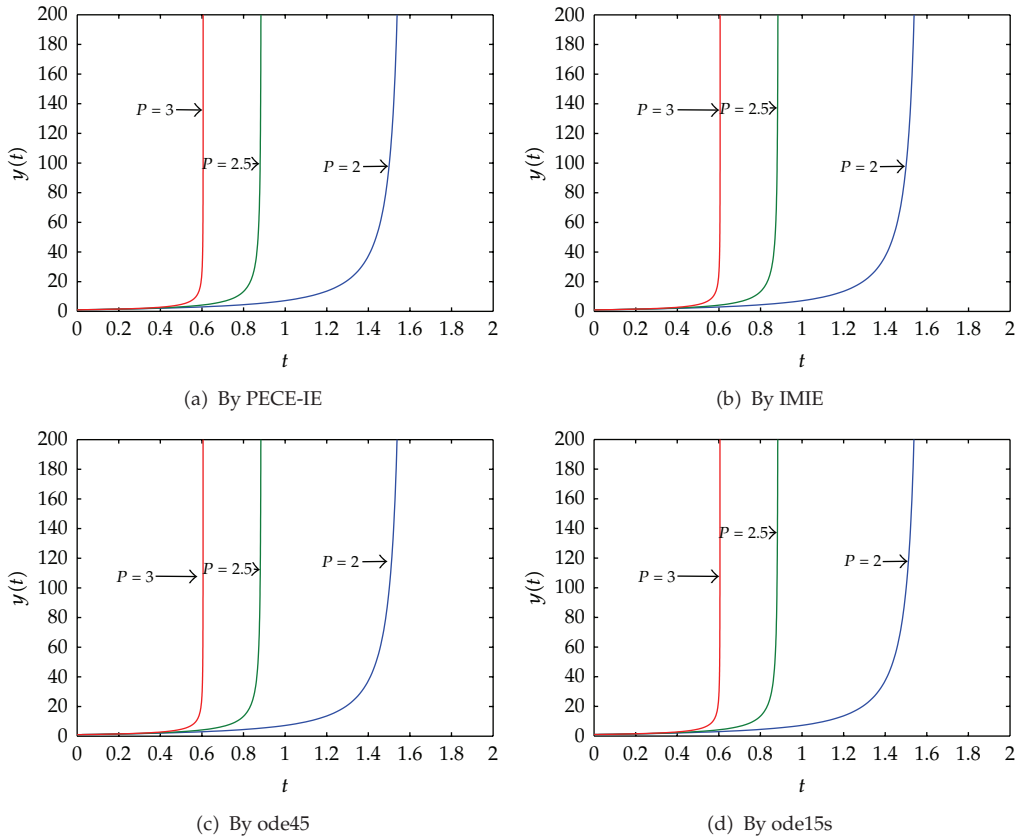
Figure 2: Numerical solution of (4.2) by repeated trapezoidal rule.

Table 5 shows the blow-up results and Figure 3 shows the solution of (4.3).

4.1. Discussion

From the results obtained, we observe that blow-up occurs earlier as the exponent p increases as shown in Figures 1, 2, and 3. Comparing the the blow-up results computed when we use the repeated trapezoidal rule and the block-by-block method, we note a significant difference in the blow-up computed, which means the choice of quadrature rule has an effect on the blow-up time. In Example 4.3, where we use convolution quadrature we get results comparable to Ma's results, (see [7]).

On the performance of the methods, we note that the IMIE method gives results which are much closer to those for ode45 which is of higher order than the other three. Thus based



(a) By PECE-IE (b) By IMIE

(c) By ode45 (d) By ode15s

Figure 3: Numerical solution of (4.3).

Table 5: Blow-up time for (4.3).

p	PECE-IE	IMIE	ode45	ode15s
2	1.574912	1.580157	1.580159	1.580153
2.5	0.882066	0.885417	0.885418	0.885413
3	0.603334	0.605826	0.605827	0.605821

on our results we conclude that the adaptive IMIE method is superior than the adaptive PECE-IE method and ode15s.

References

[1] C. Bandle and H. Brunner, "Blowup in diffusion equations: a survey," *Journal of Computational and Applied Mathematics*, vol. 97, no. 1-2, pp. 3–22, 1998.

[2] C. J. Budd, W. Huang, and R. D. Russell, "Moving mesh methods for problems with blow-up," *SIAM Journal on Scientific Computing*, vol. 17, no. 2, pp. 305–327, 1996.

[3] W. Huang, J. Ma, and R. D. Russell, "A study of moving mesh PDE methods for numerical simulation of blowup in reaction diffusion equations," *Journal of Computational Physics*, vol. 227, no. 13, pp. 6532–6552, 2008.

[4] M. Berger and R. V. Kohn, "A rescaling algorithm for the numerical calculation of blowing-up solutions," *Communications on Pure and Applied Mathematics*, vol. 41, no. 6, pp. 841–863, 1988.
[5] A. M. Stuart and M. S. Floater, "On the computation of blow-up," *European Journal of Applied Mathematics*, vol. 1, no. 1, pp. 47–71, 1990.
[6] N. Apreutesei and V. Volpert, "Properness and topological degree for nonlocal reaction-diffusion operators," *Abstract and Applied Analysis*, vol. 2011, Article ID 629692, 21 pages, 2011.
[7] J. Ma, "Blow-up solutions of nonlinear Volterra integro-differential equations," *Mathematical and Computer Modelling*, vol. 54, no. 11-12, pp. 2551–2559, 2011.
[8] H. Bellout, "Blow-up of solutions of parabolic equations with nonlinear memory," *Journal of Differential Equations*, vol. 70, no. 1, pp. 42–68, 1987.
[9] M. Wang and Y. Wang, "Properties of positive solutions for non-local reaction-diffusion problems," *Mathematical Methods in the Applied Sciences*, vol. 19, no. 14, pp. 1141–1156, 1996.
[10] P. Souplet, "Blow-up in nonlocal reaction-diffusion equations," *SIAM Journal on Mathematical Analysis*, vol. 29, no. 6, pp. 1301–1334, 1998.
[11] J. Ma, Y. Jiang, and K. Xiang, "Numerical simulation of blowup in nonlocal reaction-diffusion equations using a moving mesh method," *Journal of Computational and Applied Mathematics*, vol. 230, no. 1, pp. 8–21, 2009.
[12] P. G. Dlamini and M. Khumalo, "On the computation of blow-up solutions for semilinear ODEs and parabolic PDEs," *Mathematical Problems in Engineering*, vol. 2012, Article ID 162034, 15 pages, 2012.
[13] P. Linz, "A method for solving nonlinear Volterra integral equations of the second kind," *Mathematics of Computation*, vol. 23, pp. 595–599, 1969.
[14] J. Saberi-Nadjafi and M. Heidari, "A quadrature method with variable step for solving linear Volterra integral equations of the second kind," *Applied Mathematics and Computation*, vol. 188, no. 1, pp. 549–554, 2007.
[15] C. Lubich, "Convolution quadrature and discretized operational calculus—I," *Numerische Mathematik*, vol. 52, no. 2, pp. 129–145, 1988.
[16] C. Lubich, "Convolution quadrature and discretized operational calculus—II," *Numerische Mathematik*, vol. 52, no. 4, pp. 413–425, 1988.
[17] C. Lubich, "Convolution quadrature revisited," *BIT Numerical Mathematics*, vol. 44, no. 3, pp. 503–514, 2004.