Journal of Computational Physics 228 (2009) 6977–6990

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Moving mesh methods for blowup in reaction–diffusion equations with traveling heat source ☆

Jingtang Ma [a], Yingjun Jiang [b,*]

[a] School of Economic Mathematics, Southwestern University of Finance and Economics, Chengdu 611130, China
[b] Department of Mathematics and Scientific Computing, Changsha University of Science and Technology, Changsha 410076, China

ARTICLE INFO

Article history:
Received 18 January 2009
Received in revised form 30 May 2009
Accepted 15 June 2009
Available online 21 June 2009

MSC:
65N50
65M50
35K55
35K57

Keywords:
Blowup
Reaction–diffusion equations
Moving mesh methods

ABSTRACT

In this paper moving mesh methods are used to simulate the blowup in a reaction–diffusion equation with traveling heat source. The finite-time blowup occurs if the speed of the movement of the heat source remains sufficiently low, and the blowup procedure is not fixed at one point not like that for stationary heat source. As time goes to the blowup time, the blowup profile converges to a stationary state. In the simulation a new moving mesh algorithm is designed to deal with the difficulty caused by the delta function in the traveling heat source. The convergence rates are verified and new blowup figures are generated from the numerical experiments.

1. Introduction

When a localized energy source such as a laser passes over the surface of a material that has combustible properties, there is the possibility of a thermal blowup. Such problems can be modeled mathematically by a nonlinear parabolic equation with traveling heat source (see in [10]). It is well-known that under certain conditions on the nonlinear functions and initial value, the reaction–diffusion equations with stationary heat source have finite-time blowup solutions (see e.g. in [1]). Under the similar conditions and with the heat source traveling, the blowup still can occur at sufficiently low traveling speed and the blowup can be prevented if the heat source travels at high enough speed (see e.g. in [13,8–10]). This is understandable in the sense that the heat source is moving through a medium at a high enough speed – the heat source is continually being exposed to relatively cool surroundings, this thereby provides an enhanced ability to diffuse the supplied heat.

The aim of this paper is to develop a moving mesh algorithm for simulation of blowup in this class of reaction–diffusion equations with traveling nonlinear heat source. In particular the following 1D problem is focused

$$u_t - u_{xx} = \delta(x - x_0)F(u(x_0, t)), \quad -\infty < x < \infty, \ t > 0, \tag{1}$$

☆ J. Ma's research was supported by a grant from the "project 211 (phase III)" of Southwestern University of Finance and Economics (Grant No. 211QN09014). Y. Jiang's research was supported by Scientific Research Fund of Hunan Province Education Department (06C095).

* Corresponding author.
E-mail addresses: mjt@swufe.edu.cn (J. Ma), jiangyingjun@csust.edu.cn (Y. Jiang).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.06.008

$$u(x,0) = u_0(x), \quad -\infty < x < \infty, \tag{2}$$

$$u(x,t) \to 0 \quad \text{as } |x| \to \infty. \tag{3}$$

Here $u(x,t)$ denotes the local temperature of the medium. The initial temperature $u_0(x)$ is taken to be continuous with $u_0(x) \to 0$ as $|x| \to \infty$. The position of the heat source x_0 is given by $x_0 = x_0(t)$ and the speed by $x_0'(t)$. Moreover, we assume $x_0(t)$ is sufficiently smooth with initial state $x_0(0) = 0$. The nonlinear source function $F(x)$ is smooth and has the properties:

$$F(x) > 0,\ F'(x) > 0, \quad F''(x) > 0 \quad \text{for } x > 0.$$

It is proved by Kirk and Olmstead [9] (cf. [13]) that if the speed of the movement for the heat source remains sufficiently low, e.g.

$$|x_0'(t)| \leqslant \kappa,$$

where κ is a sufficiently small constant, then there exists a time $T > 0$ such that the blowup occurs at $t = T$ which means that $\lim_{t\to T} \max\{u(x,t)\} \to +\infty$.

Moving mesh methods have been of much success in the simulation of blowup in reaction–diffusion equations with stationary nonlinear heat source. The first paper on that is given by Budd et al. [4]. The moving mesh is generated by the moving mesh partial differential equations (MMPDEs). The discretization of the physical equations and the MMPDE in space gives an ODE system which is solved by an integration solver – DDASSL (see e.g. in [14]). The monitor function, which plays a key role in generating moving meshes, is determined by the scaling invariance. Later Huang et al. [6] prove that the scaling invariance is neither sufficient nor necessary, instead, the so-called dominance of equidistribution is sufficient to make the moving mesh method work satisfactorily. More recently, Ma et al. [12] develop a moving mesh algorithm for simulation of blowup in nonlocal reaction–diffusion equations. The readers are referred to e.g. a book [18] for the extended references on moving mesh methods and the applications.

In this paper a moving mesh algorithm is developed to simulate the blowup with traveling heat source. The blowup profile is traveling with the heat source and converges to a steady state as time approaching to the blowup time (see e.g. [8]). The major difficulty in constructing the moving mesh algorithm is that the delta function in the right-hand side of Eq. (1) incurs discontinuities to the solution derivatives when crossing the time-dependent curve $x_0(t)$. Motivated by the idea for solving delta functions on fixed mesh (see e.g. in [16]), we develop accurate approximate schemes to the terms in the transformed equations. Combining the approximations into the equation, we then obtain a second-order numerical scheme for the equation. The similar ideas have been used in [11,3] for elliptic equations with fixed localized curve.

Using a time-dependent coordinate transformation $x(\xi,t)$, we may write

$$\dot{u} \equiv \left.\frac{\partial u(x(\xi,t),t)}{\partial t}\right|_{\xi \text{ fixed}} = u_t + u_x\dot{x},$$

and thus rewrite the original equation (1) into the following form:

$$\dot{u} - u_x\dot{x} - u_{xx} = \delta(x - x_0)F(u(x_0,t)).$$

Notice that when $x \neq x_0$, the right-hand side of the above equation vanishes. Thus we may conduct discretizations on equation

$$\dot{u} - u_x\dot{x} - u_{xx} = 0.$$

Each term in the above equation contains a jump when crossing the curve $x_0(t)$. This prevents a high accuracy. Therefore, a smooth auxiliary function is constructed with the information of jumps to obtain accurate approximations. In consequence an accurate moving mesh algorithm is developed. The time-dependent coordinate transformation function $x(\xi,t)$ satisfies the MMPDEs (see e.g. in [7]) which are derived by an equidistribution principle. Among the list of MMPDEs, due to the ease of implementation, the labeled MMPDE4, MMPDE5 and MMPDE6 are popular to use. In this paper, the MMPDE6 is chosen for the computation.

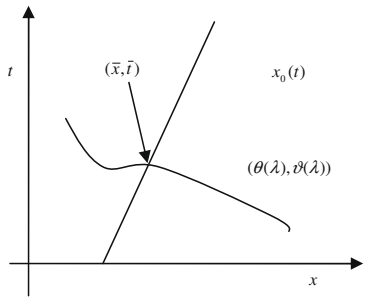

Fig. 1. Illustration figure for definition of jumps.

Here we introduce some notations which will be used throughout the paper. Assume $\varphi : (-\infty, \infty) \times [0, \infty) \to R$ such that $\varphi(x, t)$ is Lipschitz continuous on both sides of $x_0(t)$ (not include $x_0(t)$). Given that a smooth curve $(\theta(\lambda), \vartheta(\lambda))$ crosses $x_0(t)$ at $(\bar{x}, \bar{t})$ when λ increases (see Fig. 1) and $(\theta(\bar{\lambda}), \vartheta(\bar{\lambda})) = (\bar{x}, \bar{t})$ for a $\bar{\lambda}$, we define the jump of φ at $(\bar{x}, \bar{t})$ by

$$[\varphi]_{(\bar{x},\bar{t})} = \lim_{\varepsilon \to 0^+} (\varphi(\theta(\bar{\lambda} + \epsilon), \vartheta(\bar{\lambda} + \epsilon)) - \varphi(\theta(\bar{\lambda} - \epsilon), \vartheta(\bar{\lambda} - \epsilon))).$$

For a single-variable function $g(x)$, we define its jump when x crosses η by

$${}_x[g]_\eta = \lim_{\varepsilon \to 0^+} (g(\eta + \epsilon) - g(\eta - \epsilon)).$$

Without danger of confusion, we abbreviate ${}_x[g]_\eta$ by $[g]_\eta$.

In next section the new moving mesh algorithm is derived. In Section 3, a number of numerical experiments are carried out and figures are generated. Conclusions are given in the final section.

2. Moving mesh algorithm

In this section the moving mesh algorithm is carefully derived. To facilitate a better exposition, we first construct accurate approximations to the derivatives of functions with finite jumps, and then develop the approach into the moving mesh algorithm.

2.1. Approximation scheme

Assume that $g(x)$ is a continuous function in $[a, b]$ and it satisfies that $g(x) \in C^3[a, \eta]$ and $g(x) \in C^3[\eta, b]$, where $\eta \in (a, b)$. For a fixed mesh on $[a, b]$

$$a = x_0 < x_1 < \cdots < x_n = b,$$

with mesh size $h_i = x_i - x_{i-1}$, $(i = 1, \ldots, n)$, We investigate the accurate approximations of $g'(x_i)$ and $g''(x_i)$ which will be discussed by four cases.

Case 1: If $\eta \notin (x_{i-1}, x_{i+1})$, the approximations of $g'(x_i)$ and $g'(x_i)$ are quite standard, i.e. forward scheme

$$g'(x_i) = \frac{g(x_i) - g(x_{i-1})}{h_i} + O(h_i) \tag{4}$$

or backward scheme

$$g'(x_i) = \frac{g(x_{i+1}) - g(x_i)}{h_{i+1}} + O(h_{i+1}) \tag{5}$$

or central scheme

$$g'(x_i) = \frac{g(x_{i+1}) - g(x_{i-1})}{h_i + h_{i+1}} + O(h_{i+1} - h_i) + O\left(h_i^2 + h_{i+1}^2\right), \tag{6}$$

and

$$g''(x_i) = \frac{\frac{g(x_{i+1}) - g(x_i)}{h_{i+1}} - \frac{g(x_i) - g(x_{i-1})}{h_i}}{\frac{h_i + h_{i+1}}{2}} + O(h_{i+1} - h_i) + O\left(h_i^2 + h_{i+1}^2\right). \tag{7}$$

Case 2: If $\eta \in (x_{i-1}, x_i)$, to obtain accurate approximations of $g'(x_i)$ and $g'(x_i)$, we construct an auxiliary $\tilde{g}$ on $[x_{i-1}, x_{i+1}]$ as follows:

$$\tilde{g}(x) = \begin{cases} g(x) + (x - \eta)[g']_\eta + \frac{1}{2}(x - \eta)^2[g'']_\eta + \frac{1}{6}(x - \eta)^3[g''']_\eta, & x \in [x_{i-1}, \eta]; \\ g(x), & x \in (\eta, x_{i+1}]. \end{cases} \tag{8}$$

It is easy to see that $\tilde{g}(x) \in C^3[x_{i-1}, x_{i+1}]$. Thus we have accurate approximations of $\tilde{g}'(x_i)$ and $\tilde{g}''(x_i)$ – (4), (6) and (7) with g replaced by $\tilde{g}$. From the construction of $\tilde{g}$ in (8), we know that

$$\tilde{g}(x_i) = g(x_i), \quad \tilde{g}'(x_i) = g'(x_i), \quad \tilde{g}''(x_i) = g''(x_i), \quad \tilde{g}(x_{i+1}) = g(x_{i+1})$$

and

$$\tilde{g}(x_{i-1}) = g(x_{i-1}) - h_{i,1}[g']_\eta + \frac{h_{i,1}^2}{2}[g'']_\eta - \frac{h_{i,1}^3}{6}[g''']_\eta,$$

where $h_{i,1} = \eta - x_{i-1}$. Put these expressions into the approximate forms of $\tilde{g}'(x_i)$ and $\tilde{g}''(x_i)$ — (4), (6) and (7) with g replaced by $\tilde{g}$ gives that

$$g'(x_i) = \frac{g(x_i) - g(x_{i-1})}{h_i} + \frac{h_{i,1}}{h_i}[g']_\eta + O(h_i) \tag{9}$$

or

$$g'(x_i) = \frac{g(x_{i+1}) - g(x_{i-1})}{h_i + h_{i+1}} + \frac{h_{i,1}}{h_i + h_{i+1}}[g']_\eta + O(h_i + h_{i+1}) + O(h_{i+1} - h_i) + O\left(h_i^2 + h_{i+1}^2\right), \tag{10}$$

and

$$g''(x_i) = \frac{\frac{g(x_{i+1})-g(x_i)}{h_{i+1}} - \frac{g(x_i)-g(x_{i-1})}{h_i}}{\frac{h_i+h_{i+1}}{2}} + \frac{2}{h_i + h_{i+1}}\left(-\frac{h_{i,1}}{h_i}[g']_\eta + \frac{h_{i,1}^2}{2h_i}[g'']_\eta\right) + O(h_i) + O(h_{i+1} - h_i) + O(h_i^2 + h_{i+1}^2). \tag{11}$$

Case 3: For $\eta \in (x_i, x_{i+1})$, we construct the auxiliary function $\tilde{\tilde{g}}$ on $[x_{i-1}, x_{i+1}]$ in the following way:

$$\tilde{\tilde{g}}(x) = \begin{cases} g(x), x \in [x_{i-1}, \eta]; \\ g(x) - (x-\eta)[g']_\eta - \frac{1}{2}(x-\eta)^2[g'']_\eta - \frac{1}{6}(x-\eta)^3[g''']_\eta, x \in [\eta, x_{i+1}]. \end{cases} \tag{12}$$

From the fact that $\tilde{\tilde{g}}(x) \in C^3[x_{i-1}, x_{i+1}]$, we obtain the approximations of $\tilde{\tilde{g}}'(x_i)$ and $\tilde{\tilde{g}}''(x_i)$ – (5)–(7) with g replaced by $\tilde{\tilde{g}}$. The construction of $\tilde{\tilde{g}}$ in (12) gives that

$$\tilde{\tilde{g}}(x_i) = g(x_i), \quad \tilde{\tilde{g}}'(x_i) = g'(x_i), \quad \tilde{\tilde{g}}''(x_i) = g''(x_i), \quad \tilde{\tilde{g}}(x_{i-1}) = g(x_{i-1}),$$

and

$$\tilde{\tilde{g}}(x_{i+1}) = g(x_{i+1}) - h_{i,2}[g']_\eta - \frac{h_{i,2}^2}{2}[g'']_\eta - \frac{h_{i,2}^3}{6}[g''']_\eta,$$

where $h_{i,2} = x_{i+1} - \eta$. Inserting these expressions into the approximate forms of $\tilde{\tilde{g}}'(x_i)$ and $\tilde{\tilde{g}}'(x_i)$ – (5)–(7) with g replaced by $\tilde{\tilde{g}}$, we obtain that

$$g'(x_i) = \frac{g(x_{i+1}) - g(x_i)}{h_{i+1}} - \frac{h_{i,2}}{h_{i+1}}[g']_\eta + O(h_{i+1}) \tag{13}$$

or

$$g'(x_i) = \frac{g(x_{i+1}) - g(x_{i-1})}{h_i + h_{i+1}} - \frac{h_{i,2}}{h_i + h_{i+1}}[g']_\eta + O(h_i + h_{i+1}) + O(h_{i+1} - h_i) + O\left(h_i^2 + h_{i+1}^2\right), \tag{14}$$

and

$$g'(x_i) = \frac{\frac{g(x_{i+1})-g(x_i)}{h_{i+1}} - \frac{g(x_i)-g(x_{i-1})}{h_i}}{\frac{h_i+h_{i+1}}{2}} - \frac{2}{h_i + h_{i+1}}\left(\frac{h_{i,2}}{h_{i+1}}[g']_\eta + \frac{h_{i,2}^2}{2h_{i+1}}[g'']_\eta\right) + O(h_{i+1}) + O(h_{i+1} - h_i) + O\left(h_i^2 + h_{i+1}^2\right). \tag{15}$$

Case 4: For $\eta = x_i$, we consider the approximations of the left and the right derivatives. To do that we only need to simply modify the formulas in Case 2 and Case 3. With the definition of $\tilde{g}(x)$ in (8), we know that $\tilde{g}'(x_i) = g'(x_i+)$ and $\tilde{g}''(x_i) = g'(x_i+)$. Then we have

$$g'(x_i+) = \frac{g(x_i) - g(x_{i-1})}{h_i} + [g']_\eta + O(h_i) \tag{16}$$

or

$$g'(x_i+) = \frac{g(x_{i+1}) - g(x_{i-1})}{h_i + h_{i+1}} + \frac{h_i}{h_i + h_{i+1}}[g']_\eta + O(h_i + h_{i+1}) + O(h_{i+1} - h_i) + O\left(h_i^2 + h_{i+1}^2\right), \tag{17}$$

and

$$g''(x_i+) = \frac{\frac{g(x_{i+1})-g(x_i)}{h_{i+1}} - \frac{g(x_i)-g(x_{i-1})}{h_i}}{\frac{h_i+h_{i+1}}{2}} + \frac{2}{h_i + h_{i+1}}\left(-[g']_\eta + \frac{h_i}{2}[g']_\eta\right) + O(h_i) + O(h_{i+1} - h_i) + O\left(h_i^2 + h_{i+1}^2\right). \tag{18}$$

Similarly the definition of $\tilde{\tilde{g}}(x)$ in (12) gives that $\tilde{\tilde{g}}'(x_i) = g'(x_i-)$ and $\tilde{\tilde{g}}''(x_i) = g''(x_i-)$. Then this leads to

$$g'(x_i-) = \frac{g(x_{i+1}) - g(x_{i-1})}{h_i + h_{i+1}} - \frac{h_{i+1}}{h_i + h_{i+1}}[g']_\eta + O(h_i + h_{i+1}) + O(h_{i+1} - h_i) + O\left(h_i^2 + h_{i+1}^2\right), \tag{19}$$

and

$$g''(x_i-) = \frac{\frac{g(x_{i+1})-g(x_i)}{h_{i+1}} - \frac{g(x_i)-g(x_{i-1})}{h_i}}{\frac{h_i+h_{i+1}}{2}} - \frac{2}{h_i + h_{i+1}}\left([g']_\eta + \frac{h_{i+1}}{2}[g'']_\eta\right) + O(h_{i+1}) + O(h_{i+1} - h_i) + O\left(h_i^2 + h_{i+1}^2\right). \tag{20}$$

2.2. Algorithm

To apply moving mesh methods, we use a time-dependent coordinate transformation to write Eq. (1) into the following form:

$$\dot{u} - u_x\dot{x} - u_{xx} = \delta(x - x_0)F(u(x_0, t)), \tag{21}$$

where $x_0 = x_0(t)$. Since when $x \neq x_0$ the right-hand side vanishes, we conduct discretizations on equation

$$\dot{u} - u_x\dot{x} - u_{xx} = 0. \tag{22}$$

The approximations of terms on the left-hand side of the above equation are based on the information of jumps which can be calculated by the known functions x_0 and F. Thereby the right-hand side of Eq. (21) is actually involved in the numerical scheme.

In the following, we will calculate the jumps of terms $u_t, u_x, \dot{u}$ at $x = x_0(t)$. Since the original equation (1) becomes heat equation when $x \neq x_0$, we conclude that u is continuous in the whole domain and smooth with $x \neq x_0$. While at $(x_0(t), t)$, the jumps are given by

$$[u]_{(x_0(t),t)} = 0, \quad [u_x]_{(x_0(t),t)} = -F(u(x_0(t), t)). \tag{23}$$

The jump of directional derivative of $u(x, t)$ along the vector $(x_0'(t), 1)$ is zero, i.e.

$$[u_x]_{(x_0(t),t)}x_0'(t) + [u_t]_{(x_0(t),t)} = 0. \tag{24}$$

This gives that

$$[u_t]_{(x_0(t),t)} = -x_0'(t)[u_x]_{(x_0(t),t)}.$$

Combining with (1) and (23), we obtain that

$$[u_{xx}]_{(x_0(t),t)} = [u_t]_{(x_0(t),t)} = x_0'(t)F(u(x_0(t), t)), \tag{25}$$

and for the function $\dot{u}(x, t)$ of (x, t),

$$[\dot{u}]_{(x_0(t),t)} = [u_t + u_x\dot{x}]_{(x_0(t),t)} = [u_t]_{(x_0(t),t)} + \dot{x}[u_x]_{(x_0(t),t)} = (x_0'(t) - \dot{x})F(u(x_0(t), t)). \tag{26}$$

Remark 2.1. Let $\bar{t}$ satisfy that $x(\bar{t}) = x_0(\bar{t})$. Then we have

$$_t[\dot{u}]_{\bar{t}} = \sigma[\dot{u}]_{(x_0(\bar{t}),\bar{t})}. \tag{27}$$

where $_t[\dot{u}]_{\bar{t}}$ is the jump of $\dot{u}(x(t), t)$ (single-variable function) at $\bar{t}$, and $[\dot{u}]_{(x_0(\bar{t}),\bar{t})}$ is the jump of $\dot{u}(x, t)$ (two-variables function) at $(x_0(\bar{t}), \bar{t})$;

$$\sigma = \begin{cases} 1, & x(t) \text{ crosses } x_0(t) \text{ from left-hand side to the right as } t \text{ increasing (Fig.2 (Left));} \\ -1, & x(t) \text{ crosses } x_0(t) \text{ from right-hand side to the left as } t \text{ increasing (Fig.2 (Right)).} \end{cases}$$

This relation (27) will be used in the discussions of Case 4 and Case 5 from below.

Now we are in a position to describe the moving mesh algorithm. The moving meshes $\{x_j^n\}$ for all available spatial mesh node index j and temporal index n are obtained by solving the MMPDE6 (see e.g. in [7])

$$\frac{\partial^2\dot{x}}{\partial\xi^2} = -\frac{1}{\tau}\frac{\partial}{\partial\xi}\left(M\frac{\partial x}{\partial\xi}\right), \tag{28}$$

where M is the monitor function. The selection of monitor function is generally problem-dependent. For simulation of blow-up, it is simple and efficient to utilize polynomial-type monitor function, see e.g. in [4,6,12]. As mentioned in the introduction, only if the moving speed of the source term is sufficiently slow, the blowup can occur. For our problems, we particularly design monitors functions to make the moving meshes adapt with the solutions and the moving source as well. More precisely, we utilize monitor function of the form

$$M(x, t) = \alpha u^p + (1 - \alpha)((x - x_0(t))^2 + \epsilon)^{-1/4}$$

for simulation of blowup in our problem and use the gradient-based monitor function

$$M(x, t) = \alpha\left|\frac{\partial u}{\partial x}\right| + (1 - \alpha)((x - x_0(t))^2 + \epsilon)^{-1/4}$$

for non-blowup case. The choice of the constants $p > 0, 0 < \epsilon \ll 1, 0 < \alpha < 1$ will be discussed in the concrete examples in Section 3.

The moving mesh algorithm for solving the transformed equation (21) is based on the approximations of the involved terms on moving meshes, which are described by the following cases. The underlying ideas are given in the aforementioned section.

Case 1 (see Fig. 3 (the 1st one)): If $x_0(t)$ does not pass through $\left(x_{j-1}^{n+1}, x_{j+1}^{n+1}\right)$ and $\left(x_j^n, x_j^{n+1}\right)$, the approximations are given as follows. The approximation of $\dot{u}\left(x_j^{n+1}, t_{n+1}\right)$ is defined by

$$\dot{u}(x_j^{n+1}, t_{n+1}) \approx \frac{u_j^{n+1} - u_j^n}{t_{n+1} - t_n}, \tag{29}$$

where $\{t_n\}$, $(n = 0, 1, \ldots)$ are temporal meshes and $u_j^n \approx u(x_j^n, t_n)$. The approximation of $u_x(x_j^{n+1}, t_{n+1})$ is provided by

$$u_x\left(x_j^{n+1}, t_{n+1}\right) \approx \frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{h_j^{n+1} + h_{j+1}^{n+1}}, \tag{30}$$

where

$$h_j^n = x_j^n - x_{j-1}^n.$$

The approximation of $u_{xx}(x_j^{n+1}, t_{n+1})$ is given by

$$u_{xx}\left(x_j^{n+1}, t_{n+1}\right) \approx \frac{\frac{u_{j+1}^{n+1} - u_j^{n+1}}{h_{j+1}^{n+1}} - \frac{u_j^{n+1} - u_{j-1}^{n+1}}{h_j^{n+1}}}{\frac{h_j^{n+1} + h_{j+1}^{n+1}}{2}}. \tag{31}$$

Case 2 (see Fig. 3 (the 2nd one)): If $x_0(t)$ passes through $\left(x_{j-1}^{n+1}, x_j^{n+1}\right)$, then $u_x\left(x_j^{n+1}, t_{n+1}\right)$ and $u_{xx}\left(x_j^{n+1}, t_{n+1}\right)$ are approximated by

$$u_x\left(x_j^{n+1}, t_{n+1}\right) \approx \frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{h_j^{n+1} + h_{j+1}^{n+1}} + \frac{h_{j,1}^{n+1}}{h_j^{n+1} + h_{j+1}^{n+1}}[u_x]_{(x_0(t_{n+1}), t_{n+1})} = \frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{h_j^{n+1} + h_{j+1}^{n+1}} + \frac{h_{j,1}^{n+1}}{h_j^{n+1} + h_{j+1}^{n+1}}(-F(u(x_0(t_{n+1}), t_{n+1}))), \tag{32}$$

where

$$h_{j,1}^{n+1} = x_0(t_{n+1}) - x_{j-1}^{n+1},$$

and

$$\begin{aligned} u_{xx}\left(x_j^{n+1}, t_{n+1}\right) &\approx \frac{\frac{u_{j+1}^{n+1} - u_j^{n+1}}{h_{j+1}^{n+1}} - \frac{u_j^{n+1} - u_{j-1}^{n+1}}{h_j^{n+1}}}{\frac{h_j^{n+1} + h_{j+1}^{n+1}}{2}} + \frac{2}{h_j^{n+1} + h_{j+1}^{n+1}}\left(-\frac{h_{j,1}^{n+1}}{h_j^{n+1}}[u_x]_{(x_0(t_{n+1}), t_{n+1})} + \frac{\left(h_{j,1}^{n+1}\right)^2}{2h_j^{n+1}}[u_{xx}]_{(x_0(t_{n+1}), t_{n+1})}\right) \\ &= \frac{\frac{u_{j+1}^{n+1} - u_j^{n+1}}{h_{j+1}^{n+1}} - \frac{u_j^{n+1} - u_{j-1}^{n+1}}{h_j^{n+1}}}{\frac{h_j^{n+1} + h_{j+1}^{n+1}}{2}} + \frac{2}{h_j^{n+1} + h_{j+1}^{n+1}}\left(-\frac{h_{j,1}^{n+1}}{h_j^{n+1}}(-F(u(x_0(t_{n+1}), t_{n+1}))) + \frac{\left(h_{j,1}^{n+1}\right)^2}{2h_j^{n+1}}(x_0'(t_{n+1})F(u(x_0(t_{n+1}), t_{n+1})))\right). \end{aligned} \tag{33}$$

Since term $F(u(x_0(t_{n+1}), t_{n+1}))$ in the approximate forms (32) and (33) contains an unknown function value $u(x_0(t_{n+1}), t_{n+1})$, linear interpolation is used to evaluate it –

$$u(x_0(t_{n+1}), t_{n+1}) \approx \frac{x_j^{n+1} - x_0(t_{n+1})}{x_j^{n+1} - x_{j-1}^{n+1}} u_{j-1}^{n+1} + \frac{x_0(t_{n+1}) - x_{j-1}^{n+1}}{x_j^{n+1} - x_{j-1}^{n+1}} u_j^{n+1} \equiv \tilde{u}(x_0(t_{n+1}), t_{n+1}). \tag{34}$$

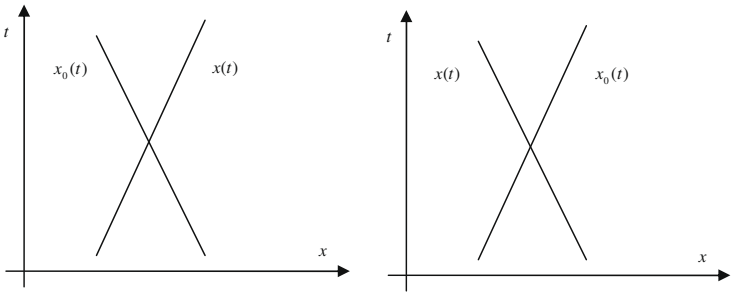

Fig. 2. Left figure: $x(t)$ crosses $x_0(t)$ from left-hand side to the right as t increasing; right figure: $x(t)$ crosses $x_0(t)$ from right-hand side to the left as t increasing.

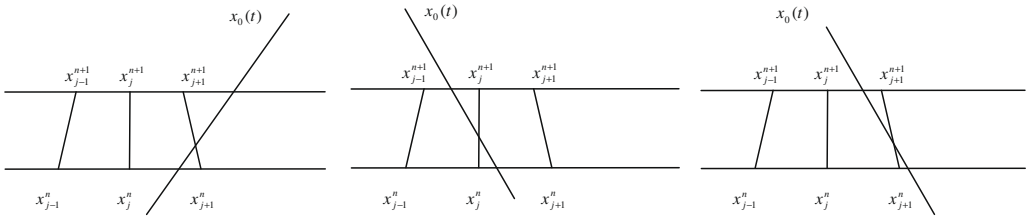

Fig. 3. Illustration figure for Cases 1–3 (from left to right).

Therefore, the final forms of approximations are

$$u_x\left(x_j^{n+1},t_{n+1}\right)\approx\frac{u_{j+1}^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}+\frac{h_{j,1}^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}(-F(\tilde{u}(x_0(t_{n+1}),t_{n+1}))),$$

and

$$u_{xx}\left(x_j^{n+1},t_{n+1}\right)\approx\frac{\frac{u_{j+1}^{n+1}-u_j^{n+1}}{h_{j+1}^{n+1}}-\frac{u_j^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}}}{\frac{h_j^{n+1}+h_{j+1}^{n+1}}{2}}+\frac{2}{h_j^{n+1}+h_{j+1}^{n+1}}\left(-\frac{h_{j,1}^{n+1}}{h_j^{n+1}}(-F(\tilde{u}(x_0(t_{n+1}),t_{n+1})))+\frac{(h_{j,1}^{n+1})^2}{2h_j^{n+1}}(x_0'(t_{n+1})F(\tilde{u}(x_0(t_{n+1}),t_{n+1})))\right). \tag{35}$$

Case 3 (see Fig. 3 (the 3rd one)): In the case $x_0(t)$ passes through $\left(x_j^{n+1},x_{j+1}^{n+1}\right)$, let

$$u(x_0(t_{n+1}),t_{n+1})\approx\frac{x_{j+1}^{n+1}-x_0(t_{n+1})}{x_{j+1}^{n+1}-x_j^{n+1}}u_j^{n+1}+\frac{x_0(t_{n+1})-x_j^{n+1}}{x_{j+1}^{n+1}-x_j^{n+1}}u_{j+1}^{n+1}\equiv\tilde{\tilde{u}}(x_0(t_{n+1}),t_{n+1}). \tag{36}$$

Then the approximations of $u_x\left(x_j^{n+1},t_{n+1}\right)$ and $u_{xx}\left(x_j^{n+1},t_{n+1}\right)$ are given by

$$u_x\left(x_j^{n+1},t_{n+1}\right)\approx\frac{u_{j+1}^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}-\frac{h_{j,2}^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}[u_x]_{(x_0(t_{n+1}),t_{n+1})}\approx\frac{u_{j+1}^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}-\frac{h_{j,2}^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}(-F(\tilde{\tilde{u}}(x_0(t_{n+1}),t_{n+1}))), \tag{37}$$

where $h_{j,2}^{n+1}=x_{j+1}^{n+1}-x_0(t_{n+1})$, and

$$\begin{aligned}u_{xx}\left(x_j^{n+1},t_{n+1}\right)&\approx\frac{\frac{u_{j+1}^{n+1}-u_j^{n+1}}{h_{j+1}^{n+1}}-\frac{u_j^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}}}{\frac{h_j^{n+1}+h_{j+1}^{n+1}}{2}}-\frac{2}{h_j^{n+1}+h_{j+1}^{n+1}}\left(\frac{h_{j,2}^{n+1}}{h_j^{n+1}}[u_x]_{(x_0(t_{n+1}),t_{n+1})}+\frac{(h_{j,2}^{n+1})^2}{2h_j^{n+1}}[u_{xx}]_{(x_0(t_{n+1}),t_{n+1})}\right)\\&\approx\frac{\frac{u_{j+1}^{n+1}-u_j^{n+1}}{h_{j+1}^{n+1}}-\frac{u_j^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}}}{\frac{h_j^{n+1}+h_{j+1}^{n+1}}{2}}-\frac{2}{h_j^{n+1}+h_{j+1}^{n+1}}\left(\frac{h_{j,2}^{n+1}}{h_j^{n+1}}(-F(\tilde{\tilde{u}}(x_0(t_{n+1}),t_{n+1})))+\frac{(h_{j,2}^{n+1})^2}{2h_j^{n+1}}(x_0'(t_{n+1})F(\tilde{\tilde{u}}(x_0(t_{n+1}),t_{n+1})))\right).\end{aligned} \tag{38}$$

Case 4 (see Fig. 4): Assume that $x_0(t)$ intersects with $\left(x_j^n,x_j^{n+1}\right)$ at $t=\bar{t}$. Then

$$\begin{aligned}\dot{u}\left(x_j^{n+1},t_{n+1}\right)&=\frac{u(x_j^{n+1},t_{n+1})-u(x_j^n,t_n)}{t_{n+1}-t_n}+\frac{\bar{t}-t_n}{t_{n+1}-t_{n_t}}[\dot{u}]_{\bar{t}}+O(t_{n+1}-t_n)\\&=\frac{u\left(x_j^{n+1},t_{n+1}\right)-u(x_j^n,t_n)}{t_{n+1}-t_n}+\sigma\frac{\bar{t}-t_n}{t_{n+1}-t_n}[\dot{u}]_{(x_0(\bar{t}),\bar{t})}+O(t_{n+1}-t_n),\end{aligned}$$

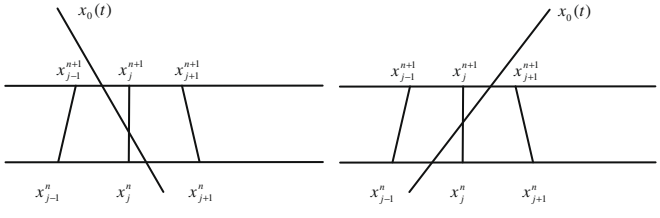

Fig. 4. Illustration figure for Case 4.

where ${}_t[\dot{u}]_{\bar{t}}$ and σ are defined as in Remark 2.1. Therefore, $\dot{u}\left(x_j^{n+1}, t_{n+1}\right)$, with the calculation of jump $[\dot{u}]$ (26), is approximated by

$$\dot{u}\left(x_j^{n+1}, t_{n+1}\right) \approx \frac{u_j^{n+1} - u_j^n}{t_{n+1} - t_n} + \sigma \frac{\bar{t} - t_n}{t_{n+1} - t_n}(x_0'(\bar{t}) - \dot{x}(\bar{t}))F(u(x_0(\bar{t}), \bar{t})). \tag{39}$$

On $[t_n, t_{n+1}], x(t)$ is given by

$$x(t) = \frac{t_{n+1} - t}{t_{n+1} - t_n} x_j^n + \frac{t - t_n}{t_{n+1} - t_n} x_j^{n+1}. \tag{40}$$

Therefore, $\dot{x}(t)$ is computed by

$$\dot{x}(t) = \frac{x_j^{n+1} - x_j^n}{t_{n+1} - t_n}.$$

$u(x_0(\bar{t}), \bar{t})$ is approximated by

$$u(x_0(\bar{t}), \bar{t}) \approx \frac{t_{n+1} - \bar{t}}{t_{n+1} - t_n} u_j^n + \frac{\bar{t} - t_n}{t_{n+1} - t_n} u_j^{n+1} \equiv \bar{u}(x_0(\bar{t}), \bar{t}).$$

Therefore, the final form of approximation is given by

$$\dot{u}\left(x_j^{n+1}, t_{n+1}\right) \approx \frac{u_j^{n+1} - u_j^n}{t_{n+1} - t_n} + \sigma \frac{\bar{t} - t_n}{t_{n+1} - t_n}\left(x_0'(\bar{t}) - \left(\frac{x_j^{n+1} - x_j^n}{t_{n+1} - t_n}\right)\right)F(\bar{u}(x_0(\bar{t}), \bar{t})). \tag{41}$$

Case 5 (see Fig. 5): For the case that $x_0(t_{n+1}) = x_j^{n+1}$, the approximate schemes for the terms in (22) and the moving mesh algorithms are discussed in the following situations – (a) and (b).

(a) For the situation in Fig. 5 (left), we evaluate (22) at $\left(x_j^{n+1}+, t_{n+1}-\right)$. The approximations of the terms have the forms:

$$\dot{u}\left(x_j^{n+1}+, t_{n+1}-\right) \approx \frac{u_j^{n+1} - u_j^n}{t_{n+1} - t_n},$$

$$u_x\left(x_j^{n+1}+, t_{n+1}-\right) \approx \frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{h_j^{n+1} + h_{j+1}^{n+1}} + \frac{h_j^{n+1}}{h_j^{n+1} + h_{j+1}^{n+1}}[u_x]_{(x_0(t_{n+1}), t_{n+1})},$$

$$u_{xx}\left(x_j^{n+1}+, t_{n+1}-\right) \approx \frac{\frac{u_{j+1}^{n+1} - u_j^{n+1}}{h_{j+1}^{n+1}} - \frac{u_j^{n+1} - u_{j-1}^{n+1}}{h_j^{n+1}}}{\frac{h_j^{n+1} + h_{j+1}^{n+1}}{2}} + \frac{2}{h_j^{n+1} + h_{j+1}^{n+1}}\left(-[u_x]_{(x_0(t_{n+1}), t_{n+1})} + \frac{h_j^{n+1}}{2}[u_{xx}]_{(x_0(t_{n+1}), t_{n+1})}\right).$$

Inserting these expressions into (22) we obtain a scheme for computing u_j^{n+1}:

$$\begin{aligned} 0 = {} & \frac{u_j^{n+1} - u_j^n}{t_{n+1} - t_n} - \left(\frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{h_j^{n+1} + h_{j+1}^{n+1}} + \frac{h_j^{n+1}}{h_j^{n+1} + h_{j+1}^{n+1}}[u_x]_{(x_0(t_{n+1}), t_{n+1})}\right)\frac{x_j^{n+1} - x_j^n}{t_{n+1} - t_n} \\ & - \left(\frac{\frac{u_{j+1}^{n+1} - u_j^{n+1}}{h_{j+1}^{n+1}} - \frac{u_j^{n+1} - u_{j-1}^{n+1}}{h_j^{n+1}}}{\frac{h_j^{n+1} + h_{j+1}^{n+1}}{2}} + \frac{2}{h_j^{n+1} + h_{j+1}^{n+1}}\left(-[u_x]_{(x_0(t_{n+1}), t_{n+1})} + \frac{h_j^{n+1}}{2}[u_{xx}]_{(x_0(t_{n+1}), t_{n+1})}\right)\right). \end{aligned} \tag{42}$$

Alternatively we can evaluate (22) at $\left(x_j^{n+1}-, t_{n+1}+\right)$. The approximations are given by

$$\dot{u}\left(x_j^{n+1}-, t_{n+1}+\right) \approx \frac{u_j^{n+1} - u_j^n}{t_{n+1} - t_n} + {}_t[\dot{u}]_{t_{n+1}} = \frac{u_j^{n+1} - u_j^n}{t_{n+1} - t_n} - [\dot{u}]_{(x_0(t_{n+1}), t_{n+1})},$$

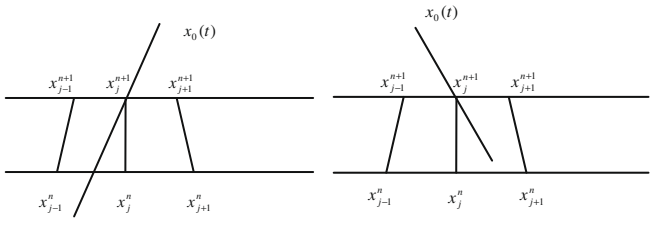

Fig. 5. Illustration figure for Case 5.

$$u_x\left(x_j^{n+1}-,t_{n+1}+\right)\approx\frac{u_{j+1}^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}-\frac{h_{j+1}^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}[u_x]_{(x_0(t_{n+1}),t_{n+1})},$$

$$u_{xx}\left(x_j^{n+1}-,t_{n+1}+\right)\approx\frac{\frac{u_{j+1}^{n+1}-u_j^{n+1}}{h_{j+1}^{n+1}}-\frac{u_j^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}}}{\frac{h_j^{n+1}+h_{j+1}^{n+1}}{2}}-\frac{2}{h_j^{n+1}+h_{j+1}^{n+1}}\left([u_x]_{(x_0(t_{n+1}),t_{n+1})}+\frac{h_{j+1}^{n+1}}{2}[u_{xx}]_{(x_0(t_{n+1}),t_{n+1})}\right).$$

Inserting these expressions into (22) yields another scheme for computing u_j^{n+1}:

$$\begin{aligned}0=&\frac{u_j^{n+1}-u_j^n}{t_{n+1}-t_n}-[\dot{u}]_{(x_0(t_{n+1}),t_{n+1})}-\left(\frac{u_{j+1}^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}-\frac{h_{j+1}^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}[u_x]_{(x_0(t_{n+1}),t_{n+1})}\right)\frac{x_j^{n+1}-x_j^n}{t_{n+1}-t_n}\\&-\left(\frac{\frac{u_{j+1}^{n+1}-u_j^{n+1}}{h_{j+1}^{n+1}}-\frac{u_j^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}}}{\frac{h_j^{n+1}+h_{j+1}^{n+1}}{2}}-\frac{2}{h_j^{n+1}+h_{j+1}^{n+1}}\left([u_x]_{(x_0(t_{n+1}),t_{n+1})}+\frac{h_{j+1}^{n+1}}{2}[u_{xx}]_{(x_0(t_{n+1}),t_{n+1})}\right)\right).\end{aligned}\tag{43}$$

Using (40) and subtracting the right-hand side of (43) from that of (42) give that

$$[\dot{u}]-[u_x]\dot{x}-[u_{xx}]=[u_t]-[u_{xx}]=0,$$

where we have used (25) in the second equality. Hence we can use either (42) or (43) to compute u_j^{n+1}. Therefore, using the following equalities:

$$[u_x]=-F(u(x_0(t_{n+1}),t_{n+1}))=-F\left(u(x_j^{n+1},t_{n+1})\right)\approx-F\left(u_j^{n+1}\right),$$

and

$$[u_{xx}]=x_0'(t_{n+1})F(u(x_0(t_{n+1}),t_{n+1}))=x_0'(t_{n+1})F\left(u(x_j^{n+1},t_{n+1})\right)\approx x_0'(t_{n+1})F\left(u_j^{n+1}\right),$$

into (42) yields the final scheme for Case 5 (Fig. 5 (left)) –

$$\begin{aligned}0=&\frac{u_j^{n+1}-u_j^n}{t_{n+1}-t_n}-\left(\frac{u_{j+1}^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}+\frac{h_j^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}\left(-F\left(u_j^{n+1}\right)\right)\right)\frac{x_j^{n+1}-x_j^n}{t_{n+1}-t_n}-\frac{\frac{u_{j+1}^{n+1}-u_j^{n+1}}{h_{j+1}^{n+1}}-\frac{u_j^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}}}{\frac{h_j^{n+1}+h_{j+1}^{n+1}}{2}}\\&-\frac{2}{h_j^{n+1}+h_{j+1}^{n+1}}\left(F\left(u_j^{n+1}\right)+\frac{h_j^{n+1}}{2}\left(x_0'(t_{n+1})F\left(u_j^{n+1}\right)\right)\right).\end{aligned}\tag{44}$$

(b) Similarly, for the situation in Fig. 5 (right), we can verify that the moving mesh algorithm based on the approximations at point $\left(x_j^{n+1}-,t_{n+1}-\right)$ is identical to that at point $\left(x_j^{n+1}+,t_{n+1}+\right)$. The moving mesh algorithm for Fig. 5 (right) is thereby given by

$$\begin{aligned}0=&\frac{u_j^{n+1}-u_j^n}{t_{n+1}-t_n}-\left(\frac{u_{j+1}^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}-\frac{h_{j+1}^{n+1}}{h_j^{n+1}+h_{j+1}^{n+1}}\left(-F\left(u_j^{n+1}\right)\right)\right)\frac{x_j^{n+1}-x_j^n}{t_{n+1}-t_n}-\frac{\frac{u_{j+1}^{n+1}-u_j^{n+1}}{h_{j+1}^{n+1}}-\frac{u_j^{n+1}-u_{j-1}^{n+1}}{h_j^{n+1}}}{\frac{h_j^{n+1}+h_{j+1}^{n+1}}{2}}\\&+\frac{2}{h_j^{n+1}+h_{j+1}^{n+1}}\left(-F\left(u_j^{n+1}\right)+\frac{h_{j+1}^{n+1}}{2}\left(x_0'(t_{n+1})F\left(u_j^{n+1}\right)\right)\right).\end{aligned}\tag{45}$$

Now it is ready to present the final form of moving mesh algorithm.

Algorithm 1. Given that the solution and mesh at t_n level – $\{u_j^n,x_j^n\}$, we compute the solution and mesh at t_{n+1} level – $\{u_j^{n+1},x_j^{n+1}\}$ by the following steps.

Step 1: Solve the physical equation (1) at **fixed mesh** $\{x_j^n\}$ over $[t_n,t_{n+1}]$ by a scheme constructed by an analogous approach as discussed from above. The computational solution $\tilde{u}_j^{n+1}$ is used to solve the MMPDE6 (28) in the next step.

Step 2: Solve the MMPDE6 (28) by

$$-\tau\frac{\left(x_{j+1}^{n+1}-2x_j^{n+1}+x_{j-1}^{n+1}\right)-\left(x_{j+1}^n-2x_j^n+x_{j-1}^n\right)}{\Delta t_n}=\frac{M\left(\tilde{u}_j^{n+1}\right)+M\left(\tilde{u}_{j+1}^{n+1}\right)}{2}(x_{j+1}^{n+1}-x_j^{n+1})-\frac{M\left(\tilde{u}_{j-1}^{n+1}\right)+M\left(\tilde{u}_j^{n+1}\right)}{2}\left(x_j^{n+1}-x_{j-1}^{n+1}\right).\tag{46}$$

Step 3: Solve the transformed equation (21) using

$$\text{Approx}(\dot{u}) - \text{Approx}(u_x)\frac{x_j^{n+1} - x_j^n}{t_{n+1} - t_n} - \text{Approx}(u_{xx}) = 0, \tag{47}$$

where the approximations of each terms at point $\left(x_j^{n+1}, t_{n+1}\right)$ have particular forms according to the different cases – Case 1, Case 2, Case 3, Case 4; for Case 5 using (44) or (45).

3. Numerical examples

We will use the first example to examine the convergence rate of Algorithm 1. In the other three examples, we simulate the blowup using Algorithm 1.

Example 3.1. We consider

$$u_t - u_{xx} = \delta(x - x_0)F(u(x_0, t)), \quad -10 \leqslant x \leqslant 10, \quad t > 0, \tag{48}$$

$$u(x, 0) = \begin{cases} \cos^2(\pi x/2), & -1 < x < 1, \\ 0, & -10 \leqslant x \leqslant -1 \text{ or } 1 \leqslant x \leqslant 10, \end{cases} \tag{49}$$

$$u(-10, t) = u(10, t) = 0, \tag{50}$$

with

$$F(u) = 1 + u^2, \quad x \in (-10, 10), \quad x_0(t) = 2t.$$

In the following, we test the convergence properties of Algorithm 1. For time we use the following graded mesh

$$\left\{t_n : t_n = \left(\frac{n}{L}\right)^2, n = 0, 1, \ldots, L\right\},$$

where L is the number of temporal meshes. For space we consider three kinds of spatial meshes – uniform meshes, adaptive graded meshes and MMPDE6 moving meshes.

3.1. Adaptively graded meshes

At t_n, we define a coordinate transformation

$$\xi = \Gamma(x) = \begin{cases} -\sqrt{x_0(t_n) - x}, & -10 \leqslant x < x_0(t_n); \\ \sqrt{x - x_0(t_n)}, & x_0(t_n) \leqslant x < 10. \end{cases}$$

Let

$$\xi_j = \Gamma(-10) + j \times \frac{\Gamma(10) - \Gamma(-10)}{N}, \quad j = 0, 1, \ldots, N.$$

Then the adaptively graded mesh nodes at time t_n are given by

$$x_j^n = \Gamma^{-1}(\xi_j), \quad j = 0, 1, \ldots, N, \tag{51}$$

(see Fig. 6).

Note that the above adaptively graded meshes satisfy the following properties:

1. The spatial meshsize satisfies $\max\{h_j^n, h_{j+1}^n\} = O(1/N^2)$ if $x_0(t_n) \in [x_{j-1}^n, x_{j+1}^n]$.

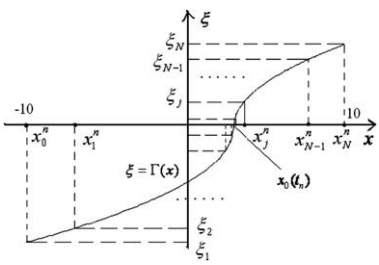

Fig. 6. Adaptively graded mesh.

2. The spatial meshsize satisfies $h_j^n - h_{j-1}^n = O(1/N^2)$.

From the way of constructing our scheme – (47), it is not difficult to see that the truncation error (Trun-Err) is bounded by

$$|\text{Trun-Err}| \leqslant \begin{cases} C\max\left\{h_{j+1}^{n+1} - h_j^{n+1}, (h_{j+1}^{n+1})^2 + (h_j^{n+1})^2, \Delta t\right\}, & \text{if } x_0(t_{n+1}) \notin \left[x_{j-1}^{n+1}, x_{j+1}^{n+1}\right], \\ C\max\left\{h_{j+1}^{n+1} + h_j^{n+1}, \Delta t\right\}, & \text{if } x_0(t_{n+1}) \in \left[x_{j-1}^{n+1}, x_{j+1}^{n+1}\right]. \end{cases}$$

Therefore the truncation error is proportional to $O(\max(1/N^2, \Delta t))$ if the adaptive graded meshes are used, and we thus can expect a second-order convergence for space.

3.2. MMPDE6 moving meshes

The meshes are generated by the MMPDE6 with the parameter $\tau = 10^{-3}$ and monitor function of the form

$$M = \alpha\left|\frac{\partial u}{\partial x}\right| + (1-\alpha)((x - x_0(t))^2 + \epsilon)^{-1/4},$$

with e.g. $\alpha = 0.1$ and $\epsilon = 10^3/N^4$. The user-defined α balances the effects of the gradient of the solution and the interface x_0 to the movement of the meshes. Another value of α might work as well. In particular choosing suitable form of time-dependent α might be more promising. Since the exact solution is not available, to test the convergence, we need to define the measurement of the error. Let $u_{N,L}(x,t_n)$ denote the numerical solution at time t_n with number of space meshes N and number of time meshes L. Our desired convergence rate is second-order, namely,

$$\max_n \|u(x,t_n) - u_{N,L}(x,t_n)\|_\infty \leqslant CN^{-2}.$$

Therefore,

$$e_{N,L} \equiv \max_n \|u_{N,L}(x,t_n) - u_{2N,2L}(x,t_n)\|_\infty \leqslant \max_n \|u(x,t_n) - u_{N,L}(x,t_n)\|_\infty + \max_n \|u(x,t_n) - u_{2N,2L}(x,t_n)\|_\infty \leqslant CN^{-2}, \tag{52}$$

and

$$\frac{e_{2N,4L}}{e_{N,L}} \approx \frac{(2N)^{-2}}{N^{-2}} = 0.25. \tag{53}$$

The computational numerics are listed in Table 1 which confirms that using adaptive meshes is more accurate than using fixed (uniform) meshes and the convergence rate is second-order.

The figures of solutions and mesh trajectories obtained by the use of MMPDE6 moving meshes are plotted in Fig. 7. It is shown that there is no blowup (see Fig. 7). This means that with the traveling speed of the heat source $x_0'(t) = 2$, the heat source is continuously being exposed to relatively cool surroundings and thereby the supplied heat is diffused in large extent, hence the blowup is prevented (see e.g. [13] for the mathematical theory).

Example 3.2. We consider the same problem as in Example 3.1 except that $x_0(t)$ is given by

$$x_0(t) = kt, \quad k = 0, 1.$$

During the solution process, we choose the integration time step $\Delta t_n = t_{n+1} - t_n$ as

$$\Delta t_n = \frac{\mu}{\left(\max\limits_{(j)}\left\{u_j^n\right\}\right)^2}, \tag{54}$$

Table 1
Error and convergence rates.

Uniform meshes						
N, L	25, 25	50, 100	100, 400	200, 1600	400, 6400	800, 25,600
$e_{N,L}$	8.2208e−2	4.4597e−2	3.0513e−2	1.9829e−2	1.1421e−2	–
$e_{2N,4L}/e_{N,L}$	0.54249	0.68418	0.64986	0.57600	–	–
Adaptively graded meshes						
N, L	25, 25	50, 100	100, 400	200, 1600	400, 6400	800, 25,600
$e_{N,L}$	2.8765e−2	6.2424e−3	1.5792e−3	4.0319e−4	1.0127e−4	–
$e_{2N,4L}/e_{N,L}$	0.21701	0.25298	0.25530	0.25483	–	–
MMPDE6 moving meshes						
N, L	25, 25	50, 100	100, 400	200, 1600	400, 6400	800, 25,600
$e_{N,L}$	3.8107e−2	9.0647e−3	2.5573e−3	6.9194e−4	1.7949e−4	–
$e_{2N,4L}/e_{N,L}$	0.23786	0.28212	0.27057	0.25940	–	–

where μ is a small positive constant. (54) is actually a discrete version of Sundman transformation which has been widely used to study blowup problems e.g. in [1,5,12].

In the test, the number of spatial mesh points is $N = 100$; the parameter τ in MMPDE6 is given by $\tau = 5 \times 10^{-4}$ and μ in the time-step formula is set by $\mu = 10^{-3}$. The monitor function is taken as

$$M(x,t) = \alpha u^2 + (1-\alpha)((x - x_0(t))^2 + \epsilon)^{-1/4},$$

with e.g. $\alpha = 0.5$ and $\epsilon = 10^{-5}$. For $k = 0$, namely, $x_0(t) \equiv 0$, the blowup time associated with the maximum value of solution $u_{\max} = 1.1 \times 10^6$ is 1.074623052402. The blowup profiles in both physical and computational variables and the evolving mesh are plotted in Fig. 8. The red line in the mesh figure is the figure for x_0. For $k = 1$, i.e., $x_0(t) = t$, the blowup time for the maximum value of solution $u_{\max} = 2.9 \times 10^6$ is 1.460099559450. The figures are shown in Fig. 9. We can see from the figures that for $x_0(t) \equiv 0$ the blowup occurs at $x = 0$, while for $x_0(t) = t$ at $x = 1.460099559450$. This is consistent with the blowup theory in [13] where blowup occurs at $x^* = x_0(T)$ (T is the blowup time).

Example 3.3. We now consider periodic motion of the heat source which is analyzed in [9]. We simulate the blowup for the problem with the same conditions as in Example 3.2 except that $x_0(t) = A\sin(\pi t)$ with e.g. $A = 1, 2$.

The numerical results are presented in Fig. 10 for $x_0(t) = \sin(\pi t)$ and Fig. 11 for $x_0(t) = 2\sin(\pi t)$. The blowup time for $x_0(t) = \sin(\pi t)$ is 2.104579096179 and for $x_0(t) = 2\sin(\pi t)$ is 4.760652758628. This obviously shows that the blowup time increases with increasing amplitude A which is consistent with the analytical results in [9].

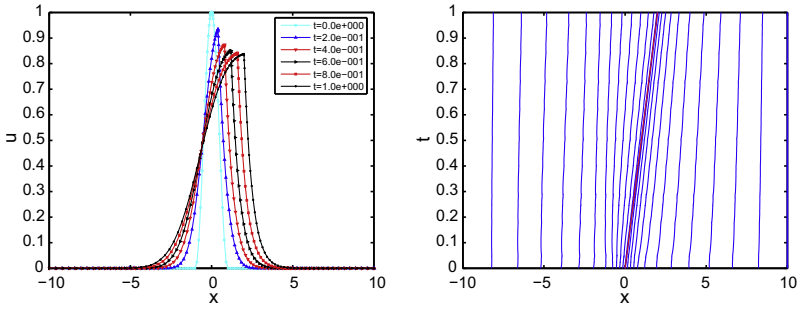

Fig. 7. Figures for **non-blowup** solutions and evolving mesh (from left to right) for Example 3.1 with $x_0(t) = 2t$.

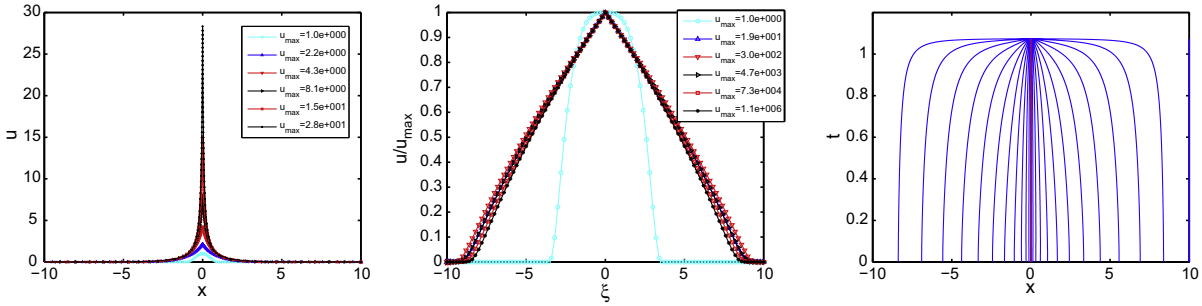

Fig. 8. Figures for blowup profiles in physical variable and computational variable and for evolving mesh (from left to right) for Example 3.2 with $x_0(t) = 0$.

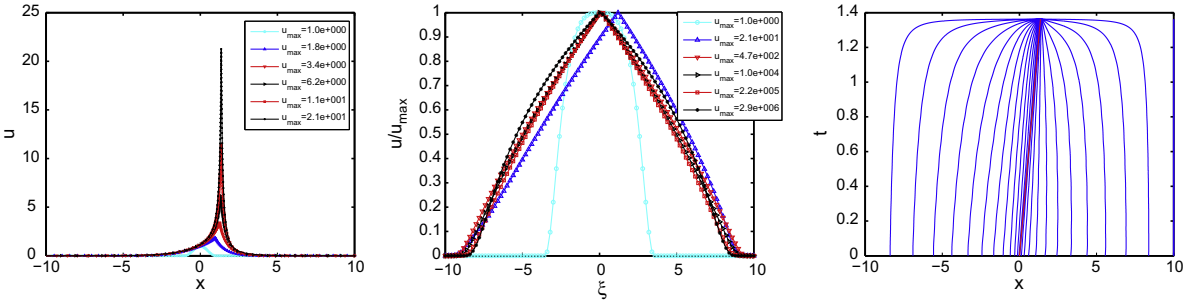

Fig. 9. Figures for blowup profiles in physical variable and computational variable and for evolving mesh (from left to right) for Example 3.2 with $x_0(t) = t$.

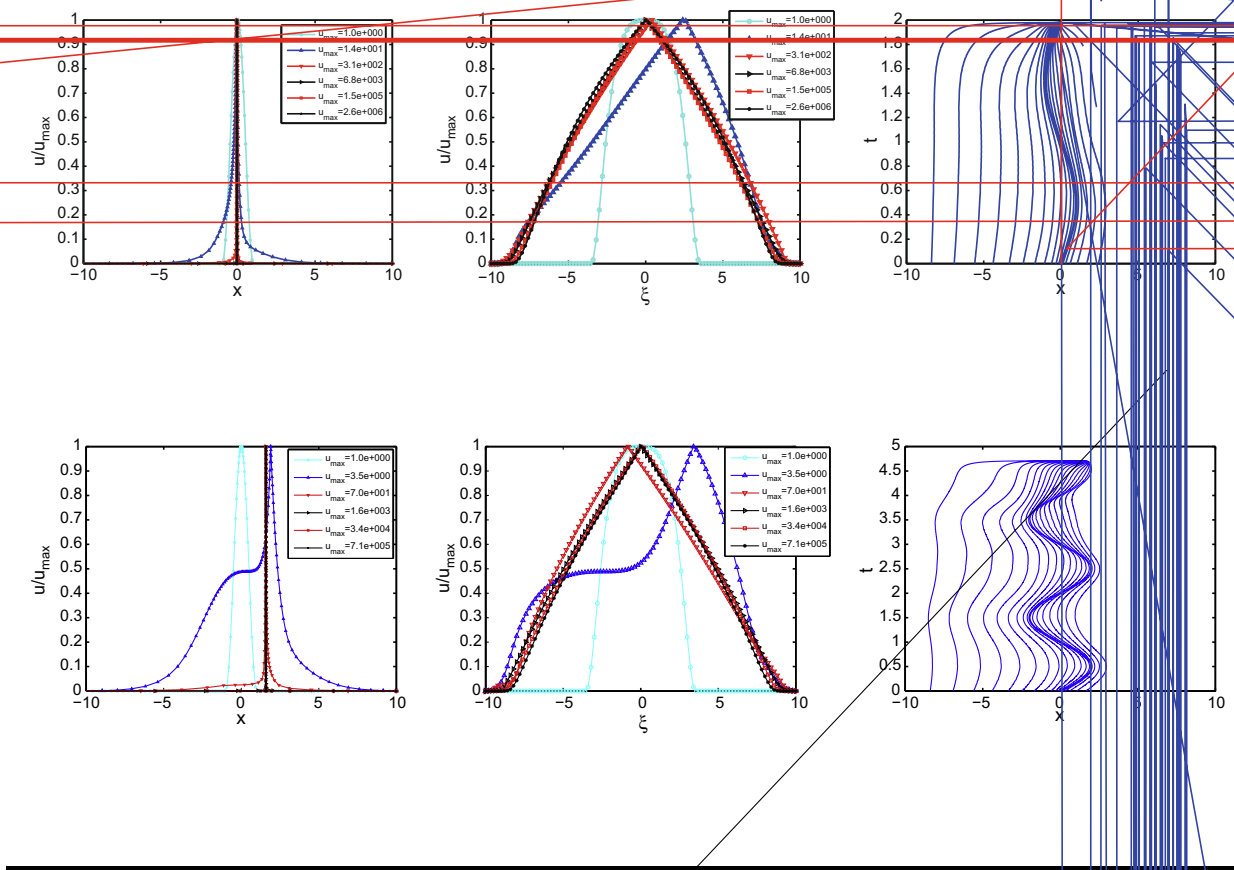

Fig. 12. Figures for blowup profiles in physical variable and computational variable and for evolving mesh (from left to right) for Example 3.4.

Example 3.4. We consider the common problem to the above with rapid deceleration of the motion

$$x_0(t) = \exp(-t).$$

We plot the results in Fig. 12. The blowup time is 1.458583253409 for maximum value of solution $u_{\max} = 8.9 \times 10^5$.

4. Conclusions

In this paper we have studied a reaction–diffusion equation with nonlinear traveling heat source where the blowup occurs with sufficiently low-speed motion for the heat source. We successfully simulated the blowup and non-blowup phenomena using a carefully designed moving mesh algorithm. We generated the interesting figures for blowup profiles which are quite different from those in the literatures (e.g. in [4,2,15,17,6,12]). In the future, we hope to extend the approach to two-dimensional problems (see in [10]) and blowup problems with two moving heat sources (see e.g. in [8]).

Acknowledgment

The authors are grateful to the two anonymous referees for their important and constructive suggestions.

References

[1] C. Bandle, H. Brunner, Blowup in diffusion equations: a survey, J. Comput. Appl. Math. 97 (1998) 3–22.
[2] C.J. Budd, R. Carretero-González, R.D. Russell, Precise computations of chemotactic collapse using moving mesh methods, J. Comput. Phys. 202 (2005) 463–487.
[3] C.J. Budd, H. Huang, R.D. Russell, Mesh selection for a nearly singular boundary value problem, J. Sci. Comput. 16 (2001) 525–552.
[4] C.J. Budd, W. Huang, R.D. Russell, Moving mesh methods for problems with blow-up, SIAM J. Sci. Comput. 17 (1996) 305–327.
[5] C.J. Budd, B. Leimkuhler, M.D. Piggott, Scaling invariance and adaptivity, Appl. Numer. Math. 39 (2001) 261–288.
[6] W. Huang, J. Ma, R.D. Russell, A study of moving mesh PDE methods for numerical simulation of blowup in reaction diffusion equations, J. Comput. Phys. 227 (2008) 6532–6552.
[7] W. Huang, Y. Ren, R.D. Russell, Moving mesh partial differential equations (MMPDEs) based upon the equidistribution principle, SIAM J. Numer. Anal. 31 (1994) 709–730.
[8] C.M. Kirk, W.E. Olmstead, The influence of two moving heat sources on blow-up in a reactive–diffusive medium, Z. Angew. Math. Phys. 51 (2000) 1–16.
[9] C.M. Kirk, W.E. Olmstead, Blow-up in a reactive–diffusive medium with a moving heat source, Z. Angew. Math. Phys. 53 (2002) 147–159.
[10] C.M. Kirk, W.E. Olmstead, Blow-up solutions of the two-dimensional heat equation due to a localized moving source, Anal. Appl. 3 (2005) 1–16.
[11] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (1994) 1019–1044.
[12] J. Ma, Y. Jiang, K. Xiang, Numerical simulation of blowup in nonlocal reaction–diffusion equations using a moving mesh method, J. Comput. Appl. Math. 230 (2009) 8–21.
[13] W.E. Olmstead, Critical speed for the avoidance of blow-up in a reactive–diffusive medium, Z. Angew. Math. Phys. 48 (1997) 701–710.
[14] L.R. Petzold, A descreption of DASSL: a differential/algebraic system solver, Sandia Labs Report SAND82-8637, Livermore, CA, 1982.
[15] R.D. Russell, J.F. Williams, X. Xu, MOVCOL4: a moving mesh code for fourth-order time-dependent partial differential equations, SIAM J. Sci. Comput. 29 (2007) 197–220.
[16] P. Smereka, The numerical approximation of a delta function with application to level set methods, J. Comput. Phys. 211 (2006) 77–90.
[17] A.R. Soheili, J.M. Stockie, A moving mesh method with variable mesh relaxation time, Appl. Numer. Math. 58 (2008) 249–263.
[18] T. Tang, J. Xu, Adaptive Computations: Theory and Algorithms, Science Press, Beijing, 2007.