INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2009; **61**:1161–1178
Published online 23 February 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.2008

Moving mesh methods for Boussinesq equation

Lee Wan-Lung*,† and Zhijun Tan

Singapore-MIT Alliance, 4 Engineering Drive 3, National University of Singapore, Singapore 117576, Singapore

SUMMARY

The Boussinesq equation is a challenging problem both analytically and numerically. Owing to the complex dynamic development of small scales and the rapid loss of solution regularity, the Boussinesq equation pushes any numerical strategy to the limit. With uniform meshes, the amount of computational time is too large to enable us to obtain useful numerical approximations. Therefore, developing effective and robust moving mesh methods for these problems becomes necessary. In this work, we develop an efficient moving mesh algorithm for solving the two-dimensional Boussinesq equation. Our moving mesh algorithm is an extension of Tang and Tang (*SIAM J. Numer. Anal.* 2003; **41**:487–515) for hyperbolic conservation laws and Zhang and Tang (*Commun. Pure Appl. Anal.* 2002; **1**:57–73) for convection-dominated equations. Several numerical fluxes (*Riemann Solvers and Numerical Methods for Fluid Dynamics*: *A Practical Introduction* (2nd edn). Springer: Berlin, 1999; *WASCOM 99": 10th Conference on Waves and Stability in Continuous Media*, Porto Ercole, Italy, 1999; 257–264; *High-order Methods for Computational Physics*. Springer: Berlin, 1999; 439–582; *J. Sci. Comput.* 1990; **5**:127–149; *SIAM J. Numer. Anal.* 2003; **41**:487–515; *Commun. Pure Appl. Anal.* 2002; **1**:57–73) are also discussed. Numerical results demonstrate the advantage of our moving mesh method in resolving the small structures. Copyright © 2009 John Wiley & Sons, Ltd.

Received 15 July 2008; Revised 18 December 2008; Accepted 7 January 2009

KEY WORDS: Boussinesq equation; moving mesh method; WENO scheme; finite volume method; PDE evolution; mesh-redistribution

1. INTRODUCTION

A common feature of mathematical models [1–6] in physics, astrophysics, mechanics, as well as in most engineering disciplines, is the appearance of systems of partial differential equations (PDEs). The computational difficulties of the problem appear when there exist very rapid variations in the solution. The required resolution may be obtained by either increasing the number of grid points or using adaptive mesh methods. This paper aims at using adaptive mesh methods for solving PDEs.

*Correspondence to: Lee Wan-Lung, Singapore-MIT Alliance, 4 Engineering Drive 3, National University of Singapore, Singapore 117576, Singapore.
†E-mail: smalwl@nus.edu.sg

If the solutions of the PDEs are smooth, we can solve the PDEs by using the uniform mesh. However, for some problems there exist very large variations in the solution, e.g. in fluid dynamics, combustion and heat transfer. The resolution of these problems requires a large number of grid points over a small portion of the physical domain to resolve the large solution variations. The use of a large number of uniform grids becomes computationally wasteful, since most of the grid points are not needed. The choice of a non-uniform mesh cannot only retain the accuracy but also improve the efficiency of an existing method by concentrating the mesh points only in the regions of large variations in the solution. Adaptive mesh methods are also used for problems whose solutions are singular.

The adaptive mesh methods and techniques for their applications have been developed for more than 20 years. One of the reasons is that the adaptive mesh methods can increase the accuracy of the numerical approximations and decrease the computational cost.

In general there are three types of grid adaptation.

- The first one is the h-method, which generates the meshes by element size refinement, see [7]. This method adds extra points to an existing mesh where they are necessary and removes them when they are no longer needed for improving local grid resolution. In general, we add many points in the region of large variation and remove some points in the smooth area.
- The second technique, p-method, enhances the order of the polynomial approximation inside some elements, see [8]. This method employs higher-order numerical schemes to improve the local accuracy. The combination of the h-method and p-method is called hp-method, see [8–10].
- The third approach, r-method, see [11–13], also called moving mesh method, maintains the existing number of grid points globally. The grid points are moved from the smooth areas to the non-smooth regions based on the same grid-redistribution principles. The goal is to concentrate the grid points in the region of large variation.

In this paper, we shall present the r-method, known as the moving mesh method for solving Boussinesq equation. The present moving mesh algorithm is an extension of Tang and Tang [12] for solving hyperbolic conservation laws and Zhang and Tang [13] for solving convection-dominated equations. Another main motivation of this work is to study the simulation results of the moving mesh method based on different types of numerical fluxes, with the objective of obtaining better solution by choosing suitable numerical fluxes. In Sections 2–4, we review and describe the strategies of our moving mesh methods. The different types of numerical fluxes under consideration are presented in Section 3. The numerical results are shown in Section 5. Some concluding remarks are made in Section 6.

2. MOVING MESH METHOD

First, let Ω_{p} be the physical domain and Ω_{c} be the computational domain. A moving mesh may be generated through a bijective map from the computational domain to the physical domain. One useful mapping uses the so-called equidistribution principle, first introduced by de Boor [14]. It involves selecting mesh points such that the measure of the solution error is equalized over each subinterval.

One important concept of the moving mesh methods is the monitor function. The basic idea of using the monitor function is to require that the density of mesh is proportional to the value of monitor function, see [12, 13].

2.1. One-dimensional case

In 1D, let x and ξ denote the physical and computational coordinates, respectively, which are (without loss of generality) assumed to be in $[a,b]$ and [0,1], respectively. A one-to-one coordinate transformation between these domains is denoted by

$$x = x(\xi), \quad \xi \in [0,1], \quad x(0) = a, \quad x(1) = b$$

We assume that moving mesh $\{x_j\}$ and uniform mesh $\{\xi_j\}$ are given by

$$a = x_0 < x_1 < \cdots < x_N = b, \quad \xi_j = \frac{j}{N}, \quad 0 \leqslant j \leqslant N$$

We describe the mesh redistribution at each time step. The mesh generation equation is based on the de Boor's *equidistribution principle* [14] and written as follows:

$$(wx_\xi)_\xi = 0 \tag{1}$$

where the function w is called monitor function which in general depends on the underlying solution and is an indicator of the degree of singularity. The idea of equidistribution principle is to choose a mesh such that a geometric measure of a function is distributed equally between adjacent nodes. As the product of wx_ξ is a constant, when value of w increases, the value of x_ξ decreases; therefore, resulting in mesh clustering. That is, the value of monitor function is proportional to the density of mesh.

2.2. Two-dimensional case

In 2D, let (x, y) and (ξ, η) denote the physical and computational coordinates, respectively, which are (without loss of generality) assumed to be in $[a,b]\times[c,d]$ and $[0,1]\times[0,1]$, respectively. A one-to-one coordinate transformation between these domains is denoted by

$$(x, y) = (x(\xi,\eta), y(\xi,\eta)), \quad (\xi,\eta) \in [0,1]\times[0,1]$$

$$x(0,\eta) = a, \quad x(1,\eta) = b, \quad y(\xi,0) = c, \quad y(\xi,1) = d$$

We assume that the moving mesh $\{(x_{j,k}, y_{j,k})\}$ is given by

$$a = x_{0,k} < x_{1,k} < \cdots < x_{N,k} = b, \quad k = 0,1,\ldots,N$$

$$c = y_{j,0} < y_{j,1} < \cdots < y_{j,N} = d, \quad j = 0,1,\ldots,N$$

and uniform mesh $\{(\xi_{j,k}, \eta_{j,k})\}$ is given by

$$(\xi_{j,k}, \eta_{j,k}) = \left(\frac{j}{N}, \frac{k}{N}\right), \quad j,k = 0,1,\ldots,N$$

One of the most important issues is how to choose satisfactory mesh map (x, y). In the two-dimensional case, the mesh map can be written as the minimizer of a functional of the following form:

$$E(\xi,\eta)=\frac{1}{2}\int_{\Omega_p}(\nabla\xi^T G_1^{-1}\nabla\xi+\nabla\eta^T G_2^{-1}\nabla\eta)\,dx\,dy \tag{2}$$

where G_1 and G_2 are given symmetric positive definite matrices called monitor functions.

The moving mesh is determined by the Euler–Lagrange equations associated with minimizing the functional $E(\xi,\eta)$ in (2):

$$\nabla\cdot(G_1^{-1}\nabla\xi)=0,\quad \nabla\cdot(G_2^{-1}\nabla\eta)=0 \tag{3}$$

One of the simplest choices of the monitor function is $G_1=G_2=wI$, where I is the identity matrix and $w>0$ is a scalar weight function, for example, $w=\sqrt{1+u_x^2+u_y^2}$. In this case, from (3), we obtain Winslow's variable diffusion method [15]:

$$\nabla\cdot\left(\frac{1}{w}\nabla\xi\right)=0,\quad \nabla\cdot\left(\frac{1}{w}\nabla\eta\right)=0 \tag{4}$$

Via using the above equations, a map between the physical domain Ω_p and computational domain Ω_c can be computed. Typically, the map transforms a uniform mesh in the computational domain to cluster grid points in the regions of the physical domain where the solution has the large gradients.

In the two-dimensional case, Ceniceros and Hou's mesh generator is defined as in [16]:

$$(wx_\xi)_\xi+(wx_\eta)_\eta=0,\quad (wy_\xi)_\xi+(wy_\eta)_\eta=0 \tag{5}$$

which will be frequently used in our work.

2.3. *Monitor function*

In this section, we will discuss some properties of the monitor function. Monitor function plays a key role in the adaptive mesh methods. A suitable choice of the monitor function generates an adaptive mesh with desirable grid qualities.

Monitor functions are problem dependent, which depend not only on mesh but also on the underlying solution and its derivatives. Using the equidistributing principle, the monitor function can be arc-length or curvature dependent. The conventional form is $w=\sqrt{1+\alpha|u|^2+\beta|\nabla u|^2}$, where α and β are some non-negative constants. For example, if we take $\alpha=\beta=0$, then a uniform grid is produced. If we take $\alpha=0$ and $\beta=1$, the effect of the gradient is well re-presented. For example, we can take $w=\sqrt{1+\alpha|u|^2}$ with $\alpha>0$ for blow-up problem, and take $w=\sqrt{1+\beta|\nabla u|^2}$ with $\beta>0$ for Burger's equation. The scaling constants α and β in the monitor function play a very important role in mesh adaption. Too small α and β have little effect for mesh adaptivity, while too large α and β cause too much mesh deformation. The optimal choice of the scaling constant requires further study.

3. PDE EVOLUTION

3.1. One-dimensional case

Although the main objective of this work is to provide an effective moving mesh algorithm for two-dimensional singular problems, it is necessary to illustrate the basic ideas by starting with one-dimensional problems. The main purpose of this section is to discuss some basic techniques for solving PDEs

$$u_t = f(u)_x \tag{6}$$

In this equation, u is a function of x and t. The notation u_t denotes $\partial u/\partial t$. Equation (6) governs the solution u at time t and position x in a one-dimensional body. We write u at point x. Thus, u is a real-valued function of two real variables.

Our moving mesh method is formed by two independent parts: PDE evolution and mesh redistribution. The most important problem is how to compute approximate values of $f(u)_x$ on a moving mesh. Of course, $f(u)_x = f(u)_\xi \cdot \xi_x$. In the general case, the expression of $f(u_{j+1/2})_\xi$ is given by $(f_{j+1} - f_j)/(\xi_{j+1} - \xi_j)$, this is a second-order approximation. Moreover, some oscillations appear in calculation. We replace f_j by $\bar{f}_j$ which is not equal to f_j but is an approximation to this value. Let us consider the expression:

$$f(u_{j+1/2})_\xi = \frac{\bar{f}_{j+1} - \bar{f}_j}{\xi_{j+1} - \xi_j} \quad \text{where } \bar{f}_j = h(u_j^-, u_j^+) \tag{7}$$

with the values u_j^- and u_j^+ obtained by high-order polynomial interpolation approximation to the solution u_j at all interfaces, see [12, 13, 17–20]. Several choices of (u_j^-, u_j^+) are discussed as follows. The function h is a monotone flux, which satisfies:

- $h(a,b)$ is a Lipschitz continuous function in both arguments;
- $h(a,b)$ is a non-decreasing function in a and a non-increasing function in b, symbolically denoting $h(\uparrow, \downarrow)$;
- $h(a,b)$ is consistent with the physical flux f, that is, $h(a,a) = f(a)$.

Examples of monotone fluxes include:

1. *Godunov flux*:

$$h(a,b) = \begin{cases} \min\limits_{a \leqslant u \leqslant b} f(u) & \text{if } a \leqslant b \\ \max\limits_{b \leqslant u \leqslant a} f(u) & \text{if } a > b \end{cases}$$

2. *Engquist-Osher flux*:

$$h(a,b) = \int_0^a \max(f'(u), 0)\,\mathrm{d}u + \int_0^b \min(f'(u), 0)\,\mathrm{d}u + f(0)$$

3. *Lax-Friedrichs flux*:

$$h(a,b) = \tfrac{1}{2}[f(a) + f(b) - \alpha \cdot (b-a)]$$

where $\alpha = \max_u |f'(u)|$. The maximum is taken over the relevant range of u. However, for simplicity, we set $\alpha = \frac{1}{2}(f'(a)+f'(b))$.

In (7), there are several choices of u_j^- and u_j^+. Below we list four possible types for u_j^- and u_j^+.

Type I (*see* [12, 13], *in case of uniform gird*):

$$u_j^- = u_{j-1/2} + \tfrac{1}{2}s_{j-1/2}, \quad u_j^+ = u_{j-1/2} - \tfrac{1}{2}s_{j+1/2} \tag{8}$$

where $s_{j+1/2}$ is an approximation of the slope u_x at $x_{j+1/2}$, defined by

$$s_{j+1/2} = (\mathrm{sign}(s_{j+1/2}^+) + \mathrm{sign}(s_{j+1/2}^-)) \frac{|s_{j+1/2}^+ \cdot s_{j+1/2}^-|}{|s_{j+1/2}^+| + |s_{j+1/2}^-| + \varepsilon}$$

Here $\varepsilon > 0$ is introduced to avoid the denominator to become zero. We use $\varepsilon = 10^{-20}$, and take

$$s_{j+1/2}^+ = u_{j+3/2} - u_{j+1/2}, \quad s_{j+1/2}^- = u_{j+1/2} - u_{j-1/2}$$

Type II (*see* [19–21]):

$$\text{WENO type for}(u_j^-, u_j^+) \tag{9}$$

Type III (*see* [18]):

$$\begin{aligned} u_j^- &= u_{j-1/2} + \frac{1}{2}\left(\frac{\mathrm{sign}(u_{j+1/2}-u_{j-1/2}) + \mathrm{sign}(u_{j-1/2}-u_{j-3/2})}{2}\right) \\ &\quad \times \min\{|u_{j+1/2}-u_{j-1/2}|, |u_{j-1/2}-u_{j-3/2}|\} \\ u_j^+ &= u_{j+1/2} - \frac{1}{2}\left(\frac{\mathrm{sign}(u_{j+3/2}-u_{j+1/2}) + \mathrm{sign}(u_{j+1/2}-u_{j-1/2})}{2}\right) \\ &\quad \times \min\{|u_{j+3/2}-u_{j+1/2}|, |u_{j+1/2}-u_{j-1/2}|\} \end{aligned} \tag{10}$$

Type IV (*see* [17]):

$$\begin{aligned} u_j^- &= u_{j-1/2} + \tfrac{1}{2}[\mathrm{sign}(u_{j+1/2}-u_{j-1/2}) \\ &\quad \times \max\{0, \min\{|u_{j+1/2}-u_{j-1/2}|, (u_{j-1/2}-u_{j-3/2})\cdot \mathrm{sign}(u_{j+1/2}-u_{j-1/2})\}\}] \\ u_j^+ &= u_{j+1/2} - \tfrac{1}{2}[\mathrm{sign}(u_{j+3/2}-u_{j+1/2}) \\ &\quad \times \max\{0, \min\{|u_{j+3/2}-u_{j+1/2}|, (u_{j+1/2}-u_{j-1/2})\cdot \mathrm{sign}(u_{j+3/2}-u_{j+1/2})\}\}] \end{aligned} \tag{11}$$

Various types of u_j^- and u_j^+ will be tested to improve the accuracy of the conservative interpolation.

3.2. Two-dimensional case

Assume that the underlying 2D PDE is of form:

$$u_t + f(x, y, t, u, u_x, u_y) = 0 \tag{12}$$

with u satisfying $u(x, y, 0) = u_0(x, y)$ and appropriate boundary conditions, $(x, y) \in \Omega_\mathrm{p}$, $(\xi, \eta) \in \Omega_\mathrm{c}$. In some computations, it is necessary to map the function $u(x, y)$ in the physical domain onto function $u(\xi,\eta)$ in the computational domain. We shall use the following transformation formulas, see [12, 13],

$$u_x = \frac{u_\xi y_\eta - u_\eta y_\xi}{J} = \frac{1}{J}[(y_\eta u)_\xi - (y_\xi u)_\eta], \quad u_y = \frac{x_\xi u_\eta - x_\eta u_\xi}{J} = \frac{1}{J}[(x_\xi u)_\eta - (y_\eta u)_\xi] \tag{13}$$

where $J = x_\xi y_\eta - x_\eta y_\xi$ is the Jacobian of the coordinate transformation. Note that the value of J is always non-zero as proved by Clement *et al.* [22].

For solving problems with large gradients, the main difficulty is how to calculate the first derivatives u_x and u_y. The use of central differences may result in some oscillations. To reduce the oscillations, following [12], we use the following approximations for u_x and u_y to approximate the formulas in (13). Below we extend the formulas (8)–(11) into 2D. By (13), we have

$$(u_x)_{j+1/2,k+1/2} = \frac{((y_\eta u)_\xi)_{j+1/2,k+1/2} - ((y_\xi u)_\eta)_{j+1/2,k+1/2}}{((y_\eta x)_\xi)_{j+1/2,k+1/2} - ((y_\xi x)_\eta)_{j+1/2,k+1/2}} \tag{14}$$

The term $((y_\eta u)_\xi)_{j+1/2,k+1/2}$ is discretized by

$$\begin{aligned}((y_\eta u)_\xi)_{j+1/2,k+1/2} &= \frac{(y_\eta \cdot \bar{u}^\xi)_{j+1,k+1/2} - (y_\eta \cdot \bar{u}^\xi)_{j,k+1/2}}{\Delta\xi} \\ &= \frac{(y_{j+1,k+1} - y_{j+1,k}) \cdot \bar{u}^\xi_{j+1,k+1/2} - (y_{j,k+1} - y_{j,k}) \cdot \bar{u}^\xi_{j,k+1/2}}{\Delta\xi \cdot \Delta\eta}\end{aligned} \tag{15}$$

where $\bar{u}^\xi_{j,k+1/2}$ is an approximation to $u_{j,k+1/2}$, given by

$$\bar{u}^\xi_{j,k+1/2} = \alpha \cdot \bar{u}^{\xi-}_{j,k+1/2} + (1-\alpha) \cdot \bar{u}^{\xi+}_{j,k+1/2}, \quad \alpha \in [0,1] \tag{16}$$

Here, $\bar{u}^{\xi-}_{j,k+1/2}$ and $\bar{u}^{\xi+}_{j,k+1/2}$ are the approximations to $u_{j,k+1/2}$, obtained by the formulas (8)–(11) in the ξ-axis. In our computations, we always set $\alpha = 0.5$. Similarly, the term $((y_\xi u)_\eta)_{j+1/2,k+1/2}$ is approximated by

$$((y_\xi u)_\eta)_{j+1/2,k+1/2} = \frac{(y_{j+1,k+1} - y_{j,k+1}) \cdot \bar{u}^\eta_{j+1/2,k+1} - (y_{j+1,k} - y_{j,k}) \cdot \bar{u}^\eta_{j+1/2,k}}{\Delta\eta \cdot \Delta\xi} \tag{17}$$

where $\bar{u}^\eta_{j+1/2,k}$ is an approximation to $u_{j+1/2,k}$, given by

$$\bar{u}^\eta_{j+1/2,k} = \alpha \cdot \bar{u}^{\eta-}_{j+1/2,k} + (1-\alpha) \cdot \bar{u}^{\eta+}_{j+1/2,k}, \quad \alpha \in [0,1] \tag{18}$$

Here, $\alpha = 0.5$. $\bar{u}^{\eta-}_{j+1/2,k}$ and $\bar{u}^{\eta+}_{j+1/2,k}$ are also the approximations to $u_{j+1/2,k}$, obtained by the formulas (8)–(11) in the η-axis. For calculating $(u_y)_{j+1/2,k+1/2}$, we use

$$(u_y)_{j+1/2,k+1/2} = \frac{((x_\xi u)_\eta)_{j+1/2,k+1/2} - ((x_\eta u)_\xi)_{j+1/2,k+1/2}}{((x_\xi y)_\eta)_{j+1/2,k+1/2} - ((x_\eta y)_\xi)_{j+1/2,k+1/2}} \tag{19}$$

The terms $((x_\xi u)_\eta)_{j+1/2,k+1/2}$ and $((x_\eta u)_\xi)_{j+1/2,k+1/2}$ are approximated by

$$((x_\xi u)_\eta)_{j+1/2,k+1/2} = \frac{(x_{j+1,k+1} - x_{j,k+1}) \cdot \bar{u}^\eta_{j+1/2,k+1} - (x_{j+1,k} - x_{j,k}) \cdot \bar{u}^\eta_{j+1/2,k}}{\Delta\eta \cdot \Delta\xi}$$

$$((x_\eta u)_\xi)_{j+1/2,k+1/2} = \frac{(x_{j+1,k+1} - x_{j+1,k}) \cdot \bar{u}^\xi_{j+1,k+1/2} - (x_{j,k+1} - x_{j,k}) \cdot \bar{u}^\xi_{j,k+1/2}}{\Delta\xi \cdot \Delta\eta}$$

where $\bar{u}^\eta_{j+1/2,k}$ and $\bar{u}^\xi_{j,k+1/2}$ are defined by (18) and (16), respectively.

4. MESH REDISTRIBUTION

Recall that our moving mesh method is formed by two independent parts: PDE evolution and mesh redistribution. We illustrated PDE evolution part in the previous section. We consider the mesh redistribution part in this section.

In the following, we shall explore some existing ideas for mesh redistribution in 1D, in order to obtain a good understanding of mesh-renewing and solution updating. We then extend the 1D mesh redistribution to 2D.

4.1. Moving mesh in 1D

In [12], a second-order conservative interpolation formula is introduced. This interpolation formula does not increase the total variation, and as a result the resulting adaptive mesh solutions satisfy several fundamental properties of the hyperbolic conservative laws.

Let $\{x_j\}$ be old mesh, $\{u_{j+1/2}\}$ be old solution, $\{\tilde{x}_j\}$ be new mesh and $\{\tilde{u}_{j+1/2}\}$ be new solution. We assume that x_j and $u_{j+1/2}$ are available for $j<0$ and $j>N$, if needed.

For ease of discussion, we set monitor function as

$$w_{j+1/2} = \sqrt{1 + \alpha \cdot (u_\xi^2)_{j+1/2}}$$

where α is a positive constant. In the equidistribution principle (1), the mesh equation can be approximated by

$$\begin{gathered} \frac{(wx_\xi)_{j+1/2} - (wx_\xi)_{j-1/2}}{\Delta\xi} = 0 \\ \Rightarrow w_{j+1/2} \cdot x_{j+1} + w_{j-1/2} \cdot x_{j-1} = (w_{j+1/2} + w_{j-1/2}) \cdot x_j \end{gathered} \tag{20}$$

The above approximation is of second-order accuracy. Using (20), a new mesh $\tilde{x}_j$ can be obtained by iterative method, such as Jacobi, Gauss–Seidel and Successive Over Relaxation. In this paper, we will use the Jacobi iteration since Jacobi iteration produces symmetric mesh even the monitor function variation is large. In our computational experience, Jacobi iteration is better than other iterations for solving blow-up problems or mesh-symmetric problems. In practice, only a few iterations (say 3–5) are required at each time level; hence, the cost for generating new mesh is not too expensive.

In the following we shall discuss how to update the solution u. For a given time, a mesh x is updated to a new mesh $\tilde{x}$. After each update of mesh, we need to pass the solution information from the old mesh to the newly obtained mesh. Hence, it is necessary to design an algorithm for updating new solution $\tilde{u}$ from x, u and $\tilde{x}$. To begin with, we assume that the difference between $\tilde{x}_{j+1/2}$ and $x_{j+1/2}$ is small. Let $\tilde{u}_{j+1/2}$ and $u_{j+1/2}$ be cell averages of solution $u(x)$ over interval $[\tilde{x}_j, \tilde{x}_{j+1}]$

and $[x_j, x_{j+1}]$, respectively, and let $c_j = x_j - \tilde{x}_j$, $\Delta x_{j+1/2} = x_{j+1} - x_j$ and $\Delta \tilde{x}_{j+1/2} = \tilde{x}_{j+1} - \tilde{x}_j$, we have as in [12]

$$\int_{\tilde{x}_j}^{\tilde{x}_{j+1}} u(\tilde{x})\,\mathrm{d}\tilde{x} = \int_{x_j}^{x_{j+1}} u(x)\,\mathrm{d}x - [(cu)_{j+1} - (cu)_j] \tag{21}$$

By (21), we obtain

$$\tilde{u}_{j+1/2} = \frac{1}{\Delta \tilde{x}_{j+1/2}} [\Delta x_{j+1/2} \cdot u_{j+1/2} - ((cu)_{j+1} - (cu)_j)] \tag{22}$$

where $(cu)_j$ is equal to $c_j \cdot u(x_j)$. Note that the above solution-updating method (22) satisfies the following mass-conservation:

$$\sum_j \Delta \tilde{x}_{j+1/2} \tilde{u}_{j+1/2} = \sum_j \Delta x_{j+1/2} u_{j+1/2} \tag{23}$$

However, if we use the linear approximation for $(cu)_j$ in (22), the formula becomes first-order accuracy. It is necessary to use a second-order accurate numerical flux $(\widehat{cu})_j$ to replace $(cu)_j$ in (22). We approximate the flux $(\widehat{cu})_j$ by an up-winding scheme:

$$\widehat{cu}_j = \begin{cases} c_j \cdot u_j^- & \text{if } c_j \geqslant 0 \\ c_j \cdot u_j^+ & \text{if } c_j < 0 \end{cases} \tag{24}$$

where (u_j^+, u_j^-) is given by (8)–(11). In this case, we use (8).

4.2. Moving mesh in 2D

In this section, we focus on the 2D moving mesh generation. It should be pointed out that the extension of the 1D equidistributing principle to 2D is not straightforward. In addition to adaptation and smoothness, other mesh properties such as skewness and orthorgonality have to be incorporated in the 2D formulation, see [12].

Let $\{x_{j,k}\}$ be the old mesh and $\{\tilde{x}_{j,k}\}$ be the new mesh in the x-coordinate, $\{y_{j,k}\}$ be the old mesh and $\{\tilde{y}_{j,k}\}$ be the new mesh in the y-coordinate, and $\{u_{j+1/2,k+1/2}\}$ be the old solution and $\{\tilde{u}_{j+1/2,k+1/2}\}$ be the new solution, respectively. Recall the equidistribution principle (5):

$$(wx_\xi)_\xi + (wx_\eta)_\eta = 0, \quad (wy_\xi)_\xi + (wy_\eta)_\eta = 0 \tag{25}$$

In our case, $\Delta\xi = \Delta\eta$, the first term in (25) is then approximated by

$$\frac{(wx_\xi)_{j+1/2,k} - (wx_\xi)_{j-1/2,k}}{\Delta\xi} + \frac{(wx_\eta)_{j,k+1/2} - (wx_\eta)_{j,k-1/2}}{\Delta\eta} = 0$$

which gives

$$w_{j+1/2,k}\cdot\left(\frac{x_{j+1,k}-x_{j,k}}{\Delta\xi}\right)-w_{j-1/2,k}\cdot\left(\frac{x_{j,k}-x_{j-1,k}}{\Delta\xi}\right)$$
$$+w_{j,k+1/2}\cdot\left(\frac{x_{j,k+1}-x_{j,k}}{\Delta\eta}\right)-w_{j,k-1/2}\cdot\left(\frac{x_{j,k}-x_{j,k-1}}{\Delta\eta}\right)=0$$

It follows from the above equation that

$$x_{j,k}=\frac{w_{j+1/2,k}x_{j+1,k}+w_{j-1/2,k}x_{j-1,k}+w_{j,k+1/2}x_{j,k+1}+w_{j,k-1/2}x_{j,k-1}}{w_{j+1/2,k}+w_{j-1/2,k}+w_{j,k+1/2}+w_{j,k-1/2}}$$

Similar approximation is used for the second equation in (25) to obtain $y_{j,k}$. New mesh is obtained by Jacobi formulas, and Jacobi iteration produces symmetric mesh.

Let $A_{j+1/2,k+1/2}$ denote area of the quadrangle of the control volume with four vertices $(x_{j+1,k+1}, y_{j+1,k+1})$, $(x_{j,k+1}, y_{j,k+1})$, $(x_{j,k}, y_{j,k})$ and $(x_{j,k+1}, y_{j,k+1})$. Denoting $c^x=x-\tilde{x}$, $c^y=y-\tilde{y}$, then we have (see [12])

$$\int_{\tilde{A}_{j+1/2,k+1/2}} u(\tilde{x},\tilde{y})\,\mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\approx\int_{A_{j+1/2,k+1/2}} u(x,y)\,\mathrm{d}x\,\mathrm{d}y-[(c_n u)_{j+1,k+1/2}-(c_n u)_{j,k+1/2}]$$
$$-[(c_n u)_{j+1/2,k+1}-(c_n u)_{j+1/2,k}] \tag{26}$$

where $c_n=c^x n_x+c^y n_y$ with the normal (n_x,n_y). Here $(c_n u)_{j+1,k+1/2}$ and $(c_n u)_{j+1/2,k+1}$ denote the values of $c_n u$ at the corresponding surface of the control volume $A_{j+1/2,k+1/2}$. We discretize the normal (n_x,n_y) as follows:

$$\begin{pmatrix} n_x \\ n_y \end{pmatrix}_{j+1,k+1/2}=\begin{pmatrix} y_{j+1,k+1}-y_{j+1,k} \\ -(x_{j+1,k+1}-x_{j+1,k}) \end{pmatrix}$$

$$\begin{pmatrix} n_x \\ n_y \end{pmatrix}_{j+1/2,k+1}=\begin{pmatrix} -(y_{j+1,k+1}-y_{j,k+1}) \\ x_{j+1,k+1}-x_{j,k+1} \end{pmatrix}$$

From (26), we obtain a conservative-interpolation, see [12], written as

$$|\tilde{A}_{j+1/2,k+1/2}|\cdot\tilde{u}_{j+1/2,k+1/2}=|A_{j+1/2,k+1/2}|\cdot u_{j+1/2,k+1/2}-[(c_n u)_{j+1,k+1/2}-(c_n u)_{j,k+1/2}]$$
$$-[(c_n u)_{j+1/2,k+1}-(c_n u)_{j+1/2,k}] \tag{27}$$

where $|\tilde{A}|$ and $|A|$ denote the areas of the control volumes $\tilde{A}$ and A, respectively. It can be verified that the above solution-updating scheme (27) satisfies the following mass-conservation:

$$\sum_{j,k}|\tilde{A}_{j+1/2,k+1/2}|\cdot\tilde{u}_{j+1/2,k+1/2}=\sum_{j,k}|A_{j+1/2,k+1/2}|\cdot u_{j+1/2,k+1/2} \tag{28}$$

However, if we use the general terms $(c_n u)_{j,k+1/2}$ and $(c_n u)_{j+1/2,k}$ in (27), the formula becomes first-order accurate. It is necessary to use second-order accurate numerical fluxes $\mathsf{Flux}_{j,k+1/2}$

and $\mathsf{Fluy}_{j+1/2,k}$ to replace $(c_n u)_{j,k+1/2}$ and $(c_n u)_{j+1/2,k}$ in (27), respectively. The formula (27) becomes

$$\tilde{u}_{j+1/2,k+1/2} = \frac{1}{\tilde{A}}[u_{j+1/2,k+1/2}\cdot A - (\mathsf{Flux}_{j+1,k+1/2} - \mathsf{Flux}_{j,k+1/2}) - (\mathsf{Fluy}_{j+1/2,k+1} - \mathsf{Fluy}_{j+1/2,k})] \tag{29}$$

where

$$A = \tfrac{1}{2}[(x_{j,k} - x_{j+1,k+1})(y_{j+1,k} - y_{j,k+1}) - (y_{j,k} - y_{j+1,k+1})(x_{j+1,k} - x_{j,k+1})]$$

$$\tilde{A} = \tfrac{1}{2}[(\tilde{x}_{j,k} - \tilde{x}_{j+1,k+1})(\tilde{y}_{j+1,k} - \tilde{y}_{j,k+1}) - (\tilde{y}_{j,k} - \tilde{y}_{j+1,k+1})(\tilde{x}_{j+1,k} - \tilde{x}_{j,k+1})]$$

The numerical flux ($\mathsf{Flux}_{j,k+1/2}$) is defined by

$$\mathsf{Flux}_{j,k+1/2} = \begin{cases} (c_n)_{j,k+1/2}\cdot \bar{u}^{\xi-}_{j,k+1/2} & \text{if } (c_n)_{j,k+1/2} \geqslant 0 \\ (c_n)_{j,k+1/2}\cdot \bar{u}^{\xi+}_{j,k+1/2} & \text{if } (c_n)_{j,k+1/2} < 0 \end{cases} \tag{30}$$

where $\bar{u}^{\xi-}_{j,k+1/2}$ and $\bar{u}^{\xi+}_{j,k+1/2}$ are defined by (8)–(11) in the ξ-axis. In this case, we use (8).

The numerical flux ($\mathsf{Fluy}_{j+1/2,k}$) is defined by

$$\mathsf{Fluy}_{j+1/2,k} = \begin{cases} (c_n)_{j+1/2,k}\cdot \bar{u}^{\eta-}_{j+1/2,k} & \text{if } (c_n)_{j+1/2,k} \geqslant 0 \\ (c_n)_{j+1/2,k}\cdot \bar{u}^{\eta+}_{j+1/2,k} & \text{if } (c_n)_{j+1/2,k} < 0 \end{cases} \tag{31}$$

where $\bar{u}^{\eta-}_{j+1/2,k}$ and $\bar{u}^{\eta+}_{j+1/2,k}$ are defined by (8)–(11) in η-axis. In this case, we use (8).

The formulas (29)–(31) are used with the following formulas:

$$(c_n)_{j,k+1/2} = (c^x_{j,k+1/2}\cdot (y_{j,k+1} - y_{j,k}) - c^y_{j,k+1/2}\cdot (x_{j,k+1} - x_{j,k}))$$

$$(c_n)_{j+1/2,k} = (-c^x_{j+1/2,k}\cdot (y_{j+1,k} - y_{j,k}) + c^y_{j+1/2,k}\cdot (x_{j+1,k} - x_{j,k}))$$

5. NUMERICAL EXAMPLE: THE BOUSSINESQ EQUATION

In this section, we mainly investigate the use of the moving mesh approach to solve the Boussinesq equation. The moving mesh method can be useful in resolving extremely small structures with reasonably small number of grid points. For detailed discussions, see [13, 16].

There are several reasons for the study of the two-dimensional Boussinesq equation. It is a simple model to address the open problem about whether finite time singularity occurs for initially smooth flows in inviscid and incompressible three-dimensional Euler flows. The inviscid two-dimensional Boussinesq equation can be used as a model for the three-dimensional axis-symmetric Euler equation with swirl. The understanding of finite time singularities may be crucial to explain small-scale structures in viscous turbulent flows. The Boussinesq equation has also potential relevance to the study of atmospheric and oceanography turbulence, as well as other astrophysical situations where rotation and stratification play a dominant role.

The Boussinesq equation is:

$$\begin{aligned} W_t+\psi_y\cdot W_x-\psi_x\cdot W_y &= -\bar{g}\cdot\theta_x \\ \theta_t+\psi_y\cdot\theta_x-\psi_x\cdot\theta_y &= 0 \\ \psi_{xx}+\psi_{yy} &= -W \end{aligned} \tag{32}$$

where θ is temperature, W is vorticity and ψ is the stream function. Without loss of generality, we take the scaled gravity constant $\bar{g}$ as 10.

We set $W(x,y,0)=0$, $\psi(x,y,0)=0$ and $\theta(x,y,0)$ as a stratified fluid with three constant regions θ_1, θ_2 and $\bar{\theta}=(\theta_1+\theta_2)/2$ connected by two thin layers in the following form:

$$\theta(x,y,0)=\begin{cases}\theta_2+(\bar{\theta}-\theta_2)\cdot H_\delta(0.5+y_s(x)-y) & \text{if } y\geqslant 0.5\\ \theta_1+(\bar{\theta}-\theta_1)\cdot H_\delta(y-y_s(x)-0.5) & \text{if } y<0.5\end{cases} \tag{33}$$

where

$$y_s(x)=\delta+\varepsilon\cdot(1+\sin(2\pi(x+0.75))) \tag{34}$$

and $H_\delta(x)$ is the mollified Heaviside function:

$$H_\delta(x)=\begin{cases}0 & \text{if } x<-\delta\\ \dfrac{x+\delta}{2\delta}+\dfrac{1}{2\pi}\sin\left(\dfrac{\pi x}{\delta}\right) & \text{if } |x|\leqslant\delta\\ 1 & \text{if } x>\delta\end{cases} \tag{35}$$

Here, we take $\theta_1=-1$, $\theta_2=1$ and $\bar{\theta}=0$. By setting $\delta=0.04$ and $\varepsilon=0.025$, we obtain two thin symmetric layers separating smoothly the three constant values of θ.

For solving Poisson equation $\psi_{xx}+\psi_{yy}=-W$, we use multigrid preconditional conjugate gradient (MGCG) method [23, 24] to get the approximate solution ψ. The distinguishing feature of the multigrid (MG) method is that a number of different grids are used on the domain, ranging from coarse to fine. A numerical solution on a coarse grid can be computed quickly, but it has low accuracy. The MG method has inherent high parallelism and improves convergence of long wave length components, which is important in iterative methods. By using this method as a preconditioner in the preconditional conjugate gradient method, an efficient method with high parallelism and fast convergence is obtained. The MG method uses matrix-dependent prolongation [25]. The MG solver is given by P. M. De Zeeuw [25], called ⌜mgd9v.f⌟. For using this MG solver, it is necessary to set the number of grid points as $N=2^m+1$, where m is an integer.

A channel geometry is assumed. In this case, the flow is bounded by horizontal walls on the top and bottom of the layer. It is assumed to be periodic in the horizontal direction. As the flow is periodic in the horizontal direction, we impose $x(\xi,\eta)-\xi$ and $y(\xi,\eta)$ to be periodic in ξ. The stream function ψ is subjected to the Dirichlet boundary condition ($\psi=0$) on the top and bottom of the computational domain and the period boundary conditions in the horizontal direction. We construct an efficient solver for solving the transformed stream function ψ by MGCG method. Our stopping criterion for the MGCG method is a tolerance of 10^{-10} and for the MG method

Table I. Outline of the numerical algorithm for solving the Boussinesq equation.

```
0. Determine the initial mesh based on the initial function.
1. Determine $\Delta t$ based on CFL-type condition so that $t^{(n+1)} = t^{(n)} + \Delta t$.
2. Mesh Redistribution.
    (a) Solve the mesh-redistributing equation by one Jacobi iteration, to get $(\tilde{x}^{(n)}, \tilde{y}^{(n)})$.
    (b) Interpolate the approximate solutions $\tilde{\theta}^{(n)}$, $\tilde{W}^{(n)}$ on the new mesh $(\tilde{x}^{(n)}, \tilde{y}^{(n)})$.
    (c) Make the iteration procedure (a)-(b) on mesh-motion and solution-interpolation
        until there is no significant change in calculated new mesh from one iteration to
        the next.
3. Advance the solution one time step based on an appropriate numerical scheme.
    (a) Compute $\psi^{(n)}$ using MGCG iteration method.
    (b) Compute $\theta^{(n+1)}$ and $W^{(n+1)}$ by RK2.
4. Start new time step (go to 2 above).
```

is a tolerance of 10^{-10}. The alternating solution time-marching procedure is applied using the second-order Runge–Kutta method. In this work, we use 2 ghost points.

For this example, we take the monitor function $w = \sqrt{1+\alpha(\theta_\xi^2+\theta_\eta^2)}$ with $\alpha = 0.1$. In adaptive mesh method, Δt is not fixed in the computations, which is determined by the minimum distance in the mesh. Hence, we set $\Delta t = \lambda \cdot \min(x_{j+1,k} - x_{j,k}, y_{j,k+1} - y_{j,k})$, where λ is the CFL constant. In our computation, we take $\lambda = 0.3$.

A flow chart for our numerical algorithm is given in Table I.

We now present the numerical results for the layered Boussinesq inviscid fluid with zero initial vorticity and temperature given by (33)–(35). We take $N = 257$, that is, use a 257×257 grid in the computational domain of $[0,1] \times [0,1]$. The flow has four-fold symmetry. In the four small symmetric regions, the vorticity alternates signs in the pattern of $(++/--)$ and $(+-/-+)$. In the mesh redistribution part, we use the formula (29) with (u_j^-, u_j^+) given by (8). This is because the computational cost of WENO scheme (9) is larger than other formulas like (8), (10) and (11) but their numerical results are the same. As a result we choose (8). Moreover, in the PDE evolution part, we use formulas (14)–(19) to solve θ_x, θ_y, W_x, W_y, ψ_x and ψ_y.

In Figure 1, we show the close-up of the adaptive meshes at $T = 0.7$ and 0.8 in $[0.3, 0.5] \times [0.5, 0.7]$ and $[0.3, 0.5] \times [0.6, 0.8]$. As expected, many grid points are moved to the regions of large variations in the temperature. Figure 2 shows the mesh compression ratio (uniform grid size to smallest adaptive grid size). It is clearly seen that the ratios of maximum to minimum of the grid size in the x and y directions change differently with time. In particular at $T = 0.85$, we obtain a compression ratio close to 9, giving an effective resolution corresponding to that of an uniform mesh with 2250^2 grid points.

Figure 3 shows the maximum value of θ and the maximum value of W. It can be shown that the maximum and minimum values of θ are 1 and -1, respectively. The maximum values of θ in our computation are always smaller than 1.001 as in Figure 3 (left). This partly indicates that our numerical solution is reliable. It can also be observed from this figure that when the time is evolving close to $t = 0.7$, some oscillations appear in the temperature. However, the maximum leakage of these oscillations is small and below 9×10^{-4}.

We have employed the WENO-type numerical flux (9) to solve Boussinesq equation. Figures 4 and 5 present the contours of θ and W for $T = 0.7$ and 0.8 with this type flux, respectively. To

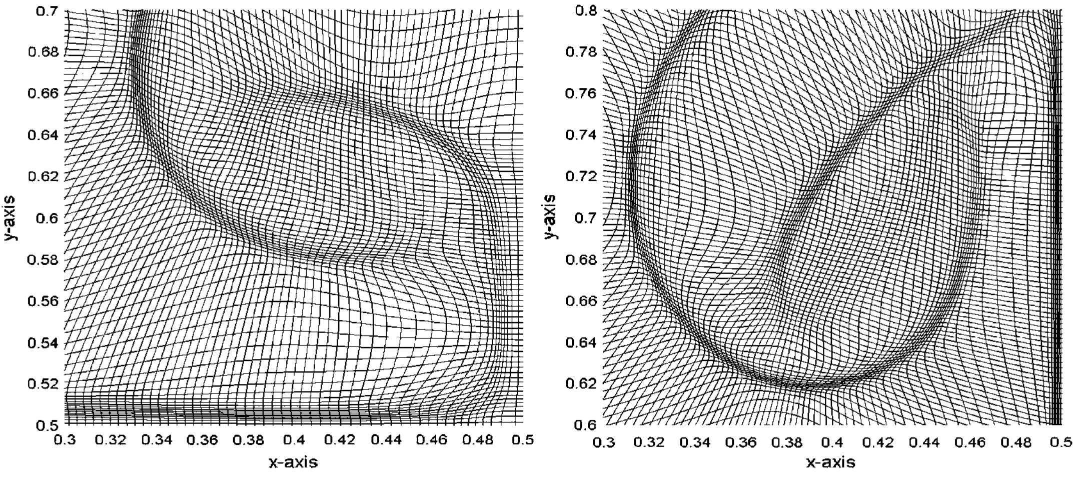

Figure 1. Close-ups of the adaptive meshes in $[0.3, 0.5]\times[0.5, 0.7]$ and $[0.3, 0.5]\times[0.6, 0.8]$ at $T=0.7$ (left) and $T=0.8$ (right), respectively.

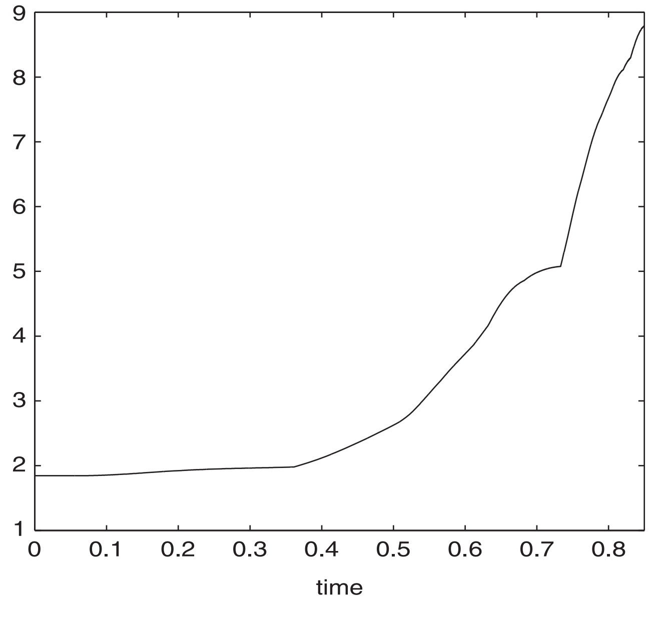

Figure 2. The mesh compression ratio (the ratio of the uniform grid size and the smallest adaptive grid size) for $0\leqslant T\leqslant 0.85$.

test for accuracy, the results of Figures 4 and 5 are compared with those fluxes of Ceniceros and Hou [16]. We can see their good agreement. We also tried other numerical fluxes, but some of them did not give satisfactory results. For example, if we use the numerical flux (8) to solve this problem, then results are incorrect. Figure 5 shows the contours of numerical results by using the

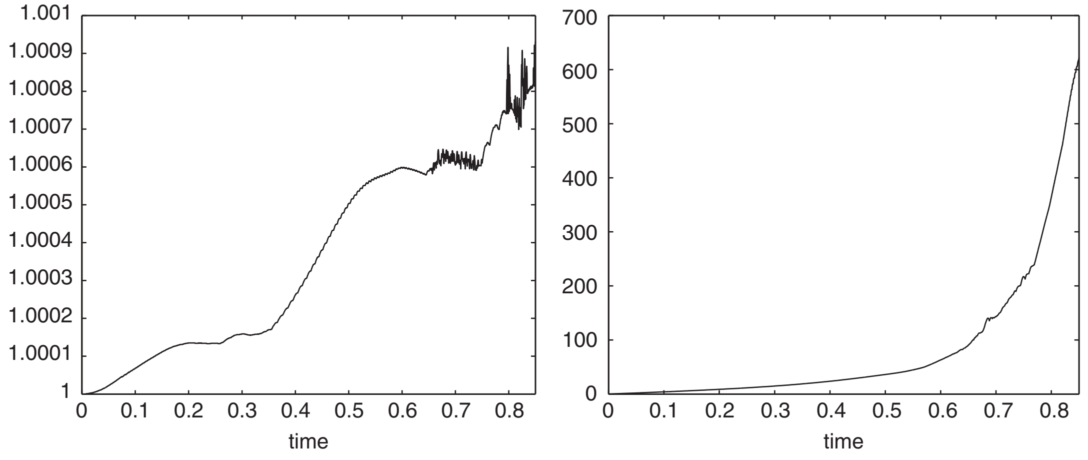

Figure 3. The maximum of temperature $\theta_{\max}$ (left) and the maximum of vorticity $W_{\max}$ (right) for $0\leqslant T\leqslant 0.85$.

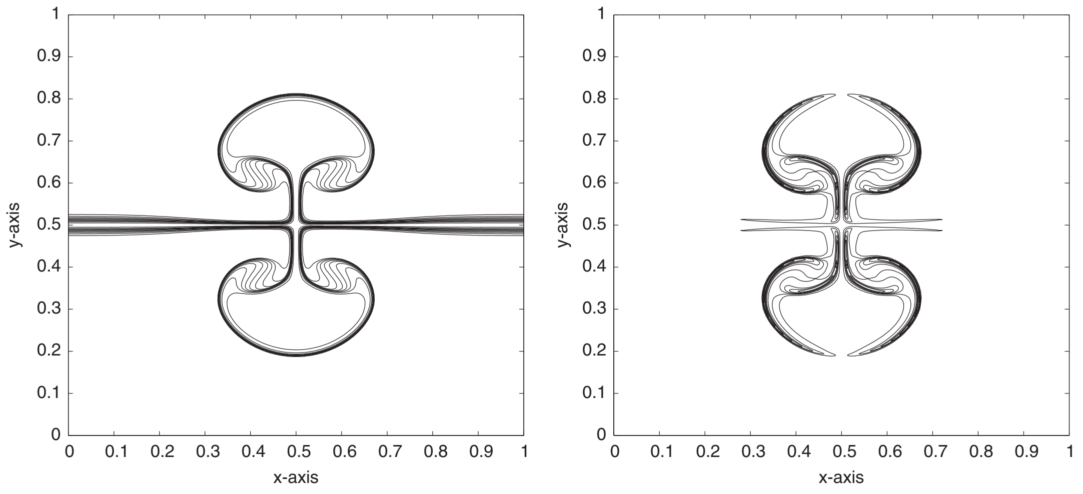

Figure 4. Temperature and vorticity contours at $T=0.7$. In PDE evolution part, (u_j^-, u_j^+) is given by (9).

WENO flux for (u_j^-, u_j^+). The plots of the contours of numerical results by using the numerical flux (u_j^-, u_j^+) in (8) are shown in Figure 6. Both figures present the contours of θ and W for $T=0.8$. In these plots, 20 equally spaced contour lines are used. The numerical results of Figure 6 are formed inaccurate. The numerical results clearly show that the WENO-[19] type finite difference is a better choice for handing this problem. The mesh resolution is insufficient after $T>0.85$. In PDE evolution part, if (u_j^-, u_j^+) is used by (10) or (11), both solutions are similar to those using (u_j^-, u_j^+) given by (8). Again the figures demonstrate that the WENO-[19] type finite difference is a better choice for handing this problem by comparison. As a result, the numerical results indicate

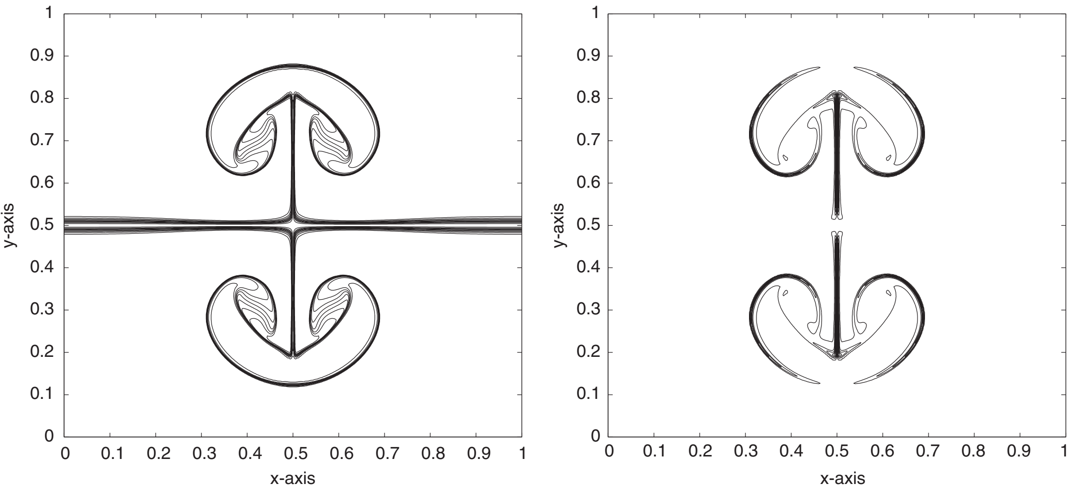

Figure 5. Temperature and vorticity contours at $T=0.8$. In PDE evolution part, (u_j^-, u_j^+) is given by (9).

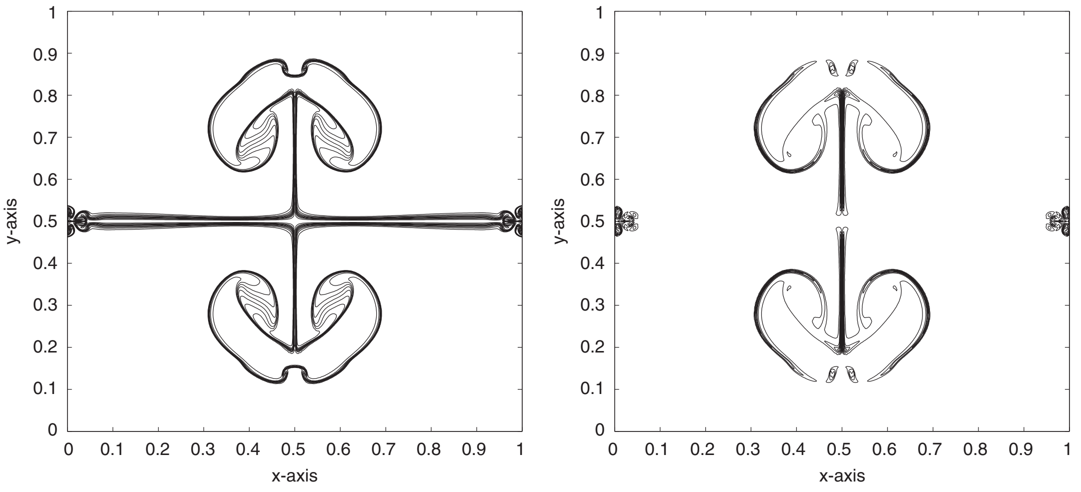

Figure 6. Temperature and vorticity contours at $T=0.8$. In PDE evolution part, (u_j^-, u_j^+) is given by (8).

that the WENO-type numerical flux (9) is better than other numerical fluxes presented in (8), (10) and (11) for solving the Boussinesq equation (32).

6. CONCLUSION

In this paper, the moving mesh methods are studied and applied to solve some two-dimensional PDEs with singularities involving solutions. We have considered some basic techniques for solving

PDEs and compared several numerical fluxes on the Boussinesq equation investigated in [16]. The underlying PDEs are mapped onto computational domain via coordinate transformation, and then the transformed PDEs are solved by the WENO scheme [19]. One of the main contributions of the current work is to use the WENO-type numerical flux (9) to replace other numerical fluxes (8), (10) and (11) for solving the Boussinesq equation. Our numerical tests indicate that the WENO-type numerical flux is better than the others. In the present method, the MGCG method is employed to solve the Poisson equation. The mesh redistribution and solution updating are also investigated. Our moving mesh algorithms are formed by two independent parts: PDE evolution and mesh-redistribution. In our numerical schemes, the PDE evolution part and the mesh-redistribution part are approximated with second-order accuracy. Computational costs of moving mesh methods can be efficiently saved with locally varying time steps [26]. It seems that the existing moving mesh methods are limited to the second-order accuracy. It will be useful to design the higher-order moving mesh schemes, which will be our future work.

REFERENCES

1. Di Y, Li R, Tang T. A general moving mesh framework in 3D and its application for simulating the mixture of multi-phase flows. *Communications in Computational Physics* 2008; **3**:582–603.
2. Qiao ZH. Numerical investigations of the dynamical behaviors and instabilities for the Gierer–Meinhardt system. *Communications in Computational Physics* 2008; **3**:406–426.
3. Tang H-Z, Tang T, Zhang PW. An adaptive mesh redistribution method for nonlinear Hamilton–Jacobi equations in two- and three-dimensions. *Journal of Computational Physics* 2003; **188**:543–572.
4. Blake KW. Moving mesh methods for nonlinear parabolic partial differential equations. *Ph.D. Thesis*, University of Reading, 2001.
5. Anderson A, Zheng X, Cristini V. Adaptive unstructured volume remeshing. I: the method. *Journal of Computational Physics* 2005; **208**(2):616–625.
6. Di Y, Li R, Tang T, Zhang PW. Moving mesh finite element methods for the incompressible Navier–Stokes equations. *SIAM Journal on Scientific Computing* 2005; **26**:1036–1056.
7. Johnson CR, MacLeod RS. Nonuniform spatial mesh adaption using a posteriori error estimates: applications to forward and inverse problems. *Applied Numerical Mathematics* 1994; **14**:311–326.
8. Babuska I, Szabo BA, Katz IN. The hp-version of the finite element method. *SIAM Journal on Numerical Analysis* 1981; **18**:515–545.
9. Cugnon F, Beckers P. Error estimation for h- and p-method. *Eighth Mechanical Engineering Chilean Congress*, Concepcion, Chile, 1998; 183–188.
10. Schwab C. *p- and hp-Finite Element Methods*, *Theory and Applications to Solid and Fluid Mechanics*. Oxford University Press: Oxford, 1998.
11. Li R. Moving mesh method and its application. *Ph.D. Thesis*, School of Mathematical Sciences, Peking University, May 2001 (in Chinese).
12. Tang H-Z, Tang T. Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws. *SIAM Journal on Numerical Analysis* 2003; **41**:487–515.
13. Zhang ZR, Tang T. An adaptive mesh redistribution algorithm for convection-dominated problems. *Communications on Pure and Applied Analysis* 2002; **1**:57–73.
14. de Boor C. *Good Approximation by Spline with Variable Knots II*. Springer Lectures Notes Series. Springer: Berlin, 1973.
15. Winslow A. Numerical solution of the quasi-linear Poisson equation in a nonuniform triangle mesh. *Journal of Computational Physics* 1967; **1**:149–172.
16. Ceniceros HD, Hou TY. An efficient dynamically adaptive mesh for potentially singular solutions. *Journal of Computational Physics* 2001; **172**:609–639.
17. Elewterio FT. *Riemann Solvers and Numerical Methods for Fluid Dynamics*: *A Practical Introduction* (2nd edn). Springer: Berlin, 1999.
18. Jannelli A. A central difference scheme for a hyperbolic model describing phytoplankton growth. *WASCOM 99": 10th Conference on Waves and Stability in Continuous Media*, Porto Ercole, Italy, 1999; 257–264.

19. Shu CW. High order ENO and WENO schemes for computational fluid dynamics. In *High-order Methods for Computational Physics*, Barth TJ, Deconinck H (eds). Lecture Notes in Computational Science and Engineering, vol. 9. Springer: Berlin, 1999; 439–582.
20. Shu C-W. Numerical experiments on accuracy of ENO and modified ENO schemes. *Journal of Scientific Computing* 1990; **5**:127–149.
21. Shu C-W, Osher S. Efficient implement of essentially non-oscillatory shock-wave schemes. *Journal of Computational Physics* 1989; **83**:32–78.
22. Clement P, Hagmeijer R, Sweers G. On the invertibility of mappings arising in 2D grid generation problems. *Numerische Mathematik* 1996; **73**:37–51.
23. Tatebe O. MGCG method: a robust and highly parallel iterative method. *Ph.D. Thesis*, University of Tokyo, March, 1997.
24. Tatebe O, Oyanagi Y. Efficient implementation of the multigrid preconditioned conjugate gradient method on distributed memory machine. *ACM/IEEE Conference on Supercomputing*. IEEE Computer Society: Washington, DC, 14–18 November 1994; 194–203.
25. De Zeeuw PM. Matrix-dependent prolongation and restrictions in a blackbox multigrid solver. *Journal of Computational and Applied Mathematics* 1990; **33**(1):1–27.
26. Tan ZJ, Zhang ZR, Tang T, Huang YQ. Moving mesh methods with locally varying time steps. *Journal of Computational Physics* 2004; **200**:347–367.