Vortex Methods for Fluid Simulation in Computer Graphics

by

Mauricio Alfredo Vines Neuwirth

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Ph.D. degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Fluid simulations for computer graphics applications have attracted the attention of many researchers and practitioners due to the enhanced realism that natural phenomena simulation adds to graphical applications. Vortex methods are receiving increasing attention from the computer graphics community for simple and direct modeling of complex flow phenomena such as turbulence. However, vortex methods have not been developed yet to the level of other techniques for fluid simulation in computer graphics. In this work we present a novel simulation framework to model inviscid flows using Lagrangian vortex particle methods. We introduce novel stable methods to solve the vorticity flow equations that produce highly detailed visual fluid simulations. We incorporate the full interplay of solids and fluids in our framework. The coupling between free-form solids, represented by arbitrary surface meshes and fluids simulated with vortex methods, leads to visually rich simulations. Previous vortex simulators only focus on modeling the solid as a boundary for the flow. We model solid boundaries using an extended potential flow at the solid surface coupled with a boundary layer simulation. This allows the accurate simulation of two processes of visual interest. The first is the introduction of surface vorticity in the main flow as turbulence (*vortex shedding*). The second is the motion of the solid induced by fluid forces, which is calculated from the dynamics of vorticity in the flow and the rate of vorticity creation at solid surfaces. We demonstrate high quality results of our methods simulating flows around solid objects and solid object propulsion due to flows. This work ameliorates one of the important omissions in the development of vortex methods for computer graphics, which is the simulation of two-way coupling of solids and fluids.

Acknowledgements

I would like to express my heartfelt gratitude to Professor Jochen Lang, for his dedicated mentorship, support and patience. It has been a privilege and a pleasure to have had the opportunity of developing my research under Professor Lang's direction. I would also like to thank Professor Won-Sook Lee for her support and technical discussion.

I would also like to thank the members of the examiner committee, Professor Dirk Arnold, Professor Ali Arya, Professor Eric Dubois and Professor Emil Petriu, for their encouraging, throughtful and constructive feedback on my work. I would like to extend my gratitude to Professor Catherine Mavriplis, for sharing her knowledge of aerodynamics with me. I would also like to thank Professor Ioan Nistor for his advice on academic and professional matters.

I would like to extend my gratitude to Ben Houston, CTO at Exocortex Technologies Inc. for bringing the field of vortex methods for fluid simulation to my attention. I would also like to thank him for sharing his knowledge on fluid simulation for computer graphics, and for welcoming into the Exocortex team during my internship throughout our several fruitful collaborations. Also I would like to thank Erik Uggeldahl from Exocortex Technologies, Inc. for his assistance in rendering the scenes presented in this thesis.

The work in this thesis was co-funded by the University of Ottawa, NSERC, Mitacs Canada and FedDev Ontario, and I would like to thank all these organizations for their generosity. I would like to thank my colleagues at the University of Ottawa for their support and comments on my work.

I would like to thank my parents, Ramon and Astrid who always encouraged me to pursue my intellectual interests. Also I would like to thank my extended family for their unconditional support, especially Raul Vines and Gladys Vasquez for their kindness and Dr. Eugenio Vines, Dr. Enrique Vines and Dr. Ximena Ortega.

Finally I would like to thank my dear friends for their encouragement and support. Thanks Alejandro Luján, Daniela Pueyo, Koichi Paxton whose presence eased the most challenging moments during my studies. Thanks to John and Shiroe Paxton, Dwight Rudisuela and Sue Smee for sharing their life experiences and guidance with me. Thanks to Yongjie Yoon, Gregorio Villalobos, Ana Carballo, Amy Hrdina, Anton Rudisuela, Seyed Miharteri and Houman Rastgar for their friendship.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Commonly seen fluid phenomena, such as rising smoke or water flow past a rock, exhibit visually attractive features and are desirable natural phenomena to include in computer graphics applications. Because of this, physical simulation of fluids has gained popularity in computer graphics over the past decade and simulation of fluids has been used in many different applications, ranging from movies and video games to virtual training software.

Fluid simulation methods employed in computer graphics depart from their counterparts in computational fluid dynamics (CFD) in engineering. CFD simulations are employed to obtain numerically accurate results for structure and machinery design. In contrast, fluid simulations for computer graphics pursue the following three main goals [129]:

- **Visual plausibility.** This is the main goal in computer graphics since the visual aspects of the simulation determine the graphical quality of results. Numerical accuracy plays a much less important role than in CFD.

- **Efficiency.** Many fluid simulations are computationally expensive, because this requires solving complex equations that describe the flow motion. Whereas in engineering applications, performance may not be a key issue, performance requirements in computer graphics applications are generally stricter. For example, interactive applications require 24 updates per second.

- **Stability.** In most graphical applications, simulation times are arbitrarily determined by the user, *e.g.*, in videogames. Therefore, stable algorithms are required to stably run simulations for extended periods of time.

Additionally, in many scenarios in computer graphics applications, it is required for fluids to behave in specific, predetermined ways to achieve a visual effect [137, 92, 134]. For example, a simulation may require that candle smoke adopt a specific user-defined shape, which often would not be a physically accurate result.

All of the above goals for fluid simulation represent research areas in their own right. These goals are given different emphasis depending on the application scenario. For example, interactive applications such as videogames require highly efficient computations, since they usually run on commodity hardware. High efficiency in this case is often obtained by sacrificing visual quality. In this kind of simulations, numerical stability is a key factor, as simulation times are arbitrary, usually defined by the user. Other applications, such as visual effects in the movie industry require a high level of visual detail and performance requirements are not as strict. These simulations are often run on farms of processors, which can perform a high amount of concurrent operations.

We focus on high-detail visual simulations of fluids for offline applications, such as visual effects for movies. In this context we focus our attention on obtaining visually plausible simulations. A reference example of the kind of visual simulations we pursue is shown in Figure 1.1.

We note that a key element of visual plausibility is to correctly model the interplay between solids and fluids, *i.e.*, the influence that solids and fluids have on each other's motion, and this is one of the main features of our methods.

We also address the issue of stability, by developing methods to solve the vorticity transport equation overcoming stability issues identified previously in literature [121, 105]. We discuss efficiency as an avenue for future improvement of the results we present in this work.

We devise novel stable methods to simulate fluids interacting with solids that produce rich visual simulations with increased details compared to state-of-the-art methods in computer graphics.

1.1 Motivation

The current state-of-the-art of fluid simulation for computer graphics is strongly focused on grid methods, where the simulation space is partitioned in a series of discrete elements (either voxels or tetrahedra). These methods present two major challenges for obtaining visually rich fluid simulations. First, the simulation space is restricted to the bounding box of the grid. Large simulations as well as full periodic flow patterns are severely

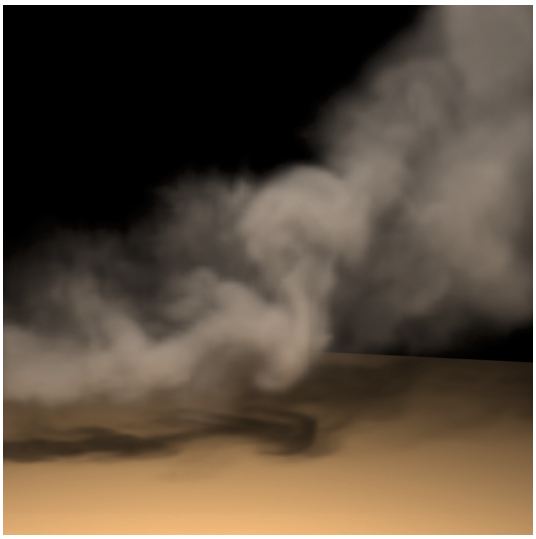


Figure 1.1: Example of smoke simulation in computer graphics from Fedkiw R., Stam J., and Jensen H. "Visual simulation of smoke," SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, © 2001 ACM, Inc. `http://doi.acm.org/10.1145/383259.383260`.

restricted because of the grid size. Second, fine sub-grid flow details are lost due to mesh resolution. The second and most critical of these challenges has been addressed by re-introducing dissipated motion in grids [41], [109].

Lagrangian methods, in contrast, do not require an explicit discretization of the full domain. Discrete simulation nodes, *e.g.*, particles, drive the dynamic behavior of fluids. Lagrangian particle methods do not depend on a limiting bounding box defining the simulation domain. Also, it is possible to represent a fluid at arbitrary levels of detail by controlling the amount of particles defining the flow motion. These characteristics enable the simulation of arbitrarily large phenomena, where a trade-off is required between the simulation dimension and the level of detail in order to maintain performance.

It is turbulence that gives fluid phenomena its visual interest, but simulating complex turbulent flows is one of the greatest challenges in fluid simulation. Turbulent flow is characterized by its randomness and its rotational nature. A natural way to model such flows is to explicitly represent and simulate them through their *vorticity*, or tendency to spin. Representing the flow through its vorticity has been introduced early on to computer graphics [52]. However, Lagrangian vortex methods have been applied only recently as a main method for high-quality 3D smoke simulations [105, 121, 5, 162, 149].

Lagrangian vortex methods have three key advantages over traditional grid methods:

they enable the representation of unbounded flows, at arbitrary scales and with minimal numerical dissipation. Simulating this kind of flows in traditional grid-based solvers is extremely challenging due to limitations imposed by grid size and resolution.

Lagrangian vortex methods also present several challenges to researchers and practitioners. The main challenge is the increased mathematical complexity associated with these methods. As opposed to other Lagrangian methods for simulating fluids (*e.g.*, smoothed particle hydrodynamics [96]), where a high amount of particles is used to approximate quantities in the simulation space, in Lagrangian vortex methods, a sparse and unevenly distributed amount of data is commonly used. This represents a challenge to approximate differential operators, and analytical solutions to the mathematical formulas are required.

Vortex methods provide a good representation of turbulent *i.e.*, highly rotational flows. However, non-rotational flows (*e.g.*, constant flows) are not properly represented by vortex methods and such flows are often dealt with separately.

To date though, vortex methods have not been as richly developed as grid-based methods. Current Lagrangian vortex simulators focus on the flow evolution and treat solid objects only as boundaries for the flow, without computing the motion induced on a solid object by a flow.

This work ameliorates this issue by introducing novel techniques for fluid simulation using Lagrangian vortex methods which address one of the major omissions in the state-of-the-art vortex methods in computer graphics: simulating the full interplay between solids and fluids. This kind of simulations require computing the force and torque exerted by the fluid on a solid object. These forces have been studied in the context of aerodynamics and naval design, where solid shapes are determined as part of the design process. In contrast, our goal is to produce visually realistic simulations of the interplay between solids and fluids for rigid solid objects of arbitrary shape. Many assumptions commonly employed in engineering applications do not hold in this case and we develop methods that are able to simulate two-way coupling of fluids and solids of general shapes, producing high quality flows suitable for high-end computer graphics applications.

Implementing two-way solid fluid coupling does not only add to the visual richness of fluid simulations, but it also calculates the motion of objects dragged by the flow. Such motions would be difficult for an artist to design manually and simulations can also greatly benefit from the straightforward simulation of turbulence that Lagrangian vortex methods provide.

1.2 Thesis Statement

Lagrangian vortex methods exhibit superior qualities for producing high-quality fluid simulations of arbitrary size and level of detail compared with Eulerian grid-based methods. Complex solid-fluid interaction can be synthesized using Lagrangian particle methods with no loss of visual fidelity compared to obtainable results using grid methods.

Lagrangian vortex methods (using either particles or curves) as primitive simulation elements enable the definition of fluid detail at arbitrary detail levels. The full spectrum of rotational flows can be defined by controlling the vorticity encoded on each simulation node. This enables the synthesis of turbulent behavior at arbitrary levels of detail. The level of detail is not restricted by a discrete partition of space, and it can be arbitrarily controlled by the user.

One of the key challenges in Lagrangian particle methods is the accurate representation of differential operators at arbitrary locations in space. These are usually approximated by a weighted average of flow properties encoded by the discrete simulation nodes. This is particularly challenging when computing the motion induced by a flow on a solid object. The force a fluid exerts on a solid object is the result of surface forces induced by pressure and viscous drag on the solid surface. Both can be represented directly in terms of vorticity related quantities, therefore vortex methods allow to accurately calculate the fluid forces exerted on a solid object when both the fluid and the solid have a Lagrangian representation.

We show that highly realistic flows can be produced with Lagrangian vortex particles and also that this technique can be applied to model the full interplay of solids and fluids. We show that extending the flow velocity with a potential flow at the solid surface and a boundary layer model suffice to realistically model a two-way coupled system of fluid and solids. Moreover, simulation of such phenomena can be achieved with enhanced visual fidelity compared with traditional grid solvers.

1.3 Contributions

In this work we propose a novel framework for animating smoke using Lagrangian vortex particles interacting with arbitrary rigid solid objects represented as polygonal surfaces. Modeling the flow around a solid object is performed employing an *inviscid* (*i.e.*, non-viscous) and irrotational flow induced by a solid surface which is coupled with a boundary layer model. The boundary layer is a very thin layer of fluid that adheres to the solid

surface. Boundary layer simulation enables us to model the flow separation from the solid surface, which is fundamental for force calculations on solid objects as demonstrated by Prandtl [111]. Then we can compute the force and torque exerted by the fluid on solid objects.

Our novel contributions, besides various practical realizations of existing methods for computer graphics are the following:

- We simulate inviscid flows around solid objects using a novel combination of vortex particle methods and a singular source distribution at the solid surface. The source distribution is discretized and computed using *panel methods*.
- We couple our inviscid potential simulation model with a physically based boundary layer model that accurately simulates flow separation from solid surfaces. Our boundary layer model is based on the simulation of viscous effects only at the solid surface and it enables the correct simulation of flows around objects of diverse shapes, as demonstrated in several examples.
- We develop a novel method for two-way coupling of solids and fluids based on the dynamics of vorticity at the solid surface, which enables computing the pressure force exerted by the fluid on the solid object.

We compute velocity fields from vortex particles using finite influence radius kernels. We establish the necessary conditions of these kernels for obtaining visually plausible simulations and we discuss the required conditions for obtaining physically accurate simulations.

1.3.1 Publications

During the development of this research, the following articles for refereed publications have been produced:

M. Vines, W.-S. Lee, and C. Mavriplis. Computer Animation Challenges for Computational Fluid Dynamics. *International Journal of Computational Fluid Dynamics*, volume 26.6-10 (June-December), pages 407-434, 2012.

M. Vines, and W.-S. Lee. A Simple Force Computation Method for Two-Way Fluid-Solid Coupling in Two-Dimensional Vortical Flows. *Computer Graphics International*, CGI 2011, Ottawa, Canada, 2011.

M. Vines, J. Mora, and W.-S. Lee. Haptic Display of 3D Liquids for Interactive Applications. *IEEE Consumer Electronics Games Innovation Conference*, ICE-GIC 2009, London, United Kingdom, 2009.

M. Vines, J. Mora, and W.-S. Lee. Real-time haptic display of fluids. In *Proc. of the 2009 C3S2E Conference*, pages 149-153, 2009. (Best Poster Award)

Additionally, the following presentations have been given by the candidate:

M. Vines, and W.-S. Lee. A Particle Method for Haptic Interaction with 3D Fluids. *University of Ottawa Research Day Poster Competition 2011.* (Third Place Award Computer Science Category)

M. Vines, and W.-S. Lee. Statistical Fluid Flow Synthesis. In *Proc. of the Graphics Interface Poster Session*, GI 2010, pages 4-5, Ottawa, Ontario, Canada, 2010 (online proceedings).

M. Vines, and W.-S. Lee. Simulation and Visualization of Fluid Mixing on Graphics Hardware. *University of Ottawa Research Day Poster Competition 2010.* (Second Place Award Computer Science Category)

M. Vines, W.-S. Lee and F. Moreaux. Registro y Parametrización de superficies 3D. In *Proc. of the First International Symposium on Regional Development*, Nayarit, Mexico, 2007 (In Spanish). (Invited paper).

1.4 Thesis Overview

We represent the flow using Lagrangian particle vortex methods. A particle encodes a spin direction and a magnitude. With this information, a spinning motion is induced in the flow by each particle. The flow velocity induced by this kind of particle is calculated by a numerical variant of the Biot-Savart Law. This kind of representation captures only the flow's rotational features. Non-rotational features are represented by a user-defined external flow (*e.g.*, a constant flow) that can be used for global simulation control.

The total free flow velocity is calculated as the sum of the user-defined external flow and the velocity field induced by the vortex particles. Interaction with solid objects is achieved in three steps:

i. Flow around the solid object is computed by adding a potential flow at the solid surface using a *source* distribution at the solid surface. This source distribution cancels any flow penetration into the object, hence satisfying a *no penetration* boundary condition on the solid object's surface.

ii. In addition to the no-penetration condition, we enforce a *slip* condition which imposes constraints on the flow tangential to the surface. This can range from a *no-slip* condition where the flow adheres to the solid surface, to a *free-slip* condition, where the flow does not adhere to the surface at all. An adherence level is defined by the user. Enforcing a slip condition on the solid surface enables us to model the boundary layer behavior. This process is simulated by the creation of vortices at the solid surface which are introduced into the main flow.

iii. Forces acting on the solid object are computed and the solid object's velocity and position are updated. Force calculations are performed based on the rate of vorticity creation at the solid surface, since this physical quantity is directly related with the pressure gradient at the solid surface.

We focus our development on simulation of gases and we demonstrate the results with our techniques using smoke examples. Smoke is a visible suspended concentration of particles and gases. As such, smoke has a highly dynamic and ill-defined shape which is commonly characterized in terms of its density distribution over space. Computational simulation of smoke is often performed by defining a region of the space which is filled with a fluid (*e.g.*, air) and introducing smoke as a density field, or particles into the simulation domain. Then smoke evolution is simulated as the evolution of a substance that is transported by the main fluid.

As opposed to the case of liquids, simulating smoke does not require tracking of a free surface, nor other features such as spray or droplets. In the smoke scenario fine flow features are defined by simulation nodes and are visualized through the smoke particle concentration which is rendered to the user.

1.4.1 Simulation Model

Our simulation is based on two main physical objects. The first is a set of Lagrangian vortex particles from which, as mentioned above, a free flow velocity is produced. These particles' positions evolve with the flow as the simulation advances. The second main component is a set of solid objects. These induce a potential flow that cancels any flow

through the surface and also emit vortices into the main flow as part of our boundary layer model.

Additionally we employ massless marker particles to represent the smoke density in the simulation. These particles advance according to the flow and solid's velocities. These particles have no impact on the simulation and are used only to visually render smoke.

The different simulation processes are organized in the following main simulation functionalities:

- **Free flow computation**. This process receives as input a set of vortex particles and the external flow and computes the total free flow velocity field.
- **Surface field calculation**. This process receives as input the free flow velocity field and a set of solid objects. It performs two main operations:
  1. Calculates the flow of the external velocity field through the solid surface.
  2. Generates a source velocity field that cancels the flow through the solid surface at each solid's face.
- **Boundary layer model**. This process receives as input the free flow external velocity field and performs the following steps:
  1. It computes the *slip* velocity, this is the tangential component of the velocity field at the surface such that the no-penetration condition is satisfied.
  2. Imposes a *slip* condition by creating vortices that cancel a user-defined proportion of the slip velocity at the solid surface.

  The output of this process is a set of newly created vortices that are introduced into the main flow, simulating the boundary layer separation process.
- **Force calculation**. This process receives as input the rate of vorticity creation at the solid surface and computes the net force and torque acting on the solid object.

After executing all the above processes, each solid object's velocity, orientation and position is updated, and both vortex and smoke particles advance. We present a high-level diagram summarizing the main component interactions in our system in Figure 1.2.

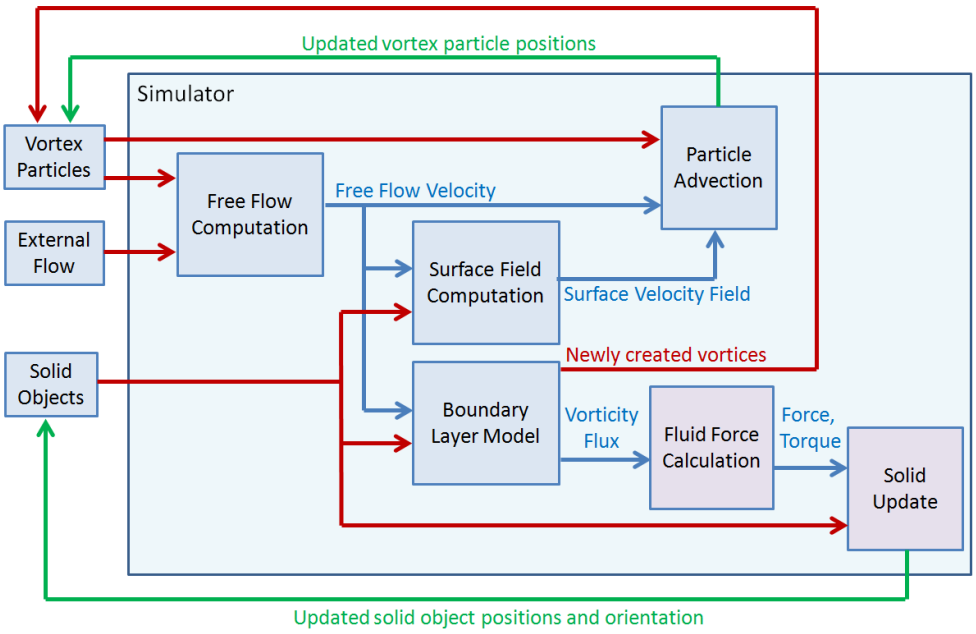


Figure 1.2: Schema of main simulation component interaction.

We demonstrate the quality of our results through simulation of commonly seen phenomena including free fall drag, turbulent flow past solid objects, and objects being suspended on jets.

1.4.2 Development Platform and Scene Generation Pipeline

We implement our methods in a prototype simulator written in C++ built using the Exocortex™ development platform under Microsoft Windows™. The Exocortex™ development platform provides a vast set of classes and functions for graphical simulation, including import and export functions for simulation objects, processor-optimized data structures and implementations of diverse algorithms for efficient graphics processing.

Each of our simulation scenes is implemented as a C++ program that loads the solid object's geometries, sets simulation parameters and invokes the main vortex particle simulator. At each time step our system writes to disk files with the simulation data which is graphically rendered using external applications.

Rendering is further performed using commercial software. Our scenes are loaded into Autodesk Softimage™ where they are rendered using Exocortex Fury2™ and MentalRay™. Images obtained with each rendering pass are composed using The Foundry Nuke™ to obtain the actual images shown in this work.

1.5 Document Organization

In Chapter 2 we present the basic flow equations in their vorticity formulations and the relevant definitions for force and torque exerted on a solid object by the flow. We also briefly introduce the main simulation paradigms and we discuss the compromises associated with each.

Then in Chapter 3 we present a review of literature of the most relevant work on fluid simulation for computer graphics. Here we discuss advances in all major topics in fluid simulation including grid and particle methods, turbulence models, solid fluid coupling and animation control. Additionally, we present specific developments on the most commonly simulated fluid phenomena, focusing on fire, smoke and liquids. Then, we briefly discuss the most commonly used techniques for rendering fluids to obtain a visually realistic representation of flows.

In Chapter 4 we present our free flow simulation model. Here we discuss our Lagrangian particle representation of vorticity and the methods we employ to solve the flow equations. We introduce novel stable methods to solve the flow equations and to synthesize realistic flow behavior.

In Chapter 5 we discuss how to model the flow around a solid object, *i.e.*, the solid object acts as a boundary for the flow. We introduce a potential flow model that ensures the impermeability of the solid object, *i.e.*, the flow does not penetrate the solid boundary. This is achieved by imposing conditions on the normal component of the flow velocity at the solid surface. We also discuss how visually realistic flows are obtained by imposing conditions on the tangential component of the flow at the solid surface, or *slip* velocity. The solid surface is modeled as a source of vorticity that is released into the flow as *vortex shedding*, which adds great visual appeal to fluid simulations.

In Chapter 6 we introduce two methods to compute fluid force and torque acting on solid objects based on a velocity-vorticity representation of the flow. The first is a simple method to compute the total force and torque acting on the center of mass of a system of solid objects. This method can be applied to couple a single solid object to a fluid, obtaining visually plausible results; however the simulation of several independent objects is not direct, limiting the applicability of this method. The second method we introduce overcomes this difficulty by considering the vorticity creation rate at the solid's surface to compute the pressure force exerted by the fluid.

In Chapter 7 we show further examples of flows interacting with objects using our methods. We compare our results with those obtained by others in the computer graphics

literature. We show how our methods can reproduce physical behavior as achieved using other state-of-the-art techniques with enhanced visual results. Then we evaluate the accuracy of our approach by comparing our numerical approximations with well-known physical results. We discuss conditions for numerical accuracy of our methods to converge to analytical solutions.

Finally in Chapter 8 we summarize our contributions, propose future research directions and present our conclusions.

Chapter 2

Background

Current methods for simulating fluids in computer animation are developed mainly from the physical description of fluid behavior. In this section we review the fundamental concepts which are the base of current fluid simulation techniques and we formally define the scope of our work.

2.1 Basic Flow Equations

Fluids are materials that deform continuously under shear stress. Among the wide range of possible fluids, we focus our attention to the class of Newtonian fluids. In these, the shear (or tangential) stress on an infinitesimal fluid element $\boldsymbol{\tau}$ is proportional to its deformation or *strain rate*. If we consider a two-dimensional fluid whose velocity is defined as a vector field $\mathbf{u} = [u_1, u_2]^T$ over the 2D space with points $\mathbf{x} = [x_1, x_2]^T$, a Newtonian fluid can be characterized succinctly as follows [48]:

$$\boldsymbol{\tau} = \mu \frac{\partial u_1}{\partial x_2}. \tag{2.1}$$

Here μ is the *dynamic viscosity* coefficient, which is a property of the fluid.

We focus our attention on the case of homogeneous Newtonian fluids, *i.e.*, we are not considering the interaction of two or more different fluids. Furthermore, we consider the case of incompressible fluids where the fluid density is constant. Such is the case of gases at low speed and liquids, when temperature changes have a negligible effect on the fluid's volume variation.

The motion of a fluid that satisfies the above conditions is determined by two fundamental equations: the Navier-Stokes Equation and the Continuity Equation. The Navier-Stokes Equation models the conservation of momentum of a body of fluid and it is derived from Newton's second law. It can be formulated as follows [48]:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u} - \frac{\nabla p}{\rho} + \mathbf{f}. \tag{2.2}$$

Here the symbols ν and ρ correspond to the fluid *kinematic viscosity* coefficient and density, respectively, which in our scenario are constant scalar values. The fluid's kinematic viscosity is defined as μ/ρ. The symbol p represents the fluid pressure and it is a scalar field and $\mathbf{f}$ corresponds to the acceleration due to external forces acting on the fluid. The symbol ∇ corresponds to the vector operator of partial derivatives, which in three dimensions corresponds to:

$$\nabla = \left[\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \frac{\partial}{\partial x_3}\right]^T.$$

Therefore ∇q corresponds to the gradient of the field q, whereas $\nabla^2 q$ corresponds to the Laplacian of q.

The first term in the right-hand side of Equation (2.2) is the *advection* term, which corresponds to the spatial acceleration of the fluid. The velocity field depends both on time and space, and given a fluid element, its position is also dependent on time. Then, the time derivative of the fluid velocity field is given by the multivariate chain rule as follows:

$$\begin{aligned}
\frac{d\mathbf{u}}{dt} &= \frac{\partial \mathbf{u}}{\partial t}\frac{dt}{dt} + \frac{\partial \mathbf{u}}{\partial x_1}\frac{dx_1}{dt} + \frac{\partial \mathbf{u}}{dx_2}\frac{dx_2}{dt} + \frac{\partial \mathbf{u}}{\partial x_3}\frac{dx_3}{dt} \\
&= \frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{u}}{\partial x_1}\mathbf{u}_1 + \frac{\partial \mathbf{u}}{\partial x_2}\mathbf{u}_2 + \frac{\partial \mathbf{u}}{\partial x_3}\mathbf{u}_3 \\
&= \frac{\partial \mathbf{u}}{\partial t} + \left(\mathbf{u}_1\frac{\partial}{\partial x_1} + \mathbf{u}_2\frac{\partial}{\partial x_2} + \mathbf{u}_3\frac{\partial}{\partial x_3}\right)\mathbf{u} \\
&= \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u}.
\end{aligned}$$

We see that the time derivative of $\mathbf{u}$, or its *total derivative*, corresponds to the variation of $\mathbf{u}$ with respect to time and space. The latter is represented by the advection term. This term can be intuitively understood as the transport of a quantity (such as heat) through a flow, *i.e.*, a quantity in the fluid will move following the velocity field

[131]. Velocity advection can be understood as *self-advection*, *i.e.*, how the velocity field moves under its own influence.

The second term on the right-hand side of Equation (2.2), corresponds to the *diffusion* term, which models the velocity damping due to viscosity. The third term corresponds to the *pressure* term. This models the fact that the fluid flows from high pressure areas to lower pressure ones. Finally the last term on the right-hand side of Equation (2.2) is the acceleration due to external forces, such as gravity.

The Continuity Equation represents the conservation of mass in a body of fluid and it is formulated as follows [48]:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0. \tag{2.3}$$

This formalizes the notion that any change in the fluid density within a fluid region corresponds to a flux of mass through the region's boundaries. For an incompressible fluid, where the density is a non-zero constant value, Equation (2.3) reduces to:

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

A velocity field that is divergence-free can be easily understood as a flow that does not have *sources* or *sinks*, which would correspond to the creation or destruction of a fluid mass, respectively. In this sense, any change in the amount of fluid in a determined region corresponds to the flow entering or exiting the region only.

Equations (2.2) and (2.4) form a system of non-linear partial differential equations for which no analytical solution is known, except for some specific cases. In computer graphics applications, these equations need to be discretized and solved numerically.

In the case of fluids with very low viscosity, such as gases, it is commonly assumed the fluid viscosity coefficient ν to be zero [48]. The dynamic behavior of these fluids is modeled by the Euler Equation, which is formulated as follows:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{\nabla p}{\rho} + \mathbf{f}. \tag{2.5}$$

The Euler Equation is commonly used by researchers and practitioners in computer graphics to model inviscid flow. In our case, we choose to employ an alternate formulation of the flow transport equation based on modeling the flow's tendency to spin, or *vorticity*. This formulation is discussed in the following section.

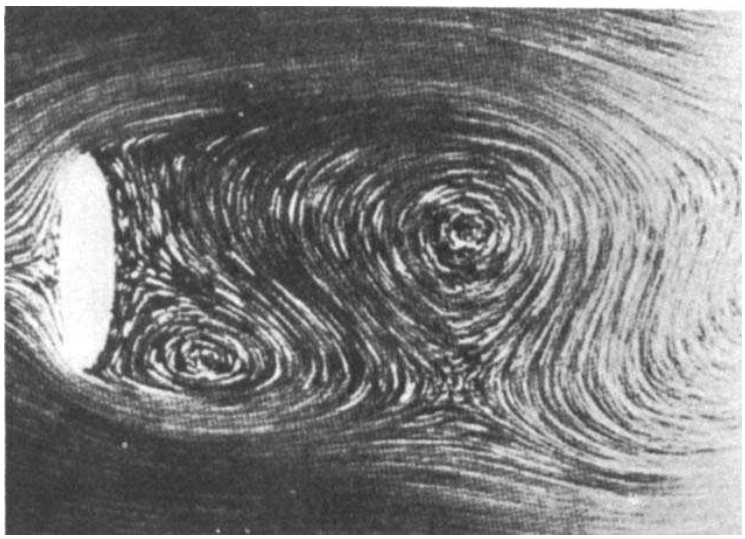


Figure 2.1: Vorticity generation in flow past an ellipsoid. Reprinted with persmission from Prandtl, L., and Tietjens, O. G. "Applied hydro- and aeromechanics", Engineering Societies Monographs. © 1957 Dover.

2.2 Vorticity

Flows commonly exhibit spinning features. This is the case in turbulent flows and flows past solid objects as shown in Figure 2.1. The tendency of a flow to spin is known as *vorticity* and it can be formulated as the curl of the velocity field [32]. The vorticity $\boldsymbol{\omega}$ of a flow is then defined as:

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}. \tag{2.6}$$

Two main features of turbulent flows are vorticity and randomness. Therefore vorticity becomes a key feature to model realistic, visually rich flows. Representing a flow by its vorticity is at the core of *Vortex methods*, which is the main modeling approach that we adopt in this work.

In the case of a flow past a solid, flow motion generates a torque on fluid particles in contact with the solid surface. This torque is then released into the main flow in a process known as *vortex shedding*. This process is illustrated in Figure 2.1 where clear rotational flow patterns can be seen in the flow past an ellipsoid.

Given a vorticity field $\boldsymbol{\omega}$, an equation for its evolution can be obtained from the Euler

Equation (or the Navier-Stokes Equation to model a viscous flow) [32]. The details of this derivation are included in Appendix A for the reader's convenience. The evolution of the vorticity field of an inviscid flow under irrotational external forces, such as gravity, is formulated as follows:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = -(\mathbf{u} \cdot \nabla)\boldsymbol{\omega} + (\boldsymbol{\omega} \cdot \nabla)\mathbf{u}. \tag{2.7}$$

The first term on the right hand side of Equation (2.7) is a vorticity advection term, and the second term corresponds to a vortex stretching term. This term models the change in direction and magnitude of the vorticity. This only appears in 3D flows, because in 2D the vorticity field is perpendicular to the plane of velocity and the stretching term vanishes. Evaluating both these terms require knowing the velocity field induced by the vorticity.

Given a vorticity field $\boldsymbol{\omega}$, the corresponding velocity field is determined by solving the following system of equations:

$$\begin{aligned} \nabla \times \mathbf{u} &= \boldsymbol{\omega}, \\ \nabla \cdot \mathbf{u} &= 0. \end{aligned} \tag{2.8}$$

By applying curl to the definition of vorticity, the above system of equations can be recast as the following Poisson equation:

$$-\nabla \times \boldsymbol{\omega} = \nabla^2 \mathbf{u}. \tag{2.9}$$

The fundamental solution to the above Poisson problem for the unknown velocity is the Biot-Savart Law, which is formulated as:

$$\mathbf{u}(\mathbf{x}) = \frac{1}{4\pi} \int_{Fluid} \boldsymbol{\omega}(\mathbf{z}) \times \frac{\mathbf{x} - \mathbf{z}}{\|\mathbf{x} - \mathbf{z}\|^3} d\mathbf{z}. \tag{2.10}$$

Therefore, for an incompressible flow, the vorticity field evolution can be evaluated using the velocity determined by the Biot-Savart Law. One of the advantages of vortex methods over classic methods based on solving the Navier-Stokes equation is that the pressure term p vanishes in the vorticity transport equation. Hence it is not necessary to introduce an additional equation to solve this term.

We solve Equation (2.7) for each time step using the velocity field given by the Biot-Savart Law. We discuss our computational representation of vorticity and the solution to the flow equations in Chapter 4.

2.3 Boundary Conditions

A partial differential equation such as equations (2.2) or (2.7) models the evolution of a vector field in an unbounded space. When the space is bounded (*e.g.*, a liquid in a bottle), it is necessary to constrain the physical quantities so that a specific behavior is guaranteed at the simulation domain boundary. Different conditions may be imposed for a specific simulation. For instance, consider a flow through a pipe. A liquid can be constrained to enter one (predefined) section of the pipe at a specific rate. This would represent an *inflow* boundary condition. If the pipe is impermeable, a constraint is needed to specify that the liquid cannot flow through the pipe surface. This would be a *no penetration* boundary condition.

Constraints such as the above can be represented generally as boundary conditions over the solutions of a partial differential equation for an unknown field $y(\mathbf{x})$ in a simulation domain Ω with boundary $\partial\Omega$ [103]. In fluid simulation for computer graphics the two following types of boundary conditions are most commonly employed:

(a) **Dirichlet**, or *first-type* boundary condition, which directly specifies the values of $\mathbf{y}$ at the boundary $\partial\Omega$, *i.e.*,

$$y = f(\mathbf{x}), \text{ for } \mathbf{x} \in \partial\Omega, \tag{2.11}$$

where $f(\mathbf{x})$ is a known function defined on $\partial\Omega$.

(b) **Neumann**, or *second-type* boundary condition, which specifies the values of the derivatives of $\mathbf{y}$ at the boundary $\partial\Omega$, *i.e.*,

$$\frac{\partial y}{\partial \mathbf{n}}(\mathbf{x}) = f(\mathbf{x}), \text{ for } \mathbf{x} \in \partial\Omega, \tag{2.12}$$

where $f(\mathbf{x})$ is a known function defined on $\partial\Omega$. Commonly $\mathbf{n}$ corresponds to the normal of the boundary $\partial\Omega$.

The boundary conditions we consider in this work are those related to the interaction of flows and solid objects. Viscous fluids adhere to a solid surface. If the solid is not moving a Dirichlet boundary condition that needs to be specified for the fluid velocity is:

$$\mathbf{u}(\mathbf{x}) = 0 \text{ for } \mathbf{x} \in \partial\Omega. \tag{2.13}$$

For an impermeable solid object, it is required that the flux of velocity through the surface is zero, this is:

$$\mathbf{u}(\mathbf{x}) \cdot \hat{\mathbf{n}} = 0. \tag{2.14}$$

Consider a static fluid in contact with a rigid planar wall. The no penetration boundary condition implies a Neumann condition for pressure across the boundary, since a pressure difference between the solid and the fluid would induce a flow. The corresponding boundary condition for pressure would be the following:

$$\frac{\partial p(\mathbf{x})}{\partial \hat{\mathbf{n}}} = 0 \text{ for } \mathbf{x} \in \partial\Omega. \tag{2.15}$$

In the case of vortex methods we do not need to specify conditions on the flow pressure as this quantity is absent from our vorticity transport definition. We discuss the required boundary conditions in our scenario in Chapter 5 and we discuss how these can be enforced in a Lagrangian simulation.

2.4 Basic Solution Methods

Simulating flows in computer graphics applications is done by discretizing the flow quantities over the simulation domain and then solving the required differential equations for obtaining the flow velocity, which is the main quantity of interest. Knowing the flow velocity enables to advect quantities in the flow such as heat, smoke density and tracking free surfaces in liquid simulations. Two main paradigms are identified to describe flows: the Eulerian and the Lagrangian approaches. In the following sections we briefly describe them.

2.4.1 Eulerian Methods

In the Eulerian approach fluid properties are computed at fixed locations. The simulation space is partitioned in a grid of cells, in which the flow properties are sampled and computed. The locations where the flow is computed depend on the grid definition and not the actual flow.

Early grid methods [46, 129] employ regular rectangular grids to sample flow properties. An example of the sampling used in the work by Foster and Metaxas [46] is shown in Figure 2.2.

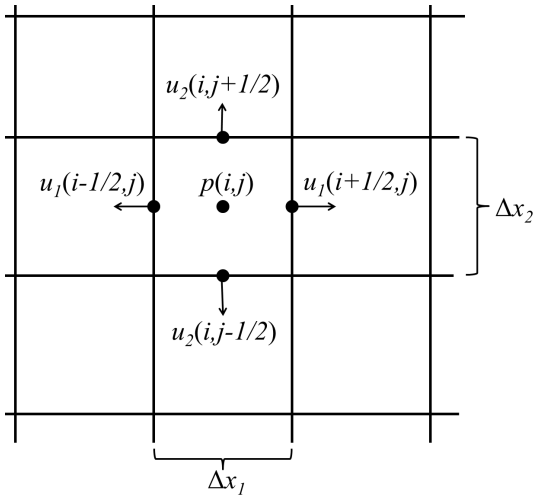


Figure 2.2: Example of a *staggered* grid used for sampling flows in 2D. Pressure values are sampled at the center of a cell (i, j), whereas the velocity components are sampled in the cell faces, *i.e.*, at locations $\pm 1/2$. In a collocated grid both pressure and velocity are sampled at the center of each cell.

.

One of the most attractive features of grid methods is the simple formulation of differential operators required for solving the flow equations. For instance, following the example grid in Figure 2.2, the flow divergence in the cell (i, j) can be easily defined as follows:

$$\nabla \cdot \mathbf{u}(i,j) = \frac{u_1(i+1/2,j) - u_1(i-1/2,j)}{\Delta x_1} + \frac{u_2(i,j+1/2) - u_2(i,j-1/2)}{\Delta x_2}. \quad (2.16)$$

Although mathematically convenient, grid methods have limitations. Simulations based on grid methods are bounded by the grid size. Any change on the size of the flow phenomenon that is being simulated requires remeshing. Also, sub-grid flow details cannot be represented and this usually leads to artificial dampening of the flow.

This discretization scheme has been widely studied and used in fluid simulations for computer graphics. Further details on different methods developed on grid methods are presented in Section 3.1.1.

2.4.2 Lagrangian Methods

In the Lagrangian approach the simulation nodes' positions evolve following a background vector field. This approach can be directly implemented using particles that advance following the flow velocity.

Particles encode their position and flow properties (*e.g.*, velocity, pressure and mass). Since fluid properties are commonly needed everywhere in space and not only at particle locations, interpolation methods are used to compute flow quantities from particles at arbitrary locations (*e.g.*, *smoothed particle hydrodynamics* in Section 3.1.5). Then a quantity q at an arbitrary location $\mathbf{x}$ can be calculated as:

$$q(\mathbf{x}) = \sum_j w(\|\mathbf{x} - \mathbf{x}_j\|) q_j. \tag{2.17}$$

The above equation is interpreted as calculating the quantity $q(\mathbf{x})$ as the sum of the contributions of each particle j to the quantity q weighted by their distance to the point of interest $\mathbf{x}$.

One of the main advantages of this kind of simulations is that the transport of quantities in the flow is trivially solved by simply advancing the particles using the flow velocity field. Another major advantage is that simulations are not necessarily confined to a bounding box, allowing greater flexibility on the size of the simulation space.

One major challenge is to to obtain accurate approximations for differential operators, as accuracy depends heavily on the weighting function $w(r)$ in Equation (2.17).

We employ a Lagrangian approach to model our flow. We encode the flow vorticity in discrete particles and we evolve them in the simulation domain. We describe in detail the discretization we use and the methods of solving the Vorticity Transport Equation (2.7) in Chapter 4.

2.5 Fluid Force on Solid Objects

The force exerted by a fluid on an immersed solid object, or *drag*, is determined by the stress tensor $\mathbb{T}$ which represents force per unit area. The stress tensor $\mathbb{T}$ can be formulated as follows [15]:

$$\mathbb{T}_{i,j} = -p\delta_{i,j} + \boldsymbol{\tau}_{i,j}, \tag{2.18}$$

where $i, j \in \{1, 2, 3\}$. Here δ is the Kronecker delta function, and $\boldsymbol{\tau}$ corresponds to the shear stress tensor. In three dimensions we have that $\mathbf{u} = [u_1, u_2, u_3]^T$ and $\mathbf{x} = [x_1, x_2, x_3]^T$. For a Newtonian fluid, the shear stress tensor is defined as:

$$\boldsymbol{\tau}_{i,j} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \tag{2.19}$$

The stress tensor encodes the surface forces the fluid exerts on the solid. Then the drag $\mathbf{F}$ can be obtained by integrating the stress tensor over the solid surface $\mathcal{S}$, *i.e.*,

$$\mathbf{F} = \int_{\mathcal{S}} \mathbb{T} \cdot \hat{\mathbf{n}} \, dS. \tag{2.20}$$

Cottet and Koumoutsakos [32] show that the integrand of the above equation can be rewritten in terms of vorticity as follows:

$$\mathbb{T} \cdot \hat{\mathbf{n}} = -p\hat{\mathbf{n}} + \mu \hat{\mathbf{n}} \times \boldsymbol{\omega}. \tag{2.21}$$

In the case of an inviscid flow, it is assumed that $\mu = 0$. Then Equation (2.20) is reduced to the following expression:

$$\mathbf{F} = -\int_{\mathcal{S}} p\hat{\mathbf{n}} \, dS. \tag{2.22}$$

Similarly, the torque around the solid's center of mass $\mathbf{x}_{CM}$ can be formulated in general terms as follows:

$$\mathbf{T} = \int_{\mathcal{S}} (\mathbf{x} - \mathbf{x}_{CM}) \times (\mathbb{T} \cdot \hat{\mathbf{n}}) \, dS. \tag{2.23}$$

Again, we replace the definition of the product of the stress tensor and the surface normal by the definition in Equation (2.21), and we obtain that for an inviscid flow, the torque around the solid's center of mass can be expressed as follows:

$$\mathbf{T} = -\int_{\mathcal{S}} (\mathbf{x} - \mathbf{x}_{CM}) \times p\hat{\mathbf{n}}\, dS \tag{2.24}$$

Both the force and torque exerted by a fluid on a solid object are required to synthesize the motion of the solid object due to a flow around it. These quantities are crucial for modeling two-way coupling of solids and fluids.

Modeling the flow directly using vorticity as in Equation (2.7) introduces the challenge of computing the fluid pressure on the solid surface, as the pressure term is not present in the vorticity transport equation. Pressure then must be computed from the vorticity field, and in particular from the rate of generation of vorticity at a solid surface, or vortex shedding. We discuss the calculation of force and torque exerted on a solid surface by a flow modeled with vortex methods in Chapter 6.

2.6 Chapter Summary

In this chapter we have presented the main equations of interest for our modeling and simulation scenario. We have defined the evolution equations for an inviscid flow and its formulation in terms of vorticity. We have also reviewed how to calculate the velocity field induced by a given vorticity field, which is defined by the Biot-Savart Law.

We have also given a general formulation of the force and torque exerted by a fluid on an immersed solid. These have to be known quantities to compute the influence of the flow on the motion of the solid object, which enables modeling two-way solid fluid coupling.

In the next chapter we present an overview of the most common techniques used in fluid simulation in computer graphics and we discuss their main features. In later chapters we discuss computational details on vorticity representation and velocity, force and torque computations.

Chapter 3

Related Work

3.1 Main Simulation Methods

A number of methods for simulating fluids in computer animation have been developed in the past years. These aim at preserving visual plausibility at low computational costs. Due to the complexity of fluid phenomena, there have been very few successful attempts at developing simple *ad hoc* algorithms to reproduce fluid behavior. Procedural liquids introduced by Fournier and Reeves [47] are among the few exceptions, where sinusoidal waves are superimposed to reproduce the appearance of a liquid surface. Peachey [106] follows a similar approach by combining sinusoidal waves with particles to simulate spray. A large body of work in fluid simulation is focused on methods to solve the Navier-Stokes equations and in this section we review the most important scenarios and techniques used in computer animation.

3.1.1 Grid Based Methods

Foster and Metaxas [46] introduce the earliest method for fluid simulation using the Navier-Stokes equations in computer animation using forward Euler time integration with a relaxation scheme to compute pressure. Here a simulation of 50x15x40 grid cells runs at 2.22 updates per second on a Silicon Graphics Crimson R4000 computer. Stam [129] addresses stability issues allowing larger time steps. Most current grid solvers are based on this method, and we highlight its main features. The momentum equation's time step is split into each of its terms (external forcing, advection, diffusion, pressure gradient), which are solved sequentially. Incompressibility or zero divergence is then enforced at the last step.

This is a first-order accurate algorithm and its results are adequate for computer animation. External forcing is solved using forward Euler integration. The advection term is solved with a semi-Lagrangian approach known as back-tracing that is used to advect other properties such as temperature and smoke density. One downside of this approach is the numerical dissipation it introduces due to interpolation. The diffusion term is solved implicitly by backwards Euler integration commonly with the conjugate gradient (CG) method. The intermediate velocity field is not necessarily solenoidal. Incompressibility is enforced in a process known as *pressure projection*.

The most expensive operations are solving the Poisson equations for diffusion and pressure projection as they require solving a linear system, compared with advection which only requires a particle tracing step and interpolation for each velocity sample. Two main improvements in this method are focused on performing efficient computations for solving the Poisson equations and reducing numerical dissipation due to interpolation in the semi-Lagrangian method.

Adaptive Grids

One problem that arises in applying grid methods is the trade-off between efficiency and visual results. High detail fluid simulations are achieved by using high resolution grids at a high computational cost. Multi-grid methods have been applied in computer animation for speeding up the evaluation of differential equations. These are based on the observation that many relaxation methods smooth high-frequency error very fast and low frequencies are smoothed very slowly. Solving an iteration of a relaxation method on a coarser grid, computational costs can be significantly reduced. Then the strategy is to evaluate the solution of the whole problem in a hierarchy of discretizations. The highest detail problem is solved via an approximation to a coarser grid which is obtained by down-sampling. The coarser grid is solved recursively by an even coarser grid. Solutions are interpolated to a finer grid and are used as initial guess for the next step of the iterative relaxation method.

In certain scenarios such as simulating a pool of water, a crucial region is the liquid surface due to its visual importance. A high resolution grid improves the visual quality of the simulation, but this is not required in areas far from the surface. Adaptive mesh refinement consists of enhancing detail only in areas of visual interest such as object boundaries and liquid surfaces, reducing the computational cost in other areas by keeping a coarser grid. This has been applied both to regular meshes in Lossaso *et al.* [88] and unstructured tetrahedral meshes in Batty *et al.* [11].

Figure 3.1: Results of liquid simulation using adaptive mesh refinement using a tetrahedron discretization from Batty *et al.* [11] where the mesh ranges from 400K to 1.1M tetrahedra. Reproduced with permission from Batty, C., Xenos, S., and Houston, B., 2010. "Tetrahedral Embedded Boundary Methods for Accurate and Flexible Adaptive Fluids," Computer Graphics Forum, 29 (2), 695- 704. © 2010, Wiley and Sons. Reproduced by kind permission of the Eurographics Association.

Losasso *et al.* [88] use a staggered grid, where each cell may be refined into eight new cells. All scalar values except pressure are stored at the nodes or corners of the cell to facilitate computations. Coarsening corresponds to creating a new cell where values are obtained by averaging the values of the corresponding fine grid cells. Refinement is achieved by adding new nodes, whose values are obtained by averaging as well. Despite the increased resolution, one problem in the octree structure is the discrete jump in grid size which does not allow a smooth transition between resolutions and introduces distortions in the flow. Batty *et al.* [11] address this problem by using an unstructured tetrahedral grid for discretization thereby increasing the visual quality of the flow obtained. This is implemented on a 4-core Intel i7 860 processor, requiring 31 seconds per time step for simulations employing 400K to 1.1M tetrahedrons. Results of this technique applied to liquid animation are shown in Figure 3.1.

Chentanez and Müller [26] reduce the grid complexity in liquid simulations by the use of *tall cells* in regular, structured grids. Liquid cells that are far from the surface are merged along the vertical direction only. These tall cells are topped by a layer of regular-size cells that provide the required level of detail at the liquid surface.

Model Reduction

One way to reduce computational costs of simulations in computer animation is to perform some pre-computation steps in advance and use this information at run-time. Treuille *et al.* [136] and Wicke *et al.* [151] adopt this strategy. Treuille *et al.* [136] combine a flow pre-computation stage with a model reduction approach. Each field in the grid can be seen as an n-dimensional vector, where n is the number of grid cells. The fluid equations are projected into a reduced space of dimension $m < n$ to be solved efficiently. A Galerkin projection is used to obtain the fluid evolution equations in the reduced space. An orthonormal matrix $\mathbf{B}$ is found such that if $\mathbf{u}$ is a vector in $\mathbb{R}^n$ and $\mathbf{r}$ its corresponding vector in $\mathbb{R}^m$, then $\mathbf{u} = \mathbf{B}\mathbf{r}$ and $\mathbf{r} \approx \mathbf{B}^T\mathbf{u}$. A linear equation of the form $\partial\mathbf{u}/\partial t = \mathbf{M}\mathbf{u}$ can be expressed in the reduced space by the following Galerkin projection:

$$\frac{\partial \mathbf{r}}{\partial t} = \mathbf{B}^T\mathbf{M}\mathbf{B}\mathbf{r}. \tag{3.1}$$

The matrix $\mathbf{B}$ is found through the following pre-computation process. Solenoidal velocity examples are obtained from several high-resolution grid simulations of fluid interacting with solids with free-slip boundary condition. These velocities form a basis $\mathbf{U}$. The orthonormal reduced basis $\mathbf{B}$ is chosen as the first m eigenvectors of $\mathbf{U}\mathbf{U}^T$ since this minimizes the reconstruction error:

$$E = \left\|\mathbf{U} - \mathbf{B}\mathbf{B}^T\mathbf{U}\right\|_F^2. \tag{3.2}$$

Treuille *et al.* [136] show that this basis is divergence-free and satisfies the free-slip condition by analyzing the linear properties of both constraints. In each simulation step, the momentum equation is split and each linear term is solved directly using the Galerkin projection above. The advection term is linearized by assuming velocities are constant over a time step. This allows the creation of a matrix $\mathbf{A}_\mathbf{u}$ that represents the advection given a specific velocity field. Advection is projected into the reduced space in two steps: first the matrix $\mathbf{A}_\mathbf{u}$ is represented as a linear combination of advection matrices for the velocity basis in the reduced space as follows:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{A}_\mathbf{u} \equiv (r_1\mathbf{A}_{\hat{\mathbf{u}}_1} + r_2\mathbf{A}_{\hat{\mathbf{u}}_2} + \cdots + r_m\mathbf{A}_{\hat{\mathbf{u}}_m})\mathbf{u}. \tag{3.3}$$

Figure 3.2: Models used for testing model reduction for fluid simulation in computer animation. Left: Leaves example from Treuille, A., Lewis, A., and Popović, Z. "Model reduction for real-time fluids," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2006, Vol. 25:3. © 2006 ACM, Inc. `http://doi.acm.org/10.1145/1179352.1141962`. Right: Spacecraft example simulated using the method by Wicke *et al.* [151], available online at `http://graphics.cs.cmu.edu/projects/modular_bases/`. Images courtesy of Carnegie Mellon Graphics Lab.

Here $\hat{\mathbf{u}}_k$, $k = 1, 2, \ldots, m$ represent the basis velocity vectors of the reduced space. Each of the advection matrices is projected then into the reduced space yielding:

$$\frac{\partial \mathbf{r}}{\partial t} = (r_1 \mathbf{B}^T \mathbf{A}_{\hat{\mathbf{u}}_1} \mathbf{B} + r_2 \mathbf{B}^T \mathbf{A}_{\hat{\mathbf{u}}_2} \mathbf{B} + \cdots + r_m \mathbf{B}^T \mathbf{A}_{\hat{\mathbf{u}}_m} \mathbf{B})\mathbf{r} = \mathbf{A}'\mathbf{r}. \tag{3.4}$$

This equation is solved exactly using the eigen-decomposition $\mathbf{A}' = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^{-1}$:

$$\mathbf{r}(t + \Delta t) = \mathbf{E} e^{\Delta t \Lambda} \mathbf{E}^{-1} \mathbf{r}. \tag{3.5}$$

Wicke *et al.* [151] indicate that this time integration is unconditionally stable. The diffusion term is represented in the reduced space directly with a Galerkin projection.

One drawback of this method is the restricted set of simulations that can be obtained from the pre-computed velocity fields. Wicke *et al.* [151] address this issue by employing modular reduced models that can be tiled to obtain a full simulation. The simulation space is divided in sub-domains and a reduced basis is obtained per sub-domain. Simulation is performed in each reduced model and additional constraints in the reduced space are used to account for discrepancies in the boundary values of neighboring domains.

Table 3.1: Performance comparison of simulations results using model reduction.

Approach	**Hardware**	**Simulation Nodes**	**Seconds per Update**
Treuille *et al.* [136] full simulation	3.6 GHz Intel Xeon CPU	256x64x128 grid	26
Treuille *et al.* [136] reduced model[a]		Same as above with 32 reduced bases	0.0017
Wicke *et al.* [151] full simulation	1.1 GHz AMD Opteron CPU, 16 GB RAM	248x196x71 grid	191
Wicke *et al.* [151] reduced model[b]		Same as above with 14 subdomains and 16 bases	0.024

[a]More than 400 hours of pre-processing were employed.
[b]33 hours of pre-processing were employed.

A comparison of performance of both approaches using the full simulation and model reduction is presented in Table 3.1.

3.1.2 Lattice Boltzmann Method

One desirable feature in simulation methods is the possibility of parallelizing the computations. A scenario where this can be achieved is when computations are performed locally in a reduced region. For this reason Wei *et al.* [145] introduce the Lattice Boltzmann method to computer animation for fire simulation. Wei *et al.* [146] further develops this method for smoke simulation. The Lattice-Boltzmann method is based on cellular automata, and models the fluid from the point of view of the microscopic interaction of packets of molecules. These particles move in a discrete lattice. A lattice specifies the connections between neighboring cells and therefore defines the possible directions for microscopic particles. Complex lattices are represented in terms of simpler sub-lattice configurations. At the core of this method are propagation and collision rules. A time step simulation consists of updating a packet distribution based on a collision operator and propagating distribution values. This propagation step is discrete and represents a jump from one simulation node to another one. Macroscopic quantities of density and velocity are calculated from distributions. The collision operator is chosen such

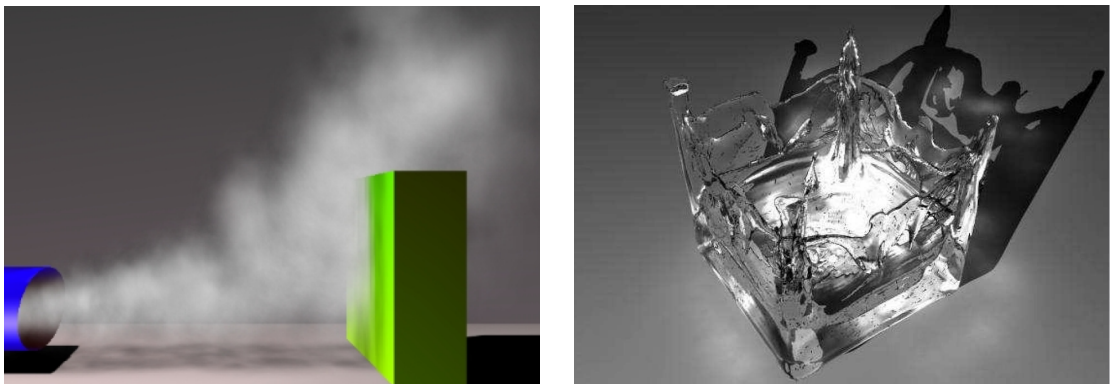


Figure 3.3: Simulation results using the Lattice Boltzmann method. Left: Smoke interaction with a solid obstacle from [146]. © 2004 IEEE. Reprinted, with permission, from Wei, X., Li W., Mueller, K. and Kaufman, A. "The lattice-Boltzmann method for simulating gaseous phenomena". IEEE Transactions on Visualization and Computer Graphics, Vol. 10:2. pages 164–176. Right: Animation of liquid inside a box from [135]. Reprinted from Proceedings of VMV 2004, Thürey, N. and Rüde, U. "Free surface Lattice-Boltzmann fluid simulations with and without level sets", pages 199-207, Copyright (2004), with permission from IOS Press.

that mass and momentum are conserved. Wei *et al.* [146] define this collision operator based on an equilibrium packet which depends on macroscopic velocity and density (see Qian and D'Humires [113]). From the equilibrium distribution, the collision operator is defined with a relaxation time scale. Chen and Doolen [22] show that the particles' behavior described with this method approximates that of the Navier-Stokes equations at a macroscopic level. This method has also been employed in other fluid simulations. Zhao *et al.* [167] develops this method for melting simulation and Park *et al.* [104] for mixing. As each particle distribution update is performed locally, the Lattice Boltzmann method can be easily parallelized. Most implementations are performed on parallel graphics hardware. Wei *et al.* [145] report performance increase by a factor of 50 with this kind of implementation. Notice also that flow is computed without solving for pressure. These two advantages make this approach attractive for interactive applications.

Türey and Rüde [135] extend this approach by incorporating free surfaces using level sets and a mass tracking method. They illustrate their technique by simulating a liquid drop in motion. Wei *et al.* [147] apply the Lattice-Boltzmann method to interaction

with different objects. Chu and Tai [29] use this method to simulate the absorption and dispersion of ink in paper incorporating deposition, liquid percolation and transport in the simulated media. Some simulation results obtained using the Lattice Boltzmann Method are shown in Figure 3.3.

3.1.3 Finite Volume Method

In the *Finite Volume* method an irregular, unstructured mesh is used to conform to objects' shapes avoiding distortions due to discretizations over a regular mesh. The finite volume method is based on solving the integral of the Navier-Stokes equations over a specified volume V and applying Stokes' theorem to obtain surface integrals. Again the momentum equation is split to achieve efficient solutions. The space is usually modeled as an unstructured tetrahedral mesh. Velocities are sampled at tetrahedron faces whereas scalar values are sampled at cell centers or at each tetrahedron's vertices. One of the main advantages of this method is the comparative simplicity with respect to regular grid methods to obtain high quality results with relatively low resolutions for complex solid objects' geometries. Feldman *et al.* [43] adapt the finite volume method to simulate fluids on tetrahedral meshes that deform by applying a modified semi-Lagrangian advection method that accounts for the mesh deformation. Wendt *et al.* [150] uses this method for modeling fluid interaction with complex boundary geometry. Klingner, *et al.* [76] use a tetrahedral mesh that conforms to solid boundaries and it is recalculated at each time step as solid boundaries move. At each time step fluid properties stored on the mesh are transferred to the new grid that is generated.

Miklós [94] employs this approach for wave simulation. A height-field approximation is used to model a liquid surface and an additional layer is used to represent the over-turning of waves. Brochu *et al.* [17] apply a finite volume method in adaptive tetrahedral mesh simulations. Sin *et al.* [124] construct a mesh from particles and perform pressure projection similarly to the finite volume method.

Chentanez *et al.* [23] generate a tetrahedral grid from a triangular mesh that defines the surface using semi-Lagrangian contouring (see Section 3.2.1). Incompressibility is enforced locally on each tetrahedron. To avoid loss of volume due to features smaller than a grid size, artificial thickening is applied to areas with small features.

Figure 3.4: Ocean simulations obtained using Fourier Transform methods. Left: ocean rendering from Tessendorf, J., 2001. "Simulating ocean water". In: SIGGRAPH 2001 course notes (Course 47), ACM. Copyright (2001) J. Tessendorf. Right: results obtained using the method by Chiu and Chang [27]. © 2006 IEEE. Reprinted, with permission, from Chiu, Y.F. and Chang, C.F., 2006. "GPU-based ocean rendering". In: 2006 IEEE international conference on multimedia and expo, 9-12 July, Toronto, Ontario, Canada. IEEE, pages 2125-2128.

3.1.4 Fourier Transform Based Solvers

There are specific scenarios where high detail simulations can be obtained at low computational costs. Such is the case of simulating flow with periodic boundary conditions. Although this kind of flow does not exist in nature, it can be used to represent large bodies of water such as the ocean. Ocean wave modeling is probably where Fourier Transform methods have been applied most extensively in computer animation. The ocean surface is modeled as superimposed sinusoidal waves which are efficiently decomposed using the Fast Fourier Transform method (FFT) for efficient computation. Random wave amplitudes produce visually plausible results.

A wave is characterized by its wave vector $\mathbf{k}$ and its magnitude, the wave number k. The direction of a wave is perpendicular to the wave crest; the larger the wave number, the smaller the space between waves. A function is represented as a sum of waves, and the Fourier transform assigns the corresponding weight to each. Tessendorf [133] presents a technique which has become the base for many current methods. Here the ocean surface is modeled as the summation of complex sinusoids for different wave vectors. Waves are propagated in both positive and negative directions in Fourier space. It is possible to

relate the wave number k and its frequency. In deep waters for example, this relationship is given by the wave number and the gravitational acceleration as $\omega^2(k) = gk$. Producing a visually attractive appearance of the ocean surface is achieved by generating random Fourier amplitudes h_0 for waves as follows:

$$h_0(\mathbf{k}) = \frac{1}{\sqrt{2}}(\xi_r + i\xi_i)\sqrt{P_h}. \tag{3.6}$$

Here ξ_r and ξ_i are two random variables with Gaussian distribution and P_h is a modified Phillips spectrum, which is a semi-empirical ocean wave fluctuation model employed in oceanography. Tessendorf [133] uses the following:

$$P_h(\mathbf{k}) = A\frac{e^{-1/(kL)^2}}{k^4}\left\|\hat{\mathbf{k}}\cdot\hat{\mathbf{w}}\right\|e^{-(kt)^2}. \tag{3.7}$$

Here A is a simulation parameter and $\mathbf{w}$ is the wind direction; L corresponds to the largest possible wave and $l < L$ is a cut-off amplitude used to improve convergence.

Cieutat *et al.* [30] further develop this method by incorporating interaction with solids, specifically ships. Chiu and Chang [27] enhance the approach by Tessendorf [133] with adaptive surface tessellation.

Stam [130] also applies Fourier Transform based methods achieving efficient and simple simulation of fluids with periodic boundary conditions. Lapierre *et al.* [81] develop a Fourier domain advection step. More recently, Long and Reinhard [86] present a 3D simulation model using cosine and sine transforms simulating free-slip boundary conditions.

3.1.5 Smoothed Particle Hydrodynamics

Although traditional grid methods allow accurate simulations, they require extremely high resolutions to handle thin features in liquids such as splashes and foam, considerably increasing the computational cost of such simulations. This motivates modeling the fluid using particles. These are either emitted to model an inflow, or initialized. For efficiency and to ensure particles are not only emitted, particles are removed after a time limit. A particle encodes its position, velocity, mass and the forces acting on it.

Desbrun and Cani [35] introduce the technique of Smoothed Particle Hydrodynamics (SPH) to computer graphics to simulate motion of deformable objects. Here a continuum is simulated with a limited set of discrete particles by interpolating quantities stored in

each particle over locations where no particle is present using a smoothing kernel function. A kernel is a finite support, normalized radial basis function that defines each particle's contribution to the computations in arbitrary locations.

Two properties of this kind of simulation that make this technique attractive are: 1) by keeping the amount of particles and the mass of each particle constant, mass conservation is automatically satisfied and 2) as particles move with the fluid, the time derivative of the particle's velocity corresponds to the substantial (or total) derivative of the velocity field, *i.e.*,

$$\frac{\partial \mathbf{u}_{particle}}{\partial t} = \frac{\partial \mathbf{u}_{flow}}{\partial t} + (\mathbf{u}_{flow} \cdot \nabla)\mathbf{u}_{flow}. \tag{3.8}$$

Therefore the advection term does not need to be calculated explicitly. Other terms in the Navier-Stokes equations are computed by calculating the influence of particles on each other using the smoothing kernels. Computing the pressure term has been addressed in different ways. Müller *et al.* [96] employ an ideal gas equation to relate pressure and density, which is calculated from particles. This approximation is used for liquids and gases and results are those of a compressible flow. Premože *et al.* [112] enforce incompressibility by solving a Poisson equation. This yields accurate results, however at a high computational cost. Becker and Teschner [13] employ the Tait equation to relate density and pressure on liquid simulations obtaining flows with minimum compression. A performance comparison for the above approaches is presented in Table 3.2. Solenthaler and Pajarola [127] propagate density fluctuations throughout the fluid and pressure is adjusted in a prediction-correction scheme.

Adams *et al.* [1] further improve performance by adaptively increasing particle res-

Table 3.2: Performance comparison of particle simulation approaches.

Approach	**Hardware**	**Simulation Nodes**	**Updates per Second**
Müller *et al.* [96]	1.8 GHz Pentium IV CPU	1300 particles	25
Premože *et al.* [112]	1.7 GHz Pentium IV CPU, 512 MB RAM	10K particles	1
Becker and Teschner [13]	3.4 GHz Pentium IV CPU, 2GB RAM	200K particles	0.133

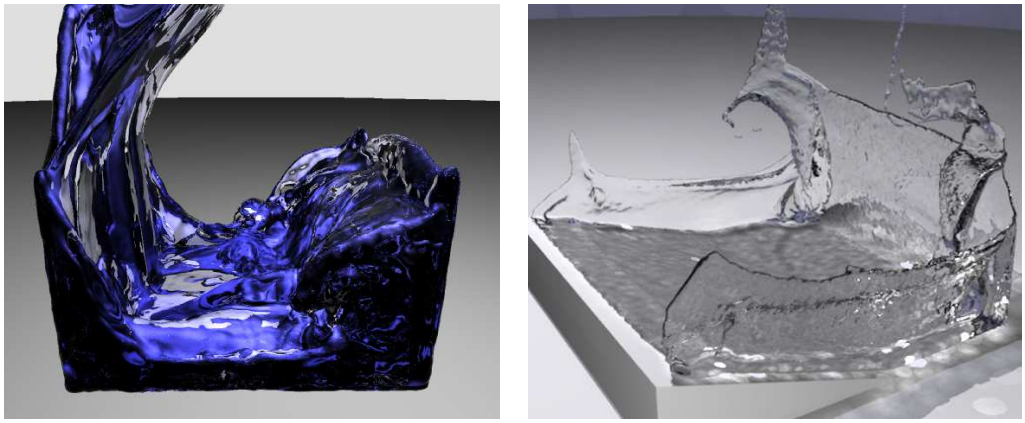


Figure 3.5: Results of simulating sloshing liquids using particle methods. Left: using SPH from Premože *et al.* [112] with 150K particles. Reproduced with permission from Premože, S., Tasdizen, T., Bigler, J., Lefohn, A., and Whitaker, R.T., 2003. "Particle-Based Simulation of Fluids". Computer Graphics Forum, 22 (3), 401-410. © 2003, Wiley and Sons. Reproduced by kind permission of the Eurographics Association. Right: using a hybrid PIC-FLIP particle simulation (see Section 3.1.7) from Zhu, Y. and Bridson, R. "Animating sand as a fluid," ACM Transactions on Graphics - Proceedings of ACM SIGGRAPH 2005, Vol. 24:3, © 2005 ACM, Inc. http://dx.doi.org/10.1145/1073204.1073298.

olution at liquid surfaces achieving a speed-up of computation time by a factor of 2.8 to 7.7 in their experiments. Solenthaler and Gross [125] simulate flow at two scales. Physical fluid properties are determined at a low resolution simulation, whereas areas of visual interest are simulated at higher resolution. Boundary conditions are defined for areas of different resolutions and force feedback is computed between them to achieve a consistent simulation. Computational costs in this approach are reduced by a factor of up to 6.7. Goswami and Pajarola, [53] determine convergence conditions and advection approximations based on identification of active and passive particles to speed up SPH simulations.

There are several challenges associated with SPH simulations. Irregular particle distribution and extraction of smooth surfaces for liquids using particles are both major problems as they may affect the visual quality of the simulation results. Also, defining a proper smoothing kernel radius to obtain a visually realistic simulation is not trivial for

every scenario. These are among the major limiting factors for the application of SPH to large simulations.

3.1.6 Vortex Methods

Vortex methods have been used in computer animation due to their ability to model complex flow with very sparse data and minimal numerical dissipation. Since Lagrangian particle methods achieve better performance in fluid simulations, modeling fluids using vortex methods in a Lagrangian framework is an attractive alternative. Gamito *et al.* [52] introduce vortex methods for the first time in computer animation for 2D fluid simulation. However, it is only recently that these methods have been applied to high quality 3D smoke animation.

The key idea in Vortex Methods is to model a flow from its vorticity. The vorticity evolution equation is obtained by applying the curl operator to the Navier-Stokes equations, which eliminates the pressure term for constant density flows and divides the advection term into a vorticity advection term and a vortex stretching term. To solve the former the velocity field has to be recovered from the vorticity field on the whole fluid region. This is done using the Biot-Savart law.

Park and Kim [105] model vorticity with vortex blobs, whereas Angelidis *et al.* [5] and Weißmann and Pinkall [149] employ closed filaments. This kind of discrete structures simplifies the evaluation of the Biot-Savart law as the integral over the whole fluid domain is evaluated only on particles, or along curves.

Since the Biot-Savart formula is singular at filament or particle locations, alternate kernels are used. Angelidis *et al.* [5] employ a fourth order polynomial kernel with limited support. Weißmann and Pinkall [149] apply a smoothed convolution of the Biot-Savart formula to both vortex filaments and passive smoke particles. This is shown to be equivalent to a global advection step using the original formulation of the Biot-Savart law.

Park and Kim [105] compute the vorticity stretching term by explicit numerical evaluation. Angelidis *et al.* [5] invoke Kelvin's theorem to account for the vortex stretching term by modifying the magnitude of vorticity as the filament enlarges and therefore keeping circulation constant. This approach has been used successfully to represent high quality smoke. Example snapshots from a simulation performed with this approach are shown in Figure 3.6.

Weißmann and Pinkall [149] reproduce smoke behavior around an object using vortex

Table 3.3: Performance comparison of vortex methods' approaches.

Approach	**Hardware**	**Simulation Nodes**	**Seconds per Update**
Angelidis *et al.* [5]	Pentium 4 2.4 GHz CPU; 256 MB RAM.	73,357 smoke particles.	4
Weißmann and Pinkall [149]	GeForce 8800 Ultra GPU	Approx. 1M smoke particles.	0.25

rings for smoke simulation and a vortex sheet to simulate solid surface vorticity. This vorticity is released into the main flow as filaments to produce vortex shedding. To avoid performance degradation due to a large number of filaments, vortex filaments are reconnected based on an energy functional minimization. In this way the amount of filaments in the simulation is kept low. A performance comparison of the approaches in Angelidis *et al.* [5] and Weißmann and Pinkall [149] is presented in Table 3.3.

Selle *et al.* [121] propose a hybrid approach using Eulerian grids and vortex particles. A background velocity field is computed on an Eulerian grid. This is used in a vortex particle simulation for advecting and stretching vortex particles. To avoid numerical instability, the vortex stretching term is normalized. The vorticity of each particle is reintroduced in the grid by performing two steps. First vorticity is interpolated at grid locations using a compact-support Gaussian kernel. Second, vorticity confinement is used to update the grid velocity with the vorticity from the particles. Here it is noted that adding vortex particles added less than a 5% of computational time, making this a suitable technique to produce visually rich flows with very little additional computational cost. An example of the results obtained with this technique is shown in Figure 3.6.

Elcott *et al.* [36] solve the vorticity transport equation on an unstructured tetrahedral mesh, where velocity, divergence and vorticity spin are treated as fluxes, which are computed for each tetrahedron. Physical quantities are represented as integral values that are stored at vertices, edges, triangles and tetrahedra.

More recently Lagrangian vortex methods have been employed to model hot buoyant smoke. Pfaff *et al.* [108] achieve this by incorporating a baroclinic term in the flow equations and simulating the interface between the hot gas and surrounding fluid using a vortex sheet. Kim *et al.* [72] simulate the creation of vortex particles in regions of high baroclinicity, introducing turbulence in a background flow.

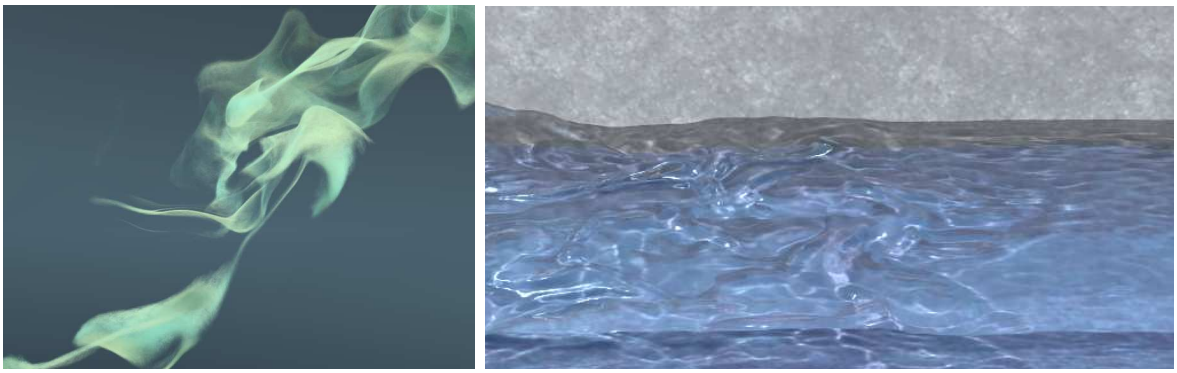


Figure 3.6: Left: Smoke example simulation using vortex rings from Angelidis, A., Neyret, F., Singh, K., and Nowrouzezahrai, D., 2006. "A controllable, fast and stable basis for vortex based smoke simulation," in Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer animation, SCA '06, Vienna, Austria Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 25-32. DOI: 10.2312/SCA/SCA06/025-032. © Eurographics Association 2006. Reproduced by kind permission of the Eurographics Association. Right: Example of water simulation using a hybrid approach with a grid and vortex particle simulation from Selle *et al.* [121]. Vortex particles are added to the left of the simulation space to display the difference in the liquid surface shape. Image from Selle A., Rasmussen N. and Fedkiw R. "A vortex particle method for smoke, water and explosions," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2005, Vol. 24:3, © 2005 ACM, Inc. `http://doi.acm.org/10.1145/1073204.1073282`.

Although the main interest in vortex methods is to obtain visually realistic representations of flows, diverse techniques to represent flow properties for analysis in engineering applications have been developed. Sadio *et al.* [119] present visualization methods based on sampling fluid properties along flow pathlines.

3.1.7 Hybrid Methods

As mentioned earlier, one problem with the semi-Lagrangian advection method proposed by Stam [129] is smoothing due to interpolation from the grid values. This is reflected as an additional viscosity which dampens the fluid. Hybrid methods have been applied to address this issue, by solving advection using particles and enforcing incompressibility using the Eulerian grid.

The most common hybrid approach is the particle-in-cell (PIC) method. Zhu and Bridson [171] apply this for simulating sand, and Horvath and Geiger [64] for fire. Particles encode the fluid properties; these are averaged into the grid, where all terms of the Navier-Stokes equations are calculated, except advection. Then the particles are advected using the grid velocity.

The PIC method is improved into the fluid implicit particle (FLIP) method as follows. Particle information is not directly updated with data from the grid. Instead, the variation of a quantity is computed from the grid, and this in turn is used to update particle information. This produces high quality results, as it does not introduce numerical dissipation. Zhu and Bridson [171] note this approach introduces noise, and a combination of PIC and FLIP updates is used in practice. An example flow obtained by this hybrid approach is shown in Figure 3.5 (right).

This idea has been applied in the context of SPH simulations as well. Raveendran *et al.* [115] interpolate particle velocities to a coarse grid, where they solve a Poisson equation for pressure. Then pressure values are interpolated back to particle positions, where pressure forces between particles are computed.

3.1.8 Discussion of Simulation Approaches

Each simulation technique presents its own advantages and disadvantages. Eulerian grids prove to easily allow precise calculations, but simulating fluids with this approach is computationally expensive. The Lattice-Boltzman method seems a promising technique due to its natural implementation in parallel processors, but deriving the Lattice-Boltzmann rules to simulate a specific phenomenon requires modifying the operators in this method

and this is not a trivial task. Achieving realistic and efficient computations in traditional grid methods require incorporating more sophisticated elements to improve performance, such as adaptive mesh refinement which is rather complex to implement in practice. Methods such as model reduction have good performance, but they require lengthy pre-computation time. Wicke *et al.* [151] report the bases used require from 1.5GB to 5.4GB of memory which is a large amount of data. Despite several improvements achieved using grid-based methods, *e.g.*, finite volumes methods and the Fourier Transform based methods, particle simulations are still among the methods with the best performance, so we adopt a Lagrangian approach for our simulation.

One common aspect of most simulation methods is the lack of modeling turbulence. This feature has been added to traditional simulations and it generally requires a large amount of computation, or hybrid approaches. These methods are surveyed in section Section 3.2.3. Vortex methods can be applied to represent all the features of interest of a flow in computer animation. Only recently these methods have been applied as complete solution methods to produce state-of-the-art animations and they are not yet fully developed in computer graphics. Recent advances in this technique in the computer graphics area and its potential to model complex flows makes it an attractive direction for development and we follow this direction in our research.

3.2 Computer Animation Applications

3.2.1 Liquids

The main issue in simulating liquids is to track the location of the surface due to its visual relevance. One alternative proposed early on by Foster and Metaxas [46] is to seed particles in the fluid and use them to track the location of the fluid. After several simulation steps however, particle positions may be uneven, producing gaps in the fluid surface. Also, extracting a smooth surface for rendering from particles is not a trivial task. In the following we outline some common methods to model a free surface that produce visually appealing results.

Level Set Methods for Water Surface Simulations

The level set technique is based on the construction of an implicit surface function ϕ whose zero-set represents the liquid surface. This function is created from simulation samples, either grid locations or particles. Usually the level set function values are set to

be negative inside the fluid and positive outside. Given an implicit surface defined in the simulation space, calculating the signed distance function can be done by numerically approximating $\|\nabla\phi\| = 1$. Geometric methods are based on searching the closest point on the surface for grid locations around the surface. For each grid location the signed distance to the surface is calculated and this is used to update the signed distance for other grid locations. A popular method to compute the signed distance is the fast marching algorithm described by Sethian [122].

The level set is advected with the fluid velocity using a semi-Lagrangian scheme [129], which smooths the level set. Since this method depends on the grid resolution, liquid is prone to losing fine detail and volume due to flow irregularities. To avoid this problem the level set method is combined with particles to avoid excessive smoothing. Foster and Fedkiw [45] propose seeding particles only near the fluid surface. Each particle is given a radius and the spherical shell of the particles defines an isocontour of the fluid surface. Once the level set is defined, both the level set and the particles are advected.

Particles near the boundary provide additional information which the level set alone cannot capture. Areas of high curvature of the level set indicate fluid splashing and particle information is used to locally modify the level set.

Enright *et al.* [39] further develop the above approach by seeding particles on both the liquid and empty sides of the surface, and storing their distance to the free surface. Particles that are advected far from the surface on the opposite side are considered *escaped*. They are used to indicate where the level set is inaccurate and then the level set is modified accordingly.

Several authors have applied level sets for simulating liquids. Adams *et al.* [1] use level sets to render liquids modeled with SPH. Losasso *et al.* [91] develop large simulations with high-quality interaction of solids and fluids by coupling SPH simulations and level sets. Losasso *et al.* [90] and Kang *et al.* [70] have applied level sets to model interfaces between different liquids. Mihalef *et al.* [93], Hong and Kim [61] and Hong *et al.* [62] demonstrate the level set technique for modeling bubbles. Examples of liquids modeled with level sets are shown in Figure 3.7.

Bargteil *et al.* [8] present the Semi-Lagrangian Contouring method to track liquid surfaces. Here a distance function is defined such that its zero set corresponds to the liquid surface. At each time step this function is advected using a semi-Lagrangian scheme. Then a triangular mesh is constructed from the zero-set of the distance function. This explicit polygonal mesh representation of the liquid surface avoids undesirable artifacts when advecting a triangular mesh directly, such as self-intersecting portions of the mesh.

(a) (b) (c)

Figure 3.7: Example water simulations using the following level set techniques: (a) Results from Enright D., Marschner S., Fedkiw R. "Animation and rendering of complex water surfaces," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2002, Vol 21:3 © 2002 ACM, Inc. `http://doi.acm.org/10.1145/566654.566645`. (b) The same scene as in (a) simulated with SPH from Losasso *et al.* [91]. © 2008 IEEE. Reprinted, with permission, from Losasso, F., Talton J., Kwatra N., Fedkiw R. "Two-way coupled SPH and particle level set fluid simulation". IEEE Transactions on Visualization and Computer Graphics, Vol. 14:4. pages 797–804. (c) Water level set constructed using adaptive particles from Adams B., Pauly M., Keiser, R., Guibas L. "Adaptively Sampled Particle Fluids," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2007, Vol 26: 3, © 2007 ACM, Inc. `http://doi.acm.org/10.1145/1276377.1276437`.

Wojtan *et al.* [153][154] also address evolving an explicit polygonal liquid surface. Here a distance field is defined and the liquid surface corresponds to its zero-set. Through the evaluation of the distance field, topological changes, such as merging and splitting of bodies of fluid, are detected and the surface is modified accordingly. Yu *et al.* [164] apply this idea to SPH simulations, where the liquid surface is advected directly using the velocity stored in particles. Then this surface is projected into an implicit surface defined by the particles. Surface tension is incorporated to the SPH particles as an additional force computed from the mean curvature of the liquid surface mesh.

Kim *et al.* [74] employ a level set to track the main body of liquid on a grid simulation using a hybrid PIC/FLIP method. Particles are then used to track fluid mass and volume in sub-grid features, and the physical interaction between them and the main body of liquid is simulated.

Misztal *et al.* [95] apply explicit surface tracking in tetrahedral meshes. Here the tetrahedral mesh adapts to the liquid surface evolution. Then surface tension and solid boundary conditions are solved as an optimization problem.

Height Field Approximations

Scenarios like the surface of a lake can be modeled with very simple techniques that produce real-time simulations that are adequate for interactive applications. Kass and Miller [71] introduce several simplifications of the fluid equations to achieve realistic and very efficient water surface motion. First, the surface is assumed to be a height field. Second, only horizontal components of the flow velocity are assumed to be relevant. Third, the horizontal velocity of the waves is approximately constant. In these cases, the Navier-Stokes equations can be simplified in 2D to:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + g\frac{\partial h}{\partial x} = 0, \tag{3.9a}$$

$$\frac{\partial d}{\partial t} + \frac{\partial ud}{\partial x} = 0. \tag{3.9b}$$

Here $h(x)$ is the height of the surface and $d(x)$ is the depth, defined as $d(x) = h(x) - b(x)$. Here $b(x)$ is the height of the container at the bottom of the fluid body. These equations can be further simplified by ignoring the advection term and linearizing around a constant value of h. By differentiating and combining the resulting equations, a second order wave equation for h is obtained. This equation has been applied in

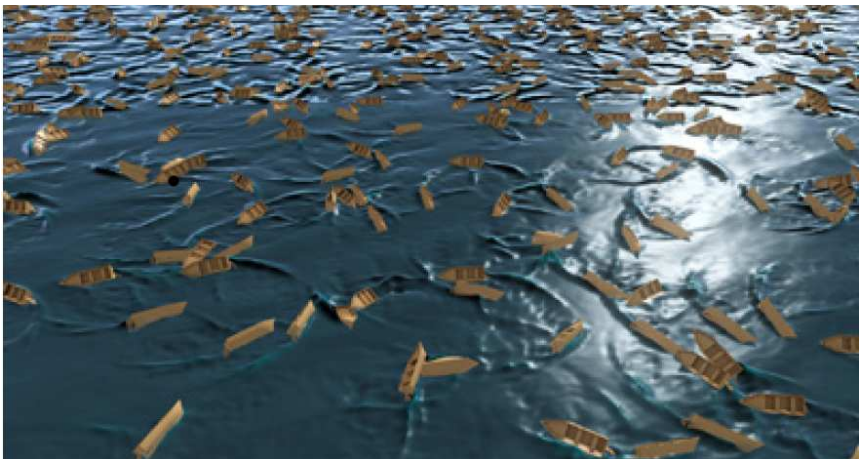


Figure 3.8: Water surface generated with a height field with solid interaction from Yuksel, C., House, D., Keyser, J. "Wave Particles," ACM Transactions on Graphics (TOG) Proceedings of ACM SIGGRAPH 2007, Vol. 26:3, 2007 ACM, Inc. `http://doi.acm.org/10.1145/1276377.1276501`.

computer animation to produce simple waves and it can be easily extended to 3D. This technique allows representation of the liquid surface as a polygonal mesh where each vertex is advanced using the wave equation. Yuksel *et al.* [165] improve this approach by incorporating particles into the height field calculation to simulate waves interacting with solid objects at very fast update rates. A large simulation consisting of 1,681 boats totaling 295,856 polygonal faces floating on a height field of resolution 256x512 and 8 million particles is shown in Figure 3.8. This scene was simulated on a commodity 2.13GHz Core 2 Duo processor achieving 4.84 updates per second. Chentanez and Müller [25] solve a similar form of the above equations on a grid which is combined with particles to produce effects that cannot be replicated within a height field model, such as splashing and overturning waves. These particles carry some mass and momentum which are removed from the main body of liquid. These quantities are re-inserted in the height field as particles fall into the main body of liquid.

3.2.2 Smoke and Fire

Simulating smoke and fire is achieved by simulating a gas that fills the entire simulation space. Now, both temperature T and smoke density s have to be calculated. Fedkiw *et al.* [41] evolve temperature and smoke density with an advection-diffusion and source transport equation. This is solved using the same methods as for liquids developed by Stam [129].

Buoyancy is taken into consideration by a Boussinesq approximation in the velocity equation. Detailed vertical smoke patterns are obtained by calculating a term that accounts for lost vorticity and introducing it into the flow as an additional external force in the Navier-Stokes Equations. This technique is known as *vorticity confinement* and is discussed in Section 3.2.3. An example of the results obtained is shown in Figure 1.1.

Fedkiw *et al.* [41] represent smoke density as a passive quantity that does not affect the flow. Bridson [15] incorporates density variations as an effect of temperature. As fluids expand due to temperature gradients, mass is not conserved if the pressure projection step employed by Stam [129] is applied. Bridson [15] incorporates density variations via the divergence to the Poisson equation for pressure to conserve mass.

Fire has been extensively studied in computer animation due to its visual features and the demand in the entertainment industry. To model fire, there are basically two problems that have to be addressed: the motion of the flame and the effect of the temperature on the surrounding fluid. Nguyen *et al.* [101] model the region where the combustion takes place, or core, with a surface represented as a level set. Assuming a constant burn speed S, the evolution equation for the level set becomes:

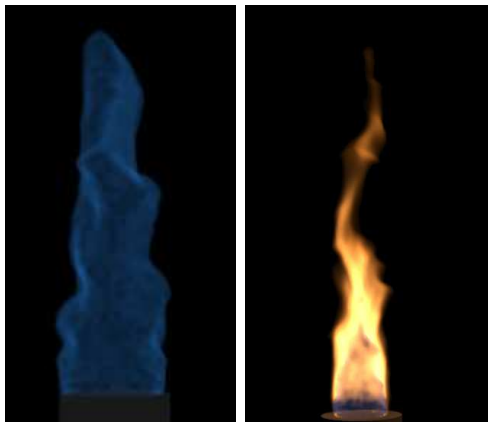
$$\frac{\partial \phi}{\partial t} + \mathbf{u}_{fuel} \cdot \nabla \phi = S. \tag{3.10}$$

Here the velocity of the fuel, $\mathbf{u}_{fuel}$, is a simulation parameter. An example of the core used in this approach is shown in Figure 3.9.

Fluid expansion is simulated when crossing the flame front. This expansion can be expressed as the following volume variation:

$$\Delta V = \left(\frac{\rho_{fuel}}{\rho_{burnt}} - 1 \right) S. \tag{3.11}$$

Here ρ_{burnt} is the density of the burnt gas, which is assumed to be smaller than the density of the fuel. The velocity of the burnt products is computed from the fuel velocity plus the speed of the volume variation in the normal direction of the flame front. As in the case of smoke, the expansion process is taken into account in the pressure projection step. A



(a) (b)

Figure 3.9: (a) Example combustion core and (b) flame result from Nguyen D., Fedkiw R., Jensen H. "Physically Based Modeling and Animation of Fire," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2002, Vol. 21:3, © 2002 ACM, Inc. `http://doi.acm.org/10.1145/566654.566643`.

weighted average of the density of the burnt products and the fuel can be considered, and the divergence value adjusted accordingly. As soon as the burnt products cross the flame front, they enter the simulation with a predefined temperature T_{max}. Then the decay of temperature is modeled by a black-body radiation conservation equation with a constant to control the amount of radiation in the simulation.

Zhao *et al.* [168] propose a model of expanding flames on solid fuels that are discretized using a grid. Flames are modeled as particles following a wind velocity field computed with the Lattice Boltzmann method. The fire front evolves tangentially to the surface of the fuel. This approach is efficiently implemented on a GPU achieving 14.2 updates per second for a 67x128x67 grid with an object consisting of 58837 voxels. An external wind field allows control over the animation, directing the fire in a specific direction and consuming an object in a specific manner. An example of the results obtained with this approach is shown in Figure 3.10.

Bridault *et al.* [14] use a multi resolution approach to handle multiple simultaneous flame simulations, such as several candles on a table. Small flame details are simulated with a small grid which is embedded in a larger, coarser grid for global control of the simulation, for instance, wind entering through a window. Flames located far from the

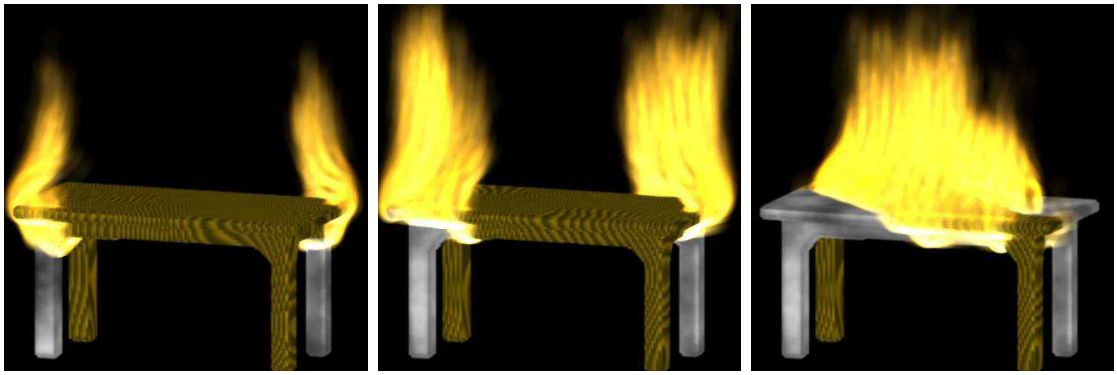


Figure 3.10: Snapshots of animation frames of a wooden table consumed by flames from [168]. Notice the tangential advancement of the fire front due to the presence of fuel in areas not yet burnt. © 2003 IEEE. Reprinted, with permission, from Zhao, Y., Wei, X., Fan, Z., Kaufman, A., Qin, H. 2003. "Voxels on fire," Proceedings of the 14th IEEE visualization 2003 (VIS03), VIS 03. Washington, DC: IEEE Computer Society, pages 271–278.

camera are simulated with particles as they do not require a high level of detail. A performance of 25 updates per second is achieved in 153 grids for up to 16 flames on a an Intel Core 2 Duo 6300 processor, before applying their multiple resolution approach which allows up to 256 flames.

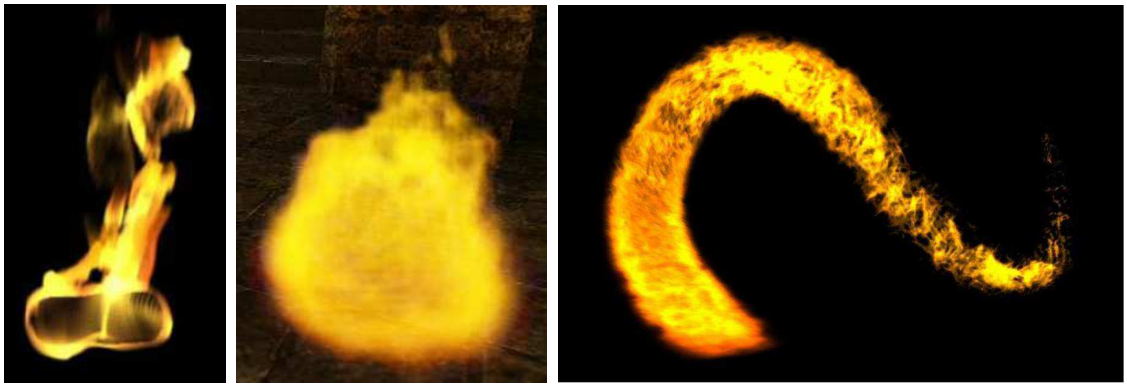
More realistic production of fire is usually performed as an off-line process. For example, Hong *et al.* [63], simulate fire using a complex Chapman-Jouguet detonation model for the front velocity, obtaining complex patterns that enhance the visual quality of flames. An example of the resulting flames is shown in Figure 3.11. Horvath and Geiger [64] perform a coarse simulation step using PIC and FLIP methods with a hierarchical decomposition. Here lower resolution grids are computed with a low-pass filter. Then velocities on the coarse grid are interpolated to locations of a higher resolution grid and are subtracted from the original values. This way, only high frequency details are kept at each grid, forming a hierarchy of different levels of detail where incompressibility is enforced at each level. Artists can increase the amount of rotational motion through vorticity confinement (see Section 3.2.3).

Achieving simulation control, as often required in the film industry, is achieved by simulating fluids departing from physically based models. One common alternative is to represent a flame by a skeleton and a surface surrounding it. The skeleton consists of

Figure 3.11: Fire simulation from Hong, J.-M., Shinar, T., Fedkiw, R. "Wrinkled Flames and Cellular Patterns," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2007, Vol. 26:3, © 2007 ACM, Inc. `http://doi.acm.org/10.1145/1276377.1276436`.

a series of connected nodes which move following a flow. Beaudoin *et al.* [12] employ a user-defined flow and flames do not affect the fluid motion. Lamorlette and Foster [80] simulate flame separation by breaking the skeleton, and noise is added to improve the flame's visual appearance. Fuller *et al.* [51] and Vanzine and Vrajitoru [138] follow the same approach and add noise in three octaves of the Kolmogorov spectrum (see Section 3.2.3) at 30 updates per second for a lattice with 135 points.

Along with fire, explosions are interesting phenomena to simulate due to their visual features. Yngve *et al.* [161] focus their attention on simulating shock waves using a viscous compressible fluid simulation, neglecting molecular vibration energy, dissociation and ionization. Fluid equations are solved using a fractional time step along with an internal energy evolution equation derived from the First Law of Thermodynamics and an ideal gas state equation. Feldman *et al.* [42] propose a different approach by simulating explosions with an incompressible fluid simulation grid and a particle simulation for fuel and soot. Flow divergence is positive where fluid is expanding and this value is used to adjust the pressure computation. Fuel and soot particles' temperature depends on the absorbed and emitted heat energy.



(a) (b) (c)

Figure 3.12: Examples of fire simulation. (a): Torch flame from Lamorlette, A., Foster, N. "Structural Modeling of Flames for a Production Environment," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2002, Vol. 21:3, © 2002 ACM, Inc. `http://doi.acm.org/10.1145/566654.566644`. (b): Results from Vanzine, Y. and Vrajitoru, D., 2008. "Pseudorandom Noise for Real-Time Volumetric Rendering of Fire in a Production System," in IEEE/ EG Symposium on Volume and Point-Based Graphics, Eurographics, 129-136. DOI: 10.2312/VG/VG-PBG08/129-136. © Eurographics Association 2008. Reproduced by kind permission of the Eurographics Association. (c): Results from Fuller, A.R., Krishnan, H., Mahrous, K.M., Hamann, B. and Joy, K.I. "Real-time procedural volumetric fire," in: Cohen, J.D., Turk, G. and Watson, B.A., eds., Proceedings of the 2007 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D), ACM Press, New York, pages 175–180. © 2007 ACM, Inc. `http://doi.acm.org/10.1145/1230100.1230131`.

3.2.3 Turbulence and Detail Preservation

Physically correct simulation of turbulence is a challenging problem in computational fluid dynamics as well as in computer animation. In the following we review some of the most important advances in this area.

Vorticity Confinement

An important drawback of grid based methods is the loss of fine detail due to grid resolution. Also as semi-Lagrangian methods average velocity values, an artificial viscous effect due to numerical dissipation appears in simulations. This becomes a problem in simulations of fluids that naturally exhibit a high degree of detail and fine vortices, such as smoke. Fedkiw *et al.* [41] introduce the *vorticity confinement* technique to address this problem without introducing non-physical methods to reproduce details such as adding random noise. Small scale structure is assumed to be due to vorticity ω. Lost vorticity due to numerical dissipation is reintroduced to the fluid as an additional force:

$$\mathbf{f} = \epsilon h(\mathbf{N} \times \boldsymbol{\omega}), \tag{3.12}$$

where $\epsilon > 0$ is an animation parameter that controls the amount of vorticity force re-introduced and h is the size of a grid cell. $\mathbf{N}$ is a normalized vorticity location vector defined to point to high vorticity regions:

$$\mathbf{N} = \frac{\nabla \|\boldsymbol{\omega}\|}{\|\nabla \|\boldsymbol{\omega}\|\|}. \tag{3.13}$$

An example of swirling smoke produced with this technique is shown in Figure 1.1. Nguyen *et al.* [101] use this technique for fire simulation and Losasso *et al.* [88] for simulations with adaptive meshes. Selle *et al.* [121] use vorticity confinement to add sub-grid detail obtained from a vortex particle simulation running parallel to a grid simulation. Hong *et al.* [62] employs vorticity confinement to achieve realistic bubble motion in an SPH simulation.

Procedural Turbulence

Different methods have been employed to add turbulence to a simulation to enhance its visual appeal. Stam and Fiume [131] simulate a background flow to which small scale velocity features are added. Small scale details are random vectors generated using

Fourier synthesis. Turbulence is controlled by the animator by modifying an energy equation based on Kolmogorov's energy spectrum.

Kolmogorov's energy spectrum is a characterization of the turbulent energy distribution and its transfer from large to small eddies, or vortices [82]. This characterization is based on a vortex wave number k. For a given wavenumber k, the turbulent energy spectrum is given by:

$$E(k) = C\psi^{\frac{2}{3}}k^{-\frac{5}{3}}. \tag{3.14}$$

Here C is a constant and ψ is the rate of energy dissipation per unit volume. This model describes a turbulent energy cascade, where energy that is produced at large scales is transferred to smaller scales. This transfer can be understood as the breakdown of large vortices into smaller ones. This process continues until energy is dissipated by viscosity ν. This occurs at a wavenumber k_ν characterized as follows:

$$k_\nu = D\nu^{-\frac{3}{4}}\psi^{\frac{1}{4}}. \tag{3.15}$$

Here D is a constant value. Neyret [100] employs advection and an energy cascade model to generate high-resolution turbulent visual representations of flow motion as textures, which are used to enhance detail of visual surfaces. Kim *et al.* [75] generate high-resolution flows employing a low resolution grid simulation, and adding high turbulence detail, which is generated using *wavelet noise*. This correspond to a noise function defined for a limited band of frequencies. This noise is added to the main flow according to an energy spectrum, which is obtained by computing the flow kinetic energy for different spectral bands. A user can control turbulent motion by defining the range of frequencies on which noise is added.

Yoon *et al.* [162] propose a different way to use particle methods for turbulence simulation. Here a background flow is generated using the approach by Stam [129] using a coarse grid. From this coarse grid, a high resolution grid is obtained via interpolation which is combined using vorticity confinement with a vortex particle simulation. Then the high resolution vorticity grid is blended with the low resolution grid with the background flow for increased detail.

Narain *et al.* [99] simulate a background flow on an Eulerian grid and turbulent energy evolution and cascading are simulated. The energy cascade is simulated from the energy distribution at different scales in the Kolmogorov spectrum. This energy is

used to generate *curl noise* particles. Curl noise corresponds to the addition of irregular flow patterns using a Perlin noise function [16]. An irregular spinning flow is obtained by applying curl to a Perlin noise vector. This flow then is added to a background flow to simulate turbulence. Narain *et al.* [99] produce highly detailed turbulent flow by simulating energy and transferring it as noise particles. This turbulent flow is produced at a much lower cost than using an equivalent fine grid. The turbulent energy distribution for a specific wave number k is assumed to be $E(k) \propto k^{-5/3}$. To facilitate simulation it is assumed that turbulence is isotropic and eddies are not correlated to the background flow. Wave numbers k_i are defined as the inverse of the wave length, this is $k_i = 1/l_i$ and at each octave of the Kolmogorov spectrum, the wave length is half the wave length in the previous octave, this is: $l_i = l_{i-1}/2$. The wave number k_0 is a user defined parameter.

Energy evolution is modeled based on an energy creation term G, an energy transfer rate to higher octaves Π and a diffusion term D as follows:

$$\frac{\partial E_0}{\partial t} + \mathbf{u} \cdot \nabla E_0 = G(E_0, \mathbf{u}) - \Pi(E_0) + D(E_0). \tag{3.16}$$

Turbulence is first produced at the scale of the background flow, which determines the *inertial scale* and then it is transferred to subsequent scales. Eddy creation is assumed to be due to a viscous effect that turbulence has on the main flow. The rate of turbulent energy production at the inertial scale used is:

$$G(E_0, \mathbf{u}) = \frac{E_0^{1/2}}{k_0} \sum_i \sum_j \left(\frac{\partial \mathbf{u}_i}{\partial x_j} + \frac{\partial \mathbf{u}_j}{\partial x_i} \right)^2. \tag{3.17}$$

Here sub-indices i and j indicate the i^{th} and j^{th} components of velocity, respectively. The factor $E_0^{1/2}/k_0$ corresponds to the eddy viscosity number and it determines the rate of eddy creation.

Energy is then transferred to smaller scales. The rate of this energy transfer from the inertial scale to higher ones is modeled by the cascade: $\Pi(E_0) = k_0 \sqrt{E_0}^3$. Energy is dissipated at the highest scales, which is simulated by simply stopping the spectrum transfer for a certain scale k_ν. Here the energy is simply dissipated and the energy at any scale above k_ν is assumed to be zero. This cut-off scale is defined as $k_\nu = \left(\Pi(E_0)/\nu^3\right)^{1/4}$.

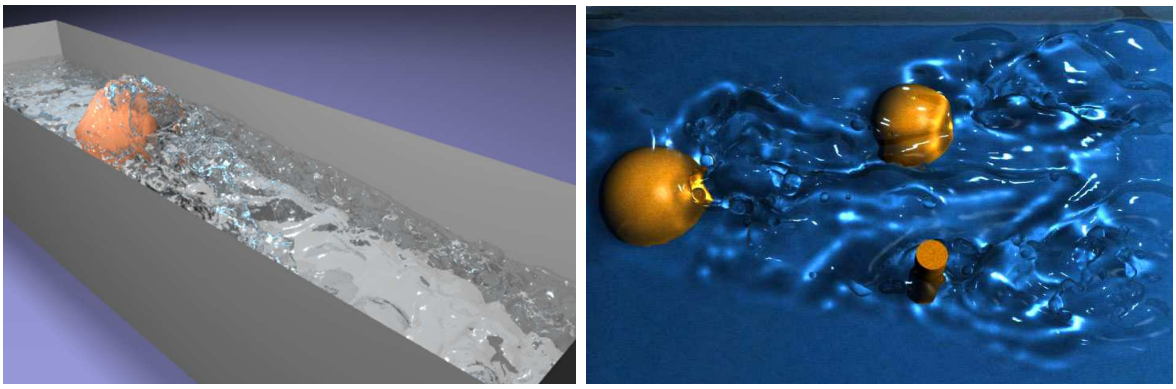


Figure 3.13: Left: Results obtained with procedural turbulence from Narain, R., Sewall, J., Carlson, M., Lin, M. C. "Fast Animation of Turbulence using transport and procedural synthesis," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2008, Vol. 27:5, © 2008 ACM, Inc. `http://doi.acm.org/10.1145/1409060.1409119`. Right: Detail of results obtained by synthetic turbulence from Pfaff N., Threy A., Selle A., Gross M. "Synthetic Turbulence using Artificial Boundary Layers," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2009, Vol. 28:5, © 2009 ACM, Inc. `http://doi.acm.org/10.1145/1618452.1618467`.

Energy is also spatially transferred to neighboring cells as a diffusion process. This diffusion process is modeled as:

$$D(E_0) = \nabla \cdot \left(l_0 \sqrt{E_0} \nabla E_0 \right). \tag{3.18}$$

Here the advection term is performed with the same scheme as the fluid solver for the background flow. Once the energy at each level is determined, turbulence is introduced through noise particles using a pre-computed 3D Perlin noise vector for each octave. An example of simulation results is shown in Figure 3.13.

Pfaff *et al.* [109] focus on vorticity creation at solid boundaries. Vorticity in the boundary layer is estimated from the universal law of the wall and is assumed to depend on the velocity and material constant only: $\omega = \beta(\mathbf{u}_s \times \mathbf{n})$, where β is a user-defined parameter accounting for both skin friction and fluid viscosity and $\mathbf{u}_s$ is the tangential component of the flow velocity at the solid surface. This boundary vorticity or *artificial boundary layer* is used to generate particles to model turbulent flow. The vorticity at

the boundary layer is pre-computed by time-averaging the tangential component of the flow velocity around a solid for different flow directions. Run-time turbulence synthesis is performed by adding vortex particles controlled by an energy model similar to the one in Narain *et al.* [99].

To determine the areas of vorticity creation, the artificial boundary layer data is used. The anisotropic part of the Reynolds stress tensor is used as an indicator of areas where vorticity should be created. Turbulent-viscosity hypothesis is adopted to compute the Reynolds stress tensor by assuming that the tensor depends on a turbulent viscosity term and the strain rate. By applying a series of simplifications, the magnitude of the stress tensor is approximated as: $\|a_{ij}\| \approx 2l_m^2 \|\omega\|^2$ using a *mixing length* (l_m) model. Vorticity is evaluated using the artificial boundary layer information found in the pre-computation step. Vorticity is then created in run-time with the following probability density:

$$p = c\Delta t \frac{\|a_{ij}\|}{\|\mathbf{u}\|^2}. \tag{3.19}$$

Here c is a constant to control vorticity creation. Vortex particle dynamics are governed by a vorticity transport equation. Particles are assigned a radius, which is modified as part of the energy transfer process. Particles with an associated wave number in the inertial scale are decayed into several particles with smaller wave numbers. In the model-dependent range, aligned particles that are closer than a radius distance are merged into a stronger particle with largest radius. Particle dissipation is achieved by eliminating particles. An example of simulation results is shown in Figure 3.13.

Zhao *et al.* [169] pre-compute different random force fields in the Fourier domain following the Kolmogorov spectrum. One of these force fields is randomly selected and added to a grid simulation as an external force. This can be done directly on the simulation grid, or on a refined grid obtained by interpolation from the original simulation, or on a SPH particle simulation. This is called *random forcing* and its purpose is twofold: add random, chaotic turbulent behavior and reduce computational time by not synthesizing turbulence on run-time. Performance comparison for procedural turbulence methods is shown in Table 3.4.

Table 3.4: Performance comparison of simulations using procedural turbulence.

Approach	**Hardware**	**Simulation Nodes**	**Seconds per Update**
Yoon *et al.* [162] full simulation.	Intel Quad Core 2.4GHz CPU; 2GB RAM	120x360x120 grid.	59
Yoon *et al.* [162] combined simulation.		30x90x30 background grid; 120x360x120 grid from particles	9.2
Narain *et al.* [99]	Intel Xeon 2.8 GHz CPU; 8GB RAM.	128x32x32 grid	45
Pfaff *et al.* [109][a]	3.0GHz Intel Core i7 CPU.	100x25x60 grid.	10
Zhao *et al.* [169] full simulation.	Intel Core 2 6300 1.86GHz, 4GB RAM.	64x32x50 grid	1.5
Zhao *et al.* [169] combined simulation.		32x16x25 background grid; 64x32x50 grid for turbulence	0.052
Zhao *et al.* [169] SPH simulation		4096 particles	0.019

[a]This employs a pre-computation step requiring 59 seconds per update per database parameter.

3.2.4 Fluid Control

One of the most challenging aspects of fluid simulation is allowing animators to control the simulation to achieve a specific visual effect. Such is the case of smoke or fire adopting a desired shape. A limited level of control can be achieved by using an external wind velocity field as proposed by Beaudoin *et al.* [12] for fire, and defining the trajectory axis of a series of vortex rings as presented by Angelidis *et al.* [5].

Treuille *et al.* [137] propose a more complex approach for control of smoke simulations using *keyframes*. These are specific simulation states identified by a user-defined smoke density distribution and velocity field. Fluid evolves in time to match keyframes by solving an optimization problem whose solution defines a wind force that drives the simulation to the defined keyframes. McNamara *et al.* [92] further develop this approach

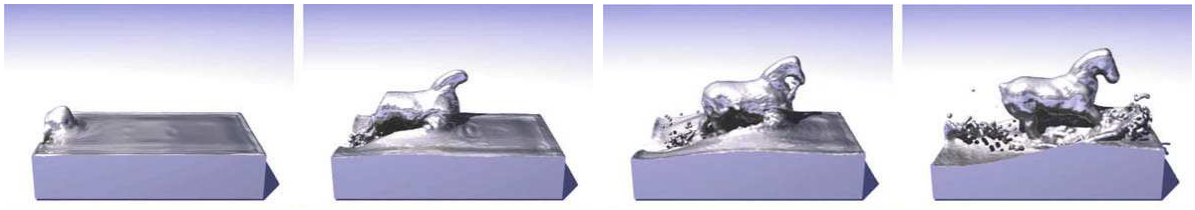


Figure 3.14: Examples of fluid simulations with animation control from Thürey, N., Keiser, R., Pauly, M., and Rüde, U., 2006. "Detail-preserving fluid control," in Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '06, Vienna, Austria Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 7-12. DOI: 10.2312/SCA/SCA06/007-013. © Eurographics Association 2006. Reproduced by kind permission of the Eurographics Association.

to allow liquid control as well by incorporating sources and scalable gradient calculation. Fattal and Lischinski [40] propose a method to evolve smoke density to a predefined configuration using a modified flow equation instead of an optimization framework. The flow equation incorporates an additional driving force which controls the transport of smoke density to a specific configuration, and a smoke gathering term to reduce the effects of dissipation.

Thürey *et al.* [134] propose a different approach, where a multi-scale approach is adopted using the Lattice-Boltzmann method (see Section 3.1.2). A coarse-level flow field is obtained by removing high frequencies in from the velocity field. Control forces are added to this coarse velocity field using particles, minimally affecting fine details. More recently, Nielsen and Bridson [102] control liquid simulations by guiding the simulation using an auxiliary shape which separates the region where high resolution is required to model the liquid surface from the inner part of the liquid where coarser resolutions can be used. Some results of visual fluid simulations with animation control are shown in Figure 3.14.

3.2.5 Other Phenomena

Whereas simulations of smoke, fire and liquids have been the focus of many researchers, there are other fluid phenomena that have been addressed in the computer graphics literature, which we briefly describe in this section.

One challenging problem is properly modeling viscous fluid behavior. Free-surface

viscous flows require special treatment of the free surface as demonstrated by Batty and Bridson [10] who employ a variational approach to model the motion of a highly viscous fluid surface.

Variable viscosity flows have also been addressed by researchers in computer graphics. One example of such flows are melting objects. Terzopoulos *et al.* [132] model both solids and fluids with particles. Solid particles are connected in a lattice, and inter-particle stiffness is modified to change the solid's softness. Carlson *et al.* [20] employ a user-defined relationship between viscosity and temperature. Then viscosity is sampled on an Eulerian grid and solved using a modified diffusion equation that accounts for variable viscosity. Lossaso *et al.* [89] create particles inside the solid object and simulate the erosion of the solid surface. As the surface erodes, particles inside the solid become part of the liquid. Wei *et al.* [144] store the matter state (liquid or solid) on a grid and simulate heat energy accumulation on the cells. Predefined rules determine the behavior fluid contained on cells. Müller, *et al.* [98] and Solenthaler *et al.* [128] present SPH models for solids and fluids that enable simulation of melting solids. Zhao *et al.* [167] model melting objects using the Lattice-Boltzmann method by modifying update rules for particle package distributions.

Mixing of fluids is an example of flow where both density and viscosity variations become important. That is the case of miscible fluids, which can be mixed at all proportions, such as water and honey, or immiscible fluids, such as water and oil. Zhu *et al.* [170] introduce a Lattice-Boltzmann equation for mixing two miscible fluids. Park *et al.* [104] develop a framework for simulating miscible and immiscible fluids of different densities also using the Lattice-Boltzmann method. Immisicble fluids phase separation is modeled using the Cahn-Hillard Equations. Kang *et al.* [70] simulate mixing in an Eulerian grid tracking volume fractions for different fluids in each cell.

Multiphase flows have also been addressed in the computer graphics literature, most notably the case of bubbles, as they enhance the visual appearance of liquids. Greenwood and House [54] employ passive bubbles that follow a background flow. Bubbles pop following a probability distribution based on the bubble's radius and lifetime. Foam is simulated by overlapping bubbles and computing pressure and viscosity forces. Similarly, Hong *et al.* [62] calculate the background flow using a grid with adaptive refinement and bubbles are modeled using SPH. Ihmsen *et al.* [65], simulate bubbles and liquids with SPH where each phase is solved separately. Then both phases are coupled by computing drag due to buoyancy and attraction forces between bubble particles. Hong and Kim [61] focus on calculating pressure profiles at bubbles' surfaces and incorporating pressure

jumps into an Eulerian grid solver. Mihalef *et al.* [93] employ marker particles to generate bubbles in areas of high curvature. Kim *et al.* [73] model bubbles as a dispersed flow using a fraction field that represents the spatial average of liquid and air.

Losasso *et al.* [90] presents a more general treatment of immiscible fluids by tracking inter-fluid interfaces using level set methods. Kang *et al.* [70] and Bao *et al.* [7] simulate immiscible interacting liquids using volume fractions to track the location of different immiscible fluids. Solenthaler and Pajarola [126] handle density discontinuities at multiple fluid surfaces using a particle density measure in an SPH simulation.

3.2.6 Discussion of Computer Animation Applications

Different techniques have been applied to model a variety of different phenomena, from smoke, fire and water, which are the most common simulations, to complex multiphase fluid interaction. In each case, physics principles are applied to different extents, achieving visually plausible simulations with varying computational costs. One of the most common simulations is fire, where highly detailed simulations can be obtained using computationally expensive methods, whereas real-time simulations are only possible for very simple simulations, generally with less detail.

We notice that turbulence is generally addressed as an addition to the fluid simulation and not as an integral part of the base fluid solver. Such are the cases of vorticity confinement and procedural turbulence. We note that vortex methods allow obtaining irregular flows in a natural way within the framework of the main simulator.

For our research, we restrict our study to the scenario of a gas simulation. The most common simulation in this class is smoke, which is often modeled as an incompressible homogeneous fluid with constant density. These assumptions do not match the physical characteristics of the real phenomenon, as smoke particles' density is different from surrounding air; however visually plausible simulations have been obtained under these assumptions, for instance by Nguyen *et al.* [101], Park and Kim [105], Angelidis *et al.* [5] and Weißmann and Pinkall [149]. In the present work we concern ourselves with the problem of coupling an inviscid, homogeneous incompressible fluid with solid objects, and we do not address multiphase fluid behavior.

3.3 Solid-Fluid Coupling

In many scenarios in computer animation fluids interact with solids. Simulation of this phenomenon is a key component to achieve visually rich results in graphical fluid simulations. One-way coupling of solid and fluids can be performed in two directions: solid-to-fluid and fluid-to-solid coupling. In solid-to-fluid coupling the solid object acts as a boundary condition for the fluid, but is unaffected by the fluid motion. Here the solid is static or it has prescribed motion. Such is the case of water simulations in Foster and Fedkiw [45] and Enright *et al.* [39]. In fluid-to-solid coupling, the solid takes the velocity from the flow but it does not affect the fluid motion. Such is the case of a particle that is passively advected in a flow which can be used to model a very light object being dragged in the flow as in Foster and Metaxas [46].

In two-way solid fluid coupling, the solid and fluid motions affect each other. Then a solid motion induces a flow in a fluid, and the flow in turn affects the solid motion. In this section we present a brief summary of the main techniques used in computer animation in general and then for the special case of vortical flows.

3.3.1 Two-Way Solid-Fluid Coupling in Computer Graphics

The simplest case of this kind of coupling as described by Bridson [15] is called weak coupling. Here the coupling process is performed in two stages: first the flow is solved applying solid-fluid boundary conditions, obtaining both the fluid pressure and the shear stress tensor. Both are then used in the second stage to compute pressure and viscous drag on the solid object.

Carlson *et al.* [19] propose a different approach where solid objects are modeled as a fluid with a higher density in an Eulerian grid. This is referred by some authors as an *immersed boundary method*. Both solid and fluid are simulated using the Navier-Stokes equations for motion. The solid object rigidity is enforced by imposing a zero rate of strain in the solid object cells. Most of the computational power required to perform solid-fluid coupling in this approach is spent solving the fluid equations and the level set for liquid surfaces. Results of applying this technique are shown in Figure 3.15.

Klingner *et al.* [76] adopt a similar approach with tetrahedral meshes where both solid and fluid accelerations are computed as a coupled system in a single pressure projection step.

Chentanez *et al.* [24] present a two-way fluid solid coupling approach where the motion of a Lagrangian solid and a fluid are solved simultaneously by solving a single system of

Figure 3.15: Animation frames of interaction of solids and fluids in Carlson *et al.* [19] a liquid is advanced from the right, collides with some solid objects which move due to the fluid motion and these in turn alter the motion which is noticeable in the lower part of the scene. Reprinted from Carlson, M., Mucha, P. and Turk, G. "Rigid fluid: animating the interplay between rigid bodies and fluid," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2004, Vol. 23:3, © 2004 ACM, Inc. `http://doi.acm.org/10.1145/1015706.1015733`.

equations for the fluid pressure projection and the solid velocity. Batty *et al.* [9] further develop the idea of solving both fluid and solid object velocities simultaneously adopting a variational approach to solve both solid and fluid velocities in a single pressure update step.

Robinson-Mosher *et al.* [117] further enhances the approach in Batty *et al.* [9] by incorporating interaction between fluids and deformable objects and thin shells. Here the transfer of momentum between the solid object and the fluid is modeled explicitly. As thin objects do not conform to the grid, fluid mass and momentum is mapped to the solid surface. The conservation of momentum is enforced at the solid surface and the results are interpolated back to the grid. An example of the results obtained with this technique is shown in Figure 3.16.

Guendelman *et al.* [55] propose a method to simulate the interaction of liquids and infinitesimally thin objects such as cloth. When interpolating and calculating differentials, leakage is avoided by tracing a ray from a fluid location to the point of interest (*e.g.*, back-traced particle position for semi-Lagrangian advection), and determining whether both points are separated by a thin shell. If so, interpolation and differential expressions are modified to account for the solid presence. Pressure differences in the fluid are transferred to the solid locations to compute solid-fluid coupling.

Kwatra *et al.* [79] present a method to simulate swimming characters. Here a character is represented as an articulated body follows motion recorded in advance. In this method, the force that the character requires to match the pre-recorded motion is computed considering a two-way coupled liquid simulation.

Müller *et al.* [97] perform solid fluid coupling in an SPH simulation. The coupling of fluid particles and solids is simplified to particle coupling by creating boundary particles at the solid surface. Then the influence of the fluid on the solid is computed from the surface particles. Solenthaler *et al.* [128] adopt a different approach where both solids and fluids are modeled using SPH. Here elastic solids are modeled by incorporating an additional elasticity force to the particles in the solid. Modeling rigid solid behavior is done by computing all the forces acting on the solid and by restricting the motion only to translations and rotations. Akinci *et al.* [2] also employ boundary particles to compute relative contributions of these particles to account for irregular particle distributions. Here simulations are also enhanced by incorporating friction and drag at the solid surface.

Solid-fluid coupling has not been addressed in detail in vortex methods for computer animation. Despite the fact that vortex methods have been introduced early on in the development of fluid simulation for computer graphics, most researchers have focused

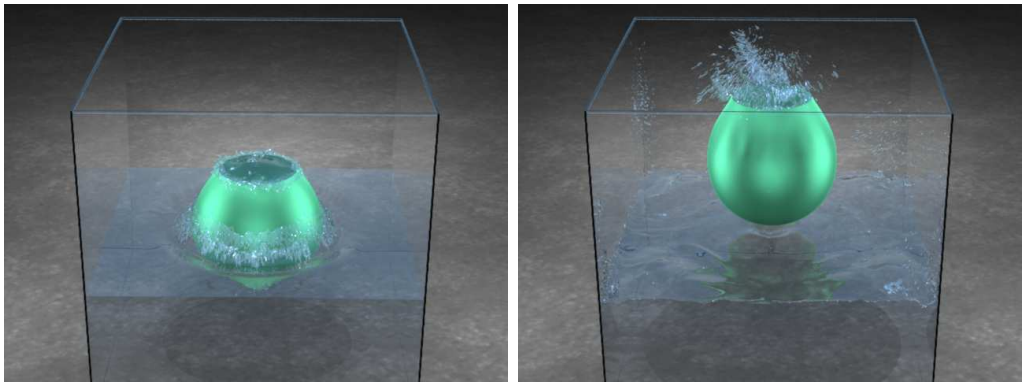


Figure 3.16: Example of coupling fluids and solids from Robinson-Mosher *et al.* [117]. A thin shell initially submerged is pulled out of water. As the bag raises it contracts and expands inducing additional motion in the contained liquid. Images from Robinson-Mosher, A., Shinar, T., Gretarsson, J., Su, J., and Fedkiw. "Two-way coupling of fluids to rigid and deformable solids and shells," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2008, Vol. 27:3, © 2008 ACM, Inc. `http://doi.acm.org/10.1145/1360612.1360645`.

their attention on simulating fluids using traditional grid methods. These provide a simple framework for computing differential quantities, which in turn are required to compute fluid forces acting on a solid surface. In the few implementations of vortex methods in computer animation only solid-to-fluid interaction is modeled. This is done by modeling the solid as a boundary condition for the flow as in Park and Kim [105], Angelidis *et al.* [5] and Weißmann and Pinkall [149]. Here the solid acts as an obstacle for the flow, but the flow does not induce a motion on the solid. Most of the development of solid-fluid interaction in vortical flows come from engineering applications.

3.3.2 Coupling of Solids and Vortical Flows in Engineering Applications

In engineering applications, the main focus is to perform numerically accurate calculations in computing drag forces on solid objects under different flow conditions. Wu [157] introduces a theory of the integral aerodynamic forces acting on a system of solid objects. Here the net force acting on a system of solid objects is related to the variation

of the first moment of vorticity of the surrounding flow. Shashikanth *et al.* [123] use this force model for representing the interaction of a single solid object interaction with an arbitrary amount of vortex rings as a Hamiltonian model. Wu *et al.* [159] use different formulations of force over objects to study the evolution of flow past solid objects.

Lighthill [84] establishes the relationship between the rate of creation of vorticity at a solid surface or *vorticity flux* with the pressure distribution at the surface. This theory has been developed by a number of researchers, most prominently Wu and Wu [156], who derive the vorticity for 3D solids by analyzing the stress tensor at the solid boundary.

The analysis of the vorticity flux at a solid surface has been applied in different works aimed to study the fluid forces acting on solids. Walther and Larssen [142] employ this method to simulate a 2D bluff body with prescribed motion submerged in a viscous flow. Calhoun [18], and Russell and Wang [118] apply this method to grid simulations.

Ploumhans *et al.* [110] use an integral force computation scheme and define a control volume around a solid object. Then the drag on the solid is computed considering the moment of vorticity and the flux of vorticity through the control surface. We note that defining an adequate control volume for the solid object may not be straightforward in the case of objects that move freely according to the flow and that may collide with each other.

Walther and Koumoutsakos [141] propose a method to model a flow with solid particles that are dragged with it. Here the flow is solved using a hybrid method, where vorticity dynamics are solved using Lagrangian particles on a background grid. This is known as vortex-in-cell method. Here flow properties from Lagrangian particles are mapped to grid locations, where velocity is solved. Then, this velocity is interpolated to particle locations.

Coquerelle and Cottet [31] use a vortex-in-cell method to model interaction of solids and fluids. Here the solid and fluid are considered a single flow where the interface is defined with a level set function. Rigid body motion is enforced by a penalty method where an additional term is added to the Navier-Stokes equations to enforce rigid motion.

3.3.3 Discussion on Solid-Fluid Coupling

It can be argued that *strong coupling*, as applied by Robinson-Mosher *et al.* [117], is the most physically accurate approach used in computer animation. Such an approach is suitable for high-quality simulations due to the computation of the momentum interchange between the fluid and the solid.

Our goal is to achieve two-way coupling of solids and fluids and we decide to develop a weak coupling approach. At each time step we solve the fluid using a splitting method and then we use the flow properties to solve the solid motion in an interleaved manner. This way, at each iteration, we simulate fluid motion, generate vorticity and then we calculate the corresponding forces exerted on each solid object using a direct formulation. Larssen [140] has employed a similar scheme in engineering applications for drag computation on solids with prescribed motion. We aim to adopt a similar approach for graphical applications. In terms of physical accuracy the requirements in computer graphics are less strict than in mechanical engineering and we can produce realistic simulations adequate for computer animation.

Lighthill [84] demonstrates the process of creation of vorticity and the pressure distribution in the solid surface are related. We employ this physical principle for modeling solid-fluid interaction in our framework. Modeling of turbulence and irregular flow past a solid object is a desirable feature in computer animation and it becomes natural to incorporate force computations to this process.

3.4 Additional Considerations

3.4.1 Rendering

In the previous sections we have discussed different methods to simulate fluids for computer animation. Once simulation data is obtained, either as an off-line process or in real time, a 2D visual representation of the simulation data and other objects in the simulation space is required to be displayed to the user. This process is known as *rendering*. This is achieved by modeling the behavior of light considering phenomena such as reflection and refraction. The Rendering Equation [69] relates the radiance L directed in a certain direction $\mathbf{e}$ from a surface point $\mathbf{x}$ to the emitted radiance L_e by the surface itself and the reflected incident radiance on $\mathbf{x}$. The Rendering Equation for time step t and light wavelength λ is:

$$L(\mathbf{x}, \mathbf{e}, \lambda, t) = L_e(\mathbf{x}, \mathbf{e}, \lambda, t) + \int_{\Omega} f(\mathbf{x}, \mathbf{e}', \mathbf{e}, \lambda, t) L_i(\mathbf{x}, \mathbf{e}', \lambda, t)(\mathbf{e}' \cdot \mathbf{n}) d\mathbf{e}'. \tag{3.20}$$

Here $\mathbf{e}$ is the direction from the point $\mathbf{x}$ to the observer, and $\mathbf{e}'$ represents the direction of incoming light to $\mathbf{x}$. Ω is the hemisphere where light is being reflected. The bi-directional reflectance distribution function f determines the distribution of reflected

light given the reflected direction $\mathbf{e}$ and the incident light direction $\mathbf{e}'$. L_i represents the amount of incident light on $\mathbf{x}$ from direction $\mathbf{e}'$ and $\mathbf{e}' \cdot \mathbf{n}$ is the cosine factor. In many cases in fluid simulation, it is required to model the behavior of light when traveling through a volume of fluid. Therefore it is necessary to model the extinction, scattering and absorption of luminous energy. The light behavior in this case is modeled based on radiative transfer. The following model presented by Raab *et al.* [114] describes the change of the amount of light at position $\mathbf{x}$ in direction $\mathbf{e}$:

$$\frac{\partial}{\partial \mathbf{e}} L(\mathbf{x}, \mathbf{e}) = L_e(\mathbf{x}, \mathbf{e}) - \sigma_t L(\mathbf{x}, \mathbf{e}) + \sigma_s \int_\Omega f_p(\mathbf{x}, \mathbf{e}', \mathbf{e}) L(\mathbf{x}, \mathbf{e}')(\mathbf{e}' \cdot \mathbf{x}) d\mathbf{e}'. \tag{3.21}$$

In this model, three main quantities are required: the scattering coefficient σ_s which is the rate of scattered light; the rate of extinction of light σ_t, and the phase function f_p which is the distribution of scattered light with respect to the different light directions. Another useful measurement that is often used is the light absorption coefficient, which corresponds to $\sigma_t - \sigma_s$.

Different solutions to determine the amount of light that reaches the observer from the simulation model objects have been proposed and they can be seen as ways to approximately solve the Rendering Equation. In this section we outline some of the most common techniques for rendering fluids.

Practical Fluid Rendering

Ray Casting

Ray Casting is a simple method to render a computational model into a 2D image. For each pixel in the 2D image, a ray is cast towards the objects in the scene. At the locations where the ray intersects an object, the surface color is evaluated [6]. This technique is also used for rendering volumes [83]. Here the start and end points of a ray are determined by the bounding box of the volume. As the ray enters the volume, the corresponding pixel color information is updated with voxel density data. As rays traverse a volumetric object, their trajectory may not coincide with the sampled density, usually at the center of a grid cell. In this case the values are interpolated to sampling locations along the ray trajectory and the color is computed from these values. Fraedrich *et al.* [49] sample particle simulation data onto a view aligned grid with adaptive sampling.

Ray casting has the advantage of being computationally inexpensive and it is used primarily on interactive applications such as video-games. For instance, Inacio *et al.* [66]

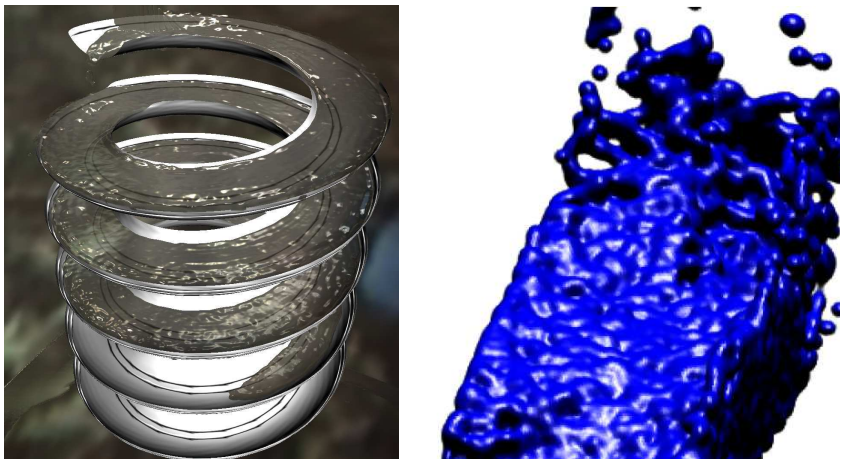


Figure 3.17: Details of rendering with Ray Casting. Left: Translucent volume rendering of liquid from Fraedrich *et al.* [49]. © 2010 IEEE. Reprinted, with permission, from Fraedrich, R., Auer S. and Westermann, R. "Efficient high-quality volume rendering of SPH data". IEEE Transactions on Visualization and Computer Graphics, Vol. 16:6. pages 1533–1540, 2010. Right: Liquid surface and splashes from Inacio *et al.* [66]. © 2010 IEEE. Reprinted, with permission, from Inacio, R. T., Nobriega T., Carvalho, D., and von Wangenheim, A. "Interactive Simulation and Visualization of Fluids with Surface Raycasting". Proceedings of the 2010 23rd SIBGRAPI Conference on Graphics, Patterns and Images, SIBGRAPI '10, pages 142–148, 2010.

use ray casting to render a fluid simulation with 8192 particles taking approximately 0.01087 seconds per update, which is approximately 91 updates per second on GPU. Results of rendered scenes using this technique are shown in Figure 3.17. One important disadvantage of this method is that it cannot properly model reflections of other objects as color is evaluated until it reaches an object.

Ray Tracing

This technique also consists on casting rays from the point of view to the simulation space. As opposed to ray casting, here complex trajectories of rays are simulated, allowing rays to be reflected on surfaces, or refracted as they travel through translucent objects. Rays are simulated until they reach a light source, or until a stopping criterion is satisfied, *e.g.*, a certain amount of collisions with objects. High quality results can be obtained

with *Monte Carlo Ray Tracing*, which employs rays to evaluate the Rendering Equation with Monte Carlo integration. Here a ray that is reflected from a surface or enters a scattering medium will change its trajectory randomly. In importance sampling the probability of a change is determined by a phase function. Nguyen *et al.* [101] use Monte Carlo techniques applying recursive random sampling on the rays traversing the media.

Frisvad *et al.* [50] successfully applies this method for high quality rendering of participating media as shown in Figure 3.18. Foster and Fedkiw [45] also use ray tracing to render liquids. Harada *et al.* [57] use ray tracing for rendering liquids modeled using SPH. Ray tracing can also be used for rendering bubbles, as in Hong *et al.* [62].

Ray tracing can be extended to model different materials. For instance, Carlson *et al.* [20] employ subsurface scattering for modeling melting solids. Subsurface scattering occurs where the light penetrates the surface and scatters, exiting the surface at a different point.

Fedkiw *et al.* [41] and Enright *et al.* [39] use *Photon Mapping* [68] for global illumination. Light rays are traced from source in the scene carrying *photons*, which interact with the objects in the scene. Upon colliding with an object a photon is reflected, transmitted or absorbed. If the photon is reflected, for instance by colliding with a mirror, its trajectory is changed and possibly its intensity scaled depending on the reflective properties of the surface. If the photon is absorbed, then it is discarded. If the photon hits a diffuse surface, this information is stored in a *photon map* and the photon is reflected. The photon map is used in a second pass of traditional ray-tracing to compute the amount of light or radiance of each pixel. Results obtained with this technique are shown in Figure 3.18.

Simulating complex ray trajectories is a costly operation. For instance Enright *et al.* [39] report that rendering the glass example in Figure 3.7 (a), takes about 15 minutes per frame. More recently optimized ray tracing methods run in real-time [56].

Surface Reconstruction

Kang *et al.* [70] render liquid surfaces and interfaces between immiscible fluids by constructing a polygonal surface from volumetric data or level sets. A popular method to reconstruct these surfaces is the marching cubes algorithm by Lorensen and Cline [87]. Here the volume is traversed considering a cube of eight voxels at a time. If for some cubes the distance field is greater than the iso-value, then these values define the surface and a polygonal patch is added. The polygonal patches are pre-computed considering all the possible combinations of cubes inside and outside the volume. As the volume

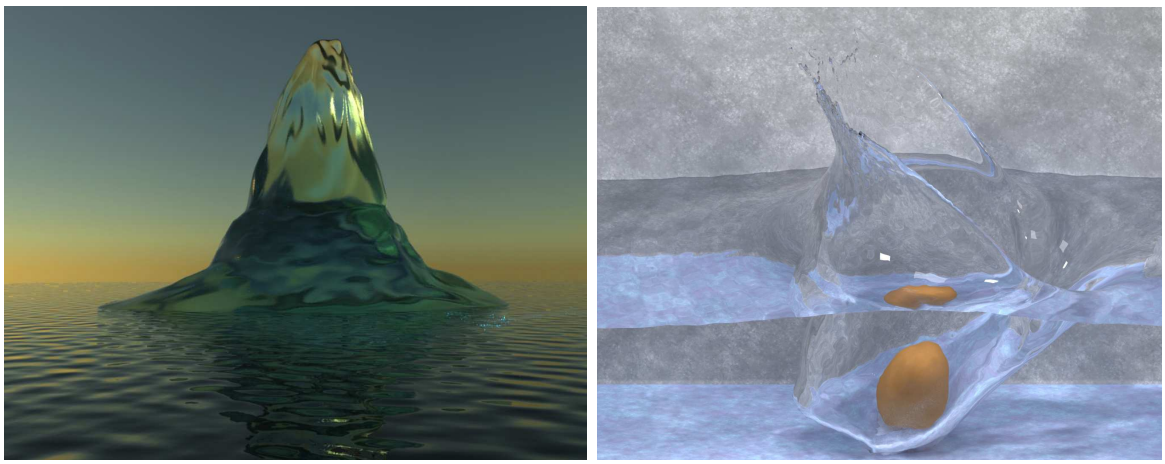


Figure 3.18: Left: Ray tracing results for ice floating over water from Frisvad *et al.* [50]. Frisvad, J. R., Christensen, N. J. and Jensen, H. W. "Computing the scattering properties of participating media using Lorenz-Mie theory," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2007, Vol 26:3 © 2007 ACM, Inc. `http://doi.acm.org/10.1145/1276377.1276452`. Right: Ray tracing results with photon mapping from Enright D., Marschner S., Fedkiw R. "Animation and rendering of complex water surfaces," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2002, Vol 21:3 © 2002 ACM, Inc. `http://doi.acm.org/10.1145/566654.566645`.

is traversed, patches are connected defining a polygonal surface enclosing the volume. Beaudoin *et al.* [12] approximate a polygonal surface from volumetric data to model visual layers of fire. Surface reconstruction from volumetric data has not only been applied to fluids, but also to other applications such as 3D surface reconstruction from medical density volumetric data, such as Magnetic Resonance Imaging (MRI).

Yu and Turk [163] reconstruct surfaces from particle-based fluid simulations. Here smooth surfaces are obtained by representing particles with anisotropic smoothing kernels, where the anisotropy scale is computed from the distribution of nearby particles and the analysis of its covariance tensor. A smooth surface is then reconstructed from the iso-surface of a density field determined by these kernels.

The main drawback of this method is its computational cost. Müller *et al.* [96] reports that a particle simulation running at 20 updates per second achieves only 5 frames per second by obtaining a surface using marching cubes on a Pentium IV 1.8 GHz processor.

Slicing

Current graphics processing hardware has the capability to store and manipulate 3D textures. These correspond to a cube of volumetric data. Camera-aligned slices are obtained from this cube by intersecting planes with the volumetric data. The slices are then drawn in back to front order so any non-transparent object occludes only what is behind it. Increasing the amount of slices increases also the resolution. This kind of rendering is available in commodity graphic cards. Fuller *et al.* [51] applies this technique for real-time fire simulation, achieving up to 141 updates per second rendering an unit-radius fire with a slice space of 0.2 on an nVidia GeForce 7800 GT graphics card. Examples of the results obtained with this technique are shown in 3.12 (c). Horvath and Geiger [64] use a similar technique to render high resolution particle simulation for fire.

Texture Mapping and Splatting

In certain applications, the cost of ray-tracing can be prohibitive, but still a relatively high quality rendering is required. Wang *et al.* [143] propose rendering a water surface using a texture map. Here the water surface color is stored in a 2D texture image. This image is mapped to the surface of an object to reproduce its aspect instead of computing it from its physical properties. He *et al.* [59] employs a similar approach. Here texture color is identified by coordinates and assigned to an ocean surface. texture coordinates are modified to render the ocean at different distances from the viewer. An example of

Figure 3.19: Left: Detail of ocean waves from He *et al.* [59] using texture mapping. © 2005 IEEE. Reprinted, with permission, from Huaiqing He, H., Liu H., Zeng, F. and Yang, G. "A way to real-time ocean wave simulation". International Conference on Computer Graphics, Imaging and Vision: New Trends, 2005. pages 409–415. Right: Detail of flames rendered from Wei *et al.* [145] using texture splats. © 2002 IEEE. Reprinted, with permission, from Wei, X., and Li, W., and Mueller, K., and Kaufman, A. "Simulating fire with texture splats". Proceedings of the conference on Visualization '02 pages 227–235.

the results obtained with this technique is shown in Figure 3.19.

Another technique to use textures is splatting. Here the contribution of each simulation element to the final image is computed by projecting voxels into the 2D image forming a footprint of the voxel which is then blended with other voxel footprints. [145] create from simulation or real fire images, a series of small textures with turbulence details, or splats. Transparency of each splat is computed based on a black body radiation distribution: brightness decreases as the distance to the fire center increases. Splats that overlap are combined using a smoothing function. Real-time performance is achieved in a GPU implementation. Müller *et al.* [96] also use splatting where surface particles from the simulation are used. A 2200 particle simulation running in a Pentium IV 1.8 GHz processor runs at 20 updates per second by rendering using splatting, five times faster than using surface reconstruction using marching cubes where performance was of 5 updates per second as discussed in Section 3.4.1.

3.4.2 Parallel flow computation

Parallelization has been used to simulate large scale simulations in computer animation, mostly in the film industry, by performing computations on farms of workstations. The high cost of this kind of implementation makes it impractical for most applications.

With increasingly available multi-core CPUs, one possibility to speed-up fluid simulations is to take advantage of this feature, distributing computation load on each core. Ihmsen *et al.* [65] adopt this approach by employing optimized data structures that allow efficient simulations with 12 million particles.

A low cost alternative is to transfer computations to the GPU, due to their increasing power even in commodity hardware, reducing the overhead from the CPU. Operations in GPU are efficiently processed in parallel, for instance by processing simultaneously several pixels in one pass. The capabilities of the GPU have led researchers to transfer costly computations to the GPU.

Early approaches for physical simulation on the GPU used graphics primitives, particularly textures. A texture is a 2D bitmap image. In traditional graphics applications, a texture is mapped to an object to give it a specific appearance. The color information in the texture is numerical and can be used as an array to store simulation data as proposed in Wu, Liu and Liu [155]. GPU programming supports 32 bits floating point precision, allowing to take advantage of the parallel processing pipeline of the GPU for general computations.

A typical GPU program used to simulate fluids receives a texture storing numerical simulation data. Each pixel corresponds to a grid cell in the simulation. The GPU processes each pixel in parallel and another texture is produced as output. This is directly applied to solve flow velocity using the splitting method proposed in Stam [129], where the result of each simulation step is stored in a new texture. Passively advected quantities like smoke density or temperature are stored in textures as well. Liu *et al.* [85] and Crane *et al.* [33] solve 3D simulations using a 3D texture, which stores volumetric data instead of pixel information. Wei *et al.* [147] stores each slice of a 3D texture as a 2D texture and process the flow using a set of 2D textures. Particle simulations can also be performed on a GPU by storing the particles positions and velocities in floating point textures.

Numerical methods can be implemented in the GPU through a piece of code called *fragment shader*. This program is loaded in the GPU and defines the changes of color in a texture pixel, which in the case of simulation, encodes velocity, density or pressure values.

Table 3.5: Performance comparison of GPU based simulations.

Approach	**Hardware**	**Simulation Nodes**	**Updates per Second**
Wu *et al.* [155] CPU simulation[a]	Pentium 2.8GHz CPU	512^2 grid	1.39
Wu *et al.* [155] GPU simulation[a]	GeForce FX5950 Ultra GPU		20
Wei *et al.* [147] CPU simulation	2.53 GHz Intel Pentium 4 CPU.	80x40x40 grid	2.8
Wei *et al.* [147] GPU Simulation	nVidia GeForce FX5900 Ultra GPU.		11.5

[a]Simulation without diffusion step.

Wu *et al.* [155] and Liu *et al.* [85] employ this technique for simulating smoke; Harris *et al.* [58] for simulating clouds, and Yang *et al.* [160] for fast fluid force computations on solids.

One simulation technique that has been successfully implemented in GPU is the Lattice Boltzmann method, which is discussed in Section 3.1.2. The local nature of the computations in each grid cell allows taking advantage of the parallel capabilities of the GPU. A performance comparison of simulations in GPU is presented in Table 3.5.

The only limitation of this kind of parallelization improvement is the simulation size, as memory in graphics processing units is more limited compared to main memory. As new technology is developed, this constraint may not be a problem for real-time applications in the near future.

GPUs can also be employed for general computation (GPGPU) using different frameworks, *e.g.*, the Compute Unified Device Architecture (CUDA) or OpenCL. CUDA implements programming structures that are similar to traditional languages such as Fortran and C/C++, and provides easy data transfer between the CPU and the GPU, while employing the full computational power of the GPU.

Amador and Gomes [3] employ CUDA grid simulations; Zhang *et al.* [166] employ multiple processing GPUs for SPH simulations, and Chentanez and Müller [25] implement hybrid grid and particle simulations using this framework. A summary of performance results using CUDA GPU programming approaches is presented in Table 3.6.

Table 3.6: Performance comparison of CUDA based simulations.

Approach	**Hardware**	**Simulation Nodes**	**Seconds per Update**
Amador and Gomes CPU simulation [3]	Intel Core2 Quad CPU Q6600@2.40GHz	128^3 grid	1.611
Amador and Gomes GPU simulation [3]	NVIDIA GeForce 8800 GT Q6600@2.40GHz		0.164
Chentanez and Müller CPU simulation [25]	Intel Core i7, 2.67 GHz	900x135 grid 250K particles	0.0945
Chentanez and Müller GPU simulation [25]	NVIDIA GTX480		0.018

3.4.3 Discussion on Computer Graphics Specific Techniques

High quality rendering of gases such as smoke can be achieved in a number of ways: either using volumetric Ray Tracing which approximates the natural behavior of light, or by simple Ray Casting which allows obtaining a lower quality but efficient rendering. In our case of gas simulation we employ passive tracker particles to determine the location of the gas. These particles define a volume which is then rendered using Ray Tracing. We employ Exocortex' high performance plug-in Fury2™ for Softimage™ to render our scenes.

We have implemented our simulator to run on the CPU, employing high-performance multi-threaded data structures for improved simulation times. We leave the implementation of our approach in parallel GPU code as a future implementation improvement.

Chapter 4

Inviscid Flow Model

In this chapter we present our computational flow model. We focus on the simulation of a homogeneous and incompressible fluid, *i.e.*, the motion of a gas at low speed where the effects of temperature are negligible. This is a common assumption for simulating smoke in computer graphics and has been used by several authors for simulating fluids [41, 5, 149]. Moreover, this assumption is also used in engineering applications, *e.g.*, aerodynamics, as it allows analyzing the behavior of flows with very low viscosity [48]. Thererfore, we assume viscosity is negligible and we treat the flow as inviscid.

Gravity is the only external force we consider, which is an irrotational vector field. We describe the method of solution of the vorticity transport equation (repeated here for the reader's convenience):

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = -(\mathbf{u} \cdot \nabla)\boldsymbol{\omega} + (\boldsymbol{\omega} \cdot \nabla)\mathbf{u}. \tag{4.1}$$

We first discuss our computational representation of vorticity. Then we address the issue of solving the above equation in our framework. We solve the evolution of vorticity at each time step using a *splitting* approach [129]. We formulate and solve a set of simpler equations, each corresponding to a term of the vorticity transport equation.

Then we present our methods to solve each of the required terms. Here we introduce novel methods to stably solve the second term in the right hand side of the vorticity equation, known as *vortex stretching*.

In this chapter we focus on an unbounded domain and we show examples of simulations of smoke to demonstrate the visual quality we achieve with our techniques. We address the fluid interaction with solid boundaries in Chapters 5 and 6.

4.1 Vorticity Representation

We assume that vorticity is concentrated on discrete particles [105], or simply *vortices*. Each vortex is identified by its position $\mathbf{z}_i$, and its vorticity $\boldsymbol{\omega}_i$, which is a vector defining a spin axis and a magnitude. As opposed to previous approaches for vortex simulation, where vorticity is spread into material elements such as vortex blobs [105] or vortex tubes [149], our vortex elements are singular, *i.e.*, they concentrate vorticity at a point in space. The notion of *circulation* is often associated with vortex elements. This corresponds to the integral of the velocity around a closed loop $\mathcal{L}$, *i.e.*,

$$\Gamma = \oint_{\mathcal{L}} \mathbf{u} \cdot d\mathbf{l}. \tag{4.2}$$

For a singular vortex, its *circulation* Γ is defined simply as $\Gamma_j = \|\boldsymbol{\omega}_j\|$. The *vorticity field* $\boldsymbol{\omega}$ of the flow is defined from the vorticity of each vortex particle as follows:

$$\boldsymbol{\omega}(\mathbf{x}) = \sum_j \delta(\mathbf{x} - \mathbf{z}_j)\boldsymbol{\omega}_j. \tag{4.3}$$

Here δ is the Dirac delta function. Note that the vorticity field is different from zero only at particle locations.

Given a vorticity field in a 3D flow, solving the vorticity transport equation requires computing the velocity field as well. In the following section, we describe how we calculate the flow velocity given a vorticity field using our representation.

The velocity field induced by a vorticity field $\boldsymbol{\omega}$ on a fluid region is defined by the solution of a Poisson problem determined by the Continuity Equation (2.4) and the vorticity definition in Equation (2.6), the Biot-Savart Law for an arbitrary vorticity field which is:

$$\mathbf{u}(\mathbf{x}) = \frac{1}{4\pi} \int_{Fluid} \boldsymbol{\omega}(\mathbf{z}) \times \frac{\mathbf{x} - \mathbf{z}}{\|\mathbf{x} - \mathbf{z}\|^3} d\mathbf{z}. \tag{4.4}$$

Assuming the vorticity is concentrated only on discrete vortices, the Biot-Savart Law is evaluated as the sum of the contributions of each vortex j as follows:

$$\mathbf{u}(\mathbf{x}) = \frac{1}{4\pi} \sum_j \boldsymbol{\omega}_j \times \frac{\mathbf{x} - \mathbf{z}_j}{\|\mathbf{x} - \mathbf{z}_j\|^3}. \tag{4.5}$$

Solving the integral over the space occupied by the fluid in the Biot-Savart Law is then reduced to summing up the contributions of each individual vortex. Two issues arise in the evaluation of the above equation. The first issue is that the terms in the summation are singular at vortex locations $\mathbf{z}_j$, therefore the flow velocity is undefined at vortex locations. The second issue is that evaluating the velocity field at vortex locations is $O(n^2)$ in the number of vortices n. This evaluation is necessary to solve the vorticity transport equation. We address the first problem by employing a regularized function instead of the the Biot-Savart formula. We deal with the second problem by limiting the influence radius of each vortex, which also allows enhanced control over the simulation. Our solution is described in detail in the following section.

4.2 Velocity Evaluation with Finite Regions of Influence

Instead of evaluating the Biot-Savart formula for calculating velocity, we adopt a different strategy: we employ a non-singular function with finite influence radius, similar to the approach in Angelidis *et al.* [5] for simulating vortex filaments. In their approach, such a kernel is used to obtain simple closed form formulas for integration along 3D curves [4]. We calculate the velocity field through a formula of the form:

$$\mathbf{u}_j(\mathbf{x}) = (\boldsymbol{\omega}_j \times (\mathbf{x} - \mathbf{z}_j))\, w(\|\mathbf{x} - \mathbf{z}_j\|). \tag{4.6}$$

The difference beween the definition in Equation (4.6) and the Biot-Savart formula is the kernel w which changes the magnitude of the velocity field. The choice of function does not affect the velocity field direction and therefore it preserves its main properties with respect to the Biot-Savart Law. Moreover, this velocity definition also facilitates integration for kinetic energy calculations (see Section 4.3.3), while producing visually similar simulation results, as shown in Figure 4.3.

The velocity field must be divergence-free, otherwise the condition of mass conservation does not hold. The following result characterizes functions that can be used as kernels that satisfy this property and we present its proof in Appendix B:

Proposition 1 *For any continuous and differentiable radial basis function* $w : \mathbb{R} \to \mathbb{R}$ *and vectors* $\boldsymbol{\omega}$, $\mathbf{x}$ *and* $\mathbf{z} \in \mathbb{R}^3$, *the following holds:* $\nabla \cdot [(\boldsymbol{\omega} \times (\mathbf{x} - \mathbf{z}))\, w(\|\mathbf{x} - \mathbf{z}\|)] = 0$.

A variety of functions may be chosen as a kernel obtaining visually plausible results. A particular kernel function can be chosen based on different criteria, such as being

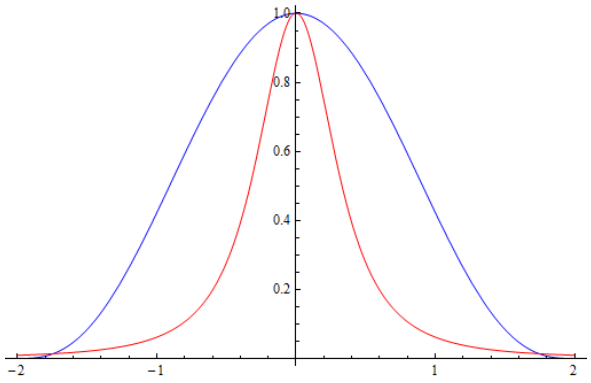


Figure 4.1: Comparison of vorticity magnitude produced by the Rosenhead-Moore kernel whose maximum is set to 1 (red) and our kernel with radius $\epsilon = 2$ (blue).

efficient to compute, or having simple closed form integral expressions over space. We demonstrate our techniques using the following polynomial kernel function with radius $\epsilon > 0$:

$$w(r) = \begin{cases} \left(1 - \frac{r^2}{\epsilon^2}\right)^3 & \text{if} \quad |r| < \epsilon \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}$$

By Proposition 1 we have that $\nabla \cdot \mathbf{u}_j = 0$, where $\mathbf{u}_j$ is defined in Equation (4.6). Also, given a collection of vortices, we have that the velocity field they induce is also divergence-free, *i.e.*, $\nabla \cdot \mathbf{u}(\mathbf{x}) = 0$, where:

$$\mathbf{u}(\mathbf{x}) = \sum_j \left(\boldsymbol{\omega}_j \times (\mathbf{x} - \mathbf{z}_j)\right) w(\|\mathbf{x} - \mathbf{z}_j\|). \tag{4.8}$$

Using the kernel in Equation (4.7) reduces the cost of computing velocity from $O(n^2)$ to $O(nk)$ where k is the number of vortices within the radius ϵ of each particle. We evaluate Equation (4.8) through a nearest neighbor search using a KD-tree structure.

We choose the kernel function in Equation (4.7) as its integral has a simple closed form which facilitates computing the kinetic energy of a vortex (See Section 4.3.3). By Proposition 1 a variety of functions can be used as well and other kernels may be chosen due to their specific properties. Other kernel choices and their properties are dicussed in Section 7.4.2.

We acknowledge that our kernel produces a different velocity field from that defined by

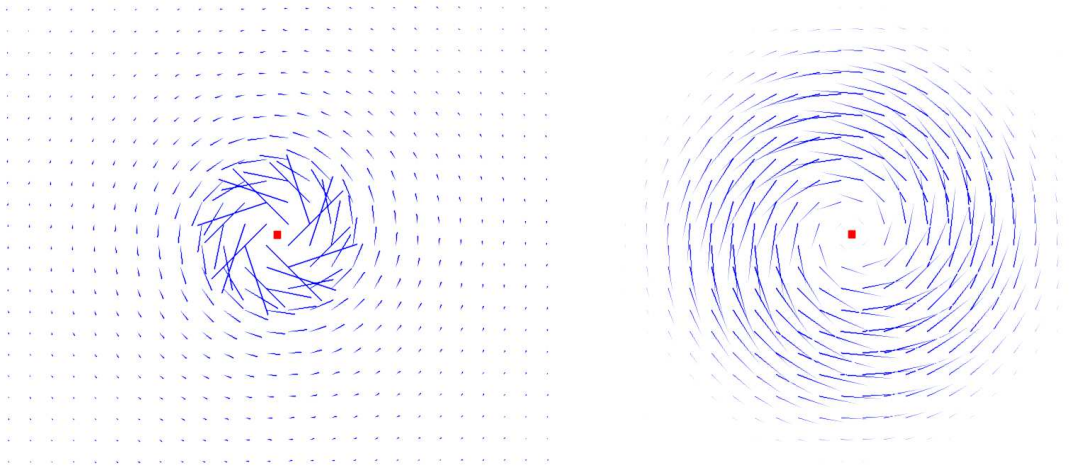


Figure 4.2: Comparison of velocity field produced by the Rosenhead-Moore kernel (left) and our kernel with radius $\epsilon = 2$ (right). The vortex location is highlighted in red.

the Biot-Savart Law, and some of its common approximations, such as the the Rosenhead-Moore Kernel [77, 108]. The Rosenhead-Moore kernel is defined as follows:

$$K(\|\mathbf{x} - \mathbf{z}\|) = \frac{1}{4\pi} \frac{1}{(\|\mathbf{x} - \mathbf{z}\|^2 + a^2)^{3/2}}. \tag{4.9}$$

Here $a > 0$ is a smoothing constant that removes the singularity in the Biot-Savart kernel. This is equivalent to represent the vortices as fininte radius blobs instead as singular particles, *i.e.*, the vorticity is no longer concentrated in a singular point in space, but in a finite volume [148].

A comparison of the velocity field magnitude produced by the Rosenhead-Moore Kernel and our kernel is presented in Figure 4.1. A comparison of the velocity fields generated by each is presented in Figure 4.2

We note that our choice of kernel does not provide an exact solution to the Poisson problem determined by the Continuity Equation (2.4) and the vorticity definition in Equation (2.6). However as demonstrated in our work, this simple kernel function allows us to create rich visual simulations. A comparison of results obtained simulating a smoke column using our velocity definition and an approximation to the Biot-Savart formula known as the Rosenhead-Moore kernel is shown in Figure 4.3.

Evaluating the velocity field using finite kernels is an effective way to reduce the complexity of evaluating vortex velocity fields and allowing a scene designer to further

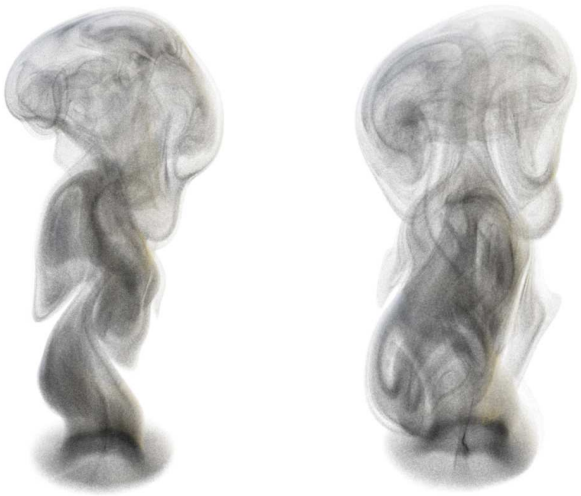


Figure 4.3: Slow rising wispy smoke with an emission radius of 1.0, simulated with vortices using the infinite radius Rosenhead-Moore kernel (left), and our finite kernel with radius $\epsilon = 2.5$ (right). An animation sequence for this figure is shown in the accompanying video at 0:40.

control the visual effect of a vortex simulation. Finite kernels can also be optimized using the Fast Multipole Method, although this is outside of the scope of our present work.

Once the velocity field induced by the vortices is known, it is possible to solve the vorticity transport equation. This is done by sequentially solving each of its terms, in a process known as *splitting* [129]. In the next section we discuss in detail the equations that need to be solved and how these solutions are combined.

4.3 Solution of the Vorticity Transport Equation

Solving for vorticity at each time step is done by performing a time integration step on the Vorticity Transport Equation. This could be done via direct numerical simulation for high-accuracy results [110], however at a rather high computational cost. Instead we solve the vorticity transport equation by splitting it into each of its terms. This defines a set of equations that are solved sequentially, *i.e.*, the solution of the first equation is used as input for the second. The equations we solve are:

$$\frac{\partial \boldsymbol{\omega}_1}{\partial t} = -(\mathbf{u} \cdot \nabla)\boldsymbol{\omega}, \tag{4.10}$$

$$\frac{\partial \boldsymbol{\omega}_2}{\partial t} = (\boldsymbol{\omega}_1 \cdot \nabla)\mathbf{u}_1. \tag{4.11}$$

This is a first-order accurate splitting scheme for solving the Vorticity Transport Equation. Equation (4.10) corresponds to a vorticity advection equation and it represents the transport of vorticity due to the flow. Equation (4.11) represents the change in vorticity magnitude and direction as the vorticity is transported, and it corresponds to the *vortex stretching* term.

Given a vorticity field $\boldsymbol{\omega}$ at a time t, solving the vorticity transport equation for $t+\Delta t$ in an unbounded flow is performed through the following steps:

1. Find the velocity field $\mathbf{u}$ induced by $\boldsymbol{\omega}$ using Equation (4.8).
2. Solve the advection term in Equation (4.10) using $\mathbf{u}$ and $\boldsymbol{\omega}$ as inputs, to obtain an intermediate vorticity field $\boldsymbol{\omega}_1$.
3. Find the velocity field $\mathbf{u}_1$ induced by $\boldsymbol{\omega}_1$ using Equation (4.8).
4. Solve the vortex stretching term in Equation (4.11) using $\mathbf{u}_1$ and $\boldsymbol{\omega}_1$ as inputs, to obtain the final vorticity field $\boldsymbol{\omega}_2$ which corresponds to the vorticity field at time $t + \Delta t$.

We have already discussed in Section 4.2 how to solve steps (1) and (3). In the next sections we detail how to solve steps (2) and (4).

4.3.1 Vortex Advection

Solving the vortex advection term corresponds to solving the following equivalent form of Equation (4.10):

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega} = 0. \tag{4.12}$$

The above equation corresponds to the total derivative of the vorticity field, since the following holds (*cf.* Section 2.1):

$$\begin{aligned}\frac{d\boldsymbol{\omega}}{dt} &= \frac{\partial\boldsymbol{\omega}}{\partial t} + \frac{\partial\boldsymbol{\omega}}{\partial x_1}\frac{dx_1}{dt} + \frac{\partial\boldsymbol{\omega}}{\partial x_2}\frac{dx_2}{dt} + \frac{\partial\boldsymbol{\omega}}{\partial x_3}\frac{dx_3}{dt} \\ &= \frac{\partial\boldsymbol{\omega}}{\partial t} + \frac{\partial\boldsymbol{\omega}}{\partial x_1}\mathbf{u}_1 + \frac{\partial\boldsymbol{\omega}}{\partial x_2}\mathbf{u}_2 + \frac{\partial\boldsymbol{\omega}}{\partial x_3}\mathbf{u}_3 \\ &= \frac{\partial\boldsymbol{\omega}}{\partial t} + (\mathbf{u}\cdot\nabla)\boldsymbol{\omega}.\end{aligned}$$

This term describes how the vorticity field is carried by the flow. In particle systems where physical quantities are encoded in particles, the transport of these quantities is realized by simply advancing the particles with the underlying velocity field. Therefore the motion of vorticity through the flow is realized directly by advancing the vortex particles.

In our framework, particle positions are determined by the following equations:

$$\frac{d\mathbf{z}_i}{dt} = \mathbf{u}(\mathbf{z}_i) \tag{4.13}$$

$$\mathbf{u}(\mathbf{z}_i) = \sum_j \left(\boldsymbol{\omega}_j \times (\mathbf{x}-\mathbf{z}_j)\right) w(\|\mathbf{x}-\mathbf{z}_j\|). \tag{4.14}$$

Then vorticity advection is solved by first evaluating the velocity field at the vortices' positions. Then this velocity is used to update each vortex position in the space. We do this using second order Runge-Kutta integration since this produces the visual results we aim for, without the extra overhead of higher order methods. In our example simulations we employ a fixed time step.

4.3.2 Vortex Stretching

To solve the vortex stretching term, a solution of the following equation has to be found:

$$\frac{\partial\boldsymbol{\omega}}{\partial t} = (\boldsymbol{\omega}\cdot\nabla)\mathbf{u} \equiv \nabla\mathbf{u}\cdot\boldsymbol{\omega}, \tag{4.15}$$

where the gradient of velocity $\nabla\mathbf{u}$ is a tensor of rank two that corresponds to the transpose of the Jacobian matrix of the velocity field $\mathbf{u}$. The vortex stretching term represents a change in orientation and magnitude of vorticity. Evaluation of this term may lead to instability due to an exponential increase in the vorticity magnitude [121].

We note that the gradient of velocity can be represented as follows:

$$\nabla \mathbf{u} = \frac{1}{2}\left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right) + \frac{1}{2}\left(\nabla \mathbf{u} - (\nabla \mathbf{u})^T\right). \tag{4.16}$$

First we focus on the second term on the right-hand side of the above equation. The following is the matrix representation of this term:

$$\left(\nabla \mathbf{u} - (\nabla \mathbf{u})^T\right) = \begin{pmatrix} 0 & \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2} & \frac{\partial u_3}{\partial x_1} - \frac{\partial u_1}{\partial x_3} \\ \frac{\partial u_1}{\partial x_2} - \frac{\partial u_2}{\partial x_1} & 0 & \frac{\partial u_3}{\partial x_2} - \frac{\partial u_2}{\partial x_3} \\ \frac{\partial u_1}{\partial x_3} - \frac{\partial u_3}{\partial x_1} & \frac{\partial u_2}{\partial x_3} - \frac{\partial u_3}{\partial x_2} & 0 \end{pmatrix}. \tag{4.17}$$

For an arbitrary vector $\mathbf{b}$, the following holds:

$$\frac{1}{2}\left(\nabla \mathbf{u} - (\nabla \mathbf{u})^T\right) \cdot \mathbf{b} = \frac{1}{2}\boldsymbol{\omega} \times \mathbf{b}. \tag{4.18}$$

Then the vortex stretching term is recast as follows:

$$\begin{aligned} \nabla \mathbf{u} \cdot \boldsymbol{\omega} &= \frac{1}{2}\left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right) \cdot \boldsymbol{\omega} + \frac{1}{2}\boldsymbol{\omega} \times \boldsymbol{\omega} \\ &= \frac{1}{2}\left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right) \cdot \boldsymbol{\omega}. \end{aligned} \tag{4.19}$$

The matrix $\left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right)$ corresponds to the strain tensor which is symmetric and it has the following eigen-decomposition:

$$\left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right) = \mathbf{Q}\mathbf{D}\mathbf{Q}^T, \tag{4.20}$$

where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{D}$ is a diagonal matrix. By defining $\mathbf{Q} = det(\mathbf{Q})\mathbf{Q}'$, the above equation can be rewritten as follows:

$$\nabla \mathbf{u} = \frac{1}{2}\mathbf{Q}'\mathbf{D}'\left(\mathbf{Q}'\right)^T, \tag{4.21}$$

where the $det(\mathbf{Q}') = 1$ and $\mathbf{D}' = det^2(\mathbf{Q})\mathbf{D}$. Since the columns of $\mathbf{Q}'$ are orthogonal and $det(\mathbf{Q}') = 1$, $\mathbf{Q}'$ corresponds to a rotation matrix. Notice also that $\mathbf{D}'$ is a scaling matrix. Then, solving vortex stretching amounts to computing a rotation and scaling matrix and applying them to the vorticity $\boldsymbol{\omega}$. Since the vorticity magnitude may increase exponentially [121], we focus on obtaining a suitable rotation matrix to change the vorticity spin direction. The direct evaluation of $\nabla \mathbf{u}$ using singular particles may

produce a non-symmetric matrix, *i.e.*, containing shearing as well. This would not be a major issue when the main simulation is driven by an underlying flow as in the work by Selle *et al.* [121], where vorticity magnitude is rescaled to avoid instability. However, in vorticity-driven flows, as many of the examples in this thesis, shearing in the gradient of velocity can distort the velocity field. Instead of rescaling vorticity, we present two novel and stable ways to derive rotation matrices for vorticity that produce visually appealing results.

Gradient estimation vortex stretching

A direct way to estimate the change in spin axis of each vortex is to estimate the strain tensor. The following is a simple way to achieve this.

For each vortex j we define a local orthonormal coordinate system $\mathbf{E}_j = \{\mathbf{e}_{j,1}, \mathbf{e}_{j,2}, \mathbf{e}_{j,3}\}$. Then we sample the flow velocity $\mathbf{u}$ at a fixed distance d from the vortex position $\mathbf{z}_j$ along each axis in $\mathbf{E}_j$. We define the following velocity differentials:

$$D\mathbf{u}_{j,k} = \frac{\mathbf{u}(\mathbf{z}_j + d\,\mathbf{e}_{j,k}) - \mathbf{u}(\mathbf{z}_j - d\,\mathbf{e}_{j,k})}{2d}. \tag{4.22}$$

Here $k \in \{1, 2, 3\}$ Then we can estimate the gradient of velocity $\nabla\mathbf{u}$ at the the vortex position $\mathbf{x}_j$ as follows:

$$\nabla\mathbf{u}_j \approx \left[D\mathbf{u}_{j,1} \quad D\mathbf{u}_{j,2} \quad D\mathbf{u}_{j,3}\right]. \tag{4.23}$$

By assuming the vorticity is concentrated in singular points in space, $\nabla\mathbf{u}_j$ may not be symmetric. We decompose $\nabla\mathbf{u}_j$ into rotation, shear and scaling matrices using a polar decomposition. Let $\mathbf{Q}_j$ be the rotation component of $\nabla\mathbf{u}_j$. Then at each time step, the vorticity update for vortex j is the following:

$$\frac{\partial\boldsymbol{\omega}_j}{\partial t} = \mathbf{Q}_j\boldsymbol{\omega}_j. \tag{4.24}$$

This is a stable vorticity update, and we show simulation examples using this method in Figure 4.5.

Advection-driven vortex stretching

A second way to estimate the rotation of each vortex's vorticity vector is by estimating the rotation of the vortex's frame of reference in a single advection step. For each vortex

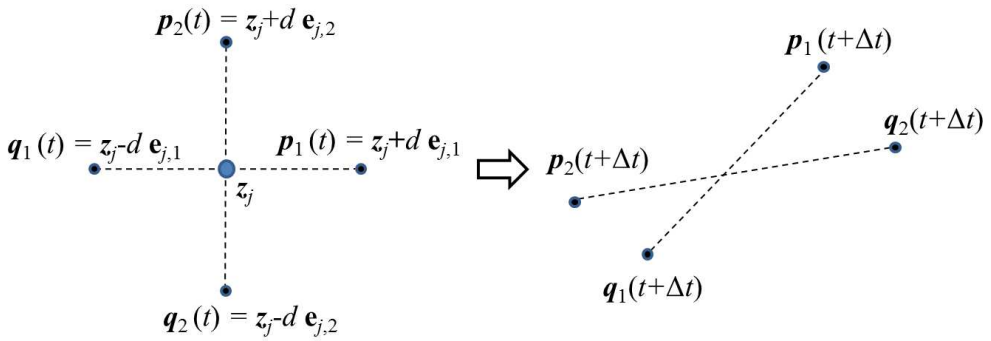


Figure 4.4: Left: sample points for strain tensor estimation. Right: Sample points after advection.

j we generate a set of marker particles, which are initially located at the same velocity sample positions as above. We define $\mathbf{p}_{j,k} = \mathbf{z}_j + d\,\mathbf{e}_{j,k}$ and $\mathbf{q}_{j,k} = \mathbf{z}_j - d\,\mathbf{e}_{j,k}$ as shown in Figure 4.4. Then we advect marker particles at positions $\mathbf{p}$ and $\mathbf{q}$. From these advected positions we obtain an advected reference frame given by the vectors:

$$\mathbf{b}_{j,k} = \mathbf{p}_{j,k}(t + \Delta t) - \mathbf{q}_{j,k}(t + \Delta t). \tag{4.25}$$

These advected axes form a new basis $\mathbf{B}_j$, from which we obtain a rotation matrix $\mathbf{R}_j$ using again a polar decomposition. Note that $\mathbf{R}_j$ is an orthogonalized basis for the vortex's advected frame of reference. Then at each time step we update vorticity for each vortex particle j as follows:

$$\frac{\partial \boldsymbol{\omega}_j}{\partial t} = \mathbf{R}_j \boldsymbol{\omega}_j. \tag{4.26}$$

This method also yields rich and detailed visual simulations of smoke, as shown in Figure 4.5. We employ this advection-driven vortex stretching approach in the simulation examples presented throughout this work.

As opposed to previously published methods [105], we do not employ viscosity to avoid numerical instability when solving vortex stretching. Such an approach would introduce unnecessary dissipation, dampening the flow velocity which degrades the visual appearance of the flow. To illustrate this fact we produce a scene originally developed by Park and Kim [105], applying our methods, shown in Figure 4.6. As opposed to their results, our smoke simulation flows freely and is not affected by dampening caused by viscosity.

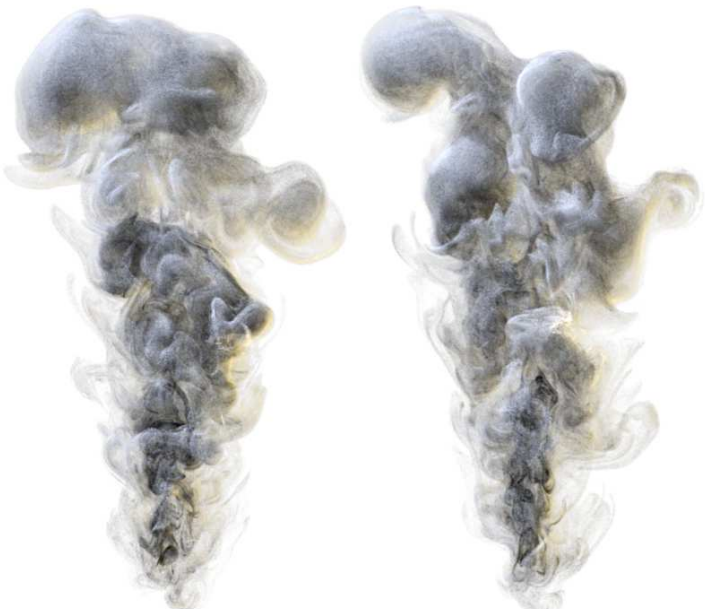


Figure 4.5: Comparison of simulation of a turbulent smoke column with vortex stretching. Left: Gradient estimation stretching. Right: Advection-driven stretching. Fast rotating smoke is obtained by generating vortices at different scales of strength and radius. Rotation is reduced by vorticity dissipation as the smoke rises, giving a more uniform appearance towards the top of the column. An animation sequence for this figure is shown in the accompanying video at 0:50.

4.3.3 Vorticity Dissipation

Since the above vortex stretching methods only introduce a change in the vorticity orientation and not in the magnitude, a turbulent energy cascade is not correctly represented. A fluid that preserves its energy at all scales would also be visually incorrect. We therefore employ a simple vorticity dissipation method that produces a visually rich result.

We employ the notion of *contained kinetic energy* [109] of a vortex. This quantity corresponds to the kinetic energy associated with the velocity field induced by a single vortex j and is formulated as follows:

$$E_j = \int_{V_j} \frac{\rho}{2} \left\| \mathbf{u}_j(\mathbf{x}) \right\|^2 dV_j, \tag{4.27}$$

where $\mathbf{u}_j$ and V_j are the vortex-induced velocity and its region of influence, respectively. For our specific kernel it can be shown that $E_j = k \left\| \boldsymbol{\omega}_j \right\|^2 \epsilon^5$, where $k = \rho\pi \cdot 0.0107$.

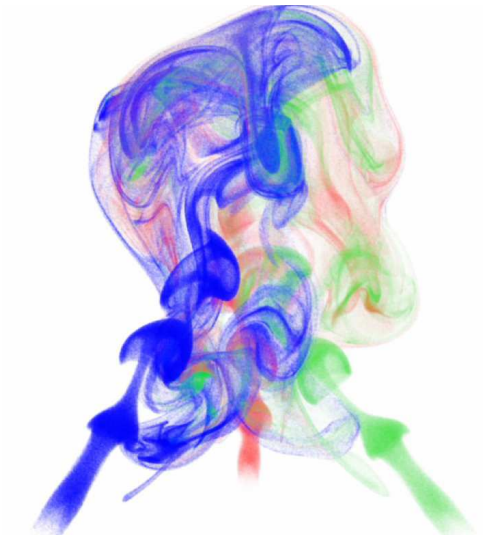


Figure 4.6: Three smoke jets colliding in a scene mirroring an example from Park and Kim [105] (Fig. 7). An animation sequence for this figure is shown in the accompanying video at 2:37.

Then, in order to preserve each individual vortex energy, any change in the vorticity magnitude induces a change in the vorticity radius given by:

$$\epsilon_{new} = \epsilon_{old} \sqrt[5]{\frac{\|\boldsymbol{\omega}_{old}\|^2}{\|\boldsymbol{\omega}_{new}\|^2}}. \tag{4.28}$$

Then at each time step, we reduce the vorticity magnitude for each vortex particle. Then we apply this formula to recalculate the radius of each vortex to preserve its contained energy.

4.4 Chapter Summary and Discussion

In this chapter we have presented our computational flow model for an unbounded simulation. We have discussed our computational representation of vorticity, which is the key quantity that we use to represent the flow. We have also introduced our method to compute the velocity induced by vortex particles using finite influence radius kernels and we have given examples of simulations showing a high degree of visual fidelity.

We have presented our novel methods to compute vortex stretching, overcoming the stability problems reported earlier in literature. As opposed to the approach by Park

and Kim [105], we do not require to manipulate viscosity in the flow to avoid instability due to computing vortex stretching, enabling us to produce highly detailed and visually rich simulations of free flows.

We summarize the advantages of Lagrangian vortex methods: First, they allow the simulation of fine details by only defining the vorticity of each vortex particle according to the required detail scale. Second, only a sparse amount of simulation data, in our case a set of vortex particles, is required to produce such simulations. Finally, the pressure term disappears from the vorticity transport equation. This eliminates the need for computing a large matrix for pressure as in traditional grid methods [129], or to obtain accurate incompressible flows using SPH [112]. In our framework, incompressibility is given by Proposition 1.

In the next chapter we present our methods to simulate flows around solid objects. Here a solid object acts as an obstacle for the flow and we define and enforce proper boundary conditions to obtain visually rich flows.

Chapter 5

Solid Boundary Conditions

In the previous chapter we have presented an unbounded flow model. In this chapter we extend this model to simulate flows bounded by static rigid solid objects. In our framework, we assume a solid object is represented by a non-self-intersecting polygonal surface.

There are two types of boundary conditions we look to enforce: the *no-penetration* boundary condition [105] and a *slip* condition.

The no-penetration boundary condition models the impermeability of the solid object. This is achieved by enforcing the following condition at the solid surface:

$$\mathbf{u}_{Solid} \cdot \hat{\mathbf{n}} = \mathbf{u}_{Fluid} \cdot \hat{\mathbf{n}}, \tag{5.1}$$

where $\mathbf{u}_{Solid}$ and $\mathbf{u}_{Fluid}$ are the solid and fluid velocity respectively, and $\hat{\mathbf{n}}$ corresponds to the solid surface normal. In our Lagrangian system, we enforce this condition by canceling the normal component of the flow velocity at the solid surface.

Park and Kim [105] use a surface vorticity distribution tangent to the boundary where they find the correct vorticity distribution to cancel flow through the surface by solving a system of $3N$ equations where N is the number of surface elements. As opposed to their work, we introduce a novel method to computer graphics where a source distribution on the surface cancels the normal component of velocity. Only N equations need to be solved in our case. This kind of solution improves flow results in the presence of complex obstacles [78].

The *slip* condition we look to enforce relates to the tangential component of the velocity at the solid surface. This condition is based on the *no-slip* boundary condition [15] for viscous fluids, which corresponds to enforcing the following constraint on the flow at

the solid object's surface:

$$\mathbf{u}_{_{Solid}} = \mathbf{u}_{_{Fluid}}. \tag{5.2}$$

This condition can be enforced through applying a no-penetration boundary condition and generating vorticity at the solid surface, canceling the tangential component of the flow velocity at the solid boundary. In our case we are interested in representing different solid surface roughness, which is achieved by scaling the generated vorticity. This way we are able to represent diverse *slip* conditions that can be seen as the effect of the solid's roughness on the flow, adding visual interest to fluid simulations.

In this chapter we present our approach for enforcing no-penetration and slip conditions. Then we discuss challenges in enforcing solid boundary conditions using Lagrangian vortex methods due to geometrical features of the solid object.

5.1 No-Penetration Condition

Our aim is to cancel the normal component of the velocity through the solid object. To achieve this, we employ an extended velocity field consisting of the external flow plus a velocity field induced by the solid surface. The addition of both velocity fields generates a flow that satisfies the non-penetration boundary condition. In the following sections we describe the theoretical and practical aspects of our approach.

5.1.1 Source Sheets

The normal component of the flow velocity can be canceled by adding a velocity field at the solid surface opposing the flow through the surface as shown in Figure 5.1. One way to obtain such a velocity field is by modeling the surface as a *source sheet* [78].

A source sheet can be seen as the limit of infinitely many sources (or sinks) distributed on a surface $\mathcal{S}$. The velocity induced by a source sheet, at a point in space $\mathbf{x}$, which does not lie on the sheet is defined as:

$$\mathbf{u}(\mathbf{x}) = \int_{\mathcal{S}} \frac{\lambda(\mathbf{z})}{4\pi} \frac{\mathbf{x}-\mathbf{z}}{\|\mathbf{x}-\mathbf{z}\|^3} d\mathbf{z}. \tag{5.3}$$

Here $\lambda(\mathbf{z})$ is the scalar source strength. The velocity at a point $\mathbf{x}_{_B}$ that lies on the sheet is determined by taking the limit $\mathbf{x} \to \mathbf{z}$ of Equation (5.3) and it corresponds to:

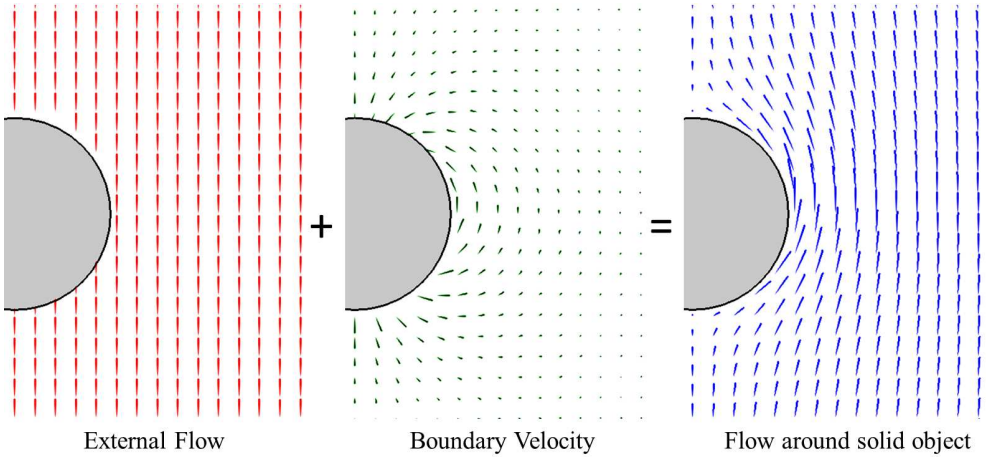


Figure 5.1: Velocity fields for boundary conditions around a sphere. Left: External uniform flow. Center: Velocity field induced by the solid boundary. Right: Result of adding the surface boundary velocity to the external flow satisfying the no-penetration boundary condition.

$$\mathbf{u}(\mathbf{x}_B) = \frac{\lambda(\mathbf{x}_B)\hat{\mathbf{n}}}{2}. \tag{5.4}$$

We use source distributions to enforce no-penetration boundary conditions. In the next section we formulate the necessary equations to achieve this goal.

5.1.2 Boundary Integral Problem

The no-penetration boundary condition is satisfied when the flux through the surface is zero, *i.e.*, $(\mathbf{u}_{Fluid} - \mathbf{u}_{Solid}) \cdot \hat{\mathbf{n}} = 0$. If the flux through the solid surface is non-zero this is corrected by introducing a source sheet at the solid boundary, such that it cancels the normal component of the total velocity at the solid surface.

We define $\Delta\mathbf{u} = \mathbf{u}_{Fluid} - \mathbf{u}_{Solid}$. The source sheet is determined by solving the unknown strength λ in the following boundary integral equation at surface locations $\mathbf{x}_B$:

$$-(\Delta\mathbf{u} \cdot \hat{\mathbf{n}})(\mathbf{x}_B) = \frac{\lambda(\mathbf{x}_B)}{2} + \int_{\mathcal{S}} \frac{\lambda(\mathbf{z})}{4\pi} \frac{\mathbf{x}_B - \mathbf{z}}{\|\mathbf{x}_B - \mathbf{z}\|^3} d\mathbf{z} \cdot \hat{\mathbf{n}}(\mathbf{x}_B). \tag{5.5}$$

The above equation is derived directly from the definition of the velocity field in Equation (5.3). The first term in the right hand side of Equation (5.5) represents the velocity induced by the point $\mathbf{x}_B$ on the sheet, obtained by the limit value of Equation

(5.3) when $\mathbf{x} \to \mathbf{z}$. The second term represents the velocity induced by the rest of the sheet on the location $\mathbf{x}_B$.

Enforcing the no-penetration boundary condition is performed by solving the unknown strength $\lambda(\mathbf{x}_B)$ such that Equation (5.5) holds. In the next section we describe how to discretize and solve Equation (5.5) to enforce the no-penetration boundary condition.

5.1.3 Discretization of Boundary Equations

We discretize Equation (5.5) to solve for the unknown scalar strength λ on the solid surface employing the *panel method* [60]. Each polygon or *panel* of the solid boundary corresponds to a source sheet which approximates the solution of Equation (5.5).

At each panel i we define a control point $\mathbf{x}_i$ at the center of the panel. We use constant strength panels. Then at each control point: (a) we sample the fluid velocity, and (b) we compute the influence of other panels.

To compute panel influence, we use the finite regularized kernel described in Section 4.2. To this end, we define the following velocity field:

$$\mathbf{u}_j^*(\mathbf{x}) = \int_j (\mathbf{x} - \mathbf{z}) w(\|\mathbf{x} - \mathbf{z}\|) d\mathbf{z}. \tag{5.6}$$

Since strength λ is constant for each panel j, the influence of panel j on the control point of panel i is given simply by $\lambda_j \mathbf{u}_j^*(\mathbf{x}_i)$. We rewrite Equation (5.5) as follows:

$$-\Delta\mathbf{u}(\mathbf{x}_i) \cdot \hat{\mathbf{n}}_i = \frac{\lambda_i}{2} + \sum_{j=1}^{N_p} \lambda_j \mathbf{u}_j^*(\mathbf{x}_i) \cdot \hat{\mathbf{n}}_i. \tag{5.7}$$

Here N_p is the number of panels. Equation (5.7) is a $N_p \times N_p$ linear system of the form $\mathbf{A}\boldsymbol{\lambda} = \mathbf{b}$.

We evaluate the integrals for $\mathbf{u}_j^*$ using Gauss-Legendre quadratures. The matrix $\mathbf{A}$ is not necessarily symmetric, nor positive-definite and the kernel radii used to evaluate the matrix coefficients determine the sparsity of $\mathbf{A}$. We note that for a rigid body, $\mathbf{A}$ is constant and we calculate it and invert it in a pre-computation step.

Influence radii ϵ of each panel are determined from a *characteristic length* L_C of the solid object. In our simulations, the characteristic length employed corresponds to the radius of the smallest cylinder that contains the object and we use $0.5L_C \leq \epsilon \leq 1.5L_C$.

Once the source strength λ is determined for each panel, the solid object induced velocity is the sum of the velocities induced by each panel, *i.e.*, :

$$\mathbf{u}_\lambda(\mathbf{x}) = \sum_{j=1}^{N_p} \lambda_j \mathbf{u}_j^*(\mathbf{x}). \tag{5.8}$$

The velocity $\mathbf{u}_\lambda$ is added to the simulation, satisfying the no-penetration boundary condition, as shown in Figure 5.1. Here the benefits of using our finite kernel become evident as the effects of source panels are not evaluated in points that are too far from the solid object where the induced velocity is negligible.

5.1.4 The Case of Finite Kernel Vortices

The above presented method poses additional challenges for modeling flows using vortex particles with a finite influence radius. For instance, consider a symmetric, regular ascending vortex jet past a sphere as the one shown in Figure 5.2. If the radii of particles are too small (*e.g.*, smaller than the sphere diameter), artifacts are introduced in the flow velocity due to two main facts: a) Not all panels in the solid object are influenced by a small radius vortex, and b) distances between vortices increase due to the insertion of the solid, hence reducing the mutual influence of particles. Then, as vortex particles flow past a solid boundary, the general features of the flow are distorted by the dominant effect of local flow structures. The exact error introduced by the finite influence radius is difficult to estimate, since source panel strength is determined by solving a large linear system. This becomes a more difficult problem as sources and vortices have different influence radii.

Satisfying the no-penetration boundary condition may also be seen as enforcing the solid boundary to be a flow *streamline*. A streamline is a curve that is tangent to the flow velocity. Park and Kim [105] achieve this by employing a vortex panel approach, where panels force the velocity field to be tangent to the solid surface. As opposed to their approach, we enforce the solid surface to be a flow streamline locally with a vortex *image*. When a solid boundary intersects the influence radius of a vortex located at $\mathbf{z}_i$, a second vortex with same radius and opposite spin is generated at the mirror position $\mathbf{z}_i'$ across the solid surface. The velocities of the vortex and its mirror position are added to ensure that the flow is tangential to the surface.

Notice that source panels are still necessary to account for the flow's irrotational features (*e.g.*, a user-defined background flow) whereas vortices and their mirrors model the

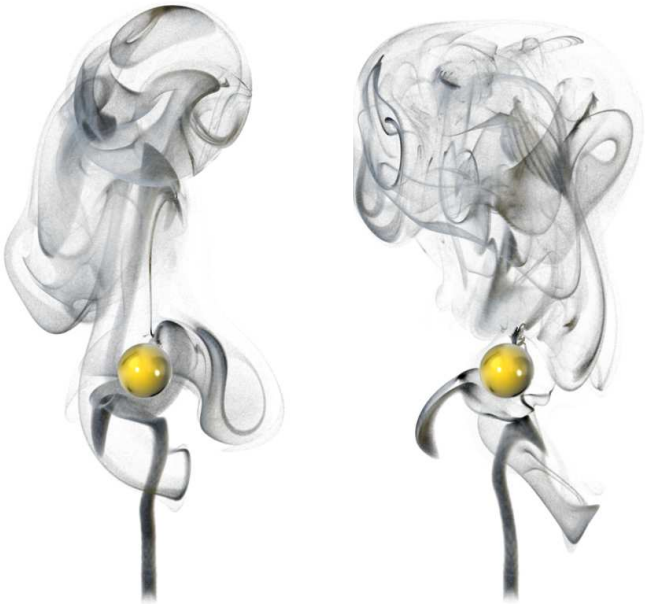


Figure 5.2: Enforcing no-penetration boundary conditions on a sphere. Left: Source panels with the Rosenhead-Moore kernel. Right: Source panels with our finite kernel. An animation sequence for this figure is shown in the accompanying video at 1:40.

rotational component of the flow. In the following examples we employ a combination of source panels and vortex images to satisfy no-penetration boundary conditions. A comparison of results obtained using the infinite influence Rosenhead-Moore kernel and our finite radius kernel is shown in Figure 5.2. We verify that the flow does not penetrate the solid object in either case. Since the kernel functions used produce different numerical results, simulations produce visually different results, however both satisfying the required no-penetration boundary condition.

5.2 Boundary Layer Model

The main disadvantage of the velocity field calculated using source panels is that the net force calculated on the solid object is zero. This can be resolved by calculating the surface vorticity around the object and then directly computing pressure differences as it is commonly done in airfoil design. However, this type of solution strategy depends on the airfoil shape and specific flow conditions, which do not hold for the turbulent flows and general solid shapes in which we are interested.

Instead, we couple our potential flow model with a boundary layer simulation. We

impose an additional constraint on the tangential components of the flow velocity at the solid boundary. This enables the simulation of a variety of slip conditions where the no-slip boundary condition [15] is achieved by canceling the tangential component of flow velocity at the solid surface. This is in addition to the no-penetration boundary condition which imposes a constraint on the normal components of the flow velocity at the solid surface as we discussed in the previous sections.

Fluid particles in the boundary layer are subject to a torque induced by the motion of other fluid particles farther from the solid surface. This torque translates into vorticity that is generated at the solid surface and eventually released into the flow as vortex shedding.

We apply our boundary layer model in three steps: First, we determine the slip velocity which is the tangential component of the flow velocity at the surface. This is done by modeling the surface with a vortex sheet whose induced velocity coincides with the slip velocity. Second, we enforce the no-slip boundary condition by emitting the surface vorticity into the main flow as new vortex particles. Third, we advect the newly created vortex particles to simulate boundary layer separation. In the following sections we describe these three processes.

5.2.1 Vortex Sheets

The solid's surface $\mathcal{S}$ is modeled using a vortex sheet $\boldsymbol{\gamma}$. This can be regarded as the limit of infinitely many vortices lying on the surface. In 3D, the vorticity $\boldsymbol{\gamma}$ is a vector quantity which is parallel to the surface [120].

The velocity generated by a vortex sheet at a point $\mathbf{x}$ not lying on the sheet is defined by:

$$\mathbf{u}(\mathbf{x}) = \int_{\mathcal{S}} \frac{\boldsymbol{\gamma}(\mathbf{z})}{4\pi} \times \frac{\mathbf{x}-\mathbf{z}}{\|\mathbf{x}-\mathbf{z}\|^3} d\mathbf{z}. \tag{5.9}$$

The velocity magnitude at a point $\mathbf{x}_B$ lying on the surface $\mathcal{S}$ is obtained by a limit process and it corresponds to:

$$\mathbf{u}(\mathbf{x}_B) = \pm \frac{\boldsymbol{\gamma}(\mathbf{x}_B) \times \hat{\mathbf{n}}(\mathbf{x}_B)}{2}. \tag{5.10}$$

The $\pm$ sign depends on which side of the sheet the limit to the point $\mathbf{x}_B$ is taken, as a vortex sheet defines a discontinuous jump in tangential velocity at the surface, as

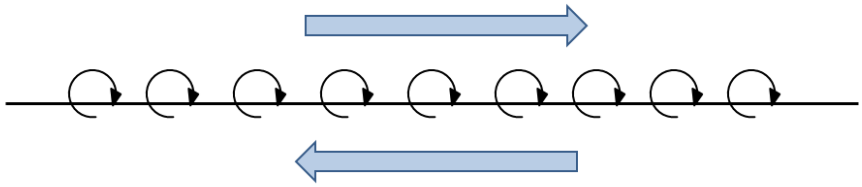


Figure 5.3: Side view of a vortex sheet with the tangential velocity induced at both sides of the sheet represented with blue arrows.

illustrated in Figure 5.3.

In the following section we explain how to employ the notion of a vortex sheet to find the surface vorticity.

5.2.2 Boundary Integral Problem

The surface vorticity is determined by the vortex sheet that produces the slip velocity, which is the tangential component of the flow velocity at the surface. This velocity is later used to define a specific slip condition on the solid object.

Similarly to enforcing the no-penetration boundary condition, we determine the surface vorticity with a boundary integral equation in the unknown $\boldsymbol{\gamma}$ at the surface. But $\boldsymbol{\gamma}$ is a vector quantity tangent to the surface and we need to solve for each of its components. Let $\hat{\mathbf{s}}$ and $\hat{\mathbf{t}}$ be the tangent and bitangent vectors of the solid surface, respectively. Then $\boldsymbol{\gamma}$ can be decomposed as follows:

$$\boldsymbol{\gamma} = \gamma^s \hat{\mathbf{s}} + \gamma^t \hat{\mathbf{t}}. \tag{5.11}$$

We find each component of the unknown vortex sheet $\boldsymbol{\gamma}$ that determines the slip velocity with a boundary integral equation of the form [32]:

$$(\Delta \mathbf{u} \cdot \hat{\mathbf{s}})(\mathbf{x}_B) = \frac{(\boldsymbol{\gamma} \times \hat{\mathbf{n}})(\mathbf{x}_B)}{2} \cdot \hat{\mathbf{s}}(\mathbf{x}_B) + \int \frac{\boldsymbol{\gamma}(\mathbf{z})}{4\pi} \times \frac{\mathbf{x}_B - \mathbf{z}}{\|\mathbf{x}_B - \mathbf{z}\|^3} d\mathbf{z} \cdot \hat{\mathbf{s}}(\mathbf{x}_B). \tag{5.12}$$

The above equation is directly formulated from the definition of a vortex sheet velocity in Equation (5.9) and it specifies a vortex sheet of strength $\boldsymbol{\gamma}$ whose induced velocity coincides with the slip velocity $(\Delta \mathbf{u} \cdot \hat{\mathbf{s}})$. An analogous equation is formulated for the $\hat{\mathbf{t}}$-component of surface vorticity as follows:

$$(\Delta \mathbf{u} \cdot \hat{\mathbf{t}})(\mathbf{x}_{_B}) = \frac{(\boldsymbol{\gamma} \times \hat{\mathbf{n}})(\mathbf{x}_{_B})}{2} \cdot \hat{\mathbf{t}}(\mathbf{x}_{_B}) + \int \frac{\boldsymbol{\gamma}(\mathbf{z})}{4\pi} \times \frac{\mathbf{x}_{_B} - \mathbf{z}}{\|\mathbf{x}_{_B} - \mathbf{z}\|^3} d\mathbf{z} \cdot \hat{\mathbf{t}}(\mathbf{x}_{_B}). \tag{5.13}$$

The solution of $\boldsymbol{\gamma}$ in Equations (5.12) and (5.13) determines the vortex sheet that defines the slip velocity of the flow at the solid boundary. We proceed in a similar way as we do for enforcing no-penetration boundary condition, by discretizing the above equations and solving them using panel methods as described in the following section.

5.2.3 Discretization Using the Panel Method

We proceed with constant strength panels similar to Section 5.1.3 and define a control point $\mathbf{x}_i$ at the center of each panel i. Under these assumptions we can rewrite Equation (5.12) evaluated at the control points, using the velocity field $\mathbf{u}^*$ defined in Equation (5.6) as follows:

$$\Delta \mathbf{u}(\mathbf{x}_i) \cdot \hat{\mathbf{s}}_i = \frac{\boldsymbol{\gamma}_i \times \hat{\mathbf{n}}_i}{2} \cdot \hat{\mathbf{s}}_i + \sum_j \gamma_j \times \mathbf{u}_j^*(\mathbf{x}_i) \cdot \hat{\mathbf{s}}_i. \tag{5.14}$$

By applying the definition of $\boldsymbol{\gamma}_j$ in each of its components, the above equation can be written as follows:

$$\Delta \mathbf{u}(\mathbf{x}_i) \cdot \hat{\mathbf{s}}_i = \frac{\gamma_i^t}{2} + \sum_j \gamma_j^s (\hat{\mathbf{s}}_i \times \hat{\mathbf{s}}_j) \cdot \mathbf{u}_j^*(\mathbf{x}_i) + \sum_j \gamma_j^t (\hat{\mathbf{s}}_i \times \hat{\mathbf{t}}_j) \cdot \mathbf{u}_j^*(\mathbf{x}_i). \tag{5.15}$$

The derivation of Equation (5.15) is detailed in Appendix C. We set the panel influence radius to be the same used in our source panel implementation. Similarly, we obtain the following discrete expression for Equation (5.13):

$$\Delta \mathbf{u}(\mathbf{x}_i) \cdot \hat{\mathbf{t}}_i = -\frac{\gamma_i^s}{2} + \sum_j \gamma_j^s (\hat{\mathbf{t}}_i \times \hat{\mathbf{s}}_j) \cdot \mathbf{u}_j^*(\mathbf{x}_i) + \sum_j \gamma_j^t (\hat{\mathbf{t}}_i \times \hat{\mathbf{t}}_j) \cdot \mathbf{u}_j^*(\mathbf{x}_i). \tag{5.16}$$

Equations (5.15) and (5.16) are a linear system of equations of the form:

$$\mathbf{B} \begin{bmatrix} \gamma_i^s \\ \vdots \\ \gamma_i^t \\ \vdots \end{bmatrix} = \begin{bmatrix} \Delta \mathbf{u}(\mathbf{x}_i) \cdot \hat{\mathbf{s}}_i \\ \vdots \\ \Delta \mathbf{u}(\mathbf{x}_i) \cdot \hat{\mathbf{t}}_i \\ \vdots \end{bmatrix}. \tag{5.17}$$

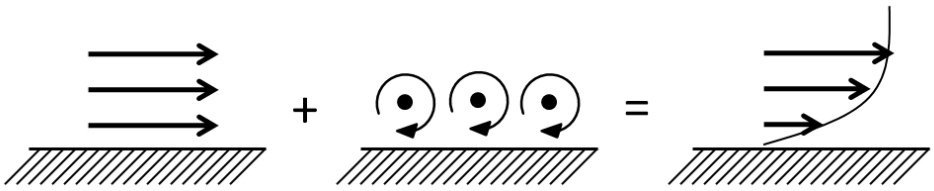


Figure 5.4: Enforcing no-slip boundary conditions around a solid object. (a) External uniform flow. (b) Velocity distribution from shed vortices. (c) Addition of (a) and (b).

The matrix $\mathbf{B}$ has dimensions $2N_p \times 2N_p$ as the influence of two components of vorticity per panel are considered. As in the case of the matrix for solving the no-penetration boundary conditions, in general the matrix $\mathbf{B}$ is not symmetric, nor positive definite but it will be sparse as long as the kernel radii in the velocity evaluation are small enough. We note that this matrix is again constant for a rigid solid object and we also calculate it and its inverse in a pre-computation step.

Simulating a deformable object would require us to re-compute and solve this matrix for each time step, since panels change their relative positions with respect to each other and so their influences on each other.

Once the surface vorticity is determined for each panel, it is possible to model the development of the boundary layer which is discussed in the next section.

5.2.4 Vortex Shedding

We can enforce diverse slip boundary conditions through vortex shedding by emitting vortices in the flow. To enforce no-slip boundary conditions, these vortices must cancel the slip velocity as shown in Figure 5.4. The velocity field induced by these vortices alters the velocity field around the object such that the boundary layer separates from the solid surface as shown in Figure 5.5.

We apply a simplified version of the method by Park and Kim [105]. We generate a new vortex at a close random distance d from each panel. The vorticity of each vortex is:

$$\boldsymbol{\omega}_i^{new} = c\boldsymbol{\gamma}_i A_i \Delta t, \tag{5.18}$$

where $\boldsymbol{\gamma}_i$ and A_i correspond to the vorticity and area of panel i respectively. The slip coefficient c controls the friction between the solid and fluid. The free-slip boundary condition, where $c = 0$, occurs when the shed vortices have zero vorticity. The no-slip

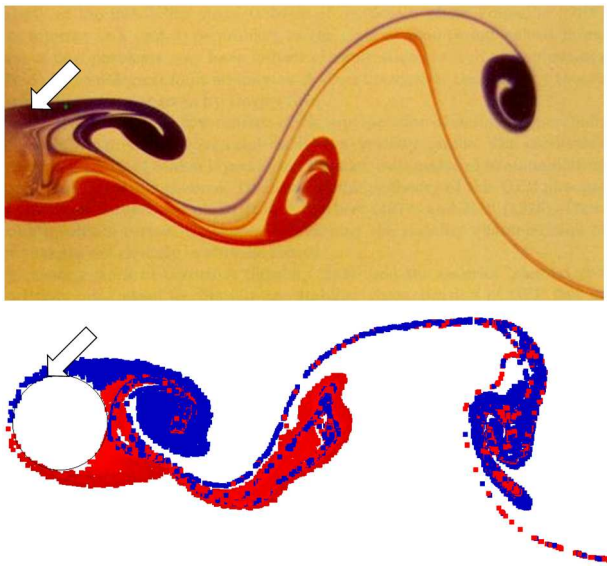


Figure 5.5: Top: Periodic vortex shedding pattern, known as *Kármán vortex street*, observed in flow past a cylinder (indicated with an arrow) from a real-life experiment from Perry, A. E., Chong, M. S. and Lim, T. T., "The vortex-shedding process behind two-dimensional bluff bodies", Journal of Fluid Mechanics, 116, 1982, pages 77–90. © Cambridge University Press. Reproduced with permission. Bottom: Kármán vortex street approximation obtained in our 2D simulation. The cylinder is indicated with an arrow. An animation sequence for this figure is shown in the accompanying video at 1:05.

boundary condition, where $c = 1$, occurs when the shed vortices have full strength. We set each vortex radius to the same radius used for panel calculations.

Park and Kim [105], diffuse vortices using a random walk, using the same strategy employed by Chorin [28] in 2D and by a particle vorticity exchange step in 3D, simulating the diffusion of vorticity from a particle to its neighbors. We do not perform these steps as the effects of diffusion in a low viscosity flow are dominant only within the boundary layer. Outside of this layer, fluid motion is dominated by advection, and we focus on the evolution of the boundary layer in this area. We can reproduce natural boundary layer evolution using this very simple model as shown in Figure 5.5. A comparison of different boundary layer separation patterns for different values of c are shown in Figures 5.6 and 5.7. An example of our methods for enforcing boundary conditions on the Stanford Bunny model is shown in Figure 5.8.

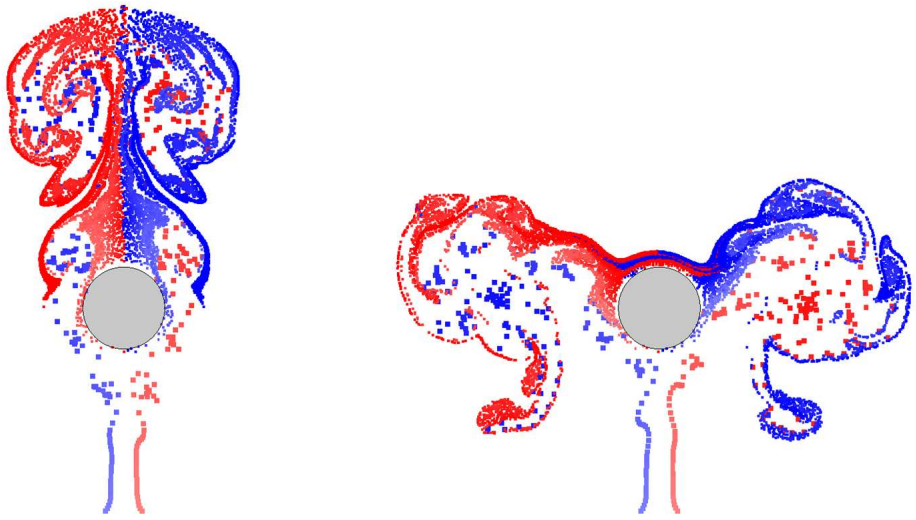


Figure 5.6: Comparison of slip flow patterns produced by a vertical vortex jet around a static 2D sphere with different slip coefficients c. Left: $c = 0$ (free slip). Right: $c = 0.7$. An animation sequence for this figure is shown in the accompanying video at 1:20.

5.3 Panel Methods Challenges

In the previous sections we have discussed our approach for enforcing boundary conditions around a solid object based on a potential flow model and a boundary layer model. We have shown that this strategy produces visually realistic flows around objects of irregular geometry as the example in Figure 5.8. In this section we present the main challenges related to our methods for enforcing boundary conditions. In particular we discuss cases of solid objects where only enforcing no-penetration boundary conditions as described in Section 5.1 may lead to inaccurate results.

5.3.1 Limitations of Source Panel Methods

Enforcing no-penetration boundary conditions using a source distribution discretized using source panels may lead to inaccurate results for certain geometries. We identify the case of concave objects as a particular case where the velocity field generated with source panels leads to incorrect flows. An example of this case is shown in Figure 5.9.

As the flow approaches a concave area, the flow velocity is reduced gradually until it stops at the solid boundary. The velocity fields involved in this case are shown in Figure 5.10. Flow directed towards the concave part of a bowl will accumulate indefinitely. This indicates an area of non-zero divergence in the concave region of the fluid. It is a

Figure 5.7: 3D flow pattern comparison with a vertical vortex jet around a static cylinder of unit radius with slip coefficients $c = 0.0$ (left) and $c = 1.0$ (right). An animation sequence for this figure is shown in the accompanying video at 1:28.

consequence of using source distributions on the solid surface; recall that divergence is a measurement of sources and sinks in the flow. Visually, the fluid "disappears" as it approaches the solid boundary, instead of pushing the flow already in the concave area out of the bowl as would be expected from physical observation.

The expected behavior can be reproduced using the boundary layer model proposed in Section 5.2. The slip velocity in a concave area is directed towards the concave region of the solid surface. A vortex that satisfies no-slip boundary condition cancels the slip velocity, therefore introducing a velocity field that is directed away from the concave region. A schema of this idea is presented in Figure 5.11.

The velocity field induced by the shed vortices produces the necessary velocity component tangent to the solid surface to achieve overflow from the concave portion of the solid surface. An example of the velocity field using shed vorticity for the case shown in Figure 5.9 is presented in Figure 5.12. A snapshot of fluid simulation with vortex shedding for the same scenario is presented in Figure 5.13.

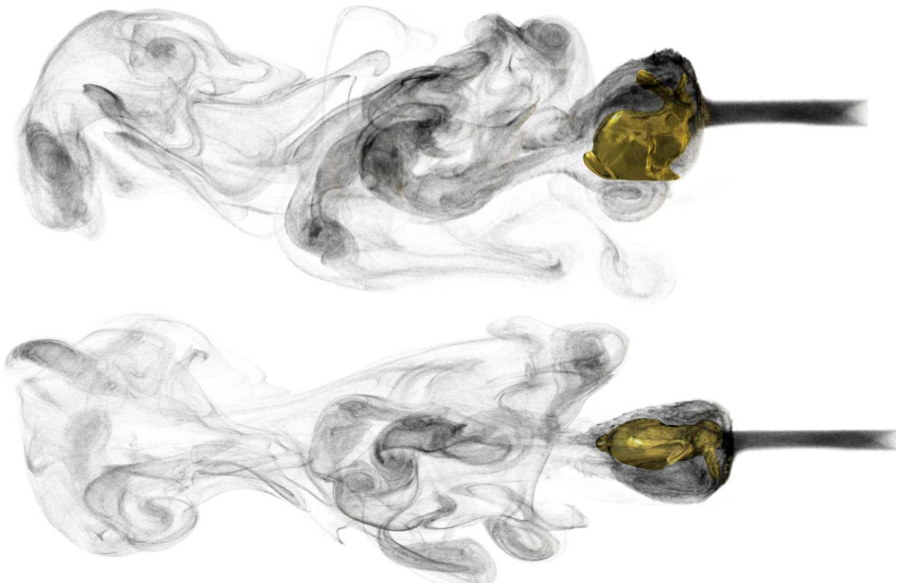


Figure 5.8: Example of flow induced by a vortex jet past a static Stanford Bunny model with no-slip condition and vortex shedding with different camera positions. Top: camera aiming at the bunny sideways. An animation sequence for this figure is shown in the accompanying video at 1:32. Bottom: camera aiming at the bunny from the top.

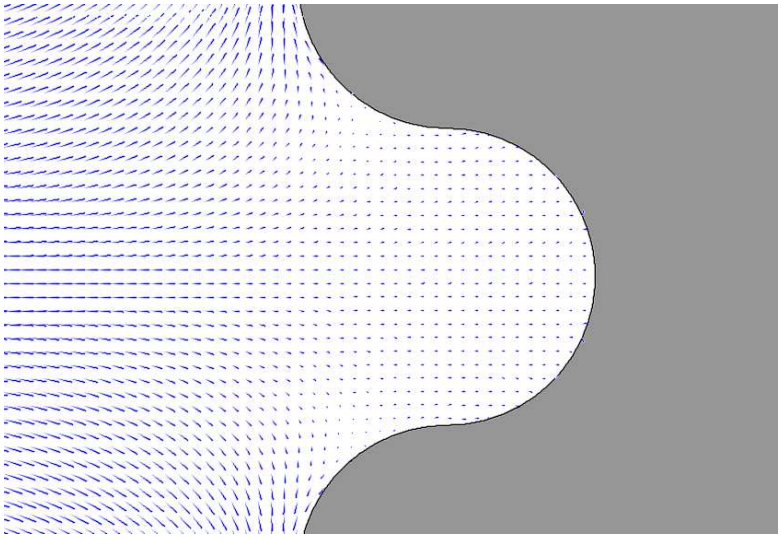


Figure 5.9: Example of flow around a concave object computed using source panels. A constant flow is directed towards the concave area of the solid object, where it slows down and accumulates indefinitely.

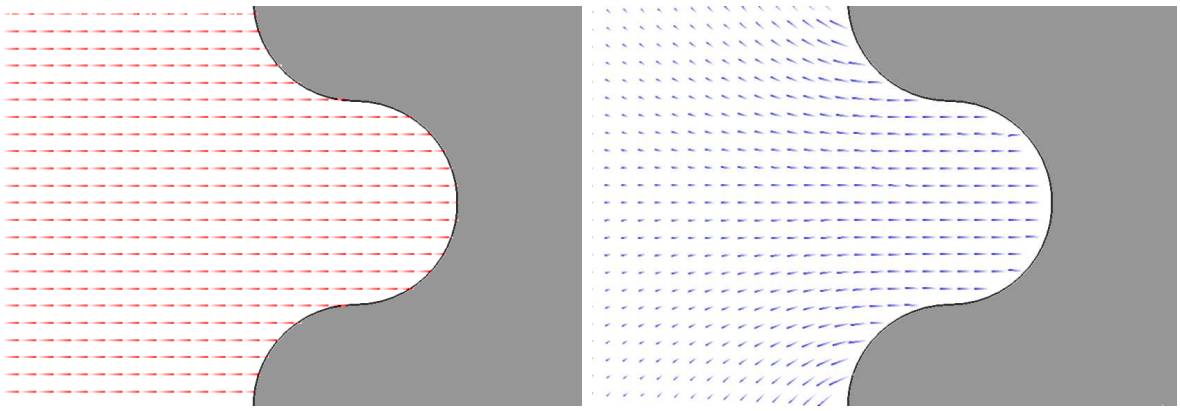


Figure 5.10: Velocity fields in generated with source panels. Left: External velocity field. Right: Source panel field.

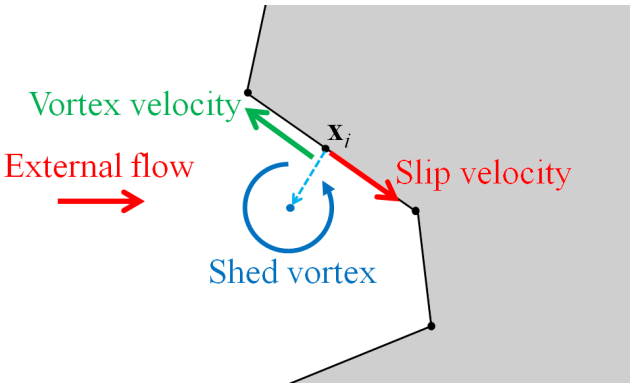


Figure 5.11: Velocity generated by shed vortices in concave areas. A vortex (marked in blue) is shed to cancel the slip velocity due to the external flow (in red). The velocity generated by the vortex (in green) directs the flow away from the concavity.

The boundary layer simulation becomes a key aspect of achieving realistic simulations of flows around objects. Boundary layer generation and separation is a natural fluid phenomenon even of those fluids that are commonly considered inviscid (*e.g.*, water and air). In the context of enforcing boundary conditions using a pure Lagrangian vortex particle method, it is not only desirable to simulate a boundary layer, but also necessary to overcome the above presented limitation of source panels.

We have discussed so far the presence of concave areas at the solid's surface, without discussing the effect of the local curvature on the flow. In the next section we discuss the main issues related to high-curvature concave areas and we discuss possible solutions.

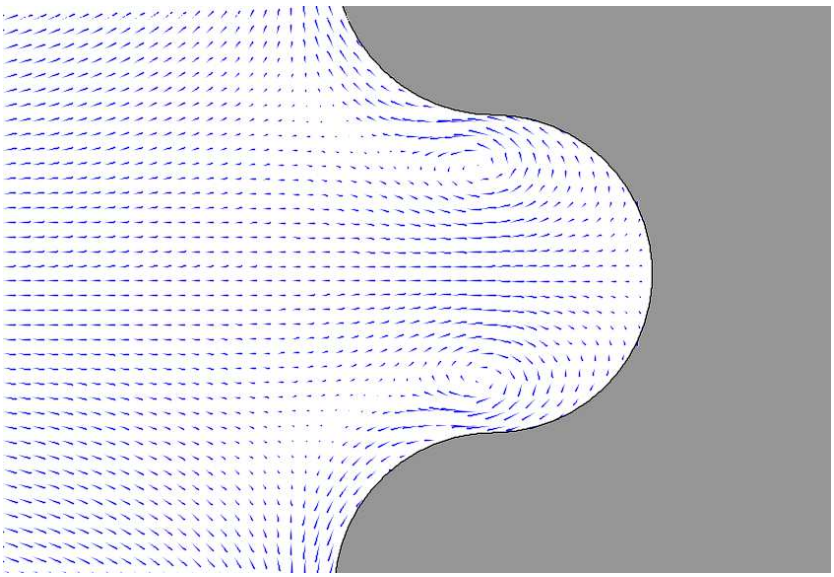
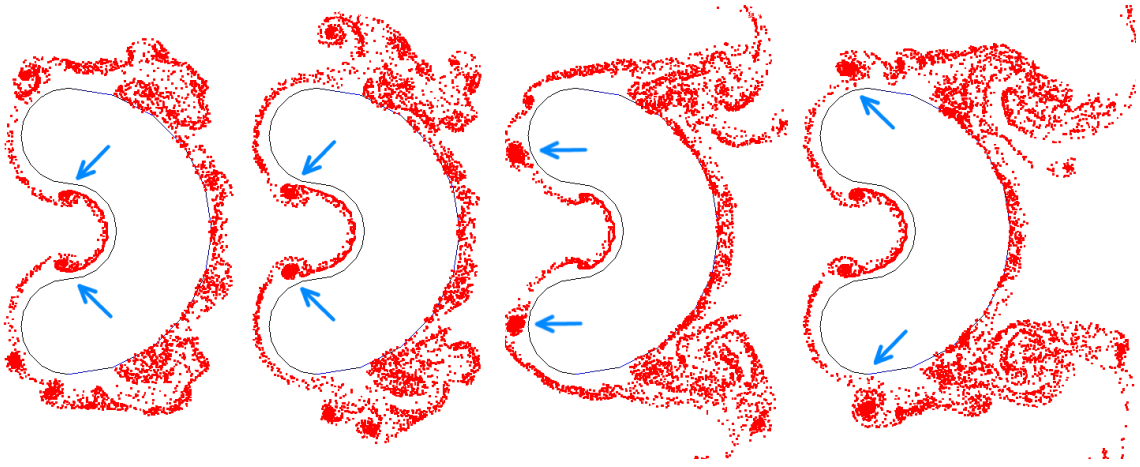


Figure 5.12: Example of velocity due to vortex shedding on concave object.



(a) (b) (c) (d)

Figure 5.13: Animation snapshots of flow towards concave object with vortex shedding. The external flow is directed towards the concave area of the solid object as in Figure 5.10. We observe the trajectory of eddies (indicated with blue arrows) resulting from our boundary layer simulation in the concave area. (a) and (b): Two eddies form inside the concave region and advance to the upper and lower halves of the surface. (c): The two eddies move outside the concave area. (d): The two eddies are located outside the concave region, while new eddies in the concave area are forming.

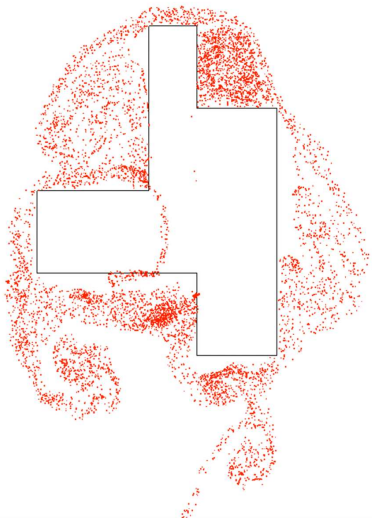


Figure 5.14: Example flow penetration through a solid object in a high curvature concave surface. A constant external flow is directed downwards. Shed vortices (colored in red) penetrate the solid surface in a concave high-curvature portion of the surface.

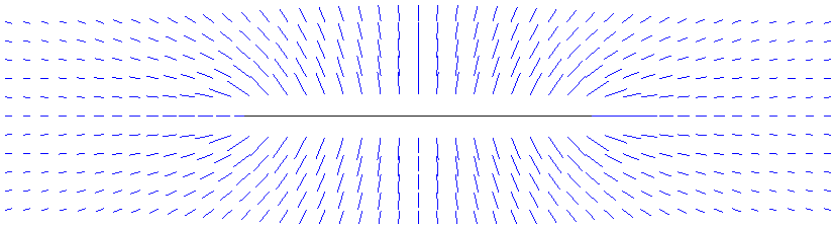


Figure 5.15: Example of velocity field induced by a single horizontal panel. The background velocity field is constant and is directed from top to bottom.

Curvature and Concavity

Concave areas of high curvature do not only pose the challenges described in the previous section, but also have a high rate of flow penetration when the flow is faced towards the concavity, as shown in Figure 5.14. We observe that at high curvature concave areas, the velocity field generated by constant source panels does not prevent flow penetration. Most flow penetration into the solid object occurs at the surface vertices, which are located far from the sampling points of the panel method, *i.e.*, the center of each solid face. We see in Figure 5.15 that the normal component of the panel induced velocity has reduced magnitude towards the panel vertices.

It can be argued that specifically designed higher-order panel methods may perform

better in this scenario. This would require developing a general strategy that would address this issue in 2D and 3D simulations. However, replacing a constant strength panel by a more complex function would also increase the mathematical complexity and computational cost of evaluating panel velocities.

We employ a simpler alternative, following the same idea as presented in the previous section for flow directed towards concave areas in general. Here our aim is to keep the cost of evaluating panel velocity low and to develop methods for improving robustness that can be easily applied in 2D and 3D simulations. We employ a curvature flow advection of the solid surface to smooth concave areas of high curvature with the following surface velocity definition:

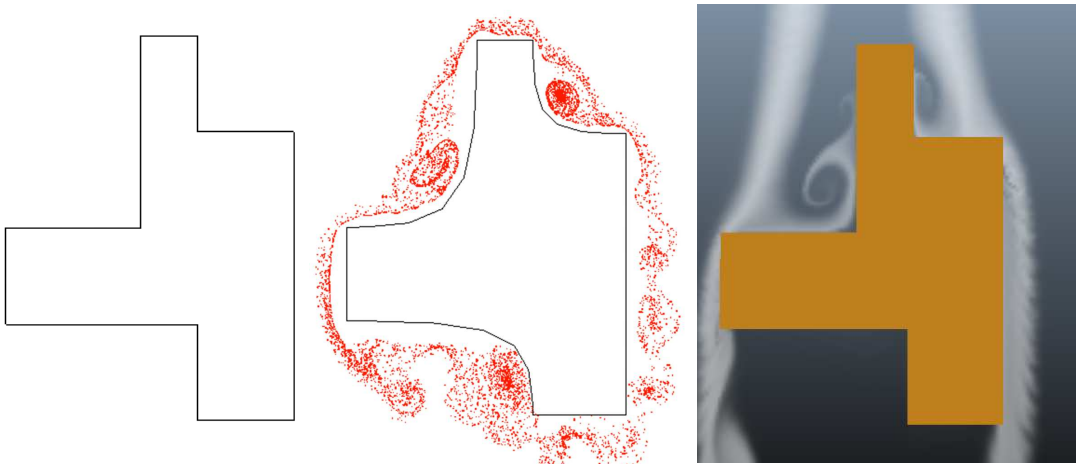
$$\mathbf{u}_{adv} = \begin{cases} -\kappa\hat{\mathbf{n}} & \text{if } \kappa < 0 \\ 0 & \text{otherwise.} \end{cases} \tag{5.19}$$

Here κ is the surface mean curvature. At each time step we compute the mean curvature for each point and advect the surface using a fictional time step. We repeat this process until the curvature magnitude is small enough. We obtain plausible results by setting a halt condition for $|\kappa| \leq \epsilon^{-1}$, where ϵ is the radius of the shed vortices. An example of simulation results employing this technique and a comparison with a standard grid solver are shown in Figure 5.16.

5.4 Chapter Summary and Discussion

In this chapter we have presented a model for simulating flows bounded by rigid solid objects. In this model we enforce the no-penetration boundary condition on the solid surface by representing it as a source sheet. This sheet is discretized using panel methods to find the required source strength that cancels the flux through the solid surface. We have coupled this simulation with a boundary layer model which is simulated by generating vorticity at the solid surface that constrains the tangential flow velocity at the solid boundary, reproducing viscous effects at the solid surface. This vorticity can be scaled to allow representing diverse slip conditions producing visually realistic flows around solid objects as demonstrated in our examples.

We have discussed the case of flow around solid objects with concave areas, which is a major challenge in the application of source panel methods. We have shown that our boundary layer simulation is a key tool to overcome the challenges in realistically simulating flow in concave areas, which is complemented with a simple and effective



(a) (b) (c)

Figure 5.16: Flow results around concave solid object. (a): Original concave object shape. (b): Flow around advected concave solid object. (c): Reference simulation employing a grid method using the Autodesk Maya^{TM} fluid simulator. We observe that confined eddies are formed in concave areas facing the flow in both simulations.

surface advection method that enables obtaining realistic flows in high curvature concave portions of the solid surface.

In this chapter we have shown simulations with one-way solid to fluid coupling. In the next chapter we discuss how to compute the force and torque exerted by the fluid on a solid object. This produces the required information to model two-way solid-fluid coupling allowing to update the trajectory of the solid object.

Chapter 6

Force Model

In previous chapters, we have discussed the simulation of inviscid flows around a solid object, to which we add the viscous effects of the boundary layer at the solid surface. This corresponds to one-way, solid to fluid coupling, *i.e.*, the solid object acts as an obstacle to the flow. In this chapter we focus on calculating the force that the fluid exerts on the solid object to simulate two-way solid-fluid coupling.

As discussed earlier in Chapter 4, one of the advantages of vortex methods is that there is no need to solve for pressure as this term is absent from the vorticity transport equation. However pressure at the solid surface is required to compute the force that the fluid exerts on the solid object. Then we need to calculate pressure from the quantities we are employing for simulation, *i.e.*, velocity and vorticity.

We introduce two methods for computing the fluid force acting on solid objects. The first is based on the non-standard theory of aerodynamic forces developed by Wu [157] for general shape bodies, or *bluff* bodies. Here the force acting on a solid object is expressed as a function of the variation of the moment of the flow vorticity, which is a global quantity on the simulation space. We demonstrate this technique with a simple experiment simulating real fluid behavior and we discuss the applicability of this method for computer animation.

The second method computes the pressure force considering only local flow at the surface. This method is based on the representation and simulation of the boundary layer. The generation and later separation of the boundary layer from the solid surface produces a pressure gradient that induces force and torque on the solid object. The surface pressure is related to the rate of vorticity generation, or *vorticity flux* at the surface. This property of flows is attractive to be used in our framework as it allows calculating forces only

considering local flow properties around the solid object. We develop our method based on this principle, which we use as our main simulation technique for achieving two-way solid-fluid coupling.

6.1 Integral Force on Solid Objects

One definition of the net force and torque acting on a solid object is given by Wu [157]. Here force and torque are related to the moments of vorticity on the flow. This idea is motivated by the lack of understanding on the applicability of circulation around a solid object to general bluff and 3D objects. Common assumptions in aerodynamic design, such as the existence of a trailing edge in airfoils do not apply to the case of general bluff bodies. Bluff bodies are of great interest in computer graphics and we introduce this approach as a simple method to compute fluid forces on general solid objects.

The first moment of vorticity $\boldsymbol{\alpha}$ of a flow is a global vector that is defined as follows:

$$\boldsymbol{\alpha} = \int_{Fluid} \mathbf{x} \times \boldsymbol{\omega} \, d\mathbf{x}. \tag{6.1}$$

This equation can be directly discretized using our Lagrangian vortex particle representation as follows:

$$\boldsymbol{\alpha} = \sum_j \mathbf{z}_j \times \boldsymbol{\omega}_j. \tag{6.2}$$

Wu [157] shows that the total aerodynamic force acting on the center of mass of a system of N_S solids can be expressed in terms of the variation of the first moment of the flow vorticity as follows:

$$\mathbf{F} = -\frac{\rho}{2}\frac{d\boldsymbol{\alpha}}{dt} + \rho \sum_k^{N_S} \frac{d}{dt} \int_{S_k} \mathbf{u}_k dV. \tag{6.3}$$

Here $\mathbf{u}_k$ corresponds to the velocity of the k^{th} solid object. The first term on the right hand side of the above equation corresponds to the rate of change of the first moment of vorticity of the flow, whereas the second term represents the force due to the mass of fluid being displaced by each solid object S_k.

The above formula can be directly discretized using our discrete version of $\boldsymbol{\alpha}$ in Equation (6.2) for a rigid solid object as follows:

$$\mathbf{F} = -\frac{\rho}{2}\frac{\boldsymbol{\alpha}^{t=n} - \boldsymbol{\alpha}^{t=n-1}}{\Delta t} + \rho \sum_{k=1}^{N_S} V_k \frac{\mathbf{u}_k^{t=n} - \mathbf{u}_k^{t=n-1}}{\Delta t}, \tag{6.4}$$

where V_k represents the volume of the k^{th} solid object.

The net torque acting on the center of mass of the solid system is related to the second moment of vorticity, which is defined as follows:

$$\boldsymbol{\beta} = \int_{Fluid} \|\mathbf{x}\|^2 \boldsymbol{\omega} \; d\mathbf{x}. \tag{6.5}$$

This equation is discretized in a similar way as done for the first moment of vorticity:

$$\boldsymbol{\beta} = \sum_j \|\mathbf{z}_j\|^2 \boldsymbol{\omega}_j. \tag{6.6}$$

Wu [157] shows that the net torque acting on the center of mass of a system of solid objects can be related to the second moment of vorticity as follows:

$$\mathbf{T} = \frac{\rho}{2}\frac{d\boldsymbol{\beta}}{dt} + \frac{\rho}{2}\sum_{k=1}^{N_S} \frac{d}{dt}\int_{S_k} \|\mathbf{x}\|^2 \hat{\mathbf{n}} \times \mathbf{u}_k \; dS \tag{6.7}$$

We can discretize this equation using the discrete formulation of $\boldsymbol{\beta}$ in Equation (6.6) and approximating the value of the integral at the solid surface's face centers as follows:

$$\begin{aligned} \mathbf{T} &= \frac{\rho}{2}\frac{\boldsymbol{\beta}^{t=n} - \boldsymbol{\beta}^{t=n-1}}{\Delta t} + \frac{\rho}{2}\sum_{k=1}^{N_S} \frac{M_k^{t=n} - M_k^{t=n-1}}{\Delta t}, \quad \text{where} \\ M_k &= \sum_{faces} A_i \|\mathbf{x}_i\|^2 \hat{\mathbf{n}}_i \times \mathbf{u}_k \end{aligned} \tag{6.8}$$

Here $\mathbf{x}_i$ and A_i are the solid object's i^{th} face center and area, respectively. With Equations (6.4) and (6.8) it is possible to compute the net force and torque exerted by the fluid on a solid object, as we demonstrate in the next section.

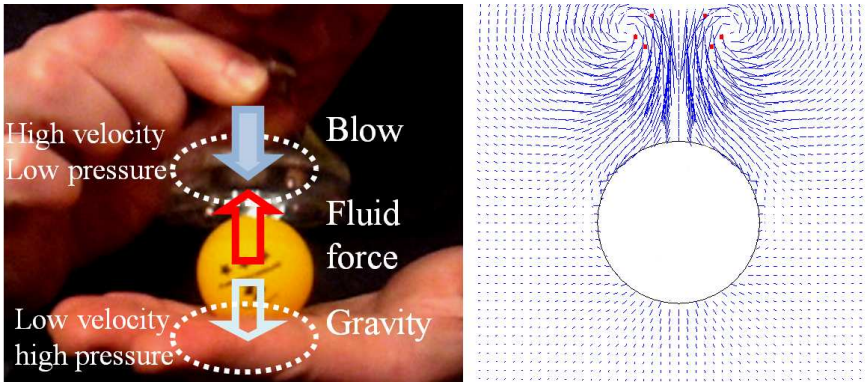


Figure 6.1: Left: Diagram showing the main phenomena involved in a real-life experiment. Air is blown through the funnel, increasing the velocity on top of a sphere. Below the sphere the velocity is low and pressure high according to Bernoulli's principle. Right: fluid velocity (in blue) induced by jet modeled with vortex particles (in red) around a sphere in the first frames of simulation. Images from [139], used with permission.

6.1.1 Simulation of Real Life Experiment

We test the above method by replicating a simple experiment with forces on a solid object in 2D. In an inviscid flow, the pressure force is directed from areas of high to low pressure. In an inviscid flow, the Bernoullli principle states that as the flow velocity increases, the fluid pressure decreases. An example of this relationship can be easily seen in steady flows, where the relationship between velocity and pressure is given by the Bernoulli equation:

$$\frac{\|\mathbf{u}\|^2}{2} + gz + \frac{p}{\rho} = constant. \tag{6.9}$$

Here g is the gravity acceleration magnitude, and z is the coordinate in the axis parallel to gravity. Bernoulli's principle also holds for unsteady flows and we test our presented approach for calculating fluid forces emulating the result of a demonstration of the Bernoulli experiment. Consider a sphere initially at rest. Air is blown towards the sphere in the direction of gravity as shown in figure 6.1. Pressure on top of the sphere drops due to the air velocity, and pressure below is higher. Therefore the fluid force on the sphere points upwards and as it is greater than gravity the sphere is lifted. Figure 6.2 shows the result of our experiment compared to a real-life experiment.

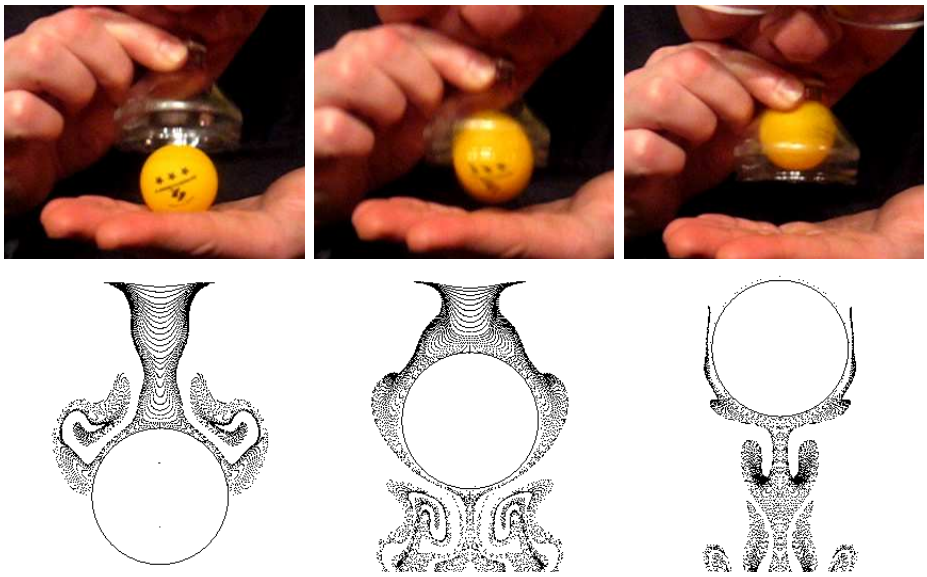


Figure 6.2: Top: Video frames of real life experience. Air is blown through the funnel decreasing pressure above the ball. Force due to high pressure in the bottom lifts the sphere. Bottom: Simulation frames. We reproduce the flow with a symmetric vortex jet and add marker particles for visualization.

6.1.2 Applicability

The simple experiment performed with this technique demonstrates its potential for reproducing realistic interactions between solids and fluids. There is a major limitation of this approach when the interest is to simulate several solid objects, since the computed force corresponds to that acting on the center of mass of the whole solid system.

When simultaneously simulating several solid objects this represents a problem as the equations for force calculation do not give values for the net force acting on each individual solid object. A possible solution would be to define a control volume around each solid object and calculate the change of vorticity moment due to interaction with solid objects plus the flux of vorticity through the control surface as proposed by Ploumhans *et al.* [110].

A major problem with this approach is that defining an appropriate control volume for each object may not be a trivial task. A naïve approach would be to consider the bounding box of the solid object, however, interaction of the bounding boxes of different solid objects may have to be considered if these objects translate and rotate freely and approach each other such that their respective bounding boxes overlap.

For simulating several rotating and translating solid objects it would be desirable to calculate fluid forces acting on each based on the local flow at each solid's surface. This would enable simulating several solid objects simultaneously. In the next section we present an approach with these characteristics. We introduce a method based on computing the pressure gradients caused by the boundary layer separation from the solid surface. This method follows naturally from our boundary layer flow model and we employ this as our main technique for two-way solid fluid coupling.

6.2 Pressure Force at a Solid Object Surface

Fluid forces on the solid surface are a consequence of the generation and evolution of the boundary layer [111]. *D'Alembert's paradox* [34], states that the net force on a solid object calculated from the equations for an inviscid and irrotational flow is zero. This means that an object, for instance a sphere, under a uniform flow would not experience any drag from the flow, which contradicts experimental data. As shown by Prandtl [111], this contradiction is caused by assuming the flow is inviscid, and it led to the development of the *Boundary Layer Theory*.

The boundary layer that adheres to the solid surface eventually separates from the solid object, and enters the main flow introducing vorticity as described in Section 5.2.4. This vorticity induces an increased velocity past the solid object and therefore a low pressure wake, according Bernoulli's principle (see Section 6.1.1). Examples of uniform velocity fields past a solid object with and without flow separation are shown in Figure 6.3.

Fluid forces acting on a solid object can be precisely characterized in terms of the vorticity flux from the solid surface into the main flow [84, 156]. In our scenario the vorticity flux corresponds to the amount of vorticity created at a solid surface, which is introduced to the main flow as vortex shedding. We employ the characterization of the fluid force acting on a solid object in terms of shed vorticity to develop our approach as described in the following section.

6.2.1 Vorticity Flux

We discussed vortex shedding in Section 5.2.4 as the process where vorticity which is generated at the solid surface is introduced in the main flow. The rate at which vorticity is introduced into the main flow corresponds to a vorticity flux that emanates from

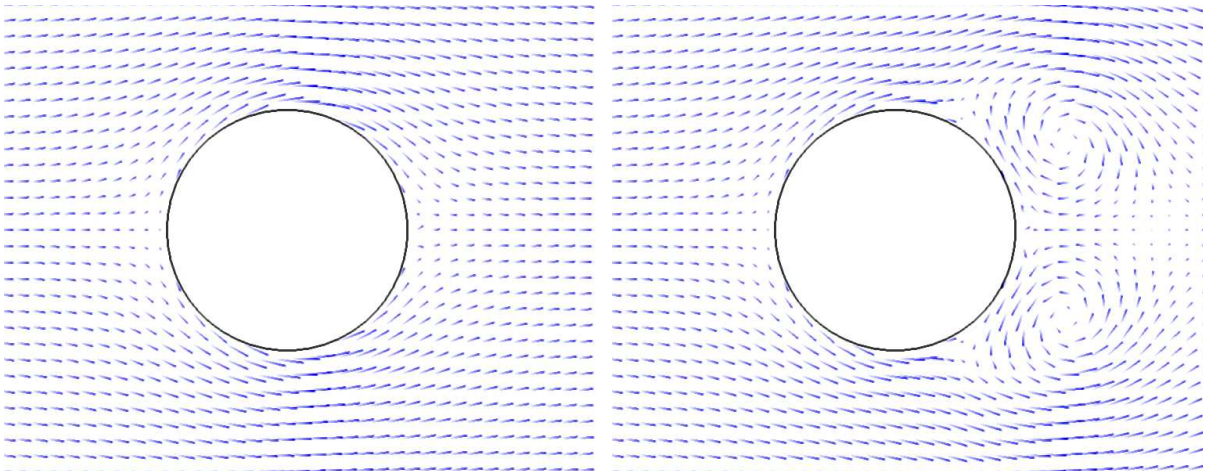


Figure 6.3: Example velocity field plot of a potential flow around a sphere (left) and flow with boundary layer separation (right).

the surface. This flux value is related to the pressure at the solid surface. Below we present this relationship. The vorticity flux is realized in our simulations through vortex shedding, therefore, computing pressure from the fluxed vorticity fits naturally in our framework.

In the boundary layer, fluid motion is dominated by viscosity and then the vorticity transport is characterized by the following diffusion equation:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = \nabla \cdot (\nu \nabla \boldsymbol{\omega}). \tag{6.10}$$

Here ν is a viscosity coefficient. The emitted vorticity from the solid object depends on the tensor $\mathbb{J} = \nu \nabla \boldsymbol{\omega}$, particularly on its normal component, which is the vorticity flux $\boldsymbol{\sigma}$ [32]:

$$\boldsymbol{\sigma} = \hat{\mathbf{n}} \cdot \mathbb{J} = \nu \hat{\mathbf{n}} \cdot \nabla \boldsymbol{\omega} = \nu \frac{\partial \boldsymbol{\omega}}{\partial \hat{\mathbf{n}}} \tag{6.11}$$

The vorticity flux $\boldsymbol{\sigma}$ from a solid surface can be expressed in terms of other physical quantities. As described by Wu and Wu [156] the three main contributions to $\boldsymbol{\sigma}$ are given by: (a) the solid object's acceleration and body forces, (b) the surface pressure, and (c) the viscous drag. In the inviscid limit, the vorticity flux can be expressed concisely as follows:

$$\begin{aligned} \boldsymbol{\sigma} &= \boldsymbol{\sigma}_a + \boldsymbol{\sigma}_p \quad \text{with:} \\ \boldsymbol{\sigma}_a = \hat{\mathbf{n}} \times (\mathbf{a} - \mathbf{f}) \qquad & \boldsymbol{\sigma}_p = \frac{1}{\rho}\hat{\mathbf{n}} \times \nabla p \end{aligned} \tag{6.12}$$

Here $\mathbf{a}$ is the solid acceleration, $\mathbf{f}$ is the acceleration due to external body forces acting on the solid and fluid such as gravity. Equation (6.12) is obtained from the definition of vorticity flux on a general viscous fluid and then applying the inviscid limit. A detailed derivation is included in Appendix D. We observe that the pressure gradient appears on the term $\boldsymbol{\sigma}_p$. This is the key quantity we employ to compute fluid forces on the solid object. We describe how to employ the vorticity flux for force calculation in the following section.

6.2.2 Forces from Vorticity Flux

Computing the surface pressure from Equation (6.12) is not trivial. Moreover, it is necessary to know the value of vorticity flux $\boldsymbol{\sigma}$ at the solid surface.

Wu and Wu [156] show a direct way to evaluate the force due to pressure at the solid surface from the moments of $\boldsymbol{\sigma}_p$ using the generalized Stokes' Theorem. In the inviscid limit, the force acting on the solid object is formulated as follows:

$$\begin{aligned} \mathbf{F} = -\int_{\mathcal{S}} p\hat{\mathbf{n}} ds &= -\frac{1}{2}\int_{\mathcal{S}} \mathbf{x} \times (\hat{\mathbf{n}} \times \nabla p) ds \\ &= -\frac{1}{2}\int_{\mathcal{S}} \rho \mathbf{x} \times \boldsymbol{\sigma}_p \, ds. \end{aligned} \tag{6.13}$$

Similarly, the torque acting on the solid object with respect to its center of mass $\mathbf{x}_{CM}$ due to surface pressure can be expressed in terms of the second moment of $\boldsymbol{\sigma}_p$ as follows:

$$\mathbf{T} = -\int_{\mathcal{S}} (\mathbf{x} - \mathbf{x}_{CM}) p\hat{\mathbf{n}} ds = -\int_{\mathcal{S}} \left(\frac{\rho}{2} \left\| \mathbf{x} - \mathbf{x}_{CM} \right\|^2 \boldsymbol{\sigma}_p \right) ds. \tag{6.14}$$

Here $\boldsymbol{\sigma}_p$ corresponds to the definition in Equation (6.12). Missing still is the vorticity flux at the solid boundary. A common approximation [21, 156, 37] for vorticity flux at the no-slip solid boundary is:

$$\boldsymbol{\sigma} = -\frac{\gamma}{\Delta t} \tag{6.15}$$

Here $\boldsymbol{\gamma}$ is the vorticity of the surface vortex sheet. This approximation comes from the fact that, at each time step, the amount of vorticity introduced in the flow needs to cancel the slip velocity. This vorticity, in turn, is determined by the vortex sheet. With the approximation in Equation (6.15), we can easily calculate the fluid force value based on the known quantities of solid acceleration, body forces and surface vorticity as follows:

$$\boldsymbol{\sigma}_p = \frac{\boldsymbol{\gamma}}{\Delta t} + \boldsymbol{\sigma}_a. \tag{6.16}$$

We replace $\boldsymbol{\sigma}_p$ by the above expression in Equation (6.13) obtaining the following definition for fluid force acting on a solid object:

$$\begin{aligned} \mathbf{F} &= -\frac{1}{2}\int_{\mathcal{S}} \rho \mathbf{x} \times \boldsymbol{\sigma}_p \\ &= -\frac{1}{2}\int_{\mathcal{S}} \rho \mathbf{x} \times \left(\frac{\boldsymbol{\gamma}}{\Delta t} + \boldsymbol{\sigma}_a\right)\, ds \\ &= -\frac{1}{2}\int_{\mathcal{S}} \rho \mathbf{x} \times \left(\frac{\boldsymbol{\gamma}}{\Delta t} + \hat{\mathbf{n}} \times (\mathbf{a} - \mathbf{f})\right)\, ds. \end{aligned} \tag{6.17}$$

Here we have used $\boldsymbol{\sigma}_a = \hat{\mathbf{n}} \times (\mathbf{a} - \mathbf{f})$ from Equation (6.12). We calculate the force and torque using the panel discretization. We evaluate the moments using the control points $\mathbf{x}_j$ of each panel described in Section 5.1.3. In our framework, we allow simulating different slip conditions, by scaling the amount of shed vorticity using a slip constant c (see Section 5.2.4). Therefore we incorporate this scaling factor into the vorticity flux calculation at each boundary panel. Then, we compute the fluid force as follows:

$$\mathbf{F} = -\frac{1}{2}\sum_{j=1}^{N_p} \rho \mathbf{x}_j \times \left(c\frac{\boldsymbol{\gamma}_j}{\Delta t} + \hat{\mathbf{n}}_j \times (\mathbf{a} - \mathbf{f})\right) A_j. \tag{6.18}$$

Here N_p corresponds to the number of panels of the solid object. Using the same algebraic manipulations we obtain the following discrete expression for torque:

$$\mathbf{T} = -\frac{1}{2}\sum_{j=1}^{N_p} \rho \left\|\mathbf{x}_j - \mathbf{x}_{CM}\right\|^2 \left(c\frac{\boldsymbol{\gamma}_j}{\Delta t} + \hat{\mathbf{n}}_j \times (\mathbf{a} - \mathbf{f})\right) A_j. \tag{6.19}$$

We compute force and torque on a solid object with these formulas and we employ these values to update its velocity and position. We update the motion of a rigid solid using a direct Euler time integration step.

An example of the results obtained using this method for calculating forces is presented in Figure 6.4, where a sphere initially in free fall interacts with two vortex jets. The first propels the sphere upwards and the second hits the sphere sideways altering its trajectory.

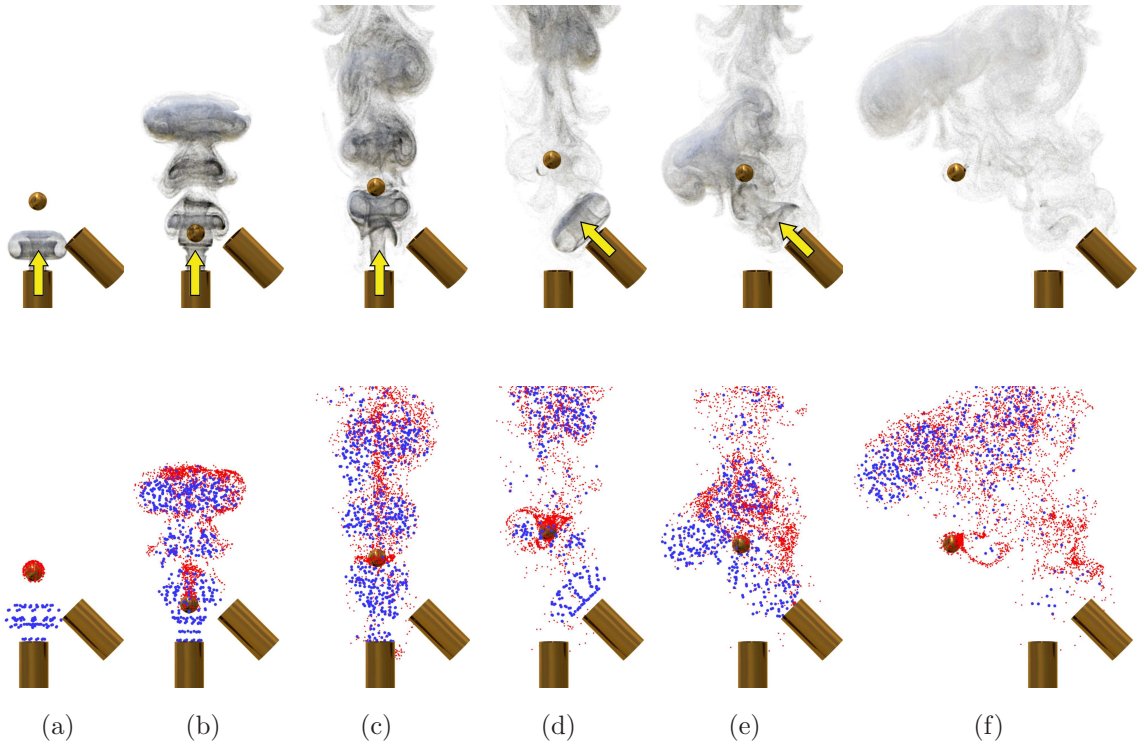


Figure 6.4: Example of interaction between a vortex jet and a sphere under the external action of gravity. The sphere is lifted by a vertical jet and then hit by a smoke jet from right to left. Top row shows the scene with rendered smoke, and bottom view shows only the vortices. Jet vortices are colored in blue and vortices generated as vortex shedding are colored in red. (a) Sphere starts falling and vertical vortex jet is emitted. (b) Vertical jet reaches falling sphere. (c) Vertical jet lifts sphere due to drag. (d) Vertical jet is turned off and sideways jet approaches the sphere from right to left. (e) Sideways jet reaches the sphere. (f) Sphere is propelled sideways as consequence of drag induced by the second jet. An animation sequence for this figure is shown in the accompanying video at 1:50.

6.3 Chapter Summary and Discussion

In this chapter we have presented two methods for computing the fluid forces acting on solid objects based on the representation that we use for the flow only in terms of its velocity and vorticity. The first is based on the relation between the rate of change of the flow's vorticity moment and the net force acting on a system of solid objects. As discussed in Section 6.1.2, a direct application of this technique would be limited to a single solid object. Simulating several solid objects would require tracking control volumes where the change of vorticity moments is measured for each solid object, and track any changes on these volumes depending on the motion of solid objects relative to others.

The second is based on the relationship between rate of vorticity generation at a solid surface and the pressure gradient. This technique has the advantage of computing fluid forces only based on vortex shedding, which is a local phenomenon at each solid surface, and therefore it can be applied directly to simulation of several solid objects on the same scene. This method is flexible enough to be applied in computer graphics simulations and we employ this as the force model for generating our results.

In our approach to compute two-way coupling of solids and fluids, at each time step we solve the fluid and solid object's motion in different substeps. Here the object is not static, as the solid object's velocity is accounted for in our panel calculations and boundary layer model, specifically in the left hand side of equations (5.7), (5.15) and (5.16). Our method to compute force and torque around each object depends on the vorticity flux, *i.e.*, the rate of vorticity creation at the solid surface. The flow calculated around the solid object in one substep determines the induced vorticity flux and therefore the force exerted by the fluid on the solid, which is calculated in a later substep.

In the next chapter we present additional visual results of our simulation method and their performance. We also present visual comparisons to fluid simulation results presented by others and we compare our numerical approximations to physical models.

Chapter 7

Additional Results and Validation

In the previous chapters we have described our methods to simulate the full interaction of flows and rigid solid objects. In this chapter we summarize the different steps of our simulation system and we provide additional visual results using our techniques. We discuss performance and the factors that have the greatest impact on efficiency.

We validate our methods by two means: First we compare our results with equivalent simulations presented previously in the computer graphics literature. Second, we evaluate the accuracy of our method, particularly of our choice of kernel, by comparing with known solutions in the field of engineering.

7.1 System Overview and Implementation Details

Our scenes are composed of two main elements, solid objects and vortex particles. Additionally we employ massless marker particles to indicate the location of smoke, which are used only for rendering and have no other effect on the physics of the simulation.

The flow velocity is calculated as the sum of the velocity induced by the vortices, the user-defined external flow and the surface-induced velocity (see Section 5.1). The velocity of each solid object is determined by the sum of the external forces (*e.g.*, gravity) and the fluid forces acting on the solid object.

We build our fluid simulator in C++ using the Exocortex^{TM} development platform, which implements multi-threaded and registry-optimized data structures, as well as space partitioning and search functions. This platform provides basic representations of solids which are extended with the elements presented in previous chapters (*e.g.*, source and vortex sheets). This system also implements functions for importing and exporting simu-

lation data in different graphics formats for later use in standard animation and rendering software.

Given the positions and velocities of the solids and particles, our simulator performs the following operations for each time step:

- Compute the flow velocity (Section 4.2).
- Enforce no-slip boundary condition: compute the vortex sheet (Section 5.2.3) and vortex shedding (Section 5.2.4).
- Enforce the no-penetration boundary condition (Section 5.1.3) using an updated velocity field with above shed vortices.
- Advect vortices and smoke particles.
- Compute forces on the solid object (Section 6.2.2) and update the solids objects' velocities and positions.

Once the simulation data is generated, it is imported into standard commercial tools for rendering. We render our scenes within Autodesk Softimage^{TM} using Mental Ray^{TM} for solids and Exocortex Fury 2^{TM} for fluid particles. The latter produces an image of smoke using the solid as an alpha mask, therefore both solid and smoke renderings must be composed in a post-production step. This is done using The Foundry Nuke^{TM}.

7.2 Additional Simulation Results and Performance

We show simulation results for fluid solid interplay in 3D using a sphere, a cylinder and Stanford bunny models. To increase performance in panel calculation and vortex shedding, highly detailed surfaces can be simplified into coarser geometries. By preserving visually important features of the original mesh, flow patterns that are consistent with these features can be employed directly in rendering. In our simulations we we employ 80 triangular panels for the sphere, 180 for the cylinder and 500 for the bunny model.

Figure 7.1 presents the interaction between a vertical vortex jet and a bunny model. This is a similar situation as in our previous example of a sphere propelled by jets shown in Figure 6.4. A bunny model in free fall interacts with a vertical ascending jet, which propels the bunny upwards, with rotation due to the torque exerted on the solid.

Figure 7.1: Example of interaction between a vortex jet and a bunny model. An animation sequence for this figure is shown in the accompanying video at 3:29.

In Figure 7.2 we show a comparison of vortex shedding patterns induced by two different solid objects in free fall. Here both solids fall through a sheet of smoke. Shed vortices in the solid objects' wake induce different patterns of rotational motion in the smoke sheet.

We show an example of the interaction of a vortex jet with a large plane in Figure 7.3. Here we simulate a rocket exhaust interacting with a large ground plane that prevents the flow to go around the plane. We achieve nevertheless a visually realistic result. A 2D schema of a vortex jet interacting in a similar way with a large planar surface is shown in Figure 7.4.

We run our experiments on a Intel Core i7-2600, 3.4GHz CPU with 16 GB RAM. We summarize our performance results for each of our scenes in Table 7.1. Memory use by our method is generally negligible (less than 5 MB on average when excluding marker particles). In our simulations we apply a vorticity dissipation step as described in Section 4.3.3 and we eliminate vortices whose strength falls below a user-defined threshold.

Most of the computational cost is associated with the evaluation of the velocity field, hence performance is strongly dependent on the amount of vortices in the simulation. Since the complexity of evaluating the velocity field is $O(nk)$ as explained in Section 4.2, we could improve computation times by reducing the kernel radii. A comparison of performance for simulations with different kernel radii is shown in Table 7.2. Here we simulate a cloud of smoke for 300 frames where vortices are confined to a unit radius

Figure 7.2: Simulation snapshots of different patterns of vortex shedding produced on a sheet of smoke by a bunny model and a sphere. An animation sequence for this figure is shown in the accompanying video at 0:05.

sphere, with 24 vortices added at each 20 frames, totalling 360 vortices for the whole simulation. We observe the expected increase of performance as the radius is reduced.

However, as shown in Figure 7.5, reducing the kernel radius is a naïve solution since this leads to a gross visual degradation of results: As the radius decreases, the magnitude of the influence vortices on each other's also decreases and therefore the velocity magnitude as well.

Furthermore, as explained later in this chapter, a kernel radius that is too small induces an underestimation of the velocity field, especially when this process is performed on the kernel radius in panel evaluation. Here accuracy is the main affected feature in the simulation.

Unlike other particle simulations such as SPH, vortex radii need to be large in vortex methods which impacts performance. A robust solution to increased performance would address directly the complexity of the underlying N-body problem. *Fast Multipole Methods* (FMM) are a large class of methods that produce controllable approximations for

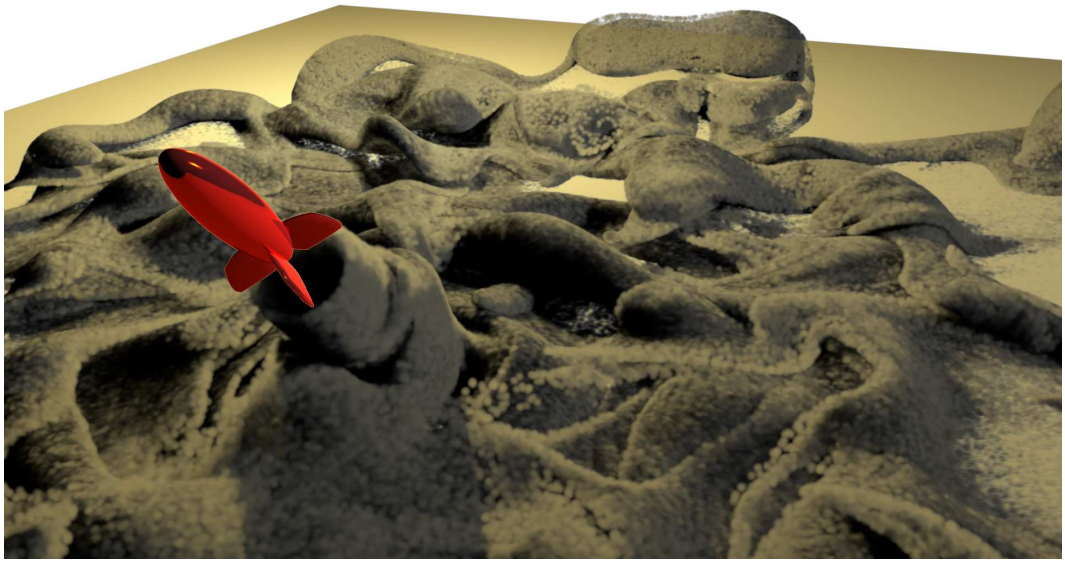


Figure 7.3: Simulation of rocket exhaust, an example of a vortex jet interacting with a large planar object. An animation sequence for this figure is shown in the accompanying video at 3:37.

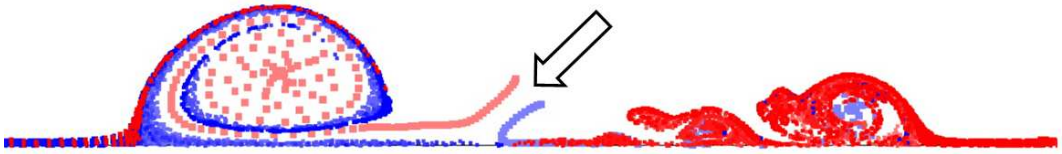


Figure 7.4: Example of 2D jet (indicated with an arrow) interacting with a large planar wall with vortex shedding. Vortices are colored by their spin: blue indicates vorticity points towards the reader and red points in the opposite direction.

solving N-body problems. It is possible to implement our techniques with such a method, achieving $O(n)$ performance employing a variant of the Black Box FMM (bbFMM) [44]. Arbitrary, non oscillatory kernels such as the one we employ are approximated through interpolation using a truncated series of Chebyshev polynomials and these are employed to perform fast summations on an octree structure. We describe a variation of the bbFMM [44] for our scenario in Appendix E.

FMM have been previously employed [105], producing a $O(nlog(n))$ algorithm. As opposed to previous work on hybrid simulations [107] where velocity evaluation is performed by interpolation between particles and the grid, we evaluate the field directly from particles.

Table 7.1: Performance results for 100 frames of simulation.

		Vortices			AVG
Scene	**Figure**	**Jet**	**Shed**	**Total**	**FPS**[a]
Smoke column	4.5	480	0	480	71.5
Smoke around sphere	5.2	128	2000	2128	4.68
Smoke around cylinder	5.7	100	4988	5088	0.6904
Smoke around bunny	5.8	154	39276	39430	0.016
Bouncing sphere	6.4	906	2438	3344	1.4315
Bouncing bunny	7.1	884	45530	46414	0.0114
Falling objects	7.2	0	56802	56802	0.0007
Rocket exhaust	7.3	2892	1683	4576	0.84

[a]We do not include marker particles used for rendering.

Table 7.2: Performance results for different kernel radius.

Radius	2.5	2.0	1.5	1.0	0.5
FPS	56.2	55.4	58.8	94.2	189.6

Then the main simulation parameters that control the visual aspect of the flow when interacting with solid objects are the slip coefficient and the vortex kernel radius. The slip coefficient determines the amount of turbulence and therefore the most prominent visual features of our simulations as demonstrated in our example of different flows around a cylinder shown in Figure 5.7. The kernel radius controls the general flow shape, as shown in Figure 7.5 and it is key to provide visually rich simulation of flows, regardless of the turbulence generated at solid surfaces.

In the following section we compare our results with previous simulations produced by other authors in the literature.

7.3 Comparison with Previous Work

We compare the visual quality of our unbounded smoke simulation with results obtained by Weißmann and Pinkall [149], shown in Figure 7.6. We obtain a smoke cloud with the general shape that breaks down into an irregular, turbulent flow with similar rotational patterns as the reference simulation. Our results are obtained using a simple particle

(a) (b) (c) (d)

Figure 7.5: Smoke simulation results for different kernel radius ϵ after 300 iterations. (a): $\epsilon = 2.5$; (b): $\epsilon = 2.0$; (c): $\epsilon = 1.5$; (d): $\epsilon = 1.0$. Smoke's emission radius is 1.0.

representation, as opposed to the simulation of 3D filaments by Weißmann and Pinkall [149]. In their approach, vorticity is represented by filaments, *i.e.*, curves in space. Their method requires integration along 3D curves, and a variational approach to connect filaments at run-time.

We compare our visual results for vortex shedding with those obtained by Weißmann and Pinkall [149], shown in Figure 7.7. Here we reproduce the visual effect caused by vortex shedding resulting from a vortex jet interacting with an ellipsoid, obtaining a highly detailed turbulent flow around the ellipsoid.

We compare the results of coupling solids and fluids by simulating the interaction of two vortex jets with a bunny model, reproducing results obtained by Klingner *et al.* [76] using tetrahedral meshes. Our method reproduces realistic motion of the bunny agreeing with the previously published results as shown in Figure 7.8.

Figure 7.9 shows an example of different patterns in vortex shedding around a sphere, changing the slip coefficient, reproducing previously published results [105]. However, with our methods we can also realistically simulate the sphere motion induced by the flow through our force computation model, unlike any other previously published work in vortex methods for computer graphics [105, 5, 149].

Finally, we compare our results with those of previously published vortex particle methods. Figure 7.10 shows turbulence produced due to the interaction of a uniform

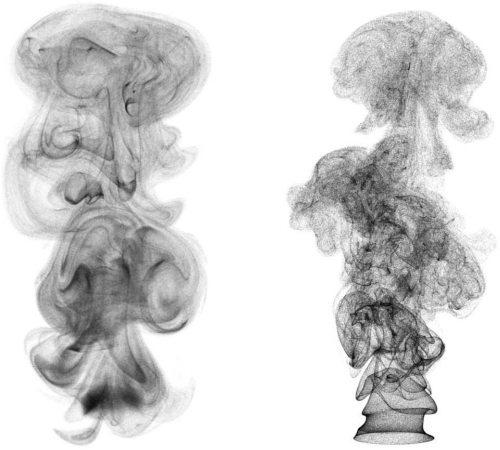


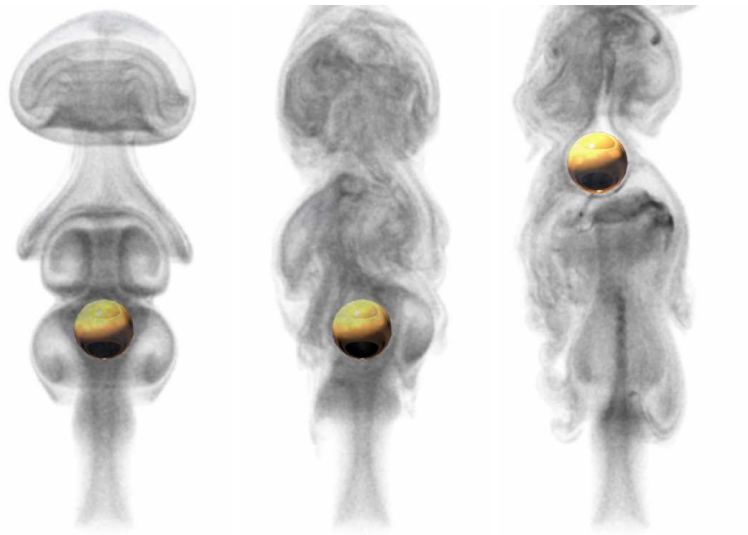
Figure 7.6: Left: Results of our smoke simulation using a vortex particle jet. Right: results using vortex rings from Weißmann, S. and Pinkall, U. "Filament-based smoke with vortex shedding and variational reconnection," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2010, Vol. 29:4. © 2010 ACM, Inc. http://doi.acm.org/10.1145/1778765.1778852.

Figure 7.7: Left: Our results of smoke around an ellipsoid with vortex shedding. Right: Results for a similar scene from Weißmann, S. and Pinkall, U. "Filament-based smoke with vortex shedding and variational reconnection," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2010, Vol. 29:4. © 2010 ACM, Inc. http://doi.acm.org/10.1145/1778765.1778852.

Figure 7.8: Left: Our results of coupling a light bunny with two vortex jets. An animation sequence for this figure is shown in the accompanying video at 2:22. Right: results of a similar scene in Klingner, B. M., Feldman, B. E., Chentanez, N. and O'Brien, J. F. "Fluid animation with dynamic meshes," ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2006, Vol. 25:3. © 2006 ACM, Inc. `http://doi.acm.org/10.1145/1141911.1141961`.

flow and a static solid wedge. This example has been used to demonstrate anisotropic turbulence generated in areas of high turbulent kinetic energy [107] (Fig.7). The approach by Pfaff *et al.* [107] requires simulating turbulent kinetic energy evolution along the flow and inserting noise particles corresponding to turbulence. In contrast, we do not need an energy evolution simulation to generate turbulence in the same regions. Moreover, Figure 7.10 shows a clear perdiodic pattern of turbulence consistent with a von Kármán vortex street (see Figure 5.5), unlike prior work by Pfaff *et al.* [107]. Additionally we employ the same example to show more complex interaction, by adding a sphere which is dragged up the ramp, producing a consistent wake of turbulence.

From these examples we observe that our simulation method based on vortex particles can be effectively used to produce visually rich interaction between solids and fluids. In



(a) (b) (c)

Figure 7.9: Flow around a unit radius sphere: (a) using free-slip condition; (b) using no-slip condition and finally (c) sphere dragged by the flow. An animation sequence for this figure is shown in the accompanying video at 2:45.

the next section, we evaluate our approximations at the solid surface by comparing our numerical results with those used as reference in engineering applications.

7.4 Comparison with Physical Models

Our simulation system produces visually realistic results, which is the goal in fluid simulation for computer graphics. Coupling solids and fluid requires obtaining good approximations of the fluid force acting on a solid object. We calculate forces on solid objects based on the flux of vorticity produced at the solid surface. This quantity is determined by the approximation results obtained using panel methods, therefore we focus our attention to evaluating the numerical results we obtain by approximating panels using our kernels. We compare the results we obtain with the analytical solution for a sphere. We perform this validation in 2D to facilitate representation and highlight the main considerations regarding our methods.

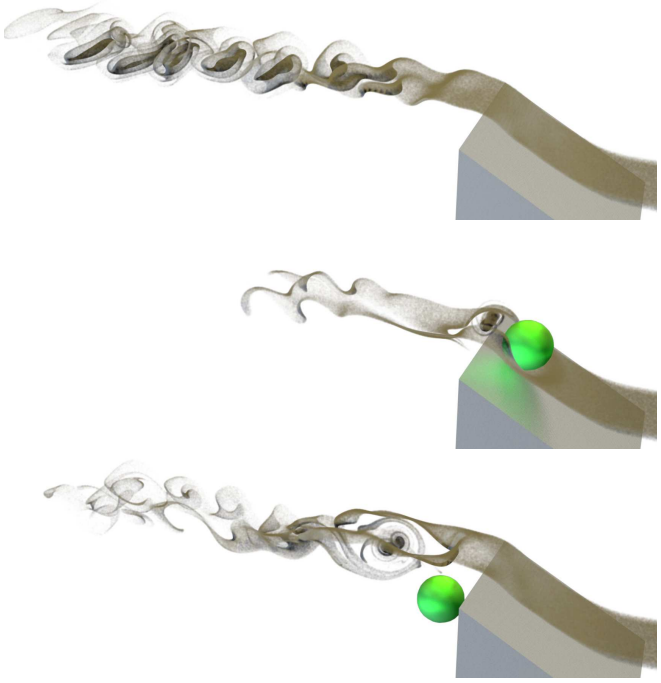


Figure 7.10: Top: Simulation of uniform flow past a wedge as presented in [107]. Here we use a slip coefficient $c = 0.5$. A periodic pattern of shed vorticity forms past the solid. Middle and bottom: same scene with a unit radius sphere with the same slip coefficient as the ramp, dragged by the flow.The sphere falls due to gravity as it reaches the top of the ramp. A consistent wake of vorticity is formed past the sphere. An animation sequence for this figure is shown in the accompanying video at 3:03.

7.4.1 Pressure Coefficient

As a main measuring tool we employ the *pressure coefficient* at the solid surface [78]. This is a measurement of the relative pressure at the solid surface with respect to a reference pressure value and it is defined as follows:

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty \|\mathbf{u}_\infty\|^2}. \tag{7.1}$$

Here $\mathbf{u}_\infty$, p_∞ and ρ_∞ correspond to the flow velocity, pressure and density far away from the solid object, respectively. This coefficient is computed from the Bernoulli Equation (6.9) considering the following [78]:

$$p + \frac{1}{2}\rho\|\mathbf{u}\|^2 = p_\infty + \frac{1}{2}\rho_\infty\|\mathbf{u}_\infty\|^2. \tag{7.2}$$

From the above equation, the following expression for $p - p_\infty$ is obtained:

$$p - p_\infty = \frac{1}{2}\rho_\infty \|\mathbf{u}_\infty\|^2 - \frac{1}{2}\rho \|\mathbf{u}\|^2. \tag{7.3}$$

Replacing the above in Equation (7.1) we obtain the following:

$$C_p = 1 - \left(\frac{\|\mathbf{u}\|}{\|\mathbf{u}_\infty\|}\right)^2. \tag{7.4}$$

This is the indicator we use to evaluate the results obtained using our panel methods. As indicated in Equations 5.15 and 5.16, the vortex panel velocity is calculated from source panel velocity, and this is the main indicator we employ in our measurements.

7.4.2 Evaluation on a Sphere

In the case of a sphere the distribution of the pressure coefficients C_p, or *pressure profile* can be determined analytically on the sphere surface and it corresponds to:

$$C_p = 1 - 4sin(\theta)^2. \tag{7.5}$$

A diagram with the analytical pressure coefficient values for the sphere as a function of the angle θ is presented in Figure 7.11.

We first evaluate the velocity field generated by our choice of finite kernel as defined in Equations (4.6) and (4.7) for source panels as in Equation (5.8). Then we evaluate the velocity around the solid surface. It is evident that using a kernel radius ϵ that is too small would distort the pressure coefficient results, by reducing their magnitude, as shown in Figure 7.12. One issue with this choice of kernel is that by increasing ϵ, the resulting flow velocity at the surface increases beyond the analytical solution. This is due to the particular shape of the kernel function used, and its evolution as radius changes, as shown in Figure 7.13. By increasing the radius the region of higher values for the kernel function also increases. Therefore, the velocity magnitude induced by a vortex at a given point increases as the kernel radius does.

Since the total surface velocity is calculated as the sum of the velocities induced by each panel (Equation (5.8)), therefore, an overestimation of the velocity induced by each panel due to the kernel evaluation leads to increasing velocity values, altering the pressure coefficients.

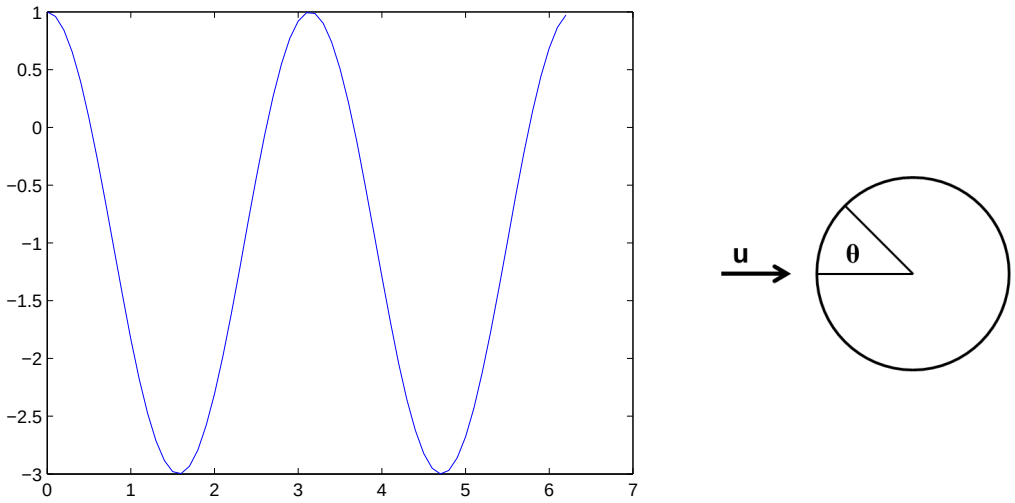


Figure 7.11: Pressure coefficient distribution on a 2D sphere surface. The Y-axis represents the value of C_p and the X-axis, the angle θ.

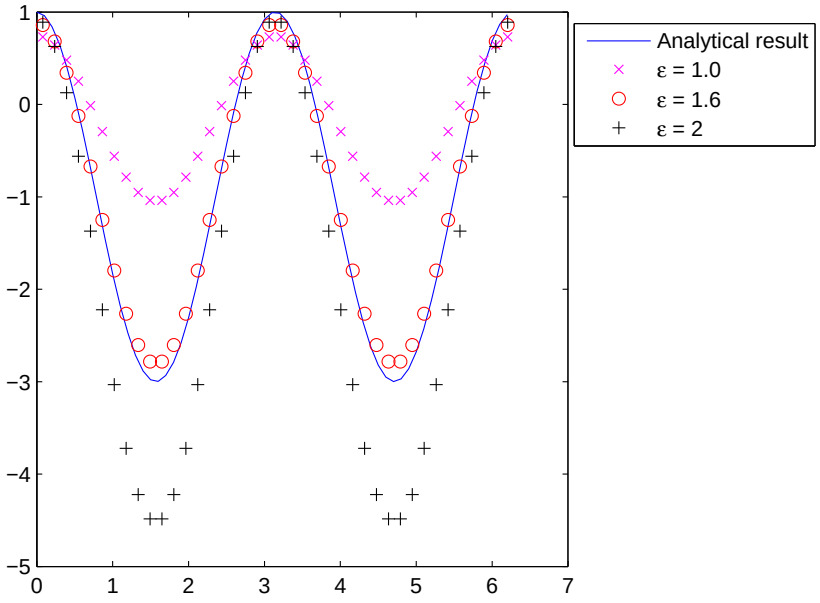


Figure 7.12: Pressure coefficients obtained using our polynomial finite kernel on a unit sphere for different radii ϵ. The pressure coefficients are computed using 40 panels.

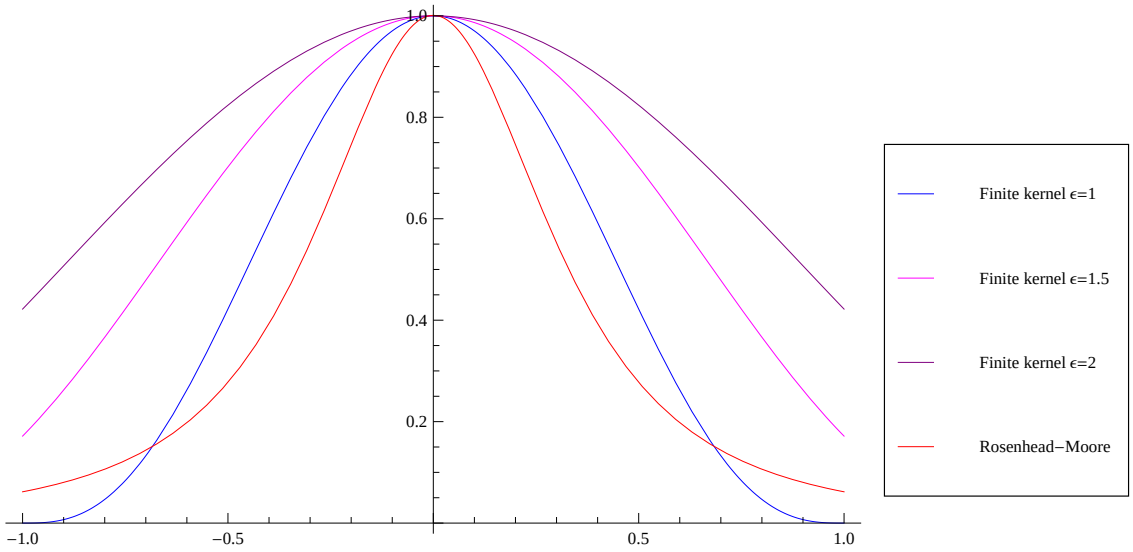


Figure 7.13: Finite kernel function curves for diferent radii compared with a Rosenhead-Moore kernel with all kernels scaled at the origin.

From this example we observe that whereas the condition in Proposition 1 guarantees a divergence-free velocity field, additional conditions are required for the velocity field to converge to known analytical solutions for different flows. This can be an issue if a specific degree of numerical accuracy is sought in calculating forces exerted on solids by fluids.

To achieve increased numerical accuracy it becomes necessary to analyze additional conditions on the kernel functions, such that they produce results with the required accuracy, in a *kernel design* process. Consider, for example the following finite-influence radius kernel:

$$w(r) = \frac{1}{4\pi(r^2 + a^2)^{3/2}} \frac{1}{2} \left[1 + cos\left(\pi \left|\frac{r}{\epsilon}\right|\right)\right]. \tag{7.6}$$

Here $a > 0$ is the smoothing constant of the Rosenhead-Moore kernel. We evaluate the pressure coefficients around the sphere using the above kernel by approximating panel velocities using Gauss-Legendre quadratures. We show results for the pressure coefficient distribution on the sphere for several radii using the above kernel in Figure 7.14.

We see that the pressure coefficients computed using the kernel in Equation (7.6) converge to the analytical solution as the radius increases. This is a desirable property

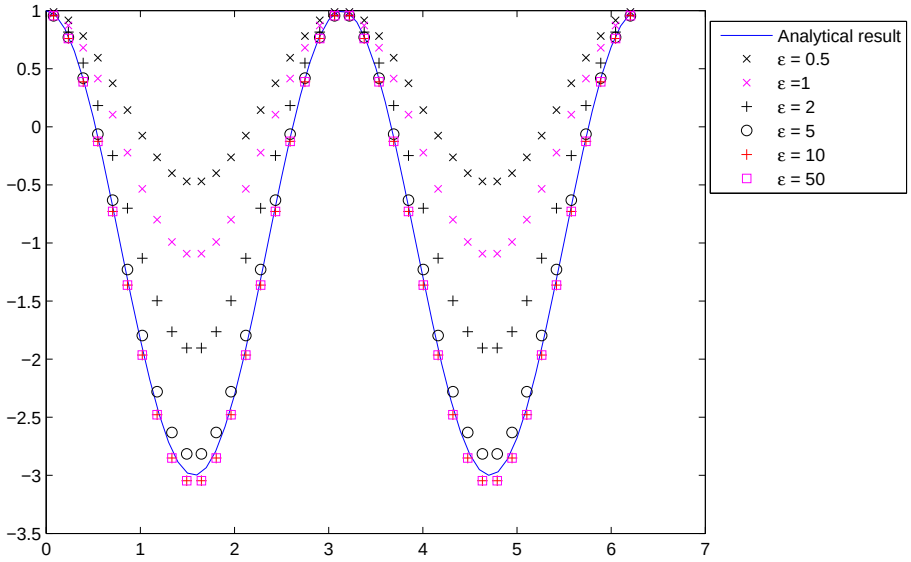


Figure 7.14: Pressure coefficients using the multiplied Rosenhead-Moore kernel in Equation (7.6). As the panel radius increases, the resulting pressure coefficients converge. A small numerical error (in average, approx. 11%) in the largest radius used is due to approximating panels using quadrature points instead of closed-form solutions for the integral.

in designing a kernel to suit specific needs. Notice that:

$$\lim_{\epsilon \to \infty} \frac{1}{2} \left[1 + cos \left(\pi \left| \frac{r}{\epsilon} \right| \right) \right] = 1. \tag{7.7}$$

With the above, it can easily verified that:

$$\lim_{\epsilon \to \infty} w(r) = \frac{1}{4\pi (r^2 + a^2)^{3/2}}. \tag{7.8}$$

Equation (7.8) states that the kernel value converges to the solution given by the Rosenhead-Moore kernel as the radius ϵ increases. We observe that the convergence results in Equation (7.6) are better than those for our previous choice of finite kernel.

Then we can formalize our first property in kernel design as follows:

$$\lim_{\epsilon \to \infty} w(r) = \frac{1}{4\pi (r^2)^{3/2}}. \tag{7.9}$$

This condition states that the kernel w must converge to that in the Biot-Savart Law. A weaker, but arguably more practical condition can be formulated as follows:

$$\lim_{\epsilon \to \infty} w(r) = K(r), \tag{7.10}$$

where $K(r)$ is a kernel function that converges to the Biot-Savart formula. An example of such kernel is the Rosenhead-Moore kernel, although other alternatives are also possible, *e.g.*, the WinckelmansLeonard kernel [152]. The choice of kernel K here defines the convergence criterion and it also becomes a design decision.

By Proposition 1 there are infinitely many functions that can be used as kernel w, and other criteria can be taken into account, *e.g.*, closed forms for integrals used in energy calculation, or the computational cost of evaluating w. We propose the study of this area as a useful and exciting avenue for future research.

7.5 Chapter Summary and Discussion

We have presented additional results from our simulation system and we have determined the main factors affecting performance. From our methods to compute the velocity field from a set of vortices, it becomes clear that the amount of vortices directly affects the amount of simulation nodes that have to be evaluated. The case of vortex shedding is a particularly challenging situation. Vortices are generated at each time step for each face in the solid surface, which generates a large amount of vortices in a limited space. This evidently increases the cost of evaluating the velocity field in those locations, despite the use of efficient search structures, such as optimized KD-trees.

We have also determined in a precise way the accuracy compromises of employing a finite kernel in the most critical process for coupling solids and fluids, which is the velocity induced by the solid surface. The velocity induced by enforcing boundary conditions is generated by our panel code, which is used for both enforcing no-penetration boundary conditions and determining the slip velocity at the solid surface. The latter is a key quantity to determine the flux of vorticity generated from the solid surface, which is the base for our force calculations.

We determine that reducing the kernel influence radii in our calculation would produce velocities that underestimate the theoretical values for pressure at the solid surface. Our initial choice of kernel also can produce an overestimation of these quantities and therefore we determine conditions over finite radii kernels to converge to analytical solutions. We

have shown a simple example of a multiplied Rosenhead-Moore kernel that produces a pressure profile on the surface that converges to the analytical solution as the radius increases in the case of a 2D sphere.

We can see from our results that our methods can produce visually rich results for fluid simulation, and also can reproduce the dynamic behavior of solid objects interacting with fluids. Our methods can be easily adapted to satisfy specific requirements of accuracy.

In the next chapter we summarize our work and our contributions and we propose directions for future research.

Chapter 8

Conclusions

In previous chapters we have presented in detail the different features of this work: Our flow model, methods for enforcing boundary conditions and our techniques to compute fluid forces acting on solid objects. We have demonstrated the quality of our results in different scenarios, showing that our results can indeed produce highly realistic and detailed fluid simulations for computer graphics applications. Our results confirm our initial statement that Lagrangian vortex methods are better suited for high-quality visual simulation of turbulent fluids than grid methods. This is because they provide arbitrary levels of detail to fluid simulations as demonstrated in our examples simulating smoke. It would be extremely challenging to produce the level of detail we have obtained in some of our examples using only grid methods for fluid simulation. Moreover, we have improved resuls over the state-of-the-art method of Pfaff *et al.* [107] where our methods produce not only highly detailed simulations, but also results that are physically consistent with real-life phenomena.

We have also demonstrated that Lagrangian particle vortex methods can be effectively applied to realistically simulate the full interplay of rigid solid objects and fluids.

In the next sections we summarize our methods, as well as our specific contributions and we discuss future research directions.

8.1 Thesis Summary

We have presented a novel fluid simulation model for inviscid, unbounded flows with two-way coupling of solids and fluids. This model consists on an inviscid flow representation using Lagrangian particles, which is coupled with a boundary layer model. This inviscid

flow is combined with methods for computing two-way coupling of rigid solid objects and fluids.

We represent a flow using Lagrangian particles that encode vorticity and a user defined background velocity field. The velocity induced by vortex particles is calculated by a modified version of the Biot-Savart formula. We define a finite influence radius for each vortex particle, hence limiting the velocity evaluation to a bounded region in space. The velocity field induced by the vortex particles and the user-defined background flow determine a free-flow velocity field.

The evolution of vorticity over time is simulated by solving the vorticity transport equation for inviscid flows. This is done by solving vorticity advection and vortex stretching. Vortex advection is solved by applying a standard Runge-Kutta integration step using the flow velocity. Solving the vortex stretching is more involved as the numerical solution of the discrete partial differential equation for this term yields an exponential growth of the vorticity magnitude, hence making simulations highly unstable. We introduce novel stable methods for solving vortex stretching based on estimating a rotation matrix for calculating the change in the vorticity spin direction. In order to avoid having vortices conserving all their internal energy, which is both physically and visually inaccurate, we introduce a simple vortex dissipation step.

Since our framework does not impose any limitation on the simulation space, our methods allow simulating unbounded flows. By controlling the vorticity on each particle, *e.g.*, using rules on vorticity generation, the user can adjust the amount of turbulence in a flow.

We have presented methods to fully represent the physical interaction between solids and fluids. This interaction is two-way coupled, *i.e.*, the fluid and solid affect each other's motion. We simulate two-way coupling of solids and fluids through the following two steps:

- First we represent the solid objects as boundaries to the flow. This is achieved through two main processes:
  - We enforce a no-penetration boundary condition on the solid surface. We perform this by modeling the solid surface as a source sheet such that cancels any flow through the surface, *i.e.*, the normal component of the flow velocity at the solid surface.
  - We enforce a slip condition at the solid surface. This condition is related to a level of adherence of the fluid to the solid surface representing its friction.

This is done by representing the solid surface as a vortex sheet such that its induced velocity corresponds to the flow's slip velocity at the solid surface. Then, vortex particles are created in a neighborhood of the solid surface such that they cancel a user-defined proportion of the slip velocity. This imposes a condition on the tangential component of the flow velocity at the solid surface. Advecting the generated vortices results in a boundary layer simulation.

- Then we calculate the fluid force exerted on the solid to simulate two-way coupling of solids and fluids. We present two methods to achieve this: one that calculates the integral force and torque on a solid system and another that calculates the force exerted by the fluid individually on each solid object. This is done by computing the total pressure force acting on the solid object due to the vorticity flux at the solid boundary, *i.e.*, the rate of creation of vorticity at the solid surface as a result of enforcing a slip condition.

We have presented our results with examples of smoke simulation. This kind of simulations often require large simulation spaces and therefore Lagrangian vortex methods represent the ideal solution to realistically simulate smoke.

We show examples of our methods with different smoke simulations interacting with solids. We demonstrate the visual quality of our results simulating flows around solid objects and objects being propelled by fluid jets. We have compared our results with state-of-the-art smoke simulations in computer graphics and we have shown that our results can represent the same phenomena with high level of visual detail, and even with improved results over previously published methods [105, 107].

We also compare results of solid-fluid interaction with a previously published approach, and we observe that our methods reproduce the expected behavior in this complex interaction.

We have observed that vortex shedding, is the most computationally expensive step in our method and hence is one of the most important factors in efficiency. A high amount of particles concentrated in a small region introduce a higher cost of velocity evaluation, even with efficient search structures. Therefore, the kernel radius plays a key role in increased performance, as increasing the radius implies evaluating the velocity field with a larger number of samples. In our experiments we show that the most important simulation parameter controlling performance is the kernel influence radius. This can be reduced to achieve higher performance, but this comes with a loss in visual quality. To

improve performance we propose the use of a kernel-independent fast multipole method, as a large kernel radius plays a key role in the visual results.

We also evaluated the accuracy of our results and found it to be strongly dependent on the kernel function used in calculating the velocity field. Increased accuracy is obtained for finite kernels when they approximate the Biot-Savart formula as the radius increases. This, however, is not an obstacle for producing visually appealing simulations of fluid flows as demonstrated in our examples.

8.2 Summary of Contributions

Our main contribution is the development of a novel Lagrangian vortex particle fluid simulation framework in computer graphics with two-way coupling with rigid solid objects of arbitrary shapes. This framework is an important step towards making vortex methods a complete simulation method that could compete with traditional, well established grid-based simulation methods.

Our specific contributions to the field of fluid simulation for computer graphics are the following:

- We present a precise characterization of the class of functions that can be used as kernels for calculating a divergence-free velocity field from a set of discrete vortex particles. Additionally we introduce conditions on finite radius kernel functions for convergence to analytical results, as a kernel design process.

- We develop novel stable methods for solving the vortex stretching term in the vorticity transport equation. These methods compute the change in spin direction for each vortex particle in our system. As opposed to previous approaches [121, 105] we do not employ an unconditionally unstable vorticity update with numerical problems that need to be alleviated by using viscosity and by clamping vector magnitudes.

  We produce visually realistic results by adding vorticity dissipation, since real-life observed flows do not preserve turbulent energy at all scales.

- We introduce a novel simulation framework in computer graphics for simulating flow around solid objects. This framework is based on the computation of a potential flow from the solid boundaries by using a panel discretization of a source sheet lying on the solid surface. This potential flow model is coupled with a vortex particle

representation of the boundary layer, similar to those previously proposed in the computer graphics literature [105]. We simplify this model to achieve realistic results in inviscid flows, since we do not handle viscosity outside the boundary layer as it is not needed. We achieve physically consistent flows with our model.

- We develop novel methods for computing fluid forces acting on solid objects, enabling two-way coupling of solids and fluids. The first method is based on calculating the force acting on a system of solid objects based on the variation of the flow's first moment of vorticity. The second method is based on calculating the pressure force acting on the surface of a submerged solid object from the rate of vorticity creation at the solid surface, or vorticity flux.

  We introduce the notion of vorticity flux to the computer graphics community as key factor involved both in visual representation of vortex shedding and in computation of forces acting on solid objects.

The contributions in this thesis are important steps in opening many research opportunities for research in fluid simulation for computer graphics using vortex methods. In the following section we outline the main directions in future research that naturally emerge from the results we have obtained.

8.3 Future Research Directions

Our approach has been developed for simulating inviscid flows such as smoke interacting with general rotating and translating rigid solid objects. Our focus is to determine the motion induced by a flow on a solid object.

As discussed in Chapter 7, further improvements can be made to accelerate our computations. We have seen that drastic reductions in vortex particle kernel radius greatly impacts the visual quality of results, and methods to solve the associated $N-$body problem are required. A good alternative, that is consistent with our approach of using finite kernels following the results in Proposition 1, would be to employ a kernel-independent fast multipole method. We describe how such an approach can be used with our methods in Appendix E.

In the following, we discuss different research directions below that can be followed in the field of fluid simulation with interacting solids using Lagrangian vortex methods based on the techniques we have presented.

Incompressible Viscous Fluids. Our model presents an inviscid flow model where viscosity is only dealt with at the solid surface. Handling a general viscous fluid requires solving a vorticity diffusion term which arises from the derivation of the vorticity transport from the Navier-Stokes Equations (see Appendix A). This can be done in 3D using a vorticity particle exchange scheme [105], or with an energy model [109] if only a very sparse set of vortices is used. However, for simulating flow around submerged objects, treating a viscous fluid with vortex methods is more involved.

Accurate calculation of surface vorticity becomes a more crucial aspect since in the case of a general viscous flow, viscous friction is not negligible, and it depends on the surface vorticity. A vortex sheet approximation is not a precise approximation of the vorticity boundary condition in general viscous flows [32]. The main difficulty of calculating a boundary condition for vorticity comes from the fact that the surface vorticity obtainable by solving the vorticity transport equations is not necessarily divergence-free at the surface [116]. This can be addressed by solving a Poisson equation on the solid surface using the flow's velocity, which would lead to an accurate solution of the surface vorticity. [116].

Compressible Fluids. Our flow model is valid only for incompressible fluids, where the velocity field is divergence-free, *i.e.*, $\nabla \cdot \mathbf{u} = 0$. The case of compressible fluids requires a different mathematical formulation, since the general definition of mass conservation in this case is given by:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0. \tag{8.1}$$

In this case the fluid density is not a constant field, and the velocity divergence is not necessarily zero. A different flow model has to be employed in this case. An example of a flow model in 2D for compressible flows with boundary conditions is presented by Eldredge *et al.* [38].

Two-way solid coupling with deformable objects. Our method to enforce no-penetration and slip boundary conditions can be trivially applied to deformable solids for one way solid-to-fluid coupling, *i.e.*, when the s solid acts as a boundary for the fluid. Achieving fluid-to-solid coupling is not trivial, though. Both methods introduced in Chapter 6 compute the net force and torque acting on the solid object's center of mass. Computing two-way solid fluid coupling with a deformable solid would require the following two main changes to our approach: First, solid objects would have to be represented by more complex structures, such as a tetrahedral mesh for applying finite

element methods [67]. Then, fluid forces would have to be calculated on each face, as opposed to the center of mass of the solid; then the solid deformation can be calculated accordingly by applying a visco-elastic force model within the solid object.

The definition of the fluid pressure gradient as a function of the vorticity flux could be used, as long as a Dirichlet pressure boundary condition is defined at the surface *i.e.*, there are constant reference pressure values defined at the surface. Although the extensions required to model two-way solid fluid coupling with deformable solids is technically feasible, simulating two-way coupled vortex flows with deformable solids is a challenging task.

Filaments and thin shells. In our approach we assume that each solid object is represented as a polygonal surface and moreover, that the object volume is non-zero. These assumptions represent major limitations to the application of our methods to the scenarios of thin shells (*e.g.*, clothes) and filaments (*e.g.*, hair).

Source sheet methods to enforce no-penetration boundary conditions introduce divergence artifacts if applied on thin shells modeled as an open polygonal surface. Consider the case of an extended rectangular cloth and a uniform flow directed towards the plane defined by the cloth. A source panel representation of the thin shell would cancel the incoming flow on one side of the sheet but induces velocity that would add up to the external flow on the other side of the sheet. This problem is not present in volumetric objects since the source sheet acts as a source and as a sink on different sides of surface.

In the case of thin filaments, a naïve approach for one-way fluid to solid coupling would be to advect filament's control points following the external velocity field. However, achieving solid to fluid coupling is not trivial, as our vortex shedding method relies on the calculation of a vortex sheet on the solid surface. Accurately determining the vorticity flux cannot be directly performed with the methods we have presented.

Liquids. As discussed in Section 3.2.1, a critical aspect of simulating liquids is the representation of the free surface. This is often done by constructing and tracking a level set function. In the context of vortex methods, several issues have to be addressed.

First, tracking the fluid position and the free surface needs to be done in an *ad-hoc* manner. Current surface tracking methods rely on the velocities obtained from a grid, where pressure is considered explicitly, or from a dense set of particles, which define the fluid locations. In contrast, the density of vortex particles used in our approach would not necessarily suffice to define the entire fluid volume. External forces (*e.g.*, gravity) would have to be incorporated explicitly to keep a liquid appearance.

Second, the interaction between vortices and the free surface needs to be modeled.

The free surface concentrates vorticity and also becomes a source of vorticity [158].

Third, liquid simulation in computer graphics often requires modeling bodies of fluid that split and merge, such as droplets and spray. Here the dynamics of vorticity on splitting and merging volumes of liquid have to be represented for a realistic visual simulation.

All the above are interesting research directions that would benefit from the methods we have developed in this thesis. We foresee these topics being addressed in the near future due to the great potential of Lagrangian vortex methods for simulation of fluids in computer graphics.

Appendix A

Derivation of the Vorticity Transport Equation

Here we provide a standard derivation of the Vorticity Transport Equation from the Euler Equations, which can be formulated as:

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

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

The flow vorticity is defined as the curl of the velocity field, *i.e.*, :

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}. \tag{A.3}$$

An equation for evolution of vorticity is obtained by applying curl to Equation (A.1). The result of applying curl to each term is discussed below in detail:

A.1 Advection

The advection term from Equation (2.2) is split into two terms: vorticity advection $(\mathbf{u}\cdot\nabla)\boldsymbol{\omega}$ and vortex stretching $(\boldsymbol{\omega}\cdot\nabla)\mathbf{u}$. These are obtained from the following formulation of the advection term:

$$(\mathbf{u} \cdot \nabla)\mathbf{u} = (\nabla \times \mathbf{u}) \times \mathbf{u} + \nabla \frac{\|\mathbf{u}\|^2}{2}. \tag{A.4}$$

Applying curl to the above equation and by applying Equation (A.3) we obtain:

$$\nabla \times (\mathbf{u} \cdot \nabla)\mathbf{u} = \nabla \times (\boldsymbol{\omega} \times \mathbf{u}) + \nabla \times \left(\nabla \frac{\|\mathbf{u}\|^2}{2} \right). \tag{A.5}$$

The last term vanishes as the curl of the gradient of a scalar field is always zero. The first term in the right hand side of Equation (A.5) is expanded by evaluating the curl of a cross product as follows:

$$\begin{aligned} \nabla \times (\boldsymbol{\omega} \times \mathbf{u}) &= \mathbf{u} \cdot \nabla \boldsymbol{\omega} - \boldsymbol{\omega} \cdot \nabla \mathbf{u} + \boldsymbol{\omega}(\nabla \cdot \mathbf{u}) - \mathbf{u}(\nabla \cdot \boldsymbol{\omega}) \\ &= \mathbf{u} \cdot \nabla \boldsymbol{\omega} - \boldsymbol{\omega} \cdot \nabla \mathbf{u}. \end{aligned} \tag{A.6}$$

Here we have used the fact that both the velocity and vorticity fields are divergence free. Notice vorticity is divergence free as for any vector field $\mathbf{V}$ in $\mathbb{R}^3$ it holds $\nabla \cdot (\nabla \times \mathbf{V}) = 0$.

A.2 Diffusion

The vorticity diffusion term is derived from the following vector calculus identity:

$$\begin{aligned} \nabla \times (\nabla \times \mathbf{u}) &= \nabla(\nabla \cdot \mathbf{u}) - \nabla^2 \mathbf{u} \\ &= -\nabla^2 \mathbf{u}. \end{aligned} \tag{A.7}$$

Here we have used again the zero divergence property of the velocity field. By taking the curl at both sides of the above equation we have:

$$\nabla \times (\nabla \times \mathbf{u}) = \nabla \times (-\nabla^2 \mathbf{u}). \tag{A.8}$$

We expand the left hand side of the above equation obtaining:

$$\begin{aligned} \nabla \times (\nabla \times (\nabla \times \mathbf{u})) &= \nabla \times (\nabla \times \boldsymbol{\omega}) \\ &= \nabla(\nabla \cdot \boldsymbol{\omega}) - \nabla^2 \boldsymbol{\omega} \\ &= -\nabla^2 \boldsymbol{\omega}. \end{aligned} \tag{A.9}$$

Then it follows that $\nabla \times (\nabla^2 \mathbf{u}) = \nabla^2 \boldsymbol{\omega}$.

A.3 Pressure

The pressure term in Equation (2.2) vanishes for a constant density flow, since:

$$\begin{aligned} \nabla \times \left(\frac{\nabla p}{\rho} \right) &= \frac{1}{\rho} \nabla \times \nabla p + \nabla(\frac{1}{\rho}) \times \nabla p \\ &= \frac{\nabla \rho}{\rho^2} \times \nabla p. \end{aligned} \tag{A.10}$$

Here we have used that the curl of a gradient field is zero. The resulting term corresponds to the *baroclinic* term of the vorticity transport equation. Notice that for constant density flows, $\nabla \rho = 0$ and this term vanishes.

A.4 Vorticity Transport Equation

Applying curl to both sides of Equation (2.2) we obtain:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = -(\mathbf{u} \cdot \nabla)\boldsymbol{\omega} + (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \frac{\nabla\rho \times \nabla p}{\rho^2} + \nabla \times \mathbf{f}. \tag{A.11}$$

In the case external forces acting on the fluid are irrotational (such as gravity), the last term in Equation (A.11) is zero. Then we have the following transport equation:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = -(\mathbf{u} \cdot \nabla)\boldsymbol{\omega} + (\boldsymbol{\omega} \cdot \nabla)\mathbf{u}. \tag{A.12}$$

Appendix B

Proof of Proposition 1

Below we provide our proof of Proposition 1, which is repeated here for the reader's convenience:

Proposition 2 *For any continuous and differentiable radial basis function* $w : \mathbb{R} \to \mathbb{R}$ *and vectors* $\boldsymbol{\omega}$, $\mathbf{x}$ *and* $\mathbf{z} \in \mathbb{R}^3$, *the following holds:* $\nabla \cdot [(\boldsymbol{\omega} \times (\mathbf{x} - \mathbf{z}))\, w(\|\mathbf{x} - \mathbf{z}\|)] = 0$.

Proof:

Let $\mathbf{r} = \mathbf{x} - \mathbf{z}$ and $r = \|\mathbf{r}\|$. Then we have:

$$(\boldsymbol{\omega} \times (\mathbf{x} - \mathbf{z}))\, w(\|\mathbf{x} - \mathbf{z}\|) = (\boldsymbol{\omega} \times \mathbf{r}) w(r). \tag{B.1}$$

By applying the divergence operator, we obtain the following:

$$\nabla \cdot ((\boldsymbol{\omega} \times \mathbf{r}) w(r)) = (\nabla \cdot (\boldsymbol{\omega} \times \mathbf{r})) w(r) + (\boldsymbol{\omega} \times \mathbf{r}) \cdot \nabla w(r). \tag{B.2}$$

We verify by direct evaluation that $\nabla \cdot (\boldsymbol{\omega} \times (\mathbf{x} - \mathbf{z})) = 0$, hence the first term of the right hand side of Equation (B.2) is zero.

We focus on the second term of the right hand side of Equation (B.2). By the chain rule we have that:

$$\nabla w(r) = \frac{dw}{dr} \nabla r. \tag{B.3}$$

Therefore, we have the following expression for the divergence:

$$\nabla \cdot ((\boldsymbol{\omega} \times \mathbf{r}) w(r)) = (\boldsymbol{\omega} \times \mathbf{r}) \cdot \frac{dw}{dr} \nabla r. \tag{B.4}$$

Notice that $\nabla r = \nabla \|\mathbf{r}\| = \frac{\mathbf{r}}{\|\mathbf{r}\|}$. Replacing this value in Equation (B.4) and rearranging the terms we have:

$$\begin{aligned} \nabla \cdot ((\boldsymbol{\omega} \times \mathbf{r}) w(r)) &= \frac{1}{\|\mathbf{r}\|} ((\boldsymbol{\omega} \times \mathbf{r}) \cdot \mathbf{r}) \frac{dw}{dr} \\ &= 0. \end{aligned} \tag{B.5}$$

Therefore, for $\mathbf{u}(\mathbf{x}) = (\boldsymbol{\omega} \times (\mathbf{x} - \mathbf{z}))\, w(\|\mathbf{x} - \mathbf{z}\|)$ we have:

$$\nabla \cdot \mathbf{u}(\mathbf{x}) = 0.$$

■

Appendix C

Derivation of the Discrete Vortex Panels Boundary Equation

Below we show our derivation of the discrete boundary equations to calculate vortex panels using our velocity definition.

The surface vorticity in panel j is represented by $\boldsymbol{\gamma}_j = \gamma_j^s \hat{\mathbf{s}}_j + \gamma_j^t \hat{\mathbf{t}}_j$ and the boundary integral equation for the $\hat{\mathbf{s}}$-component for the slip velocity at the control point $\mathbf{x}_i$ corresponds to:

$$\Delta \mathbf{u}(\mathbf{x}_i) \cdot \hat{\mathbf{s}}_i = \frac{\boldsymbol{\gamma}_i \times \hat{\mathbf{n}}_i}{2} \cdot \hat{\mathbf{s}}_i + \sum_j \int_j \frac{\boldsymbol{\gamma}_j}{4\pi} \times \frac{\mathbf{x}_i - \mathbf{z}}{\|\mathbf{x}_i - \mathbf{z}\|^3} d\mathbf{z} \cdot \hat{\mathbf{s}}(\mathbf{x}_i). \tag{C.1}$$

Then, by replacing the definition of $\boldsymbol{\gamma}$ in the first term in the right-hand side of Equation (C.1) we obtain:

$$\begin{aligned} \frac{\boldsymbol{\gamma}_i \times \hat{\mathbf{n}}_i}{2} \cdot \hat{\mathbf{s}}_i &= \frac{(\gamma_i^s \hat{\mathbf{s}}_i + \gamma_i^t \hat{\mathbf{t}}_i) \times \hat{\mathbf{n}}_i}{2} \cdot \hat{\mathbf{s}}_i \\ &= \frac{-\gamma_i^s \hat{\mathbf{t}}_i + \gamma_i^t \hat{\mathbf{s}}_i}{2} \cdot \hat{\mathbf{s}}_i \\ &= \frac{\gamma_i^t}{2}. \end{aligned} \tag{C.2}$$

Notice that $\hat{\mathbf{s}} \times \hat{\mathbf{t}} = \hat{\mathbf{n}}$. By assuming $\boldsymbol{\gamma}$ is constant in panel j and replacing the definition of $\boldsymbol{\gamma}$ in the integral term of Equation (C.1) we obtain:

$$\int_j \frac{\boldsymbol{\gamma}_j}{4\pi} \times \frac{\mathbf{x}_i - \mathbf{z}}{\|\mathbf{x}_i - \mathbf{z}\|^3} d\mathbf{z} \cdot \hat{\mathbf{s}}_i = \left((\gamma_i^s \hat{\mathbf{s}}_i + \gamma_i^t \hat{\mathbf{t}}_i) \times \int_j \frac{1}{4\pi} \frac{\mathbf{x}_i - \mathbf{z}}{\|\mathbf{x}_i - \mathbf{z}\|^3} d\mathbf{z} \right) \cdot \hat{\mathbf{s}}_i. \tag{C.3}$$

Notice the integral in the right-hand side of Equation (C.3) corresponds to the velocity induced by a unit strength source panel. By replacing this expression with the velocity

field u^* in Equation (5.6) we rewrite Equation (C.3) as follows:

$$\int_j \frac{\boldsymbol{\gamma}_j}{4\pi} \times \frac{\mathbf{x}_i - \mathbf{z}}{\|\mathbf{x}_i - \mathbf{z}\|^3} d\mathbf{z} \cdot \hat{\mathbf{s}}_i = ((\gamma_j^s \hat{\mathbf{s}}_j + \gamma_j^t \hat{\mathbf{t}}_j) \times \mathbf{u}_j^*(\mathbf{x}_i)) \cdot \hat{\mathbf{s}}_i \tag{C.4}$$

Then by developing the triple product on the right hand side of the above equation, the boundary equation Equation (C.1) is rewritten as follows:

$$\Delta \mathbf{u}(\mathbf{x}_i) \cdot \hat{\mathbf{s}}_i = \frac{\gamma_i^t}{2} + \sum_j (\hat{\mathbf{s}}_i \times (\gamma_j^s \hat{\mathbf{s}}_j + \gamma_j^t \hat{\mathbf{t}}_j)) \cdot \mathbf{u}_j^*(\mathbf{x}_i). \tag{C.5}$$

The above equation can be rewritten so the coefficients of the vortex panel matrix are identifiable as follows:

$$\Delta \mathbf{u}(\mathbf{x}_i) \cdot \hat{\mathbf{s}}_i = \frac{\gamma_i^t}{2} + \sum_j \gamma_j^s (\hat{\mathbf{s}}_i \times \hat{\mathbf{s}}_j) \cdot \mathbf{u}_j^*(\mathbf{x}_i) + \sum_j \gamma_j^t (\hat{\mathbf{s}}_i \times \hat{\mathbf{t}}_j) \cdot \mathbf{u}_j^*(\mathbf{x}_i). \tag{C.6}$$

Similarly, an expression for the boundary integral for the $\hat{\mathbf{t}}$-component of vorticity is the following:

$$\Delta \mathbf{u}(\mathbf{x}_i) \cdot \hat{\mathbf{t}}_i = -\frac{\gamma_i^s}{2} + \sum_j \gamma_j^s (\hat{\mathbf{t}}_i \times \hat{\mathbf{s}}_j) \cdot \mathbf{u}_j^*(\mathbf{x}_i) + \sum_j \gamma_j^t (\hat{\mathbf{t}}_i \times \hat{\mathbf{t}}_j) \cdot \mathbf{u}_j^*(\mathbf{x}_i). \tag{C.7}$$

Appendix D

Derivation of the Vorticity Flux Equation

We present the proof below for the reader's convenience. The proof here is a shorter version of the original by Wu and Wu [156].

The Cauchy equations for motion describe the momentum transport in any continuum and are formulated as follows:

$$\mathbf{a} \equiv \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} \right) = \frac{1}{\rho} \nabla \cdot \mathbb{T} + \mathbf{f}. \tag{D.1}$$

Here $\mathbf{a}$ corresponds to the acceleration, $\mathbb{T}$ to the stress tensor and $\mathbf{f}$ to the acceleration due to body forces. The divergence of $\mathbb{T}$ represents the surface force vector. For constant viscosity flows, this can be expressed using the Stokes-Helmholtz decomposition as follows:

$$\nabla \cdot \mathbb{T} = -\nabla\varphi - \nabla \times \mathbf{A} \quad \text{where} \quad \nabla \cdot \mathbf{A} = 0. \tag{D.2}$$

The Stokes-Helmholtz decomposition represents a vector as two processes: a compression/expanding longitudinal process φ and a transversal shearing process $\mathbf{A}$. Then by applying Equations D.2 and D.1, we obtain the following:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \nabla \times \mathbf{f} - \nabla \times \frac{1}{\rho}(\nabla\varphi + \nabla \times \mathbf{A}). \tag{D.3}$$

By algebraically manipulating the above equation we obtain:

$$\begin{aligned} \frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \nabla \times \mathbf{f} + \frac{\nabla\rho}{\rho^2} \times \nabla\varphi \\ - \frac{1}{\rho}\nabla \times (\nabla\varphi) - \nabla \times \frac{1}{\rho}(\nabla \times \mathbf{A}). \end{aligned} \tag{D.4}$$

Since the curl of the gradient of a scalar field φ is always zero, the above equation reduces to:

$$\begin{aligned}\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u}\cdot\nabla)\boldsymbol{\omega} = (\boldsymbol{\omega}\cdot\nabla)\mathbf{u} + \nabla\times\mathbf{f} + \frac{\nabla\rho}{\rho^2}\times\varphi \\ - \nabla\times\frac{1}{\rho}(\nabla\times\mathbf{A}).\end{aligned} \tag{D.5}$$

By integrating the above equation over the whole domain $\mathcal{D}$, the total change in vorticity is expressed as:

$$\begin{aligned}\frac{D}{Dt}\int_{\mathcal{D}}\boldsymbol{\omega}d\mathbf{x} = \int_{\mathcal{D}}\left((\boldsymbol{\omega}\cdot\nabla)\mathbf{u} + \frac{\nabla\rho}{\rho^2}\times\nabla\varphi\right)d\mathbf{x} \\ + \int_{\mathcal{D}}\nabla\times\mathbf{f}d\mathbf{x} - \int_{\mathcal{D}}\nabla\times\frac{1}{\rho}(\nabla\times\mathbf{A})d\mathbf{x}.\end{aligned} \tag{D.6}$$

Here D/Dt is the total derivative defined as:

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + (\mathbf{u}\cdot\nabla). \tag{D.7}$$

The second and third terms in the right hand side of Equation (D.6) are transformed to surface integrals on the domain boundary $\partial\mathcal{D}$ yielding:

$$\begin{aligned}\frac{D}{Dt}\int_{\mathcal{D}}\boldsymbol{\omega}d\mathbf{x} = \int_{\mathcal{D}}\left((\boldsymbol{\omega}\cdot\nabla)\mathbf{u} + \frac{\nabla\rho}{\rho^2}\times\nabla\varphi\right)d\mathbf{x} \\ + \int_{\partial\mathcal{D}}\hat{\mathbf{n}}\times\mathbf{f}d\mathbf{x} - \int_{\partial\mathcal{D}}\hat{\mathbf{n}}\times\frac{1}{\rho}(\nabla\times\mathbf{A})d\mathbf{x}.\end{aligned} \tag{D.8}$$

The first term on the right hand side of the above equation corresponds to the vorticity change due to stretching and baroclinicity. The second term corresponds to external forces. The generation of vorticity at the surface then must be represented in the third term [156]. To derive the expression of the vorticity flux, the following vector identities are used:

$$(\mathbf{b}\times\nabla)\times\mathbf{c} = (\nabla\mathbf{c})\cdot\mathbf{b} - \mathbf{b}(\nabla\cdot\mathbf{c}), \tag{D.9}$$

$$\mathbf{b}\times(\nabla\times\mathbf{c}) = (\nabla\mathbf{c})\cdot\mathbf{b} - \mathbf{b}\cdot(\nabla\mathbf{c}). \tag{D.10}$$

By combining the above equations we obtain:

$$(\mathbf{b}\times\nabla)\times\mathbf{c} - \mathbf{b}\times(\nabla\times\mathbf{c}) = \mathbf{b}\cdot(\nabla\mathbf{c}) - \mathbf{b}(\nabla\cdot\mathbf{c}). \tag{D.11}$$

By setting $\mathbf{b} = \hat{\mathbf{n}}$ and $\mathbf{c} = \mathbf{A}$ we get:

$$(\hat{\mathbf{n}}\times\nabla)\times\mathbf{A} - \hat{\mathbf{n}}\times(\nabla\times\mathbf{A}) = \hat{\mathbf{n}}\cdot(\nabla\mathbf{A}) - \hat{\mathbf{n}}(\nabla\cdot\mathbf{A}). \tag{D.12}$$

By applying the condition $\nabla \cdot \mathbf{A} = 0$ from Equation (D.2), and rearranging the terms in Equation (D.12) we obtain:

$$-\hat{\mathbf{n}} \times (\nabla \times \mathbf{A}) = \hat{\mathbf{n}} \cdot (\nabla \mathbf{A}) - (\hat{\mathbf{n}} \times \nabla) \times \mathbf{A}. \tag{D.13}$$

The Cauchy equations of motion can be conveniently rewritten as follows:

$$\rho(\mathbf{a} - \mathbf{f}) + \nabla\varphi = -\nabla \times \mathbf{A}. \tag{D.14}$$

Equation (D.13) is rewritten using the definition of $\nabla \times \mathbf{A}$ in the above equation yielding:

$$\hat{\mathbf{n}} \times (\rho(\mathbf{a} - \mathbf{f}) + \nabla\varphi) = \hat{\mathbf{n}} \cdot (\nabla \mathbf{A}) - (\hat{\mathbf{n}} \times \nabla) \times \mathbf{A}. \tag{D.15}$$

Noting that $\hat{\mathbf{n}} \cdot (\nabla \mathbf{A}) = \frac{\partial \mathbf{A}}{\partial \hat{\mathbf{n}}}$, we rearrange the terms in the above equation to obtain:

$$\frac{\partial \mathbf{A}}{\partial \hat{\mathbf{n}}} = \hat{\mathbf{n}} \times (\rho(\mathbf{a} - \mathbf{f}) + \nabla\varphi) + (\hat{\mathbf{n}} \times \nabla) \times \mathbf{A}. \tag{D.16}$$

We observe that for a Newtonian fluid with constant viscosity, the Stokes-Helmholtz decomposition of the stress tensor is:

$$\nabla \cdot \mathbb{T} = -\nabla p - \mu \nabla \times \boldsymbol{\omega} \quad \text{where} \quad \nabla \cdot \boldsymbol{\omega} = 0. \tag{D.17}$$

Then it follows that the flux definition from Equation (D.16) corresponds to:

$$\nu \frac{\partial \boldsymbol{\omega}}{\partial \hat{\mathbf{n}}} = \hat{\mathbf{n}} \times (\rho(\mathbf{a} - \mathbf{f})) + \hat{\mathbf{n}} \times \frac{1}{\rho} \nabla p + \nu(\hat{\mathbf{n}} \times \nabla) \times \boldsymbol{\omega}. \tag{D.18}$$

The above corresponds to the definition of vorticity flux from a solid boundary to the fluid due to viscosity. In the inviscid limit $\nu \to 0$, we rewrite the vorticity flux as follows:

$$\nu \frac{\partial \boldsymbol{\omega}}{\partial \hat{\mathbf{n}}} = \hat{\mathbf{n}} \times (\rho(\mathbf{a} - \mathbf{f})) + \hat{\mathbf{n}} \times \frac{1}{\rho} \nabla p. \tag{D.19}$$

Appendix E

Kernel Independent Fast Multipole Method

Fong and Darve [44] present a *Fast Multipole Method* (FMM) for evaluating continuous, non oscillatory kernels based solely on kernel evaluation. Unlike traditional FFMs, this method does not require an analytical expansion of the kernel which makes the method suitable to be applied in our scenario. Here we show how to apply their technique, originally designed to calculate potentials, to evaluate vortex velocity in our method.

The aim of a FMM is to efficiently compute sums of the form:

$$f(x_i) = \sum_{j=1}^{N} K(x_i, z_j)\sigma_j \tag{E.1}$$

This is achieved by finding an approximation of the kernel K. Fong and Darve [44] provide such an approximation for the kernel K based on an interpolation scheme of the following form:

$$K(x, z) \approx \sum_{l=1}^{n}\sum_{m=1}^{n} K(\bar{x}_l, \bar{z}_m)S_n(\bar{x}_l, x_i)S_n(\bar{z}_m, z_j). \tag{E.2}$$

Here S_n is an interpolation function constructed using Chebyshev polynomials and their roots $\{\bar{x}_l\}$ and $\{\bar{z}_m\}$ which are used as interpolation nodes. Here S_n has the form:

$$S_n(x, z) = \frac{1}{n} + \frac{2}{n}\sum_{k=1}^{n-1} T_k(x)T_k(z), \tag{E.3}$$

where $T_k(x)$ is the first-kind Chebyshev polynomial of order k, in the interval $[-1, 1]$. The values $\{\bar{x}_m\}$ in Equation (E.2) are the roots of $T_n(x)$ for $m = 1, \ldots, n$.

Equation (E.2) is then applied to construct the following approximation of f in Equation (E.1):

$$f(x_i) \approx \sum_{l=1}^{n} S_n(\bar{x}_l, x_i) \sum_{m=1}^{n} K(\bar{x}_l, \bar{z}_m) \sum_{j=1}^{N} \sigma_j S_n(\bar{z}_m, z_j). \tag{E.4}$$

For details on the theoretical aspects of this approximation we refer the reader to the original work by Fong and Darve [44]. This approximation is valid in the square domain $[-1,1] \times [-1,1]$ but a linear transformation can be used to map into this unit interval as required.

Below we show how to apply Equation (E.2) to approximate the three-dimensional vortex velocity, defined as follows:

$$\mathbf{u}(\mathbf{x}_i) = \sum_{j=1}^{N} \boldsymbol{\omega}_j \times (\mathbf{x}_i - \mathbf{z}_j) w(\mathbf{x}_i, \mathbf{z}_j), \tag{E.5}$$

where w is our kernel function. Notice that a three-dimensional kernel w can be approximated using Chebyshev polynomials as follows [44]:

$$w(\mathbf{x}, \mathbf{z}) \approx \sum_{\mathbf{l}} \sum_{\mathbf{m}} w(\bar{\mathbf{x}}_\mathbf{l}, \bar{\mathbf{z}}_\mathbf{m}) R_n(\bar{\mathbf{x}}_\mathbf{l}, \mathbf{x}) R_n(\bar{\mathbf{z}}_\mathbf{m}, \mathbf{z}). \tag{E.6}$$

Here $R_n(\mathbf{x}, \mathbf{z})$ is the following product of the interpolation functions:

$$R_n(\mathbf{x}, \mathbf{z}) = S_n(x_1, z_1) S_n(x_2, z_2) S_n(x_3, z_3), \tag{E.7}$$

where $\bar{\mathbf{x}}_\mathbf{l} = [x_{l_1}, x_{l_2}, x_{l_3}]$ and $\bar{\mathbf{z}}_\mathbf{m} = [z_{m_1}, z_{m_2}, z_{m_3}]$ are vectors of interpolation nodes for each dimension [44], *i.e.* $m_d, l_d = 1, \ldots, n$.

With the above, we can approximate the solution of Equation (E.5) by replacing the kernel value w by the expression in Equation (E.6) as follows:

$$\begin{aligned} \mathbf{u}(\mathbf{x}_i) \approx \sum_{\mathbf{l}} \sum_{\mathbf{m}} \sum_{j=1}^{N} [& w(\bar{\mathbf{x}}_\mathbf{l}, \bar{\mathbf{z}}_\mathbf{m}) R_n(\bar{\mathbf{x}}_\mathbf{l}, \mathbf{x}_i) R_n(\bar{\mathbf{z}}_\mathbf{m}, \mathbf{z}_j) \\ & \boldsymbol{\omega}_j \times (\mathbf{x}_i - \mathbf{z}_j)]. \end{aligned} \tag{E.8}$$

We need to rewrite the above equation into a form similar to Equation (E.4). Notice that the above can be rewritten as follows:

$$\begin{aligned} & \mathbf{u}(\mathbf{x}_i) \approx \mathbf{u}_1 - \mathbf{u}_2, \quad \text{where:} \\ & \mathbf{u}_1 = \sum_{\mathbf{l}} \sum_{\mathbf{m}} \sum_{j=1}^{N} w(\bar{\mathbf{x}}_\mathbf{l}, \bar{\mathbf{z}}_\mathbf{m}) R_n(\bar{\mathbf{x}}_\mathbf{l}, \mathbf{x}_i) R_n(\bar{\mathbf{z}}_\mathbf{m}, \mathbf{z}_j) \boldsymbol{\omega}_j \times \mathbf{x}_i, \\ & \mathbf{u}_2 = \sum_{\mathbf{l}} \sum_{\mathbf{m}} \sum_{j=1}^{N} w(\bar{\mathbf{x}}_\mathbf{l}, \bar{\mathbf{z}}_\mathbf{m}) R_n(\bar{\mathbf{x}}_\mathbf{l}, \mathbf{x}_i) R_n(\bar{\mathbf{z}}_\mathbf{m}, \mathbf{z}_j) \boldsymbol{\omega}_j \times \mathbf{z}_j. \end{aligned} \tag{E.9}$$

After algebraic manipulations, $\mathbf{u}_1$ can be rewritten as follows:

$$\begin{aligned}\mathbf{u}_1 = &- \sum_{\mathbf{l}} \mathbf{x}_i R_n(\bar{\mathbf{x}}_\mathbf{l}, \mathbf{x}_i) \times \\ &\left(\sum_{\mathbf{m}} w(\bar{\mathbf{x}}_\mathbf{l}, \bar{\mathbf{z}}_\mathbf{m}) \sum_{j=1}^{N} R_n(\bar{\mathbf{z}}_\mathbf{m}, \mathbf{z}_j) \boldsymbol{\omega}_j \right).\end{aligned} \tag{E.10}$$

Notice also that $\mathbf{u}_2$ can be rewritten as follows:

$$\mathbf{u}_2 = \sum_{\mathbf{l}} R_n(\bar{\mathbf{x}}_\mathbf{l}, \mathbf{x}_i) \sum_{\mathbf{m}} w(\bar{\mathbf{x}}_\mathbf{l}, \bar{\mathbf{z}}_\mathbf{m}) \sum_{j=1}^{N} R_n(\bar{\mathbf{z}}_\mathbf{m}, \mathbf{z}_j) \boldsymbol{\omega}_j \times \mathbf{z}_j. \tag{E.11}$$

Then, following the same idea as Fong and Darve [44], we can proceed as follows to calculate $\mathbf{u}(\mathbf{x}_i)$:

1. Calculate the interpolation nodes $\bar{\mathbf{z}}_\mathbf{m}$ and then compute $V_\mathbf{m} = \sum_{j=1}^{N} R_n(\bar{\mathbf{z}}_\mathbf{m}, \mathbf{z}_j)\boldsymbol{\omega}_j$, for $m_i = 1, \ldots, n$. This requires $n^3 N$ steps.

2. Calculate the interpolation nodes $\bar{\mathbf{x}}_\mathbf{l}$ and then compute $W_\mathbf{l} = \sum_\mathbf{m} V_\mathbf{m} w(\bar{\mathbf{x}}_\mathbf{l}, \bar{\mathbf{z}}_\mathbf{m})$, for $l_i = 1, \ldots, n$. This requires $(n^3)^2 = n^6$ steps.

3. Calculate: $\mathbf{u}_1 = -\sum_\mathbf{l} \mathbf{x}_i R_n(\bar{\mathbf{x}}_\mathbf{l}, \mathbf{x}_i) \times W_\mathbf{l}$, for $i = 1, \ldots, N$. This requires $n^3 N$ steps.

4. Calculate: $V^*_\mathbf{m} = \sum_{j=1}^{N} R_n(\bar{\mathbf{z}}_\mathbf{m}, \mathbf{z}_j)(\boldsymbol{\omega}_j \times \mathbf{z}_j)$, for $m_i = 1, \ldots, n$. This requires $n^3 N$ steps.

5. Calculate: $W^*_\mathbf{l} = \sum_\mathbf{m} V^*_\mathbf{m} w(\bar{\mathbf{x}}_\mathbf{l}, \bar{\mathbf{z}}_\mathbf{m})$, for $l_i = 1, \ldots, n$. This requires $(n^3)^2 = n^6$ steps.

6. Calculate: $\mathbf{u}_2 = \sum_\mathbf{l} \mathbf{x}_i R_n(\bar{\mathbf{x}}_\mathbf{l}, \mathbf{x}_i) \times W^*_\mathbf{l}$, for $i = 1, \ldots, N$. This requires $n^3 N$ steps.

7. Finally calculate $\mathbf{u}(\mathbf{x}_i) = \mathbf{u}_1 - \mathbf{u}_2$.

The total amopunt of steps is then $2n^6 + 4n^3 N$. If $n << N$, the above summation would require $O(N)$ steps to evaluate the velocity $\mathbf{u}(\mathbf{x}_i)$ for $i = 1, \ldots, N$.

Bibliography

[1] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas. Adaptively sampled particle fluids. *ACM Trans. Graph.*, 26(3):48:1–48:8, July 2007.

[2] N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, and M. Teschner. Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans. on Graph.*, 31(4):62:1–62:8, July 2012.

[3] G. Amador and A. Gomes. A CUDA-based implementation of stable fluids in 3D with internal and moving boundaries. In *Proceedings of the 2010 International Conference on Computational Science and Its Applications*, ICCSA '10, pages 118–128, Washington, DC, USA, 2010. IEEE Computer Society.

[4] A. Angelidis and F. Neyret. Simulation of smoke based on vortex filament primitives. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pages 87–96, New York, NY, USA, 2005. ACM.

[5] A. Angelidis, F. Neyret, K. Singh, and D. Nowrouzezahrai. A controllable, fast and stable basis for vortex based smoke simulation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '06, pages 25–32, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[6] A. Appel. On calculating the illusion of reality. In *IFIP Congress (2)*, pages 945–950, 1968.

[7] K. Bao, X. Wu, H. Zhang, and E. Wu. Volume fraction based miscible and immiscible fluid animation. *Comput. Animat. Virtual Worlds*, 21(34):401–410, May 2010.

[8] A. W. Bargteil, T. G. Goktekin, J. F. O'brien, and J. A. Strain. A semi-lagrangian contouring method for fluid simulation. *ACM Trans. Graph.*, 25(1):19–38, Jan. 2006.

[9] C. Batty, F. Bertails, and R. Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph.*, 26(3):100:1–100:8, July 2007.

[10] C. Batty and R. Bridson. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, pages 219–228, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[11] C. Batty, S. Xenos, and B. Houston. Tetrahedral embedded boundary methods for accurate and flexible adaptive fluids. *Computer Graphics Forum*, 29(2):695–704, 2010.

[12] P. Beaudoin, S. Paquet, and P. Poulin. Realistic and controllable fire simulation. In *Graphics Interface 2001*, GRIN'01, pages 159–166, Toronto, Ont., Canada, Canada, 2001. Canadian Information Processing Society.

[13] M. Becker and M. Teschner. Weakly compressible SPH for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 209–217, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[14] F. Bridault, M. Leblond, F. Rousselle, and C. Renaud. Real-time rendering and animation of plentiful flames. In D. S. Ebert and S. Mérillou, editors, *Proceedings of the Eurographics Workshop on Natural Phenomena, NPH 2007, Prague, Czech Republic, 2007*, pages 31–38. Eurographics Association, 2007.

[15] R. Bridson. *Fluid Simulation for Computer Graphics*. A. K. Peters, Natick, MA, USA, 2008.

[16] R. Bridson, J. Houriham, and M. Nordenstam. Curl-noise for procedural fluid flow. *ACM Trans. Graph.*, 26(3), July 2007.

[17] T. Brochu, C. Batty, and R. Bridson. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph.*, 29(4):47:1–47:9, July 2010.

[18] D. Calhoun. A cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions. *Journal of Computational Physics*, 176(2):231 – 275, 2002.

[19] M. Carlson, P. J. Mucha, and G. Turk. Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph.*, 23(3):377–384, Aug. 2004.

[20] M. Carlson, P. J. Mucha, R. B. Van Horn, III, and G. Turk. Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, pages 167–174, New York, NY, USA, 2002. ACM.

[21] C. Casciola, R. Piva, and P. Bassanini. Vorticity generation on a flat surface in 3D flows. *Journal of Computational Physics*, 129(2):345 – 356, 1996.

[22] S. Chen and G. D. Doolen. Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30:329–364, 1998.

[23] N. Chentanez, B. E. Feldman, F. Labelle, J. F. O'Brien, and J. R. Shewchuk. Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 219–228, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[24] N. Chentanez, T. G. Goktekin, B. E. Feldman, and J. F. O'Brien. Simultaneous coupling of fluids and deformable bodies. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '06, pages 83–89, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[25] N. Chentanez and M. Müller. Real-time simulation of large bodies of water with small scale details. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 197–206, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[26] N. Chentanez and M. Müller. Real-time eulerian water simulation using a restricted tall cell grid. *ACM Trans. Graph.*, 30(4):82:1–82:10, July 2011.

[27] Y.-F. Chiu and C.-F. Chang. GPU-based ocean rendering. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 2125 –2128. IEEE, july 2006.

[28] A. J. Chorin. Vortex models and boundary layer instability. *SIAM Journal on Scientific and Statistical Computing*, 1(1):1–21, 1980.

[29] N. S.-H. Chu and C.-L. Tai. MoXi: real-time ink dispersion in absorbent paper. *ACM Trans. Graph.*, 24(3):504–511, July 2005.

[30] J.-M. Cieutat, J.-C. Gonzato, and P. Guitton. A general ocean waves model for ship design. In *Proceedings of Virtual Concept*. Ecole Supérieure des Technologies Industrielles Avancées (ESTIA), 2003.

[31] M. Coquerelle and G.-H. Cottet. A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies. *J. Comput. Phys.*, 227(21):9121–9137, November 2008.

[32] G.-H. Cottet and P. Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge University Press, 2000.

[33] K. Crane, I. Llamas, and S. Tariq. Real time simulation and rendering of 3D fluids. In H. Nguyen, editor, *GPU Gems 3*, chapter 30. Addison-Wesley, 2007.

[34] J. L. R. d'Alembert. *Essai d'une nouvelle théorie de la résistance des fluides*. David l'aîné, 1752.

[35] M. Desbrun and M.-P. Cani. Smoothed particles: A new paradigm for animating highly deformable bodies. In R. Boulic and G. Hegron, editors, *Eurographics Workshop on Computer Animation and Simulation (EGCAS)*, pages 61–76. Springer-Verlag, NY, USA, Aug 1996. Published under the name Marie-Paule Gascuel.

[36] S. Elcott, Y. Tong, E. Kanso, P. Schröder, and M. Desbrun. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph.*, 26(1), Jan. 2007.

[37] J. D. Eldredge. Numerical simulation of the fluid dynamics of 2D rigid body motion with the vortex particle method. *J. Comput. Phys.*, 221(2):626–648, February 2007.

[38] J. D. Eldredge, T. Colonius, and A. Leonard. A vortex particle method for two-dimensional compressible flow. *J. Comput. Phys.*, 179(2):371–399, July 2002.

[39] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph.*, 21(3):736–744, July 2002.

[40] R. Fattal and D. Lischinski. Target-driven smoke animation. *ACM Trans. Graph.*, 23(3):441–448, Aug. 2004.

[41] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 15–22, New York, NY, USA, 2001. ACM.

[42] B. E. Feldman, J. F. O'Brien, and O. Arikan. Animating suspended particle explosions. *ACM Trans. Graph.*, 22(3):708–715, July 2003.

[43] B. E. Feldman, J. F. O'Brien, B. M. Klingner, and T. G. Goktekin. Fluids in deforming meshes. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pages 255–259, New York, NY, USA, 2005. ACM.

[44] W. Fong and E. Darve. The black-box fast multipole method. *J. Comput. Phys.*, 228(23):8712–8725, Dec. 2009.

[45] N. Foster and R. Fedkiw. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 23–30, New York, NY, USA, 2001. ACM.

[46] N. Foster and D. Metaxas. Realistic animation of liquids. *Graph. Models Image Process.*, 58(5):471–483, September 1996.

[47] A. Fournier and W. T. Reeves. A simple model of ocean waves. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 75–84, New York, NY, USA, 1986. ACM.

[48] R. W. Fox, A. T. McDonald, and P. J. Pritchard. *Introduction to fluid mechanics; 6th ed.* Wiley, Hoboken, NJ, 2004.

[49] R. Fraedrich, S. Auer, and R. Westermann. Efficient high-quality volume rendering of SPH data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1533–1540, Nov. 2010.

[50] J. R. Frisvad, N. J. Christensen, and H. W. Jensen. Computing the scattering properties of participating media using Lorenz-Mie theory. *ACM Trans. Graph.*, 26(3), July 2007.

[51] A. R. Fuller, H. Krishnan, K. Mahrous, B. Hamann, and K. I. Joy. Real-time procedural volumetric fire. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, I3D '07, pages 175–180, New York, NY, USA, 2007. ACM.

[52] M. N. Gamito, P. F. Lopes, and M. R. Gomes. Two-dimensional simulation of gaseous phenomena using vortex particles. In *In Proceedings of the 6th Eurographics Workshop on Computer Animation and Simulation*, pages 3–15. Springer-Verlag, 1995.

[53] P. Goswami and R. Pajarola. Time adaptive approximate SPH. In *Proceedings of the 8th Workshop on Virtual Reality Interactions and Physical Simulations, VRIPHYS 2011*, pages 19–28. Eurographics Association, 2011.

[54] S. T. Greenwood and D. H. House. Better with bubbles: enhancing the visual realism of simulated fluid. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 287–296, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[55] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph.*, 24(3):973–981, July 2005.

[56] J. Günther, S. Popov, H.-P. Seidel, and P. Slusallek. Realtime ray tracing on GPU with BVH-based packet traversal. In *IEEE Symposium on Interactive Ray Tracing, 2007. RT '07.*, pages 113 –118, sept. 2007.

[57] T. Harada, S. Koshizuka, and Y. Kawaguchi. Smoothed particle hydrodynamics on GPUs. In *Proc. of Computer Graphics International*, pages 63–70, 2007.

[58] M. J. Harris, W. V. Baxter, T. Scheuermann, and A. Lastra. Simulation of cloud dynamics on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, HWWS '03, pages 92–101, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[59] H. He, H. Liu, F. Zeng, and G. Yang. A way to real-time ocean wave simulation. In *Computer Graphics, Imaging and Vision: New Trends, 2005. International Conference on*, pages 409 – 415, july 2005.

[60] J. L. Hess. Higher order numerical solution of the integral equation for the two-dimensional neumann problem. *Computer Methods in Applied Mechanics and Engineering*, 2(1):1 – 15, 1973.

[61] J.-M. Hong and C.-H. Kim. Discontinuous fluids. *ACM Trans. Graph.*, 24(3):915–920, July 2005.

[62] J.-M. Hong, H.-Y. Lee, J.-C. Yoon, and C.-H. Kim. Bubbles alive. *ACM Trans. Graph.*, 27(3):48:1–48:4, August 2008.

[63] J.-M. Hong, T. Shinar, and R. Fedkiw. Wrinkled flames and cellular patterns. *ACM Trans. Graph.*, 26(3), July 2007.

[64] C. Horvath and W. Geiger. Directable, high-resolution simulation of fire on the GPU. *ACM Trans. Graph.*, 28(3):41:1–41:8, July 2009.

[65] M. Ihmsen, J. Bader, G. Akinci, and M. Teschner. Animation of air bubbles with SPH. In *GRAPP 2011 - Proceedings of the International Conference on Computer Graphics Theory and Applications, Vilamoura, Algarve, Portugal, March 5-7, 2011*, pages 225–234. SciTePress, 2011.

[66] R. T. Inacio, T. H. C. Nobrega, D. D. B. Carvalho, and A. v. Wangenheim. Interactive simulation and visualization of fluids with surface raycasting. In *Proceedings of the 2010 23rd SIBGRAPI Conference on Graphics, Patterns and Images*, SIBGRAPI '10, pages 142–148, Washington, DC, USA, 2010. IEEE Computer Society.

[67] G. Irving, C. Schroeder, and R. Fedkiw. Volume conserving finite element simulations of deformable models. *ACM Trans. Graph.*, 26(3), July 2007.

[68] H. W. Jensen. Global illumination using photon maps. In *Proceedings of the eurographics workshop on Rendering techniques '96*, pages 21–30, London, UK, UK, 1996. Springer-Verlag.

[69] J. T. Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA, 1986. ACM.

[70] N. Kang, J. Park, J. Noh, and S. Y. Shin. A hybrid approach to multiple fluid simulation using volume fractions. *Computer Graphics Forum*, 29(2):685–694, 2010.

[71] M. Kass and G. Miller. Rapid, stable fluid dynamics for computer graphics. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '90, pages 49–57, New York, NY, USA, 1990. ACM.

[72] D. Kim, S. Lee, O. Song, and H. Ko. Baroclinic turbulence with varying density and temperature. *Visualization and Computer Graphics, IEEE Transactions on*, 18(9):1488–1495, Sept. 2012.

[73] D. Kim, O.-y. Song, and H.-S. Ko. A practical simulation of dispersed bubble flow. *ACM Trans. Graph.*, 29(4):70:1–70:5, July 2010.

[74] J. Kim, D. Cha, B. Chang, B. Koo, and I. Ihm. Practical animation of turbulent splashing water. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '06, pages 335–344, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[75] T. Kim, N. Thürey, D. James, and M. Gross. Wavelet turbulence for fluid simulation. *ACM Trans. Graph.*, 27(3):50:1–50:6, Aug. 2008.

[76] B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O'Brien. Fluid animation with dynamic meshes. *ACM Trans. Graph.*, 25(3):820–825, July 2006.

[77] R. Krasny and L. Kaganovskiy. Computation of vortex ring dynamics. In *International Conference on High Reynolds Number Vortex Interactions*, pages 46–113. Oxford University Press, 2005.

[78] A. M. Kuethe and C.-Y. Chow. *Foundations of Aerodynamics: Bases of Aerodynamics Design*. John Wiley and Sons, 1998.

[79] N. Kwatra, C. Wojtan, M. Carlson, I. A. Essa, P. J. Mucha, and G. Turk. Fluid simulation with articulated bodies. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):70–80, Jan. 2010.

[80] A. Lamorlette and N. Foster. Structural modeling of flames for a production environment. *ACM Trans. Graph.*, 21(3):729–735, July 2002.

[81] S. Lapierre, A. Entezari, and T. Möller. Practicalities of fourier-domain fluid simulation. Technical report, Simon Fraser University, Canada, 2003.

[82] M. Leiseur. *Turbulence in fluids: Stochastic and numerical modelling*. Kluwer Academic Publisher, Dordrecht, The Netherlands, 1990.

[83] M. Levoy. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.*, 8(3):29–37, May 1988.

[84] M. J. Lighthill. Introduction: Boundary layer theory. In L. Rosenhead, editor, *Laminar Boundary Theory*, pages 46–113. Oxford University Press, 1963.

[85] Y. Liu, X. Liu, and E. Wu. Real-time 3D fluid simulation on GPU with complex obstacles. In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, PG '04, pages 247–256, Washington, DC, USA, 2004. IEEE Computer Society.

[86] B. Long and E. Reinhard. Real-time fluid simulation using discrete sine/cosine transforms. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 99–106, New York, NY, USA, 2009. ACM.

[87] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 163–169, New York, NY, USA, 1987. ACM.

[88] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph.*, 23(3):457–462, Aug. 2004.

[89] F. Losasso, G. Irving, E. Guendelman, and R. Fedkiw. Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):343–352, May 2006.

[90] F. Losasso, T. Shinar, A. Selle, and R. Fedkiw. Multiple interacting liquids. *ACM Trans. Graph.*, 25(3):812–819, July 2006.

[91] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled SPH and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):797–804, July 2008.

[92] A. McNamara, A. Treuille, Z. Popović, and J. Stam. Fluid control using the adjoint method. *ACM Trans. Graph.*, 23(3):449–456, Aug. 2004.

[93] V. Mihalef, D. Metaxas, and M. Sussman. Simulation of two-phase flow with subscale droplet and bubble effects. *Computer Graphics Forum*, 28(2):229–238, 2009.

[94] B. Miklós. Real-time fluid simulation using height fields. Semester thesis, Swiss Federal Institute of Technology Zurich, 2004.

[95] M. K. Misztal, R. Bridson, K. Erleben, J. A. Bærentzen, and F. Anton. Optimization-based fluid simulation on unstructured meshes. In *Proceedings of*

Virtual Reality Interaction and Physical Simulation, VRIPHYS 2010. Eurographics Association, 2010.

[96] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[97] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 141–151, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[98] M. Müller, S. Schirm, M. Teschner, B. Heidelberger, and M. Gross. Interaction of fluids with deformable solids: Research articles. *Comput. Animat. Virtual Worlds*, 15(3-4):159–171, July 2004.

[99] R. Narain, J. Sewall, M. Carlson, and M. C. Lin. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Trans. Graph.*, 27(5):166:1–166:8, December 2008.

[100] F. Neyret. Advected textures. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pages 147–153, Aire-la-Ville, Switzerland, 2003. Eurographics Association.

[101] D. Q. Nguyen, R. Fedkiw, and H. W. Jensen. Physically based modeling and animation of fire. *ACM Trans. Graph.*, 21(3):721–728, July 2002.

[102] M. B. Nielsen and R. Bridson. Guide shapes for high resolution naturalistic liquid simulation. *ACM Trans. Graph.*, 30(4):83:1–83:8, July 2011.

[103] C.-V. Pao. *Nonlinear Parabolic and Elliptic Equations*. Plenum Press, New York, NY, 1992.

[104] J. Park, Y. Kim, D. Wi, N. Kang, S. Y. Shin, and J. Noh. A unified handling of immiscible and miscible fluids. *Comput. Animat. Virtual Worlds*, 19(3-4):455–467, September 2008.

[105] S. I. Park and M. J. Kim. Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pages 261–270, New York, NY, USA, 2005. ACM.

[106] D. R. Peachey. Modeling waves and surf. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 65–74, New York, NY, USA, 1986. ACM.

[107] T. Pfaff, N. Thuerey, J. Cohen, S. Tariq, and M. Gross. Scalable fluid simulation using anisotropic turbulence particles. In *ACM SIGGRAPH Asia 2010 papers*, pages 174:1–174:8, New York, NY, USA, 2010. ACM.

[108] T. Pfaff, N. Thuerey, and M. Gross. Lagrangian vortex sheets for animating fluids. *ACM SIGGRAPH 2012 Papers*, 2012.

[109] T. Pfaff, N. Thuerey, A. Selle, and M. Gross. Synthetic turbulence using artificial boundary layers. *ACM Trans. Graph.*, 28(5):121:1–121:10, December 2009.

[110] P. Ploumhans, G. S. Winckelmans, J. K. Salmon, A. Leonard, and M. S. Warren. Vortex methods for direct numerical simulation of three-dimensional bluff body flows: application to the sphere at Re = 300, 500, and 1000. *J. Comput. Phys.*, 178(2):427–463, May 2002.

[111] L. Prandtl. On motion of fluids with very little viscosity. In *Third Congress of Mathematics*. Heidelberg, 1904.

[112] S. Premože, T. Tasdizen, J. Bigler, A. Lefohn, and R. T. Whitaker. Particle-based simulation of fluids. *Computer Graphics Forum*, 22(3):401–410, 2003.

[113] Y. H. Qian, D. D'Humires, and P. Lallemand. Lattice BGK models for Navier-Stokes equation. *EPL (Europhysics Letters)*, 17(6):479, 1992.

[114] M. Raab, D. Seibert, and A. Keller. Unbiased global illumination with participating media. In *Proceedings of Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 591–605, 2006.

[115] K. Raveendran, C. Wojtan, and G. Turk. Hybrid smoothed particle hydrodynamics. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, pages 33–42, New York, NY, USA, 2011. ACM.

[116] D. Rempfer. On boundary conditions for incompressible Navier-Stokes problems. *Applied Mechanics Reviews*, 59(3):107–125, May 2006.

[117] A. Robinson-Mosher, T. Shinar, J. Gretarsson, J. Su, and R. Fedkiw. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph.*, 27(3):46:1–46:9, August 2008.

[118] D. Russell and Z. J. Wang. A cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow. *J. Comput. Phys.*, 191(1):177–205, October 2003.

[119] F. Sadlo, R. Peikert, and M. Sick. Visualization tools for vorticity transport analysis in incompressible flow. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):949–956, Sept. 2006.

[120] P. G. Saffman. *Vortex Dynamics*. Cambridge University Press, 1992.

[121] A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph.*, 24(3):910–914, July 2005.

[122] J. A. Sethian. Fast marching methods. *SIAM Rev.*, 41(2):199–235, June 1999.

[123] B. N. Shashikanth, A. Sheshmani, S. D. Kelly, and J. E. Marsden. Hamiltonian structure for a neutrally buoyant rigid body interacting with N vortex rings of arbitrary shape: the case of arbitrary smooth body shape. *Theoretical and Computational Fluid Dynamics*, 22(1):37–64, 2008.

[124] F. Sin, A. W. Bargteil, and J. K. Hodgins. A point-based method for animating incompressible flow. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 247–255, New York, NY, USA, 2009. ACM.

[125] B. Solenthaler and M. Gross. Two-scale particle simulation. *ACM Trans. Graph.*, 30(4):81:1–81:8, July 2011.

[126] B. Solenthaler and R. Pajarola. Density contrast SPH interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, pages 211–218, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[127] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible SPH. *ACM Trans. Graph.*, 28(3):40:1–40:6, July 2009.

[128] B. Solenthaler, J. Schläfli, and R. Pajarola. A unified particle model for fluid-solid interactions. *Comput. Animat. Virtual Worlds*, 18(1):69–82, Feb. 2007.

[129] J. Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[130] J. Stam. A simple fluid solver based on the FFT. *J. Graph. Tools*, 6(2):43–52, September 2001.

[131] J. Stam. Real-time fluid dynamics for games. In *Proceedings of the Game Developer Conference*, GDC 2003. International Game Developers Association, 2003.

[132] D. Terzopoulos, J. Platt, and K. Fleischer. Heating and melting deformable models (from goop to glop). In *Graphics interface 1989*, pages 219–226, Toronto, Ont., Canada, Canada, 1989. Canadian Information Processing Society.

[133] J. Tessendorf. Simulating ocean water. In *SIGGRAPH 2001 course Notes (Course 47)*. ACM, 2001.

[134] N. Thürey, R. Keiser, M. Pauly, and U. Rüde. Detail-preserving fluid control. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '06, pages 7–12, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[135] N. Thürey and U. Rüde. Free surface Lattice-Boltzmann fluid simulations with and without level sets. In *Proceedings of VMV 2004*. Aka GmbH, November 2004.

[136] A. Treuille, A. Lewis, and Z. Popović. Model reduction for real-time fluids. *ACM Trans. Graph.*, 25(3):826–834, July 2006.

[137] A. Treuille, A. McNamara, Z. Popović, and J. Stam. Keyframe control of smoke simulations. *ACM Trans. Graph.*, 22(3):716–723, July 2003.

[138] Y. Vanzine and D. Vrajitoru. Pseudorandom noise for real-time volumetric rendering of fire in a production system. In *IEEE/ EG Symposium on Volume and Point-Based Graphics*, pages 129–136. Eurographics, 2008.

[139] M. Vines and W.-S. Lee. A simple force computation method for two-way fluid-solid coupling in two-dimensional vortical flows. In *Computer Graphics International*, CGI 2011, Ottawa, Canada, 2011.

[140] J. H. Walther. *Discrete Vortex Method for Two-Dimensional Flow Past Bodies of Arbitrary Shape Undergoing Prescribed Rotary and Translational Motion*. PhD thesis, Technical University of Denmark, 1994.

[141] J. H. Walther and P. Koumoutsakos. Three-dimensional vortex methods for particle-laden flows with two-way coupling. *Journal of Computational Physics*, 167(1):39 – 71, 2001.

[142] J. H. Walther and A. Larssen. Two dimensional discrete vortex method for application to bluff body aerodynamics. *Journal of Wind Engineering and Industrial Aerodynamics*, 67-68:183–193, 1997.

[143] C. Wang, Z. Wang, J. Jin, and Q. Peng. Real-time simulation of ocean wave based on cellular automata. In E. Wu, H. Sun, and D. Qi, editors, *Proceedings of CAD/Graphics'2003*, pages 26–31, 2003.

[144] X. Wei, W. Li, and A. Kaufman. Melting and flowing of viscous volumes. In *Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)*, CASA '03, Washington, DC, USA, 2003. IEEE Computer Society.

[145] X. Wei, W. Li, K. Mueller, and A. E. Kaufman. Simulating fire with texture splats. In *Proceedings of the conference on Visualization '02*, VIS '02, pages 227–235, Washington, DC, USA, 2002. IEEE Computer Society.

[146] X. Wei, W. Li, K. Mueller, and A. E. Kaufman. The lattice-boltzmann method for simulating gaseous phenomena. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):164–176, March 2004.

[147] X. Wei, Y. Zhao, Z. Fan, W. Li, F. Qiu, S. Yoakum-Stover, and A. E. Kaufman. Lattice-based flow field modeling. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):719–729, November 2004.

[148] S. Weißmann and U. Pinkall. Real-time interactive simulation of smoke using discrete integrable vortex filaments. In *Proceedings of Workshop in Virtual Reality*

Interactions and Physical Simulation (VRIPHYS) 2009, pages 1–10, Aire-la-Ville, Switzerland, Switzerland, 2000. Eurographics Association.

[149] S. Weißmann and U. Pinkall. Filament-based smoke with vortex shedding and variational reconnection. *ACM Trans. Graph.*, 29(4):115:1–115:12, July 2010.

[150] J. D. Wendt, W. Baxter, I. Oguz, and M. C. Lin. Finite volume flow simulations on arbitrary domains. *Graph. Models*, 69(1):19–32, January 2007.

[151] M. Wicke, M. Stanton, and A. Treuille. Modular bases for fluid dynamics. *ACM Trans. Graph.*, 28(3):39:1–39:8, July 2009.

[152] G. Winckelmans and A. Leonard. Contributions to vortex particle methods for the computation of three-dimensional incompressible unsteady flows. *Journal of Computational Physics*, 109(2):247 – 273, 1993.

[153] C. Wojtan, N. Thürey, M. Gross, and G. Turk. Deforming meshes that split and merge. *ACM Trans. Graph.*, 28(3):76:1–76:10, July 2009.

[154] C. Wojtan, N. Thürey, M. Gross, and G. Turk. Physics-inspired topology changes for thin fluid features. *ACM Trans. Graph.*, 29(4):50:1–50:8, July 2010.

[155] E. Wu, Y. Liu, and X. Liu. An improved study of real-time fluid simulation on GPU: Research articles. *Comput. Animat. Virtual Worlds*, 15(3-4):139–146, July 2004.

[156] J. Wu and J. Wu. Vorticity dynamics on boundaries. In J. W. Hutchinson and T. Y. Wu, editors, *Advances in Applied Mechanics*, volume 32, pages 119 – 275. Elsevier, 1996.

[157] J. C. Wu. Theory for Aerodynamic Force and Moment in Viscous Flows. *AIAA Journal*, 19:432–441, Apr. 1981.

[158] J.-Z. Wu. A theory of threedimensional interfacial vorticity dynamics. *Physics of Fluids*, 7(10):2375–2395, 1995.

[159] J.-Z. Wu, X.-Y. Lu, and L.-X. Zhuang. Integral force acting on a body due to local flow structures. *Journal of Fluid Mechanics*, 576:265–286, 2007.

[160] M. Yang, J. Lu, A. Safonova, and K. J. Kuchenbecker. GPU methods for real-time haptic interaction with 3D fluids. In *Haptic Audio visual Environments and Games, 2009. HAVE 2009. IEEE International Workshop on*, pages 24 –29. IEEE, nov. 2009.

[161] G. D. Yngve, J. F. O'Brien, and J. K. Hodgins. Animating explosions. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 29–36, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[162] J.-C. Yoon, H. R. Kam, J.-M. Hong, S. J. Kang, and C.-H. Kim. Procedural synthesis using vortex particle method for fluid simulation. *Computer Graphics Forum*, 28(7):1853–1859, 2009.

[163] J. Yu and G. Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 217–225, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[164] J. Yu, C. Wojtan, G. Turk, and C. Yap. Explicit mesh surfaces for particle based fluids. In *Proceedings of Eurographics 2012*. Eurographics Association, 2012.

[165] C. Yuksel, D. H. House, and J. Keyser. Wave particles. *ACM Trans. Graph.*, 26(3), July 2007.

[166] F. Zhang, L. Hu, J. Wu, and X. Shen. A SPH-based method for interactive fluids simulation on the multi-GPU. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '11, pages 423–426, New York, NY, USA, 2011. ACM.

[167] Y. Zhao, L. Wang, F. Qiu, A. E. Kaufman, and K. Mueller. Melting and flowing in multiphase environment. *Computers & Graphics*, 30(4):519–528, 2006.

[168] Y. Zhao, X. Wei, Z. Fan, A. Kaufman, and H. Qin. Voxels on fire. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 36–, Washington, DC, USA, 2003. IEEE Computer Society.

[169] Y. Zhao, Z. Yuan, and F. Chen. Enhancing fluid animation with adaptive, controllable and intermittent turbulence. In *Proceedings of the 2010 ACM SIG-*

GRAPH/Eurographics Symposium on Computer Animation, SCA '10, pages 75–84, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[170] H. Zhu, X. Liu, Y. Liu, and E. Wu. Simulation of miscible binary mixtures based on Lattice Boltzmann method: Research articles. *Comput. Animat. Virtual Worlds*, 17(3-4):403–410, July 2006.

[171] Y. Zhu and R. Bridson. Animating sand as a fluid. *ACM Trans. Graph.*, 24(3):965–972, July 2005.