A MOVING MESH FINITE ELEMENT ALGORITHM FOR SINGULAR PROBLEMS IN TWO AND THREE SPACE DIMENSIONS

RUO LI †, TAO TANG ‡, AND PINGWEN ZHANG †

ABSTRACT. A framework for adaptive meshes based on the Hamilton-Schoen-Yau theory was proposed by Dvinsky. In a recent work [15], we extended Dvinsky's method to provide an efficient moving mesh algorithm which compared favorably with the previously proposed schemes in terms of simplicity and reliability. In this work, we will further extend the moving mesh methods based on harmonic maps to deal with mesh adaptation in three space dimensions. In obtaining the variational mesh, we will solve an *optimization problem* with some appropriate constraints, which is in contrast to the traditional method of solving the Euler-Lagrange equation directly. The key idea of this approach is to update the interior and boundary grids simultaneously, rather than considering them separately. Application of the proposed moving mesh scheme is illustrated with some two- and three-dimensional problems with large solution gradients. The numerical experiments show that our methods can accurately resolve detailed features of singular problems in 3D.

Key words. finite element method, moving mesh method, harmonic map, partial differential equations, optimization

1. Introduction

Moving mesh methods have important applications in a variety of physical and engineering areas such as solid and fluid dynamics, combustion, heat transfer, material science etc. The physical phenomena in these areas develop dynamically singular or nearly singular solutions in fairly localized regions, such as shock waves, boundary layers, detonation waves etc. The numerical investigation of these physical problems may require extremely fine meshes over a small portion of the physical domain to resolve the large solution variations. In multi-dimensions, developing effective and robust adaptive grid methods for these problems becomes necessary. Successful implementation of the adaptive strategy can increase the accuracy of the numerical approximations and also decrease the computational cost.

Several moving mesh techniques have been introduced in the past, in which the most advocated method is the one based on solving elliptic PDEs first proposed by Winslow [23]. Winslow's formulation requires the solution of a nonlinear, Poisson-like equation to generate a mapping from a regular domain in a parameter space Ω_c to an irregularly shaped domain in physical space Ω. By connecting points in the physical space corresponding to discrete points in the parameter space, the physical domain can be covered with a

† School of Mathematical Science, Peking University, Beijing 100871, People's Republic of China. Email: pzhang@math.pku.edu.cn.

‡ Department of Mathematics, The Hong Kong Baptist University, Kowloon Tong, Hong Kong, Email: ttang@math.hkbu.edu.hk.

computation mesh suitable for the solution of finite difference/element equations. Brackbill and Saltzman [5] formulated the grid equations in a variational form to produce satisfactory mesh concentration while maintaining relatively good smoothness and orthogonality. Their approach has become one of the most popular methods used for mesh generation and adaptation. In [4], Brackbill incorporates an efficient directional control into the mesh adaptation, thereby improving both the accuracy and efficiency of the numerical schemes.

Dvinsky [9] suggests the possibility that harmonic function theory may provide a general framework for developing useful mesh generators. His method can be viewed as a generalization and extension of Winslow's method. However, unlike most other generalizations which add terms or functionals to the basic Winslow grid generator, his approach uses a single functional to accomplish the adaptive mapping. The critical points of this functional are *harmonic maps*. Meshes obtained by Dvinsky's method enjoy desirable properties of harmonic maps, particularly regularity, or smoothness.

Motivated by the work of Dvinsky, a moving mesh strategy based on harmonic mapping was proposed and studied in [15]. The key idea is to construct the harmonic map between the physical space and a parameter space by an *iteration* procedure. This procedure is simple, easy to program, and also enables us to keep the map harmonic even after long time of numerical integration. In practice, there are three types of adaptive methods using finite element approach, namely the *h-method*, *p-method*, and *r-method*. In the h-method, the overall method contains two parts, i.e. a solution algorithm and a mesh selection algorithm. These two parts are independent in the sense that the change of the PDEs will affect the first part only. However, in some of the existing r-method (also known as *moving mesh method*), these two parts are strongly associated with each other and as a result any change of the PDEs will result in the rewriting of the whole code. The algorithm proposed in [15] keeps the advantages of the r-method (e.g., keep the number of nodes unchanged) and for the h-method (e.g., the two parts in the code are independent of each other). The simplicity and reliability were demonstrated by a number of numerical examples in two space dimensions.

To completely specify the coordinates transformation, the moving mesh methods must be supplemented with suitable boundary conditions. In theory, as pointed out in [4, 6], there are a number of ways to redistribute the mesh points on the boundary, such as using homogeneous Neumann boundary conditions, extrapolating the interior mesh points to the boundary, and relocating the mesh points by solving a lower-dimensional moving-mesh PDE. However, these methods seem quite inefficient if 3D problems are being considered. In this work, we present a moving mesh method, which is based on the minimization of the mesh energy, for solving problems in three space dimensions. In the mesh-restructuring step, we solve an *optimization problem* with some appropriate constraints, which is in contrast to the traditional method of solving the Euler-Lagrange equation directly. The key idea of this approach is to treat the interior and boundary grids as a whole, rather than considering them separately. Therefore, the new solution algorithm also provides an alternative boundary grid re-distribution technique, which turns out to be useful in solving 3D problems.

The main contributions of this work are two folds: Firstly, we discuss theoretical and numerical issues in obtaining the optimal mesh by the minimization of the mesh energy. Secondly, the algorithm is applied to two nonlinear 3D partial differential equations, one scalar and one system. To our knowledge, there have existed very few moving mesh results for three-dimensional problems. It seems that some of the existing moving mesh methods have great difficulty when being implemented to 3D problems. Using the moving mesh methods proposed in this work, we are able to carry out the 3D simulations successfully. Moreover, our numerical schemes are robust to the choice of the monitor functions, with no extra efforts in choosing some user-defined parameters in the monitors. These observations again demonstrate that the moving mesh methods based on harmonic mapping are very useful in terms of simplicity and reliability.

An outline of the paper is as follows. In Section 2 we give a brief description of the numerical schemes to be used in this work. In Section 3 we describe a finite element scheme for obtaining the moving meshes based on an optimization approach. Numerical results for two- and three-dimensional problems are obtained in Section 4. Finally, Section 5 contains some concluding remarks.

2. The framework of our numerical scheme

We consider the time-dependent PDEs in a domain $\Omega \subset \mathbb{R}^n$

$$\vec{u}_t = L(\vec{u}) \quad \text{in} \quad \Omega \times (0, T] \tag{2.1}$$

with boundary and initial conditions

$$B\vec{u}|_{\partial\Omega} = \vec{u}_b \text{ in } \partial\Omega \times [0, T] \tag{2.2}$$

$$\vec{u}|_{t=0} = \vec{u}_0 \text{ in } \Omega\,. \tag{2.3}$$

To solve the above time-dependent problem, we will introduce the moving mesh method based on harmonic mapping.

2.1. **Harmonic mapping.**

A mapping based on the harmonic mapping theory is to minimize the following functional, with standard summation convention assumed,

$$E(\xi) = \sum_k \int_\Omega G^{ij} \frac{\partial \xi^k}{\partial x^i} \frac{\partial \xi^k}{\partial x^j} dx \tag{2.4}$$

see, e.g., [9, 15]. The Euler-Lagrange equation of the functional is

$$\frac{\partial}{\partial x^i} \left(G^{ij} \frac{\partial \xi^k}{\partial x^j} \right) = 0. \tag{2.5}$$

A detailed description of a moving mesh algorithm based on solving (2.5) can be found in [15].

For some physical problems, large solution gradients may exist initially or may be developed to the boundaries in a later time. As a consequence, boundary-point redistribution should be made in order to improve the quality of the adaptive mesh. With the algorithms proposed in the past, special strategies are provided to re-distribute the boundary grids. In 2D computations, one common practice is to re-distribute the grids *inside* the physical domain by solving (2.5) and to move the boundary points by solving some appropriate 1-D moving mesh equations, see, e.g. [6, 15, 21]. In other words, the grid re-distributions for the interior domain and boundaries are carried out *separately*. It seems that the extension of this approach to three space dimension is quite difficult, in particular when a finite element approach is used in envolving the underlying partial differential equations.

In the following, we will introduce a new approach which will re-distribute the interior and boundary grids *simultaneously*. To begin with, we consider physical problems in two space dimension. For simplicity, assume that Ω is a polyhedron. Let Γ_i and $\Gamma_{c,i}$ denote the corresponding edges of Ω and Ω_c, respectively. The geometrical constraint to the boundary grid movement is to keep the geometrical character of the physical domain unchanged. This implies that the vertexes (edges) of the physical domain will be mapped to the corresponding vertexes (edges) of the computational domain. Therefore, it is reasonable to consider the following mapping set from $\partial\Omega$ to $\partial\Omega_c$,

$$\begin{aligned} K \;=\; \{\xi_b \in C^0(\partial\Omega) \;\Big|\; & \xi_b : \partial\Omega \to \partial\Omega_c\,; \\ & \xi_b|_{\Gamma_i} \text{is a linear segment and strictly increasing.}\} \end{aligned} \tag{2.6}$$

One example of such a map is illustrated in Fig. 1. It follows from the theory of Eell and Simpsons [10] that for every $\xi_b \in \mathbf{K}$ there exists a unique $\xi : \Omega \to \Omega_c$, such that $\xi|_{\partial\Omega} = \xi_b$ and ξ is the extreme of the functional (2.4). Let us denote the mapping as $\xi = P(\xi_b)$, and consider the optimization problem

$$\begin{aligned} \min \quad & E(P(\xi_b)) \\ s.t. \quad & \xi_b \in \mathbf{K} \end{aligned} \tag{2.7}$$

where the functional E is defined by (2.4). Since E is convex and P is linear, it is easy to see that

$$\begin{aligned} \frac{1}{2}E\Big(P(\xi_b^{(1)})\Big) + \frac{1}{2}E\Big(P(\xi_b^{(2)})\Big) \;&\geq\; E\Big(\frac{1}{2}P(\xi_b^{(1)}) + \frac{1}{2}P(\xi_b^{(2)})\Big) \\ &\geq\; E\Big(P(\frac{1}{2}\xi_b^{(1)} + \frac{1}{2}\xi_b^{(2)})\Big). \end{aligned}$$

Therefore, $E(P(\cdot))$ is a convex functional, and as a result the optimization problem (2.7) has a unique solution in a close subset of $\mathbf{K}$.

Based on the above discussion, we will solve the following constrained optimization problem:

$$\begin{aligned} \min \quad & \sum_k \int_\Omega G^{ij} \frac{\partial \xi^k}{\partial x^i} \frac{\partial \xi^k}{\partial x^j} dx \\ s.t. \quad & \xi|_{\partial\Omega} = \xi_b \in \mathbf{K}\,. \end{aligned} \tag{2.8}$$

Note that the boundary values ξ_b are not fixed, instead they are also unknowns as same as the interior grids ξ. This is one of the main differences between the present approach and the one proposed in our earlier work [15] where a Dirichlet problem is solved for ξ. By solving the above problem, the meshes on the logical domain will be updated at each iteration (see next subsection for the iteration issue). The difference between the initial mesh and this updated mesh in the logical domain will yield the grid re-distribution for both the interior *and* boundary grids (based on the formula (3.9) to be given in the next section). One advantage of this approach is that the overall moving mesh scheme can be easily implemented for 3D problems.

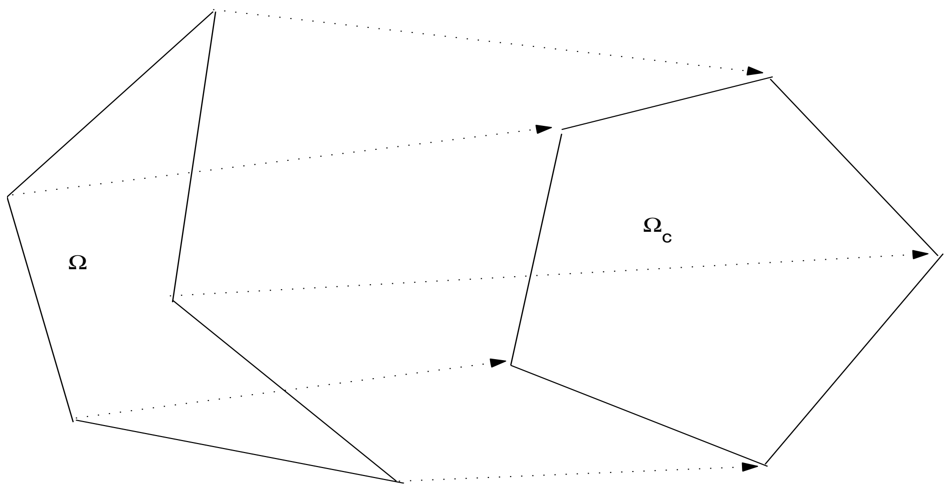


FIGURE 1. A map between $\partial\Omega$ and $\partial\Omega_c$.

2.2. **Boundary and interior grid redistribution.** To solve problem (2.1)-(2.3), we will separate the computation into two parts: mesh-moving and time-stepping. The mesh-moving is a procedure of *iteration* to construct the harmonic map between the physical mesh and the logical mesh. Each iteration step is to move the mesh *closer* to the harmonic map. In the process of the numerical computation, we always keep the initial mesh in the logical domain fixed. This mesh is not used to solve any PDEs, but its error with the solution of the optimization problem (2.8) is used to move the mesh in the physical domain. More precisely, in the first step we choose a convex domain Ω_c as the logical domain on which an initial mesh will be constructed. Traditionally (see e.g. [23, 15]), the initial mesh in the logical domain is obtained by solving the Poisson equation $\Delta\vec{\xi} = 0$ with some Dirichlet boundary condition. However, in order to match the new global approach (2.8) we generate the initial mesh by solving the following optimization problem:

$$\begin{aligned} &\min \quad \sum_k \int_\Omega \sum_i \left(\frac{\partial \xi^k}{\partial x^i}\right)^2 dx \\ &s.t. \quad \xi|_{\partial\Omega} = \xi_b \in \mathbf{K}. \end{aligned} \tag{2.9}$$

In practice, if the physical domain is convex and is of regular shape (say a convex polygon) then we can simply choose the physical domain as the logical domain with uniform initial mesh.

Once the initial mesh (in the logical domain) is given, it will be kept *unchanged* throughout the computation. The role of this initial mesh in Ω_c, denoted by $\vec{\xi}^{(0)}$, is used as a reference grid only. After the solution u is computed at time level $t = t_n$, the inverse matrix of the monitor, G^{ij} (which in general depends on u), can be updated. By solving (2.8), we will obtain a mesh in the logical domain, denoted by $\vec{\xi}^*$. If the difference between this $\vec{\xi}^*$ and the initial mesh $\vec{\xi}^{(0)}$ is not small, we move the mesh in the physical space to obtain the updated values for u in the resulting new grid based on the following principles:

- (a) obtain the error between the solution of (2.8) and the fixed (initial) mesh in the logical domain,
- (b) obtain the direction and the magnitude of the movement for $\vec{x}$ by using the error of $\vec{\xi}$, and
- (c) update $\vec{u}$ on the new grid by solving a system of ODEs.

This procedure is *repeated* until the difference between $\vec{\xi}^*$ and the initial mesh $\vec{\xi}^{(0)}$ is sufficiently small.

2.3. **Time-forwarding.** After the interior and boundary grids are well re-distributed based on the solution at $t = t_n$, we can use some appropriate numerical methods to solve the underlying PDEs at $t = t_{n+1}$ on the updated mesh in the physical space. This step is simple: it is irrelevant with the adaptive method and can be any appropriate finite element codes. The following is one of the possible methods, which will be used in our numerical experiment sections. It follows from the equation (2.1) that

$$\int_\Omega \{\vec{u}_t - L(\vec{u})\}\, v d\vec{x} = 0 \qquad \forall v \in V_T(\Omega)$$

where $V_T(\Omega)$ is a finite element space. Using the expression for $\vec{u}$ in the finite element space, i.e. $\vec{u} = U_i(t)\Phi^i(\vec{x})$, and letting v be the basis function Φ^j, we obtain

$$\int_\Omega \left\{ \frac{\partial U_i}{\partial t} \Phi^i \Phi^j - L(\vec{u})\Phi^j \right\} d\vec{x} = 0\,, \tag{2.10}$$

which is a systems of ODEs for $U_i(t)$. It can be solved by any efficient ODE solvers such as multi-stage Runge-Kutta schemes.

We point out that the method based on (2.10) only serves for the numerical experiments in this paper. In fact, this step is very flexible: any available methods/codes for the equation (2.1) can be employed in this step.

Remark 2.1. To extend the above theory to 3D, the only modification needed is to change the definition $\mathbf{K}$ in (2.6) slightly. In this case, the physical boundary is approximated by some piecewise planes which will be mapped into the corresponding boundaries in the logical domain.

3. Finite element discretization based on the optimization approach

Again, for simplicity we will demonstrate the main ideas for 2D geometry in this section. The extension to 3D is straightforward, and will be briefly outlined at the end of this section. Let us discretize the optimization problem (2.8) in the linear finite element space. The triangulation of the physical domain is $\mathcal{T}$, with T_i as its elements, and X_i as its nodes. The corresponding triangulation on the computational domain is $\mathcal{T}_c$, with $T_{i,c}$ as its elements, and $\mathcal{A}_i$ as its nodes. The linear finite element space on the mesh is denoted as $H_h^1(\Omega)$. If the basis function on node X_i is denoted by ϕ^i, then ξ can be approximated by $\xi_i\phi^i$ (here the standard summation convention is assumed). The coordinates of X_i are $(X_i^1\ X_i^2)^T$. Let the inner nodes be indexed from 1 to N_{inner} and the boundary nodes be indexed from $N_{inner}+1$ to N. The coordinates of the nodes $\mathcal{A}_i$ on computational domain is denoted as $(\mathcal{A}_i^1\ \mathcal{A}_i^2)^T$. Denote $X = (X^1\ X^2)^T$, $\mathcal{A} = (\mathcal{A}^1\ \mathcal{A}^2)^T$, in which $X^k = (X_1^k\ \cdots\ X_N^k)^T$, $\mathcal{A}^k = (\mathcal{A}_1^k\ \cdots\ \mathcal{A}_N^k)^T$, $k = 1, 2$. The objective function in (2.8) is approximated by

$$\sum_k \int_\Omega G^{ij} \frac{\partial \phi^\alpha}{\partial x^i} \frac{\partial \phi^\beta}{\partial x^j} dx \xi_\alpha^k \xi_\beta^k. \tag{3.1}$$

The weak form of the first constraint in (2.8) is

$$\int_\Omega G^{ij} \frac{\partial \xi^k}{\partial x^j} \frac{\partial \omega}{\partial x^i} dx = 0, \qquad \forall \omega \in H_0^1(\Omega)$$

which leads to

$$\int_\Omega G^{ij} \frac{\partial \phi^\alpha}{\partial x^i} \frac{\partial \phi^\beta}{\partial x^j} dx \xi_\alpha^k = 0, \quad 1 \le \beta \le N_{inner}. \tag{3.2}$$

As for the boundary points, we recall the assumption that ξ map a boundary (linear) segment L on $\partial\Omega$ to a linear segment L_c on $\partial\Omega_c$. This gives that

$$< \mathcal{A}^i, \mathbf{n}^i >= b^i \tag{3.3}$$

where $<>$ denotes the standard inner product, $\mathbf{n}^i$ is the normal direction of a fixed segment of the boundary of Ω_c and b^i is a given number. Since each $X^i \in \Gamma_i$ is mapped to a known segment of $\Gamma_{i,c}$, the relevant $\mathbf{n}^i$ and b^i are determined.

In the following, we will form a linear system for $\mathcal{A}$. Denote

$$H = \left(\int_\Omega G^{ij} \frac{\partial \phi^\alpha}{\partial x^i} \frac{\partial \phi^\beta}{\partial x^j} dx \right)_{1 \le \alpha,\beta \le N.} \tag{3.4}$$

We further split the matrix H into the following form:

$$\begin{array}{cccc} & & & \\ H = & \left(\begin{array}{cc} H_{11} & H_{12} \\ H_{21} & H_{22} \end{array} \right) & & \begin{array}{l} \leftarrow 1 \text{ to } N_{inner} \text{ row} \\ \leftarrow N_{inner}+1 \text{ to } N \text{ row} \end{array} \\ & \begin{array}{cc} \uparrow & \uparrow \\ 1 \text{ to } N_{inner} \text{ column} & N_{inner}+1 \text{ to } N \text{ column} \end{array} & & \end{array}$$

Correspondently, we use X_{inner}, X_{bound}, $\mathcal{A}_{inner}$ and $\mathcal{A}_{bound}$ to denote the inner and boundary part of the node coordinates. Then the objective function is given by

$$\begin{pmatrix} \mathcal{A}^{1,T} & \mathcal{A}^{2,T} \end{pmatrix} \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \begin{pmatrix} \mathcal{A}^1 \\ \mathcal{A}^2 \end{pmatrix}. \tag{3.5}$$

Recall the earlier assumption that ξ maps a (linear) boundary segment L on $\partial\Omega$ to a linear segment L_c on $\partial\Omega_c$. This assumption leads to the following linear system

$$\begin{pmatrix} 0 & A_{12} & 0 & A_{22} \end{pmatrix} \begin{pmatrix} \mathcal{A}^1_{inner} \\ \mathcal{A}^1_{bound} \\ \mathcal{A}^2_{inner} \\ \mathcal{A}^2_{bound} \end{pmatrix} = \vec{b}. \tag{3.6}$$

With the above preparation, the optimization problem (2.8) is equivalent to solving the following linear system:

$$\begin{pmatrix} H_{11} & H_{12} & 0 & 0 & 0 \\ H_{21} & H_{22} & 0 & 0 & A'_{12} \\ 0 & 0 & H_{11} & H_{12} & 0 \\ 0 & 0 & H_{21} & H_{22} & A'_{22} \\ 0 & A_{12} & 0 & A_{22} & 0 \end{pmatrix} \begin{pmatrix} \mathcal{A}^1_{inner} \\ \mathcal{A}^1_{bound} \\ \mathcal{A}^2_{inner} \\ \mathcal{A}^2_{bound} \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vec{b} \end{pmatrix} \tag{3.7}$$

where λ is the Lagrange multiplier. After obtaining the solution of (3.7), we can solve a linear system in the form of (4.5) in [15] to obtain $\partial\vec{x}/\partial\xi$ in each element E. If we take the volume of the element as the weight, the weighted average error of X at the i-th node is defined by

$$\delta X_i = \frac{\displaystyle\sum_{E\in T_i} |E| \left.\frac{\partial\vec{x}}{\partial\xi}\right|_{in\ E} \delta\mathcal{A}_i}{\displaystyle\sum_{E\in T_i} |E|} \tag{3.8}$$

in which $|E|$ is the volume of the element E. It can be shown that the above volume weighted average converges to a smooth solution in measure when the size of mesh goes to 0. The location of the nodes in the new mesh T^* on the physical domain is taken as

$$X^* = X + \tau\delta X \tag{3.9}$$

in which τ is a parameter in $[0,1]$. The motion of one element based on (3.9) is illustrated in Fig. 2. The rest of the numerical implementations are the same as that described in Section 5 of [15], except the generation of the initial mesh in the computational domain. With the present optimization approach, the initial logical mesh is prepared by using a free slip boundary condition rather than the Dirichlet boundary condition. In other words, the logical mesh is obtained by solving the optimization problem (2.9).

Remark 3.1. Computational complexity. Let us make some comparison on operation numbers between the method used in [15] and the one in this work. In the former approach, the major work in obtaining the new grids are to find $\mathcal{A}^1_{inner}$ and $\mathcal{A}^2_{inner}$ by solving

$$\begin{pmatrix} H_{11} & 0 \\ 0 & H_{11} \end{pmatrix} \begin{pmatrix} \mathcal{A}^1_{inner} \\ \mathcal{A}^2_{inner} \end{pmatrix} = \begin{pmatrix} \vec{g}_1 \\ \vec{g}_2 \end{pmatrix} \tag{3.10}$$

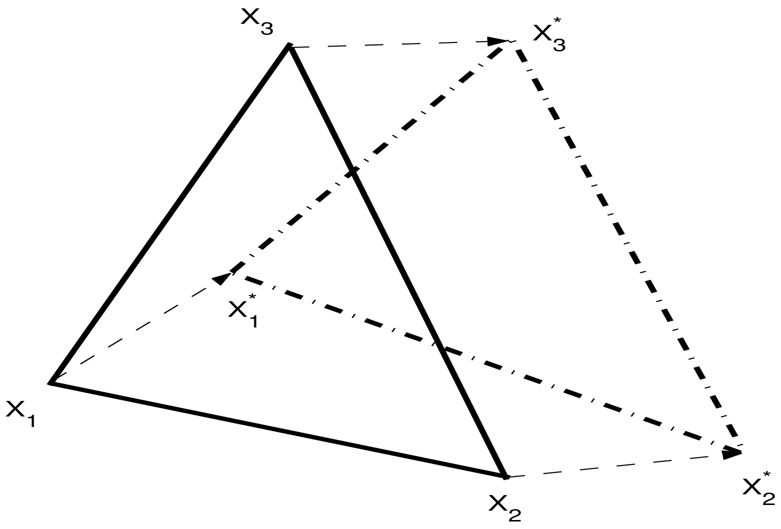


FIGURE 2. A demonstration of the element motion.

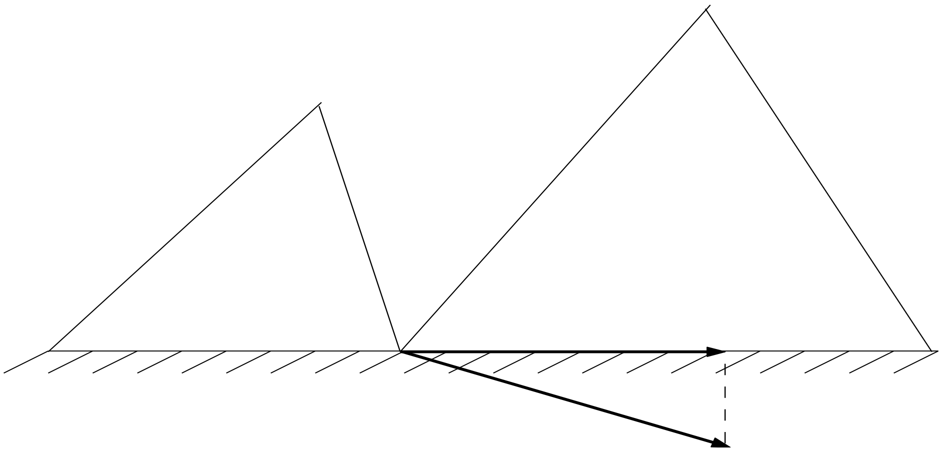


FIGURE 3. A demonstration of boundary projection.

where the right-hand side is some given vector. Since the sizes of $H_{21}, H_{12}, H_{22}, A_{12}$ and A_{22} are substantially smaller that that for H_{11}, the workload for the method used in [15] and the one in this work is similar.

Remark 3.2. Boundary projection. In practical computation, we also need to project the moving vector of the boundary nodes onto the boundary. Otherwise, the residual error accumulation will cause the boundary nodes away from the physical boundary. Although the direction of the motion on the logical boundary is always following the boundary, the interpolation using the formula (3.8) may push the boundary points away from the physical boundary. One of the remedies is to project them back to the physical domain, as illustrated in Fig. 3.

4. Numerical experiments

In this section, we will implement the moving mesh method described in the last two sections to some test problems. To make some useful comparisons, we begin by revisiting some 2D examples whose moving mesh results have been obtained by several authors, see, e.g. [6, 17, 15].

4.1. **2D examples.**

Example 4.1. *Our first example in 2D is to compute a moving oblique shock whose governing equation is the Burgers' equation*

$$\frac{\partial U}{\partial t} + UU_x + UU_y = a\Delta U$$

defined in the unit square $\Omega = (0,1)^2$*. The initial condition and Dirichlet boundary condition are chosen such that the exact solution to the underlying problem is*

$$U(x, y; t) = \Big(1 + \exp((x + y - t)/(2a)\Big)^{-1}.$$

In our computation the viscosity coefficient is chosen as $a = 0.005$, and the monitor function is chosen as $\sqrt{1 + |\nabla U|^2 I}$. It is noted that the smaller a is, the more convection dominates, and the higher the concentration of mesh points required around the wave front, which makes the use of the moving mesh methods meaningful. Since the exact solution of the underlying problem is given, it is convenient to compare the errors obtained by various methods.

In [15], a simple redistribution strategy is proposed as follows. The basic idea is to move the boundary points by solving 1-D moving mesh equations (refer to as the 1D boundary grid redistribution method). Without lose of generality, we consider a simple boundary $[a, b]$ in the x-direction. Solving the two-point boundary value problem for $(\omega x_\xi)_\xi = 0$ with uniform mesh in ξ will lead to a new boundary redistribution. Assume $[x_j, x_{j+1}] \subset [a, b]$. Then there exists exactly one element T_j whose one edge is $[x_j, x_{j+1}]$. Note that the gradient monitor in T_j is a constant (due to the use of the linear element). We let the monitor function $\omega|_{[x_j,x_{j+1}]}$ equals to this constant, which establishes a connection between the boundary and interior grid redistributions.

In Fig. 4, we plot the moving meshes obtained by using the moving mesh approach proposed in this work with 20×20 nodes (about 930 triangular elements). It is observed that they are very similar to those obtained in [15] where the meshes are moved by solving the Euler-Lagrange equations. In Fig. 5, the L^1-errors obtained by using the two approaches are compared, with no significant difference observed.

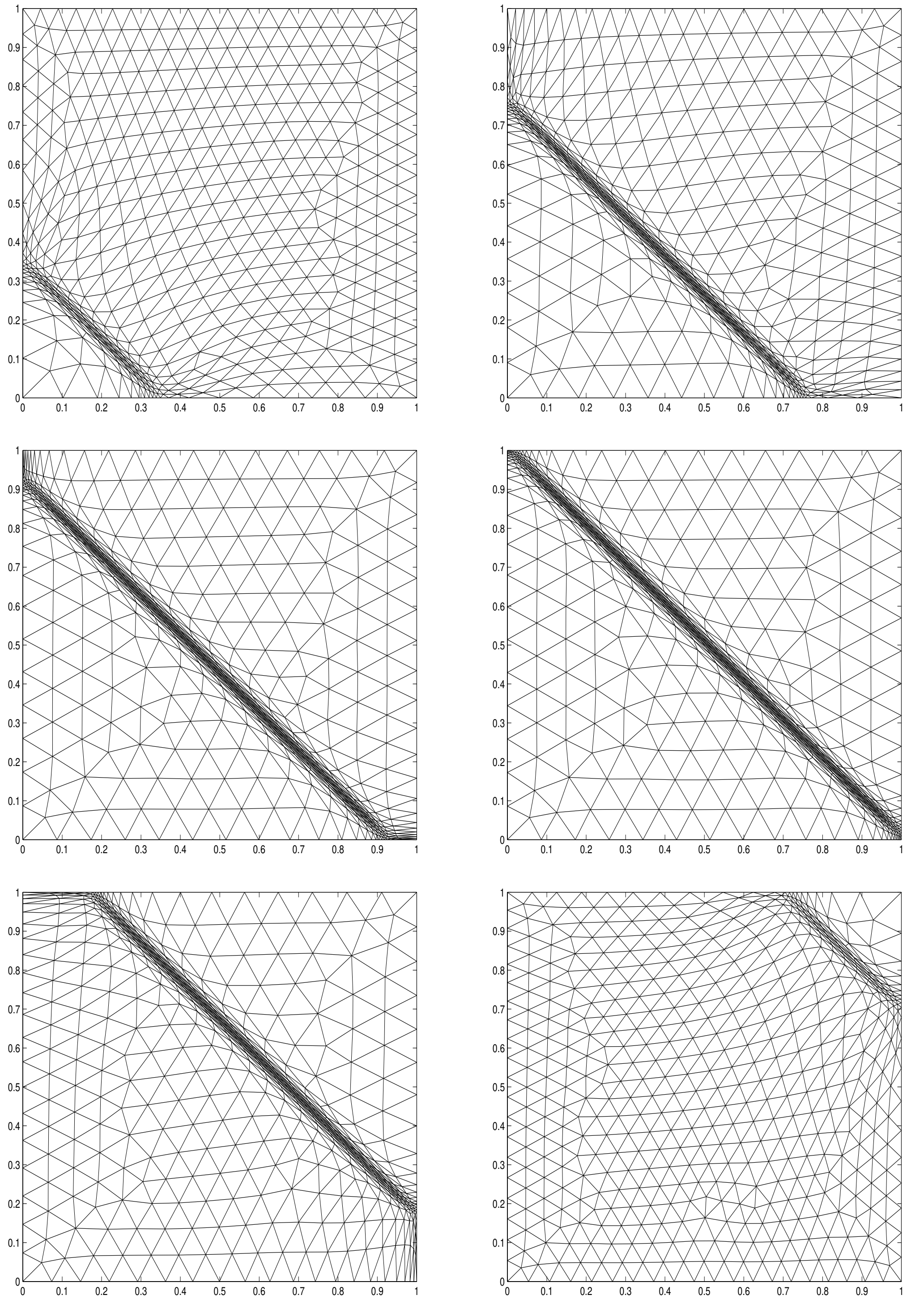


FIGURE 4. Example 4.1: The adaptive meshes using 20^2 nodes from $t = 0.15$ to $t = 1.85$, with the moving mesh approach proposed in this work.

Our next numerical example is a coupled nonlinear reaction-diffusion system modeling a combustion process [1]. The main purpose of this example is to demonstrate that for some

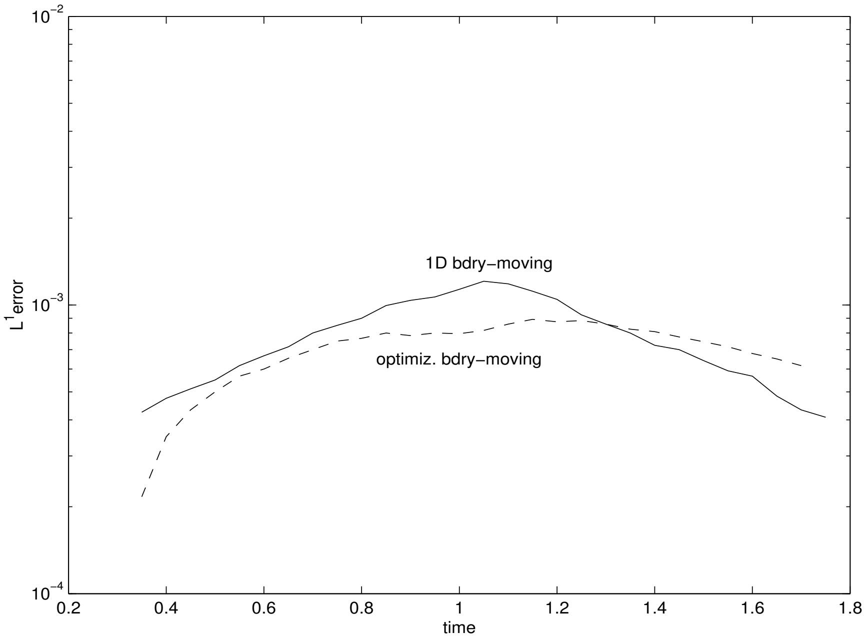


FIGURE 5. Example 4.1: The L^1-error in time obtained by using the method of [15] (solid line) and the method proposed in this work (dashed line), with 20×20 nodes.

problems there is indeed some difference between the moving mesh methods described in [15] and in this work.

Example 4.2. *The mathematical model is a system of coupled nonlinear reaction-diffusion equations:*

$$\frac{\partial u}{\partial t} - \nabla^2 u = -\frac{R}{\alpha\delta} u e^{\delta(1-1/T)},$$
$$\frac{\partial T}{\partial t} - \frac{1}{Le}\nabla^2 T = \frac{R}{\delta Le} u e^{\delta(1-1/T)},$$

for $\vec{x} = (x_1, x_2)^T \in \Omega = (-1,1)^2, t > 0$ *and where* u *and* T *represent respectively the dimensionless species concentration and temperature of a chemical which is undertaking a one-step reaction. The initial and boundary conditions are*

$$u|_{t=0} = T|_{t=0} = 1, \qquad in\ \Omega,$$
$$u|_{\partial\Omega} = T|_{\partial\Omega} = 1, \qquad for\ t > 0.$$

The physical parameters are set to be $Le = 0.9, \alpha = 1, \delta = 20$ and $R = 5$. This problem has sharp wave front moving towards the boundary $\partial\Omega$. For this problem, a detailed numerical procedure for time discretization was provided in [15]. In our computation, we used 30×30 nodes and the monitor function is chosen as $\sqrt{1+|\nabla T|^2}\, I$. Fig. 6 depicts the adaptive meshes at $t = 0.283$ obtained by solving the Euler-Lagrange equation together with the 1D boundary grid-redistribution technique and by the method proposed in this work. It is observed that both methods can efficiently solve this nonlinear system with large solution gradients. However, it is quite obvious that the meshes obtained by using

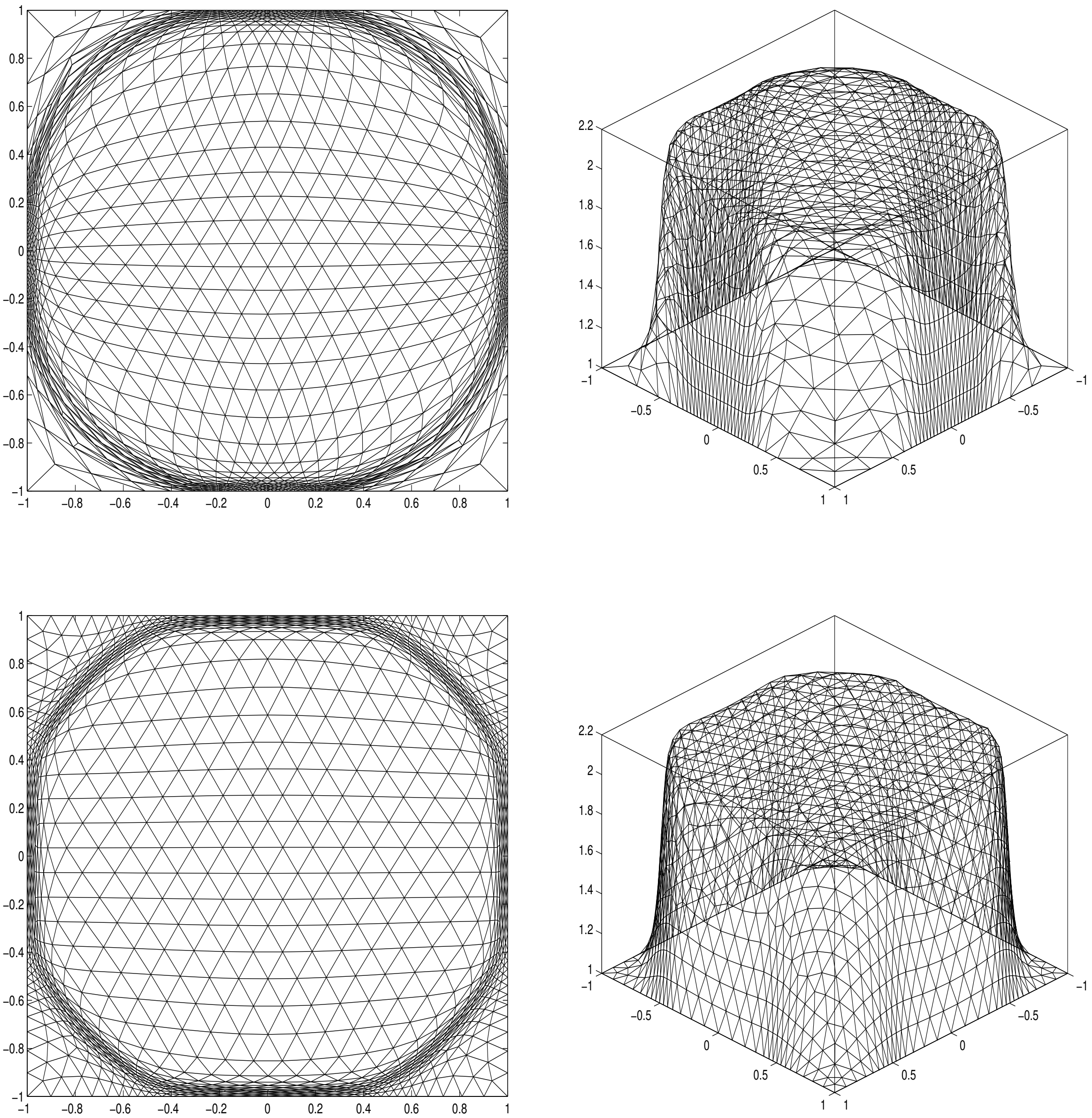


FIGURE 6. Example 4.2: The adaptive mesh and temperature T obtained by using the method of [15] (top) and the one proposed in this work (bottom), at $t = 0.283$.

the optimization method appear to be more reasonable, which adapts more grid points into the reaction interface. The solution errors, obtained by using a benchmark solution with very fine-mesh, are compared in Fig. 7. The results indicate that the L^1 and L^∞ errors are reduced by using the optimization approach, although the reduction is not significant.

4.2. **3D examples.** For simplicity, we assume the solution domain is a unit cube $[0, 1]^3$. Initially, the physical domain is divided into some small cubes (with same size), and then every cube is cut into six tetrahedron. In this case, the logical domain and its initial mesh are chosen as the same as the corresponding counterpart in the physical domain.

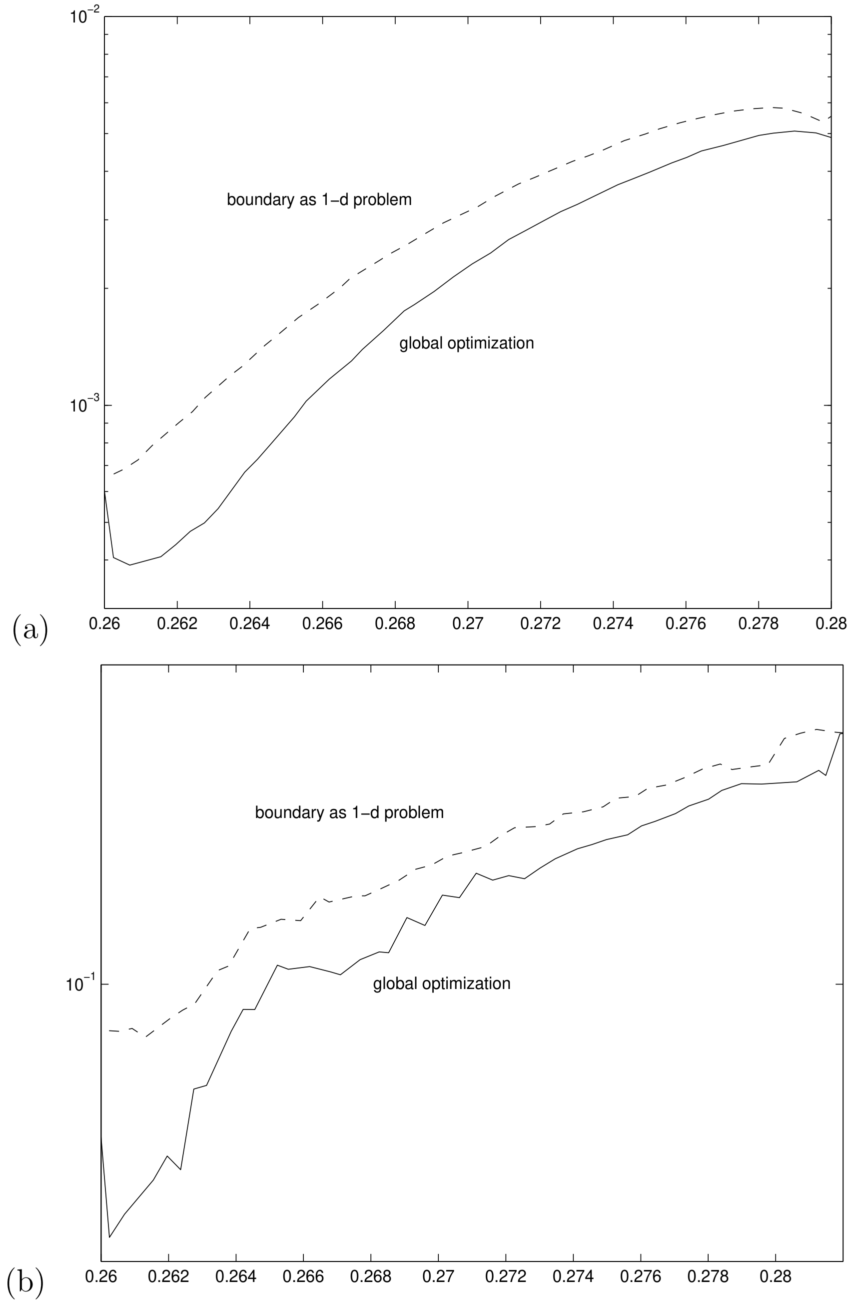


FIGURE 7. Numerical errors for Example 4.2 obtained by using the 1D boundary-grid re-distribution (dashed line) and by using the optimization approach proposed in this work (solid line): (a) L^1-error and (b) L^∞-error.

In order to avoid possible mesh tangling, the following strategy is proposed. In the mesh-motion step, consider a tetrahedron with vertexes $\vec{x}_i$, whose moving directions are $\delta\vec{x}_i$, $0 \le i \le 3$. It can be shown that the mesh may be tangled if the step length τ used in the mesh motion formula (3.9) is larger than a critical number τ^* which is defined as the least positive root for the equation

$$\det \begin{pmatrix} 1 & 1 & 1 & 1 \\ \vec{x}_0 - \tau\delta\vec{x}_0 & \vec{x}_1 - \tau\delta\vec{x}_1 & \vec{x}_2 - \tau\delta\vec{x}_2 & \vec{x}_3 - \tau\delta\vec{x}_3 \end{pmatrix} = 0.$$

The parameter τ used in (3.9) will be set as half of the minimal τ^* over all the tetrahedron. In our numerical computations, it is found that the selection of $\tau = 0.5$ for (3.9) causes no numerical difficulties (i.e. no mesh tangling) in almost all situations.

Example 4.3. *Our first example in 3D is concerned with the Burgers' equation*

$$\frac{3}{2}\frac{\partial U}{\partial t} + UU_x + UU_y + UU_z = a\Delta U$$

defined in $\Omega = (0,1)^3$. *The initial condition and Dirichlet boundary condition are chosen such that the exact solution to the underlying problem is*

$$U(x,y,z;t) = \Big(1 + \exp((x+y+z-t)/2a)\Big)^{-1}.$$

In our computation, the diffusion coefficient is again chosen as $a = 0.005$, and the monitor function is chosen as $\sqrt{1+|\nabla U|^2}\, I$. For this small coefficient a, the underlying physical problem corresponds to s viscous shock traveling from the left down corner to the right-up corner. Note that the exact solution is independent of the independent variable z, and the solution has very large gradients along the 2D plane $x+y+z=t$ for $0 < t < 3$.

In Fig. 8, the L^1- and L^∞ errors obtained by using the uniform mesh and the moving mesh are displayed. In both cases, the solution domain is divided into $17 \times 17 \times 17$ small cubes, and then every cube is cut into six tetrahedron. For the moving mesh method, it is observed that the L^1-error, as a function of time, is almost oscillation free, and in average is about 10 times smaller than that for the uniform mesh. For viscous shock problems, it is expected that the L^∞-error may not be reduced with the moving mesh approach, but it is observed from Fig. 8(b) that the moving mesh solution yields about 4 times error reduction. In Fig. 9, the grid distribution at various times is demonstrated. In the red color region, the numerical solution is approximately 1, and in the blue zone it is about 0. The green zone is the transition region where ideally more grids should be placed. In fact, Fig. 9 clearly indicate that our moving mesh algorithms cluster quite a number of grid points at this region where the solution has the largest gradients. It is also seen that the *locations* of the viscous shock at various time are clearly presented in Fig. 9, which provides one of the most useful information in the viscous shock problems.

Example 4.4. *The last example is the same as Example 4.2, except that the solution domain becomes* $\Omega = (-1,1)^3$.

In Example 4.3, the viscous Burgers' problem is a scalar equation whose largest solution gradients occur in a simple 2D plane. In Example 4.4, the problem is a nonlinear *system* whose largest solution gradients occur in a quite complicated surface, which is more or less like a sphere $x^2+y^2+z^2=r^2$, where the radius r is a function of time. We solve the underlying PDEs by simply extending the 2D finite element codes developed in [15]: we first consider a finite element spatial discretization, leaving the time variable continuous (method of lines); and then a RK3 is used for the time discretization.

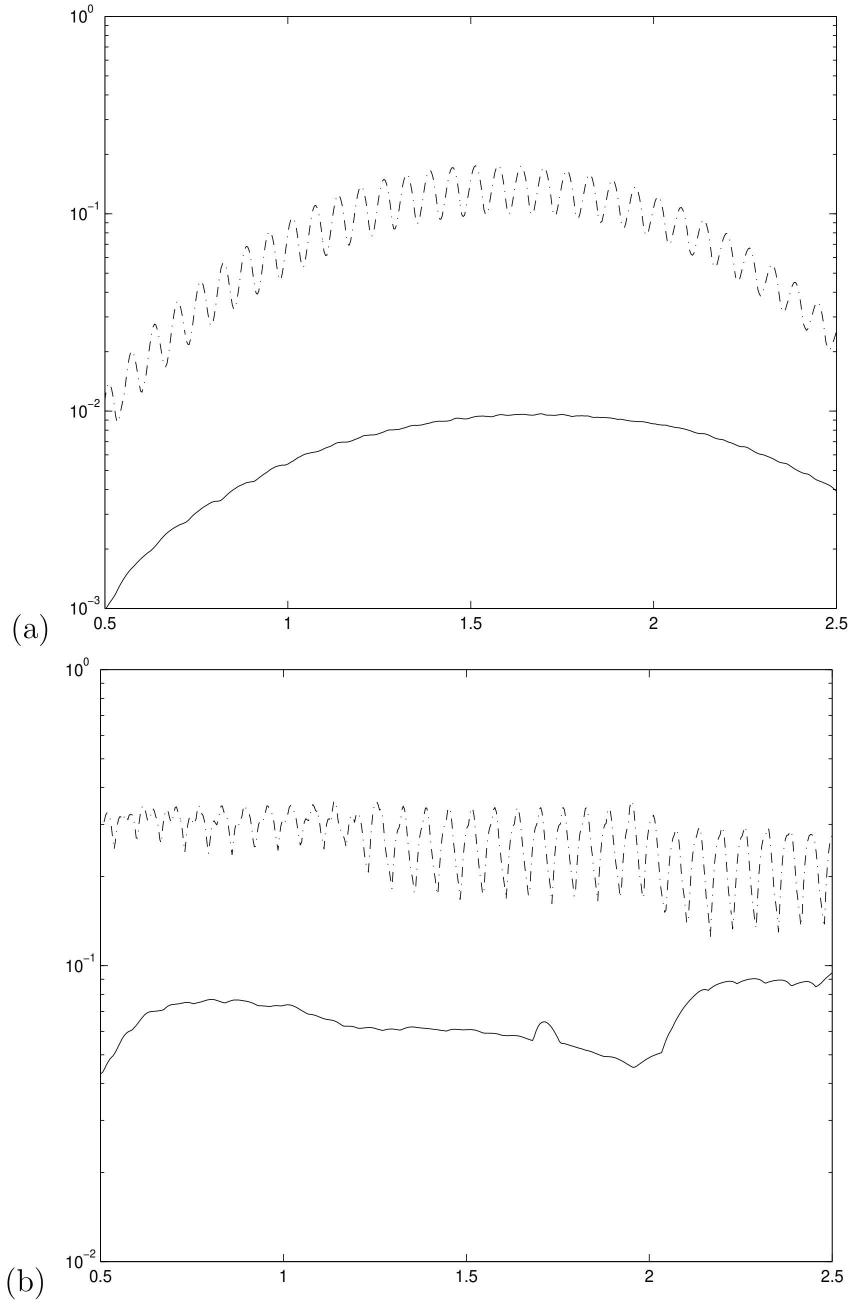
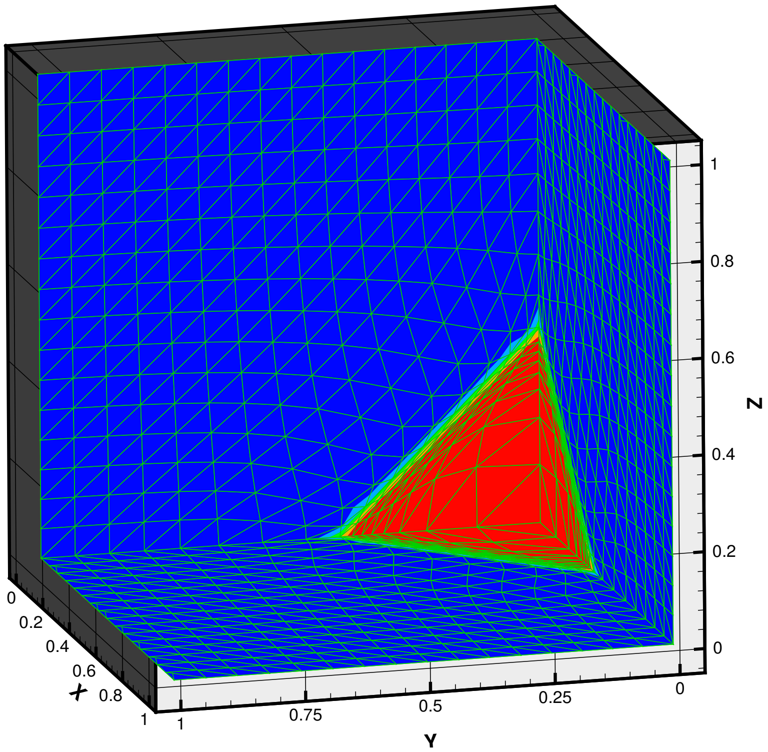
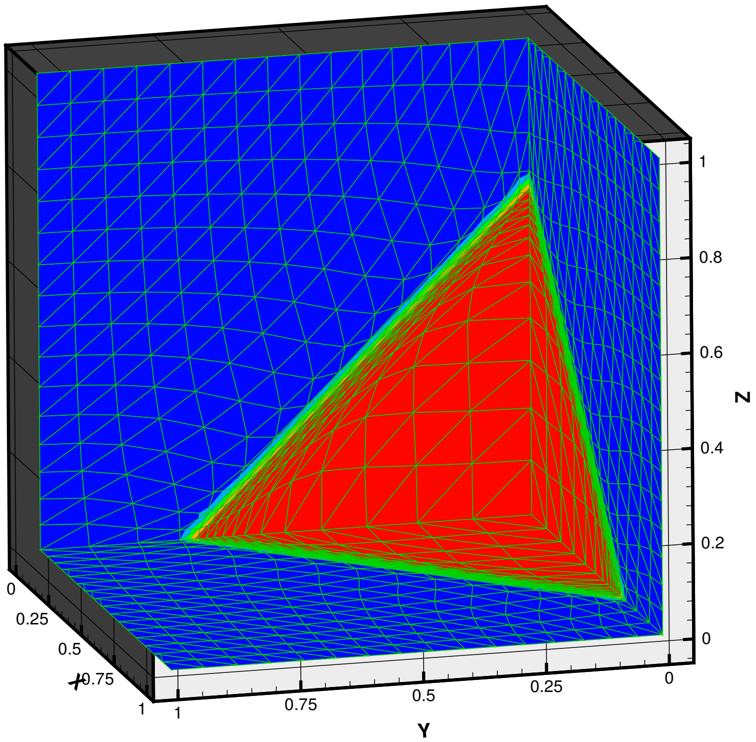


FIGURE 8. Numerical errors for 3D Burgers' equation: (a) L^1-error and (b) L^∞ error. In each figure, the top curve is with the uniform mesh and the bottom is with the moving mesh.

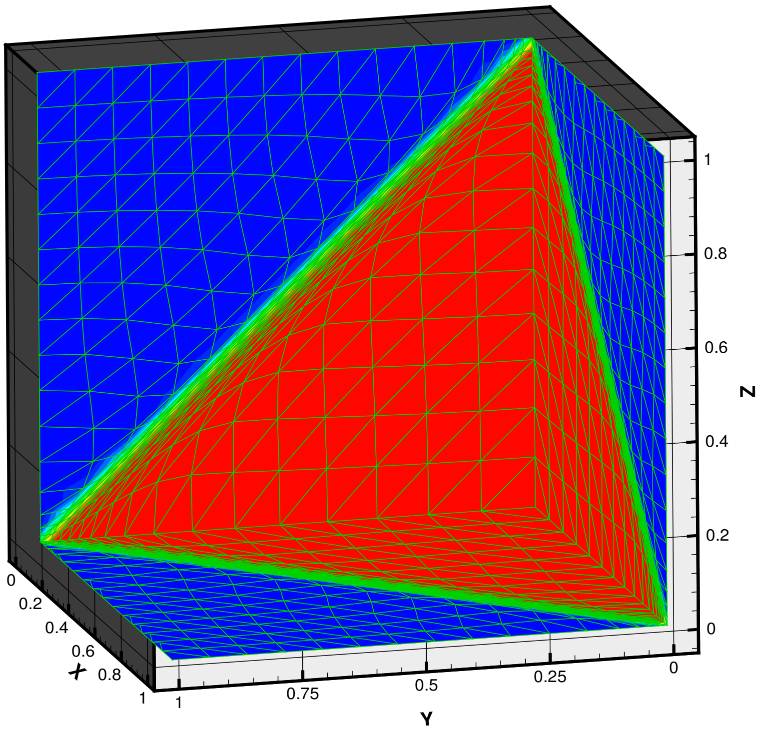
In Figs. 10 and 11, the moving meshes from different angles obtained by using the same number of nodes as in the last example are plotted, together in the figures are the numerical solutions for the temperature T at various time levels. In the red color region, the numerical solution for the temperature is 1, and in the blue zone it is a constant greater than 1. It is observed that although the temperature T has a very thin layer of large variation, our moving mesh scheme adapts the mesh extremely well to the regions


X
0
0.2
0.4
0.6
0.8
1
Y
1
0.75
0.5
0.25
0
Z
0
0.2
0.4
0.6
0.8
1

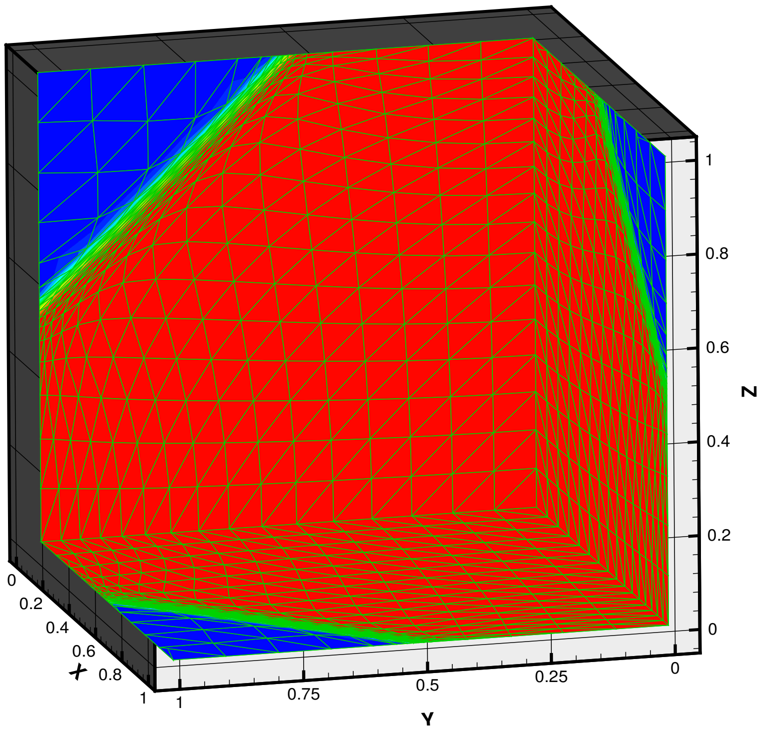
(a)


X
0
0.25
0.5
0.75
1
Y
1
0.75
0.5
0.25
0
Z
0
0.2
0.4
0.6
0.8
1

(b)



(c)



(d)

FIGURE 9. The grid distribution for Example 4.3 at (a): $t = 0.4$, (b): $t = 0.7$, (c): $t = 1$ and (d): $t = 1.5$.

with large solution gradients. Unlike the regular finite element meshes, an arbitrary cut-plane may not have enough grid points for the moving meshes. With Tech Plot software, the nearby points are projected to the chosen plane. As observed in Figs. 10 and 11, the projection may yield a few *spurious* elements, since only (in average) 17×17 nodes on each given plane.

We close this section with some discussions on the monitor function $(G^{ij})^{-1}$. There are several papers investigating the selection of the monitor functions, including Beckett et al. [2, 3], Brackbill [4] and Huang and Russell [12]. One observation in our numerical experiments is that the numerical algorithm developed in this work is robust for the selection of the monitor functions. For both the viscous shock and the combustion problems, a simple monitor function $\sqrt{1 + c|\nabla u|^2}\, I$ works very well, where u is the underlying physical solution and $c > 0$ is some constant. In our numerical experiences, some moving mesh methods (such as moving finite-volume method, see, e.g., [21]) depend heavily on the choice of feasible values of c, in particular for three-dimensional computations. However, with the moving mesh algorithm developed in this work, this has never been found a problem: we can always choose $c = 1$, and the numerical solutions are quite insensitive to the choice of the parameter c.

5. Concluding remarks

In this work, we have extended the moving mesh methods based on harmonic maps to deal with mesh adaptation in three space dimensions. The main difference between the numerical scheme proposed in this work and the one proposed in [15] is the following: In obtaining the variational mesh, we solve an *optimization problem* with some boundary-point constraints; but in [15] the moving meshes are obtained by solving the Euler-Lagrange equation, together with the boundary-point adjustments. In the latter case, the grid re-distributions for the interior domain and boundaries are carried out *separately*. However, this approach seems difficult when being extended to 3D problems, in particular if a finite element approach is used in spatial discretizations. In this work, by solving the constrained optimization problem we are able to implement the moving mesh methods to three dimensional problems successfully.

The next step of the research is to apply our moving mesh algorithm to solve some practical problems. Some useful applications based on various moving mesh techniques can be found in a number of papers, including [7, 18, 19, 21] for computational fluid dynamics, [8, 13] for computational geometry and [11, 16, 22] for grid generation. In solving real-life problems with moving mesh strategy some additional difficulties may occur. For example, when using the moving mesh methods to solve hyperbolic system of conservation laws, the additional requirements such as mass conservation and entropy consistency may introduce extra numerical difficulties in the mesh moving process. As a result, there are only very few successful moving mesh results for multi-dimensional hyperbolic system, in particular for 3D. In our future study, we will apply the moving mesh algorithms developed in this work to some practical problems in areas such as fluid mechanics, chemical engineering and image processing. Our particular attention will be given to solving the practical problems in higher space dimensions.

Acknowledgment: This work was supported by the Special Funds for Major State Basic Research Projects of China (grant # G1999032804) and the Hong Kong Research Grants Council (grant # 2033/99P and 2044/00P). Part of this research was carried out while the first author was visiting the Hong Kong Baptist University.

References

[1] S. Adjerid and J. E. Flaherty, A moving finite element method with error estimation and refinement for one-dimensional time dependent partial differential equations. *SIAM J. Numer. Anal.*, **23**, 778-796 (1986).

[2] G. Beckett, J. A. Mackenzie, A. Ramage and D. M. Sloan, On the numerical solution of one-dimensional PDEs using adaptive methods based on equidistribution, *J. Comput. Phys.* 167 (2001), pp. 372-392.

[3] G. Beckett, J. A. Mackenzie and M. L. Robertson, A moving mesh finite element method for the solution of two-dimensional Stephan problems, *J. Comput. Phys.* 168 (2001), pp. 500-518.

[4] J.U. Brackbill, An adaptive grid with directional control, *J. Comput. Phys.* 108 (1993), pp. 38-50.

[5] J.U. Brackbill and J.S. Saltzman, Adaptive zoning for singular problems in two dimensions, *J. Comput. Phys.* 46 (1982), pp. 342–368.

[6] 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), pp. 221–244.

[7] H. D. Ceniceros and T. Y. Hou, An efficient dynamically adaptive mesh for potentially singular solutions. To appear in *J. Comput. Phys.*, 2001.

[8] V. Critini, J. Blawzdziewicz and M. Loewenberg, An adaptive mesh algorithm for evolving surfaces: simulations of drop breakup and coalescence, *J. Comput. Phys.* 168 (2001), pp. 445-463.

[9] A.S. Dvinsky, Adaptive grid generation from harmonic maps on Riemannian manifolds, *J. Comput. Phys.* 95 (1991), pp. 450–476.

[10] J. Eell and J.H. Sampson, Harmonic mappings of Riemannian manifolds, *Amer. J. Math.* **86**, 109-160 (1964).

[11] Ph. Hoch and M. Rascle, Hamilton-Jacobi equations on a manifold and applications to grid generation or refinement. Private communication.

[12] W.Z. Huang and R.D. Russell, Moving mesh strategy based on a gradient flow equation for two-dimensional problems, *SIAM J. Sci. Comput.* **20**, 3, 998–1015 (1999).

[13] S. Kawk and C. Pozrikidis, Adaptive triangulation of evolving, closed, or open surfaces by the advancing-front method, *J. Comput. Phys.* **145**, 61 (1998).

[14] R. Li, *Moving Mesh Method and its Application*, PhD Thesis (in Chinese), School of Mathematical Sciences, Peking University, May 2001.

[15] R. Li, T. Tang, and P.W. Zhang, Moving mesh methods in multiple dimensions based on harmonic maps, *J. Comput. Phys.*, **170**, 562-588 (2001).

[16] G. Liao, F. Liu, C. de la Pena, D. Peng and S. Osher, Level-set-based deformation methods for adaptive grids. *J. Comput. Phys.* **159**, 103-122 (2000).

[17] P.K. Moore and J.E. Flaherty, Adaptive local overlapping grid methods for parabolic systems in two space dimensions, *J. Comput. Phys.* **98**, 54–63 (1992).

[18] R. Pember, J. Bell, P. Collela, W. Crutchfield and M. Welcome, An adaptive cartesian grid method for unsteady compressible flow in irregular regions, *J. Comput. Phys.*, **120**, 278-304 (1995).

[19] W. Ren and X.-P. Wang, An iterative grid redistribution method for singular problems in multiple dimensions, *J. Comput. Phys.*, **159**, 246-273 (2000).

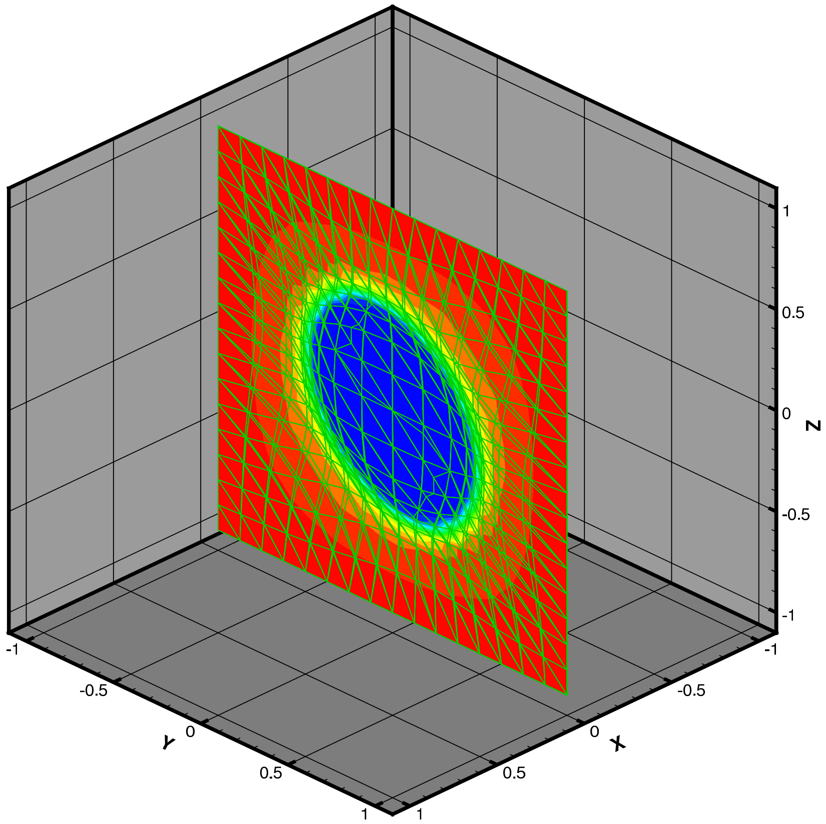
[20] R. Schoen and S.-T. Yau, On univalent harmonic maps between surfaces, *Invent. Math.* **44**, 265-278 (1978).

[21] H.-Z. Tang and T. Tang, Moving mesh methods for one- and two-dimensional hyperbolic conservation laws, Submitted to *SIAM J Numer. Anal.*, 2001. Also available in `http://www.math.ntnu.no/conservation/2001/014.html.`

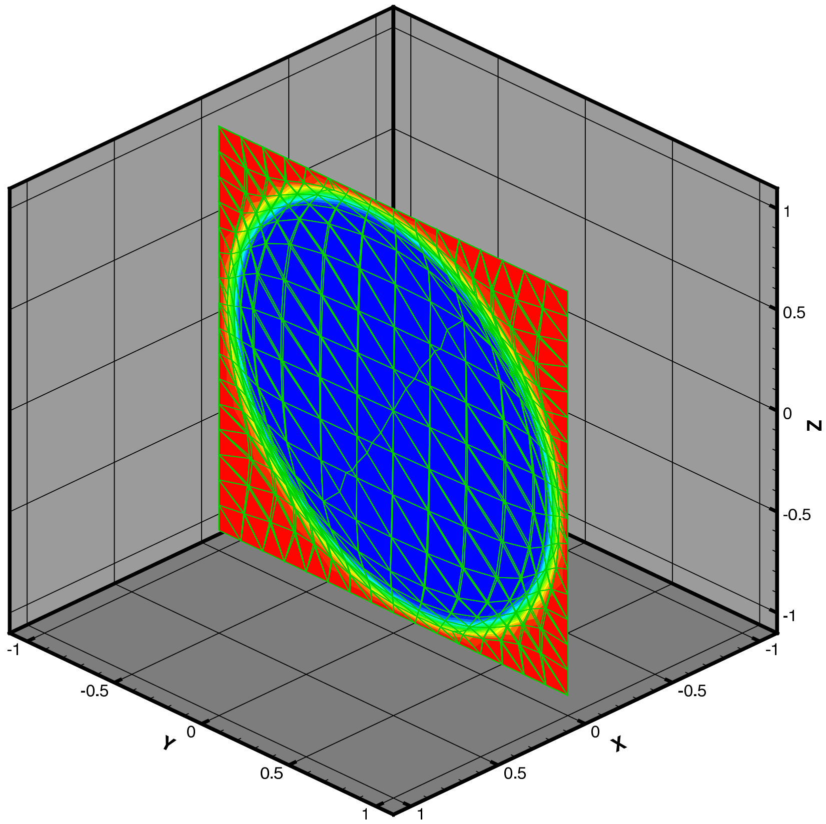
[22] J F. Thompson, Z.U A. Warsi and C. W. Mastin, *Numerical Grid Generation* (North Holland, New York, 1985).

[23] A. Winslow, Numerical solution of the quasi–linear Poisson equation, *J. Comput. Phys.* 1 (1967), pp.149–172.

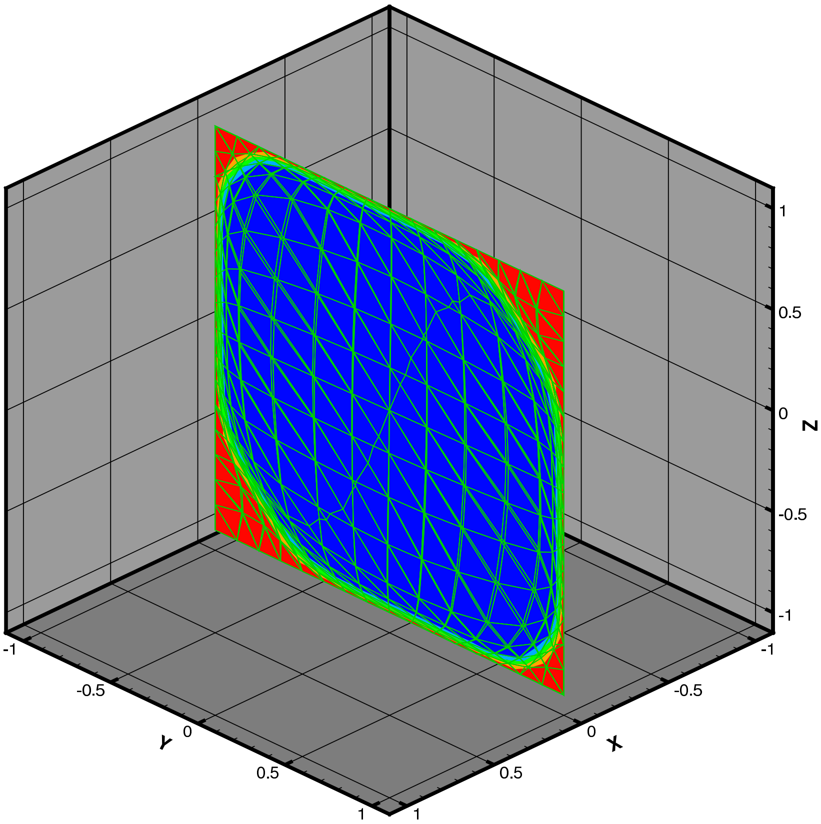
(a)



(b)



(c)



(d)

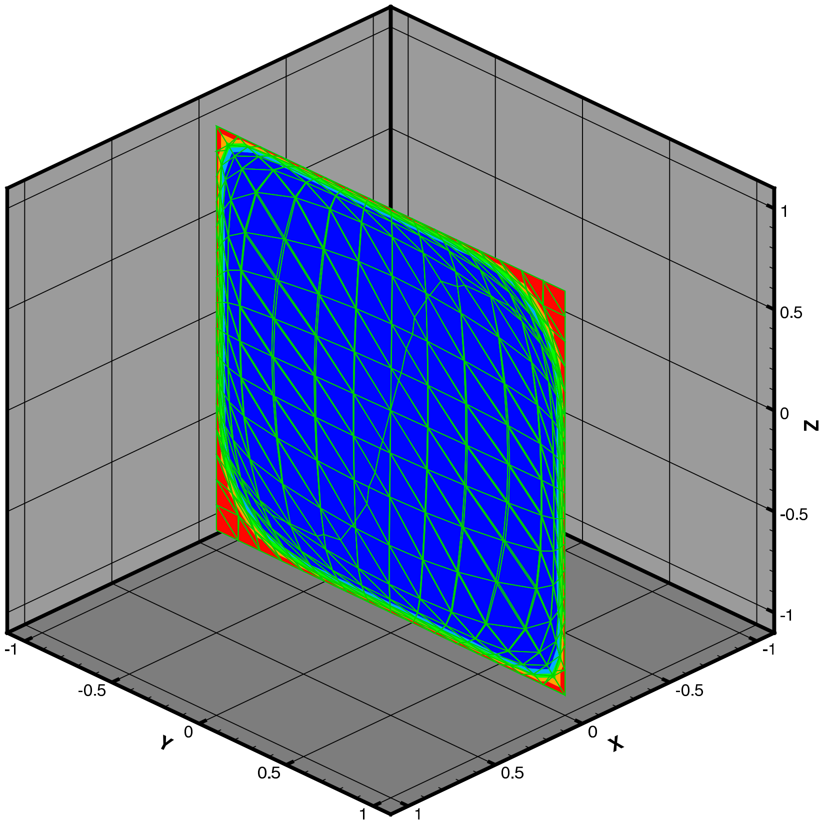
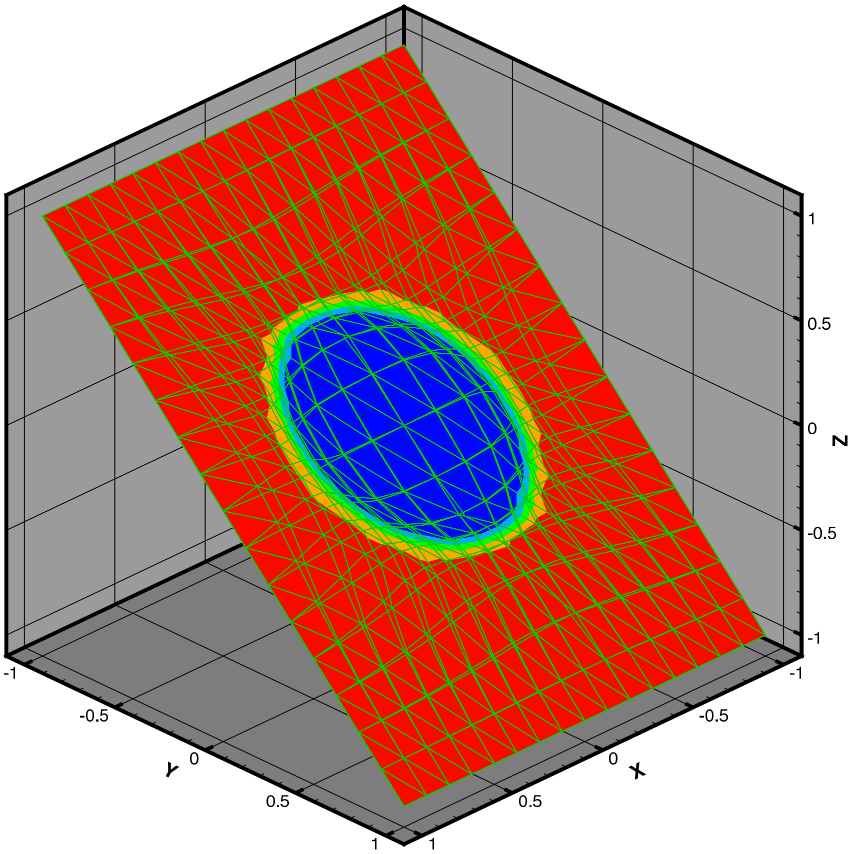
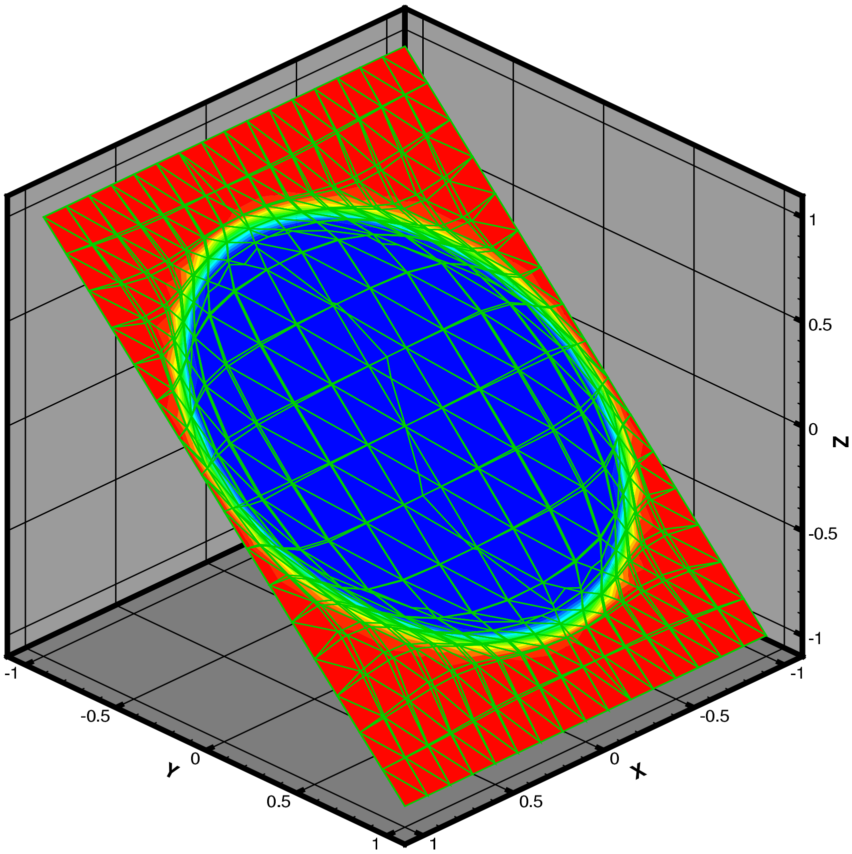


FIGURE 10. Example 4.4: The grid distribution on the middle cut-plane parallel to the y-z plane at (a): $t = 0.312$, (b): $t = 0.324$, (c): $t = 0.331$ and (d): $t = 0.335$.

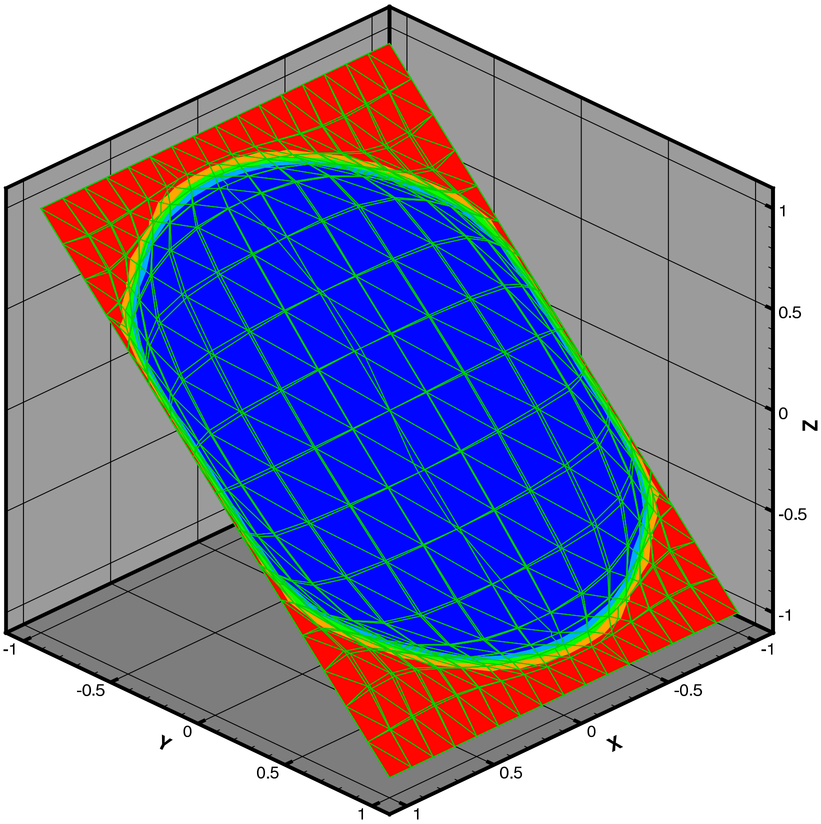
(a)



(b)



(c)



(d)

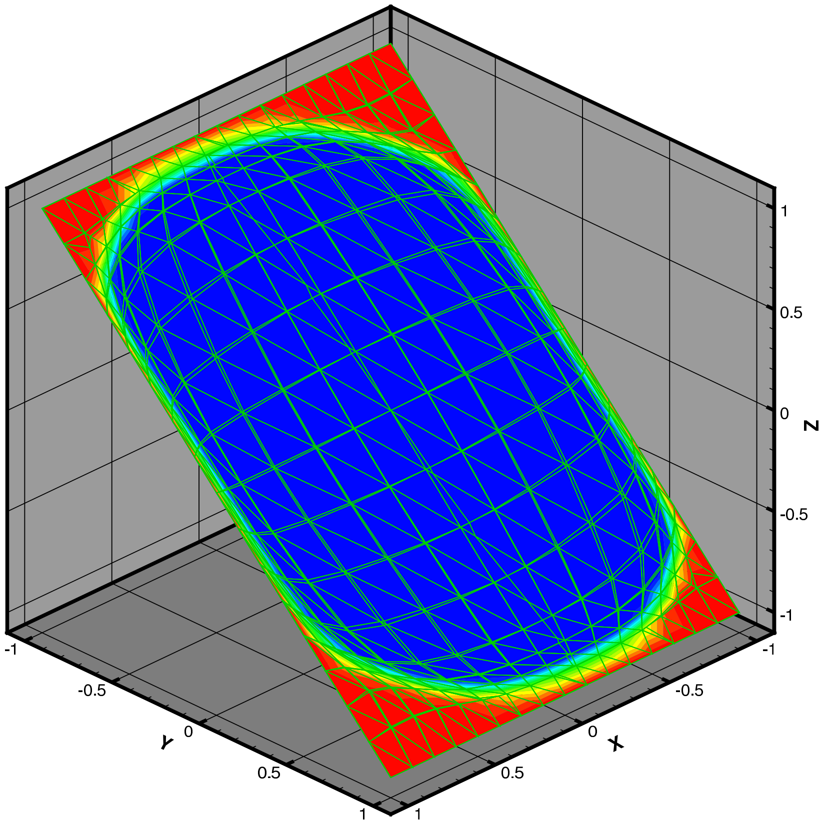


FIGURE 11. Same as Fig. 10, except that the cut-plane is rotated 45^o.