Available online at www.sciencedirect.com

ScienceDirect

Applied Mathematics and Computation 195 (2008) 76–85

APPLIED MATHEMATICS AND COMPUTATION

www.elsevier.com/locate/amc

Moving mesh method with local time step refinement for blow-up problems

Ali R. Soheili *, Soheil Salahshour

Department of Mathematics, University of Sistan and Baluchestan, Zahedan, Iran

Abstract

In this paper, the dynamical moving mesh method is merged with the local time stepping technique and the new method is applied for blow-up problems. It has some benefit in approximating an accurate blow-up time by starting at any positive initial time ($t_0 > 0$).

Our numerical experiment shows that without applying the local time stepping method, wrong blow-up time will be obtained, if the equation integrates from $t_0 > 0$.

© 2007 Elsevier Inc. All rights reserved.

Keywords: Moving mesh method; Local time step refinement; Blow-up problem

1. Introduction

A wide class of mathematical idealization in physical models has singular property in finite time. Semi-linear parabolic equation which has a blow-up behavior is a kind of model that describes ignition in chemical reaction.

Sometimes, in such models important changes occur which change the properties of model. Thus the position and time of these changing are important, if one wants to know *when, where and how* these changes occur accurately. Such equation has no exact solution, and should be solved numerically.

Moving mesh method in the last decade was used for solving time-dependent PDEs with large variation in solutions widely. For solving such equations, moving mesh methods are divided to static and dynamic parts. In dynamic state, mesh equation and original differential equation are solved simultaneously. It is well known that the time steps associated with the moving mesh methods are proportional to the smallest mesh size in space. It is then natural to use local time stepping refinement (LTSR) to increase the efficiency of moving mesh methods. Local time stepping for one-dimensional conversation laws was first proposed by Osher and Sanders [8]. Tang and co-workers [11] then proposed variable time stepping for one and two dimension at conversation

* Corresponding author.
E-mail addresses: soheili@math.usb.ac.ir (A.R. Soheili), soheilsalahshour@yahoo.com (S. Salahshour).

0096-3003/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2007.04.073

laws, which use semi-dynamic moving mesh method such that the CFL condition be used to obtain time interval refinement to compute the solution of system.

All of the moving mesh methods were investigated solutions of blow-up problem by starting from $t_0 = 0$. Sometimes, we need to decrease the relation between the initial condition and the first time step $t_0 = 0$. By starting from $t_0 > 0$ and using the same initial condition, the accurate blow-up time or the blow-up point may not be obtained. In the section of numerical results, we demonstrate that the MMPDE method with the LTSR may be applied in this case to modify the solution such that the accurate blow-up time or the blow-up point of the system be obtained like $t_0 = 0$.

According to our numerical experiment, a logical relative relation between variation of the blow-up time t^* and variation of the first time step t_0 is earned.

This paper is organized as follows: In Section 2, we describe the moving mesh methods with local time stepping refinement technique for 1D problems. In Section 3, we consider blow-up equation and discrete it based on computational coordinates, then convert it to quasi-lagrangian form. In Section 4, several new smooth monitor functions are studied to investigate a better solution and integrate the equation further close to the blow-up time. Numerical results have been presented in this section for different monitor functions with and without LTSR.

2. Moving mesh method based on local time stepping refinement

One of the main tasks of adaptive methods for solving partial differential equations is a suitable mesh generation at each time step. This mesh generation is basically derived with the equidistribution principle and usually formulated with respect to the monitor function [5,6,10]. The local time stepping refinement is added for these kinds of moving mesh methods [11]. For solving problems using moving mesh, the positions that have quick and sudden changes should be found and the mesh points should be concentrated into these positions.

Most of the monitor functions are defined based on some physical and geometrical properties. Various functions are proposed such as the arclength lacking enough ability to tend the approximation of blow-up problem to the self similar solution. Another important monitor function is $M(u) = u^{p-1}$ based upon the analysis of blow-up problem [1]. Two new monitor functions were considered and a comparison of the numerical results among these monitor functions is made. Sometimes, the phenomenon which is not set in appropriate coordinates for doing mathematical computations, transfers to another appropriate coordinate which helps in doing mathematical calculation easily. The first coordinate system is the physical coordinate and the second one is the computational coordinate. Suppose that (x, t) is presentation in physical coordinates and (ξ, t) are computational coordinates, let function $u(x, t)$ define $x_L < x < x_R$, where

$$x_L = x_0 < x_1(t) < \cdots < x_{n-1}(t) < x_n = x_R.$$

Each of the two coordinates are defined on $[0, 1]$. Usually mesh points are distributed uniformly on computational domain

$$\xi_i = \frac{i}{n}, \quad i = 1, 2, \ldots, n, \tag{1}$$

where n is the number of partition on $[0, 1]$.

2.1. The equidistribution principle

The equidistribution principle is the key strategy for the chosen appropriate mesh; say if there is a measure of error from M, the appropriate mesh should be chosen based on the following below rule: the error in all of the subintervals should be distributed equally.

Now, with using a suitable monitor function, the area of sudden and rapid changes of $u(x, t)$ is determined and also the moving mesh equation (in dynamic case) or mesh redistribution technique(in static case), try to concentrate most mesh points in these positions. The one to one function which connects the physical and the computational coordinates usually has the form

$$\xi(x,t) = \frac{\int_0^x M(\tilde{x},t)\mathrm{d}\tilde{x}}{\theta(t)}, \tag{2}$$

where

$$\theta(t) = \int_0^1 M(\tilde{x},t)\mathrm{d}\tilde{x}. \tag{3}$$

If mesh points ξ_i are obtained by x_i based on (2), then

$$\int_{x_i}^{x_{i+1}} M(\tilde{x},t)\mathrm{d}\tilde{x} = \int_0^{x_{i+1}} M(\tilde{x},t)\mathrm{d}\tilde{x} - \int_0^{x_i} M(\tilde{x},t)\mathrm{d}\tilde{x} = \theta(t)[\xi(x_{i+1},t) - \xi(x_i,t)]. \tag{4}$$

Consider ξ as a uniform mesh, then

$$\int_{x_i}^{x_{i+1}} M(\tilde{x},t)\mathrm{d}\tilde{x} = \frac{\theta(t)}{n}, \quad i = 1,2,\ldots,n-1. \tag{5}$$

In other words, we have

$$\int_{x_i}^{x_{i+1}} M(\tilde{x},t)\mathrm{d}\tilde{x} = \int_{x_i}^{x_{i-1}} M(\tilde{x},t)\mathrm{d}\tilde{x} = \frac{\theta(t)}{n}, \tag{6}$$

equivalently, and on differentiating is obtained

$$\frac{\partial}{\partial\xi}\left(M\frac{\partial x}{\partial\xi}\right) = 0, \quad x(0,t) = 0, \quad x(1,t) = 1. \tag{7}$$

A mesh is equidistributed when it satisfies in (7).

In order to obtain an accurate and non-oscillatory solution, it is necessary to smoothen the mesh points. In [3], a technique is used which smoothens the nodes concentration defined by $\frac{1}{x_{i+1}-x_i}$. In [12], it has been proven that smoothing node is basically equivalent to smoothing the monitor function over all points. We apply a smooth monitor function which is proposed in [6]

$$\widetilde{M}_i = \frac{\sum_{k=i-i_p}^{i+i_p} M_k \left(\frac{\gamma}{\gamma+1}\right)^{|k-i|}}{\sum_{i-i_p}^{i+i_p} \left(\frac{\gamma}{\gamma+1}\right)^{|k-i|}}, \tag{8}$$

where i_p is a nonnegative integer and γ is a positive constant such that in this paper $\gamma = 2$ and $i_p = 2$.

2.2. Moving mesh methods for PDEs

For moving mesh points based on natural behavior of the problem, equations are derived to do it. Such equations are named moving mesh partial differential equations (MMPDEs). Different MMPDEs and their stability are discussed in [6,7].

For various MMPDEs which are proposed in [6], we apply MMPDE4 and MMPDE6

$$\tau\frac{\partial}{\partial\xi}\left(M\frac{\partial\dot{x}}{\partial\xi}\right) = -\frac{\partial}{\partial\xi}\left(M\frac{\partial x}{\partial\xi}\right) \quad \text{(MMPDE4)} \tag{9}$$

and

$$\tau\frac{\partial^2\dot{x}}{\partial\xi^2} = -\frac{\partial}{\partial\xi}\left(M\frac{\partial x}{\partial\xi}\right) \quad \text{(MMPDE6)}, \tag{10}$$

where $\dot{x}$ denotes $\frac{\partial x}{\partial t}|_{\xi\mathrm{Fix}}$ and τ is a small parameter which tends the mesh toward equidistribution state [2,9]. By central finite differences the following discrete form of moving mesh equations for MMPDE4 and MMPDE6 are obtained

$$\tau(\widetilde{M}_{i+\frac{1}{2}}(\dot{x}_{i+1} - \dot{x}_i) - \widetilde{M}_{i-\frac{1}{2}}(\dot{x}_i - \dot{x}_{i-1})) = -(\widetilde{M}_{i+\frac{1}{2}}(x_{i+1} - x_i) - \widetilde{M}_{i-\frac{1}{2}}(x_i - x_{i-1})) \tag{11}$$

and

$$\tau(\dot{x}_{i+1} - 2\dot{x}_i + \dot{x}_{i-1}) = \left(\widetilde{M}_{i+\frac{1}{2}}(x_{i+1} - x_i) - \widetilde{M}_{i-\frac{1}{2}}(x_i - x_{i-1})\right). \tag{12}$$

Notice that MMPDE4 and MMPDE6 have time scale T_{mesh} for mesh adaption toward equidistribution state [2] such that

$$T_{\text{mesh}} = \mathrm{O}(\tau) \quad (\text{MMPDE4}),$$
$$T_{\text{mesh}} = \mathrm{O}\left(\frac{\tau}{M}\right) \quad (\text{MMPDE6}).$$

If T_{mesh} is much bigger than the natural time scale $\mathrm{O}(t^* - t)$, then the mesh cannot adapt with the behavior of solution. Therefore if $t^* - t \ll \tau$, then MMPDE4 cannot create mesh with enough rate. In comparison of MMPDE4, consider MMPDE6 [1,2,9] with monitor $M(u) = u^{p-1}$, then

$$\frac{\tau}{M} = \tau(t^* - t) \quad \text{if } p = 2, \tag{13}$$
$$= \frac{\tau}{\beta}(t^* - t) \quad \text{if } p > 2, \tag{14}$$

where $\beta = \frac{1}{p-1}$. Eqs. (13) and (14) demonstrate that time scale is a multiple of τ such that it is less than $t^* - t$. This difference leads to fact that MMPDE6 could obtain solutions when $t \to t^*$.

2.3. Local time stepping refinement

Consider discretization of the problem based on computational coordinates. According to dynamical moving mesh method, this equation should be coupled with one of MMPDEs (9) or (10). Therefore we have an ODE system

$$\begin{cases} \text{discretization of the problem in the computational coordinates} \\ (11) \text{ or } (12). \end{cases} \tag{15}$$

Usually this system is solved with starting value at $t_0 = 0$. Sometimes, we have to start from $t_0 > 0$, thus this difference between $t_0 = 0$ and $t_0 > 0$ leads to the creation of error. In other words, in physical and natural problem, we may have the exact solution at $t_0 > 0$ and may want the solution on the time interval $[0, \mathrm{Tf}]$, where Tf should be bigger than blow-up time. The efficiency of local time stepping refinement is present here such that it does not need to start from $t_0 = 0$ and if the integration starts for all $t_0 < \mathrm{Tf}$, then accurate blow-up time similar to $t_0 = 0$ can be obtained. The details of the Local time stepping refinement method (LTSR) are given below:

Let time steps of the problem have the form

$$\{t_0, t_1, \ldots, t_n\},$$

where $t_n \leqslant \mathrm{Tf}$.

Now, we apply the refinement scheme at each time step, for example, on the first time step $[t_0, t_1]$. Set $\Delta t = \frac{t_1 - t_0}{k_0}$, where $k_0 \in N$ is a constant and depends on the local truncation error and the tolerance

$$t^{0+(k_0-i)\Delta t} = t_0 + (k_0 - i)\Delta t, \quad i = k_0, k_0 - 1, \ldots, 0. \tag{16}$$

Thus the time integration from t_0 to t_1 involves k_0 substep such that

$$t = t_0 \to t = t_0 + \Delta t \to t = t_0 + 2\Delta t \to \cdots \to t_0 + k_0\Delta t = t_1 \tag{17}$$

solves k_0 ODE systems similar (15) to approximate $u(x, t_1)$ and $x(t_1)$. In each ODE system, we need the initial conditions which are obtained by solving the pervious ODE system. In other words, the initial condition of the jth ODE system is the approximation of the $(j-1)$th ODE system at t_j.

Generally, suppose that we are at time level $t = t_n$ and want to move toward $t = t_{n+1}$, similarly, consider $\Delta t = \frac{t_{n+1} - t_n}{k_n}$ such that

$$t^{n+(k_n-i)\Delta t} = t_n + (k_n - i)\Delta t, \quad i = k_n, k_n - 1, \ldots, 0. \tag{18}$$

Values of $u(x, t_{n+1})$ and $x(t_{n+1})$ are obtained by solving k_n ODE system

$$t = t_n \to t = t^{n+\Delta t} \to t = t^{n+2\Delta t} \to \cdots \to t = t^{n+k_n\Delta t} = t_{n+1}. \tag{19}$$

This process continues until $t_{i+1} \leqslant \mathrm{Tf}$.

If we start to integrate the blow-up problem from $t_0 = 0$ and $u(x, t_0) = u_0(x)$, then both solutions of with and without local time stepping refinement (LTSR) methods are the same. Now assume $0 < t_0 < t^*$, then the numerical approximation with initial condition $u(x, t_0) = u_0(x)$ and without LTSR obtained wrong blow-up time, but our experience confirms that we can capture the right blow-up time in this case with LTSR method.

The numerical solution without LTSR for different values of $t_0 \in (0, t^*)$ shows that the blow-up time transfers linearly to the right, meaning

$$T_0' = T_0 + \Delta t_0, \tag{20}$$

where T_0' and T_0 are the blow-up times with initial time t_0 and t_0', respectively, and $\Delta t_0 = t_0' - t_0$.

The LTSR method may be derived as follows:

Algorithm

```
Step 1: Set $x_j = \frac{j}{N}$, $j = 1, 2, \ldots, N$ and obtain $u(x, 0)$ with initial value $x_j$ and set $i = 0$.
Step 2: Set $\Delta t = \frac{t_{i+1} - t_i}{k_i}$, $k_i \in N$ and $s = 1$.
Step 3: Solve ODE system (15) at time level $t = t^{i+s\Delta t}$, the initial value of which in this step is obtained by solving ODE system (15) at $t = t^{i+(s-1)\Delta t}$.
Step 4: Set $u^{i+(s-1)\Delta t} = u^{i+s\Delta t}$, $x^{i+(s-1)\Delta t} = x^{i+s\Delta t}$ and $s = s + 1$.
Step 5: If $s \leqslant k_i$ then go to step 3, or else $i = i + 1$
Step 6: If $t_i \leqslant \mathrm{Tf}$ then go to step 2, or else break.
```

3. Semi-linear parabolic PDE with blow-up

Consider the following blow-up problem:

$$\begin{cases} u_t = u_{xx} + u^p, \quad p > 1, \\ u(0, t) = u(1, t) = 0, \\ u(x, 0) = u_0(x) > 0. \end{cases} \tag{21}$$

It has been shown by several authors [4] that if $u_0(x)$ is sufficiently large, positive and has a single non-degenerate maximum, then there is a blow-up time $t^* < \infty$ and a unique blow-up point x^* such that

$$u(x, t) \to \infty \quad \text{if } t \to t^*, \tag{22}$$

$$u(x, t) \to u(x, t^*) < \infty \quad \text{if } x \neq x^*, t \to t^*. \tag{23}$$

For numerical computation, we transform (21) to the computational coordinates ξ, then obtain the difference equation on uniform mesh. Therefore, (21) is first transformed to quasi-Lagrangian form

$$\dot{u} - \frac{u_\xi}{x_\xi}\dot{x} = \frac{1}{x_\xi}\left(\frac{u_\xi}{x_\xi}\right)_\xi + u^p \tag{24}$$

and by using central finite difference discretization, we have

$$\dot{u}_i - \frac{u_{i+1} - u_{i-1}}{x_{i+1} - x_{i-1}}\dot{x}_i = \frac{2}{x_{i+1} - x_{i-1}}\left(\frac{u_{i+1} - u_i}{x_{i+1} - x_i} - \frac{u_i - u_{i-1}}{x_i - x_{i-1}}\right) + u_i^p, \tag{25}$$

where $u_0 = u_N = 0$ and $i = 1, 2, \ldots, N - 1$.

4. Numerical results

We solve Eq. (21) with $p = 2$ and smoothing technique for monitor function with $i_p = 2$ and $\gamma = 2$. Based on dynamical moving mesh methods, the mesh equation should be coupled with the physical PDE. The resulting coupled systems of nonlinear ODEs which govern the mesh and the solution, are (11) or (12) and (25), then integration in time is done using "ODE15s" in *Matlab 7.0.4*. We set the tolerances of absolute and relative error 10^{-8}. Homogeneous boundary conditions are proposed such that $u_0 = u_N = 0$ and the initial solution profile[Budd's paper] $u(x,0) = 20\sin(\pi x)$ is taken.

The efficiency of the method with four monitor functions and various t_0 is presented. The self similarity of various solution profiles computed over time is most easily determined by comparing directly to the following asymptotic formula derived by [1] such that $(\frac{u}{u_{\max}})^{p-1} \simeq \cos^2(\pi(\xi - 0.5))$ where $u_{\max} := \max u$ for all x represents how far the code is capable of computing into the singularity. Ideally, $u_{\max}$ is as large as possible, plots for $N = 40$ points and τ is selected fixed and equal to 10^{-5}. In particular, our best estimate of blow-up time is $t^* \approx 8.2319 \times 10^{-2}$.

Example. Consider the blow-up equation

$$\begin{cases} u_t = u_{xx} + u^2, \\ u(0,t) = u(1,t) = 0, \\ u(x,0) = 20\sin(\pi x). \end{cases} \tag{26}$$

Applying this method with and without using LTSR method obtains similar results. For investigating the efficiency of LTSR method, consider $t_0 > 0$ and calculate $u(x,t)$ and $x(t)$ of problem (26). Consider $t_0 = 0.02$.

By starting from $t_0 = 0.02$ and according to Table 1, it is demonstrated that the blow-up time changes and a wrong blow-up time is obtained. The variation of blow-up time is satisfied based on (20).

If we start from $t_0 = 0.02$, normally the number of time steps will be reduced in comparison to starting from $t_0 = 0$. Table 2 shows these differences.

According to the use of these monitors, $M(u) = u^{p-1}$ have accurate blow up time with respect to the others, because this monitor is obtained based on the analysis of blow-up problem. In other words, it coincides with the natural behavior of blow-up problem.

Now, we want to show the difference of mesh trajectories for these four monitor functions during time steps, which is selected by ode15s.

With Monitor function $u^{\frac{p-1}{2}}$ proposed in [9], we have high rate for moving toward self similar blow-up solution, but the new monitor $u^{\frac{p-1}{1.5}}$ could capture the $u_{\max}$ bigger than the others (see Table 3). The monitor u^{p-1} which is obtained by analysis of the blow-up problem has a stable behavior from changing t_0 from 0 to 0.02 such that it has the same time steps for both cases listed in Table 2.

Table 1
Using MMPDE6 with four smooth monitor functions ($i_p = 2, \gamma = 2$), t^*_{O} demonstrate the blow-up time without LTSR on $[0, \text{Tf}]$ and t^*_N is the blow-up time which obtain without LTSR derived on $[0.02, \text{Tf}]$

MMPDE	Monitor	t^*_{O}	t^*_{N}
6	$u^{\frac{p-1}{0.5}}$	8.2115×10^{-2}	1.02115×10^{-1}
6	u^{p-1}	8.2318×10^{-2}	1.02318×10^{-1}
6	$u^{\frac{p-1}{1.5}}$	8.2361×10^{-2}	1.02361×10^{-1}
6	$u^{\frac{p-1}{2}}$	8.2376×10^{-2}	1.02376×10^{-1}

Table 2
MMPDE6 with four smooth monitor functions ($i_p = 2, \gamma = 2$), NTS(1) and NTS(2) present number of time steps in ODE15s with $t_0 = 0$ and $t_0 = 0.02$, respectively

MMPDE	Monitor	NTS(1)	NTS(2)
6	$u^{\frac{p-1}{0.5}}$	2889	2886
6	u^{p-1}	2554	2554
6	$u^{\frac{p-1}{1.5}}$	1776	1765
6	$u^{\frac{p-1}{2}}$	1694	1684

Both time steps are obtained without using LTSR method.

Table 3
Using MMPDE6 with four smooth monitor functions ($i_p = 2$), without using LTSR method on time interval [0, Tf]

MMPDE	Monitor	$U_{\max}$	CPU time (s)
6	$u^{\frac{p-1}{0.5}}$	4.43077×10^{13}	182
6	u^{p-1}	2.72471×10^{13}	126
6	$u^{\frac{p-1}{1.5}}$	7.90741×10^{13}	79
6	$u^{\frac{p-1}{2}}$	7.71913×10^{13}	63

Now for a summary on using MMPDE6 (see Fig. 1), the difference of using various monitor functions such as how these functions move toward self similar blow-up is demonstrated. Scaled solution of the blow-up problem with using MMMPDE6 be presented in Fig. 2. We try to carry out these calculations with MMPDE4 such that all computations are used with smoothing monitor functions which are smoothed by (8). Similarly, in this case system, (15) is used with (11) such that $u(x,t)$ and $x(t)$ are the solution of these ODE systems solved by ode15s. Like before, four different monitor functions are applied and the solution with different initial time t_0 is compared to present the efficiency of LTSR method. These numerical results are presented in Tables 4–6.

First consider $t_0 = 0$, in this case for all monitor functions without using LTSR have the same solutions with using LTSR, but with starting from $t_0 = 0.02$ (or for each t_0 that satisfies in $0 < t_0 < t^*$), find out the blow-up time moving 0.02 without using LTSR. By using LTSR method with $t_0 = 0.02$, the accurate blow-up time is obtained as $t_0 = 0$.

Table 5 shows that the number of time steps which are selected by ode15s are roughly decreased when $t_0 = 0.02$ instead of $t_0 = 0$.

The mesh trajectories for different monitor functions versus the number of time steps with MMPDE4 are plotted in Fig. 3.

With comparing these monitor functions, it is natural that they have different $u_{\max}$. The numerical result without LTSR method of $u_{\max}$ and CPU time is presented in Table 6.The results show that the monitor function u^{p-1} has captured the maximum $u_{\max}$ and the least CPU times belong to the $u^{\frac{p-1}{1.5}}$.

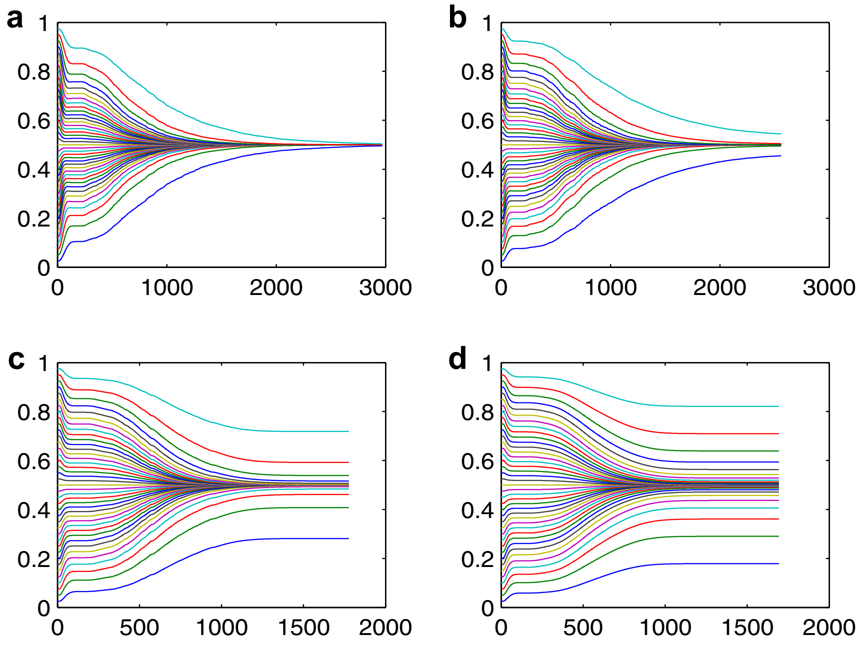

Fig. 1. Mesh trajectories using MMPDE6 with smooth monitor functions such that plot (a) $u^{\frac{p-1}{0.5}}$, (b) u^{p-1}, (c) $u^{\frac{p-1}{1.5}}$ and (d) $u^{\frac{p-1}{2}}$, where $p = 2$.

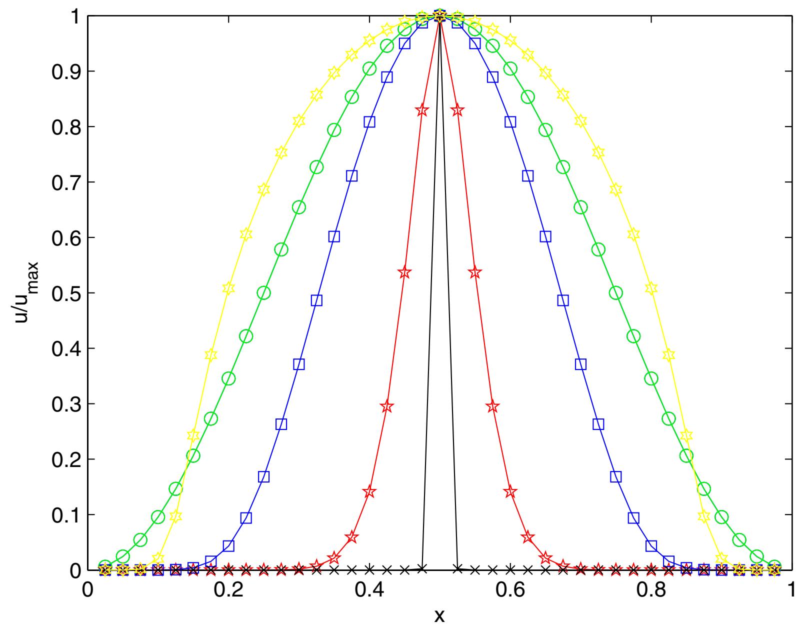

Fig. 2. Scaled solution of the blow-up problem with using MMPDE6 and with four smoothed monitor functions ($i_p = 2, \gamma = 2$) such that $u^{\frac{p-1}{0.5}}$ (☆), asymptotic solution (○), u^{p-1} (◇), $u^{\frac{p-1}{1.5}}$ (∗) and $u^{\frac{p-1}{2}}$ (×).

Table 4
Using MMPDE4 with four smooth monitor functions ($i_p = 2$)

MMPDE	Monitor	t^*_{O}	t^*_{N}
4	$u^{\frac{p-1}{0.5}}$	8.214118×10^{-2}	1.0214118×10^{-1}
4	u^{p-1}	8.232483×10^{-2}	1.0232483×10^{-1}
4	$u^{\frac{p-1}{1.5}}$	8.236359×10^{-2}	1.0236359×10^{-1}
4	$u^{\frac{p-1}{2}}$	8.237755×10^{-2}	1.0237755×10^{-1}

t^*_{O} demonstrates the blow-up time without LTSR and t^*_{N} is the blow-up which is obtained without LTSR derived on $[0.02, Tf]$.

Table 5
Using MMPDE4 with four smooth monitor functions ($i_p = 2$), NTS(1) present ode15s time steps with $t_0 = 0$ and NTS(2) demonstrate ode15s time steps with $t_0 = 0.02$

MMPDE	Monitor	NTS(1)	NTS(2)
4	$u^{\frac{p-1}{0.5}}$	1490	1475
4	u^{p-1}	8301	8301
4	$u^{\frac{p-1}{1.5}}$	1365	1351
4	$u^{\frac{p-1}{2}}$	1129	1107

Both ode15s time steps are obtained without using LTSR method.

Table 6
Using MMPDE4 with four smooth monitor functions ($i_p = 2$),without using LTSR method on time interval [0, Tf]

MMPDE	Monitor	$U_{\max}$	CPU time
4	$u^{\frac{p-1}{0.5}}$	5.914900×10^{9}	123
4	u^{p-1}	9.440564×10^{10}	1098
4	$u^{\frac{p-1}{1.5}}$	5.080135×10^{9}	80
4	$u^{\frac{p-1}{2}}$	2.388994×10^{10}	105

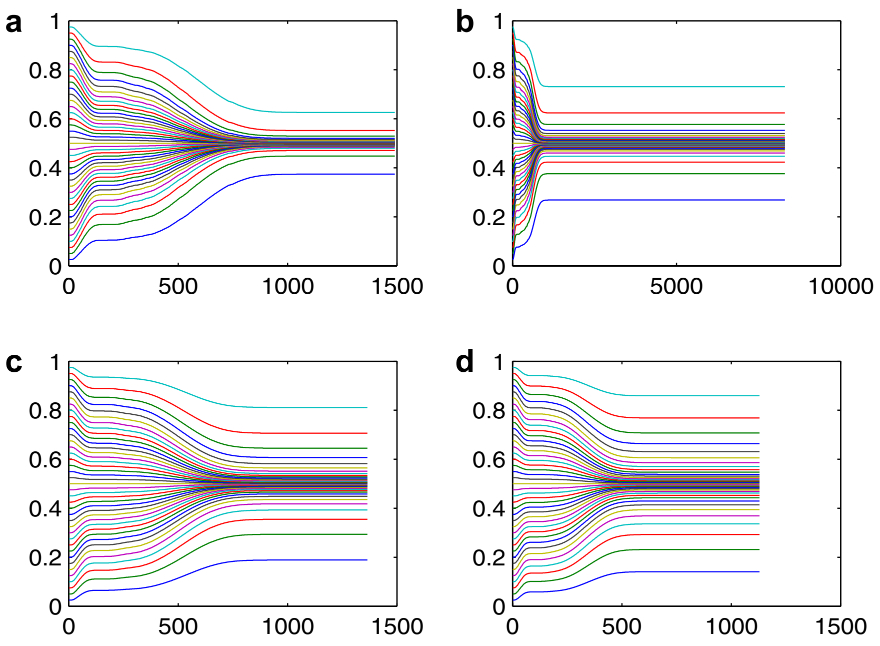

Fig. 3. The mesh trajectories with using MMPDE4 and four smooth monitor functions versus the number of time steps in (a) $u^{\frac{p-1}{0.5}}$, (b) u^{p-1}, (c) $u^{\frac{p-1}{1.5}}$ corresponds to $u^{\frac{p-1}{2}}$ (d).

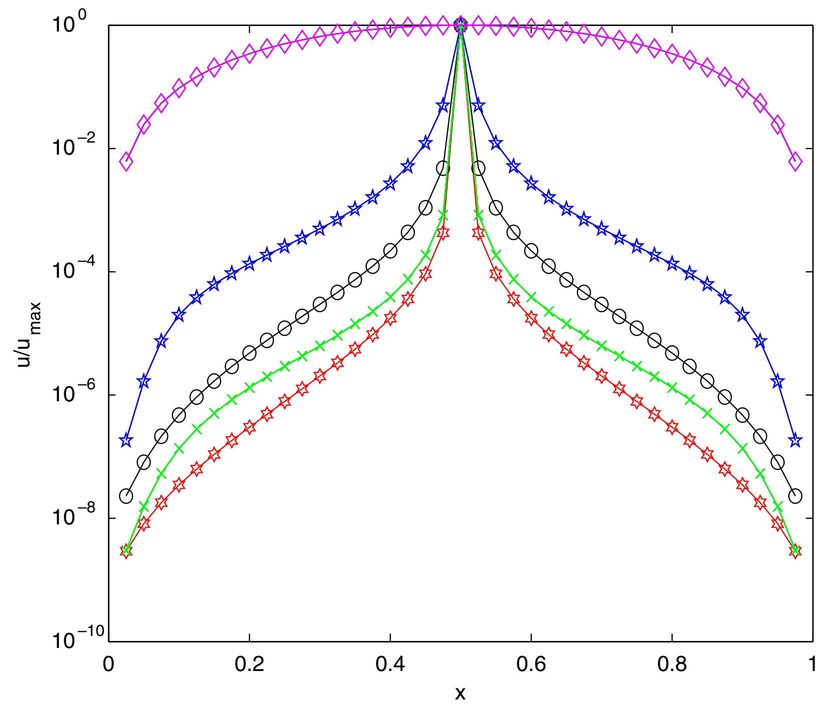

Fig. 4. Scaled solution of the blow-up problem with using MMPDE4 and with four smoothed monitor functions, asymptotic solution (◇), $u^{\frac{p-1}{0.5}}$ (☆), u^{p-1} (○), $u^{\frac{p-1}{1.5}}$ (×) and $u^{\frac{p-1}{2}}$(✱).

Fig. 4 plots the scaled approximation $u/u_{\max}$ versus the physical coordinate x of the blow-up problem with MMPDE4 for four different monitor functions. The top curve is the asymptotic solution.

References

[1] C.J. Budd, W. Huang, R.D. Russell, Moving mesh methods for problems with blow up, SIAM J. Comput. 17 (1996) 305–327.

[2] C.J. Budd, R. Carretero-González, R.D. Russell, Precise computations of chemotactic collapse using moving mesh methods, J. Comput. Phys. 202 (2) (2005) 463–487.

[3] E.A. Dorfi, L.o'c. Drury, Simple adaptive grids for 1-D initial value problems, J. Comput. Phys. 69 (1987) 175–195.

[4] A. Friedman, B. Mcleod, Blow up of positive solutions of semilinear heat equations, Indiana Univ. Math. J. 34 (1985) 425–447.

[5] W. Huang, Y. Ren, R.D. Russell, Moving mesh partial differential equations(MMPDEs) base on the equidistribution principle, SIAM J. Number. Anal. 31 (1994) 709–730.

[6] W. Huang, Y. Ren, R.D. Russell, Moving mesh methods based upon moving mesh partial differential equations, J. Comput. Phys. 113 (1994) 279–290.

[7] S. Li, L.R. Petzold, Y. Ren, Stability of moving mesh systems of partial differential equations, SIAM J. Sci. Comput. 20 (2) (1999) 719–738.

[8] S. Osher, R. Sanders, Numerical approximation to nonlinear conversation laws with locally varying time and space grids, Math. Comp. 41 (1983) 321–336.

[9] A.R. Soheili, J.M. Stockie, A moving mesh method with variable mesh relaxation time, Applied Numerical Mathematics, in press, doi:10.1016/j.apnum.2006.11.014.

[10] J.M. Stockie, J.A. Mackenzie, R.D. Russell, A moving mesh method for one-dimensional hyperbolic conservation laws, SIAM J. Sci. Comput. 22 (5) (2001) 1791–1813.

[11] Z. Tan, Z. Zhang, Y. Huang, T. Tang, Moving mesh methods with locally varying time steps, J. Comp. Phys. 200 (2004) 347–367.

[12] J.G. Verwer, J.G. Blom, R.M. Furzeland, P.A. Zegeling, A moving grid method for one-dimensional PDEs based on the method of lines, in: J.E. Flaherty et al. (Eds.), Adaptive Method for Partial Differential Equations, SIAM, philadelphia, 1989.