

# Adaptive Grid Based on Geometric Conservation Law Level Set Method for Time Dependent PDE

Ali R. Soheili, Maryam A. Ameri

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

Received 21 July 2007; accepted 25 February 2008

Published online 27 May 2008 in Wiley InterScience (www.interscience.wiley.com).

DOI 10.1002/num.20360

A new method for mesh generation is formulated based on the level set functions, which are solutions of the standard level set evolution equation with the Cartesian coordinates as initial values (Liao et al. *J Comput Phys* 159 (2000), 103–122; Osher and Sethian *J Comput Phys* 79 (1988), 12; Sethian, *Level set methods and fast marching methods*, Cambridge University Press, 1999; Di et al. *J Sci Comput* 31 (2007), 75–98). The intersection of the level contours of the evolving functions form a new grid at each time. The velocity vector in the evolution equation is chosen according to the Geometric Conservation Law (GCL) method (Cao et al., *SIAM J Sci Comput* 24 (2002), 118–142.). This method has precise control over the Jacobian of transformation because of using the GCL method. © 2008 Wiley Periodicals, Inc. *Numer Methods Partial Differential Eq* 25: 582–597, 2009

*Keywords:* adaptive grid; level set function; level contour; geometric conservation law method

## I. INTRODUCTION

Many partial differential equations (PDEs) arising from science and engineering have a common feature that they have a small portion of the physical domain where small node separation are required to resolve large variations of the solution. Examples include problems having boundary layers, shock waves, the combustion, and heat transfer theory. Numerical solution of these PDEs using a uniform mesh may be formidable when the systems involve more than two spatial dimensions, since the number of mesh points can become very large. On the other hand, to improve efficiency and accuracy of the numerical solution it is natural to consider more mesh points in the region of large variation of solution than the rest of physical domain. With this basic idea of mesh adaptation, the number of mesh points can be much smaller, thus significant economy can be gained [1–4]. Essential topic to mesh adaptation is the ability to control the size, shape and orientation of mesh elements [5]. In the first step, the element information on size, shape and orientation is specified throughout the domain. The specification is often based on an error indicator, an error estimate, or a physical consideration. In the second step, an algorithm is developed for generating the needed adaptive mesh according to the specification of element information. In

*Correspondence to:* Ali R. Soheili, Department of Mathematics, University of Sistan and Baluchestan, Zahedan, Iran (e-mail: soheili@math.usb.ac.ir)

© 2008 Wiley Periodicals, Inc.

this article, we use a general idea of moving grids. We consider a scalar or vector field satisfying

$$u_t(x, t) = L(u), \tag{1}$$

where  $L$  is a differential operator defined on a physical domain  $\Omega$  in  $R^n, n = 1, 2, 3$ . A continuous view for mesh adaptation is to consider an adaptive mesh to generate as image of a computational mesh under a reversible coordinate transformation  $x = x(\xi) : \Omega_c \rightarrow \Omega$ , where  $\Omega_c$  is the computational mesh domain, [4, 6–8]. Assume that an uniform computational mesh is chosen, the size, shape and orientation of mesh elements are then completely determined by  $x = x(\xi, t)$ . Consequently, the control of elements is equivalent to the control of  $x = x(\xi, t)$ . We transform Eq. (1) by  $x = \phi^{-1}(\xi, t)$  and solve the transformed equation on a fixed grid on the computational domain that  $\phi$  is the level set function. It is notable that variational methods [7, 9, 10], define this transformation as the solution of a system of PDEs which controls various aspects of the grid such as orthogonality, smoothness and cell size. The resulting system of PDEs for grid generation is often nonlinear and its solution requires intensive computation. The moving mesh methods based on moving mesh partial differential equations are another type of the adaptive mesh method specially designed for numerical solution of time dependent PDEs. In this method, the mesh nodes are typically moved by a mesh equation, to adapt to the evolutionary features of physical solution. Mesh movement strategies are designed in the guidance of the equidistribution principle. This type of method developed in [3, 10, 11]. Another velocity based moving mesh methods are deformation method [12] and GCL (Geometric Conservation Law) method [2], we will describe the last method in the next section.

**II. GEOMETRIC CONSERVATION LAW LEVEL SET METHOD**

In this section, a new moving grid method is formulated, named the geometric conservation law level set method or GCLSM. The usual evolution equation for the level set functions (as in Ref. [13]) in the cartesian coordinates as initial values is solved. The velocity vector in the evolution equation is chosen according to the GCL method [2]. The intersection points of the level sets of the evolving solutions will form a new grid at each time.

The focus is on the equidistribution principle

$$J\rho = \frac{\sigma}{|\Omega_c|}, \tag{2}$$

where  $J$  is the Jacobian of the coordinate transformation  $x = \phi^{-1}(\xi, t) : \Omega_c \rightarrow \Omega, \rho = \rho(x, t)$  is a given monitor function, and  $\sigma = \sigma(t) = \int_{\Omega} \rho(x, t) dx$ . In fact, in this paper, we construct  $\phi$  such that  $J(\phi) = \frac{\rho|\Omega_c|}{\sigma}$ .

Suppose that the solution of (1) has been computed at time step  $t = t_{k-1}$ . we seek a transformation  $\phi : \Omega \rightarrow \Omega_c$  such that:

$$J(\phi) = \frac{\rho|\Omega_c|}{\sigma},$$

$$\phi(x, t_{k-1}) = \phi_{k-1}(x) \quad x \text{ in } \Omega.$$

First, we describe the method in one dimension. Without loss of generality, we assume both of  $x$  and  $\xi$  lie in interval  $[0, 1]$ .

Suppose that a positive monitor function,  $\rho(x, t)$ , is given, we want to construct a grid of  $N + 1$  nodes in  $[0, 1]$  at each time as following:

$$x_0(t) = 0 < x_1(t) < x_2(t) < \dots < x_i(t) < x_{i+1}(t) < \dots < x_N(t) = 1$$

In one dimension, the equidistribution principle is

$$\frac{\partial x}{\partial \xi} = \frac{\sigma}{\rho|\Omega_c|}, \tag{3}$$

this means,

$$\frac{x_{i+1} - x_i}{\frac{1}{N}} = \frac{\sigma}{\rho} \quad (\text{as } N \rightarrow \infty). \tag{4}$$

We seek a level set function  $\phi$  from  $[0, 1]$  to  $[0, 1]$  which map  $x_i$  to  $k_i = \frac{i}{N}, i = 0, 1, 2, \dots, N$  on  $[0, 1]$  over the  $y$ -axis.

The condition (4) is equivalent to,

$$\frac{\frac{1}{N}}{x_{i+1} - x_i} = \frac{\rho}{\sigma}, \tag{5}$$

namely,

$$\frac{\phi(x_{i+1}) - \phi(x_i)}{x_{i+1} - x_i} = \frac{\rho}{\sigma}, \tag{6}$$

or

$$\frac{\partial \phi}{\partial x} = \frac{\rho}{\sigma}. \tag{7}$$

In 1D, this equation can be solved by direct integration,

$$\phi(x) = \int_0^1 \frac{\rho(s, t)}{\sigma(t)} ds. \tag{8}$$

The preimages of the points  $k_i = \frac{i}{N}$  under the transformation  $x \rightarrow \phi(x, t)$  are the level sets of  $\phi$  and they form new nodes. This idea extends to multiple dimensions as follows, the goal is to generate a nodal mapping  $\Phi$  from  $\Omega \rightarrow \Omega_c$  such that  $J(\Phi) = \frac{\rho|\Omega_c|}{\sigma}$ . To begin, let us define the basic concept of modeling a moving front by the level sets. Suppose that there is a moving front with a velocity field  $\mathbf{v} = (x_t, y_t, z_t)$ , where  $\mathbf{x} = (x, y, z)$  is the position of a particle at time  $t$ . We introduce a smooth function  $\Phi(x, y, z, t)$  with the property that the front is given by the zero level set of  $\Phi$ , i.e.  $\Phi(x, y, z, t) = 0$ . By differentiating with respect to  $t$ , we get  $\Phi_t + \Phi_x x_t + \Phi_y y_t + \Phi_z z_t = 0$ , which can be written as

$$\Phi_t(x, y, z, t) + \langle \nabla \Phi, \mathbf{v} \rangle = 0, \tag{9}$$

and in one dimension converts to

$$\phi_t(x, t) + v\phi_x = 0. \tag{10}$$

This is the evolution equation for level set function  $\phi$ , [14, 15]. In two dimension, we construct two function  $\phi$  and  $\psi$  by (9) with a suitable vector field  $v$ , then the intersections of their level set

curves will be the new nodes. Let  $\phi(x, y, t)$  and  $\psi(x, y, t)$  be solutions of the following evolution equation,

$$\begin{cases} \phi_t(x, y, z, t) + \langle \nabla \phi, \mathbf{v} \rangle = 0, \\ \psi_t(x, y, z, t) + \langle \nabla \psi, \mathbf{v} \rangle = 0. \end{cases} \tag{11}$$

The initial conditions are  $\phi(x, y, 0) = x$ ,  $\psi(x, y, 0) = y$ , respectively. The boundary conditions are

$$\begin{aligned} \phi(0, y, t) = 0, \quad \phi(1, y, t) = 1, \quad \phi(x, 0, t) = \phi(x, 1, t) = x, \\ \psi(x, 0, t) = 0, \quad \psi(x, 1, t) = 1, \quad \psi(0, y, t) = \psi(1, y, t) = y, \end{aligned} \tag{12}$$

then let  $\Phi = (\phi, \psi)$ .

In three dimension, we obtain three functions  $\phi_1, \phi_2, \phi_3$  from the following equations

$$(\phi_i)_t + \langle \nabla \phi_i, \mathbf{v} \rangle = 0, \quad i = 1, 2, 3 \tag{13}$$

with the same type of initial and boundary conditions as in 2D. A suitable vector field  $\mathbf{v}$  can be determined according to the GCL method. Then the intersections of level sets form the nodes of the moving grid.

Now, we describe the method of computing  $\mathbf{v}$ . In the GCL method, the equidistribution condition is

$$J\rho = \frac{\sigma}{|\Omega_c|}, \tag{14}$$

where  $J$  be the Jacobian of the coordinate transformation  $x = x(\xi) : \Omega_c \rightarrow \Omega$ ,  $\rho = \rho(x, t)$  is a given monitor function and  $\sigma = \sigma(t) = \int_{\Omega} \rho(x, t) dx$ .

Let  $K(t)$  be an arbitrary time dependent sub-domain of  $\Omega$  and  $K_c$  be the corresponding fixed sub-domain in  $\Omega_c$  under  $x = x(\xi)$ . Integrating (14) over  $K_c$  yields

$$\int_{K(t)} \frac{\rho}{\sigma} dx = \frac{|K_c|}{|\Omega_c|}. \tag{15}$$

By differentiating with respect to time, we have

$$\frac{d}{dt} \int_{K(t)} \frac{\rho}{\sigma} dx = 0, \tag{16}$$

or

$$\int_{K_t} \frac{\rho}{\sigma} dx + \int_{\partial K(t)} \frac{\rho}{\sigma} \dot{x} n ds = 0, \tag{17}$$

where the mesh speed,  $\dot{x}$ , characterizes the movement of surface  $\partial K(t)$  and  $n$  is the outward normal to the surface. Using the Gauss' theorem, (17) can be written as

$$\int_{K_t} \frac{\rho}{\sigma} dx + \int_{K(t)} \nabla \left( \frac{\rho}{\sigma} \dot{x} \right) dx = 0. \tag{18}$$

It follows from the arbitrariness of  $K(t)$  that

$$\frac{\partial}{\partial t} \left( \frac{\rho}{\sigma} \right) + \nabla \left( \frac{\rho}{\sigma} \dot{x} \right) = 0. \tag{19}$$

This is basically the continuity equation in fluid dynamics, with “flow velocity”  $\dot{x}$  and “fluid density”  $\frac{\rho}{\sigma}$ . It can be considered as a condition for determining the divergence of mesh speed  $\dot{x}$ . Thus the equidistribution condition (19) is insufficient to determine the vector field  $\dot{x}$ . The motivation for finding supplementary conditions is provided by the Helmholtz decomposition theorem for vectors:

A continuous and differentiable vector field can be decomposed into the orthogonal sum of a gradient of a scalar field and the curl of a vector field. Therefore,  $\dot{x}$  can be determined by specifying both its divergence through (19) and its curl. We require  $\dot{x}$  to satisfy

$$\nabla \times w(\dot{x} - v_{\text{ref}}) = 0, \quad (20)$$

where  $w > 0$  is a weight function and  $v_{\text{ref}}$  is a user-specified reference vector field. The requirement (20) implies that there is a potential function  $H$  such that

$$w(\dot{x} - v_{\text{ref}}) = \nabla H, \quad (21)$$

or

$$\dot{x} = \frac{1}{w} \nabla H + v_{\text{ref}}. \quad (22)$$

Inserting (22) into (19) leads to,

$$\nabla \left( \frac{\rho}{w\sigma} \nabla H \right) = -\frac{\partial}{\partial t} \left( \frac{\rho}{\sigma} \right) - \nabla \left( \frac{\rho}{\sigma} v_{\text{ref}} \right) \quad \text{in } \Omega. \quad (23)$$

The boundary condition can be obtained by requiring the mesh points not to move out the domain, i.e.,  $\dot{x}n = 0$  where  $n$  denotes the outward normal to  $\partial\Omega$ . Using (22), we have the boundary condition

$$\frac{\partial H}{\partial n} = -wv_{\text{ref}}n \quad \text{on } \Omega. \quad (24)$$

To summarize, the potential function,  $H$ , is determined by the elliptic equation (23) with the Neumann boundary condition (24).

Now let  $H$  is known, the mesh speed  $\dot{x}$  can be obtained by (22).

There are two more obvious choices for the weight function,  $w = \frac{\rho}{\sigma}$  and  $w = 1$ . The choice of control vector field  $v_{\text{ref}}$  is very dependent to problem. When physical intuition for choosing  $v_{\text{ref}}$  is not available, the best opinion is simply to choose  $v_{\text{ref}} = 0$ . With  $w = \frac{\rho}{\sigma}$ ,  $v_{\text{ref}} = 0$  this method converts to method on [13].

### III. ALGORITHM FOR SOLVING TIME-DEPENDENT PDES

In this section, we describe the procedures of using our method for solving time-dependent PDE (1) which works for any dimension. At first, we describe this algorithm for one-dimensional PDEs precisely and present a brief explanation for multidimensional cases in section (III.B).

#### A. Algorithm for One-Dimensional PDEs

**Step 1.** A monitor function,  $\rho$ , is determined by the solution being calculated, (at  $t = 0$  we determine  $\rho$  by using initial condition for  $u$ ).

**Step 2.** The nodes velocity,  $v$ , is determined by  $v = \frac{1}{w} \frac{\partial H}{\partial x} + v_{\text{ref}}$  where  $H$  satisfies:

$$\frac{\partial}{\partial x} \left( \frac{\rho}{w\sigma} \frac{\partial H}{\partial x} \right) = -\frac{\partial}{\partial t} \left( \frac{\rho}{\sigma} \right) - \frac{\partial}{\partial x} \left( \frac{\rho}{\sigma} v_{\text{ref}} \right),$$

with the Neumann boundary condition:

$$\frac{\partial H}{\partial x} = -wv_{\text{ref}}.$$

**Step 3.** We update  $\phi$  from the following equation,

$$\phi_t(x, t) + \phi_x(x, t).v = 0, \tag{25}$$

with the initial and boundary condition

$$\phi(x, 0) = x, \quad \phi(0, t) = 0, \quad \phi(1, t) = 1. \tag{26}$$

**Step 4.** Any  $\phi(x, t) = a$  has appropriate point on  $x$ -axis that shows new nodes, in fact for finding new nodes in advanced time, we must find inverse of  $\phi$ . Consider

$$\phi(x, t) = a. \tag{27}$$

Differentiating (27) with respect to  $t$ , we get

$$\phi_t(x, t) + \phi_x \dot{x} = 0, \tag{28}$$

by comparing (25) and (28), we get

$$\dot{x} = v, \tag{29}$$

namely, the nodes velocity is equal to  $v$ .

**Step 5.** After finding new nodes, the values of  $u$  should be computed on this time level, let

$$U(\phi(x, t), t) = u(x, t),$$

then

$$U_t = u_x \dot{x} + u_t,$$

where  $u_t = L(u)$  by (1) and  $\dot{x} = v$  by (29). The derivatives existing in  $L(u)$ , such as  $u_x$  and  $u_{xx}$  are also transformed. For example,

$$U_\phi = u_x x_\phi \Rightarrow u_x = \frac{U_\phi}{x_\phi} = \frac{U_\phi}{\frac{\sigma}{\rho|\Omega_c|}} = \frac{\rho|\Omega_c|}{\sigma} U_\phi,$$

$$U_{\phi\phi} = u_{xx} (x_\phi)^2 + u_x x_{\phi\phi} \Rightarrow u_{xx} = \frac{U_{\phi\phi} - u_x x_{\phi\phi}}{(x_\phi)^2} = \left( \frac{\rho|\Omega_c|}{\sigma} \right)^2 (U_{\phi\phi} - u_x x_{\phi\phi}).$$

The higher derivatives can be obtained similarly. The transformed equation for  $U(\phi, t)$  takes the form of

$$U_t = \tilde{L}(U), \tag{30}$$

where  $\tilde{L}$  is a differential operator in  $\phi$ . Finally (30) will be solved on a fixed uniform grid.

**B. Algorithm for Multidimensional PDEs**

The method of solving time dependent PDE(1) for two- and three-dimensional PDEs, are similar to it's one-dimensional case, in such a way at any time level, after determining monitor function,  $\rho$ , in setp1, we obtain the nodes velocity,  $\mathbf{v}$ , by  $\mathbf{v} = \frac{1}{w} \nabla H + v_{ref}$ , where  $H$  satisfies (23), with the Neumann boundary condition (24). Then, in step 3 for updating the level set function  $\Phi$ , we use (11) for 2D problems and (13) for 3D problems with the appropriate initial and boundary conditions which introduced in section (II). As we told already, the intersection of level contours of the level set functions form the new nodes. Finally, after finding the new nodes, we obtain the rates of  $u$  by transforming the PDE to it's Lagrangian form, similar to one-dimensional case. For example, in two-dimensional case, let

$$U(\phi(x, y, t), \psi(x, y, t), t) = u(x, y, t),$$

by differentiating respect to  $t$ , we have

$$U_t = u_x \dot{x} + u_y \dot{y} + u_t,$$

where  $u_t = L(u)$  and  $\dot{\mathbf{x}} = (\dot{x}, \dot{y})^T = \mathbf{v}$ . The derivatives existing in  $L(u)$ , such as  $u_x, u_y, u_{xx}, u_{xy}$  and  $u_{yy}$ , are also transformed. For instance, we can obtain  $u_x$  and  $u_y$ , from the following equations uniquely,

$$\begin{aligned} U_\phi &= u_x x_\phi + u_y y_\phi, \\ U_\psi &= u_x x_\psi + u_y y_\psi, \end{aligned}$$

because,

$$\begin{vmatrix} x_\phi & y_\phi \\ x_\psi & y_\psi \end{vmatrix} = \frac{1}{J(\Phi)} \neq 0$$

The higher derivatives can be obtained similarly.

**IV. NUMERICAL EXPERIMENTS**

In this section, we implement the presented algorithm for one-dimensional problems. For this purpose, we present two numerical examples. These problems are characterized by moving discontinuities, that means the discontinuities move in time, and so the solution at a particular point in space can change very rapidly. The solution of such these problems on a fixed uniform spatial mesh, need the very small time step to receive sufficient accuracy, but using the adaptive mesh for finding the solution of these problems derives improvement on the accuracy and efficiency. We have used the new adaptive mesh for the mentioned problems with the arclength monitor function,  $\rho = \sqrt{1 + \alpha u_x^2}$ . To obtain an accurate and nonoscillatory solution, it is necessary to smoothen the mesh trajectories. In [3], a technique be used which smoothen the node concentration by smoothen the monitor function over all points. They applied a smooth monitor function as following:

$$\tilde{\rho}_i = \frac{\sum_{k=i-ip}^{i+ip} \rho_k \left(\frac{\gamma}{\gamma+1}\right)^{|k-i|}}{\sum_{k=i-ip}^{i+ip} \left(\frac{\gamma}{\gamma+1}\right)^{|k-i|}},$$

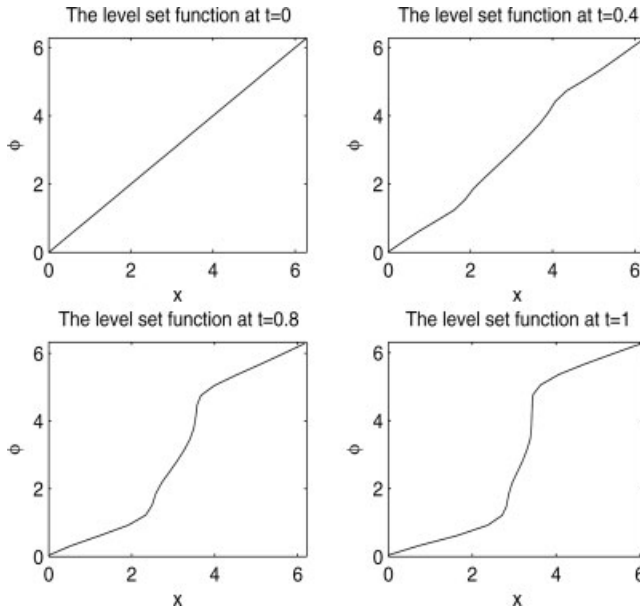


FIG. 1. Level set functions for the first example with  $N = 21$  mesh points at time levels  $t = 0, 0.4, 0.8,$  and  $t = 1$ .

where  $ip$  is a nonnegative integer and  $\gamma$  is a positive constant. In this paper, we select the mentioned formula for smoothing monitor function with  $ip = 1$  and  $\gamma = 1$ . Also, in these examples we choose  $w = \frac{\rho}{\sigma}, v_{ref} = 0$  in step 2.

**Example 1.** This example is the well-known inviscid Burger’s equation, where considered as a test problem in [8, 16].

$$u_t = - \left( \frac{u^2}{2} \right)_x, \quad 0 \leq x \leq 2\pi, \quad t > 0$$

subject to the initial condition,

$$u(x, 0) = 0.5 + \sin(x).$$

We wish to verify the efficiency of the moving mesh algorithm. For this purpose, we solve this example up to  $t = 1$  with our GCLSM algorithm.

In Figure 1,  $\phi$  is plotted along with the nodes of the moving grid by GCLSM with  $N = 21$  mesh points at different times,  $t = 0, 0.4, 0.8,$  and  $t = 1$ . Figure 2 shows mesh trajectories up to  $t = 1$  and the numerical solution at  $t = 1$  with  $N = 21$ .

Similarly, in Figure 3,  $\phi$  is plotted along with the nodes of the moving grid with  $N = 41$  mesh points at  $t = 0, 0.4, 0.8,$  and  $t = 1$ , also Figure 4 shows mesh trajectories up to  $t = 1$  and the computed solution at  $t = 1$  with  $N = 41$ .

**Example 2.** In this example, we implement the present work for the PDE which it’s solution,  $u(x, t)$ , is a given function. We consider the following Burger’s equation,

$$u_t + uu_x - \epsilon u_{xx} = 0, \quad 0 \leq x \leq 2$$



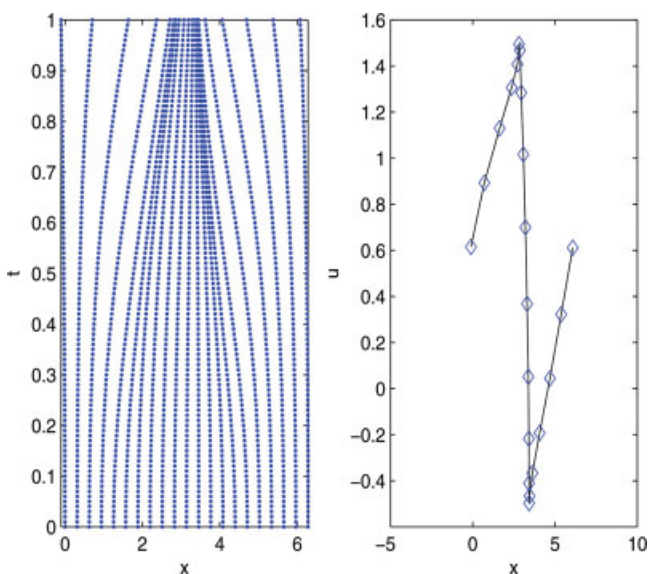


FIG. 2. The mesh trajectories and computed solution with GCLSM of 21 mesh points for Burger's equation in first example. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

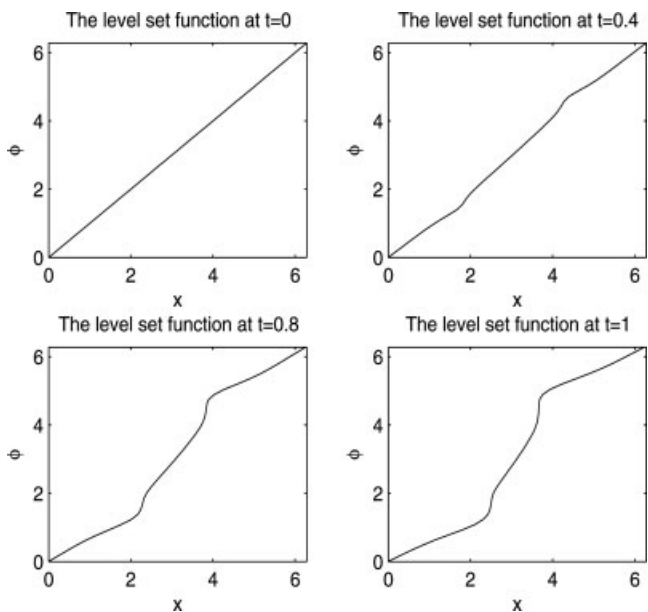


FIG. 3. Level set functions for the first example with  $N = 41$  mesh points at time levels  $t = 0, 0.4, 0.8,$  and  $t = 1$ .

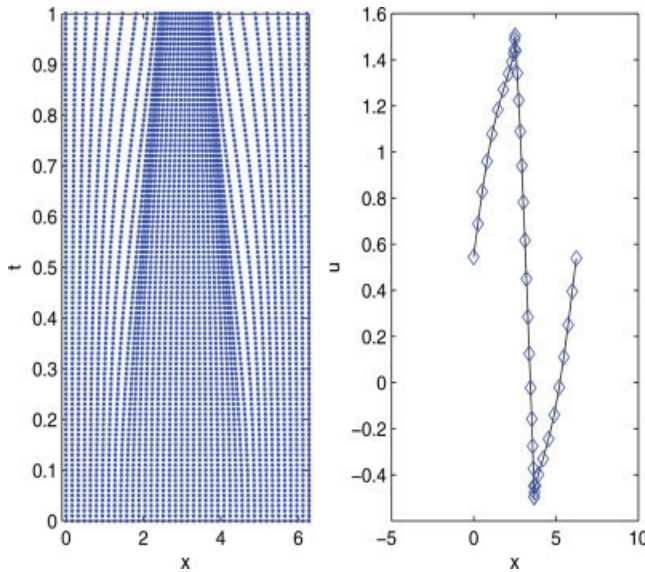


FIG. 4. The mesh trajectories and computed solution with GCLSM of 41 mesh points for Burger’s equation in first example. [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

The initial condition is chosen so that the exact solution is given by,

$$u(x, t) = \frac{\mu + \lambda + (\mu - \lambda)e^{\frac{\lambda}{\epsilon}(x-\mu t)}}{1 + e^{\frac{\lambda}{\epsilon}(x-\mu t)}}.$$

In above equation as  $\epsilon$  increases, the effect of second derivative,  $u_{xx}$ , also increases, subsequently its solution becomes smoother. So for comparison we solve this problem for different values of  $\epsilon$ , also we set  $\lambda = 0.5$  and  $\mu = 0.4$  in our numerical experiments.

In Figure 5, the level set function  $\varphi$  is plotted along with the nodes of moving grid with  $N = 21$  and Figure 6 shows mesh trajectory and the computed solution of PDE,  $u$ , for  $\epsilon = 0.08$ . Similarly, Figures 7–10 show the level set functions, mesh trajectories and the computed solutions of PDE for  $\epsilon = 0.04$  and  $\epsilon = 0.01$ .

We plotted the solution of PDE on a uniform initial mesh (at  $t = 0$ ) and on a moving mesh (at  $t = 0.7$ ) for 21 mesh points in Figures 11, 12, and 13,  $\epsilon$  is considered 0.08, 0.04, and 0.01 respectively. These figures show that the discontinuity move in time and also the new adaptive mesh algorithm has the ability to capture this discontinuity.

In Table I,  $L_\infty$ -error and CPU time are listed for different number of mesh points on fixed mesh and moving mesh, ( $\epsilon = 0.08, 0.04$ , and  $0.01$ ). It is observed, in the smoothest case i.e  $\epsilon = 0.01$ , the GCL calculations with 21 grid points are as accurate as for a fixed mesh of 41 mesh points, in addition the former requires less CPU time. In general, it can be said, the short CPU time in this method makes it acceptable and efficient.

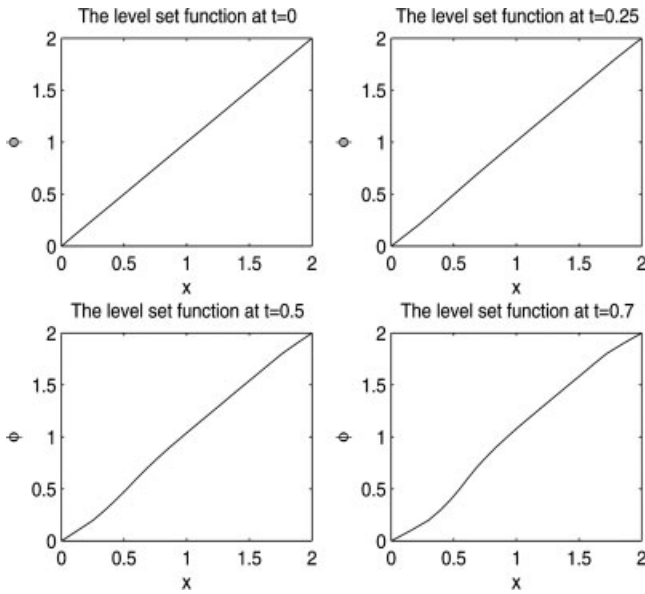


FIG. 5. Level set functions for the second example with  $N = 21$  mesh points at  $t = 0, 0.25, 0.5,$  and  $t = 0.7$  ( $\mu = 0.5, \lambda = 0.4, \varepsilon = 0.08$ ).

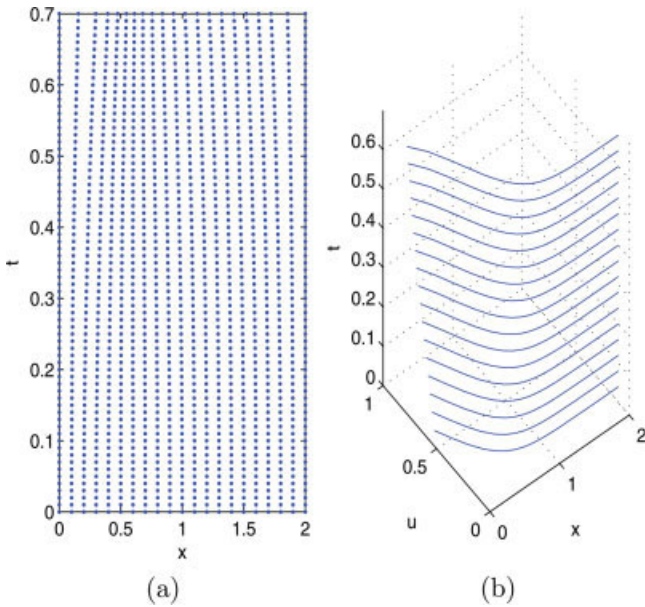


FIG. 6. The mesh trajectory and computed solution of the second example for  $0 \leq t \leq 0.7$  with GCLSM of 21 mesh point ( $\mu = 0.5, \lambda = 0.4, \varepsilon = 0.08$ ). [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

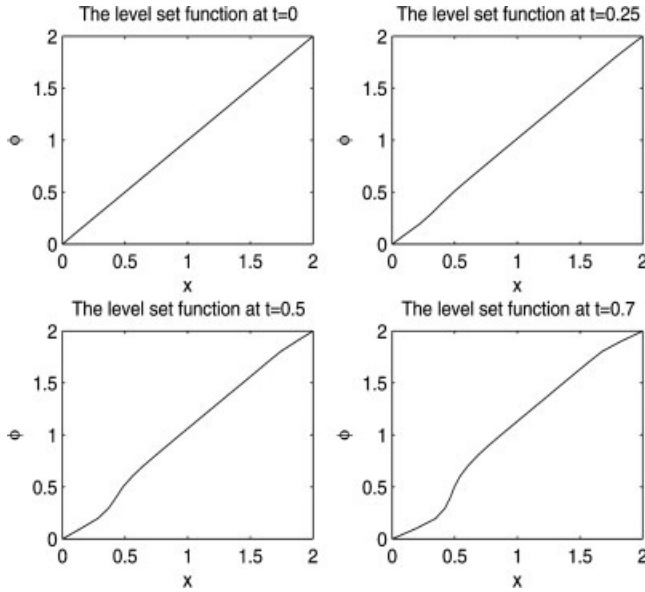


FIG. 7. Level set functions for the second example with  $N = 21$  mesh points at  $t = 0, 0.25, 0.5,$  and  $t = 0.7$  ( $\mu = 0.5, \lambda = 0.4, \varepsilon = 0.04$ ).

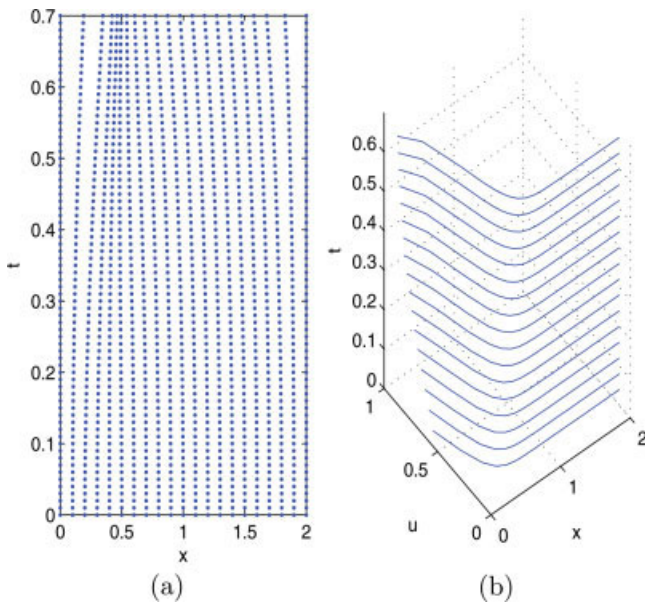


FIG. 8. The mesh trajectory and computed solution of the second example for  $0 \leq t \leq 0.7$  with GCLSM of 21 mesh point ( $\mu = 0.5, \lambda = 0.4, \varepsilon = 0.04$ ). [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

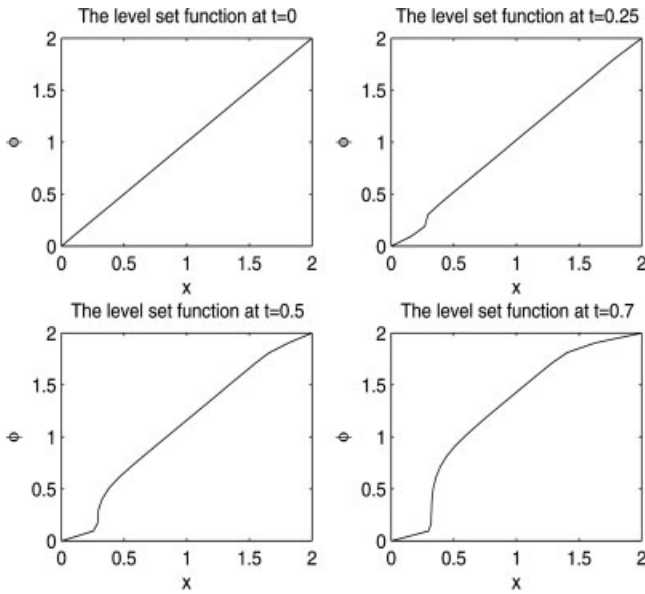


FIG. 9. Level set functions for the second example with  $N = 21$  mesh points at  $t = 0, 0.25, 0.5,$  and  $t = 0.7$  ( $\mu = 0.5, \lambda = 0.4, \varepsilon = 0.01$ ).

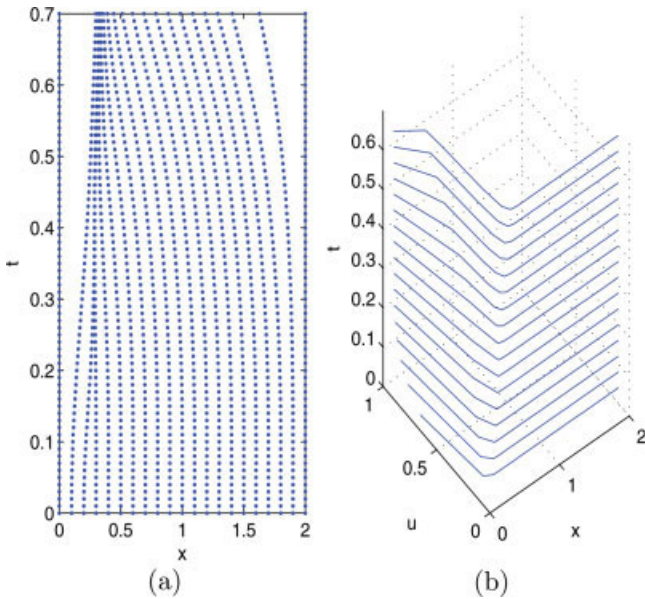


FIG. 10. The mesh trajectory and computed solution of the second example for  $0 \leq t \leq 0.7$  with GCLSM of 21 mesh point ( $\mu = 0.5, \lambda = 0.4, \varepsilon = 0.01$ ). [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

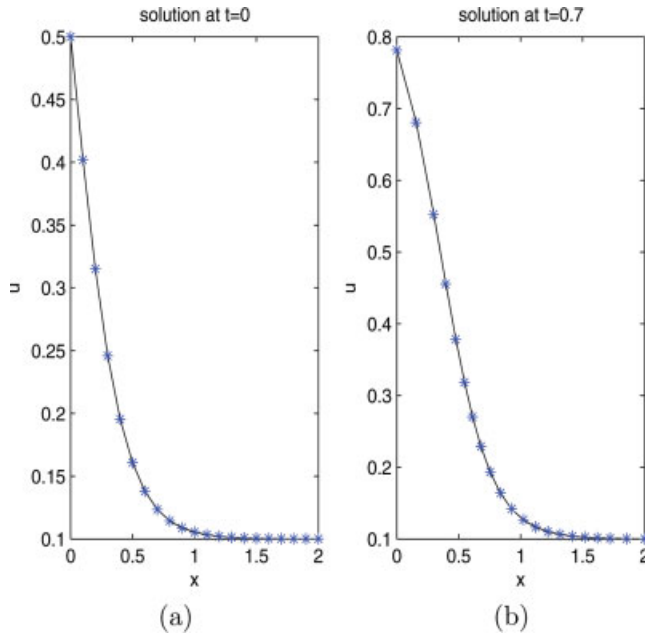


FIG. 11. The computed solution (\*) and exact solution (solid line) at  $t = 0$  on uniform mesh (a) and at  $t = 0.7$  with GCLSM (b) of 21 mesh point for the second example ( $\mu = 0.5, \lambda = 0.4, \varepsilon = 0.08$ ). [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

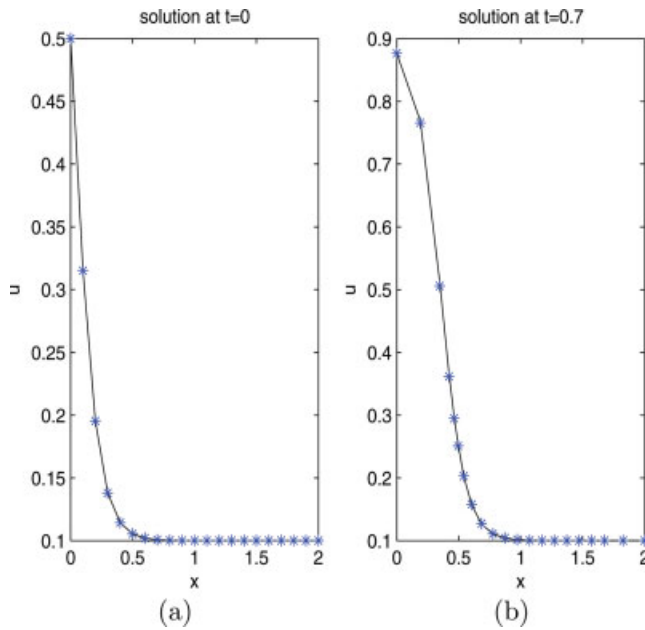


FIG. 12. The computed solution (\*) and exact solution (solid line) at  $t = 0$  on uniform mesh (a) and at  $t = 0.7$  with GCLSM (b) of 21 mesh point for the second example ( $\mu = 0.5, \lambda = 0.4, \varepsilon = 0.04$ ). [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

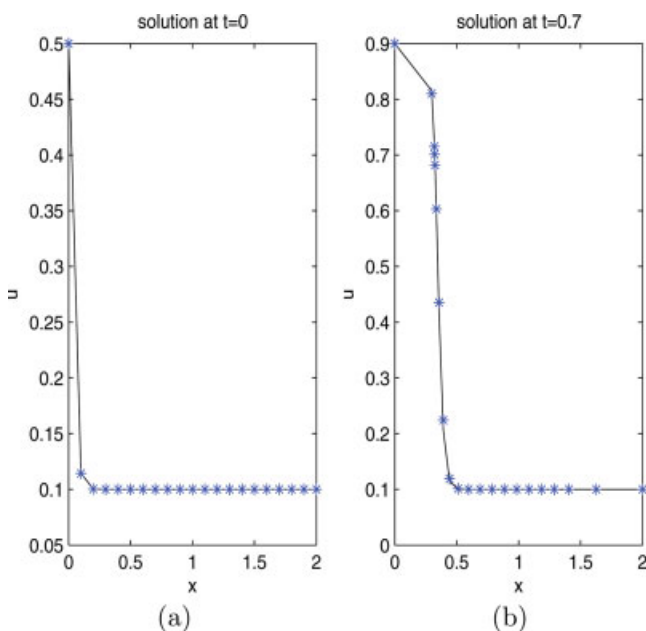


FIG. 13. The computed solution (\*) and exact solution (solid line) at  $t = 0$  on uniform mesh (a) and at  $t = 0.7$  with GCLSM (b) of 21 mesh point for the second example ( $\mu = 0.5, \lambda = 0.4, \varepsilon = 0.01$ ). [Color figure can be viewed in the online issue, which is available at [www.interscience.wiley.com](http://www.interscience.wiley.com).]

TABLE I. The error and the rate of convergence for GCLSM solution of the second example at  $t = 0.7$ , obtained by the arclength monitor function with  $\alpha = 0$ , i.e., uniform mesh and  $\alpha = 2$ , i.e., moving mesh for values of  $\varepsilon = 0.08, 0.04$ , and  $0.01$ .

Case	Description	$N$	$L^\infty$ -error	CPU time
$\varepsilon = 0.08$ $\mu = 0.5$ $\lambda = 0.4$	Fixed mesh ( $\alpha = 0$ )	21	0.0057	0.509
		31	0.0035	0.908
		41	0.0026	0.908
	Moving mesh ( $\alpha = 2$ )	21	0.0023	0.8706
		31	0.0015	0.9844
		41	0.0009	1.029
$\varepsilon = 0.04$ $\mu = 0.5$ $\lambda = 0.4$	Fixed mesh ( $\alpha = 0$ )	21	0.0302	0.512
		31	0.0168	0.705
		41	0.0116	0.912
	Moving mesh ( $\alpha = 2$ )	21	0.0215	0.8743
		31	0.0114	0.9869
		41	0.0101	1.029
$\varepsilon = 0.08$ $\mu = 0.5$ $\lambda = 0.4$	Fixed mesh ( $\alpha = 0$ )	21	0.1916	0.521
		31	0.0574	0.718
		41	0.0501	0.916
	Moving mesh ( $\alpha = 2$ )	21	0.0503	0.8750
		31	0.0391	0.9863
		41	0.0346	1.0313



## V. CONCLUSION

We have introduced a new adaptive mesh method for solving one-dimensional PDEs that is efficient. We have used the level set equation for producing an adaptive mesh, that means for obtaining a moving mesh in any time level, we propagate the level set function with appropriate velocity which obtains by the GCL method. Also we have presented a brief explanation of our method to multidimensional problems, but the most obvious extension of this work is to multidimensional PDEs that we will develop it in details in our next work.

## References

1. G. Bekett, J. A. Mackenzie, and M. L. Robertson, An r-adaptive finite element method for the solution of two-dimensional phase-field equations, *Commun Comput Phys* 1 (2006), 805–826.
2. W. Cao, W. Huang, and R. D. Russell, A moving mesh method based on the geometric conservation law, *SIAM J Sci Comput* 24 (2002), 118–142.
3. W. Huang, Y. Ren, and R. Russell, Moving mesh partial differential equations(MMPDEs) based on the equidistribution principle, *SIAM J Num Anal* 31 (1994), 709–730.
4. R. Li, T. Tang, and P-W. Zhang, Moving mesh methods in multiple dimensions based on harmonic maps, *J Comput Phys* 170 (2001), 562–588.
5. W. Huang, Measuring mesh qualities and application to variational mesh adaptation, *SIAM J Sci Comput* 26 (2005), 57–74.
6. W. Huang, Practical aspects of formulation and solution of moving mesh partial differential equations, *J Comput Phys* 171 (2001), 753–775.
7. W. Huang, Variational mesh adaptation: isotropy and equidistribution, *J Comput Phys* 174 (2001), 903–924.
8. H.-Z. Tang and T. Tang, Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws, *SIAM J Numer Anal* 41 (2003), 487–515.
9. J. M. Sanz-Serna and I. Christie, A simple adaptive technique for nonlinear wave problems, *J Comput Phys* 67 (1986), 348–360.
10. J. M. Stockie, J. A. Mackenzie, and R. D. Russell, A moving mesh method for one dimensional hyperbolic conservation law, *SIAM J Sci Comput* 22 (2001), 1791–1813.
11. A. R. Soheili and J. M. Stockie, A moving mesh method with variable mesh relaxation time, *Appl Numer Math* 58 (2008), 249–263.
12. P. Bochev, G. liao, and G. dela Pena, Analysis and computation of adaptive moving grids by deformation, *Numer Meth PDEs* 12 (1996), 489.
13. G. Liao, F. Liu, G. dela Pena, D. Peng, and S. Osher, Level-set-based deformation methods for adaptive grids, *J Comput Phys* 159 (2000), 103–122.
14. S. Osher and J. Sethian, Front propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations, *J Comput Phys* 79 (1988), 12.
15. J. A. Sethian, *Level set methods and fast marching methods*, Cambridge University Press, New York, 1999.
16. Z. Zhang, Moving mesh method with conservative interpolation based on L2-projection, *Commun Comput Phys* 1 (2006), 930–944.
17. Y. Di, R. Li, T. Tang, and P. Zhang, Level set calculations for incompressible two-phase flows on a dynamically adaptive grid, *J Sci Comput* 31 (2007), 75–98.