Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.2011.m9017

Vol. 4, No. 1, pp. 92-112
February 2011

Simulating Three-Dimensional Free Surface Viscoelastic Flows using Moving Finite Difference Schemes

Yubo Zhang[1] and Tao Tang[2,*]

[1] *Department of Computer Science, 2063 Kemper Hall, University of California at Davis, One Shields Avenue, Davis, CA 95616-8562, USA.*
[2] *Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong.*

Received 10 November 2009; Accepted (in revised version) 21 August 2010

Available online 29 October 2010

Abstract. An efficient finite difference framework based on moving meshes methods is developed for the three-dimensional free surface viscoelastic flows. The basic model equations are based on the incompressible Navier-Stokes equations and the Oldroyd-B constitutive model for viscoelastic flows is adopted. A logical domain semi-Lagrangian scheme is designed for moving-mesh solution interpolation and convection. Numerical results show that harmonic map based moving mesh methods can achieve better accuracy for viscoelastic flows with free boundaries while using much less memory and computational time compared to the uniform mesh simulations.

AMS subject classifications: 65M20, 65N22

Key words: Moving mesh, free surface, viscoelastic flow, solution interpolation.

1. Introduction

Modeling and simulating viscoelastic flows with free boundary have been challenging due to the fact that the constitutive equation adds more complexity to the original Navier-Stokes equation and the moving boundary in free surface flow often requires high resolution meshes to achieve good computational results.

Several computational techniques have been developed for moving interface problems, including volume-of-fluid methods [13], level set methods [20, 21, 25] and diffuse-interface methods [1]. There are also a large number of modifications and hybrid techniques proposed by different people. For example, [26] combines some of the advantages of the volume-of-fluid method with the level set method to obtain a method which is generally superior to either method alone and [10] improved the mass conservation properties

*Corresponding author. *Email addresses:* ybzhang@ucdavis.edu (Y. B. Zhang), ttang@math.hkbu.edu.hk (T. Tang)

of the level set method by using Lagrangian marker particles to rebuild the level set in regions which are under-resolved. These techniques were successfully applied to multi-phase or free surface flows (see e.g. [15, 23, 27, 33]). The main challenge is that very fine computational resolution is needed for resolving thin interfaces. Therefore it is practical to implement these methods on adaptive meshes since the problem scale grows rapidly especially in 3D cases. In past years, many adaptive mesh techniques have been proposed which can be classified as adaptive mesh refinement methods and adaptive mesh redistribution methods, see [11, 19, 30, 34]. Using adaptive mesh methods for moving interface problems is also straightforward. For example, [34] simulated two-phase viscoelastic flows using phase-field model with local refined meshes and [8] simulated incompressible two-phase flows using level set method with adaptively redistributed meshes.

In this work, we simulated free surface viscoelastic flows using a moving mesh method (i.e., adaptive mesh redistribution method in the sense of [19]). In particular, we will use the moving mesh algorithms developed in Li *et al.* [16, 17] which redistribute mesh nodes based on harmonic mappings. The moving mesh method based on harmonic mapping has been applied successfully to several complex problems including incompressible flow [6–8], reaction-diffusion systems [22], and dendritic growth [14, 31, 32]. The goal of the moving mesh method is to reduce the computational cost and to enhance the accuracy in resolving the moving interfaces. We designed a moving finite difference based framework which is much faster.

We use incompressible Navier-Stokes equations coupled with Oldroyd-B constitutive equation as the basic models. Phase-field model is used for two-phase flows and level set method is used for free surface flows. We follow Chorin's projection method [4] to keep the velocity field divergence free. For free surface viscoelastic flows, we also split the momentum equation into several sub-equations including convection, diffusion and stress integration. Courant et al. [5] proposed a simple method based on characteristics for discretizing advection equations. These semi-Lagrangian type schemes are popular in many areas because they can be made unconditionally stable. For example, when we use upwind schemes in the level set convection, negative values may become positive near boundaries where velocities point inward if the CFL condition does not hold. But the semi-Lagrangian schemes will not change the sign. In our moving finite difference framework, we adopt the simplest semi-Lagrangian scheme which traces back a straight line characteristic and uses trilinear interpolation to estimate the data, and thus is first-order accurate in both space and time. The main difference is that this scheme is performed on the logical domain and the velocity field is transformed from the physical domain to the logical domain. Although higher order schemes (e.g., BFECC [9] with semi-Lagrangian building blocks [24]) can be implemented on moving meshes, additional storage and complexity related to Jacobian transformations may become problems. Numerical experiments show that the first-order semi-Lagrangian scheme works quite well on moving meshes, which is consistent with the fact that the moving mesh has the ability to redistribute the errors according to the regularity of the solutions. The diffusion and projection will lead to two Poisson equations which are solved using preconditioned conjugate gradient (PCG) method with Jacobi preconditioner. Here a symmetric discretization is used for free surface conditions in order to use

the fast PCG solver. All the differential operators are discretized on non-uniform moving meshes and we only use Jacobian transformations in the gradient operator approximation. Laplacian operator is discretized using a simple seven point stencil.

The following section will briefly introduce the physical models for incompressible viscoelastic flows and fluid interfaces. The moving mesh method based on harmonic mapping will be discussed in Section 3. In Section 4 we will propose a finite difference based moving mesh method for solving 3D free surface viscoelastic flows. The last two sections will give numerical examples and conclusions.

2. Basic models

2.1. Incompressible flow

The incompressible fluid flow is typically modeled by the incompressible Navier-Stokes equations

$$\nabla \cdot \mathbf{u} = 0, \tag{2.1a}$$

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = \nabla\cdot\left(-p\mathbf{I}+\mathbf{T}\right) + \rho\mathbf{g}, \tag{2.1b}$$

where $\mathbf{u}$ is the velocity, ρ is the density, p is the pressure, $\mathbf{g}$ is the gravity and $\mathbf{T}$ is the extra-stress tensor. These equations are derived from the conservation laws which conserve mass and momentum. The density and temperature of the fluid are assumed to be constant. In Newtonian flows the extra-stress tensor is $\mathbf{T} = 2\mu\mathbf{D}$ where μ is the viscosity and $\mathbf{D} = \frac{1}{2}\left[(\nabla\mathbf{u})^{\mathrm{T}} + \nabla\mathbf{u}\right]$ is the strain rate. This implies that the stress is proportional to the strain rate in Newtonian flows.

2.2. Viscoelastic flow

For viscoelastic flows, the relation between the stress and strain rate is nonlinear due to the long chain molecular structure of the polymer fluids. In this section, we briefly introduce the Upper Convected Maxwell (UCM) model and Oldroyd-B model by Oldroyd [18]. There are other good models such as FENE-P but they do not have much difference in the implementation.

To study viscoelastic fluids, the UCM model is a simple one to begin with. In this model the relation between the extra-stress tensor and the strain rate is given by

$$\mathbf{T} + \lambda\mathbf{T}^{\nabla} = 2\mu\mathbf{D}, \tag{2.2}$$

where λ is the relaxation time of stress, μ is the total viscosity and

$$\mathbf{T}^{\nabla} = \frac{\partial \mathbf{T}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{T} - (\nabla\mathbf{u})^{\mathrm{T}}\cdot\mathbf{T} - \mathbf{T}\cdot(\nabla\mathbf{u}) \tag{2.3}$$

is the upper-convected derivative of stress tensor.

In the Oldroyd-B model the extra-stress tensor is divided into polymeric and Newtonian parts $\mathbf{T} = \tau + 2\mu_s\mathbf{D}$, where μ_s is the Newtonian viscosity. The constitutive relation between polymeric stress and strain rate is given by

$$\tau + \lambda\tau^{\nabla} = 2\mu_p\mathbf{D},$$

where $\mu_p = \mu - \mu_s$ is the polymeric viscosity. Now the conservation of mass, momentum with the constitutive equation can be written as

$$\nabla \cdot \mathbf{u} = 0, \tag{2.4a}$$

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = \nabla\cdot\left(-p\mathbf{I} + \tau + 2\mu_s\mathbf{D}\right) + \rho\mathbf{g}, \tag{2.4b}$$

$$\tau + \lambda\tau^{\nabla} = 2\mu_p\mathbf{D}. \tag{2.4c}$$

2.3. Fluid interfaces

To model fluid interfaces, we adopt phase-field method for two-phase flow and use level set method for free surface flow. Both methods use a scalar field function to represent fluid interfaces implicitly, reads

$$\phi(\mathbf{x}) = c, \tag{2.5}$$

where ϕ is the implicit function and c is a constant.

The level set method was developed for tracking interfaces and shapes by Osher and Sethian [20,21,25]. In the level set method, the function ϕ is defined as a signed distance to the fluid interfaces. Then ϕ is convected by the fluid flow through a simple convection equation

$$\frac{\partial\phi}{\partial t} + \mathbf{u}\cdot\nabla\phi = 0, \tag{2.6}$$

where $\mathbf{u}$ is the fluid velocity. In order to maintain the signed distance to the fluid interfaces after several time steps, we need to reinitialize the function ϕ. In our implementation, we keep the values of the nodes next to the interfaces and calculate the values of the nodes away from the interfaces using a dynamic programming strategy. For each nodes, the values of $5^d - 1$ neighboring nodes may be recalculated. Only a few steps of this process are needed since the nodes far from the interface dose not affect the computation results.

In small-scale fluid motion, surface tension plays an important role. And the momentum equation (2.1b) coupled with surface tension will be

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = \nabla\cdot\left(-p\mathbf{I} + \mathbf{T}\right) + \sigma\kappa\delta\mathbf{n} + \rho\mathbf{g}, \tag{2.7}$$

where σ is the surface tension, κ is the curvature, $\delta = \delta(\phi)$ is the delta function and $\mathbf{n}$ is the outward unit normal. We can obtain the outward unit normal

$$\mathbf{n} = -\frac{\nabla\phi}{|\nabla\phi|} \tag{2.8}$$

and the curvature

$$\kappa = \nabla \cdot \mathbf{n} \tag{2.9}$$

from the implicit level set function $\phi = \phi(\mathbf{x})$. For free surface flow problems, we assume $\phi(\mathbf{x}) > 0$ for all $\mathbf{x}$ inside the fluid.

3. The moving mesh method

3.1. The mesh equation

In Li et al.'s framework [16, 17], the underlying partial differential equation and the mesh equation is solved on unstructured meshes using finite element formulations. However, in finite difference method, a similar mesh equation defined in the logical domain is used

$$\nabla_\xi \cdot \left(M \nabla_\xi \mathbf{x} \right) = 0. \tag{3.1}$$

The above equation comes from a different functional defined on logical domain, see [3, 28]. Eq. (3.1) is used for structured meshes when we solve 3d free surface flow problems in Section 5. And we suggest the following monitor function for the level set method

$$m(\mathbf{x}) = \sqrt{1 + \alpha_1 \exp\left\{ -\frac{\phi^2(\mathbf{x})}{2\epsilon^2} \right\}}, \tag{3.2}$$

where α_1 is a scaling parameter and ϵ is a constant in the order of cell size. This parameter ϵ should be chosen such that the delta function can be discretized within high resolution grids. We choose $\epsilon = 2\tilde{h}$ where $\tilde{h}$ is the logical grid spacing and we will explain it later. Fig. 1 is an example of how the mesh equation (3.1) works on structured grids.

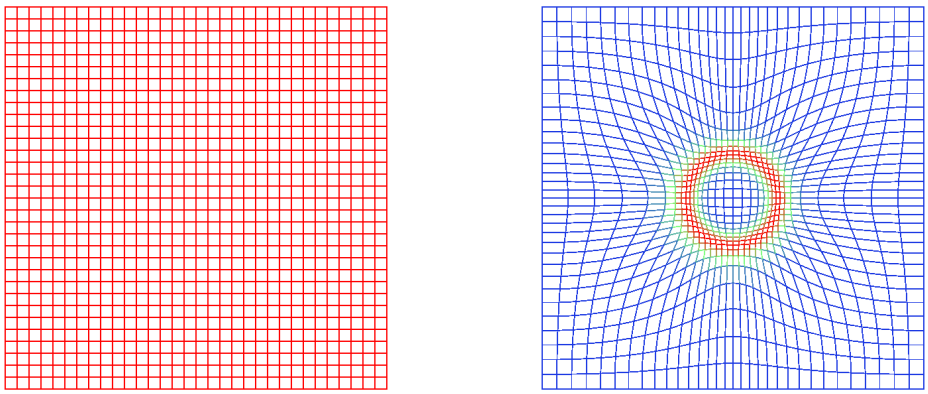

Figure 1: Moving mesh using structured grids. Left: The uniform mesh in logical domain. Right: The physical mesh obtained from Eq. (3.1) whose color represent the values of monitor function.

3.2. Solution updating

Another key issue in moving mesh methods is the solution updating during mesh redistribution process. Fortunately this can be treated as a convection step due to the fact that the mesh nodes are moved continuously if certain iterative method is used for solving Eq. (3.1). In Li's framework, the following equation is solved on unstructured meshes using standard finite element formulation

$$u^\star = u + \mathbf{r}\cdot\nabla u, \tag{3.3}$$

where u is the previous solution, $u^\star$ is the updated solution and $\mathbf{r}$ is the mesh offset. In finite difference based framework, the updating step is performed on the logical domain since the interpolation is much easier on uniform structured meshes. Instead of using (3.3), we suggest the following equation

$$\tilde{u}^\star = \tilde{u} + (\mathbf{J}\mathbf{r})\cdot\nabla_\xi \tilde{u}. \tag{3.4}$$

Here $\tilde{u}$ and $\tilde{u}^\star$ are the previous and updated solutions defined on the logical domain. $\mathbf{J} = \frac{\partial \xi}{\partial \mathbf{x}}$ is the Jacobian transformation from the physical domain to the logical domain. Eq. (3.4) can be solved using explicit semi-Lagrangian type schemes which is discussed in Section 4.

3.3. Choice of time steps

In moving mesh implementation, another important consideration is about adaptive time step due to the varying cell size and velocity field. Time step must be determined based on physical and artificial parameters to avoid inaccuracy or instability so that we do not need to set time step manually. From the basic model equations described in Section 2, we consider the terms involving spatial derivatives except the Laplacian since these operators are treated implicitly. For the momentum equation (2.1b), the CFL condition

$$\Delta t < C\frac{h}{u} \tag{3.5}$$

should be satisfied. Here $C < 1$ is a constant, h is the grid spacing and u is the velocity norm. In the level set method, the surface tension leads to an additional time constraint

$$\Delta t < \sqrt{C\frac{\rho h^3}{\sigma}}, \tag{3.6}$$

where ρ is the density and σ is the surface tension. In general, we use

$$\Delta t < \min\left\{\Delta t_{\max}, C_1\frac{h}{u}, C_2\sqrt{\frac{\rho h^3}{\sigma}}\right\} \tag{3.7}$$

for the level set calculations of free surface flows. Here $C_1, C_2 < 1$ are constants and $\Delta t_{\max}$ is an upper limit of time step. We choose $\Delta t_{\max} = 0.01$ as maximum time step in all numerical experiments.

4. The moving finite difference framework

4.1. Model equations

The free surface viscoelastic flow can be described using the following system

$$\nabla \cdot \mathbf{u} = 0, \tag{4.1a}$$

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = \nabla\cdot\left(-p\mathbf{I} + \tau + 2\mu_s\mathbf{D}\right) + \sigma\kappa\delta\mathbf{n} + \rho\mathbf{g}, \tag{4.1b}$$

$$\tau + \lambda\left[\frac{\partial \tau}{\partial t} + \mathbf{u}\cdot\nabla\tau - (\nabla\mathbf{u})^{\mathrm{T}}\cdot\tau - \tau\cdot(\nabla\mathbf{u})\right] = 2\mu_p\mathbf{D}, \tag{4.1c}$$

$$\frac{\partial \phi}{\partial t} + \mathbf{u}\cdot\nabla\phi = 0, \tag{4.1d}$$

where the Oldroyd-B model is adopted for the viscoelastic flow behaviors and the level set method is used for the fluid interface capturing. Here $\mathbf{u}$ is the velocity, ρ is the density, p is the pressure, $\mathbf{I}$ is the identity tensor, τ is the polymer stress tensor, μ_s is the Newtonian viscosity, $\mathbf{D} = \frac{1}{2}\left[(\nabla\mathbf{u})^{\mathrm{T}} + \nabla\mathbf{u}\right]$ is the strain rate, σ is the surface tension, κ is the curvature of interface, δ is the delta function based on the level set ϕ, $\mathbf{n}$ is the outward unit normal of interface, $\mathbf{g}$ is the gravity, λ is the relaxation time and μ_p is the polymeric viscosity. The boundary conditions at solid walls are:

$$\mathbf{u}\cdot\mathbf{n} = 0, \tag{4.2}$$

$$\frac{\partial p}{\partial \mathbf{n}} = 0, \tag{4.3}$$

$$\frac{\partial \tau}{\partial \mathbf{n}} = 0. \tag{4.4}$$

Other boundary conditions such as inflow and outflow can also be designed for specific problems. In this case, the boundary conditions for the polymer stress are implemented using the zeroth-order extrapolation suggested by Trebotich *et al.* [29]. At fluid interfaces, we also need the free boundary condition:

$$\left(p - p_{air} - \sigma\kappa\right)\mathbf{n} = \left(2\mu_s\mathbf{D} + \tau\right)\cdot\mathbf{n}, \tag{4.5}$$

where p_{air} is the pressure of air.

To ensure the divergence free condition for the velocity field $\mathbf{u}$, we use Chorin's projection method [4] to solve the momentum equation (4.1b) such that the constraint (4.1a) can be satisfied. First, we compute the intermediate velocity field $\mathbf{u}^\star$ explicitly by ignoring the pressure term in the momentum equation (4.1b)

$$\frac{\mathbf{u}^\star - \mathbf{u}_n}{\Delta t} = -\mathbf{u}_n\cdot\nabla\mathbf{u}_n + \frac{1}{\rho}\nabla\cdot\left(\tau + 2\mu_s\mathbf{D}\right) + \frac{1}{\rho}\sigma\kappa\delta\mathbf{n} + \mathbf{g}, \tag{4.6}$$

where $\mathbf{u}_n$ is the velocity field at n^{th} time step. Then we have

$$\frac{\mathbf{u}_{n+1} - \mathbf{u}^\star}{\Delta t} = -\frac{1}{\rho}\nabla p. \tag{4.7}$$

Taking the divergence of both sides and requiring that $\nabla \cdot \mathbf{u}_{n+1} = 0$, we get

$$\frac{\Delta t}{\rho}\nabla^2 p = \nabla \cdot \mathbf{u}^\star. \tag{4.8}$$

Eq. (4.8) is a Poisson equation from which we obtain the pressure p. Finally we can evaluate $\mathbf{u}_{n+1}$ from (4.7). Eq. (4.7) can be rewritten as

$$\mathbf{u}_{n+1} = \mathbf{u}^\star - \frac{\Delta t}{\rho}\nabla p, \tag{4.9}$$

which is the Helmholtz-Hodge decomposition if Neumann boundary condition $\frac{\partial p}{\partial \mathbf{n}} = 0$ on the domain boundary is used. The free surface boundary condition (4.5) should also be applied at fluid interfaces for free surface problems and p_{air} can be set to zero for simple cases.

In the next section, we will fully discretize the above system together with the mesh equation (3.1) using finite difference method. Our goal is to design efficient finite difference schemes for fairly large problems (i.e. 128^3 to 512^3 mesh nodes).

4.2. Finite difference approximation

In this section, we use a simple iterative method to solve the mesh equation (3.1). Then a semi-Lagrangian scheme on moving grids is suggested for solution updating and convection. We also propose the finite difference approximation for the gradient, divergence and Laplacian operators which is required for solving the whole system (4.1a) to (4.1d). Other issues is discussed in the end of this section such as the approximation of surface tension term and the general framework.

4.2.1. Solving the mesh equation

Using the central difference scheme for the Laplacian operator on uniform grids, the mesh equation (3.1) with the monitor function (3.2) can be discretized into the following form

$$\mathbf{A}(m(\mathbf{x}))\,\mathbf{x} = \mathbf{0}, \tag{4.10}$$

where the coefficient matrix $\mathbf{A}$ depends on the monitor function m and the solution $\mathbf{x}$. Here the discretization of Eq. (3.1) can be written as

$$\begin{aligned}
&\frac{m_{i-\frac{1}{2},j,k}\left(\mathbf{x}_{i-1,j,k} - \mathbf{x}_{i,j,k}\right)}{h^2} + \frac{m_{i+\frac{1}{2},j,k}\left(\mathbf{x}_{i+1,j,k} - \mathbf{x}_{i,j,k}\right)}{h^2} \\
&\quad + \frac{m_{i,j-\frac{1}{2},k}\left(\mathbf{x}_{i,j-1,k} - \mathbf{x}_{i,j,k}\right)}{h^2} + \frac{m_{i,j+\frac{1}{2},k}\left(\mathbf{x}_{i,j+1,k} - \mathbf{x}_{i,j,k}\right)}{h^2} \\
&\quad + \frac{m_{i,j,k-\frac{1}{2}}\left(\mathbf{x}_{i,j,k-1} - \mathbf{x}_{i,j,k}\right)}{h^2} + \frac{m_{i,j,k+\frac{1}{2}}\left(\mathbf{x}_{i,j,k+1} - \mathbf{x}_{i,j,k}\right)}{h^2} = 0,
\end{aligned}$$

and

$$m_{i-\frac{1}{2},j,k} = \frac{m_{i-1,j,k} + m_{i,j,k}}{2}, \qquad m_{i+\frac{1}{2},j,k} = \frac{m_{i+1,j,k} + m_{i,j,k}}{2},$$
$$m_{i,j-\frac{1}{2},k} = \frac{m_{i,j-1,k} + m_{i,j,k}}{2}, \qquad m_{i,j+\frac{1}{2},k} = \frac{m_{i,j+1,k} + m_{i,j,k}}{2},$$
$$m_{i,j,k-\frac{1}{2}} = \frac{m_{i,j,k-1} + m_{i,j,k}}{2}, \qquad m_{i,j,k+\frac{1}{2}} = \frac{m_{i,j,k+1} + m_{i,j,k}}{2}.$$

To solve (4.10), we suggest an iterative method and rewrite (4.10) as

$$\mathbf{A}_k \mathbf{x}_k = \mathbf{0}, \tag{4.11}$$

where $\mathbf{x}_k$ is the solution at k^{th} step and $\mathbf{A}_k = \mathbf{A}\left(m\left(\mathbf{x}_k\right)\right)$ is the approximation of the left hand side of (3.1) at k^{th} step. For each step, we decompose $\mathbf{A}_k$ as

$$\mathbf{A}_k = \mathbf{D}_k - \mathbf{L}_k - \mathbf{U}_k, \tag{4.12}$$

where $\mathbf{D}_k$, $-\mathbf{L}_k$ and $-\mathbf{U}_k$ are the diagonal part, lower and upper triangular part of $\mathbf{A}_k$, respectively. Then the offset of mesh nodes $\Delta\mathbf{x}_k$ can be obtained using

$$\Delta\mathbf{x}_k = \frac{1}{2}\left[\mathbf{D}_k^{-1}\left(\mathbf{L}_k + \mathbf{U}_k\right)\mathbf{x}_k - \mathbf{x}_k\right], \tag{4.13}$$

and the mesh is updated through

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k. \tag{4.14}$$

After each mesh updating step (4.14), we should initialize or update the solution on the new mesh. In our framework, the mesh is moved in two situations. Before the simulation, the initial moving of mesh nodes is based on the initial values of certain solutions such as the initial level set or the velocity field which guarantees the simulation will start with a high quality adaptive mesh rather than a uniform mesh. Therefore we need to re-initialize the solution immediately after (4.14). During the simulation, the mesh will move as the solution changes and we need to 'move' the solution as well, using (3.4) with $\mathbf{r} = \Delta\mathbf{x}_k$. It is pointed out that the number of moving iterations for the preprocessing is much more than that in each simulation step since the evolution of the level set and velocity field are continuous. Continuous adaptation is also an advantage of moving mesh method compared to other adaptive mesh methods since the computational cost for the mesh adaptation is much lower (i.e. one or two moving iterations per step without memory reallocation). A straight forward way to accelerate the initial moving process is to use multi-grid techniques which is easy to implement on structured grids.

4.2.2. The semi-Lagrangian scheme for solution updating and convection

Courant *et al.* [5] proposed a simple method of characteristics scheme for discretizing advection equations. These semi-Lagrangian type schemes are popular in many areas because

they can be made unconditionally stable. Here we extend this scheme on moving meshes due to its simplicity and stability. Both the solution updating and the convection for the level set, velocity and polymer stress can be written in the semi-discretized form on the physical domain

$$\frac{\varphi(\mathbf{x}, t+\Delta t)-\varphi(\mathbf{x}, t)}{\Delta t}+\mathbf{v}\cdot\nabla\varphi=0, \tag{4.15}$$

where φ is the field to be convected and $\mathbf{v}$ is the velocity of the convection. We set $\Delta t=1$ and $\mathbf{v}=-\mathbf{r}$ for solution updating where $\mathbf{r}$ is the offset of mesh nodes. For other convection terms in (4.1a) to (4.1d), we use $\mathbf{v}=\mathbf{u}$, and φ can be ϕ, $\mathbf{u}$ or τ. Rewrite (4.15) on the logical domain, we have

$$\frac{\tilde{\varphi}(\xi, t+\Delta t)-\tilde{\varphi}(\xi, t)}{\Delta t}+(\mathbf{J}\mathbf{v})\cdot\nabla\tilde{\varphi}=0, \tag{4.16}$$

where $\mathbf{J}$ is the Jacobian transformation from the physical domain to the logical domain. At this point, we use

$$\tilde{\varphi}(\xi, t+\Delta t)=\tilde{\varphi}(\xi-(\mathbf{J}\mathbf{v}\Delta t), t) \tag{4.17}$$

to evaluate the solution on the logical domain for each mesh nodes. Here $\tilde{\varphi}(\xi-(\mathbf{J}\mathbf{v}\Delta t), t)$ is approximated using trilinear interpolation because it is unconditionally stable which does not introduce new extrema at high gradient regions. The dissipation effect of this scheme on the moving grid is further reduced compared to the uniform grid scheme.

4.2.3. Velocity and polymer stress extrapolation

For free surface problems, the velocity field and the polymer stress field on the air side is undefined. In order to make the semi-Lagrangian convection and other finite difference schemes work correctly around the fluid interfaces, we need to extrapolate the velocity and polymer stress from the fluid side to the air side. Our extrapolation process is done through diffusion using the following equations defined in Ω_{air}:

$$\frac{\partial\mathbf{u}}{\partial t}=\nabla\cdot(\nu\nabla\mathbf{u}), \tag{4.18}$$

$$\frac{\partial\tau}{\partial t}=\nabla\cdot(\nu\nabla\tau), \tag{4.19}$$

where

$$\nu(\mathbf{x})=\begin{cases}0, & \mathbf{x}\in\Omega_{air},\\ 10^6, & \mathbf{x}\in\partial\Omega_{air}.\end{cases} \tag{4.20}$$

We fix the velocity and polymer stress on the fluid side and diffuse it to the air side using a sufficiently large diffusion coefficient. To get the steady state solution, the time derivative is removed from the diffusion equations:

$$\nabla\cdot(\nu\nabla\mathbf{u})=0, \tag{4.21}$$

$$\nabla\cdot(\nu\nabla\tau)=0. \tag{4.22}$$

Rather than forming the stiffness matrix and solving the whole implicit linear system, we only perform a few Jacobi iterations to update the velocity and stress values near the fluid interfaces.

4.2.4. Approximation of differential operators on moving meshes

In the system (4.1a) to (4.1d), we have three types of spatial derivatives to discretize. Assume $u(\mathbf{x})$ is a continuous function defined in the physical domain Ω and $\tilde{u}(\xi)$ is the same function defined in the logical domain $\tilde{\Omega}$. We also let

$$\mathbf{J}(\xi) = \left(\frac{\partial \mathbf{x}}{\partial \xi}\right)^{-1}$$

be the Jacobian transformation from the logical domain to the physical domain. For the gradient operator, we use the following transformation

$$\nabla u = \mathbf{J}^{\mathrm{T}} \nabla_{\xi} \tilde{u}, \tag{4.23}$$

where ∇_{ξ} is the gradient operator in the logical domain. If both the Jacobian $\mathbf{J}$ and the gradient ∇_{ξ} are discretized using second order central difference, this scheme will have second order accuracy in space. However, using (4.23) directly can lead to new extrema and possible instability. We have tested (4.23) in our simulations and the computations always break down after certain projection steps when the meshes are non-uniform. A remedy is to limit the gradient values evaluated through (4.23). We use the slopes of u on the physical meshes as the limit values so that ∇u can be bounded. For the divergence operator, the following approximation is used

$$\nabla \cdot \mathbf{u} = \mathbf{J} : \nabla_{\xi} \tilde{\mathbf{u}}, \tag{4.24}$$

where $\mathbf{u}$ is a vector function.

It is obvious that (4.23) and (4.24) will reduce to the standard central difference approximation of the gradient and divergence operator on uniform orthogonal grids. However, we realized that approximating the Laplacian operator by using Jacobian transformation is not practical in 3D cases since we need to compute so many auxiliary variables and at least 19 stencil points should be used. Therefore we suggest a 7 point scheme which can be evaluated directly on the physical mesh. First, we define the following forward and backward difference notations

$$\begin{aligned}
&\delta_i^+ u_{i,j,k} = u_{i+1,j,k} - u_{i,j,k}, \qquad && \delta_j^+ u_{i,j,k} = u_{i,j+1,k} - u_{i,j,k},\\
&\delta_k^+ u_{i,j,k} = u_{i,j,k+1} - u_{i,j,k}, \qquad && \delta_i^- u_{i,j,k} = u_{i,j,k} - u_{i-1,j,k},\\
&\delta_j^- u_{i,j,k} = u_{i,j,k} - u_{i,j-1,k}, \qquad && \delta_k^- u_{i,j,k} = u_{i,j,k} - u_{i,j,k-1},
\end{aligned}$$

and we let

$$\begin{aligned}
\Delta_i \mathbf{x}_{i,j,k} &= \frac{1}{2}\left\|\mathbf{x}_{i+1,j,k} - \mathbf{x}_{i-1,j,k}\right\|_2, \\
\Delta_j \mathbf{x}_{i,j,k} &= \frac{1}{2}\left\|\mathbf{x}_{i,j+1,k} - \mathbf{x}_{i,j-1,k}\right\|_2, \\
\Delta_k \mathbf{x}_{i,j,k} &= \frac{1}{2}\left\|\mathbf{x}_{i,j,k+1} - \mathbf{x}_{i,j,k-1}\right\|_2, \\
\Delta_i^+ \mathbf{x}_{i,j,k} &= \left\|\mathbf{x}_{i+1,j,k} - \mathbf{x}_{i,j,k}\right\|_2, \qquad \Delta_j^+ \mathbf{x}_{i,j,k} = \left\|\mathbf{x}_{i,j+1,k} - \mathbf{x}_{i,j,k}\right\|_2, \\
\Delta_k^+ \mathbf{x}_{i,j,k} &= \left\|\mathbf{x}_{i,j,k+1} - \mathbf{x}_{i,j,k}\right\|_2, \qquad \Delta_i^- \mathbf{x}_{i,j,k} = \left\|\mathbf{x}_{i,j,k} - \mathbf{x}_{i-1,j,k}\right\|_2, \\
\Delta_j^- \mathbf{x}_{i,j,k} &= \left\|\mathbf{x}_{i,j,k} - \mathbf{x}_{i,j-1,k}\right\|_2, \qquad \Delta_k^- \mathbf{x}_{i,j,k} = \left\|\mathbf{x}_{i,j,k} - \mathbf{x}_{i,j,k-1}\right\|_2.
\end{aligned}$$

Then the approximation of the Laplacian operator can be written as

$$\nabla^2 u \approx \frac{\frac{\delta_i^+ u_{i,j,k}}{\Delta_i^+ \mathbf{x}_{i,j,k}} - \frac{\delta_i^- u_{i,j,k}}{\Delta_i^- \mathbf{x}_{i,j,k}}}{\Delta_i \mathbf{x}_{i,j,k}} + \frac{\frac{\delta_j^+ u_{i,j,k}}{\Delta_j^+ \mathbf{x}_{i,j,k}} - \frac{\delta_j^- u_{i,j,k}}{\Delta_j^- \mathbf{x}_{i,j,k}}}{\Delta_j \mathbf{x}_{i,j,k}} + \frac{\frac{\delta_k^+ u_{i,j,k}}{\Delta_k^+ \mathbf{x}_{i,j,k}} - \frac{\delta_k^- u_{i,j,k}}{\Delta_k^- \mathbf{x}_{i,j,k}}}{\Delta_k \mathbf{x}_{i,j,k}}. \tag{4.25}$$

The above approximation (4.25) is compatible with the standard central difference scheme for the Laplacian on orthogonal grids but also works on moving grids. We use (4.25) to approximate the viscosity term $\nabla \cdot \left(2\mu_s \mathbf{D}\right) = \mu_s \nabla^2 \mathbf{u}$ in Eq. (4.1b) which is solved implicitly by moving the term to the left hand side. The left hand side of Eq. (4.8) is also approximated using (4.25). However, special care should be taken for both (4.1b) and (4.8) when the fluid interfaces are present. Here we use a formula which is similar to [12] and the boundary conditions at fluid interfaces are

$$\frac{\partial \mathbf{u}}{\partial \mathbf{n}} = 0 \tag{4.26}$$

and

$$p = p_{air} + \sigma\kappa + \left(2\mu_s \mathbf{D} + \tau\right) \cdot \mathbf{n} \cdot \mathbf{n}. \tag{4.27}$$

For example, if we want to evaluate

$$\frac{u_{i+1,j,k} - u_{i,j,k}}{\mathbf{x}_{i+1,j,k} - \mathbf{x}_{i,j,k}}, \tag{4.28}$$

where $\mathbf{x}_{i,j,k} \in \Omega$ but $\mathbf{x}_{i+1,j,k} \notin \Omega$, then we evaluate

$$\frac{u_{fs} - u_{i,j,k}}{\mathbf{x}_{fs} - \mathbf{x}_{i,j,k}}, \tag{4.29}$$

where u_{fs} is the free surface boundary condition for u and $\mathbf{x}_{fs}$ is the location of fluid interface between $\mathbf{x}_{i,j,k}$ and $\mathbf{x}_{i+1,j,k}$. One advantage of the above discretization is that the resulting linear system is still symmetric positive definite therefore fast preconditioned conjugate gradient solvers can be applied. Here the Jacobi preconditioner works well for Poisson equations discretized on moving meshes. For example, a 262144×262144 sparse linear system can be solved within 500ms on a 2.4GHz CPU.

4.2.5. Approximation of delta function

The delta function δ in the surface tension term of equation (4.1b) can be approximated using the approach described in [27]:

$$\delta_\epsilon(\phi) = \begin{cases} \dfrac{1+\cos\left(\frac{\pi\phi}{\epsilon}\right)}{2\epsilon}, & |\phi| < \epsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{4.30}$$

Here ϵ is a parameter which should be larger than the grid spacing near the interface and we set $\epsilon = \tilde{h}$ where $\tilde{h}$ is the logical grid spacing. As we mentioned in Chapter 3, we choose $\epsilon = 2\tilde{h}$ for the monitor function (3.2). This will ensure that the delta function can be discretized within high resolution grids. The error introduced by smoothing the surface tension force is $\mathcal{O}(\tilde{h})$, see [2]. Here $\tilde{h}$ is the average grid spacing near the interface on a moving mesh which is smaller than the uniform grid spacing h. Therefore the error of surface tension approximation is reduced on a moving mesh.

4.2.6. General framework

In general, our moving finite difference framework works as follows:

1. Initialize level-set function on the mesh.
2. Compute monitor function and move the current mesh.
3. Go to step 4 if the mesh reaches a desirable quality, otherwise go to step 1.
4. Solve the system (4.1a) to (4.1d) on the mesh.
   - Update the Jacobian transformation using central difference schemes.
   - Update the time step based on the CFL condition.
   - Convect the level set, velocity and polymer stress using semi-Lagrangian scheme.
   - Apply the polymer stress, surface tension and gravity to the velocity field explicitly.
   - Project the velocity field to satisfy the divergence free condition.
   - Diffuse the velocity field implicitly.
   - Project the velocity field to satisfy the divergence free condition.
   - Apply other terms of the constitutive equation to the polymer stress explicitly.
5. Compute new monitor function and move the mesh.
6. Update all solutions on the new mesh.
7. Go to step 4 if the mesh reaches a desirable quality, otherwise go to step 5.

5. Numerical examples

5.1. Rotating sphere

The first numerical experiment is to validate the accuracy and efficiency of our semi-Lagrangian scheme on the moving mesh. In this test, a three dimensional sphere of radius 0.25 is placed in the center of a $1 \times 1 \times 1$ domain and a rotational velocity field with axis $(1,1,1)$ and angular velocity $2\pi(rad/s)$ is imposed on the whole domain. We compare the shape of sphere to its original shape and measure the L^∞-error of the spherical interface. Both uniform meshes and moving meshes with different resolutions are used and we do not re-initialize the level set function in the computation because this process will introduce additional errors. The parameters for the monitor function (3.2) are $\alpha = 1000$ and $\epsilon = 2\tilde{h}$ where $\tilde{h}$ is the logical grid spacing. We use the time step $\Delta t = 0.1 h_{\min}$ where h_{min} is the smallest physical grid spacing. Table 1 is an error comparison at $t = 0.1$ between uniform meshes and moving meshes.

Table 1: Accuracy and performance comparison between moving mesh and uniform mesh. The result is sorted by L^∞-error of the interfaces at $t = 0.1$.

Mesh Size	Mesh Type	L^∞ Error	CPU Time
64^3	Moving	2.04e-3	13419ms
128^3	Uniform	3.16e-3	62735ms
32^3	Moving	5.79e-3	577ms
64^3	Uniform	6.32e-3	3828ms

Theoretically speaking, the sphere should maintain its shape exactly. The errors are introduced by the finite difference scheme. From Table 1, we see that the semi-Lagrangian scheme works better on the moving meshes. The error produced by 64^3 moving mesh simulation is smaller than the error produced by 128^3 uniform mesh simulation. This is also true between 32^3 moving mesh and 64^3 uniform mesh. Meanwhile, a considerable amount of CPU time is saved because we use much less computational nodes. We also calculated the convergence order for all cases which is listed in Table 2.

Table 2: Error and convergence order of all test cases.

Mesh Size	Uniform Mesh		Moving Mesh	
	Error	Order	Error	Order
16^3	2.50e-2	-	1.69e-2	-
32^3	1.26e-2	0.99	5.79e-3	1.55
64^3	6.32e-3	0.99	2.04e-3	1.51
128^3	3.16e-3	1.00	9.56e-4	1.09

In Table 2, the convergence order of uniform mesh simulations is stable. The order from moving meshes does not mean much since the mesh size is unpredictable. We also measured the computational cost of all test cases. We run the simulations on a PC with one 2.4GHz CPU and 2GB RAM. Table 3 records the CPU time of the numerical tests above.

Table 3: CPU time statistics.

Size	Uniform Mesh	Moving Mesh
16^3	1ms	32ms
32^3	234ms	577ms
64^3	3828ms	13419ms
128^3	62735ms	264109ms

With the help of moving mesh strategy, we can use less memory and time cost to achieve better accuracy in the level set calculations. From these test cases, we validated that the semi-Lagrange scheme for moving meshes works properly in rotational divergence free velocity fields. Fig. 2 is a slice of mesh from which we see that the mesh cells are redistributed according to the interface.

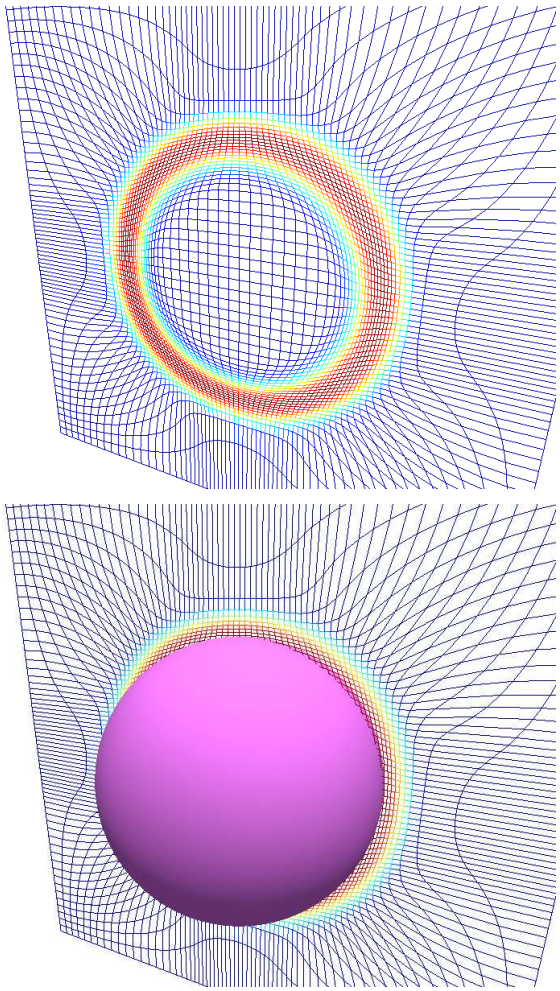

Figure 2: A slice of mesh in the numerical example in Section 5.1.

5.2. Relaxation of cubic drops

In this numerical experiment, we solve the full system of (4.1a) to (4.1d) to study the surface tension effect and non-Newtonian effect. A cubic drop of size $0.002\times0.002\times0.002$ is placed in the center of a $0.01\times0.01\times0.01$ domain and its shape will become round as the drop relaxed due to the surface tension. The physical parameters are $\rho=1000$, $\mu_s=\mu_p=0.05$, $\sigma=0.05$ and $\lambda=1$. We test the convergence of our moving finite difference schemes through comparing the shape deformation history to the results from uniform mesh simulations. Here we use the 128^3 uniform mesh simulation as a reference. The deformation is defined as $D=(L-S)/(L+S)$ where L and S are the longest and shortest distances from the drop center to the interface. Fig. 3 plots the values of D using both uniform and moving meshes with different resolutions.

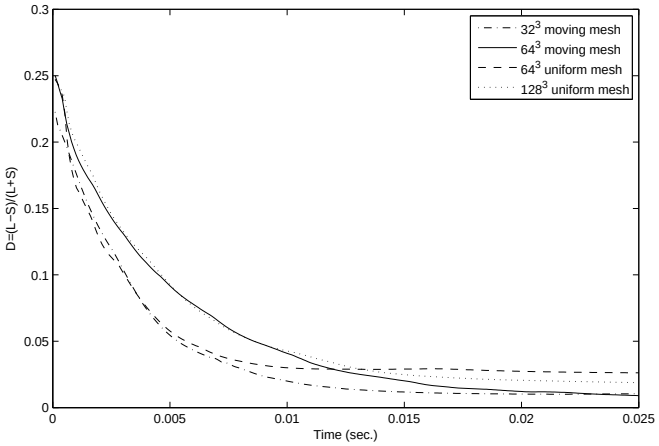

Figure 3: Deformation history of a cubic viscoelastic drop. Both uniform and moving meshes with different resolutions are used.

In Fig. 3, the values of D decrease very fast at the beginning. This is caused by large surface tensions at the eight corners of the initial cubic drop where the curvature is large. We find that the 32^3 moving mesh result and the 64^3 uniform result are close to each other while the 64^3 moving mesh result is close to 128^3 uniform mesh result during this period. Here the relaxation of the drop is faster on coarse meshes. This may be caused by the lower viscous and viscoelastic forces since these forces dissipate faster on coarse meshes. At the end, the values of D should be zero since the shape of the drop will become a sphere. Here the moving mesh results are better than the uniform mesh results. The reason is that we have much smaller cells near fluid interfaces when using moving meshes and the approximation of the fluid interfaces is better. In general, if we use the 128^3 uniform result as a reference, we conclude that the 64^3 moving mesh works well. We achieve better accuracy and less computational cost which is consistent to the previous numerical experiment in Section 5.1.

The volume change percentage of the drop is also measured and plotted in Fig. 4.

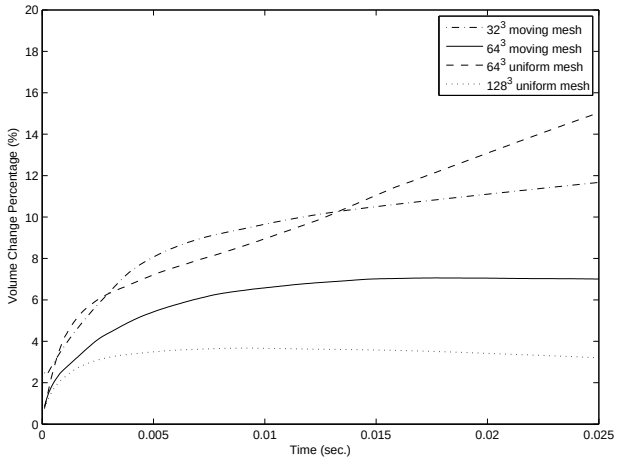

Figure 4: Volume change percentage of a cubic viscoelastic drop. Both uniform and moving meshes with different resolutions are used.

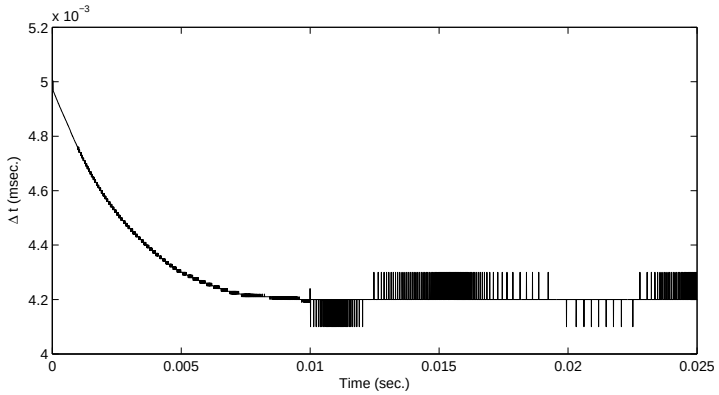

Figure 5: History of adaptive time steps of 64^3 moving mesh simulation. Here the surface tension constraint (3.6) is dominant.

It is shown in Fig. 4 that the volume of the drop in the 64^3 moving mesh simulation and the 128^3 uniform mesh simulation is controlled while others are not. The worst case is the 64^3 uniform mesh and the best is the 128^3 uniform mesh. This concludes that the convergence order of our projection scheme is higher on uniform meshes because of the mesh orthogonality. On the other hand, moving mesh will further reduce the volume change compared to the uniform mesh with the same resolution.

The history of adaptive time steps is plotted in Fig. 5. Here the surface tension constraint (3.6) is dominant therefore the time step is depend on the smallest cell size. This figure implies that the smallest cell size decreases continuously until the drop is relaxed. The reason behind the phenomenon is that the surface area of the drop decreases and more cells are distributed per unit area.

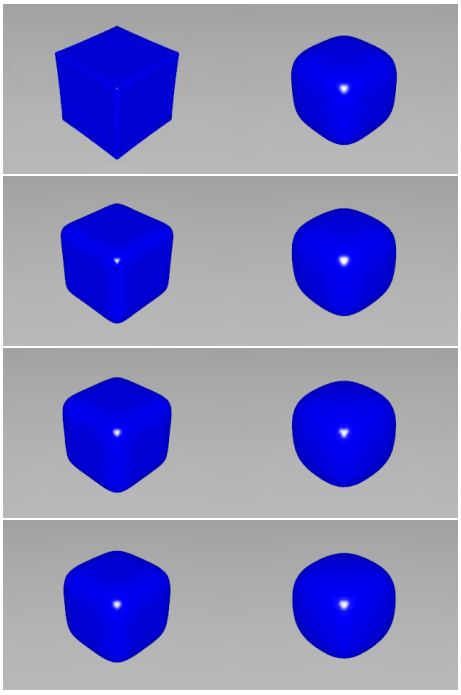

Figure 6: Full rendering of the drop deformation history. Left: t=0ms/1ms/2ms/3ms; Right: t=4ms/5ms/6ms/7ms.

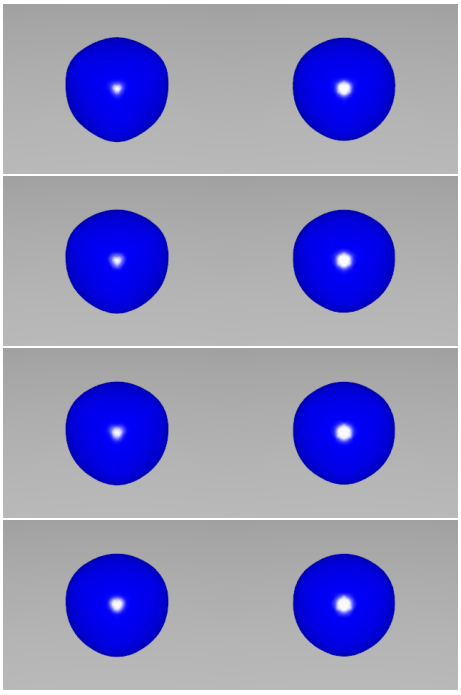

Figure 7: Full rendering of the drop deformation history. Left: t=8ms/9ms/10ms/11ms; Right: t=12ms/13ms/14ms/15ms.

5.3. Interaction with moving solid boundary

In this numerical test, we coupled a moving solid boundary to the flow field. The domain size is $0.1 \times 0.1 \times 0.1$ and a layer of viscoelastic fluids of height 0.02 is placed at the bottom. In addition, a vertical solid cylinder of diameter 0.01 is placed at the center of the domain. The physical parameters are $\rho = 1000$, $g = 9.8$, $\mu_s = \mu_p = 0.1$ and $\lambda = 10$. The cylinder will rise with vertical speed 0.1 until its bottom reaches the domain boundary. The fluid is attached to the cylinder and we use Dirichlet boundary condition for velocity and Neumann boundary condition for the level set function. In Fig. 8 we compare the results between the moving mesh simulation and uniform mesh simulations with different mesh sizes. The moving mesh results agreed well with the higher resolution uniform mesh results.

6. Conclusions

In this work, we simulated free surface viscoelastic flows using harmonic map based moving mesh method. The basic model equations is based on the incompressible Navier-Stokes equations and we adopt the Oldroyd-B constitutive model for viscoelastic flows. In addition, the level set method is adopted for free surface flows. We designed a novel

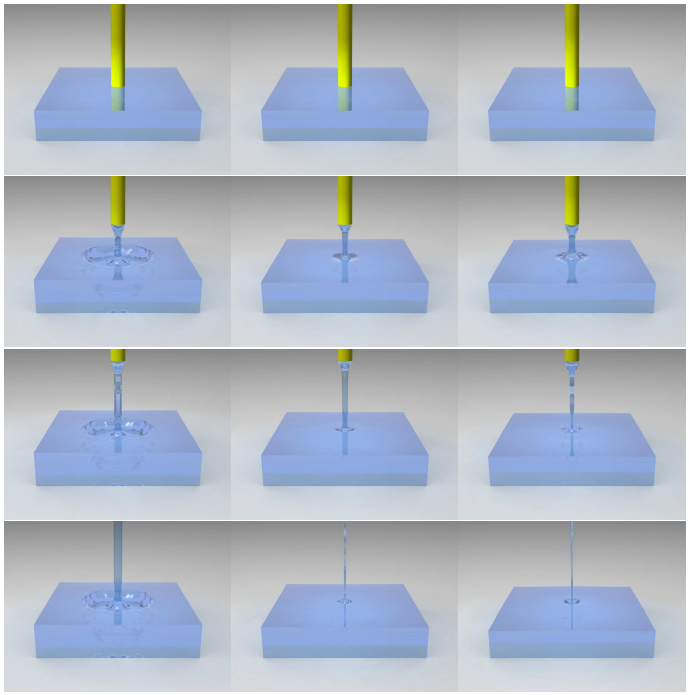

Figure 8: Full rendering of the simulation with t=0s/0.25s/0.5s/2s. Left: 32^3 uniform mesh simulation; Middle: 64^3 uniform mesh simulation; Right: 32^3 moving mesh simulation.

and efficient moving finite difference framework for 3D free surface viscoelastic flows in which we proposed a logical domain semi-Lagrange scheme for moving mesh solution interpolation and convection. The new framework is aimed to 3D free surface problems which has millions of mesh nodes and the memory usage and time cost is optimized. Through numerical experiments, we validated the accuracy and efficiency of our solver. The convergence test for the moving finite difference method show that moving mesh method is effective for viscoelastic flows with moving boundaries.

In the future, we will give some error estimate of our moving finite difference approximation. Moreover, complex numerical experiments will be carried out to study how the polymeric viscosity and relaxation time will affect the fluid motion. In addition, parallel implementation of our framework will also be done on graphics processing unit (GPU) since some preliminary results shown that many numerical schemes and solvers runs tens or hundreds times faster on GPUs.

References

[1] D.M. Anderson, G.B. McFadden, and A.A. Wheeler. Diffuse-interface methods in fluid mechanics. *Annual Reviews in Fluid Mechanics*, 30(1):139–165, 1998.

[2] J.U. Brackbill, D.B. Kothe, and C. Zemach. A continuum method for modeling surface tension. *Journal of Computational Physics*, 100(2):335–354, 1992.

[3] H.D. Ceniceros and T.Y. Hou. An efficient dynamically adaptive mesh for potentially singular solutions. *Journal of Computational Physics*, 172(2):609–639, 2001.
[4] A.J. Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968.
[5] R. Courant, E. Isaacson, and M. Rees. On the solution of nonlinear hyperbolic differential equations by finite differences. *Commun. Pure Appl. Math.*, 5:243–255, 1952.
[6] Y. Di, R. Li, and T. Tang. A general moving mesh framework in 3D and its application for simulating the mixture of multi-phase flows. *Commun. Comput. Phys*, 3:582–602, 2008.
[7] Y. Di, R. Li, T. Tang, and P. Zhang. Moving mesh finite element methods for the incompressible Navier-Stokes equations. *SIAM J. Sci. Comput.*, 26(3):1036–1056, 2005.
[8] Y. Di, R. Li, T. Tang, and P. Zhang. Level set calculations for incompressible two-phase flows on a dynamically adaptive grid. *J. Sci. Comput.*, 31(1-2):75–98, 2007.
[9] T.F. Dupont and Y. Liu. Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *Journal of Computational Physics*, 190(1):311–324, 2003.
[10] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.
[11] W. M. Feng, P. Yu, S. Y. Hu, Z. K. Liu, Q. Du, and L. Q. Chen. A fourier spectral moving mesh method for the cahn-hilliard equation with elasticity. *Commun. Comput. Phys*, 5:582–599, 2009.
[12] F. Gibou, R.P. Fedkiw, L.T. Cheng, and M. Kang. A second-order-accurate symmetric discretization of the Poisson equation on irregular domains. *Journal of Computational Physics*, 176(1):205–227, 2002.
[13] C.W. Hirt and B.D. Nichols. Volume of fluid/VOF/ method for the dynamics of free boundaries. *Journal of Computational Physics*, 39:201–225, 1981.
[14] X.-L. Hu, R. Li, and T. Tang. A multi-mesh adaptive finite element approximation to phase field models. *Commun. Comput. Phys*, 5:1012–1029, 2009.
[15] D. Jacqmin. Calculation of two-phase Navier-Stokes flows using phase-field modeling. *Journal of Computational Physics*, 155(1):96–127, 1999.
[16] R. Li, T. Tang, and P. Zhang. A moving mesh finite element algorithm for singular problems in two and three space dimensions. *Journal of Computational Physics*, 177(2):365–393, 2002.
[17] R. Li, P. Zhang, and T. Tang. Moving mesh methods in multiple dimensions based on harmonic maps. *Journal of Computational Physics*, 170(2):562–588, 2001.
[18] J.G. Oldroyd. Non-Newtonian flow of liquids and solids. *Rheology: Theory and Applications*, 1:653–682, 1956.
[19] E.S. Oran and J.P. Boris. *Numerical Simulation of Reactive Flow*. Cambridge University Press, 2001.
[20] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences, 2003.
[21] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed- Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.
[22] Z.-H. Qiao. Numerical investigations of the dynamical behaviors and instabilities for the gierer-meinhardt system. *Communications in Computational Physics*, 3:406–426, 2008.
[23] R. Scardovelli and S. Zaleski. Direct numerical simulation of free-surface and interfacial flow. *Annual Review of Fluid Mechanics*, 31(1):567–603, 1999.
[24] A. Selle, R. Fedkiw, B.M. Kim, Y. Liu, and J. Rossignac. An unconditionally stable MacCormack method. *Journal of Scientific Computing*, 35(2):350–371, 2008.
[25] J.A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computa-*

tional Geometry, Fluid Mechanics, Computer Vision, and Materials Science. Cambridge University Press, 1999.

[26] M. Sussman and E.G. Puckett. A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows. *Journal of Computational Physics*, 162(2):301–337, 2000.

[27] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114(1):146–159, 1994.

[28] H.-Z. Tang and T. Tang. Adaptive mesh methods for one-and two-dimensional hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, 41(2):487–515, 2004.

[29] D. Trebotich, P. Colella, and G.H. Miller. A stable and convergent scheme for viscoelastic flow in contraction channels. *Journal of Computational Physics*, 205(1):315 – 342, 2005.

[30] A. van Dam and P.A. Zegeling. Balanced monitoring of flow phenomena in moving mesh methods. *Commun. Comput. Phys*, 7:138–170, 2010.

[31] H.-Y. Wang and R. Li. Mesh sensitivity for numerical solutions of phase-field equations using r-adaptive finite element methods. *Communications in Computational Physics*, 3:357–375, 2008.

[32] H.-Y. Wang, R. Li, and T. Tang. Efficient computation of dendritic growth with r-adaptive finite element methods. *Journal of Computational Physics*, 227(12):5984–6000, 2008.

[33] K. Yokoi. A numerical method for free-surface flows and its application to droplet impact on a thin liquid layer. *Journal of Scientific Computing*, 35(2):372–396, 2008.

[34] P. Yue, C. Zhou, J.J. Feng, C.F. Ollivier-Gooch, and H.H. Hu. Phase-field simulations of interfacial dynamics in viscoelastic fluids using finite elements with adaptive meshing. *Journal of Computational Physics*, 219(1):47–67, 2006.