COMMUNICATIONS IN COMPUTATIONAL PHYSICS
Vol. **1**, No. 5, pp. 930-944

Commun. Comput. Phys.
October 2006

Moving Mesh Method with Conservative Interpolation Based on L^2-Projection

Zhengru Zhang*

School of Mathematical Sciences, Beijing Normal University, Beijing 100875, China.

Received 12 October 2005; Accepted (in revised version) 14 February 2006

Communicated by Chi-Wang Shu

Abstract. We develop an efficient one-dimensional moving mesh algorithm for solving partial differential equations. The main contribution of this paper is to design an effective interpolation scheme based on L^2-projection for the moving mesh method. The proposed method preserves not only the mass-conservation but also the first order momentum of the underlying numerical solution at each mesh redistribution step. Numerical examples are presented to demonstrate the effectiveness of the new interpolation technique.

Key words: Moving mesh method; conservative interpolation; L^2-projection.

1 Introduction

In the past two decades, there has been important progress in developing adaptive mesh methods for PDEs. Mesh adaptivity is usually of two types: local mesh refinement and moving mesh method. In local mesh refinement methods, see, e.g., [1], an adaptive mesh is obtained by adding or removing grid points to achieve a desired level of accuracy. They allow a posteriori error estimates to guide the mesh refinement and coarsening. However, local mesh refinement methods require complicated data structures, and increase computer code complexity. In the moving mesh method, the mesh points are moved continuously in the whole domain to concentrate in regions where the solution has the largest variations. Unlike the local mesh refinement methods, the moving mesh method keeps a fixed number of mesh points throughout the computation.

*Correspondence to: Zhengru Zhang, School of Mathematical Sciences, Beijing Normal University, Beijing 100875, P.R. China. Email: zrzhang@bnu.edu.cn

There have been a variety of moving mesh techniques, including the variational approach [2, 21]; the moving finite element methods [7, 13]; the moving mesh PDEs approach [4, 5, 10]; the moving mesh methods based on harmonic mapping of [9, 11, 12]. We can classify the moving mesh methods into two classes: interpolation free moving mesh methods and moving mesh methods using interpolation. The earlier works such as [8, 13] are directly based on the equidistribution principle which was first proposed by de Boor [3]. Generally speaking, the earlier moving mesh methods have a common feature that the grid equation is strongly coupled with the underlying PDEs, and consequently, their numerical approximation leads to a nonlinear system even if the underlying PDEs are linear. Since the mesh redistribution and the PDEs are solved simultaneously, the interpolation of solution information from the old mesh to the new mesh is unnecessary. So we call this class of methods interpolation-free moving mesh methods. Another group of interpolation-free moving mesh methods was developed by Russell and his research group; see, e.g., [4]. Their moving mesh equation is written as moving mesh partial differential equations (MMPDEs) based on the equidistribution principle.

Li et al. [11] developed a moving finite element method containing two independent parts: solution evolution and mesh redistribution. These two parts are relatively independent in the sense that the change of the underlying PDEs will only affect the first part (solution evolution), while the second part (mesh redistribution) is kept unchanged. An immediate advantage of this class of methods is the computer code simplicity. In the mesh redistribution part, the adaptive mesh is usually generated through some iterative algorithm, which makes the solution interpolation from one iteration step to the next indispensable. In the moving finite element paper [11], a convection equation is solved to pass the solution in the old mesh to the new mesh. Later, Tang and Tang [18] proposed a moving mesh finite volume method, which uses conservative interpolation to pass the soution information from the old mesh to the new mesh. This method has recently been applied to the Hamilton-Jacobi equations [19], convection-dominated problems [24, 25], and phase-field problems [17]. The conventional interpolation method, such as third order polynomial interpolation [14], is applied to solve the Schrödinger equations involving static blow-ups.

The main objective of this work is to propose an interpolation algorithm based on L^2-projection for the moving mesh method. Similar to [11, 18, 25], our moving mesh approach includes two independent parts: PDE solution evolution and mesh redistribution. The first part is problem-dependent, that is to say, we need to design appropriate numerical solvers for different governing equations. In the second part, we need to transfer the solution information from the old mesh to the new one. This is similar to an Arbitrary Lagrangian-Eulerian (ALE) method, where a remapping algorithm plays an essential and important role. A good remapping scheme should possess properties such as accuracy, conservation and computational efficiency. Many authors have worked on developing remapping strategies. Recently, Cheng and Shu [6] proposed a high-order accurate conservative remapping method on staggered meshes. Their method is built on the framework of ENO interpolation and reconstruction, so it has the good properties such as high-order

accuracy and essential non-oscillatory performance. In this paper, the mesh is generated using the equidistribution principle, and then the solution in the old mesh is interpolated to the new mesh by using an L^2-projection technique.

The idea of applying L^2-projection to moving mesh interpolation is new. An important property of the proposed interpolation is that the mass-conservation and momentum are preserved after interpolation, which is an essential requirement for many practical problems. The paper is organized in two main parts. The first part (Sections 2 and 3) introduces our moving mesh strategy and derives the interpolation scheme based on L^2-projection. In the second part (Section 4), several numerical experiments including KdV equation and Kuramoto-Sivashinsky equation are presented to show the effectiveness of the proposed algorithm. Finally, some concluding remarks will be given in Section 5.

2 Moving mesh algorithm

As mentioned in Section 1, our solution procedure contains two independent parts: PDE solution evolution and mesh redistribution. Here, we give a brief derivation of the 1D mesh generation equation. Let $\boldsymbol{x} = (x_1, \cdots, x_d)$ and $\boldsymbol{\xi} = (\xi_1, \cdots, \xi_d)$ denote the physical and computational coordinates, respectively. Here d denotes the number of spatial dimensions. A one-to-one coordinate transformation from the computational (or logical) domain Ω_c to the physical domain Ω_p is denoted by $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi})$, $\boldsymbol{\xi} \in \Omega_c$. Similarly, the inverse form is denoted by $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{x})$, $\boldsymbol{x} \in \Omega_p$. In the variational approach, the mesh map is to find the minimizer of the functional

$$E(\boldsymbol{\xi}) = \frac{1}{2} \sum_k \int_{\Omega_p} \nabla \xi_k^T G_k^{-1} \nabla \xi_k d\boldsymbol{x}, \tag{2.1}$$

where the G_k's are given symmetric positive definite matrices called monitor functions. In general, a monitor function depends on the solution or its derivatives of the underlying PDE. The corresponding Euler-Lagrange equation for the variational mesh functional (2.1) is given by

$$\nabla \cdot (G_k^{-1} \nabla \xi_k) = 0, \qquad 1 \le k \le d. \tag{2.2}$$

The simplest choice of the monitor function is $G_k = \omega I,\ 1 \le k \le d$, where I is the identity matrix and ω is a weight function. In this sense, the equation (2.2) reduces to Winslow's variable diffusion method [21]:

$$\nabla \cdot \left(\frac{1}{\omega} \nabla \xi_k \right) = 0, \quad 1 \le k \le d. \tag{2.3}$$

Therefore, the above equation determines a map between the physical domain Ω_p and the computational domain Ω_c.

For 1D problems, assume the physical domain is $[a, b]$ and the computational domain is unit interval $[0, 1]$, respectively. The 1D Euler-Lagrange equation becomes $(\omega^{-1} \xi_x)_x = 0$,

Table 1: Outline of 1D moving mesh finite volume algorithm.

Initialize variables
0. Determine the initial mesh based on the initial function.
1. Determine Δt based on some stability condition so that $t^{n+1} = t^n + \Delta t$.
2. Advance the solution one time step based on an appropriate numerical scheme.
3. Mesh Redistribution

a. Solve the mesh redistributing equation (a generalized Laplacian equation) by one Gauss-Seidel iteration, to get $\boldsymbol{x}^{(k),n}$
b. Interpolate the approximate solutions on the new grid $\boldsymbol{x}^{(k),n}$
c. A weighted average of the locally calculated monitor at each computational cell and the surrounding monitor values.
d. The iteration procedure (a.)-(c.) on grid-motion and solution-interpolation is continued until there is no significant change in calculated new grids from one iteration to the next.

Start new time step (go to **1** above).

$x \in [a, b]$. Equivalently, from the above equation we can obtain the conventional 1-D equidistribution principle equation:

$$(\omega x_\xi)_\xi = 0, \quad \xi \in [0, 1]. \tag{2.4}$$

In this work the mesh equation (2.4) is used to generate the moving mesh. A complete solution procedure is displayed in Table 1.

In the step **3**(a) of Table 1, the iterative step is performed following Gauss-Seidel formula:

$$x_j^{(k+1)} = \left(\omega_{j+1}^{(k)} x_{j+1}^{(k)} + \omega_{j-1}^{(k)} x_{j-1}^{(k+1)}\right) / \left(\omega_{j+1}^{(k)} + \omega_{j-1}^{(k)}\right). \tag{2.5}$$

In the step **3**(c), the monitor function is smoothed as indicated below:

$$\omega_j \leftarrow \frac{1}{4}(\omega_{j-1} + 2\omega_j + \omega_{j+1}). \tag{2.6}$$

In the step **3**(b), the interpolation is realized through an L^2-projection technique which will be described in detail in Section 3.

3 Interpolation based on L^2-projection

In the step 3(b) of Table 1, we need to transform the solution information from the old mesh to the newly produced mesh. Let $\{x_j\}$ be the old mesh and $\{\tilde{x}_j\}$ be the new one.

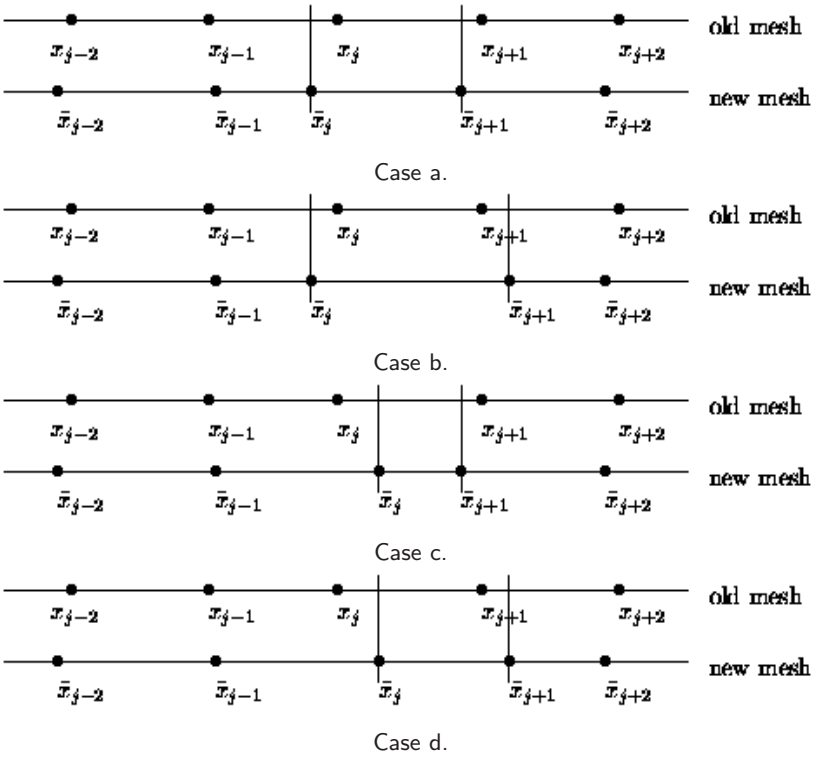


Figure 1: Demonstration of the new and old mesh distribution.

The numerical solution should be projected from the approximation function space V_h defined on $\{x_j\}$ to another approximation function space W_h defined on $\{\tilde{x}_j\}$. Let

$$V(I_{j+\frac{1}{2}}) = \{\phi_{0,j+\frac{1}{2}}(x), \phi_{1,j+\frac{1}{2}}(x), \cdots, \phi_{p,j+\frac{1}{2}}(x)\} \tag{3.1}$$

be a local basis of the space V_h in cell $I_{j+\frac{1}{2}}$, and

$$W(\tilde{I}_{j+\frac{1}{2}}) = \{\tilde{\phi}_{0,j+\frac{1}{2}}(x), \tilde{\phi}_{1,j+\frac{1}{2}}(x), \cdots, \tilde{\phi}_{p,j+\frac{1}{2}}(x)\} \tag{3.2}$$

be a local basis of the space W_h in cell $\tilde{I}_{j+\frac{1}{2}}$. Assume we have already constructed a piecewise polynomial approximation $u_h(x) \in V_h$ in the mesh $\{x_j\}$. That is, in the control volume $I_{j+\frac{1}{2}}$, we have

$$u_h(x) = \sum_{i=0}^{p} a_{i,j+\frac{1}{2}} \phi_{i,j+\frac{1}{2}}(x), \qquad x \in I_{j+\frac{1}{2}}, \tag{3.3}$$

where p denotes the degree of the piecewise polynomial, and the coefficients $\{a_{i,j+\frac{1}{2}}\}$ will be determined through some kind of resconstruction such as a slope limiter or an

essentially non-oscillatory (ENO) method. After one step of Gauss-Seidel iteration (step 3(a) in Table 1), we obtain the new mesh distribution $\{\tilde{x}_j\}$. In the new control volume $\tilde{I}_{j+\frac{1}{2}}$, we also have the following form of piecewise polynomial approximation $\tilde{u}_h(x) \in W_h$:

$$\tilde{u}_h(x) = \sum_{i=0}^{p} \tilde{a}_{i,j+\frac{1}{2}} \tilde{\phi}_{i,j+\frac{1}{2}}(x), \quad x \in \tilde{I}_{j+\frac{1}{2}}. \tag{3.4}$$

The coefficients $\{\tilde{a}_{i,j+\frac{1}{2}}\}$ are to be determined based on piecewise polynomial distribution defined on the old mesh, i.e., (3.3). In this work, an L^2-projection will be used to find the $\tilde{u}_h(x)$ in (3.4). Since the approximation function space consists of piecewise functions, the L^2-projection leads to a local and simple linear system.

Multiplying both sides of (3.4) by the basis function $\tilde{\phi}_{l,j+\frac{1}{2}}(x) \in W(\tilde{I}_{j+\frac{1}{2}})$ and integrating the resulting equation over the cell $\tilde{I}_{j+\frac{1}{2}}$ lead to the following equations:

$$\int_{\tilde{I}_{j+\frac{1}{2}}} u_h(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx = \int_{\tilde{I}_{j+\frac{1}{2}}} \tilde{u}_h(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx, \quad \tilde{\phi}_{l,j+\frac{1}{2}}(x) \in W(\tilde{I}_{j+\frac{1}{2}}), \tag{3.5}$$

for $l = 0, 1, \cdots, p$. We replace $u_h(x)$ and $\tilde{u}_h(x)$ above by (3.3) and (3.4), respectively.

We now consider all the possible cases for the grid structures as shown in Fig. 1. If the grid points of the old and new meshes are distributed as shown in Fig. 1(a), then (3.5) becomes

$$\begin{aligned} & \sum_{i=0}^{p} \tilde{a}_{i,j+\frac{1}{2}} \int_{\tilde{I}_{j+\frac{1}{2}}} \tilde{\phi}_{i,j+\frac{1}{2}}(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx \qquad (3.6)\\ = & \int_{\tilde{x}_j}^{x_j} u_h(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx + \int_{x_j}^{\tilde{x}_{j+1}} u_h(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx \\ = & \sum_{i=0}^{p} a_{i,j-\frac{1}{2}} \int_{\tilde{x}_j}^{x_j} \phi_{i,j-\frac{1}{2}}(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx + \sum_{i=0}^{p} a_{i,j+\frac{1}{2}} \int_{x_j}^{\tilde{x}_{j+1}} \phi_{i,j+\frac{1}{2}}(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx, \end{aligned}$$

for $\tilde{\phi}_{l,j+\frac{1}{2}}(x) \in W(\tilde{I}_{j+\frac{1}{2}})$, $l = 0, 1, \cdots, p$. Since (3.3) has been constructed, all the terms in the right-hand side of (3.6) are known. Therefore, (3.6) forms a linear system for the unknown $\{\tilde{a}_{0,j+\frac{1}{2}}, \tilde{a}_{1,j+\frac{1}{2}}, \cdots, \tilde{a}_{p,j+\frac{1}{2}}\}$. Similar systems can be obtained for Cases (b)-(d): For the case (b) in Fig. 1, we have

$$\begin{aligned} & \sum_{i=0}^{p} \tilde{a}_{i,j+\frac{1}{2}} \int_{\tilde{I}_{j+\frac{1}{2}}} \tilde{\phi}_{i,j+\frac{1}{2}}(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx \qquad (3.7)\\ = & \int_{\tilde{x}_j}^{x_j} u_h(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx + \int_{x_j}^{x_{j+1}} u_h(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx + \int_{x_{j+1}}^{\tilde{x}_{j+1}} u_h(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx; \end{aligned}$$

for the case (c) in Fig. 1, we have

$$\sum_{i=0}^{p} \tilde{a}_{i,j+\frac{1}{2}} \int_{\tilde{I}_{j+\frac{1}{2}}} \tilde{\phi}_{i,j+\frac{1}{2}}(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx = \int_{\tilde{x}_j}^{\tilde{x}_{j+1}} u_h(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx; \tag{3.8}$$

and for the case (d) in Fig. 1, we get

$$\sum_{i=0}^{p} \tilde{a}_{i,j+\frac{1}{2}} \int_{\tilde{I}_{j+\frac{1}{2}}} \tilde{\phi}_{i,j+\frac{1}{2}}(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx = \int_{\tilde{x}_j}^{x_{j+1}} u_h(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx + \int_{x_{j+1}}^{\tilde{x}_{j+1}} u_h(x)\tilde{\phi}_{l,j+\frac{1}{2}}(x)dx. \tag{3.9}$$

In our computation, we set $p = 1$. The polynomial basis function spaces are defined as

$$\begin{aligned} V(I_{j+\frac{1}{2}}) &= \{\phi_{0,j+\frac{1}{2}}(x), \phi_{1,j+\frac{1}{2}}(x)\} \equiv \{1, x - x_j\}, \\ W(\tilde{I}_{j+\frac{1}{2}}) &= \{\tilde{\phi}_{0,j+\frac{1}{2}}(x), \tilde{\phi}_{1,j+\frac{1}{2}}(x)\} \equiv \{1, x - \tilde{x}_j\}. \end{aligned} \tag{3.10}$$

After obtaining $\{\tilde{a}_{i,j+\frac{1}{2}}\}$ in one of the four cases, the solution information on the new mesh $\{\tilde{I}_{j+\frac{1}{2}}\}$, i.e., (3.4), can be obtained, provided that the solution information on the old mesh, (3.3), is given. In this work, the piecewise linear polynomial (3.3) is constructed using the minmod slope limiter. That is, on each cell, the approximation function $u_h(x)$ is defined in the form

$$u_h(x) = \bar{u}_{j+\frac{1}{2}} + \sigma_{j+\frac{1}{2}}(x - x_{j+\frac{1}{2}}), \quad x \in I_{j+\frac{1}{2}}, \tag{3.11}$$

where $\sigma_{j+\frac{1}{2}}$ is defined by

$$\sigma_{j+\frac{1}{2}} = \text{minmod}\left(\frac{\bar{u}_{j+\frac{3}{2}} - \bar{u}_{j+\frac{1}{2}}}{|I_{j+\frac{1}{2}}|}, \frac{\bar{u}_{j+\frac{1}{2}} - \bar{u}_{j-\frac{1}{2}}}{|I_{j-\frac{1}{2}}|}\right). \tag{3.12}$$

Here the minmod function of two arguments is defined by

$$\text{minmod}(a, b) = \begin{cases} a & \text{if } |a| < |b| \text{ and } ab > 0, \\ b & \text{if } |b| < |a| \text{ and } ab > 0, \\ 0 & \text{if } ab \le 0. \end{cases}$$

If a and b are of the same sign, then this selects the one with smaller absolute value; else, it takes the value zero. This choice of slope gives second-order accuracy for smooth solutions while still satisfying the TVD property.

With the piecewise linear construction, the coefficients in (3.3) have the form

$$a_{0,j+\frac{1}{2}} = \bar{u}_{j+\frac{1}{2}} + \sigma_{j+\frac{1}{2}}(x_j - x_{j+\frac{1}{2}}), \quad a_{1,j+\frac{1}{2}} = \sigma_{j+\frac{1}{2}}. \tag{3.13}$$

A high resolution method such as an ENO and WENO scheme can be used to construct the high-order piecewise polynomial defined in (3.3); see for example [6].

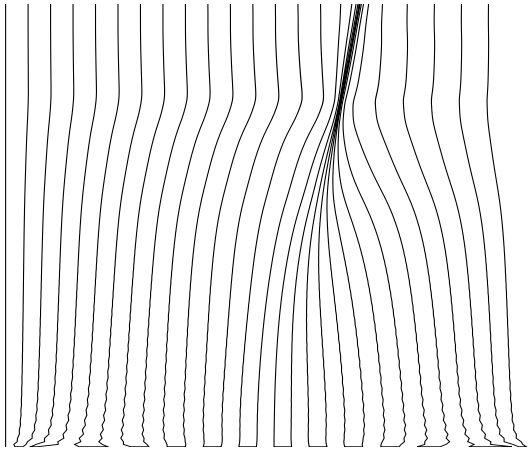


Figure 2: Example 4.1: mesh trajectory for $0 < t < 2$.

In (3.5), taking the test function as 1 leads to

$$\int_{\tilde{I}_{j+\frac{1}{2}}} u_h(x)dx = \int_{\tilde{I}_{j+\frac{1}{2}}} \tilde{u}_h(x)dx. \tag{3.14}$$

So the L^2-projection interpolation guarantees the conservation of mass in the following discrete sense:

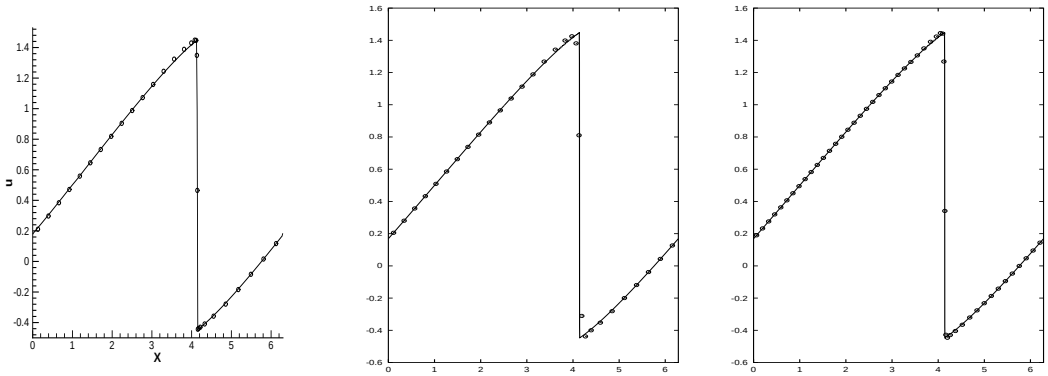
$$\sum_j \int_{I_{j+\frac{1}{2}}} u_h(x)dx = \sum_j \int_{\tilde{I}_{j+\frac{1}{2}}} \tilde{u}_h(x)dx. \tag{3.15}$$

It is noted that the interpolation scheme in [18] also has the property (3.15), but the present interpolation has the mass-conservation held in *each cell*, see (3.14). Another important difference is that the present approach also preserves solution momentum in each cell,

$$\int_{\tilde{I}_{j+\frac{1}{2}}} u_h(x)(x-\tilde{x}_j)dx = \int_{\tilde{I}_{j+\frac{1}{2}}} \tilde{u}_h(x)(x-\tilde{x}_j)dx. \tag{3.16}$$

This can be verified by replacing the basis function in (3.5) by $x - \tilde{x}_j$.

We close this section by pointing out that the L^2-projection interpolation proposed in this section can be regarded as some combination of the high-order conservative remapping developed in [6] and the discontinuous Galerkin method based on non-polynomial approximation spaces developed in [23]. As proved in [23], the transfer of solution information from one approximation space V_h to another approximation space W_h does not destroy the accuracy.



(a) This work: $N = 30$ (b) [18]: $N = 30$ (c) [18]: $N = 50$

Figure 3: Example 4.1: comparison of the adaptive mesh solutions at $t = 2$ with that given by using the interpolation method in [18]. The solid line denotes the exact solution and 'o' denotes the numerical solution.

4 Numerical experiments

In this section, we implement our moving mesh method proposed in the previous sections for several 1D model problems. The cell-centered finite volume method is used to solve these PDEs. For time evolution, a 3-stage Runge-Kutta method proposed by Shu and Osher [15] is employed. For a given ODE system $u'(t) = L(u)$, it is discretized in time by

$$\begin{aligned} u_j^{(1)} = u_j^n + \Delta t L(u_j^n), \quad u_j^{(2)} = \frac{3}{4}u_j^n + \frac{1}{4}\left[u_j^{(1)} + \Delta t L(u_j^{(1)})\right], \\ u_j^{n+1} = \frac{1}{3}u_j^n + \frac{2}{3}\left[u_j^{(2)} + \Delta t L(u_j^{(2)})\right]. \end{aligned} \tag{4.1}$$

Example 4.1. [Burgers' equation.] This example is the inviscous Burgers' equation

$$u_t + \left(\frac{u^2}{2}\right)_x = 0, \quad 0 < x < 2\pi\,, \tag{4.2}$$

subject to the initial condition $u(x, 0) = \frac{1}{2} + \sin(x)$, and a periodic boundary condition.

The monitor function is chosen as $\omega = \sqrt{1 + 0.5|u_x|^2}$. The number of Gauss-Seidel iteration used in each time step is 5. The numerical scheme for solving the Burgers' equation is the second-order MUSCL finite volume scheme (with the Lax-Friedrichs flux). Fig. 2 displays the grid trajectory up to $t = 2$. It is observed that the mesh points begin to cluster at about $t = 1.2$, which indicates the shock begins to form at this time. It is also observed in Fig. 3 that 30 grid points are sufficient to capture the steep solution using the proposed interpolation method. For the same problem, the method proposed in [18] requires 50 grid points in order to obtain a comparable resolution near the shock region. This implies that the the L^2-projection interpolation approach is more appropriate than the conservative interpolation method proposed in [18].

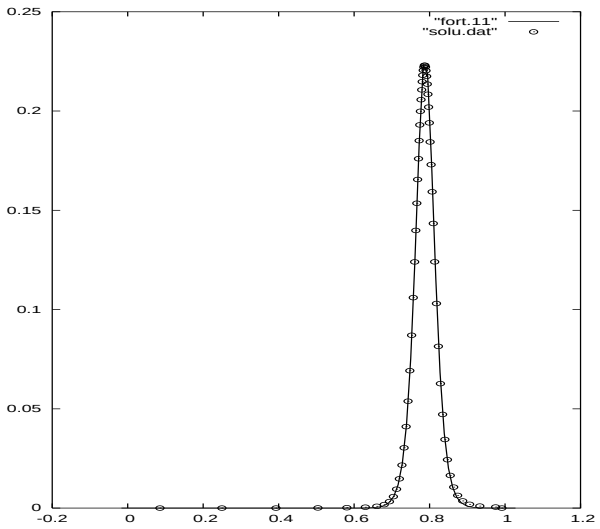


Figure 4: Example 4.2: the adaptive mesh solution at $t = 0.5$ with $N = 60$. The solid line denotes the exact solution and 'o' means the numerical solution.

Example 4.2. [Generalized Korteweg-de Vries equation.] Here we apply the proposed moving mesh method to one-dimensional generalized Korteweg-de Vries equation (GKdV equation):

$$u_t + f(u)_x + \epsilon u_{xxx} = 0, \quad f(u) = \eta u + \frac{u^{p+1}}{p+1}, \tag{4.3}$$

with the initial condition $u(x,0) = A\text{sech}^{2/p}[K \cdot (x - x_0)]$.

This problem has the exact solution $u(x,t) = A \cdot \text{sech}^{2/p}[K \cdot (x - x_0) - wt]$, where

$$K = p \cdot \sqrt{\frac{A^p}{2\epsilon(p+1)(p+2)}}, \quad w = K \cdot \left(\eta + \frac{2A^p}{(p+1)(p+2)}\right).$$

In our numerical experiment, we set $\eta = 1, p = 1, \epsilon = 0.2058 \times 10^{-4}, A = 0.2275$, and $x_0 = 0.25$. Since the problem involves a 3rd order derivative, the monitor function should involve higher-order derivatives. The monitor function for this example is taken as $\omega = \sqrt{1 + |u_{\xi\xi}|^2}$. It is seen from Fig. 4 that most of the grid points are moved to capture the peak. Around the peak region, the adaptive mesh has the largest mesh density. The numerical solution obtained using our moving mesh method has a good agreement with the exact solution (denoted by solid line in Fig. 4).

Example 4.3. [Kuramoto-Sivashinsky equation.] In this example, we show the order of accuracy for the proposed moving mesh algorithm. The Kuramoto-Sivashinsky (K-S) equation is of the form:

$$u_t + uu_x - u_{xx} + u_{xxxx} = 0, \quad -\infty < x < \infty. \tag{4.4}$$

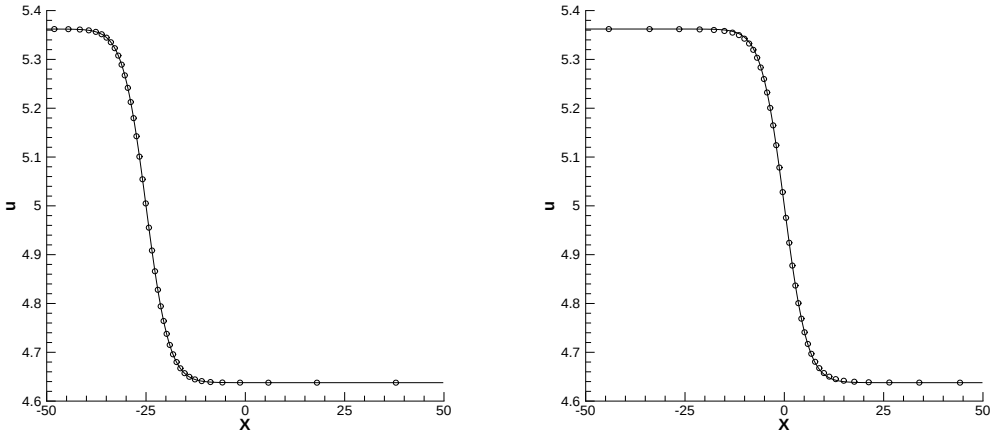


Figure 5: Example 4.3: the adaptive mesh solution at $t = 0$ and $t = 5$ with $N = 40$. The solid line denotes the exact solution and 'o' denotes the numerical solution. $c = 5$, $k = \frac{1}{2\sqrt{19}}$, $x_0 = -25$.

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

$$u(x,t) = c + \frac{15}{19\sqrt{19}}(-3\tanh(k(x-ct-x_0)) + \tanh^3(k(x-ct-x_0))). \tag{4.5}$$

It is necessary to describe the numerical solution solver for the above K-S equation. We rewrite the equation (4.4) as a first order system:

$$\begin{aligned} u_t + f(u)_x - v_x + w_x &= 0, \\ v - u_x = 0, \quad p - v_x = 0, \quad w - p_x &= 0. \end{aligned} \tag{4.6}$$

Integrating the above system over the cell $[x_j, x_{j+1}] \times [t^n, t^{n+1}]$ leads to the following finite volume scheme:

$$\begin{aligned} \frac{u_j^{n+1} - u_j^n}{\Delta t} + \Delta_+ \hat{f}_j^n - \Delta_+ \hat{v}_j^n + \Delta_+ \hat{w}_j^n &= 0, \\ v_j^{n+1} - \Delta_+ \hat{u}_j^n = 0, \quad p_j^{n+1} - \Delta_+ \hat{v}_j^n = 0, \quad w_j^{n+1} - \Delta_+ \hat{p}_j^n &= 0, \end{aligned} \tag{4.7}$$

where $\Delta_+ w_j = (w_{j+\frac{1}{2}} - w_{j-\frac{1}{2}})/\Delta x_{j+\frac{1}{2}}$. The "hat" terms in (4.7) are those cell boundary terms from the integration by parts, called numerical fluxes which are single-valued functions defined on the edges. The numerical fluxes are computed based on different guiding principles for different governing equations to ensure stability. In our computations, we can take $\hat{f} = \hat{f}(u^-, u^+)$, $\hat{u} = u^-$, $\hat{v} = v^+$, and $\hat{p} = p^-$, where the simple Lax-Friedrichs flux is applied to $\hat{f}$,

$$\hat{f}(u^-, u^+) = \frac{1}{2}(f(u^-) + f(u^+) - \alpha(u^+ - u^-)), \quad \alpha = \max|f'(u)|. \tag{4.8}$$

Table 2: L^2-error and convergence order for the Kuramoto-Sivashinsky equation (4.4) with exact solution (4.5) at $t = 1$. $c = 0.2$, $k = \frac{1}{2\sqrt{19}}$, $x_0 = -10$.

N	20	40	80	160
L^2-error	8.145e-3	1.600e-3	2.267e-4	5.758e-5
rate	-	2.43	2.81	1.98

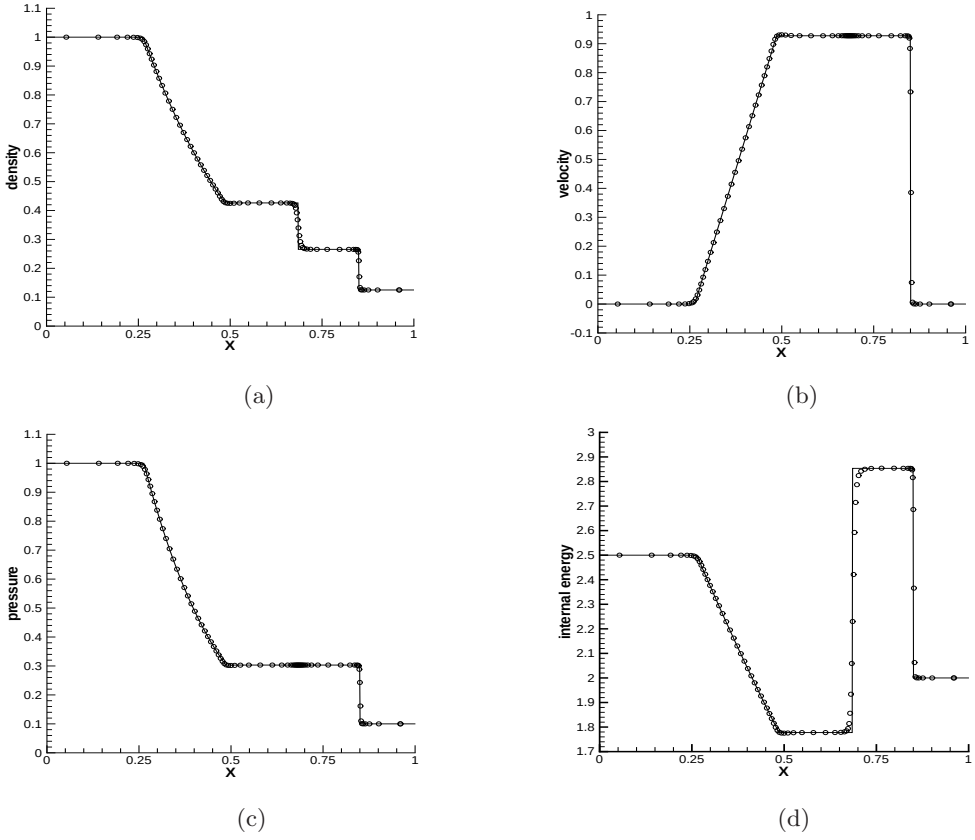


Figure 6: Example 4.4: the adaptive mesh solution at $t = 0.2$ with $N = 80$. The solid line denotes the exact solution and 'o' denotes the numerical solution. (a): density, (b): velocity, (c): pressure, (d): internal energy.

In (4.8), the maximum is taken between u^+ and u^-. The values of u^+ and u^- are calculated through the piecewise linear construction in the form (3.11), and v^+, v^-, p^+, p^- are obtained in a similar way.

In Fig. 5 we plot the wave propagation profile for the K-S equation (4.4)-(4.5) with $c = 5, k = \frac{1}{2\sqrt{19}}$ and $x_0 = -25$. We also present the L^2-errors and rate of convergence in Table 2. The corresponding rate of convergence is about two; and the errors obtained by using our moving mesh method are almost the same as those computed by using the local discontinuous Galerkin method [22]. More precisely, the results in Table 2 are almost

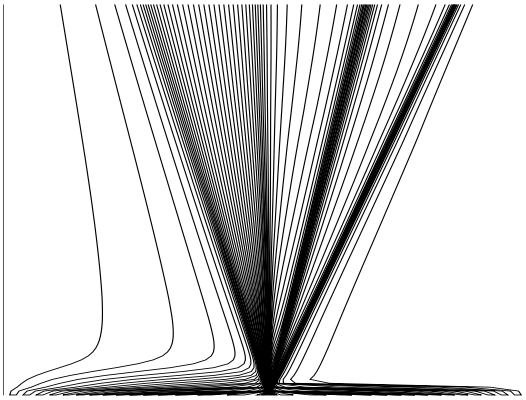


Figure 7: Example 4.4: adaptive mesh points trajectory with $N = 80$.

identical to those given in Table 4.2 of [22].

Example 4.4. [Euler equations of gas dynamics.] In this example, we test our adaptive mesh algorithm for the one-dimensional Euler equations of gas dynamics,

$$\begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}_t + \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix}_x = 0\,, \tag{4.9}$$

where ρ, u, p, and E are density, velocity, pressure, and total energy, respectively. The above system is closed by the equation of state, $p = (\gamma - 1)(E - \rho u^2/2)$ with $\gamma = 1.4$. The initial data are chosen as

$$(\rho, \rho u, E) = \begin{cases} (1, 0, 2.5) & \text{if } x < 0.5, \\ (0.125, 0, 0.25) & \text{if } x > 0.5. \end{cases}$$

This well-known problem has been studied by Stockie, Mackenzie and Russell [16] and Tang and Tang [18], using moving mesh methods. The monitor function employed in our computation is the simple second derivative based monitor: $\omega = \sqrt{1 + 30|\rho_{\xi\xi}|^2}$. It is noted that the moving mesh methods used in [16] and [18] require special attention to choose an appropriate monitor function. In Fig. 6, we show the numerical solution obtained using the proposed moving mesh method. The exact solution is computed using the Riemann solver given in [20]. It is found in Fig. 6 that the contact wave and the shock discontinuities are well resolved using 80 grid points. Moreover, the mesh trajectory is presented in Fig. 7. It is noted that in order to obtain the same resolution the moving mesh approach in [18] requires at least 100 grid points.

5 Concluding remarks

In this paper, we have presented a 1-D moving mesh method using a conservative interpolation based on L^2-projection. The proposed method guarantees the mass-conservation

in discrete sense. Numerical examples are presented to illustrate the effectiveness of the proposed approach. It is demonstrated that to obtain the same solution resolution our moving mesh approach requires fewer grid points than the method presented in [18]. We also point out that the idea of L^2-projection can be easily extended to handle higher-order interpolation schemes used in the moving mesh methods.

In upcoming works, we will study the L^2-projection-based interpolaton for the moving mesh methods in high space dimensions, which seems much more challenging. In some sense, this task can be viewed as an application of the remapping method proposed in [6] and the non-polynomial approximation developed in [23].

Acknowledgments

I am grateful to Prof. C.-W. Shu for introducing me the idea of L^2-projection and for several helpful discussions. I also thank Prof. T. Tang and Prof. H.Z. Tang for many useful suggestions. Part of the research was supported by NSFC Tian Yuan grant 10526009.

References

[1] M. J. Berger and P. Collela, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys., 82 (1989), 62-84.
[2] J. U. Brackbill and J. S. Saltzman, Adaptive zoning for singular problems in two dimensions, J. Comput. Phys., 46 (1982), 342-368.
[3] C. de Boor, Good Approximation by Splines with Variable Knots II, Springer Lecture Notes Series 363, Springer-Verlag, Berlin, 1973.
[4] W. M. Cao, W. Z. Huang and R. D. Russell, An r-adaptive finite element method based upon moving mesh PDEs, J. Comput. Phys., 149 (1999), 221-244.
[5] H. D. Ceniceros and T. Y. Hou, An efficient dynamiclly adaptive mesh for potentially singular solutions, J. Comput. Phys., 172 (2001), 609-639.
[6] J. Cheng and C.-W. Shu, A high order accurate conservative remapping method on staggered meshes. To appear in Appli. Numer. Math.
[7] F. Davis and J. E. Flaherty, An adaptive finite element method for initial-boundary value problems for partial differential equations, SIAM J. Sci. Statist. Comput., 3 (1982), 6-27.
[8] E. A. Dorfi and L.O'c. Drury, Simple adaptive grids for 1-d initial value problems, J. Comput. Phys., 69 (1987), 175-195.
[9] A. S. Dvinsky, Adaptive grid generation from harmonic maps on Riemannian manifolds, J. Comput. Phys., 95 (1991), 450-476.
[10] S. Li and L. Petzold, Moving mesh methods with upwinding schemes for time-dependent PDEs, J. Comput. Phys., 131 (1997), 368-377.
[11] 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.
[12] R. Li, T. Tang and P. W. Zhang, A moving mesh finite element algorithm for singular problems in two and three space dimensions, J. Comput. Phys., 177 (2002), 365-393.
[13] K. Miller and R. N. Miller, Moving finite elements I, SIAM J. Numer. Anal., 18 (1981), 1033-1057.

[14] W. Q. Ren and X. P. Wang, An iterative grid redistribution method for singular problems in multiple dimensions, J. Comput. Phys., 159 (2000), 246-272.
[15] C.-W. Shu and S. Osher, Efficient implement of essentially non-oscillatory shock-wave shemes, II, J. Comput. Phys., 83 (1989), 32-78.
[16] J. M. Stockie, J. A. Mackenzie and R. D. Russell, A moving mesh method for one-dimensional hyperbolic conservation laws, SIAM J. Sci. Comput., 22 (2001), 1791-1813.
[17] Z. J. Tan, T. Tang and Z. R. Zhang, A simple moving mesh method for one- and two-dimensional phase-field equations, J. Comput. Appl. Math., 190 (2006), 252-269.
[18] 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.
[19] H. Z. Tang, T. Tang and P. W. Zhang, An adaptive mesh redistribution method for nonlinear Hamilton-Jacobi equations in two and three dimensions, J. Comput. Phys., 188 (2003), 543-572.
[20] E. F. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics, Springer-Verlag, 1999.
[21] A. Winslow, Numerical solution of the quasi-linear Poisson equation, J. Comput. Phys., 1 (1967), 149-172.
[22] Y. Xu and C.-W. Shu, Local discontinuous Galerkin methods for the Kuramoto-Sivashinsky equations and the Ito-type coupled KdV equations, to appear in Computer Methods in Applied Mechanics and Engineering, 2006.
[23] L. Yuan and C.-W. Shu, Discontinuous Galerkin method based on non-polynomial approximation spaces. To appear in J. Comput. Phys..
[24] Z. R. Zhang, Moving Mesh Methods for Convection-Dominated Equations and Nonlinear Conservation Problems, Ph.D. Thesis, Hong Kong Baptist University, 2003.
[25] Z. R. Zhang and T. Tang, An adaptive mesh redistribution algorithm for convection-dominated problems, Commun. Pure Appl. Anal., 1 (2002), 341-357.