Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 162034, 15 pages
doi:10.1155/2012/162034

Research Article

On the Computation of Blow-up Solutions for Semilinear ODEs and Parabolic PDEs

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 31 August 2011; Revised 8 November 2011; Accepted 8 November 2011

Academic Editor: Robertt A. Fontes Valente

We introduce an adaptive numerical method for computing blow-up solutions for ODEs and well-known reaction-diffusion equations. The method is based on the implicit midpoint method and the implicit Euler method. We demonstrate that the method produces superior results to the adaptive PECE-implicit method and the MATLAB solver of comparable order.

1. Introduction

Reaction-diffusion equations model a wide range of problems in physics, biology, and chemistry. They explain how the concentration of one or more substances distributed in space changes under the influence of two processes: chemical reactions and diffusion. These substances can be basic particles in physics, bacteria, molecules, cells, and so forth. The substances reside in a region $\Omega \subset \mathbb{R}^d, d \geq 1$.

The reaction-diffusion equation is a semilinear parabolic partial differential equation of the form

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

Equation (1.1) can be viewed as a heat conduction problem, where $u(x,t)$ is the temperature of a substance in a bounded domain $\Omega \subset \mathbb{R}^d$ and $f(t,x,u)$ represents a heat source. $\Delta u(t,x)$ is referred to as the diffusion term and $f(t,x,u)$ as the reaction term. In this case, convection does not take place, so f does not depend on ∇u.

For sufficiently large initial function $u_0(x)$ the solution of (1.1) will blowup in finite time. Blow-up occurs when the solution of the partial differential equation ceases to exist in finite time; that is, there is $T_b < \infty$ (*blow-up time*) so that

$$\lim_{t \to T_b^-} \|u(t,\cdot)\| = +\infty. \tag{1.2}$$

Bebernes and Eberly [1] state that a necessary condition for blow-up in finite time is if

$$\int_{u_0}^{\infty} f^{-1}(s)ds < \infty. \tag{1.3}$$

Kaplan [2] showed that for convex source terms $f = f(u)$ satisfying (1.3) diffusion cannot prevent blow-up if the initial state is large enough. In most papers, blow-up properties are discussed in the case where the nonlinear term in (1.1) is of the form $f = f(u)$ where $\Omega \subset \mathbb{R}^d$ is bounded and $t \in (0,T)$, where T is finite. In most of the work the case where $d = 1$ has been studied. However, more recently, Brunner et al. [3] studied the numerical solution of blow-up problems within the context of unbounded domains.

Stuart and Floater [4] 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. Bandle and Brunner [5] present a survey of the theory and the numerical analysis of blow-up solutions for quasilinear reaction-diffusion equations. Budd et al. [6] proposed the use of moving mesh partial differential equation (MMPDE) methods for solving (1.1). In this method the function $u(x,t)$ is discretized to give the solution values $u_i(t)$ defined on a moving mesh $x_i(t)$, $i = 0,\dots,N$. A more general study of the MMPDE is presented in [7] and the references therein. More recently Ma et al. [8, 9] have used the moving mesh methods to numerically study blow-up in nonlocal reaction diffusion equations and partial integrodifferential equations in general.

In this paper we will use the method of lines (MOLs) to solve (1.1). In this method the PDE is discretized in space, which leads to a system of ODEs with initial conditions. The numerical solution can then be obtained by solving the ODE initial value problems (see [10]). We introduce an adaptive method based on the implicit midpoint method and the implicit Euler method to solve the resulting system of ODEs. More work has been done on PsDEs with autonomous nonlinear reaction term. In this work we also give numerical results for PDEs with nonautonomous nonlinear term.

2. Description of Methods Used

In this work we use variable step methods to compute the blow-up time. We use one-point collocation methods and compare the results with MATLAB solvers ode45 and ode15s. 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 adjust the stepsize in accordance with $|S1 - S2|$.

2.1. One-Point Collocation Methods

One-point collocation methods are a family of methods of the form

$$y_{n+1} = y_n + h_n f\left(t_n + c_1 h_n, (1 - c_1) y_n + c_1 y_{n+1}\right), \tag{2.1}$$

where $c_1 \in [0, 1]$. The specified cases are

(i) $c_1 = 0$ corresponds to explicit Euler method;

(ii) $c_1 = 1/2$ corresponds to implicit midpoint method;

(iii) $c_1 = 1$ corresponds to implicit Euler method.

Note that all the one-point collocation methods are of order 1; however, the implicit midpoint method ($c_1 = 1/2$) achieves order 2 local superconvergence (see [11]).

2.2. Adaptive PECE-Implicit Euler Method

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

$$S1 = y_{n+1} = y_n + h_n f\left(t_{n+1}, y_p\right), \tag{2.2}$$

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

2.3. Adaptive Implicit Midpoint-Implicit Euler Method

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

$$y_{n+1} = y_n + h_n f\left(t_n + \frac{h_n}{2}, \frac{y_n + y_{n+1}}{2}\right). \tag{2.3}$$

First we get y_{n+1} using the implicit Euler method and substitute in (2.3) to get $S2$, that is,

$$y_p = y_n + h_n f\left(t_{n+1}, y_{n+1}\right), \tag{2.4}$$

then,

$$S2 = y_{n+1} = y_n + h_n f\left(t_n + \frac{h_n}{2}, \frac{y_n + y_p}{2}\right). \tag{2.5}$$

2.4. 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 with the adaptive implicit midpoint-implicit Euler method we are introducing.

3. Blow-up for ODEs

We will first consider this simple case

$$y'(t) = \lambda y(t) + by^p(t), \quad t > 0,\ y(0) = y_0 > 0, \tag{3.1}$$

with $\lambda < 0$ and $b > 0$.

3.1. Analytic Solution

Theorem 3.1. *Given the system* (3.1), *its solution will blowup in finite time for*

$$y_0 > \left(\frac{-b}{\lambda}\right)^{1/(1-p)} \tag{3.2}$$

and at

$$T_b = \frac{1}{\lambda(1-p)} \ln \frac{b}{\lambda y_0^{1-p} + b} \tag{3.3}$$

(see Brunner [11]).

Proof. Note that (3.1) is a Bernoulli equation, whose solution is

$$y(t) = \left[\frac{-b}{\lambda} + \left(y_0^{(1-p)} + \frac{b}{\lambda}\right)e^{\lambda(1-p)t}\right]^{1/(1-p)}. \tag{3.4}$$

Then, $y \to \infty$ when

$$\frac{-b}{\lambda} + \left(y_0^{(1-p)} + \frac{b}{\lambda}\right)e^{\lambda(1-p)t} = 0. \tag{3.5}$$

Thus

$$T_b = \frac{1}{\lambda(1-p)} \ln \frac{b}{\lambda y_0^{1-p} + b}. \tag{3.6}$$

Table 1: Blow-up solutions of (3.1).

p	Blow-up time (T_b)
1.1	27.03555
1.5	2.455894
2	0.693147
3	0.143841

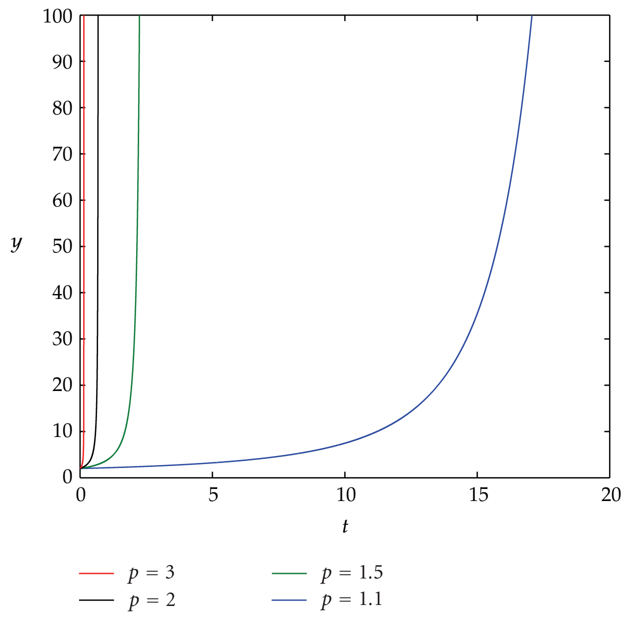


Figure 1: Analytic solution of (3.1).

T_b is finite if

$$\lambda y_0^{1-p} + b > 0. \tag{3.7}$$

Thus

$$y_0 > \left(\frac{-b}{\lambda}\right)^{1/(1-p)}, \tag{3.8}$$

as required. □

We compute the blow-up time for $\lambda = -1$, $b = 1$, $y_0 = 2$, and $p = 1.1, 1.5, 2, 3$. The results are shown in Table 1, and a graph showing the analytic solution of (3.1) is shown in Figure 1.

Table 2: Blow-up time for (3.1).

p	PECE-implicit Euler	Midpoint-implicit Euler	ode45	ode15s
1.1	26.89733	27.03530	27.03555	27.03528
1.5	2.442115	2.455886	2.455895	2.455857
2	0.6889782	0.6931451	0.6931474	0.6931295
3	0.1429036	0.1438402	0.1438411	0.1438358

Table 3: Error for (3.1).

p	PECE-implicit Euler	Midpoint-implicit Euler	ode45	ode15s
1.1	0.861780	0.000250	0	0.000270
1.5	0.013779	0.000008	0	0.000038
2	0.004169	0.000002	0	0.000018
3	0.000937	0.000001	0	0.000005

3.2. Numerical Computation

We solve (3.1) with $\lambda = -1$, $b = 1$, $y_0 = 2$, and $p = 1.1, 1.5, 2, 3$. Tables 2 and 3 show the blow-up times and the errors of each method, respectively, and Figure 2 shows the graphs of the solution of (3.1). The tolerance used for the computations is $1e - 6$.

3.3. Discussion

The blow-up results for the different methods are very close to the analytic value as shown in Table 2. From Table 3, we see that ode45, which is of higher order than the other three methods, gives the best results than the other three methods. The adaptive implicit midpoint-implicit Euler gives a better result than the other two methods of comparable order, that is, adaptive PECE-implicit Euler method and ode15s. The adaptive PECE-implicit Euler method gives a quite large error and requires a very small tolerance to get a result which is close to the exact value. From the results, we observe that as the value of p in (3.1) is increased the blow-up time occurs earlier and is much later for values of p much closer to 1.

We seek to determine whether the performance of the methods is the same in the case where we have a reaction-diffusion equation.

4. Reaction-Diffusion Equation: One Space Dimension

In this section we compute the blow-up time for a one space dimension reaction-diffusion equation with an autonomous reaction term and a nonautonomous reaction term.

4.1. Autonomous Reaction Term

We solve the system (1.1) with the autonomous reaction term $f = u^p(t,x)$, where $p > 1$ and the domain Ω is just the real line, that is, $d = 1$ and $\Omega = (0,1)$. The system becomes

$$\begin{gathered} u_t(t,x) - u_{xx}(t,x) = u^p(t,x), \quad t > 0,\ x \in (0,1),\ (p > 1), \\ u(0,x) = u_0(x) \geq 0, \quad x \in (0,1), \\ u(t,x) = 0, \quad t > 0,\ x \in \partial\Omega. \end{gathered} \tag{4.1}$$

We use the method of lines (MOLs) to discretize (4.1) in space. For the spatial discretization, we choose a uniform mesh $D_h := \{x_m : 0 = x_0 < x_1 < \cdots < x_{M+1} = 1\}$ (with $x_m = mh$) on Ω and replace $u_{xx}(t, x_m)$ $(1 \leq m \leq M)$ by the standard central difference approximation. We use the function $A\sin(\pi x)$ as the initial function with different values of $A > 0$. We get the following system of ODEs for $U_m(t) \approx u(t, x_m)(1 \leq m \leq M)$:

$$\begin{gathered}
U_m'(t) = \frac{U_{m+1}(t) - 2U_m(t) + U_{m-1}(t)}{h^2} + U_m^p(t), \quad (1 \leq m \leq M), \\
U_0(t) = U_{M+1}(t) = 0, \\
U_m(0) = A\sin(\pi x_m).
\end{gathered} \tag{4.2}$$

We solve the system (4.2) with $p = 2$ and $h = (1-0)/M$. Tables 4, 5, and 6 show the blow-up results obtained with $M = 50, 100$, and 200, respectively, and with different values of A in the initial function $A\sin(\pi x)$. Figures 3, 4, 5, and 6 show the graphs of the solution of (4.1) for $A = 10$ and $A = 12$.

4.2. Nonautonomous Reaction Term

We now solve the system (1.1) with the non-autonomous reaction term $f = t^k x^r u^p(t,x)$, where $p > 1$ and the domain Ω is just the real line, that is, $d = 1$ and $\Omega = (0,1)$. The system becomes

$$\begin{gathered}
u_t(t,x) - u_{xx}(t,x) = t^k x^r u^p(t,x), \quad t > 0, \ x \in (0,1), \ (p > 1), \\
u(0,x) = u_0(x) \geq 0, \quad x \in (0,1), \\
u(t,x) = 0, \quad t > 0, \ x \in \partial\Omega.
\end{gathered} \tag{4.3}$$

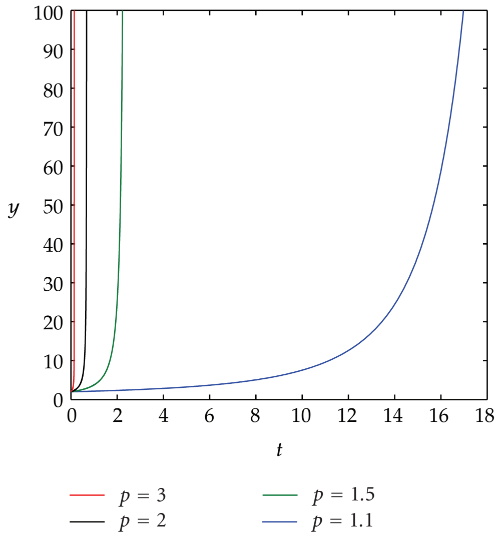
As in (4.1) we use the method of lines to discretize (4.3) to obtain the following system:

$$\begin{gathered}
U_m'(t) = \frac{U_{m+1}(t) - 2U_m(t) + U_{m-1}(t)}{h^2} + t^k x_m^r U_m^p(t), \quad (1 \leq m \leq M), \\
U_0(t) = U_{M+1}(t) = 0, \\
U_m(0) = A\sin(\pi x_m).
\end{gathered} \tag{4.4}$$

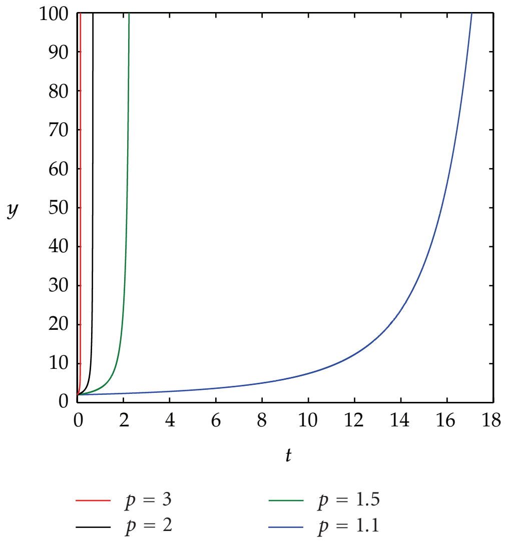
We solve (4.4) for $k = 0, 1$ and $r = 1, 2$, Tables 7 and 8 show the blow-up times obtained from the different values of k and r.

4.3. Discussion

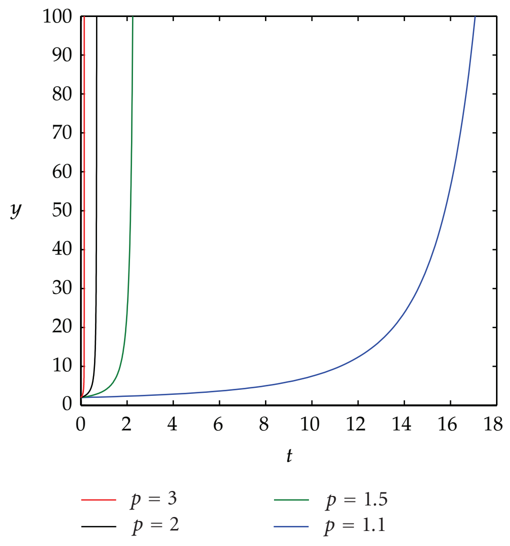
We observe that as we increase the amplitude of the initial function, A, the blow-up time tends to occur earlier. We also observe that for smaller values of A, there is no blow-up.



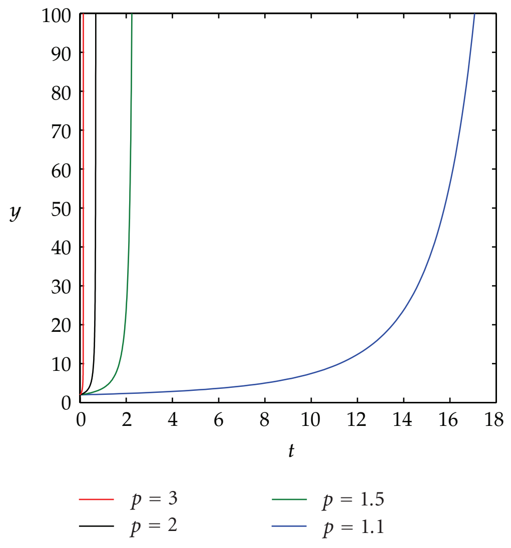
(a) By PECE-implicit Euler



(b) By midpoint-implicit Euler



(c) By ode45



(d) By ode15s

Figure 2: Numerical solution of (3.1).

Considering the reaction-diffusion equation, and comparing the autonomous against the non-autonomous reaction term cases, we observe that the introduction of the non-autonomous term ensures that a much larger amplitude, A, in the initial function, is required for blow-up to occur. We also note that increasing k for fixed r or increasing r for fixed k increases the minimum amplitude for blow-up to occur.

On the performance of the methods, we note a similar trend to what we observed in the ODE case. The adaptive implicit midpoint-implicit Euler method gives results that are significantly superior to the adaptive PECE-implicit Euler method and ode15s. In fact, its performance is comparable to ode45.

Table 4: Blow-up time for (4.1) with $M = 50$.

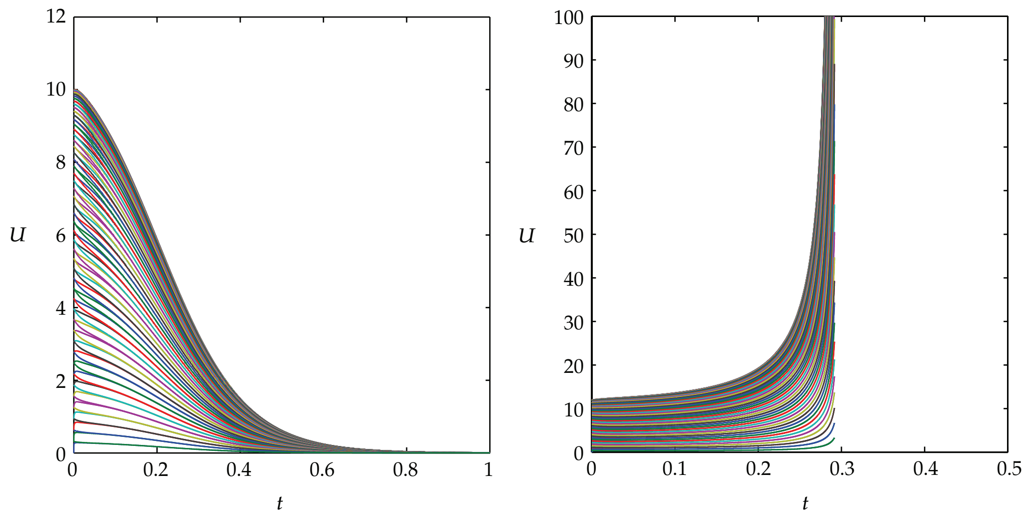
	Blow-up time (T_b)			
A	PECE-implicit Euler	Midpoint-implicit Euler	ode45	ode15s
8.0	No blow-up	No blow-up	No blow-up	No blow-up
10.0	No blow-up	No blow-up	No blow-up	No blow-up
10.5	No blow-up	No blow-up	No blow-up	No blow-up
11.0	No blow-up	No blow-up	No blow-up	No blow-up
11.1	No blow-up	No blow-up	No blow-up	No blow-up
11.2	0.6184	0.6204	0.6225	0.6198
11.3	0.4611	0.4629	0.4633	0.4625
11.4	0.4019	0.4036	0.4038	0.4033
11.5	0.3649	0.3666	0.3668	0.3663
12.0	0.2729	0.2745	0.2746	0.2743
12.5	0.2280	0.2295	0.2296	0.2294
13.0	0.1989	0.2005	0.2004	0.2002
13.5	0.1777	0.1791	0.1791	0.1790
14.0	0.1613	0.1626	0.1627	0.1626
14.5	0.1480	0.1493	0.1494	0.1493
15.0	0.1370	0.1383	0.1384	0.1383
15.5	0.1277	0.1290	0.1290	0.1290
16.0	0.1197	0.1209	0.1209	0.1209

Table 5: Blow-up time for (4.1) with $M = 100$.

	Blow-up time (T_b)			
A	PECE-implicit Euler	Midpoint-implicit Euler	ode45	ode15s
8.0	No blow-up	No blow-up	No blow-up	No blow-up
10.0	No blow-up	No blow-up	No blow-up	No blow-up
10.5	No blow-up	No blow-up	No blow-up	No blow-up
11.0	No blow-up	No blow-up	No blow-up	No blow-up
11.1	No blow-up	No blow-up	No blow-up	No blow-up
11.2	No blow-up	No blow-up	No blow-up	No blow-up
11.3	No blow-up	No blow-up	No blow-up	No blow-up
11.4	0.5322	0.5330	0.5340	0.5325
11.5	0.4323	0.4331	0.4335	0.4328
12.0	0.2923	0.2924	0.2925	0.2922
12.5	0.2382	0.2389	0.2390	0.2388
13.0	0.2062	0.2063	0.2063	0.2062
13.5	0.1826	0.1833	0.1833	0.1832
14.0	0.1651	0.1658	0.1658	0.1656
14.5	0.1511	0.1518	0.1518	0.1516
15.0	0.1396	0.1403	0.1403	0.1402
15.5	0.1299	0.1306	0.1306	0.1305
16.0	0.1216	0.1222	0.1223	0.1222

Table 6: Blow-up time for (4.1) with $M = 200$.

	Blow-up time (T_b)			
A	PECE-implicit Euler	Midpoint-implicit Euler	ode45	ode15s
8.0	No blow-up	No blow-up	No blow-up	No blow-up
10.0	No blow-up	No blow-up	No blow-up	No blow-up
10.5	No blow-up	No blow-up	No blow-up	No blow-up
11.0	No blow-up	No blow-up	No blow-up	No blow-up
11.1	No blow-up	No blow-up	No blow-up	No blow-up
11.2	No blow-up	No blow-up	No blow-up	No blow-up
11.3	No blow-up	No blow-up	No blow-up	No blow-up
11.4	No blow-up	No blow-up	No-blow-up	No blow-up
11.5	0.5074	0.5077	0.5085	0.5067
12.0	0.3034	0.3037	0.3038	0.3034
12.5	0.2441	0.2444	0.2444	0.2442
13.0	0.2094	0.2097	0.2097	0.2095
13.5	0.1853	0.1856	0.1856	0.1855
14.0	0.1672	0.1675	0.1675	0.1673
14.5	0.1528	0.1531	0.1531	0.1530
15.0	0.1410	0.1413	0.1413	0.1412
15.5	0.1311	0.1314	0.1314	0.1313
16.0	0.1227	0.1229	0.1229	0.1229



(a) with $A = 10$

(b) with $A = 12$

Figure 3: Numerical solution of (4.1) obtained using PECE-implicit Euler.

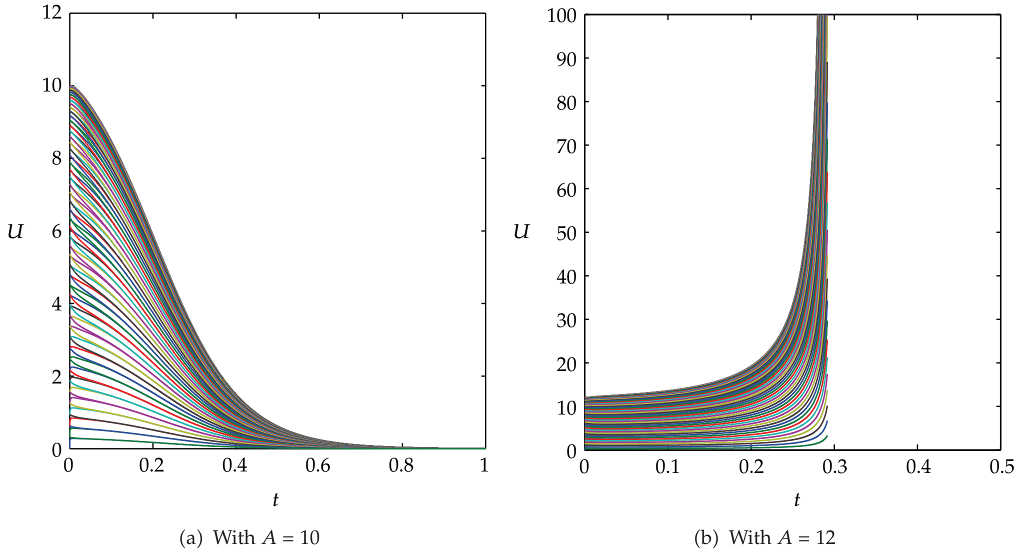


Figure 4: Numerical solution of (4.1) obtained using midpoint-implicit Euler.

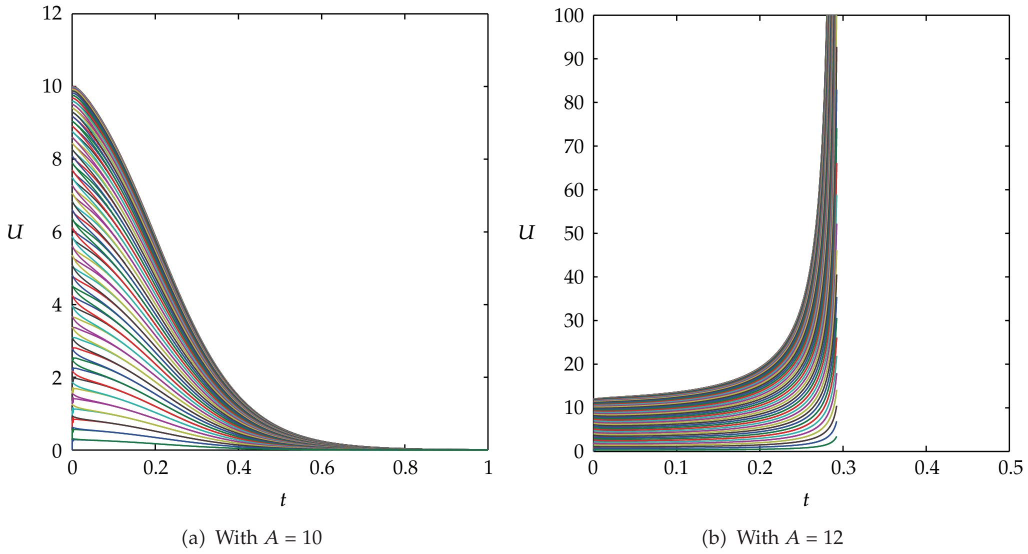


Figure 5: Numerical solution of (4.1) obtained using ode45.

5. Reaction-Diffusion Equation: Two Space Dimensions

We solve the system (1.1) with the reaction term $f = u^p(t,x,y)$, where $p > 1$ with $\Omega = \mathbb{R}^2$. The system becomes

$$\begin{gathered} u_t(t,x,y) - u_{xx}(t,x,y) - u_{yy}(t,x,y) = u^p(t,x,y), \quad t > 0,\ x,y \in (0,1),\ (p > 1), \\ u(0,x,y) = u_0(x,y) \geq 0, \quad x,y \in (0,1), \\ u(t,x,y) = 0, \quad t > 0,\ x,y \in \partial\Omega. \end{gathered} \tag{5.1}$$

Table 7: Blow-up time for (4.1) with $M = 100$ and $k = 0$.

		Blow-up time (T_b)			
r	A	PECE-implicit Euler	Midpoint-implicit Euler	ode45	ode15s
1	21.0	No blow-up	No blow-up	No blow-up	No blow-up
	21.5	No blow-up	No blow-up	No blow-up	No blow-up
	22.0	0.4670	0.4679	0.4681	0.4675
	22.5	0.3305	0.3312	0.3313	0.3310
	23.0	0.2780	0.2787	0.2787	0.2785
	23.5	0.2454	0.2461	0.2461	0.2460
	24.0	0.2221	0.2228	0.2228	0.2227
2	36.0	No blow-up	No blow-up	No blow-up	No blow-up
	36.5	No blow-up	No blow-up	No blow-up	No blow-up
	37.0	0.3673	0.3681	0.3681	0.3679
	37.5	0.3038	0.3045	0.3045	0.3043
	38.0	0.2683	0.2690	0.2690	0.2688
	38.5	0.2438	0.2444	0.2445	0.2443
	39.0	0.2252	0.2258	0.2259	0.2257

Table 8: Blow-up time for (4.1) with $M = 100$ and $k = 1$.

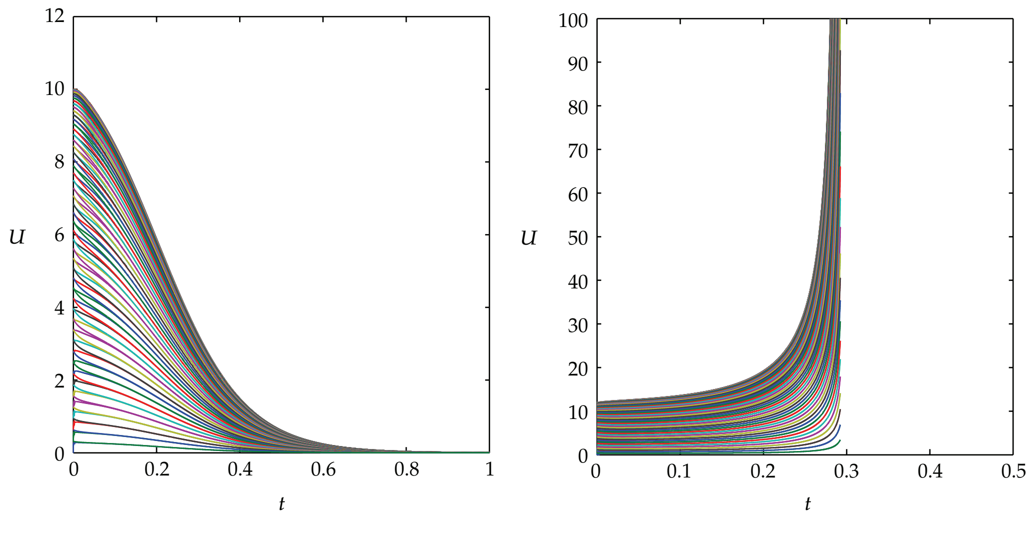
		Blow-up time (T_b)			
r	A	PECE-implicit Euler	Midpoint-implicit Euler	ode45	ode15s
1	215	No blow-up	No blow-up	No blow-up	No blow-up
	216	0.8041	0.8367	0.8367	0.8410
	217	0.6872	0.6981	0.6981	0.6956
	218	0.6310	0.6378	0.6378	0.6384
	219	0.5936	0.5987	0.5987	0.5990
	220	0.5655	0.5696	0.5697	0.5684
2	370	No blow-up	No blow-up	No blow-up	No blow-up
	371	0.8054	0.8671	0.8671	0.8754
	372	0.6949	0.7126	0.7126	0.7142
	373	0.6427	0.6533	0.6533	0.6541
	374	0.6080	0.6157	0.6157	0.6162
	375	0.5819	0.5881	0.5881	0.5884

We use the method of lines (MOLs) to discretize (5.1) in space. For the spatial discretization, we choose uniform meshes for x and y, $D_h := \{x_m : 0 = x_0 < x_1 < \cdots < x_{M+1} = 1\}$ and $I_h := \{y_n : 0 = y_0 < y_1 < \cdots < y_{N+1} = 1\}$, respectively, (with $x_m = mh$ and $y_n = nh$) on Ω. We replace $u_{xx}(t, x_m, y_n)$ and $u_{yy}(t, x_m, y_n)$ $(1 \leq m \leq M, 1 \leq n \leq N)$ by the standard central difference approximation. We use the function $A\sin(\pi x)\sin(\pi y)$ as the initial function with different values of $A > 0$. We get the following system of ODEs for $U_{m,n}(t) \approx u(t, x_m, y_n)$ $(1 \leq m \leq M, 1 \leq n \leq N)$:

$$\begin{gathered}
U'_{m,n}(t) = \frac{U_{m+1,n}(t) + U_{m-1,n}(t) - 4U_{m,n} + U_{m,n+1}(t) + U_{m,n-1}(t)}{h^2} + U^p_{m,n}(t), \\
U_{0,n}(t) = U_{M+1,n}(t) = U_{m,0} = U_{m,N+1} = 0, \\
U_{m,n}(0) = A\sin(\pi x_m)\sin(\pi y_n), \quad 1 \leq m \leq M,\ 1 \leq n \leq N.
\end{gathered} \tag{5.2}$$

Table 9: Blow-up time for (5.1) with $M = 10$ and $N = 10$.

	Blow-up time (T_b)			
A	PECE-implicit Euler	Midpoint-implicit Euler	ode45	ode15s
21	no blow-up	no blow-up	no blow-up	no blow-up
22	no blow-up	no blow-up	no blow-up	no blow-up
23	0.4717971	0.4721022	0.4721028	0.4720943
24	0.2034135	0.2036998	0.2037004	0.2036986
25	0.1564947	0.1567547	0.1567553	0.1567541
26	0.1306537	0.1308970	0.1308976	0.1308967
27	0.1133313	0.1135619	0.1135625	0.1135617
28	0.1005905	0.1008109	0.1008115	0.1008108
29	0.0906900	0.0909017	0.0909023	0.0909017
30	0.0827090	0.0829133	0.0829139	0.0829133



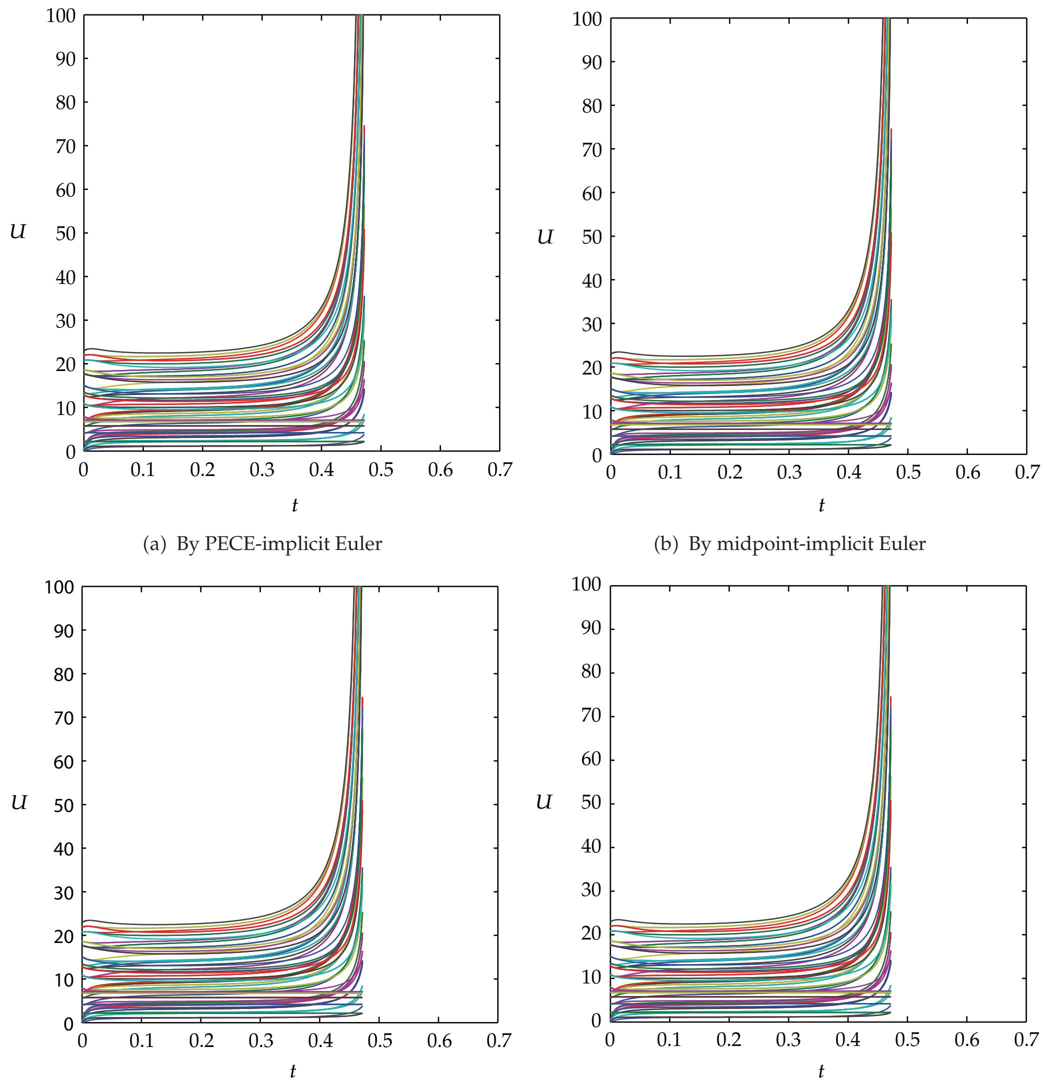
(a) With $A = 10$ (b) With $A = 12$

Figure 6: Numerical solution of (4.1) obtained using ode15s.

We solve the system (5.2) with $p = 2$ and $h = (1 - 0)/M$. Figure 7 shows the solution of (5.2), and Table 9 shows the blow-up times obtained using the four methods.

5.1. Discussion

We observe similar results for the two space dimensions reaction-diffusion equation to the one space dimension case; that is, the adaptive implicit midpoint-implicit Euler method gives significantly better results than the adaptive PECE-implicit Euler method and ode15s. Its results are comparable with the higher order, computationally costly ode45.



(a) By PECE-implicit Euler

(b) By midpoint-implicit Euler

(c) By ode45

(d) By ode15s

Figure 7: Numerical solution of (5.1).

6. Future Work

We would like to extend the blow-up computations to the Volterra integrodifferential equations, that is, cases where the reaction term is nonlocal.

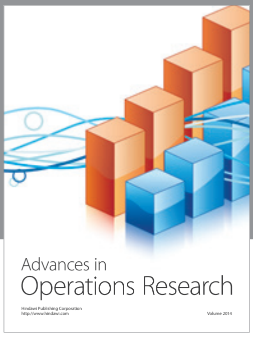
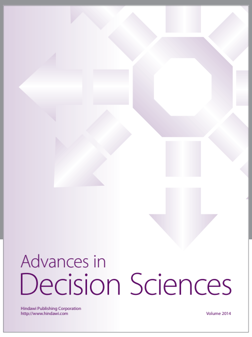
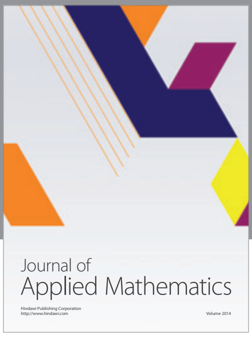
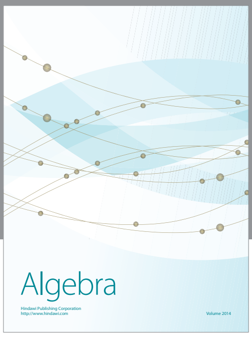
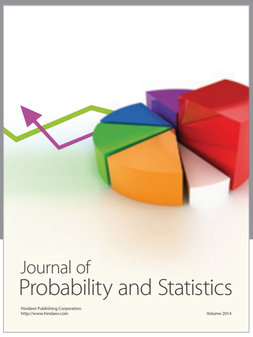
Given the relative simplicity of this new method, and cheaper computational expense, we conclude that it is much better than the higher-order RK-based solver, for implementation on problems of the nature studied in this paper. We would further like to test the new method on other problems in engineering and applied science, with the objective of proposing it for wider implementation.

Acknowledgment

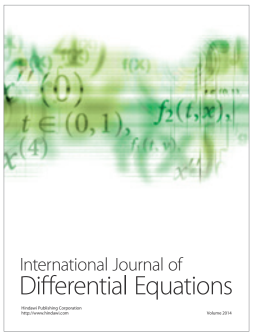
The authors gratefully acknowledge the valuable discussions with Professor Hermann Brunner on finite-time blow-up problems.

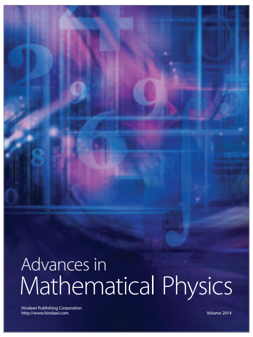
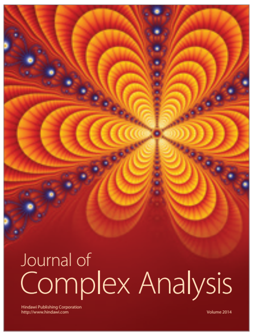
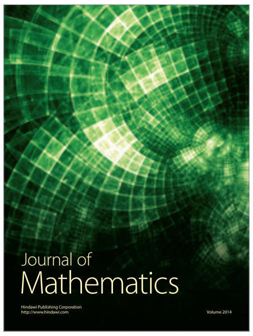
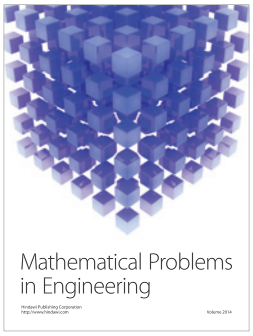
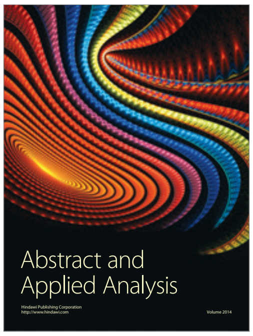
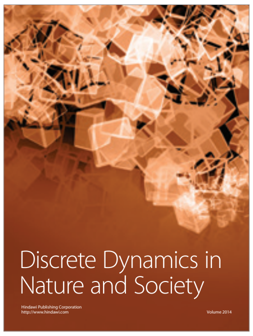
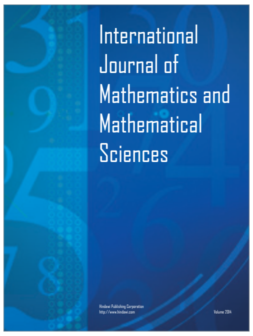
References

[1] J. Bebernes and D. Eberly, *Mathematical Problems from Combustion Theory*, vol. 83 of *Applied Mathematical Sciences*, Springer, New York, NY, USA, 1989.
[2] S. Kaplan, "On the growth of solutions of quasi-linear parabolic equations," *Communications on Pure and Applied Mathematics*, vol. 16, pp. 305–330, 1963.
[3] H. Brunner, X. Wu, and J. Zhang, "Computational solution of blow-up problems for semilinear parabolic PDEs on unbounded domains," *SIAM Journal on Scientific Computing*, vol. 31, no. 6, pp. 4478–4496, 2009/10.
[4] 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.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] J. Ma, Y. Jiang, and K. Xiang, "On a moving mesh method for solving partial integro-differential equations," *Journal of Computational Mathematics*, vol. 27, no. 6, pp. 713–728, 2009.
[10] J. Chen, Numerical Study of Blowup Problems and Conservation Laws with Moving Mesh Methods, 1996, http://ir.lib.sfu.ca/handle/1892/8368.
[11] H. Brunner, *Collocation Methods for Volterra Integral and Related Functional Differential Equations*, vol. 15 of *Cambridge Monographs on Applied and Computational Mathematics*, Cambridge University Press, Cambridge, UK, 2004.

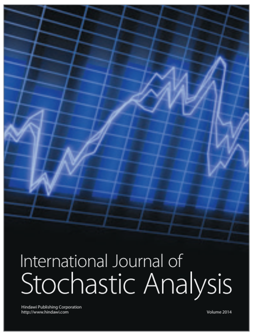

Advances in
Operations Research

Advances in
Decision Sciences

Journal of
Applied Mathematics

Algebra

Journal of
Probability and Statistics

The Scientific
World Journal


International Journal of
Differential Equations

International Journal of
Combinatorics

Advances in
Mathematical Physics

Journal of
Complex Analysis

Journal of
Mathematics

Mathematical Problems
in Engineering

Abstract and
Applied Analysis

Discrete Dynamics in
Nature and Society

International
Journal of
Mathematics and
Mathematical
Sciences

Journal of
Discrete Mathematics

Journal of
Function Spaces

International Journal of
Stochastic Analysis

Journal of
Optimization