Layer-based Simulation for Three-Dimensional Fluid Flow in Spherical Coordinates

Ruihong Cen, Bo Ren

Abstract—Fluid flows in spherical coordinates have raised the interest of the graphics community in recent years. The majority of existing works focus on 2D manifold flows on a spherical shell, and there are still many unresolved problems for 3D simulations in spherical coordinates, such as boundary conditions for arbitrary obstacles and flexible artistic controls. In this paper, we propose a practical spherical-coordinate simulator for flow motions in 3D domains. Based on a layer-by-layer structure and a boundary-aware pressure solving scheme, we are able to recover horizontal and vertical flow motions in the presence of arbitrary terrain shapes within a spherical shell of finite thickness. Our proposed method straightforwardly builds on the conventions of previous 2D-manifold spherical-coordinate simulations and provides flexible artistic control strategies for art design.

Index Terms—Physically based animation, fluid simulation.

1 Introduction

Visually charming flow motions appear within finite-thickness spherical shells. For example, the flows on the planet and interactions with complex topography make up various interesting and magnificent meteorological phenomena, which often appear in science fiction movies and attract much effort towards the simulation of them. In the meteorological science community, the question of how the atmosphere evolves on Earth under some determined initial conditions is well studied. There exists some general atmospheric circulation models(or climate models) that are widely used in the area of meteorological science, such as ECHAM [1], NCAR-CCSM [2], GFS model, etc. Although the above climate models predict the evolution of the atmosphere with high accuracy, the computational costs of these models are extremely high in addition to their complexity, which is a great challenge to the computer graphics community. On the other hand, in graphics one may want in the animation that the thickness of the fluid domain is to some extent comparable with the ground radius, which would have a far larger ratio than that of earth atmosphere thickness with its radius. For such simulations, a practical physical model of the fluid dynamics system for spherical-coordinate flow motions in 3D domains is required. Compared with Cartesian-coordinate based ones, simulation in the spherical domain is better suited for the precise recovery and control of motions featuring radial-tangential flows, e.g. precisely controlling laminar flows at constant height over most of the sphere surface but reproduce vertical motion at certain places to mimic a meteorological effect.

There have been some good works addressing the simulation of fluid in the spherical coordinate, most of them focusing on the two-dimensional surface of a sphere. Hill and Henderson [3] achieved success in simulating incompressible flow on the surface of a sphere in spherical coordinate and noted that the geometric terms from the equation of fluid motion are physically important. They introduced a spectral filter to enforce the boundary conditions at the North pole and the South pole to solve the singularities problem near poles. Yang et al. [4] improved the previous work [3] by using staggered grid discretization and proposed a more simple treatment at poles. Huang et al. [5] proposed a novel local advection scheme on the surface of a sphere to avoid geometric terms. In addition to these grid-based methods, there are other methods using meshes to discretize the surface [6] [7] [8]. Ishida et al. [9], Ringler et al. [10] and Oron et al. [11] considered the thickness of the spherical structure for thin-film simulation. Despite the success of this thin-film model, they only consider the fluid motion within a thin layer near the spherical surface. Recently, Cui et al. [12] propose a decomposition-based method for 3D flows in spherical or cylindrical coordinates and achieve impressive results. However, artificial propelling forces are applied for arbitrary obstacle boundary conditions in their works, and may face difficulty integrating with existing layered models widely adopted for the simulation of clouds or general circulations (e.g. [13]

In this paper, we propose a novel method to simulate the 3D flow motions under spherical coordinate. Our model provides physically-based calculation of fluid flows within 3D spherical coordinates ϕ, θ, r, with $\phi \in (0, 2\pi)$, $\theta \in (0, \pi)$ and $r \in (r_{min}, r_{max})$, where the arbitrary boundaries from landscape shapes are handled physically with a geometry-aware projection solver , which integrates spherical harmonics expansion with a volume ratio scheme to solve the Poisson equations. To resolve the low computational efficiency of a full 3D spherical harmonics expansion solver and to provide more control convenience of the radial-tangential flow motions within the spherical coordinates, we further decompose the spherical shell domain into a layer-by-layer structure. The layers are discretized by a staggered grid of the same size in radian (described in Fig.1 and section 3.2) with unified distance between each other. A rough 3D solver is first applied in the 3D domain and then followed by a 2D

- *Ruihong Cen and Bo Ren are with TMCC, College of Computer Science, Nankai University, Tianjin, 300000, China.*
- *Corresponding author: Bo Ren (rb@nankai.edu.cn).*

precise solver of the 2D-Poisson equations on the layers, which is integration-friendly with existing methods on 2D spherical-coordinate simulations. Our method is able to capture fluid phenomena featuring flow-ground interaction with arbitrary terrain shapes and combined horizontal and vertical motion of swirling flows under the spherical coordinate. In summary, the main contributions of our paper are as follows:

- A 3D spherical-coordinate solver for incompressible inviscid fluid flow on planetary in computer graphics with physical handling of arbitrary landscape boundary conditions.
- A layered-structure algorithm that is able to efficiently perform graphical 3D spherical incompressible inviscid fluid flow simulations.
- A flexible method allowing intuitive artistic controls on the planetary flow.

2 Related works

Fluid simulations in Cartesian coordinates have been thoroughly studied in computer graphics, such as using Eulerian grid [14], Lagrangian particles [15] [16], or hybrid methods [17] [18]. However, flows within spherical manifolds, such as atmospheric flows or flows on bubble surfaces, that evolves on spherical geometry receives relatively less attention. Yaeger et al. [19] first achieved planetary atmospheric flows on the surface of a sphere using a grid-based discretization method. Hill and Henderson [3] proposed an algorithm framework to avoid the singularity problems at poles using grids constructed in spherical coordinates. Later, Yang et al. [4] improved its results by using a staggered grid and a Fourier-transform-based Poisson equation solver. In [5], a novel local advection step was further proposed for reducing artifacts appearing near the pole for flows on spherical surfaces. These works consider the fluid to be on an infinitely thin spherical shell, which is a 2D manifold in 3D space. Recently, Cui et al. [12] successfully simulated incompressible fluid in three-dimensional spherical/cylindrical geometries. Following [20] and [21], they established principal basis functions and enrich functions which support FFT-based reconstruction over the radial domain to simulate divergence-free fluid flows in spheres, spheroids, and cylinders. They achieved great success in simulating fluid flow in three-dimensional spherical geometry using spherical coordinate, but the method lacked flexible control over flow motion and only used a simple propelling-force-based strategy for arbitrary obstacle boundaries in the simulation domain. In our work, we take into consideration of the effect of arbitrary terrain shapes in the three-dimensional space within pressure solvers. The proposed method is a 3D spherical-space fluid simulator with straightforward implementation and control flexibility.

In meteorological science, the three-dimensional atmospheric motion is conventionally separated into two parts consisting of horizontal motions and vertical motions. For the horizontal motion, spectral methods [22] which use a number of orthogonal basis functions are commonly used to solve the partial differential equations, as is described in section 2.3 of [1]. Silberman et al. [23] investigated to use spherical harmonics functions for representing the stream functions and demonstrated a procedure to solve it. Baer et al. [24] and Ellsaeser et al. [25] utilized a similar procedure as in [23] to solve the vorticity equation. Robert et al. [26] proposed a semi-implicit time integration algorithm for spectral method solvers. Bourke et al. [27] and Hoskins et al. [28] proposed the multi-layer primitive equations represented by spherical harmonics, which were applied in the general circulation model [1]. They use structured grids to discretize the spherical space. Layer-based model is widely accepted in meteorological science. There are several vertical coordinate systems to discretize the vertical direction. Height which discretizes vertical direction based on height is the most intuitive way in meteorological science, however, less useful in simplifying the computation of vertical equations. Phillips et al. [29] proposed a σ-coordinate which σ is calculated by pressure. Suarez et al. [30] modified the σ-coordinate on the basis of [29]. Hsu et al. [31] proposed to use θ-coordinate which is based on isentropic vertical coordinate. In the computer graphics community, height coordinate for vertical dimension is more intuitive and suitable for visual effects.

On the other hand, triangle meshes are also used to simulate surface flow on spherical geometry in compute graphics. Ringler [10] used uniform triangle meshes to fully fill the sphere, which was able to avoid the singularity problem occurring at the North pole and the South pole. Elcott [7] used discrete exterior calculus(is known as DEC, see [32]) to solve vortex equations on a two-dimensional surface and recovered fluid flows on the sphere, which avoided the singularity problem at poles. Azencot [6] [33] also proposed meshes-based methods using functional map for triangular discretized surfaces. All of these works are on surface flow lacking considerations of vertical motions. For thin-film structures that have significantly small vertical scale than horizontal scale, Ishida et al. [9] introduced models for soap films, Ringler et al. [10] for airflow on the earth, and Oron et al. [11] for thin-film equation. The main difference between us and the above work is that we consider the thickness size that is comparable to the horizontal scale.

3 Methodology

In this section, we first briefly recap the Navier-Stokes equations in 3-dimensional spherical coordinate in section 3.1. Then we describe our spatial discretization scheme using a layer-based structure in section 3.2. Finally, a geometry-aware projection step is derived from the proposed discretization in section 3.4.

3.1 Equations in 3D Spherical Coordinate

The N-S equations for incompressible fluids in the Cartesian coordinate are below, bold letters denote vectors and no bold capital letters denote scalars:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla P + \mathbf{f}, \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{2}$$

where $\mathbf{u}$ is fluid velocity in Cartesian coordinate $\mathbf{u}(u_x, u_y, u_z)$, ρ is fluid density, P is pressure and $\mathbf{f}$ is body force. According to Appendix 2 in [34], some operators

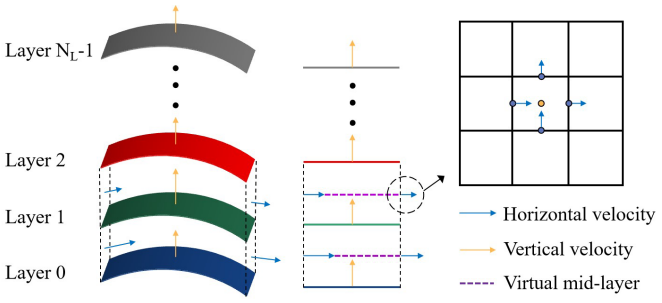


Fig. 1: An illustration of our layer-by-layer staggered grid.

in the spherical coordinate are listed below. The material derivative is

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u_r}\frac{\partial}{\partial r} + \frac{\mathbf{u}_\theta}{r}\frac{\partial}{\partial \theta} + \frac{\mathbf{u}_\phi}{r\ sin\theta}\frac{\partial}{\partial \phi}. \tag{3}$$

The component-wise equations for the momentum equation can be written as follows:

$$\frac{\partial \mathbf{u_r}}{\partial t} + \mathbf{u_r}\frac{\partial \mathbf{u_r}}{\partial r} - \frac{{\mathbf{u}_\theta}^2 + {\mathbf{u}_\phi}^2}{r} = -\frac{1}{\rho}\frac{\partial P}{\partial r} + \mathbf{f_r}, \tag{4}$$

$$\frac{\partial \mathbf{u}_\theta}{\partial t} + \frac{\mathbf{u}_\theta}{r}\frac{\partial \mathbf{u}_\theta}{\partial \theta} + \frac{\mathbf{u}_\theta \mathbf{u_r}}{r} - \frac{{\mathbf{u}_\phi}^2 cot\theta}{r} = -\frac{1}{\rho r}\frac{\partial P}{\partial \theta} + \mathbf{f}_\theta, \tag{5}$$

$$\begin{aligned}\frac{\partial \mathbf{u}_\phi}{\partial t} + \frac{\mathbf{u}_\phi}{r\ sin\theta}\frac{\partial \mathbf{u}_\phi}{\partial \phi} + \frac{\mathbf{u}_\phi \mathbf{u_r}}{r} - \frac{\mathbf{u}_\theta \mathbf{u}_\phi cot\theta}{r} \\ = -\frac{1}{\rho r \sin\theta}\frac{\partial P}{\partial \phi} + \mathbf{f}_\phi.\end{aligned} \tag{6}$$

The elements $({\mathbf{u}_\theta}^2 + {\mathbf{u}_\phi}^2)/r$, $(\mathbf{u}_\theta \mathbf{u_r})/r$, $({\mathbf{u}_\phi}^2 cot\theta)/r$, $(\mathbf{u}_\phi \mathbf{u_r})/r$ and $(\mathbf{u}_\theta \mathbf{u}_\phi cot\theta)/r$ are so-called geometric terms representing the rate of change of coordinate basis. These terms have been studied in previous graphics research with handy calculations on 2D spherical surfaces (see [3], [5]).

The formulation of gradient in spherical coordinate of a physical quantity Q can be written as:

$$\nabla_S Q = \frac{\partial Q}{\partial r}\vec{\mathbf{r}} + \frac{1}{r}\frac{\partial Q}{\partial \theta}\vec{\theta} + \frac{1}{r\ \sin\theta}\frac{\partial Q}{\partial \phi}\vec{\phi}, \tag{7}$$

and the divergence of a vector $\mathbf{u}$ can be written as:

$$\nabla_S \cdot \mathbf{u} = \frac{1}{r^2}\frac{\partial}{\partial r}(r^2\mathbf{u}) + \frac{1}{r\ \sin\theta}\frac{\partial}{\partial \theta}(\sin\theta \mathbf{u}) + \frac{1}{r\ \sin\theta}\frac{\partial \mathbf{u}}{\partial \phi}. \tag{8}$$

The subscript S indicates that the operator is in the spherical coordinate.

3.2 Layered discretization

In our approach, we discretize the three-dimensional simulation domain into a layer-by-layer structure. We choose such a discretization for several reasons. First, it follows the intuition shown by the meteorological science that the planetary general circulation is characterized by the altitudes, and directly extends previous advances in 2D-spherical-coordinate studies, making use of the existing implementation framework. Second, a layered discretization will bring convenience to the 3D advection and projection calculation described in section 3.3 and section 3.4. Third, using the proposed discretization scheme, one can conveniently and precisely control radial-tangential flow motions at given heights using our layered structure.

In Fig. 1, we show our discretization scheme. We assume the atmosphere starts from a minimum radial value r_{min} and ends at a maximum radial value r_{max}, forming a finite-thickness shell that is the simulation domain. Spherical "horizontal layer"s are placed with equal distance Δh, radically dividing the domain into multiple layers. We denote N_θ as the number of discrete grids in the θ direction and N_ϕ as the number of discrete grids in the ϕ direction within each layer, and denote N_L as the number of layers. Then letting $N_\phi = 2 \times N_\theta$ achieves $\Delta\theta = \Delta\phi = (\pi/N_\theta)$ in radian.

In the 3D layered structure, we separate the vector fields into horizontal and radial components. For example, we separate the 3d velocity into a horizontal velocity $\mathbf{u}_h$ within each horizontal layer consisting of $\mathbf{u}_\phi$ along $\vec{\phi}$ direction and $\mathbf{u}_\theta$ along θ direction, and a vertical velocity $\mathbf{u_r}$ along the $\mathbf{r}$ direction. We stored the velocity in a staggered grid form. $\mathbf{u_h}$ is at the middle of two layers, located in staggered grid form in $\phi - \theta$ direction, $\mathbf{u_r}$ is at the center of the grid in each layer, this spherical MAC grid is illustrated in Fig. 1. We summarized the location of each velocity component as follows:

$$\begin{aligned}Location(\mathbf{u_{r_{i,j,k}}}) &= ((i+0.5)\Delta\phi, \quad (j+0.5)\Delta\theta, \quad k\Delta r), \\ Location(\mathbf{u}_{\phi_{\mathbf{i,j,k}}}) &= (i\Delta\phi, \quad (j+0.5)\Delta\theta, \quad (k+0.5)\Delta r), \\ Location(\mathbf{u}_{\theta_{\mathbf{i,j,k}}}) &= ((i+0.5)\Delta\phi, \quad j\Delta\theta, \quad (k+0.5)\Delta r).\end{aligned} \tag{9}$$

3.3 Advection Calculation

Using the layered grid described above, we can separate the velocity advection step into two parts: a horizontal velocity advection and a vertical velocity advection. First, the horizontal velocity advection is performed within each $\phi - \theta$ layer. Inspired by the geodesic method of [5] and the method of [3], we proposed a novel advection scheme for our 3D layer-based discretization. We first perform a first-order semi-Lagrangian backward tracking along the great circle using u_ϕ, u_θ similar to Huang et.al [5] within a layer, finding the horizontal position. More advection schemes such as RK2 or RK3 advection can be applied for higher order accuracy, which has been proven in [5]. Next, a first-order semi-Lagrangian-like advection is applied in the vertical direction, a radial shift distance $\Delta r = \mathbf{u_r}\Delta t$ is calculated according to local vertical velocity $\mathbf{u_r}$. This shift distance is then added to the above horizontal position forming a 3D position p^*, and a linear interpolated value $\mathbf{u}^* = (\mathbf{u}_\mathbf{r}^*, \mathbf{u}_\theta^*, \mathbf{u}_\phi^*)$ can be obtained at p^*. The above procedure is demonstrated in Fig. 2.

As discussed by Hill and Henderson [3], there are external geometric terms that need to be calculated. In the 2D situation, the advection scheme of Huang et.al [5] ensures $\cot\theta$ is zero and therefore term $(\mathbf{u}_\phi^2)\cot\theta/r$ and term $(\mathbf{u}_\theta \mathbf{u}_\phi \cot\theta)/r$ can be ignored. Then, we use a similar method to [3] to solve the remaining geometric term. For calculation convenience, we ignore the term $({\mathbf{u}_\phi}^2 + {\mathbf{u}_\theta}^2)/r$ for $\mathbf{u_r}$ calculation. This will bring much simpler calculations for the remaining two terms. Such a trade-off between calculation error and simplicity is acceptable because the radius scale is much bigger than the velocity scale in practice.

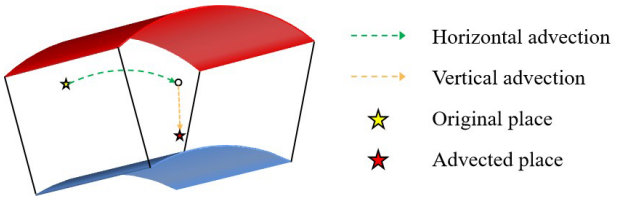


Fig. 2: Advection scheme.

Moreover, solving this nonlinear problem requires iterative methods such as Newton's method which is time consuming, and there will be interpolation problems, because we stored the three components in different locations. In our experiments, it doesn't bring additional artifacts. Therefore the remaining two terms can be solved through the following equations:

$$\frac{\mathbf{u}_\theta^{**} - \mathbf{u}_\theta^{*}}{\Delta t} = \frac{\mathbf{u}_\theta^{**}\mathbf{u}_\mathbf{r}^{*}}{r}, \quad \frac{\mathbf{u}_\phi^{**} - \mathbf{u}_\phi^{*}}{\Delta t} = \frac{\mathbf{u}_\phi^{**}\mathbf{u}_\mathbf{r}^{*}}{r}. \tag{10}$$

The final velocity components given by the advection step are as follows, and $G = \frac{\Delta t}{r}$:

$$\mathbf{u}_\mathbf{r}^{**} = \mathbf{u}_\mathbf{r}^{*}, \quad \mathbf{u}_\theta^{**} = \frac{\mathbf{u}_\theta^{*}}{1 - G\mathbf{u}_\mathbf{r}^{*}}, \quad \mathbf{u}_\phi^{**} = \frac{\mathbf{u}_\phi^{*}}{1 - G\mathbf{u}_\mathbf{r}^{*}}. \tag{11}$$

3.4 Projection Calculation

In the three-dimensional spherical coordinate, the projection equations for the velocity components are as follows:

$$\mathbf{u}_\mathbf{r}^{\mathbf{n+1}} = \mathbf{u}_\mathbf{r}^{**} - \frac{\Delta t}{\rho}\nabla_S P \cdot \vec{\mathbf{r}}, \tag{12}$$

$$\mathbf{u}_\theta^{\mathbf{n+1}} = \mathbf{u}_\theta^{**} - \frac{\Delta t}{\rho r}\nabla_S P \cdot \vec{\theta}, \tag{13}$$

$$\mathbf{u}_\phi^{\mathbf{n+1}} = \mathbf{u}_\phi^{**} - \frac{\Delta t}{\rho r \sin\theta}\nabla_S P \cdot \vec{\phi}, \tag{14}$$

where P is pressure which is the solution of a Poisson equation 15 using spherical gradient and divergence, $\mathbf{u}^{**}$ is velocity after advection step:

$$\nabla_S^2 P = \frac{\rho}{\Delta t}\nabla_S \cdot \mathbf{u}^{**}. \tag{15}$$

Solving a three-dimensional Poisson equation in a sphere is a difficult problem and often needs spectral methods [22], to avoid the singularity problem which occurs at two poles ($\theta = 0$, $\theta = \pi$) and the center (r = 0). Recently, a direct Poisson solver in spherical coordinate has been proposed in the field of computational physics [35], which provides a theoretical foundation for our problem. However, their method relies on a spherical harmonics expansion and faces the dilemma of having high computational cost using more base functions or having high calculation errors for horizontal components using fewer base functions. Moreover, Lin's work [35] does not consider terrain-like boundary conditions but only considers the topmost and bottom boundary conditions.

Inspired by their method, we propose a layer-based solution for the projection calculation that has moderate calculation cost, is visually plausible, and is able to cope with arbitrary terrain boundary shapes. To feasibly develop an efficient algorithm framework, we split the projection step into two steps, the first step is a coarse 3D projection which will be discussed in section 3.4.2, the second is a 2D projection step which will be discussed in section 3.4.3. Before introducing those projection steps, we will first introduce an enhancement to the spherical harmonics expansion where we use a volume ratio not occupied by terrain to enable geometry-aware incompressible solve in spherical-coordinate domains in section 3.4.1.

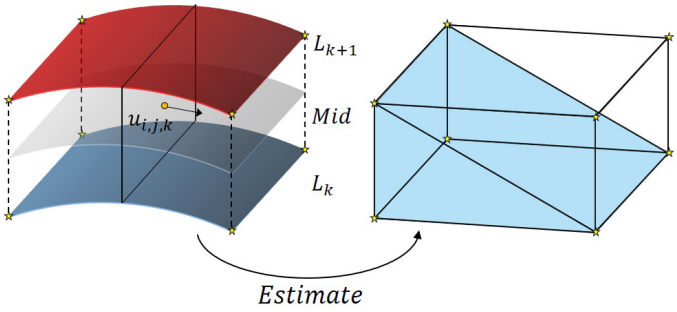


Fig. 3: We calculate the volume ratio not occupied by the terrain in the element composed of eight surrounding points, which are marked with eight stars in the figure. We approximate this space as a cuboid, as shown on the right side of the figure, blue volume denotes the space that is not occupied by terrain. We will estimate the volume ratio of fluid in this cuboid, as will be discussed in section 3.4.1

3.4.1 Boundary representation

In the beginning, we import the terrain as triangle meshes. To represent the boundary of the imported terrain in our simulator, we estimate the volume ratio not occupied by the terrain in the surrounding space of each velocity component, $\mathbf{u}_\mathbf{r}, \mathbf{u}_\phi, \mathbf{u}_\theta$. As we showed in Fig.3, velocity $u_{i,j,k}$ is stored at the middle layer between layer k and layer k+1 in a staggered grid form, as described in section 3.2, thus the surrounding space of this velocity component is composed of eight point which are marked with star in Fig.3 in our 3D simulator. The volume ratio not occupied by the terrain in this surrounding space can be estimated by the signed distance field(hereinafter referred to as SDF) relative to the terrain mesh, we will describe the detailed implementation in section 4.2.

.

Since we assume that the shape does not change and does not move, this step only needs to be calculated in the initialize step. By utilizing the volume ratio to represent terrain, our proposed method could handle a variety of complex shapes, in the next section we discuss how this boundary representation allows our simulator to handle complex terrain in spherical coordinate.

3.4.2 Boundary-aware 3D projection

On the basis of a direct Poisson solver proposed by Lin et al [35], we improved it to handle the shape of the terrain. As the first step, we apply our novel 3D Poisson solver to $\mathbf{u}^{**}$.

In the spherical coordinate, the formulation of the Poisson equation is as follows in Eqn.(16). The singularities occur at the north pole($\theta = 0$) and the south pole($\theta = \pi$). In our simulator, the range of r is $r \in [r_{min}, r_{max}]$ and $r_{min} > 0$, therefore no singularity problem at $r = 0$.

$$\nabla_S^2 P = \frac{1}{r^2}\frac{\partial}{\partial r}(r^2\frac{\partial P}{\partial r}) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta\frac{\partial P}{\partial\theta}) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 P}{\partial\phi^2} = F. \tag{16}$$

To satisfy the incompressible condition Eqn. (2), the right hand side of Eqn. (16) is the divergence of velocity after advection $\mathbf{u}^{**}$ in the spherical coordinate form. To reduce the artifacts due to irregular boundary geometry, there are several solutions in computer graphics. Batty [36] treated the fluid-solid coupling problem in a variational framework, which translate the coupling problem to a minimization problem. They utilized the volume of fluid of each velocity component to adjust the Poisson equation to achieve a fine visual effect near the irregular solid boundary. Ng et.al. [37] improved the boundary treatment of Batty et.al. [36] to get a second-order accuracy near the solid boundaries. Inspired by Batty's work [36], we develop a boundary-aware projection scheme for spherical coordinate simulation.

We take a similar approach to handle fluid-solid coupling. We start from the fluid-solid coupled projection equation in [36] as is shown in Eqn.(17), D is the gradient finite difference operator, M is the contained volume ratio not occupied by the terrain of each velocity component.

$$D^T M D P = \frac{\rho}{\Delta t} D^T M_F \mathbf{u}^{**}. \tag{17}$$

In spherical coordinates, using the precomputed volume ratio, we rewrite Eqn.(17) into the following form, where $m_r, m_\phi, m_\theta \in m$ is the volume ratio not occupied by terrain of each velocity component as described in section 3.4.1:

$$\begin{aligned}&\frac{\partial}{\partial r}(m_r r^2\frac{\partial P}{\partial r}) + \frac{1}{\sin\theta}\frac{\partial}{\partial\theta}(m_\theta\sin\theta\frac{\partial P}{\partial\theta}) + \frac{1}{\sin^2\theta}\frac{\partial}{\partial\phi}(m_\phi\frac{\partial P}{\partial\phi})\\ &= m_r\frac{\partial}{\partial r}(r^2\frac{\partial P}{\partial r}) + \frac{m_\theta}{\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta\frac{\partial P}{\partial\theta}) + \frac{m_\phi}{\sin^2\theta}\frac{\partial^2 P}{\partial\phi^2}\\ &+ r^2\nabla_S m\cdot\nabla_S P\\ &= r^2\frac{\rho}{\Delta t}\nabla_S\cdot(m\mathbf{u}^{**}).\end{aligned} \tag{18}$$

According to m, we divide the terrain geometry into three parts: (1) solid part where $m = 0$. (2) fluid part where $m = 1$. (3) fluid-solid boundary part where $m \in (0, 1)$. In fluid and solid part, m is a constant value (0 or 1), therefore $\nabla_S m$ equals zero, which means the term $\nabla_S m \cdot \nabla_S P$ in Eqn.(18) can be ignored. In the fluid-solid boundary part, which is a narrow-band width area at the surface of the terrain, the gradient of $\nabla_S m$ doesn't equal to zero. Fig.4 illustrates the gradient of m in the fluid-solid boundary part. It is intuitive that the non-zero gradient of m appears at the surface of the terrain, pointing from a smaller m to a bigger m in the normal direction of the terrain's surface. Using the free-slip boundary condition, the velocity near the boundary should be zero in the normal direction of terrain and should be along the surface of the terrain, therefore we

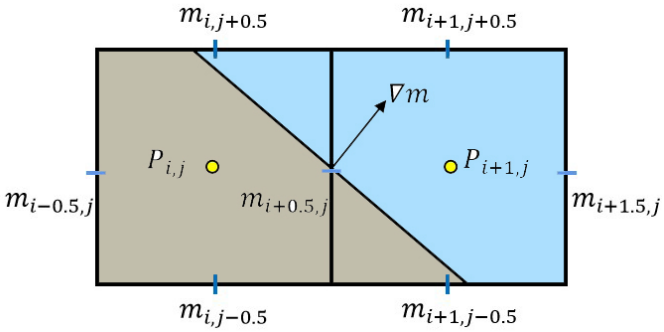


Fig. 4: Illustration of the gradient of m in simple 2D case.

have $\nabla_S m \cdot \mathbf{u^{n+1}} = 0$ on the surface of terrain. Notice that $(\mathbf{u^{n+1}} - \mathbf{u}^{**})/\Delta t = -\nabla P/\rho$, therefore we could derive the following formulation:

$$\begin{aligned}\nabla_S m\cdot\nabla_S P &= -\frac{\rho}{\Delta t}\nabla_S m\cdot(\mathbf{u^{n+1}} - \mathbf{u}^{**})\\ &= -\frac{\rho}{\Delta t}(\nabla_S m\cdot\mathbf{u^{n+1}} - \nabla_S m\cdot\mathbf{u}^{**})\\ &= \frac{\rho}{\Delta t}\nabla_S m\cdot\mathbf{u}^{**}\end{aligned} \tag{19}$$

We substitute Eqn.(19) into Eqn.(18) and rewrite it as follows:

$$\begin{aligned}&\frac{m_r}{r^2}\frac{\partial}{\partial r}(r^2\frac{\partial P}{\partial r}) + \frac{m_\theta}{r^2\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta\frac{\partial P}{\partial\theta}) + \frac{m_\phi}{r^2\sin^2\theta}\frac{\partial^2 P}{\partial\phi^2} \approx\\ &\frac{m^*}{r^2}\frac{\partial}{\partial r}(r^2\frac{\partial P}{\partial r}) + \frac{m^*}{r^2\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta\frac{\partial P}{\partial\theta}) + \frac{m^*}{r^2\sin^2\theta}\frac{\partial^2 P}{\partial\phi^2} =\\ &m^*\nabla_S^2 P = \frac{\rho}{\Delta t}\nabla_S\cdot(m\mathbf{u}^{**}) - \frac{\rho}{\Delta t}\nabla_S m\cdot\mathbf{u}^{**}.\end{aligned} \tag{20}$$

We estimate the value m^* by means of its surrounding volume ratio not occupied by terrain, in 3D case $m^* = (m_{r1} + m_{r_2} + m_{\phi 1} + m_{\phi 2} + m_{\theta 1} + m_{\theta 2})/6$. Dividing both sides of Eqn.(20) by m^* yields a regular Poisson equation and can be solved numerically in our discretization.

We follow the idea of Lin [35] to solved the Poisson equation. We can expand the solution of Eqn. (16) by spherical harmonics expansion [38] as follows:

$$\begin{aligned}&P(r,\theta,\phi) =\\ &\sum_{n=0}^{N}(\sum_{m=0}^{n}(P_m^n(\cos\theta))(p_{mn}(r)\cos m\phi + \tilde{p}_{mn}(r)\sin m\phi)),\end{aligned} \tag{21}$$

where P_m^n is the associated Legendre polynomial of order m and degree n, $p_{mn}(r)$ and $\tilde{p}_{mn}(r)$ are the real and imaginary parts of the coefficients of $P_m^n(r)$, N is the highest order of P_m^n in spherical harmonics expansion.

We denote the right hand side of Eqn.(20) as $F(r, \phi, \theta)$, it has a similar spherical harmonics expansion with $f_{mn}, \tilde{f}_{mn}$

as its coefficients, which is given by:

$$f_{mn}(r) = M\int_0^{2\pi}\int_0^{\pi} F(r,\phi,\theta)P_m^n(\cos\theta)\cos m\phi\sin\theta d\theta d\phi,$$
$$\tilde{f}_{mn}(r) = M\int_0^{2\pi}\int_0^{\pi} F(r,\phi,\theta)P_m^n(\cos\theta)\sin m\phi\sin\theta d\theta d\phi,$$
$$M = \frac{(2n+1)(n-m)!}{2(n+m)!N(m)}, \tag{22}$$

where N(0) = 2π and N(m) = π for $m \neq 0$. As the spherical harmonics functions are eigenfunctions of Laplacian operator in spherical coordinate, i.e., $\nabla_S^2 P_m^n = -n(n+1)P_m^n$, substitute Eqn.(21) into Eqn.(16), the formulation will reduced to a one dimension ordinary ODE equation, written in finite differential form as below:

$$(\frac{r_{i-0.5}^2}{\Delta r^2})p_{mn}^{i-1} + \left(n(n+1) - \frac{r_{i+0.5}^2 + r_{i-0.5}^2}{\Delta r^2}\right)p_{mn}^i + (\frac{r_{i+0.5}^2}{\Delta r^2})p_{mn}^{i+1} = r_i^2 f_{mn}^i, \tag{23}$$

where p_{mn}^i is the coefficient of P_n^m in layer i, $r_i = r_{min} + (i+0.5)\Delta r$, $i = 0, \cdots, N_r - 1$. Eqn. (23) is a tridiagonal linear system with size $N_L \times N_L$, which can be conveniently solved in serial or parallel computing as described in [39]. After getting all of the coefficients p_{mn} and $\tilde{p}_{mn}$, $P(r,\phi,\theta)$ is reconstructed by Eqn. (21). In order to perform efficiently, we pre-compute the entries of this tridiagonal matrix before starting the simulation, as described in section 4.

p_{mn}^{-1} and $p_{mn}^{N_r}$ are the ghost value. There are Neumann boundary conditions at the topmost and bottom layers for the ghost values in order that fluid wouldn't pass through the ground and is contained near the surface: $u_{r_0}^{n+1} = 0$ and $u_{r_N}^{n+1} = 0$. Substituting these boundary conditions to Eqn.(12), we get:

$$\frac{P(\phi,\theta,0) - P(\phi,\theta,-1)}{\Delta r} = \frac{\rho}{\Delta t}\mathbf{u}_\mathbf{r}^{**}(\phi,\theta,0)$$
$$\frac{P(\phi,\theta,N_r) - P(\phi,\theta,N_{r-1})}{\Delta r} = \frac{\rho}{\Delta t}\mathbf{u}_\mathbf{r}^{**}(\phi,\theta,N_r). \tag{24}$$

As discussed in [35], the above Neumann boundary conditions is equivalent to $(p_{mn}^0 - p_{mn}^{-1})/\Delta r = u_{mn}^0$ and $(p_{mn}^{N_r} - p_{mn}^{N_r-1})/\Delta r = u_{mn}^{N_r}$, where u_{mn}^0 and $u_{mn}^{N_r}$ are the coefficient of associated Legendre polynomials in spherical harmonics expansion of $\frac{\rho}{\Delta t}u_r^{**}$ at bottom and topmost layer.

We observe that in our layered grid structure, each layer can possibly exchange volume with other space through its two layer-layer boundaries above/below it. From incompressibility we know the total flux between each layer is zero. On the other hand, the layer at r_{min} exchanges volume only through the upper boundary, so one can conclude that the total flux across every layer-layer boundary should be zero to satisfy the incompressible condition. Thus, we first extract the radial component of velocities solved from the 3D solver as described in section 3.4.2, which corresponds to the 3D projection step in Algorithm 1. Although such a direct result has an error (will be explained in section 3.6), it serves as a good estimation of the radial velocity. Then a re-balance step is introduced to ensure their flux sum exactly equals to zero. We will describe the detailed re-balance step in section 3.5.

3.4.3 Boundary-aware 2D projection

In practice, the above solver requires a large number of spherical harmonics base functions making it inefficient for graphics applications , on which we will provide a further discussion in section 3.6. In our experiments, using a small number of base functions generates divergence especially near the polar areas, resulting in obvious visual artifacts. As a result, the outcome of the above solver can not be used directly with moderate computational cost. For computational efficiency, we in turn move on to another 2d solver step described as follows.

We rewrite the divergence-free form of Eqn. (8):

$$\frac{1}{r\ \sin\theta}\frac{\partial}{\partial\theta}(\sin\theta\ \mathbf{u}_\theta) + \frac{1}{r\ \sin\theta}\frac{\partial\mathbf{u}_\phi}{\partial\phi} = -\frac{1}{r^2}\frac{\partial}{\partial r}(r^2\mathbf{u}_\mathbf{r}). \tag{25}$$

We denote ∇_h as the differential operator within the horizontal layer. Eqn.(25) shows how the vertical velocity influences the horizontal velocity under the incompressible condition. Therefore the 2D projection equation coupled with the 3D projection equation as described in section 3.4.2 becomes:

$$\nabla_H^2 P = \frac{\rho}{\Delta t}\nabla_H\cdot\mathbf{u}_\mathbf{h}^{**} + \frac{\rho}{r^2\Delta t}\frac{\partial}{\partial r}(r^2\bar{\mathbf{u}}_\mathbf{r}), \tag{26}$$

where $\bar{\mathbf{u}}_\mathbf{r}$ is the vertical velocity after the re-balancing step, which will be discussed in section 3.5. .We adjust the right hand side of Eqn.(26) to improve the 2D projection step in the same way as in the 3D projection step. Therefore Eqn.(26) will become as follows, where $m_H^{**} = (m_{\theta 1} + m_{\theta 2} + m_{\phi 1} + m_{\phi 2})/4$ is the mean of surrounding volume ratio not occupied by terrain:

$$m_H^{**}\nabla_H^2 P = \frac{\rho}{\Delta t}\left(\frac{\partial(m_\theta\sin\theta\mathbf{u})}{r\sin\theta\partial\theta} + \frac{\partial(m_\phi\mathbf{u})}{r\sin\theta\partial\phi} + \frac{\partial(r^2 m_r\bar{\mathbf{u}}_\mathbf{r})}{r^2\partial r} - \nabla_H m\mathbf{u}\right). \tag{27}$$

Then, a 2D Poisson solver is performed within each layer, and we update the horizontal velocity by Eqn. (14) and Eqn. (13) with solved pressure, which is the solutions of Eqn.(26).

We use the Fast Fourier Transform(FFT) based method [40] to solve the above Poisson equation as the same as [4]. Since each layer computes this step individually, Eqn. (26) should be solved $N_L - 1$ times, the same as the number of layers.

It is to be noted that, although in practice we only use the vertical component of velocities solved from the 3D Poisson solver, we can not discard the 3D projection step. This is because individual 2d projection solvers will never know when and where there should be a vertical circumflux without the calculation of the 3d projection solver.

3.5 Re-balancing Vertical motions

As mentioned above, the total flux between layers should be zero, which is not automatically assured by the 3D projection due to its numerical errors using a limited number of base functions. So we introduce a re-balancing step for vertical velocity to ensure incompressibility within the fluid layers by enforcing the inter-layer zero-flux boundary condition.

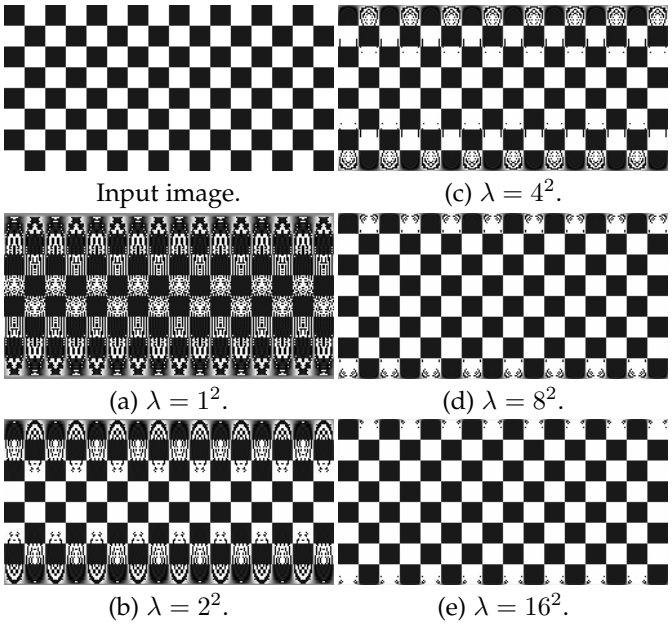


Fig. 5: Illustration of spherical harmonics expansion to two different input map with different sample times per grid, the darker the pixel color, the smaller the relative error.

We first compute the flux of each layer. Notice that the grid cell area is uneven, which is determined by ϕ, θ, and radius. We use the following equation to compute the grid cell area first:

$$s_{i,j,k} = (R + k\Delta r)^2(\sin\theta_j - \sin\theta_{j-1})\Delta\phi. \tag{28}$$

For each layer, we compute the fluxes by summing up the product of vertical velocity and corresponding grid cell area. Finally, we adjust the speed of different grids to make sure the flux of this layer is zero. For example, if the flux of Layer k is F_k, we adjust a specific velocity velocity $u_{r_{i,j,k}}$ at grid(i,j,k):

$$\mathbf{u}^{\mathbf{after}}_{\mathbf{r_{i,j,k}}} = \mathbf{u}^{\mathbf{before}}_{\mathbf{r_{i,j,k}}} - \frac{F_k}{S_k}, \tag{29}$$

where S_k is Layer k's surface area which is calculated using the formula $S = 4\pi r^2$.

3.6 The orders of spherical harmonics expansion

As we mentioned in section 3.4.2, spherical harmonics expansion is utilized to numerically solve the Poisson equation in 3D spherical coordinate. To get a more accurate solution requires higher order of Legendre polynomials in spherical harmonics expansion, as shown in Eqn.(21). However, for higher order spherical harmonic expansion, due to the lack of fast algorithms like FFT for Fourier transform, it is very time consuming. We will demonstrate in this section that the Spherical harmonic expansion scheme alone does not provide a good balance between efficiency and precision. In this section, we will discuss how many Legendre polynomials for operating more accurate spherical harmonics expansion in our proposed 3D projection step are needed.

We discretize each staggered grid layer with N_ϕ grids in ϕ direction and N_θ grids in θ direction, $N_\phi = 2 \times N_\theta$. As the highest order of associated Legendre polynomials from a discretized $N_\phi \times N_\theta$ input in spherical harmonics expansion l_{max} is up to $N_\theta - 1$, in order to expand with higher order coefficients, we need to upsample the original $N_\phi \times N_\theta$ input to a larger size, for example, $2N_\phi \times 2N_\theta$, then the highest order of associated Legendre polynomials will become $2N_\theta - 1$. We divide each grid of the original input into multiple sub-grids. We denote λ as the number of subdivision, which means we get $\sqrt{\lambda}N_\phi \times \sqrt{\lambda}N_\phi$ grids after upsampling, and l_{max} becomes $\sqrt{\lambda}N_\theta - 1$. After inverse spherical harmonics expansion, we should reduce the upsampling size to the original size, which can be done by averaging the corresponding λ grids values and put it on each grid of the original $N_\phi \times N_\theta$ size.

For error analysis, we use a 128×256 grayscale lattice map as input, the grayscale of each pixel corresponds to the data value on the grid at the same location. We use different λ to expand the same input and then inverse transform to an original size output, as shown in Fig.5. The upper left corner of Fig.5 is the grayscale lattice map input, and the rest of the figures are the output with different λ, the more similar it is to the input, the stronger its expressive ability. As we can see, with λ increases(the more associated Legendre polynomials are used), the recovered output is closer to the original input. As the λ value increases, the errors of the spherical harmonics expansion is reduced from the equator region to the pole region. However, even with $\lambda = 16^2$, which is already of high computational cost, the errors are still not negligible.

Table.1 lists the running time of different λ cases. It can be seen that the time required for computation is not linearly related to l_{max}. We then set $\lambda = 4^2$ in our combined 3D-2D projection setup, which has a fast and relatively good estimation from the 3D solver and achieves a precise divergence-free field through the 2D solver.

4 Implementation

Our overall algorithm framework is shown in Algorithm 1. We first determine parameters including N_ϕ, N_θ, N_L, etc. In each timestep, we first perform the advection step which is described in section 3.3. After the advection step, we apply external force to the velocity by the following equation:

$$\mathbf{u}^{**} = \mathbf{u}^{*} + \Delta t\mathcal{F}, \tag{30}$$

where u^* is the output of the advection step, $\mathcal{F}$ is the acceleration caused by force term. Afterwards, we execute the projection steps to achieve divergence-free conditions. We first execute 3D projection using spherical harmonics expansion as described in the section 3.4.2. Then a Re-balance step is applied to ensure each layer's inlet flux is in equilibrium with outlet flux. After a Re-balancing step, we execute 2D projection. In the implementation, we calculate the required spherical harmonics on the CPU side in the preprocessing stage, each step in the main loop of simulation can be implemented in high parallelism on the GPU side.

In the following subsections, we describe some implementation details for our multi-layer simulator.

	grid size	samples/grid	l_{max}	expand time(ms)	integral times(ms)
Fig.5.(a)	128×256	1	127	2	<1
Fig.5.(b)	128×256	4	255	6	3
Fig.5.(c)	128×256	16	511	25	20
Fig.5.(d)	128×256	64	1023	139	154
Fig.5.(e)	128×256	256	2047	869	775

TABLE 1: Comparison of time consumption of spherical harmonic expansion at different grid size and sample count

Algorithm 1 Overall algorithm famework

Parameters:
discretized number along ϕ direction, N_ϕ;
discretized number along θ direction, N_θ;
number of layers, N_L;
radius, R;
timestep, Δt;
gap size between each layers, Δr;
sample times per grid λ

Initialize:
velocities u_ϕ^0, u_θ^0, u_r^0;
m_ϕ,m_θ,m_r represent terrain, compute from the input mesh;
Preparing associate Legendre Polynomials
Calculating entries of tridiagonal matrix
t $\leftarrow$ 0

For t $<$ T **do**:
Advection Scheme(u_ϕ^t, u_θ^t, u_r^t, Δt)
swap velocity buffers
Add external force
swap velocity buffers
3D projection step($u\phi^t, u\theta^t, u_r^t$)
Re-balance step
swap velocity buffers
2D projection step($u\phi^t, u\theta^t$)
swap velocities buffers
t $\leftarrow$ t+ Δ t

End For

4.1 Compute entries of tridiagonal matrix

Eqn.(23) is a tridiagonal linear system in the 3D projection step. Since the entries are consistent in every timestep, we pre-compute the entries before simulation. The format of the tridiagonal matrix is as follows:

$$\begin{vmatrix} b_0 & c_0 & 0 & \cdots & 0 & 0 \\ a_1 & b_1 & c_1 & \cdots & 0 & 0 \\ 0 & a_2 & b_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{N_r-1} & b_{N_r-1} \end{vmatrix}$$

According to Eqn.(23), the value of a_i, b_i and c_i are:

$$a_i = \frac{r_{i-0.5}^2}{\Delta r^2},\ b_i = -\frac{r_{i+0.5}^2 + r_{i-0.5}^2}{\Delta r^2} + n(n+1),\ c_i = \frac{r_{i+0.5}^2}{\Delta r^2}. \tag{31}$$

There are boundary conditions Eqn.(24) at the bottom and topmost layer, therefore we adjust the value of a_0, b_0, b_{N_r-1} and c_{N_r-1} as $a_0 = 0$, $b_0 = (-r_{0.5}^2/\Delta r^2) + n(n+1)$, $b_{N_r-1} = (-r_{N_r-0.5}^2/\Delta r^2) + n(n+1)$, $c_{N_r-1} = 0$:

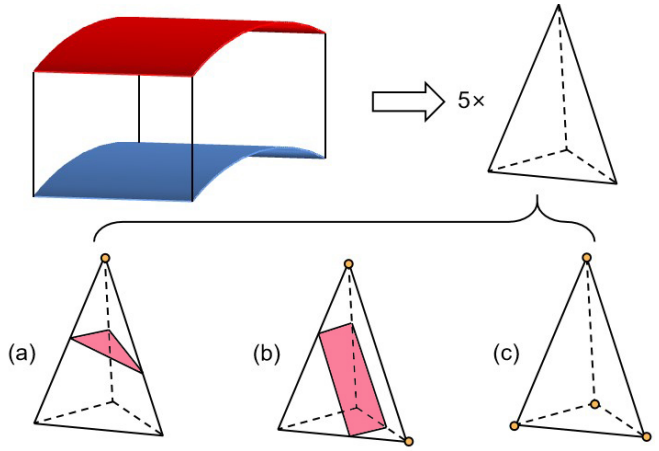


Fig. 6: The volume ratio not occupied by the terrain of each element could be estimated by decomposing it into five tetrahedrons and weighted-averaging their ratio. The volume ratio in a tetrahedron can be calculated via four SDFs of its vertices.

4.2 Estimate volume ratio not occupied by the terrain of each velocity component

As illustrated in Fig.6, the surrounding space of each velocity component can be roughly decomposed into five tetrahedron:Tet_{ABCE}, Tet_{CEFG}, Tet_{ADCG}, Tet_{AGHE} and Tet_{AECG}. We estimate the volume ratio of these five tetrahedrons. As these five tetrahedrons are not the same volume, therefore we perform a weighted average for these five tetrahedrons based on the volume ratio to obtain the corresponding volume ratio not occupied by the terrain of each velocity component. This can be done in the pre-computing step. To estimate the volume ratio not occupied by the terrain of a tetrahedron, we simply utilize its four points' signed distance value relative to the terrain mesh, if a signed distance is negative, it indicates the point is inside the mesh, else if it's positive, the point is outside the mesh, if it's zero, it is on the mesh. We will briefly describe the calculation.

First we denote S_a, S_b, S_c, S_d as the signed distance of each four vertices, and they satisfy the relationship $S_a > S_b > S_c > S_d$. There will be three situations, as shown in Fig.6:

- $S_c > 0 > S_d$ or $S_a > 0 > S_b$, which corresponds to case (a) in Fig.6, the purple plane approximately expresses the surface of terrain and divides a tetrahedron into two parts;
- $S_b > 0 > S_c$, which is corresponding to case (b) in Fig.6;

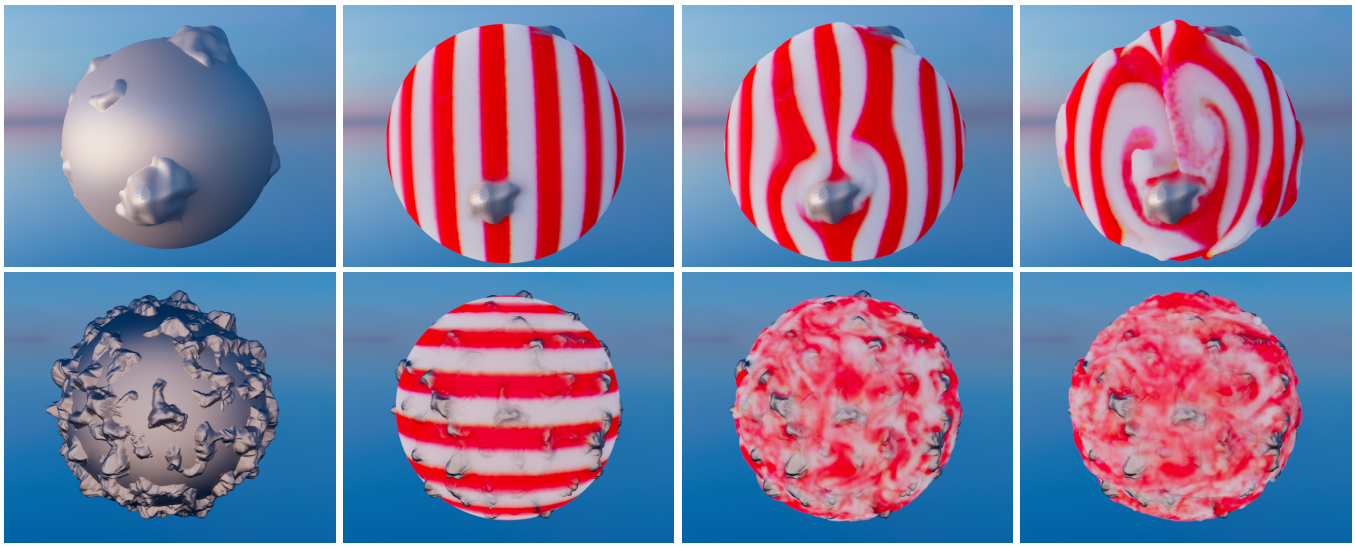


Fig. 7: Fluid flows on customized terrain. Upper row: a small number of big mountains; Bottom row: a large number of small mountains where the terrain shape is more complicated and fractured. Our method successfully recovers the 3D fluid motion in the existence of obstacles.

- $S_d > 0$ or $S_a < 0$, which is corresponding to case (c) in Fig.6, the tetrahedron is fully solid or fully fluid.

In case (a), we assume $S_a > 0 > S_b$, the volume ratio can be computed as follows:

$$m = 1 - \theta_{ad}\theta_{ac}\theta_{ab}. \tag{32}$$

In case (b), the volume ratio can be computed as follows:

$$m = \theta_{db}\theta_{da} + \theta_{cb}\theta_{ca} + \theta_{db}\theta_{ca} - \theta_{db}\theta_{da}\theta_{ca} - \theta_{cb}\theta_{ca}\theta_{db}. \tag{33}$$

where $\theta_{xy} = S_x/(S_x - S_y)$, and x,y $\in$ {a,b,c,d}. The above algorithm can be easily implemented in parallel.

4.3 The thickness of the spherical shell

In theory, our method has no limit on the maximum thickness. Since we fixed the discretization number of each layer, the element size on the outermost layer will be relatively larger as the radius increases, and the accuracy of the volume fraction which represents the geometry on a layer will decrease. Therefore, we recommended that the maximum thickness be within 50% of the radius in practice.

In addition, the gap between layers in the vertical direction should be comparable to the scale of the discrete grid in the $\phi - \theta$ horizontal direction, since we focus on the situation that the motion in the vertical direction is comparable to that in the $\phi - \theta$ direction. When our method degrades to only one layer, it's consistent with the previous 2D surface method [3] [4] [5] and can also be used for thin-film simulations.

5 Experiment Results

In this section, we perform several experiments to demonstrate the effectiveness of our proposed method in simulating fluid flows in a three-dimensional spherical shell domain.

We implement our proposed method on an NVIDIA graphic card GeForce GTX 1080Ti with CPU Intel Core i7-4790K and 32GB RAM. All of our experiments are simulated over a spherical shell domain with a minimum radius of 10.0 length unit. We choose a fixed timestep of 0.002s in all our experiments. All of the submitted animations are 24 fps and each frame contains 21 timesteps of calculations. All of the experiment results are attached in video format in supplemental materials.

5.1 Flow over rough terrain

In this scenario we create a planet with rough terrain and spread a moderately thick layer of red-white striped cloud over it. The upper row of Fig.7 is a simpler input terrain. In the initial step, we blow the cloud in one direction and it will flow through the terrain on the planet. With the projection step, we propose the fluid motion is geometric-aware. When the fluid passes the uneven hill, it flows around its periphery in the horizontal direction and will climb up the slope in the vertical direction. We also create another planet with more complex terrain as input, as illustrated in the bottom row of Fig.7. Our proposed boundary representation can handle such complex obstacles mesh as input. We use curl-noise [41] to initialize a divergence-free velocity. The experiment produces a visually plausible effect.

5.2 Release from a dam

In this scenario, we place a dam-like structure at the circumpolar pole with several exits underneath. Gas is stored in the structure and released through outlets at the bottom, where it interacts with external obstacles, as illustrated in Fig.8. Thanks to our proposed boundary treatment, the gas shows correct behaviors alongside various terrain shapes.

5.3 Flow over complex geometries

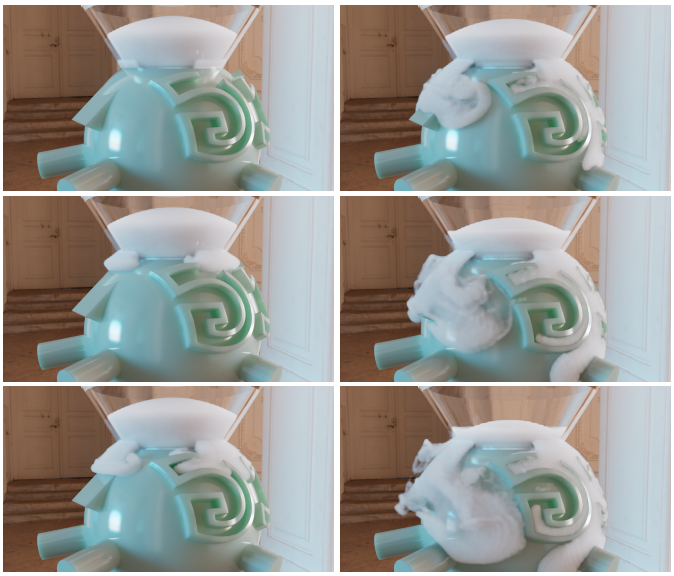


Fig. 8: Smoke released from a dam. We build up a dam-like structure at the circumpolar pole. Gas stored at this structure then releases at the bottom and interacts with exterior obstacles.

	Grid size	N_L	λ	dt(s)	Δr	s/step
Less Mountain	128×256	40	16	0.002	0.10	1.604
More Mountain	128×256	40	16	0.002	0.05	1.426
Dam	128×256	40	16	0.002	0.10	1.873
Armadillo	128×256	40	16	0.002	0.10	1.625
Little cow	128×256	40	16	0.002	0.10	1.654
Control	64×128	40	16	0.002	0.10	0.637
Goddess	128×256	40	16	0.002	0.10	1.508
Compare(2D+3D)	128×256	20	16	0.002	0.10	1.096
Compare(3D)	128×256	20	64	0.002	0.10	5.993

TABLE 2: Information of each examples.

In Fig.10 and Fig.12, we demonstrate our method's ability to handle shapes significantly different from a sphere. The virtual bottom boundary of the simulation domain is shown in Fig.9. Over the armadillo, we place a smoke source at its left hand and initialize a wind blowing from its left hand to its right hand. Vigorous swirls can be observed when the flow encounters the armadillo's head and ears.

In Fig.12 we set the simulation domain around the neck of a little cow. We initialize with a curl noise velocity field superimposed with a velocity along the equator to make the cloud flows around the tiny cow. The result shows rich 3D vortices in this experiment. The above experiment results further validate that our proposed method can handle 3D motions over complex geometry.

Noting that for this type of non-spherical geometry, it is more common to simulate in the Cartesian coordinate system. In this case, we use the Houdini Pyro solver based on [14] to simulate fluid flow in the same scenario of Fig. 10, as shown in Fig. 11. In both coordinates the viscosity coefficient is set to zero, and the gravity force is pointing to the origin. Our method produces more smooth fluid motion and vigorous vortices.

5.4 Controlling

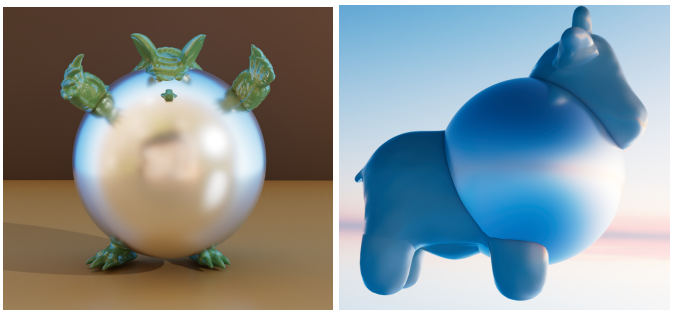


Fig. 9: The virtual bottom boundary of the armadillo case and little cow case.

Our layer-based method can help artists easily control the motion of fluid in the spherical shell space, and the scenario of Fig.13 is an example to validate it. Using our spherical coordinate algorithm, it will be convenient to make the fluid flow exactly at a given height, rise and descend along a given radius, etc., which is hard for Cartesian coordinate simulator to accomplish.

We can use external forces to control the fluid. After solving for the advection term, users could customize a time-dependent external force $f(\phi, \theta, r, t)$ and use forward Euler integration to modify the flows, which is convenient for artistic control. In Fig. 13, we placed customized force in vertical and horizontally at appropriate time to alter the flows to form an interlocking ring structure.

5.5 Compared with high-precision 3D projection solver

To verify the correctness of our proposed 3D-2D solver, in this example, we compare our proposed 3D-2D projection scheme to a pure 3D projection solver from [35]. For the reference pure 3D projection solver, we use a sample count of 8^2 per grid for spherical harmonics expansion. Such a high order of spherical harmonics expansion will inevitably lead to extremely long calculation times. In contrast, the sample count per grid is set to be 4^2 in the 3D-2D projection solver and significantly reducing the computation time. The calculation time for one step is 5.993s for pure 3D projection solver and 1.096s for our 3D-2D projection solver, our algorithm achieves a nearly 6 times speed up. Fig.14 shows the result of two methods. The two solvers show consistent results. It is interesting to note that our 3D-2D projection solver shows better divergence-free property even though we use fewer associate Legendre polynomials in the 3D solver. The results show that our method can ensure the overall divergence-free condition and is effective in our layer-based discretized model.

5.6 An ablation study about the 2D projection

To validate the effectiveness of our proposed 3D-2D solver, we implement an ablation study. Whether to use 2D projection (as described in section 3.4.3) as a control variable, we performed our proposed 3D-2D projection method and a pure 3D projection with the same orders of spherical harmonics on the same terrain, as shown in figure 15. The experiment result shows that our proposed mixture 3D-2D projection method has better divergence-free velocity under the same orders of spherical harmonics as the

Fig. 10: Smoke flow through armadillo. We set a smoke source on Armadillo's left hand, and with the wind speed, the smoke is drifting to its right hand. Smoke will pass through Armadillo's head and ears along the way and generate vortices. This example also uses the spherical shell space as the simulation domain, and the geometry of the armadillo is superimposed on a base sphere as terrain.

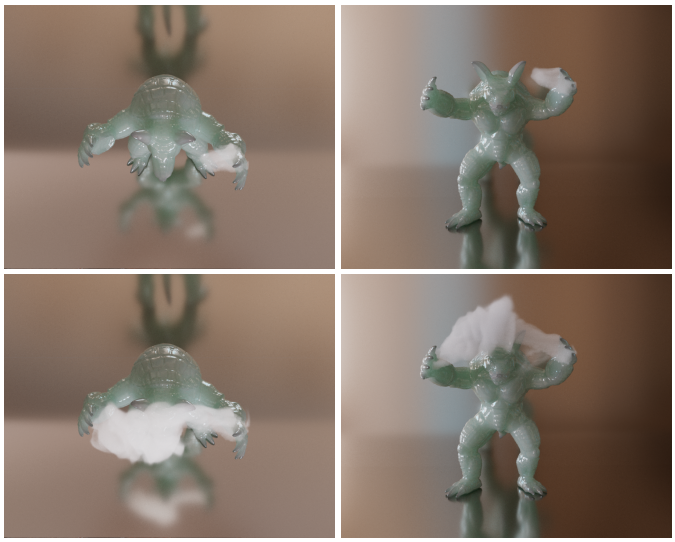


Fig. 11: In the similar scenario with Fig. 10, simulate in the Cartesian coordinate.

Fig. 12: A tiny cow geometry is imported as terrain. The cloud around the cow is initialized by a divergence-free velocity, due to our geometric-aware 3D solver, the cloud could flow around the tiny cow and climb along the surface.

pure 3D projection has. Moreover, our proposed method can produce better vortices without unevenly distributed clouds, compared to the result of the pure 3D projection. This comparison result also validates the effectiveness of our proposed method. While using lower-order spherical harmonics expansion to save computational effort, the extra 2D projection can ensure better divergence-free condition and have better visual effects.

6 Conclusion and Future Works

We have presented a layer-based method for simulating three-dimensional planetary flow in the spherical coordinate. Based on our layer-by-layer structure and a boundary-aware pressure solving scheme, we are able to recover horizontal and vertical flow motions under the existence of arbitrary terrain shapes within a spherical shell of finite thickness. The proposed method is also flexible for artistic control and is easy to implement.

Our method has several limitations. Although the method we implemented is three-dimensional, the main application scenarios are focused on the three-dimensional spherical shell structure without the flows at the center of the sphere, which will be also interesting and is worthy of future study. Our work mainly focuses on the static obstacles and it's worth extending it to handle the moving obstacles in the future. Since the expression accuracy of spherical harmonic expansion near the pole is relatively poor compared with that near the equator, there will be a slight artifact near the pole, which needs to be improved in the future. Although we do not observe artifacts due to the re-balance step, it possibly can cause discontinuity in the simulation. In the advection step, we decomposed it into horizontal and vertical directions, and a full spherical-coordinate-based backtracking scheme is worthy of future exploration. The real-world atmosphere features variable air density along the altitude direction, but we assume the density of the fluid is constant in this work. The current framework does not support simulating phase change and multi-phase flow which are important for cloud, rain, and

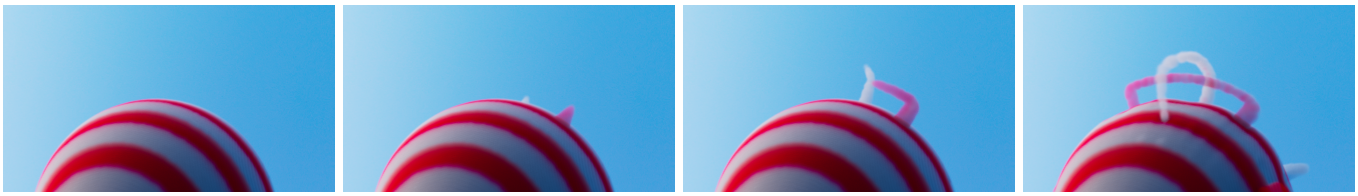


Fig. 13: Our method proposes convenient uplift and downlift control over the spherical domain. In this example, we set up and down force controls at several positions on the sphere, and make it form multiple ring structures.

Fig. 14: Comparison with pure 3D Poisson solver proposed by [35] in projection. The left column is our proposed 3D-2D projection solver and the right column is a pure 3D projection solver. Our method is nearly 6 times faster than the pure 3D solver while effectively ensuring incompressibility.

Fig. 15: An ablation study to validate the effectiveness of the 2D projection in our proposed 3D-2D projection method. The left column is the result of our proposed 3D-2D projection, and the right column is the result of removing the 2D projection. The bottom row of pictures uses particles to track the velocity field and is clearer to observe the differences. The blue boxed area highlights the main difference.

other common phenomena, they will deserve future investigations. Currently, the grid number within each layer is constant, therefore the grid sizes of the upper layers are bigger than that of the bottom layers. An adaptive grid solution could be adopted in the future work.

Acknowledgments

This work is supported by the Natural Science Foundation of China (62272245), the Open Project Program of the State Key Lab of CAD&CG (Grant No. A2207), Zhejiang University, and the Fundamental Research Funds for the Central Universities (Nankai University, No. 63233080). We thank the anonymous reviewers for providing us with professional suggestions.

References

[1] E. Roeckner, G. Bäuml, L. Bonaventura, R. Brokopf, M. Esch, M. Giorgetta, S. Hagemann, I. Kirchner, L. Kornblueh, E. Manzini *et al.*, "The atmospheric general circulation model echam 5. part i: Model description," 2003.

[2] M. Huber, "modeling approach using the ncar ccsm," *Causes and consequences of globally warm climates in the Early Paleogene*, vol. 369, p. 25, 2003.

[3] D. J. Hill and R. D. Henderson, "Efficient fluid simulation on the surface of a sphere," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 2, pp. 1–9, 2016.

[4] B. Yang, W. Corse, J. Lu, J. Wolper, and C.-F. Jiang, "Real-time fluid simulation on the surface of a sphere," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 2, no. 1, pp. 1–17, 2019.

[5] W. Huang, J. Iseringhausen, T. Kneiphof, Z. Qu, C. Jiang, and M. B. Hullin, "Chemomechanical simulation of soap film flow on spherical bubbles," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 41–1, 2020.

[6] O. Azencot, S. Weißmann, M. Ovsjanikov, M. Wardetzky, and M. Ben-Chen, "Functional fluids on surfaces," in *Computer Graphics Forum*, vol. 33, no. 5. Wiley Online Library, 2014, pp. 237–246.

[7] S. Elcott, Y. Tong, E. Kanso, P. Schröder, and M. Desbrun, "Stable, circulation-preserving, simplicial fluids," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 1, pp. 4–es, 2007.

[8] L. Shi and Y. Yu, "Inviscid and incompressible fluid simulation on triangle meshes," *Computer Animation and Virtual Worlds*, vol. 15, no. 3-4, pp. 173–181, 2004.

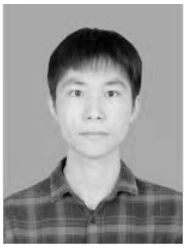
[9] S. Ishida, P. Synak, F. Narita, T. Hachisuka, and C. Wojtan, "A model for soap film dynamics with evolving thickness," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 31–1, 2020.

[10] T. D. Ringler, R. P. Heikes, and D. A. Randall, "Modeling the atmospheric general circulation using a spherical geodesic grid: A new class of dynamical cores," *Monthly Weather Review*, vol. 128, no. 7, pp. 2471–2490, 2000.

[11] A. Oron, S. H. Davis, and S. G. Bankoff, "Long-scale evolution of thin liquid films," *Reviews of modern physics*, vol. 69, no. 3, p. 931, 1997.

[12] Q. Cui, T. Langlois, P. Sen, and T. Kim, "Spiral-spectral fluid simulation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 6,

pp. 1–16, 2021.
[13] U. Vimont, J. Gain, M. Lastic, G. Cordonnier, B. Abiodun, and M.-P. Cani, "Interactive meso-scale simulation of skyscapes," in *Computer Graphics Forum*, vol. 39, no. 2. Wiley Online Library, 2020, pp. 585–596.
[14] J. Stam, "Stable fluids," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999, pp. 121–128.
[15] M. Müller, D. Charypar, and M. H. Gross, "Particle-based fluid simulation for interactive applications." in *Symposium on Computer animation*, vol. 2, 2003.
[16] M. Macklin and M. Müller, "Position based fluids," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 1–12, 2013.
[17] J. U. Brackbill and H. M. Ruppel, "Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions," *Journal of Computational physics*, vol. 65, no. 2, pp. 314–343, 1986.
[18] Y. Zhu and R. Bridson, "Animating sand as a fluid," *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 965–972, 2005.
[19] L. Yaeger, C. Upson, and R. Myers, "Combining physical and visual simulation—creation of the planet jupiter for the film "2010"," *Acm Siggraph Computer Graphics*, vol. 20, no. 4, pp. 85–93, 1986.
[20] Q. Cui, P. Sen, and T. Kim, "Scalable laplacian eigenfluids," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–12, 2018.
[21] T. De Witt, C. Lessig, and E. Fiume, "Fluid simulation using laplacian eigenfunctions," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 1, pp. 1–11, 2012.
[22] K. J. Burns, G. M. Vasil, J. S. Oishi, D. Lecoanet, and B. P. Brown, "Dedalus: A flexible framework for numerical simulations with spectral methods," *Physical Review Research*, vol. 2, no. 2, p. 023068, 2020.
[23] I. Silberman, *Planetary waves in the atmosphere*. New York University, 1953.
[24] F. Baer and G. W. Platzman, "A procedure for numerical integration of the spectral vorticity equation," *Journal of Atmospheric Sciences*, vol. 18, no. 3, pp. 393–401, 1961.
[25] H. W. Ellsaesser, "Evaluation of spectral versus grid methods of hemispheric numerical weather prediction," *Journal of Applied Meteorology and Climatology*, vol. 5, no. 3, pp. 246–262, 1966.
[26] A. Robert, "The integration of a spectral model of the atmosphere by the implicit method," in *Proc. WMO/IUGG Symposium on NWP, Tokyo, Japan Meteorological Agency*, vol. 7, 1969, pp. 19–24.
[27] W. Bourke, "An efficient, one-level, primitive-equation spectral model," *Monthly Weather Review*, vol. 100, no. 9, pp. 683–689, 1972.
[28] B. Hoskins and A. Simmons, "A multi-layer spectral model and the semi-implicit method," *Quarterly Journal of the Royal Meteorological Society*, vol. 101, no. 429, pp. 637–655, 1975.
[29] N. A. Phillips, "A coordinate system having some special advantages for numerical forecasting," *J. Meteorol.*, vol. 14, pp. 184–195, 1957.
[30] M. J. Suarez, A. Arakawa, and D. A. Randall, "The parameterization of the planetary boundary layer in the ucla general circulation model: Formulation and results," *Monthly weather review*, vol. 111, no. 11, pp. 2224–2243, 1983.
[31] Y.-J. G. Hsu and A. Arakawa, "Numerical modeling of the atmosphere with an isentropic vertical coordinate," *Monthly Weather Review*, vol. 118, no. 10, pp. 1933–1959, 1990.
[32] S. Elcott and P. Schröder, "Building your own dec at home," in *ACM SIGGRAPH 2005 Courses*, 2005, pp. 8–es.
[33] O. Azencot, O. Vantzos, and M. Ben-Chen, "Advection-based function matching on surfaces," in *Computer Graphics Forum*, vol. 35, no. 5. Wiley Online Library, 2016, pp. 55–64.
[34] C. K. Batchelor and G. Batchelor, *An introduction to fluid dynamics*. Cambridge university press, 2000.
[35] T.-S. Lin, W.-F. Hu, and C. Misbah, "A direct poisson solver in spherical geometry with an application to diffusiophoretic problems," *Journal of Computational Physics*, vol. 409, p. 109362, 2020.
[36] C. Batty, F. Bertails, and R. Bridson, "A fast variational framework for accurate solid-fluid coupling," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, pp. 100–es, 2007.
[37] Y. T. Ng, C. Min, and F. Gibou, "An efficient fluid–solid coupling algorithm for single-phase flows," *Journal of Computational Physics*, vol. 228, no. 23, pp. 8807–8829, 2009.
[38] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. US Government printing office, 1964, vol. 55.
[39] Y. Zhang, J. Cohen, and J. D. Owens, "Fast tridiagonal solvers on the gpu," *ACM Sigplan Notices*, vol. 45, no. 5, pp. 127–136, 2010.
[40] M.-C. Lai and W.-C. Wang, "Fast direct solvers for poisson equation on 2d polar and spherical geometries," *Numerical Methods for Partial Differential Equations: An International Journal*, vol. 18, no. 1, pp. 56–68, 2002.
[41] R. Bridson, J. Houriham, and M. Nordenstam, "Curl-noise for procedural fluid flow," *ACM Transactions on Graphics (ToG)*, vol. 26, no. 3, pp. 46–es, 2007.



Ruihong Cen is currently a graduate student in the College of Computer Science, Nankai University. He received a bachelor's degree from the School of Mechanical & Automotive Engineering, South China University of Technology. His research interest lies in the physically-based simulation in computer graphics.

Bo Ren received the PhD degree from Tsinghua University in 2015. He is currently an associate professor in the College of Computer Science, Nankai University, Tianjin. His research interests include physically-based simulation, 3D scene reconstruction and analysis.