INSTITUTO NACIONAL DE MATEMÁTICA PURA E APLICADA

On the simulation of fluids for computer graphics

Ives José de Albuquerque Macêdo Júnior

Adviser: Luiz Carlos Pacheco Rodrigues Velho

Rio de Janeiro
November 2007

Aos meus pais e às minhas irmãs.

"— Alô, cotovia!
Aonde voaste,
Por onde andaste,
Que saudades me deixaste?

— Andei onde deu o vento.
Onde foi meu pensamento
Em sítios, que nunca viste,
De um país que não existe ...
Voltei, te trouxe a alegria.

— Muito contas, cotovia!
E que outras terras distantes
Visitaste? Dize ao triste.

— Líbia ardente, Cítia fria,
Europa, França, Bahia ...

— E esqueceste Pernambuco,
Distraída?

— Voei ao Recife, no Cais
Pousei na Rua da Aurora.

— Aurora da minha vida
Que os anos não trazem mais!

— Os anos não, nem os dias,
Que isso cabe às cotovias.
Meu bico é bem pequenino
Para o bem que é deste mundo:
Se enche com uma gota de água.
Mas sei torcer o destino,
Sei no espaço de um segundo
Limpar o pesar mais fundo.
Voei ao Recife, e dos longes
Das distâncias, aonde alcança
Só a asa da cotovia,
— Do mais remoto e perempto
Dos teus dias de criança
Te trouxe a extinta esperança,
Trouxe a perdida alegria."

Cotovia, Manuel Bandeira

Agradecimentos

Antes de tudo, gostaria de agradecer a meus pais por sempre se esforçarem para me oferecer uma boa educação. Agradeço o apoio e o conforto que eles e minhas irmãs me deram ao longo de todos esses anos perto e longe de casa.

Agradeço a Sílvio Melo, grande amigo e orientador de graduação, o qual me apresentou a Computação Gráfica e as belezas dessa disciplina. Sem seus ensinamentos e sua confiança em mim, jamais chegaria até aqui.

Sou muito grato a meu orientador Luiz Velho por me agüentar nesses anos de mestrado, confiar em mim e me guiar até o desenvolvimento deste trabalho, o que lhe deve ter exigido bastante paciência.

Aos professores Luiz Henrique de Figueiredo e Paulo Cezar Pinto Carvalho, agradeço pelas excelentes aulas e conversas acerca de vários temas.

Agradeço aos demais professores que tive aqui no IMPA: Alfredo Iusem, Benar Fux Svaiter, Vladas Sidoravicius, Mikhail Solodov, André Nachbin, Christian Schaerer e Thomas Lewiner, os quais me ensinaram, com a excelência característica do IMPA, praticamente tudo o que sei de matemática.

Aos demais funcionários do IMPA, agradeço o seu esforço e cuidados para manter excelentes as condições de estudo e trabalho no instituto.

Gostaria de agradecer a todos os membros de minha família que me acolheram tão bem quando de minha chegada ao Rio de Janeiro, em particular, à minha tia Ivone.

Fico grato a todos aqueles com quem tive o enorme prazer de dividir uma moradia longe de casa: Vivi, Xanda, Ivana, Renato, Börje, Vitor, Paulo e Fernando. Foram muito felizes nossas noites de conversas e refeições em família, mais recentemente, os almoços de Betinha.

Agradeço aos colegas e amigos do Visgraf pelo companheirismo, em particular a Emilio, Dimas, Geisa, Cicconet, Hedlena, Margareth, Ana Regina, Anderson, Aldo, Giordano, Sérgio "Meu Caro", Evilson, Serginho, Otávio, Adriana, Ricardo, Dalia, Asla e Esdras "Meus Ovos".

Jamais esquecerei dos demais amigos que fiz aqui no Rio (minha outra e querida família de longe de casa): Emilio, Cristina, Júlio, Papito, Ítalo, Augusto, Daniel, Marcelo RH, Tertuliano, Guilherme, Marcelo Cicconet, Paty, Zanforlin, Laura, Braulia, Vanessa, Adriana, Geisa, Dimas, Clarisse... Ufa! Desisti de tentar enumerá-los... Enfim, MUITO OBRIGADO a todos os meus amigos por todo o seu carinho e amizade! Acharam que eu iria esquecer?!

Não deixaria de lembrar dos amigos que ajudaram na revisão dessa dissertação: Dimas, Emílio, Börje e Júlio Daniel. Obrigado, companheiros!

Finalmente, devo expressar minha gratidão de maneira geral ao IMPA, pela excelência de suas instalações e pessoas, e ao CNPq, por me conceder duas bolsas de estudos (mestrado e, agora, doutorado).

P.S.: Sei que ainda tenho MUITO mais o que (e a quem) agradecer, mas, por favor, compreenda que já tá na hora d'eu terminar essa bagaça...

Resumo

A onipresença e a complexidade dos fenômenos naturais são responsáveis tanto pela alta demanda de *softwares* capazes de simulá-los quanto pelas dificuldades em projetar tais ferramentas. Na indústria de efeitos especiais, existe uma necessidade de retratar fenômenos decorrentes da interação entre elementos da natureza. Entretanto, os requisitos impostos por domínios de aplicação como esse diferem daqueles oriundos dos cenários clássicos da engenharia, aplicação tradicional da Dinâmica dos Fluidos Computacional.

O aumento tanto no poder computacional quanto na disponibilidade de memória vêm possibilitando a simulação de vários fenomênos naturais em *hardware* de baixo-custo, quando a *plausibilidade visual* desses fenômenos é suficiente. Esses desenvolvimentos, e a demanda por animações "realísticas" tanto para filmes quanto para jogos de computador, têm encorajado a comunidade de computação gráfica a trabalhar no projeto de novas técnicas para a simulação de fenômenos físicos especializadas nesses domínios de aplicação.

Por essas razões, a literatura sobre simulação de fluidos, já bastante extensa devido a suas importantes aplicações nas engenharias, tem crescido rapidamente por causa da atenção recente da comunidade de computação gráfica. Esse cenário tem nos motivado a estudar os fundamentos da simulação de fluidos, ainda com vistas a sua utilização em animação por computador. Com o propósito de reportar nossos estudos, esta dissertação tem dois objetivos principais: *aprender* tanto os fundamentos teóricos da dinâmica dos fluidos quanto os principais métodos para simular escoamentos de fluidos; e *auxiliar outros estudantes*, agrupando os principais conceitos envolvidos na animação de fluidos num texto introdutório, junto a referências relevantes para tópicos específicos e tendências da pesquisa em computação gráfica.

Palavras-chave: dinâmica dos fluidos, animação por computador, fenômenos naturais, equações de Euler, equações de Navier-Stokes, Fluidos Estáveis, SPH

Abstract

The ubiquity and complexity of natural phenomena are responsible both for the high demand of software capable to simulate them and for the difficulties in designing such tools. In the special effects industry, there is a need of depicting phenomena arising from the interactions among the elements of nature. However, the requirements imposed by this application domain differ from those in the classical engineering settings with which Computational Fluid Dynamics is traditionally concerned.

The increase in computing power and memory availability has made possible to simulate many natural phenomena in commodity hardware, when the *visual plausibility* of computed motions is enough. These developments and the demand for "realistic" animations, both for films and games, have encouraged the graphics community to work on the design of specialized methods and techniques to simulate physical phenomena for these applications.

For these reasons, the literature on fluid simulation, already very large due to its important applications in engineering, has been growing fast because of the recent attention from the computer graphics community. This scenario has motivated us to study the fundamentals of fluid simulation, still with computer graphics applications in mind. With the purpose of reporting our studies, the goals of this thesis are twofold: to *learn* both the theoretical fundamentals of fluid dynamics and the main methods for the computer simulation of fluid flows; and to *help other students*, by grouping the main concepts involved in simulating fluids for animation in an introductory text along with relevant references to specific topics and current research trends.

Keywords: fluid dynamics, computer animation, Euler's equations, natural phenomena, Navier-Stokes' equations, Stable Fluids, SPH

Contents

1 Introduction **1**
- 1.1 Motivation and Goals . . . 1
- 1.2 Plausibility *versus* Accuracy . . . 2
- 1.3 Overview of the Thesis . . . 3

2 Commented Bibliography **5**
- 2.1 Mathematics and Physics of Fluids . . . 6
- 2.2 Computational Fluid Dynamics . . . 7
- 2.3 Fluid Simulation in Computer Graphics . . . 9
- 2.4 Comments and Further Reading . . . 11

3 The Equations of Motion **13**
- 3.1 Basic Principles and Assumptions . . . 14
- 3.2 Lagrangian and Eulerian Descriptions . . . 16
- 3.3 Conservation of Mass and Incompressibility . . . 21
- 3.4 Balance of Momentum . . . 24
- 3.5 Ideal Fluids and Euler's Equations . . . 26
- 3.6 Newtonian Viscous Fluids and the Navier-Stokes Equations . . . 27
- 3.7 Pressure and Incompressibility . . . 30
- 3.8 Rotation and Vorticity . . . 32
- 3.9 Comments and Further Reading . . . 37

4 Numerical Simulation of Fluids for Animation **39**
- 4.1 Stable Fluids . . . 40
  - 4.1.1 Flow regime and governing equations . . . 40
  - 4.1.2 Field representation and *spatial* discretization . . . 40
  - 4.1.3 Operator splitting/fractional-stepping . . . 42
  - 4.1.4 Semi-Lagrangian advection . . . 44

4.1.5 Explicit forcing and vorticity confinement 45
4.1.6 Implicit diffusion 45
4.1.7 Pressure projection 46
4.1.8 Experiments 48
4.2 Smoothed Particle Hydrodynamics 53
4.2.1 Flow regime and governing equations 53
4.2.2 Field representation and *fluid* discretization 54
4.2.3 The discretized governing equations 58
4.2.4 Experiments 61
4.3 Comments and Further Reading 63

5 Final Remarks **65**

A A Simple *Stable Fluids* Solver **67**

B A Simple *Smoothed Particle Hydrodynamics* Solver **75**

List of Figures

4.1 A simple flow example 50
4.2 A more complex example 51
4.3 Generation of vorticity in Navier-Stokes' flows 52
4.4 Animating a breaking dam 62

Chapter 1

Introduction

> *"As Aladdin rubbed the lamp to try to get a better look, the lamp sprang magically to life! It was all Aladdin could do to hold onto the bucking lamp as it spewed colored smoke and magic sparkles! Like a volcano erupting, the lamp launched a long, blue stream upward. The blue smoke twisted and expanded as it rose toward the ceiling. Finally, it became an enormous, blue genie!"*, **Disney's Aladdin**[1]

1.1 Motivation and Goals

The ubiquity and complexity of natural phenomena are responsible both for the high demand of tools capable of simulating them and for the difficulties in designing such tools. In the special effects industry, there is a need to depict phenomena arising from the interactions among elements of nature (e.g. clouds, mountains, rivers, trees). Frequently, those interactions either cannot just be captured by real cameras (e.g. explosions of space-stations) or they are too expensive, even dangerous, to shoot in location (e.g. the inundation of a populated city, or fire spreading through a forest). Many other applications demand visually plausible computer simulations of natural phenomena, including *computer games* and *flight simulators*.

The increase in computing power and memory availability has made possible to simulate many natural phenomena in commodity hardware, when the visual plausibility of computed motions is enough. These developments

[1] Story Plot as reported in the Animated Storybook at Disney.com, c.f. http://www.geocities.com/aladdin_it/Aladdin_eindex.html

and the demand for "realistic" animations have encouraged the computer graphics community to work on the design of methods and techniques to simulate physical phenomena. For these reasons, the computer animation literature on simulation of the elements of nature has grown significantly in the past decade. Many of those works deal with the animation of fluid flows, phenomena omnipresent in our everyday life and complex *per se*. In fact, when asked which is the single hardest shot they did in *"Shrek"*, the *DreamWorks SKG* principal and producer of *"Shrek"*, Jeffrey Katzenberg, answered: *"It's the pouring of milk into a glass."*

As such, the literature on fluid simulation, already very large due to its important applications in engineering, has been growing fast because of the recent attention from the computer graphics community. This has motivated us to study more about the simulation of fluids, with computer graphics applications in mind. For these reasons, the goals of this thesis are twofold:

- to *learn* both the theoretical fundamentals of fluid dynamics and the main methods for the computer simulation of fluid flows;
- to *help other students* by grouping the main concepts involved in the simulation of fluids for computer graphics in an introductory text along with relevant references to specific topics and current research trends.

Since most of the literature on the foundations of Computational Fluid Dynamics (CFD) was written either with theoretical or engineering purposes, where the needs are rather different from computer graphics', one should take care on how to employ the developments from CFD in animation applications. We comment on these distinct viewpoints in the following section.

1.2 Plausibility *versus* Accuracy

With its roots in engineering disciplines, most works on the numerical simulation of fluid dynamics focus on the *accuracy* of computed flows. However, computer graphics applications demand different qualities in the methods they employ to simulate fluid motion: *plausibility*, *efficiency* and *complexity*.

Visual plausibility of the motion is paramount in applications desiring to immerse a person in some virtual environment (e.g. *flight simulators*, *computer games*, *theatrical films*). This is the driving goal of techniques in computer graphics [BHW96], but, perhaps influenced by the classical literature

on Computational Fluid Dynamics, it is not always exploited in published works (which stay conservative, relying just on accurate methods).

Many works have benefited from exploiting this "relaxed" accuracy requirement in designing more *efficient* schemes to simulate natural phenomena. Even this efficiency need is somewhat different from that in engineering, where high-end computers and clusters are often available to perform days (sometimes *weeks*) of computations. Animators need a quick feedback from a simulation (often performed just on a "good" desktop), since many parameters are tunned to achieve a desired motion effect. This way, the efficiency requirements in computer graphics applications are, in a sense, more stringent than in engineering's. The systematic exploitation of visual perception and human attention in designing efficient methods for plausible animation of physical phenomena is a subject of active research [OD01, ODGK03, O'S05].

Another property which needs to be taken into account concerns the *complexity* of the method with respect to its implementation and its integration with other simulators designed for other physical phenomena. This is more of a practical "requirement", since a technique is hardly implemented if it is too hard to code, and the special effects industry is often interested in motions resulting from complex interactions among rigid, deformable and fluid objects ("multiphysics" interactions). Complexity also plays a role in the interface used for simulations, as the animators are not experts in fluid dynamics, there is a requirement that the methods should be easy to use (besides the discussed efficiency of feedback).

All those differences in requirements make it difficult to directly adopt standard CFD techniques in computer graphics. Therefore, there is a need to develop specialized algorithms and methods for special effects, what has been accomplished by both adapting existing techniques from CFD and developing novel ones to meet the graphics' requirements.

As this is an introductory text, we will not pursue further issues regarding these requirements, they are overcome by employing well adopted techniques to simulate fluid motion for computer graphics applications.

1.3 Overview of the Thesis

This thesis is organized to provide the reader a gradual introduction to the simulation of fluids for computer graphics applications. In Chapter 2 we comment part of the available literature concerned with the main topics treated

on this text: *mathematics and physics of fluids*, *numerical analysis and computational fluid dynamics* and *computer animation of fluid flows*. In Chapter 3, the governing equations that model the motion of fluids are derived directly from basic physical and mathematical assumptions. That chapter provides the fundamentals of mathematical fluid dynamics needed to understand most of the computer graphics literature on fluid animation. In Chapter 4 we describe two of the most influent methods for fluid animation introduced to computer graphics: *Stable Fluids* and *Smoothed Particle Hydrodynamics*. In Chapter 5, a discussion of some efforts which have been (and are intended to be) made concludes the main text. For completeness, appendices A and B contain the `C` code of our basic implementations of *Stable Fluids* and *SPH*.

Chapter 2

Commented Bibliography

This chapter is intended to provide the reader with an overview (somewhat biased) to available literature both on the prerequisites and fundamentals of mathematical and computational fluid dynamics and on works developed in the computer graphics community to simulate the natural phenomena of fluid flows with animation and depiction purposes.

Both theoretical and numerical fluid dynamics communities are very prolific. For this reason, we have neither the intention to provide a comprehensive list of published works nor the ambition to give a complete picture of this vast discipline. Even the restriction to those efforts in the computer graphics literature would be a superb task to anyone pursuing such goal. That is why, in the present chapter, we hope to provide more something like a *"student report"*. A guide for those interested in studying fluid mechanics with animation ambitions. The discussion is restricted to materials relevant to the simulation of fluids and doesn't even touch efforts on the visualization of simulated flows. The goal of providing a guide for students and researchers starting in this field motivated us to employ a different writting style for this chapter, almost informal compared to the following chapters.

We first provide references on the mathematical modeling of the physics of fluids and comment on the disciplines required to be able to follow those references. The discussion following that concerns the literature dealing with the discretization of the governing equations and differential equations in general, as well as the basics of numerical analysis and linear algebra on which the "computational theory" relies. After that, we overview published works of the computer graphics community regarding the animation of fluid flows. This chapter ends with comments and some tips for further studies.

2.1 Mathematics and Physics of Fluids

As would be expected, a good understading of the foundations of mathematical and computational fluid dynamics requires knowledge of basic graduate mathematics. To be able to follow the references we cite here, the reader should feel comfortable with *real analysis of several variables*, *linear algebra* (both theoretical and computational) and some elements from the theory (and practice) of *ordinary* and *partial differential equations* (and, sometimes, results regarding *functions of one complex variable*). References for these subjects can be provided by people in the math department nearest to you.

Assuming those subjects are familiar to the reader, we provide the references which helped us in preparing this text. The literature on mathematical modeling of the physics of fluids is rather broad, the available textbooks focus on varied audiences (e.g. mechanical engineers, physicists and applied mathematicians) and topics (e.g. aero, hydro and fire dynamics). Our choices rely on the bibliography adopted in an introductory graduate course on fluid dynamics, directed to applied mathematics students, and some references we found (and enjoyed) on our way through.[1]

The classical [Bat99] provides a **broad** (and *verbal*) treatment of the physics and mathematical modeling of fluid dynamics, very nice a reference for those interested (and somewhat mathematically biased) in the subtleties of modeling the physics of fluids. Those who prefer a more formal (but intended to be introductory) view of fluid mechanics will appreciate the *succinct* approach adopted in [CM93]. We recommend [Mey82] as a good balance on the presentation of the physics and the mathematical aspects of fluid dynamics (not to mention the book was written in a concise and clear language directed to students of applied mathematics and physics).

For those who can read portuguese, the books [Nac01, MN91] provide both an introduction to mathematical fluid dynamics. Nachbin's text is more accessible to beginners who are familiar with elements of complex analysis of one variable, while Melo and Neto's is intended for those comfortable with the language of differential equations (altought the first chapters provide a nice structure for deriving the governing equations of fluid motion and influenced our developments in the next chapter).

We also found useful the course notes [Nac07, Mei07]. Altought the

[1] The mentioned course was lectured by Professor André Nachbin on the March-June semester of 2007 here at IMPA.

first is only available at IMPA's copy room, the notes of Professor Chiang Mei can be found on the internet (by October 2007) as part of a "school-wide modular program for *Fluid Mechanics*" hosted at the MIT's website `http://web.mit.edu/fluids-modules/www/`.

2.2 Computational Fluid Dynamics

Since the available references for CFD is even larger than that dedicated to mathematical fluid dynamics and the methods most commonly used in computer graphics for the simulation of fluid flows are rather classical, we chose to study these specific methods rather than attempt to cover this overwhelming literature. We pursued standard textbooks on computational linear algebra and the numerical analysis of differential equations as well as some of the classical papers most frequently cited by the computer graphics community.

On the subject of numerical linear algebra, we adopted the textbooks [TB97, Dem97]. The first of these introduce both basic and advanced methods in a very modular way, suitable both for teaching this subject as well as for individual study. Demmel's book complements that text providing a nice covering of *state-of-the-art* techniques and issues related to their computer implementation. For those who aren't still comfortable with the theory of linear algebra, but are interested in studying it with applied endeavours, we recommend [Str80] as a very didatic reference with plenty of examples.

With respect to classical numerical analysis, the book [SB02] provides a rather complete covering of the basic problems and solutions. For implementation of the methods by means of computer programs, the *Numerical Recipes* series [PTVF92] are the classic reference, providing lots of comments and source code (available in `FORTRAN`, `C` and `C++`).

The reason for providing all those references on numerical linear algebra and analysis is to build the foundations on which the *numerical analysis of differential equations* is supported. A good introduction to this topic is [Ise96], it contains a study of discretizations of both ordinary and partial differential equations with simple examples without restricting the text to one particular method (e.g. *finite differences schemes*, *finite elements methods*). For a discussion on issues related to the discretization of ODE's, the book [AP98] provides a thorough analysis of numerical schemes for both non-stiff and stiff equations. This is one reason for which concepts as *consistency*, *stability* and *convergence* of time-stepping methods are introduced, what is

done in a manner similar to that applied on numerical methods for PDE's.

As one of the most employed methods to discretize differential equations, *finite difference schemes* comprehend the most used methods in computer graphics to simulate the physics of fluids through their governing equations. The references we recommend to study FD methods are the textbooks by LeVeque and Strikwerda [LeV07, Str04] and an unfinished monograph by Trefethen [Tre96] (freely available on the internet by October 2007). LeVeque's book covers the numerical analysis of finite difference methods for both ordinary and partial differential equations in a formal, but still very didatic, way. Strikwerda's text is a bit more dry and focuses on FD schemes for PDE's, but is a classic reference and has a nice material on stability analysis. Trefethen's unfinished book is a fine (low cost, but of good quality) reference for some topics on finite differences methods, but sometimes the ausence of interesting stuff cited in the text is rather annoying. *No free lunch...*

Provided the fundamentals of the basic numerical methods for differential equations and their analysis, we comment highly cited (in the computer graphics fluid simulation literature) papers on numerical analysis and CFD.

We believe the most cited CFD paper in the CG literature is [HW65], in this work, Harlow and Welch introduced the *marker and cell* (MAC) method to simulate the behaviour of non-steady viscous incompressible free-surface fluid flows. That paper used a finite-difference scheme to discretize the incompressible Navier-Stokes equations on a *staggered grid* and a set of marker particles, advected by the computed flow, both to determine the computational cells filled with fluid and to depict/visualize the free-surface flow.

Although the use of a staggered grid contributed to diminish effects of numerical dissipation, the direct discretization of the NSE by an explicit finite difference scheme imposed to [HW65], as stability requirements (similar to the CFL condition derived in [CFL67] to linear, constant coefficients PDE's), severe (to CG applications at least) restrictions on the step size used by the time-stepping method. Those restrictions encouraged the computer graphics community to look for computationally cheap strategies to overcome the CFL condition. The most applied of those strategies was introduced in CG by Jos Stam at SIGGRAPH in 1999 [Sta99], it employed a splitting method [MQ02, PTVF92] to decompose the Navier-Stokes equations into a sequence of simpler subproblems to which specialized methods were used. The most important of these methods tackled the *convection* and the *diffusion* equations and the incompressibility condition with, respectively, a *semi-Lagrangian advection* scheme [SC91], a standard (implicit) *backward*

Euler discretization scheme [Ise96] and Chorin's projection method [Cho68]. These improvements made possible to take arbitrarily large time steps in the simulation, at the expense of some loss in accuracy (but still producing motions **plausible** enough for animation purposes).

The relatively coarse grids used in animation applications, combined with other factors, contributed to cause much numerical diffusion in the flow simulations, evidenced by loss of the swirling behaviour of rotating gases. To tackle this problem, Fedkiw et al. [FSJ01] employed a forcing term, the *vorticity confinement* force, in the Euler equations which induced an increase in the vortical details in the flow. This was done based on the work of Steinhoff and Underhill [SU94] in the interaction of vortex rings arising in simulations of air flows around helicopters' rotor blades.

More recently, with the widespread use of *smoothed particle hydrodynamics* (SPH) methods in the computer graphics community, many articles have cited the seminal works of Lucy, Gingold and Monaghan [Luc77, GM77] on the application of particle systems (also known as meshfree/meshless methods) to simulate astronomical fluids and study the formation of stars.

2.3 Fluid Simulation in Computer Graphics

Physically-plausible animation of fluid motion is a long-standing goal in the computer graphics community. Many models to describe the complex phenomena arising in the interaction of fluids with the environment have been proposed for computer depiction. In this chapter, we overview some classical and *some* recent papers from the graphics literature which take an approach grounded on the governing equations of fluid flows.

The work of Kass and Miller [KM90] was one of the first to solve PDE's for the computer animation of fluids. Their model departed from the *shallow-water/long-wave* equations to model the height-field (free-surface) of a travelling water wave with amplitude and depth small compared to its wavelength. After neglecting the non-linear term in these equations, they arrived at the classical *wave equation*, a systematic derivation (from Boussinesq's) of both shallow-water and wave equations can be found in [Nac07]. To numerically integrate the resulting PDE, Kass and Miller adopted an implicit finite-differences scheme, based on the *alternating-direction method* when simulating "3D" waves, to avoid the requirement of small time-steps.

Restricting to inviscid (non-viscous), irrotational and incompressible flows,

[WH91] was able to animate (non-turbulent) aerodynamic transport with a modeling approach similar to that employed in the classical potential theory of fluids [CM93]. In [SF93], Stam and Fiume depicted gaseous flows using elements from the theory of turbulence and *advection-diffusion* equations.

Only in 1995, Chen and da Vitoria-Lobo [CdVL95] used a FD-discretization of the full 2D Navier-Stokes equations to animate fluid flows. To "simulate" 3D fluids, their work solved the 2D NSE and used a height-field, calculated from the (scaled) pressure values, to depict a free-surface. In the same year, Stam and Fiume introduced in computer graphics the *Smoothed Particle Hydrodynamics* method of [Luc77, GM77] to depict the motion of gases [SF95].

It seems the first work to employ the 3D Navier-Stokes equations in the animation of fluids was developed by Foster and Metaxas in 1996 [FM96a, FM96b]. Their work uses the MAC-scheme of Harlow and Welch [HW65] to simulate viscous incompressible flows, and presented detailed descriptions of how to deal with various boundary conditions encountered in typical flow regimes. Since Foster and Metaxas were interested in providing tools for computer **animation** of fluids, their subsequent work dealed with *control* of fluid flows [FM97a] and coarse-grid simulations of turbulent gases [FM97b].

Although Foster and Metaxas were able to animate full three-dimensional flows, the adoption of the original MAC-scheme forced time-steps so small that the refinement process commonly employed in animation production was rendered inviable. This problem inspired the work of Stam [Sta99], in which the *Stable Fluids* (SF) scheme was introduced. Employing a time splitting of the Navier-Stokes equations, Stable Fluids solves a sequence of "simple" PDE's using classic unconditionally stable methods, allowing the simulation to take arbitrarily large time-steps without numerical blow-up.[2]

The original Stable Fluids method suffered from severe numerical diffusion and volume loss in its simulations, these problems were circumvented in [FSJ01, FF01], which introduced "vorticity confinement" forces to keep rotational detail in gaseous flows and the use of level-sets [OF03] to represent and advect the fluid-air interface. These improvements made possible the simulation of gases and bulk water with high visual fidelity and details. Latter, [EMF02] coupled level-sets with particles and a simplified semi-Lagrangian advection scheme which, coupled with the hierarchical representation of the computational domain presented in [LGF04], allowed an efficient simulation of stunning details in water flows.

[2] In chapter 4, we provide a more detailed description of the Stable Fluids method.

Since Stam's seminal work [Sta99], many authors developed extensions to the basic Stable Fluids scheme [FSJ01, FF01, Sta01, NFJ02, CMIT02, FOA03, Sta03, RNGF03, GBO04, CMT04, SRF05] and alternative approaches [FOK05, FOKG05, KFCO06, ETK+07, BBB07, CFL+07], although they adopted the general splitting structure of the orginal Stable Fluids method.

Other interesting researches regard meshfree (particle-based) methods. Even though introduced in computer graphics by [SF95] for gases and [DG96] for elastic solids, the SPH method only became popular after the work of Müller et al. [MCG03], in which it was demonstrated that it could be used to simulate 3D fluids at interactive rates. Althought one of the most popular, [MCG03] was not the first to use physically-based particle methods to simulate the Navier-Stokes equations. In [GLG95], Gamito et al. presented a particle method based on the vorticity formulation of the governing equations to simulate gaseous flows. Also in 2003, Premože et al. [PTB+03] employed the *Moving Particle Semi-implicit* (MPS) scheme to simulate water flows, but the use of MPS required a grid structure to solve a Poisson equation, which did not free the simulation from using grids. Since then, many papers proposed extensions and other meshless schemes to simulate fluid flows [MST+04, MSKG05, PK05, PPLT06, CACC06, BT07, APKG07].

The computer graphics literature on the simulation of fluid flows is large and diverse. Many people is working to develop efficient methods to generate visually realistic fluid imagery. Interesting research trends include *flow control for animation* [FM97a, TMPS03, MTPS04, FL04, REN+04, SY05a, SY05b, TKPR06, KMT06], *alternative methods* [Thu07, BWHT07, WBOL07, GN07, SMML07] and implementations on *graphics processing units* (GPU's) [Har04, SCC04, SCC05, CLT07].

More introductory references on the simulation of fluid flows (and related topics) for computer graphics applications can be found in the course notes [BMF07, DEF+04, WB01] and in the overview papers [FM00, Sta00, Igl04].

2.4 Comments and Further Reading

We presented an overview of some available references on the theoretical and computational aspects of fluid dynamics and how they have been exploited by the computer graphics community to simulate fluid motion for animation.

In this chapter, we intended to provide the citations most recurrent in the fluid animation literature. Our belief is that a more thorough understanding

of the discipline of mathematical fluid dynamics can be acquired through the study of classic texts on the subject. Besides those cited, some of the classics include [Lam93, CF76, HM76, Whi99]. Another which seems to adopt an interesting viewpoint, and somewhat different from the cited texts, is the book [WG00] that models and simulates fluids by *cellular automata*, an approach already brought to computer graphics and has generated rather impressive imagery [Thu07].

Other references which seem to be worthy, but we didn't inspect yet, regard the *generalized simplified marker and cell* (GENSMAC) method [TM94, TFC$^+$01] and its use in the simulation of 3D flows, both Newtonian and non-Newtonian [TDM96, TGC$^+$04]. This scheme is based on earlier work by Amsden and Harlow [AH70] which introduced the *simplified marker and cell* (SMAC) method and which has also been employed in the (real-time) animation of fluid flows by an implementation on GPU's [SCC04, SCC05].

An interesting result from studying the mathematics of fluid dynamics is that most of the modeling done for fluids is very similar to that employed in mathematical elasticity. Hence, for those also interested in the physically-based animation of elastic media, we recommend the textbooks [MH94, Lov44] as the analogous to those references cited in the first section of this chapter. We have also found a reference written in portuguese [Pat87] which would provide an analogous for Nachbin's text, but may be a bit harder to find because it is *out-of-print* by now.

Chapter 3

The Equations of Motion

This chapter is devoted to present the basic concepts and results from mathematical fluid dynamics. The governing equations of fluid motion will be derived from the fundamental conservation laws and assumptions of continuum fluid mechanics. Our derivations will be kept general up till the point where the need to specialize our results to the most commonly simulated flows in computer graphics is necessary (i.e. *incompressible inviscid flows* and *incompressible Newtonian flows with uniform viscosity*).

From this theoretical treatment, we hope to gain a solid understanding of the assumptions behind the equations of motion used in physically-based fluid animation. We have chosen the concepts and results most recurrent in the computer graphics literature and some we believe could be more exploited in the development of animation techniques.

For reasons of space and time we have chosen to omit derivations and results from some topics of continuum fluid mechanics also exploited by the computer graphics community (e.g. *the shallow-water equations* and *non-newtonian fluids*). However, at the end of this chapter, we provide some comments about those topics and references for further reading.

Our developments result from studying (some topics from) five books [CM93, MN91, Nac01, Mey71, Bat99] and two sets of lecture notes from courses on "Fluid Dynamics" [Nac07, Mei07]. If a more in-depth treatment of mathematical fluid dynamics is needed, the reader is strongly encouraged to pursue those references.

3.1 Basic Principles and Assumptions

To mathematically model the physical phenomenon of fluid motion, we need to make some basic assumptions about our objects of study. The theory of continuum fluid mechanics is grounded on principles which every fluid flow is supposed to obey:

- *the law of conservation of mass*: also known as the *Lomonosov-Lavoisier law*, states that

  > "the mass of a closed system of substances will remain constant, regardless of the processes acting inside the system."

  In fact, this law is only **approximately** true. Since, according to special relativity, the relativistic/apparent mass (m) depends on one's frame of reference. However, for low speeds (much lower than light's), the variation in apparent mass is negligible to a high accuracy.

  For our mathematical modeling of fluid dynamics, we assume that *"mass is neither created nor destroyed."* $\left(\frac{dm}{dt} = 0\right)$

- *Newton's laws of motion*: the three physical laws which provide relationships between the forces acting on a body and the motion of the body. They form the basis for *classical mechanics*, from which *continuum fluid mechanics* is a subdiscipline. The modern understanding of Newton's three laws of motion is[1]:

First Law "if no external force acts on a particle, then it is possible to select a set of reference frames, called *inertial reference frames*, observed from which the particle moves without any change in velocity."

Second Law "observed from an inertial reference frame, the *net force* $\left(\vec{F}\right)$ on a particle equals the time rate of change of its *linear momentum* ($\vec{p} = m\vec{v}$). This law can also be stated as $\vec{F} = \frac{d\vec{p}}{dt} = m\vec{a}$."

Third Law "whenever A exerts a force on B, B simultaneously exerts a force on A with the same magnitude in the opposite direction."

[1] http://en.wikipedia.org/wiki/Newton%27s_laws_of_motion

The second law defines the *net force* on a body in terms of its *linear momentum*. This principle is called *the law of balance of momentum* and is recurrent in our derivations.[2]

- *the first law of thermodynamics*: which states that

  > "The increase in the *internal energy* (U) of a thermodynamic system is equal to the *amount of energy added to the system as the result of heating* (Q) minus the *amount of energy lost as the result of work done by the system on the surroundings* (W)."[3]

  in equations this means $dU = \delta Q - \delta W$.[4] In simple words, it accounts for the assumption that *"energy is neither created nor destroyed."*

- *continuum hypotesis*: although fluids are composed of molecules that collide with one another and solid objects, at a large scale, they may be considered as a *continuum*, i.e. their properties can be assumed continuous functions defined on subsets of $\mathbb{R}^n$.[5] This assumption is called *the continuum hypotesis* and provides us a very good approximation for the macroscopic phenomena we are interested in.

Besides those general principles of *continuum fluid mechanics*, for easy of presentation, we assume a standard *smoothness convention* [Mey71, CM93]: "the fluid properties (e.g. velocity, density and stress tensor to be introduced later) are smooth enough, with respect to position and time on the closure of the fluid domain (to be mathematically defined lated), in such a way that the standard operations of calculus may be performed on them."

Having set our fundamental assumptions and basic conservation laws, we are able to begin our mathematical modeling of fluid dynamics.

---

[2] "Notice that the surface of the Earth does not define an inertial frame of reference because it is rotating and orbiting and because of Earth's gravity. However, since the speed of rotation and revolution change relatively slowly, the inertial force is tiny. Therefore, Newton's laws of motion remain a good approximation on earth. In a non-inertial frame of reference, inertial forces must be considered for Newton's laws to remain valid."

[3] http://en.wikipedia.org/wiki/First_law_of_thermodynamics

[4] As our derivations assume that no amount of *heat energy* is exchanged in the system (i.e. $\delta Q = 0$), for our purposes, this law reads $dU = -\delta W \iff \delta W = -dU$.

[5] For our considerations, we **always** have $n \in \{2, 3\}$.

3.2 Lagrangian and Eulerian Descriptions

Our object is to **mathematically** describe the motion of a fluid region during a given time interval. Such a description can be based on the idea of studying the trajectory taken by each one of the "infinitesimal fluid particles" that compose the fluid. This idea leads us to associating fluid motion with a geometrical transformation from the configuration this fluid domain presents at a reference time instant into the configuration it presents at another time.

To mathematically model "fluid motion", we first need to formalize the concepts discussed and our intuitions through *mathematical definitions*.

Definition 1 (fluid region)**.** *Let $\mathcal{D} \subseteq \mathbb{R}^n$ be an open and bounded set such that $\partial\mathcal{D}$ is smooth.*[6] *If $\mathcal{D}$ is supposed to be "filled with a fluid", we call $\mathcal{D}$ a* fluid region.

Definition 2 (reference configuration and time instant)**.** *The* reference configuration *is a fluid region Ω_0 fixed at the* reference time instant *$t_0 \in \mathcal{I} \subseteq \mathbb{R}$, where $\mathcal{I}$ is the open* time interval *during which we are interested in studying the fluid dynamics.*

Definition 3 (fluid motion and flow map)**.** *The* fluid motion *is the family $\{\varphi_t\}_{t\in\mathcal{I}}$ of continuous maps $\varphi_t : \overline{\Omega_0} \to \mathbb{R}^n$ whose φ_t, for each $t \in \mathcal{I}$, maps the position $\mathbf{x}_0 \in \Omega_0$ of a fluid particle in the reference configuration to this particle's position at time t. With this, we can define the* fluid flow map *as the function $\varphi : \overline{\Omega_0} \times \mathcal{I} \to \mathbb{R}^n$ such that $\varphi(\mathbf{x}_0, t) = \varphi_t(\mathbf{x}_0)$.*[7]

After mathematically modeling the notion of "fluid motion" as a family of maps, we need to study the properties these maps are supposed to possess (besides continuity). The intuitive idea that two different bodies cannot simultaneously occupy the same portion of space can be expressed as a property for our definition of fluid motion: *"for each $t \in \mathcal{I}$, $\varphi_t\big|_{\Omega_0}$ is smooth and has a smooth inverse on $\Omega_t := \varphi_t(\Omega_0)$"*. Another intuitive property ensures the continuity of $\varphi(\mathbf{x}_0, \cdot) : \mathcal{I} \to \mathbb{R}^n$, for each $\mathbf{x}_0 \in \Omega_0$.[8] Intuitively, this means

[6] Notice the use of our *smoothness convention*. In this case, the convention was used to avoid the technical requirements that allow us to invoke the *Divergence Theorem* in subsequent calculations.

[7] Don't confuse the subscript $_t$ with a time derivative!
And notice that $\varphi_{t_0}(\mathbf{x}_0) = \varphi(\mathbf{x}_0, t_0) = \mathbf{x}_0$.

[8] In fact, we assume φ is smooth on $\Omega_0 \times \mathcal{I}$ and its time derivatives (at least the first two of them) are continuous on $\overline{\Omega_0} \times \mathcal{I}$.

that "no particle can suddenly disappear and reappear at another position", i.e., the trajectory of a fluid particle is (at least) continuous.

Definition 4 (trajectory)**.** *From the definition and properties of φ, we can notice that the* trajectory $\mathbf{x} : \mathcal{I} \to \mathbb{R}^n$ *of a fluid particle initially located at a point $\mathbf{x}_0 \in \Omega_0$ can be defined by $\mathbf{x}(t) = \varphi(\mathbf{x}_0, t)$, and, from our smoothness convention, $\mathbf{x}$ describes a smooth curve in $\mathbb{R}^n$.*

This description of fluid motion through the fluid flow map and the fluid particles' trajectories with respect to the reference configuration is the so called *Lagrangian description* and the points in Ω_0 are known as *material coordinates*.

Another approach to describe the particles' motion is through the definition of a smooth vector field $\mathbf{u} : \overline{\mathcal{D}} \times \mathcal{I} \to \mathbb{R}^n$ defined in a fixed fluid region $\mathcal{D}$. We define $\mathbf{u}(\mathbf{x}, t)$ as the velocity of the fluid particle which passes through $\mathbf{x} \in \mathcal{D}$ at time $t \in \mathcal{I}$:

$$\mathbf{u}(\varphi(\mathbf{x}_0, t), t) = \frac{\partial \varphi}{\partial t}(\mathbf{x}_0, t), \quad \mathbf{x}_0 := \varphi_t^{-1}(\mathbf{x})$$

From this definition of the velocity field $\mathbf{u}$, we are able to provide an equivalent definition to a particle's trajectory $\mathbf{x}$ as the (unique) solution of the ordinary differential equation (sometimes known as the *trajectory equation*):

$$\frac{d\mathbf{x}}{dt} = \mathbf{u}(\mathbf{x}, t), \quad \mathbf{x}(t_0) = \mathbf{x}_0$$

This is the so called *Eulerian description* and the points in $\mathcal{D}$ are known as *spatial coordinates*.

Defined the trajectory of a fluid's particle, we would like to measure the rates in which quantities change along a particle's trajectory, i.e., we want to be able to take time derivatives along trajectories. With this goal, let $f : \mathcal{D} \times \mathcal{I} \to \mathbb{R}$ be a smooth function and $\mathbf{x} : \mathcal{I} \to \mathbb{R}^n$ be a trajectory. The time derivative of f along the trajectory $\mathbf{x}$ at a time instant t is calculated (using the chain rule) as:

$$\begin{aligned} f_{\mathbf{x}}(t) &:= f(\mathbf{x}(t), t) \\ f'_{\mathbf{x}}(t) &= \nabla f(\mathbf{x}(t), t) \cdot \frac{d\mathbf{x}}{dt}(t) + \frac{\partial f}{\partial t}(\mathbf{x}(t), t) \\ &= \left(\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f \right)(\mathbf{x}(t), t) \end{aligned}$$

From these observations, we can define a time differentiation operator which acts along particles' trajectories.

Definition 5 (material derivative and derivation along trajectories)**.** *Defining the* material derivative *operator*[9] *as* $\frac{D}{Dt} := \frac{\partial}{\partial t} + (\mathbf{u} \cdot \nabla)$*, the* derivative of f along the trajectory *passing through point* $\mathbf{x}$ *at time* t *is given by* $\frac{Df}{Dt}(\mathbf{x}, t)$*.*

With the material derivative, we are able to calculate time derivatives of functions along *material curves* (trajectories). As an example, from the velocity vector field $\mathbf{u}(\mathbf{x}, t)$, we can compute the fluid *acceleration (vector) field* $\mathbf{a}(\mathbf{x}, t)$ as simply as:[10]

$$\mathbf{a}(\mathbf{x}, t) = \frac{D\mathbf{u}}{Dt}(\mathbf{x}, t) = \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u}\right)(\mathbf{x}, t)$$

where the material derivative is applied componentwise (which also allows us to apply it on tensor fields). Note the connection between Lagrangian and Eulerian descriptions of the acceleration field made possible through the material derivative, such a connection is recurrent both in theory and algorithm development.

By now, we are able to demonstrate one of our most useful results.

Theorem 1 (Reynolds' Transport Theorem)**.**
Let $f : \overline{\mathcal{D}} \times \mathcal{I} \to \mathbb{R}$ *as in the smoothness convention and* $\Omega_t := \varphi(\Omega_0, t) \subseteq \mathcal{D}$*, for each* $t \in \mathcal{I}$ *and* $\Omega_0 \subseteq \mathcal{D}$ *the (arbitrary) reference fluid region, then*

$$\frac{d}{dt}\int_{\Omega_t} f\, dV = \int_{\Omega_t} \left(\frac{Df}{Dt} + f \operatorname{div} \mathbf{u}\right) dV$$

where dV *denotes the volume element.*

Before providing a proof for this result, we demonstrate the following lemma concerning the time derivative of $J(\mathbf{x}_0, t) := \det(\nabla \varphi(\mathbf{x}_0, t))$ (where the Jacobian of φ is taken with respect to spatial coordinates).

[9] Also known as *transport derivative*, *substantial derivative*, *convective derivate*...

[10] From this, we see that, given the velocity field $\mathbf{u}$, the acceleration field $\mathbf{a}$ is **not** given by $\frac{\partial \mathbf{u}}{\partial t}$ (a *very* common mistake).

Lemma 1. $\frac{\partial}{\partial t} J(\mathbf{x}_0, t) = J(\mathbf{x}_0, t) [\operatorname{div} \mathbf{u}(\varphi(\mathbf{x}_0, t), t)]$

Proof of Lemma 1. First of all, notice that

$$\begin{aligned}
\frac{\partial}{\partial t} \nabla \varphi_i(\mathbf{x}_0, t) &= \nabla \frac{\partial \varphi_i}{\partial t}(\mathbf{x}_0, t) \\
&= \nabla [\mathbf{u}_i(\varphi(\mathbf{x}_0, t), t)] \\
&= \nabla \mathbf{u}_i(\varphi(\mathbf{x}_0, t), t) \cdot \nabla \varphi(\mathbf{x}_0, t) \\
&= \sum_{j=1}^{n} \frac{\partial \mathbf{u}_i}{\partial \mathbf{x}_j}(\varphi(\mathbf{x}_0, t), t) \nabla \varphi_j(\mathbf{x}_0, t)
\end{aligned}$$

From this equality, the n-linearity and the antisymmetry of $\det(\cdot)$, we have

$$\begin{aligned}
\frac{\partial}{\partial t} J(\mathbf{x}_0, t) &= \frac{\partial}{\partial t} \det(\nabla \varphi(\mathbf{x}_0, t)) \\
&= \frac{\partial}{\partial t} \det(\nabla \varphi_1(\mathbf{x}_0, t), \ldots, \nabla \varphi_n(\mathbf{x}_0, t)) \\
&= \sum_{i=1}^{n} \det\left(\nabla \varphi_1(\mathbf{x}_0, t), \ldots, \frac{\partial}{\partial t} \nabla \varphi_i(\mathbf{x}_0, t), \ldots, \nabla \varphi_n(\mathbf{x}_0, t)\right) \\
&= \sum_{i=1}^{n} \det\left(\ldots, \sum_{j=1}^{n} \frac{\partial \mathbf{u}_i}{\partial \mathbf{x}_j}(\varphi(\mathbf{x}_0, t), t) \nabla \varphi_j(\mathbf{x}_0, t), \ldots\right) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial \mathbf{u}_i}{\partial \mathbf{x}_j}(\varphi(\mathbf{x}_0, t), t) \det\left(\ldots, \nabla \varphi_j(\mathbf{x}_0, t), \ldots\right) \\
&= \sum_{i=1}^{n} \frac{\partial \mathbf{u}_i}{\partial \mathbf{x}_i}(\varphi(\mathbf{x}_0, t), t) \det(\nabla \varphi_1(\mathbf{x}_0, t), \ldots, \nabla \varphi_n(\mathbf{x}_0, t)) \\
&= \det(\nabla \varphi(\mathbf{x}_0, t)) \sum_{i=1}^{n} \frac{\partial \mathbf{u}_i}{\partial \mathbf{x}_i}(\varphi(\mathbf{x}_0, t), t) \\
&= J(\mathbf{x}_0, t) [\operatorname{div} \mathbf{u}(\varphi(\mathbf{x}_0, t), t)]
\end{aligned}$$

□

Proof of Reynolds' Transport Theorem. From the smoothness of $\varphi_t\big|_{\Omega_0}$ (and its inverse's), the continuity of $\varphi(\mathbf{x}_0, \cdot)$ and that $\varphi(\cdot, t_0) \equiv \mathrm{id}(\cdot)$, we have the positivity of $J(\mathbf{x}_0, t)$ for each $\mathbf{x}_0 \in \Omega_0$ and each $t \in \mathcal{I}$. From this, the preceding lemma and the *Change of Variables Theorem*:

$$\begin{aligned}
\frac{d}{dt}\int\limits_{\Omega_t} f(\mathbf{x}, t)\, dV &= \frac{d}{dt}\int\limits_{\varphi(\Omega_0, t)} f(\mathbf{x}, t)\, dV \\
&= \frac{d}{dt}\int\limits_{\Omega_0} f(\varphi(\mathbf{x}_0, t), t)\, J(\mathbf{x}_0, t)\, dV \\
&= \int\limits_{\Omega_0} \left(\frac{\partial f}{\partial t} + \nabla f \cdot \frac{\partial \varphi}{\partial t}\right)(\varphi(\mathbf{x}_0, t), t)\, J(\mathbf{x}_0, t)\, dV \\
&\quad + \int\limits_{\Omega_0} f(\varphi(\mathbf{x}_0, t), t)\, \frac{\partial}{\partial t} J(\mathbf{x}_0, t)\, dV \\
&= \int\limits_{\Omega_0} \left(\frac{\partial f}{\partial t} + \nabla f \cdot \mathbf{u}\right)(\varphi(\mathbf{x}_0, t), t)\, J(\mathbf{x}_0, t)\, dV \\
&\quad + \int\limits_{\Omega_0} f(\varphi(\mathbf{x}_0, t), t)\, J(\mathbf{x}_0, t)\, [\operatorname{div} \mathbf{u}(\varphi(\mathbf{x}_0, t), t)]\, dV \\
&= \int\limits_{\Omega_0} \frac{Df}{Dt}(\varphi(\mathbf{x}_0, t), t)\, J(\mathbf{x}_0, t)\, dV \\
&\quad + \int\limits_{\Omega_0} (f \operatorname{div} \mathbf{u})(\varphi(\mathbf{x}_0, t), t)\, J(\mathbf{x}_0, t)\, dV \\
&= \int\limits_{\Omega_0} \left(\frac{Df}{Dt} + f \operatorname{div} \mathbf{u}\right)(\varphi(\mathbf{x}_0, t), t)\, J(\mathbf{x}_0, t)\, dV \\
&= \int\limits_{\varphi(\Omega_0, t)} \left(\frac{Df}{Dt} + f \operatorname{div} \mathbf{u}\right)(\mathbf{x}, t)\, dV \\
&= \int\limits_{\Omega_t} \left(\frac{Df}{Dt} + f \operatorname{div} \mathbf{u}\right)(\mathbf{x}, t)\, dV
\end{aligned}$$

□

The following corollary of the transport theorem establishes a connection between the Lagrangian and the Eulerian viewpoints (in an integral form).

Corollary 1. *With Ω_t and f as in the transport theorem, let Ω_1 be the* fixed *set in $\mathbb{R}^n$ which coincides with Ω_t at $t = t_1$. Then at the (arbitrary) time t_1,*

$$\frac{d}{dt}\int\limits_{\Omega_t} f\,dV = \frac{\partial}{\partial t}\int\limits_{\Omega_1} f\,dV + \int\limits_{\partial\Omega_1} f\mathbf{u}\cdot\mathbf{n}\,dA$$

where $\mathbf{n}$ is the unit outward normal, and dA the surface element, on $\partial\Omega_1$.

Proof. From the identity $\operatorname{div}(f\mathbf{u}) = \nabla f\cdot\mathbf{u} + f\operatorname{div}\mathbf{u}$, it suffices to note that $\frac{Df}{Dt} + f\operatorname{div}\mathbf{u} = \frac{\partial f}{\partial t} + \operatorname{div}(f\mathbf{u})$. From the transport and divergence theorems,

$$\begin{aligned}\frac{d}{dt}\int\limits_{\Omega_t} f\,dV &= \int\limits_{\Omega_1}\left(\frac{Df}{Dt} + f\operatorname{div}u\right)dV = \int\limits_{\Omega_1}\left(\frac{\partial f}{\partial t} + \operatorname{div}(f\mathbf{u})\right)dV\\ &= \int\limits_{\Omega_1}\frac{\partial f}{\partial t}\,dV + \int\limits_{\Omega_1}\operatorname{div}(f\mathbf{u})\,dV = \frac{\partial}{\partial t}\int\limits_{\Omega_1} f\,dV + \int\limits_{\partial\Omega_1} f\mathbf{u}\cdot\mathbf{n}\,dA\end{aligned}$$

□

The left-hand side of this expression denotes the rate of change of the "f-content" of the *fixed body of fluid* occupying the region $\Omega_1 \subseteq \mathcal{D}$ at time t_1. The first term on the right-hand side is the rate of change of the f-content of this *fixed spatial domain*. And the last term is the rate of outflow of f through the fixed boundary of Ω_1 ("flux of f through $\partial\Omega_1$") [Mey71].

3.3 Conservation of Mass and Incompressibility

In the previous section, we deduced some preliminary results based on our (still too abstract) mathematical model of fluid motion. From now on, we begin to describe our physical assumptions and conservation laws from section 3.1 in terms of mathematical properties of our model.

This section deals with the first of our basic principles, *the law of conservation of mass*, and its relationship with incompressibility in fluids. The assumption which ensures the conservation of mass of a fluid region in motion can be mathematically stated as:

Definition 6 (mass and density)**.** *There exists a function $\rho(\mathbf{x},t)$ such that, for any fluid region $\mathcal{W}\subseteq\Omega_0$ and every $t\in\mathcal{I}$*

$$0<m(\mathcal{W},t)=\int\limits_{\varphi(\mathcal{W},t)}\rho(\mathbf{x},t)\,dV\qquad\text{and}\qquad\frac{d}{dt}m(\mathcal{W},t)=0$$

where $m(\mathcal{W},t)$ is the mass *of the fluid in $\varphi(\mathcal{W},t)$ and the function $\rho(\mathbf{x},t)$ defined this way is called* mass density*.* [11]

From this definition, the conservation of mass can be stated in terms of ρ and $\mathbf{u}$ as an application of the *Reynolds' Transport Theorem*:

$$0=\frac{d}{dt}m(\mathcal{W},t)=\frac{d}{dt}\int\limits_{\varphi(\mathcal{W},t)}\rho(\mathbf{x},t)\,dV=\int\limits_{\varphi(\mathcal{W},t)}\left(\frac{D\rho}{Dt}+\rho\operatorname{div}\mathbf{u}\right)(\mathbf{x},t)\,dV$$

This way, from the smoothness properties of ρ, we have the equivalence

$$\boxed{\int\limits_{\varphi(\mathcal{W},t)}\left(\frac{D\rho}{Dt}+\rho\operatorname{div}\mathbf{u}\right)dV=0\Longleftrightarrow\frac{D\rho}{Dt}+\rho\operatorname{div}\mathbf{u}=0}\tag{3.1}$$

in which the left-hand side is called *the integral form of the law of conservation of mass*, while the right-hand side is the so called *differential form of the law of conservation of mass* (or the *continuity equation*).

From these forms of mass conservation, and applying once more the transport theorem, we can prove the following result.

Corollary 2. *With Ω_t and f as in the Reynolds' Transport Theorem,*

$$\frac{d}{dt}\int\limits_{\Omega_t}\rho f\,dV=\int\limits_{\Omega_t}\rho\frac{Df}{Dt}\,dV$$

Proof.

$$\begin{aligned}\frac{d}{dt}\int\limits_{\Omega_t}\rho f\,dV&=\int\limits_{\Omega_t}\left(\frac{D}{Dt}(\rho f)+(\rho f)\operatorname{div}\mathbf{u}\right)dV\\&=\int\limits_{\Omega_t}\left(\rho\frac{Df}{Dt}+f\left(\frac{D\rho}{Dt}+\rho\operatorname{div}\mathbf{u}\right)\right)dV=\int\limits_{\Omega_t}\rho\frac{Df}{Dt}\,dV\end{aligned}$$

□

[11] Sometimes we may refer to ρ as *density* or *specific mass*. As pointed in section 3.1, we adopt the smoothness convention for ρ, this way $\rho>0$ *almost everywhere*.

Although real fluids change volume, most fluid flows of "interest" can be approximated, to a high degree of accuracy, by *incompressible* flows. This means that, for many practical applications (e.g. the simulation of "everyday" fluids for computer animation), we can assume the incompressibility of our fluid flows.[12]

Definition 7 (incompressible flow)**.** *We say that φ denotes an* incompressible flow *when, for any fluid region $\mathcal{W} \subseteq \Omega_0$ and every $t \in \mathcal{I}$*

$$\text{volume}\left(\mathcal{W}\right) = \text{volume}\left(\varphi\left(\mathcal{W}, t\right)\right) \Leftrightarrow \int\limits_{\mathcal{W}} dV = \int\limits_{\varphi(\mathcal{W},t)} dV \Leftrightarrow \frac{d}{dt} \int\limits_{\varphi(\mathcal{W},t)} dV = 0$$

From this definition and the transport theorem, for incompressible flows,

$$0 = \frac{d}{dt} \int\limits_{\varphi(\mathcal{W},t)} dV = \int\limits_{\varphi(\mathcal{W},t)} \operatorname{div} \mathbf{u}\, dV \quad \Longleftrightarrow \quad \operatorname{div} \mathbf{u} = 0$$

where the right-hand side is often known as the *incompressibility condition.*[13]

The incompressibility condition and lemma 1 also result that *a flow is incompressible if and only if $J \equiv 1$*, since $J\left(\cdot, t_0\right) = 1$.

From the equation of continuity 3.1, and the fact that $\rho > 0$, *a fluid is incompressible if and only if $\frac{D\rho}{Dt} = 0$, that is, the mass density is constant following the fluid.* If the fluid is *homogeneous* (i.e. $\rho =$ constant in space), it also follows that the flow is incompressible if and only if ρ is constant in time. So, for a homogeneous incompressible fluid, $\rho\left(\mathbf{x}, t\right) = \rho_0 > 0$.[14]

[12] Notice this doesn't hold for simulations involving high-speed flows like *sonic booms* and *blast waves*. The study of how fluids behave in these situations is generally called *"compressible flow"*. It's usually so much more complicated and expensive to simulate these high-speed flows that, even if we have sonic booms and blast waves, they're invisible and most audiences have no idea really how they behave, so it's generally a much better idea (for computer graphics applications) to *hack* together something that looks cool than try to simulate them accurately. [BMF07]

[13] A velocity field $\operatorname{div} \mathbf{u} = 0$ is often called *divergenceless* or *divergence-free*.

[14] Notice that this does **not** mean that incompressibility $\Longrightarrow$ homogeneity.

3.4 Balance of Momentum

In this section we incorporate in our model the second of our basic principles, the *Newton's laws of motion*. From these, the second is the one mathematically translated into properties of our model, while both others impose simplifying assumptions in our derivations (e.g. symmetry in binary forces due to the third law).

To ensure the law of balance of momentum (Newton's second law of motion), we first need to define it in the context of continuum fluid mechanics.

Definition 8 (linear momentum)**.** *The* (linear) momentum *of a fluid region* Ω_t *is given by the integral*[15]

$$\int_{\Omega_t} \rho\left(\mathbf{x}, t\right) \mathbf{u}\left(\mathbf{x}, t\right)\, dV.$$

With this definition of momentum, Newton's second law can be stated as

$$\{\text{net force on } \Omega_t\} = \frac{d}{dt}\int_{\Omega_t} \rho\mathbf{u}\, dV = \int_{\Omega_t} \rho\frac{D\mathbf{u}}{Dt}\, dV,$$

where the second equality holds from corollary 2.[16] The net force on Ω_t is due to contributions from *external forces* (also known as *body* or *long-range* forces) and *internal forces* (also known as surface *stress* forces).

The canonical example of external force is the *gravity* field. These forces act *per unit volume* on the fluid, so they can be modeled as the product $\rho\mathbf{b} : \Omega_t \times \mathcal{I} \to \mathbb{R}^n$ (where $\mathbf{b} : \Omega_t \times \mathcal{I} \to \mathbb{R}^n$ is a vector field denoting the external force field *per unit mass* and, abusing nomenclature, will be called **the** *body force field*).

An example of surface stress force is the *hydrostatic pressure* force one feels when diving, this force is due to the weight of a fluid and increases with the depth (evidencing the influence of the rest of fluid). These forces are modeled as a field $\tau : \overline{\Omega_t} \times \mathcal{I} \times \mathbb{S}^{n-1} \to \mathbb{R}^n$, which depends on the position in the fluid region's boundary $\mathbf{x}$, time instant t and the (outward) normal $\mathbf{n}$ at the point $\mathbf{x}$ on the boundary $\partial\Omega_t$.

[15] Intuitively, notice that $\rho\mathbf{u}\, dV = (\rho\, dV)\mathbf{u}$ denotes the linear momentum (of an "infinitesimal fluid element") as defined by Newton's second law ($m\vec{v}$).

[16] Again, intuitively, the quantity $\rho\frac{D\mathbf{u}}{Dt}\, dV = (\rho\, dV)\frac{D\mathbf{u}}{Dt}$ denotes the net force exerted on an "infinitesimal fluid element" (from Newton's second law, $\vec{F} = m\vec{a}$).

This way, the net force on Ω_t is given by

$$\begin{aligned}\{\text{net force on } \Omega_t\} &= \{\text{surface stress forces}\} + \{\text{body forces}\} \\ &= \int\limits_{\partial\Omega_t} \tau\left(\mathbf{x}, t, \mathbf{n}\right)\, dA + \int\limits_{\Omega_t} \rho\left(\mathbf{x}, t\right) \mathbf{b}\left(\mathbf{x}, t\right)\, dV\end{aligned}$$

which can be further simplified using a theorem of Cauchy ensuring, as a consequence of a continuous dependence of τ on $\mathbf{n}$ and the balances of linear and angular momenta, the existence of a tensor field $\mathbf{T} : \overline{\Omega_t} \times \mathcal{I} \to \mathbb{R}^{n\times n}$ such that $\tau\left(\mathbf{x}, t, \mathbf{n}\right) = \mathbf{T}\left(\mathbf{x}, t\right) \cdot \mathbf{n}$ and $\mathbf{T}\left(\mathbf{x}, t\right)$ is symmetric $\left(\forall\left(\mathbf{x}, t\right) \in \overline{\Omega_t} \times \mathcal{I}\right)$.[17]

From this result, and the *Divergence Theorem* for tensor fields,

$$\boxed{\int\limits_{\Omega_t} \rho \frac{D\mathbf{u}}{Dt}\, dV = \int\limits_{\partial\Omega_t} \mathbf{T} \cdot \mathbf{n}\, dA + \int\limits_{\Omega_t} \rho\mathbf{b}\, dV = \int\limits_{\Omega_t} \left(\operatorname{Div}\mathbf{T} + \rho\mathbf{b}\right)\, dV} \tag{3.2}$$

where $\operatorname{Div}\mathbf{T}$ is the vector in which each component corresponds to the divergence of the corresponding row in $\mathbf{T}$ (seen as a vector field). This equality ensures the balance of momentum in our model and is known as *the integral form of the law of balance of momentum*. As we did for mass conservation, assuming regularity of the fields involved, we get *the differential form of the law of balance of momentum*

$$\boxed{\rho \frac{D\mathbf{u}}{Dt} = \operatorname{Div}\mathbf{T} + \rho\mathbf{b}.} \tag{3.3}$$

Notice that no further assumptions on the structure of $\mathbf{T}$ have been made to arrive at these forms of balance of momentum. This proves useful when one desires to study the behavior of *Non-Newtonian fluids*, which are not dealt with in this text, because all we have done so far has been general enough to accomodate these materials as well. In the next two sections, we specialize our derivations assuming a certain structure for $\mathbf{T}$.

[17] Although we don't prove the *Cauchy's Theorem* [HM76, Pat87], the particular result $\tau\left(\mathbf{x}, t, -\mathbf{n}\right) = -\tau\left(\mathbf{x}, t, \mathbf{n}\right)$, necessary for the linearity of τ on $\mathbf{n}$, is a direct (and simple) consequence of Newton's third law of motion.[MN91]

3.5 Ideal Fluids and Euler's Equations

Our first simplifying assumption on the structure of the tensor field $\mathbf{T}$ is that it doesn't impose tangential surface forces, i.e. $\mathbf{T}(\mathbf{x}, t) \cdot \mathbf{n}$ is parallel to $\mathbf{n}$. From this assumption, $\mathbf{T}(\mathbf{x}, t)$ must be a multiple of the identity matrix

$$\mathbf{T} = -p\mathbf{I} \quad \Longrightarrow \quad \operatorname{Div} \mathbf{T} = -\nabla p$$

where $p : \overline{\Omega_t} \times \mathcal{I} \to \mathbb{R}$ is called *(mechanical) pressure.*[18]

This assumption of inexistence of tangential forces on a fluid domain's surface is an approximation which neglects effects due to viscous friction, that is the reason why fluid flows derived from it are called *inviscid* or also

Definition 9 (ideal fluids)**.** *A fluid flow is called* ideal *if the surface stress forces are given by a tension field such that* $\tau(\mathbf{x}, t, \mathbf{n}) = -p(\mathbf{x}, t)\,\mathbf{n}$.

From these considerations, 3.2 and 3.3, we deduce both the integral and differential forms of the law of balance of momentum for ideal fluid flows,

$$\int_{\Omega_t} \rho \frac{D\mathbf{u}}{Dt}\, dV = \int_{\Omega_t} (-\nabla p + \rho \mathbf{b})\, dV \quad \Longleftrightarrow \quad \rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \rho \mathbf{b}$$

Together with the forms of the law of conservation of mass, the equations above form a system of $n + 2$ unknowns ($\mathbf{u}$, ρ and p)[19] and $n + 1$ equations (n from balance of momentum and 1 from conservation of mass).

One way to "close" this system of equations is coupling p and ρ through an *equation of state*, which specifies the pressure as a function of density. In this way, we have $p(\mathbf{x}, t) = f(\rho(\mathbf{x}, t))$, where $f : \mathbb{R}_+ \to \mathbb{R}$ is smooth.[20]

Common examples of equations of state used in computer graphics are:

- $p(\rho) = c^2(\rho - \rho_0)$, where c is the *sound speed* in fluid (the maximum speed attainable by a propagating wave) and ρ_0 a reference density;

[18] The minus sign in $-p\mathbf{I}$ is an account to the intuitive concept that a positive pressure indicates a compression imposed on the fluid domain, i.e. tension is oriented in the opposite direction of the surface's normal field.

[19] Notice that we assume $\mathbf{b}$ is known.

[20] For a more in-depth discussion of the physical interpretation (and an implicit form) for *equations of state*, see [Bat99].

- $p(\rho) = p_0 \left[\left(\frac{\rho}{\rho_0} \right)^{\gamma} - 1 \right]$, where p_0 and ρ_0 are the reference pressure and density, respectively, and γ a constant (typically chosen as $\gamma = 7$).[21]

Another very common way to ensure that all degrees of freedom have been tackled is to assume both incompressibility and homogeneity of the fluid flow, this way we have the system of partial differential equations known as the *Euler's equations* (for incompressible homogeneous inviscid flows):

$$\boxed{\frac{D\mathbf{u}}{Dt} = -\nabla \tilde{p} + \mathbf{b}, \quad \operatorname{div} \mathbf{u} = 0, \quad \mathbf{u} \parallel \partial \mathcal{W}} \tag{3.4}$$

where $\tilde{p} := \frac{1}{\rho_0} p$ and the third equation is a boundary condition on the flow asserting that no fluid "traverses walls" but is "allowed" to freely slip along them without friction effects. That is the reason why this condition is sometimes refered to as *(free-)slip*, *no-stick* or *inviscid boundary condition*.[22]

In the next chapter, we will present two classes of numerical schemes used for the simulation of fluid flows in computer graphics, the first is based on the Eulerian description of fluid motion and addresses the degrees of freedom by assuming incompressibility and homogeneity of the fluid flows, while the second method employs the Lagrangian description and makes use of an equation of state to animate quasi-incompressible fluid flows.

3.6 Newtonian Viscous Fluids and the Navier-Stokes Equations

As noted in previous section, the assumption of ideal fluids neglects effects due to viscous friction. This assumption is reasonable for some gaseous flows but it is still too restrictive for many fluids (e.g. water, milk, beer, etc.).

In this section, we make different simplifying assumptions on the structure of the tensor field $\mathbf{T}$ in order to model some common *"everyday fluid flows"*.

[21] The first equation was proposed in [MFZ97] and is very popular in computer graphics; the second was adapted from $p(\rho) = (1 + B) \left(\frac{\rho}{\rho_0} \right)^n - B$, suggested as an equation of state for water by [Bat99], where n and B are chosen as 7 and $3,000$ respectively.

[22] Different boundary conditions can be imposed on the fluid domain, also often used in computer graphics are the *inflow/outflow*, *open boundary* and the *free-surface* conditions discussed elsewhere [FM96a, SCP+04, FF01, BMF07] and the *no-slip* boundary condition we discuss later.

Instead of assuming that $\mathbf{T}(\mathbf{x},t) = -p(\mathbf{x},t)\,\mathbf{I}$ as for ideal fluids, we now assume that $\mathbf{T}(\mathbf{x},t) = -p(\mathbf{x},t)\,\mathbf{I} + \boldsymbol{\sigma}(\mathbf{x},t)$, where $\boldsymbol{\sigma}$ is such that [CM93]:

1. $\boldsymbol{\sigma}$ *depends linearly on the velocity gradients* $\nabla\mathbf{u}$, i.e. $\boldsymbol{\sigma}$ is related to $\nabla\mathbf{u}$ by some linear transformation at each point.[23]

2. $\boldsymbol{\sigma}$ *is invariant under rigid body rotations*, i.e.

$$\mathbf{U} \in SO(n) \Longrightarrow \boldsymbol{\sigma}\left(\mathbf{U}\cdot\nabla\mathbf{u}\cdot\mathbf{U}^{-1}\right) = \mathbf{U}\cdot\boldsymbol{\sigma}(\nabla\mathbf{u})\cdot\mathbf{U}^{-1}$$

   This means that, under rigid body rotations, there is no diffusion of momentum due to viscosity.

3. $\boldsymbol{\sigma}$ *is symmetric.* This can be deduced from the symmetry of $\mathbf{T}$ (due to balance of angular momentum [MH94]).

From these assumptions, the *Newtonian fluids*' assumptions, follows [CM93]

$$\boldsymbol{\sigma} = \lambda\,\mathrm{tr}(\mathbf{D})\,\mathbf{I} + 2\mu\mathbf{D}$$

where $\lambda(\mathbf{x},t)$ and $\mu(\mathbf{x},t)$ are the non-negative *coefficients of viscosity* and $\mathbf{D} := \frac{1}{2}\left[\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right]$ is the symmetric part of $\nabla\mathbf{u}$.

Since $\mathrm{tr}(\mathbf{D}) = \mathrm{div}\,\mathbf{u}$, we have $\mathbf{T} = -p\mathbf{I} + \lambda(\mathrm{div}\,\mathbf{u})\,\mathbf{I} + \mu\left[\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right]$. From these considerations, the *differential form of the law of balance of momentum for Newtonian viscous flows* follows:[24]

$$\boxed{\rho\frac{D\mathbf{u}}{Dt} = -\nabla p + \nabla\left[\lambda(\mathrm{div}\,\mathbf{u})\right] + \mathrm{Div}\left(\mu\left[\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right]\right) + \rho\mathbf{b}} \tag{3.5}$$

A common assumption is to consider $\lambda(\mathbf{x},t) = \lambda_0$ and $\mu(\mathbf{x},t) = \mu_0$, yielding[25]

$$\boxed{\rho\frac{D\mathbf{u}}{Dt} = -\nabla p + (\lambda+\mu)\,\nabla(\mathrm{div}\,\mathbf{u}) + \mu\Delta\mathbf{u} + \rho\mathbf{b}} \tag{3.6}$$

[23] This means $\boldsymbol{\sigma}(\mathbf{x},t) = f(\nabla\mathbf{u}(\mathbf{x},t))$, for some $f : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ linear. Sometimes, we use the notation $\boldsymbol{\sigma}(\nabla\mathbf{u})$ instead of $\boldsymbol{\sigma}(\mathbf{x},t)$ to make explicit the dependence on $\nabla\mathbf{u}$.

[24] We omitted the integral form for space reasons.

[25] Notice the use of the laplacian $\Delta := \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$, which is applied componentwise in a vector field $\mathbf{u}$ yielding another vector field $\Delta\mathbf{u}$, and the identity $\mathrm{Div}\left[(\nabla\mathbf{u})^T\right] = \nabla(\mathrm{div}\,\mathbf{u})$.

Which, adding the incompressibility assumption, amounts in the most commonly used form for the law of balance of momentum in computer graphics:

$$\boxed{\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \mu \Delta \mathbf{u} + \rho \mathbf{b}} \tag{3.7}$$

Notice that, if we assumed incompressibility *before* the homogeneity of λ and μ, we would have no dependence on λ because

$$\boxed{\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \operatorname{Div}\left(\mu \left[\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right]\right) + \rho \mathbf{b},}$$

which is a form used to model varying viscosity incompressible fluid flows.[26]

The assumptions of homogeneity (of both ρ and μ) and incompressibility yield, along with the differential form of the law of conservation of mass, results in the most common system of PDE's for the modeling of fluid motion, the *Navier-Stokes equations* (for incompressible homogeneous Newtonian viscous flows):[27]

$$\boxed{\frac{D\mathbf{u}}{Dt} = -\nabla \tilde{p} + \nu \Delta \mathbf{u} + \mathbf{b}, \quad \operatorname{div} \mathbf{u} = 0, \quad \mathbf{u}|_{\partial \mathcal{W}} = 0} \tag{3.8}$$

where $\tilde{p} := \frac{1}{\rho} p$, $\nu := \frac{1}{\rho} \mu$ is known as the *kinematic viscosity* and the third equation denotes a boundary condition on the flow asserting that the flow "adheres to walls", due to effects of viscous friction. For this reason, this condition is refered to as the *no-slip* or *viscous boundary condition*.

In the next chapter, when we discuss the Lagrangian scheme known as *Smoothed Particle Hydrodynamics*, it is noticed the omission of one term from the differential form of the law of balance of momentum in the PDE's used by a popular paper in the computer graphics community. Such omission is due to an assumption of flow incompressibility, which cannot be ensured by their implementation of SPH (not designed for really incompressible flows).

[26] This one should be the form of the momentum equation used in [CMIT02], however it seems they forgot the second parcel appearing in the diffusive term.

[27] Notice that the expression modeling the law of balance of momentum is often called Navier-Stokes equation. In this text, we use the name *Navier-Stokes equation(s)* interchangeably to mean the systems of PDE's coupling one of the differential forms for the law of balance of momentum derived in this section, the differential form of the law of conservation of mass and the *no-slip boundary condition* we comment later (and, possibly, an equation of state, if the incompressibility assumption is not made).

3.7 Pressure and Incompressibility

In section 3.5, we introduced the mechanical pressure p as the magnitude of contact forces acting on a fluid surface in a direction parallel to the normal $\mathbf{n}$. Also, we asked fluid incompressibility and homogeneity to determine p.

We now shall explore the role of the pressure in incompressible flow. With this aim, we exploit the following vector field decomposition theorem.

Theorem 2 (Helmholtz-Hodge Decomposition Theorem)**.** *Let $\mathcal{W} \subseteq \mathbb{R}^n$ be a fluid region and $\mathbf{u} : \overline{\mathcal{W}} \to \mathbb{R}^n$ a smooth vector field. There exists a unique decomposition of $\mathbf{u}$ such that, for smooth $\mathbf{u}_s : \overline{\mathcal{W}} \to \mathbb{R}^n$ and $\mathbf{u}_p : \overline{\mathcal{W}} \to \mathbb{R}^n$,*

$$\mathbf{u} = \mathbf{u}_s + \mathbf{u}_p, \quad \nabla \cdot \mathbf{u}_s \equiv 0 \text{ on } \mathcal{W}, \quad \mathbf{u}_s \cdot \mathbf{n} \equiv 0 \text{ on } \partial\mathcal{W}, \quad \mathbf{u}_p \equiv \nabla\phi \text{ on } \mathcal{W}$$

for some smooth $\phi : \overline{\mathcal{W}} \to \mathbb{R}$.

Proof. The existence of such a decomposition is due to the existence of a function $\phi : \overline{\mathcal{W}} \to \mathbb{R}$ such that, for smooth $f : \mathcal{W} \to \mathbb{R}$ and $g : \partial\mathcal{W} \to \mathbb{R}$,

$$-\Delta\phi = f \text{ on } \mathcal{W}, \quad \frac{d\phi}{d\mathbf{n}} = g \text{ on } \partial\mathcal{W} \quad \Longleftrightarrow \quad \int_{\mathcal{W}} f \, dV = -\int_{\partial\mathcal{W}} g \, dA.$$

Taking $f := -\nabla \cdot \mathbf{u}$ and $g := \mathbf{u} \cdot \mathbf{n}$, the divergence theorem ensures the right-hand side of the above equivalence, which guarantees a ϕ as before. Defining $\mathbf{u}_p := \nabla\phi$ and $\mathbf{u}_s := \mathbf{u} - \mathbf{u}_p$, it is easy to verify the stated properties.

To prove uniqueness, we suppose that $\mathbf{u} = \mathbf{u}_s^a + \mathbf{u}_p^a = \mathbf{u}_s^b + \mathbf{u}_p^2$, then

$$0 = (\mathbf{u}_s^a - \mathbf{u}_s^b) + (\mathbf{u}_p^a - \mathbf{u}_p^b) = (\mathbf{u}_s^a - \mathbf{u}_s^b) + \nabla(\phi^a - \phi^b).$$

Taking the L^2 inner product with $(\mathbf{u}_s^a - \mathbf{u}_s^b)$, we have that

$$\begin{aligned} 0 &= \int_{\mathcal{W}} \left\{ \|\mathbf{u}_s^a - \mathbf{u}_s^b\|_2^2 + (\mathbf{u}_s^a - \mathbf{u}_s^b) \cdot \nabla(\phi^a - \phi^b) \right\} dV \\ &= \int_{\mathcal{W}} \|\mathbf{u}_s^a - \mathbf{u}_s^b\|_2^2 \, dV + \int_{\mathcal{W}} \nabla \cdot \left[(\phi^a - \phi^b)(\mathbf{u}_s^a - \mathbf{u}_s^b) \right] dV \\ &= \int_{\mathcal{W}} \|\mathbf{u}_s^a - \mathbf{u}_s^b\|_2^2 \, dV + \int_{\partial\mathcal{W}} (\phi^a - \phi^b) \left[(\mathbf{u}_s^a - \mathbf{u}_s^b) \cdot \mathbf{n} \right] dA = \int_{\mathcal{W}} \|\mathbf{u}_s^a - \mathbf{u}_s^b\|_2^2 \, dV \end{aligned}$$

where the second equality is due to $\nabla \cdot \mathbf{u}_s^a = \nabla \cdot \mathbf{u}_s^b = 0$ on $\mathcal{W}$, the third to the divergence theorem and the final is due to $\mathbf{u}_s^a \cdot \mathbf{n} = \mathbf{u}_s^b \cdot \mathbf{n} = 0$ on $\partial\mathcal{W}$. This implies that $\|\mathbf{u}_s^a - \mathbf{u}_s^b\|_{L^2} = 0$, hence we must have $\mathbf{u}_s^a = \mathbf{u}_s^b$ and $\mathbf{u}_p^a = \mathbf{u}_p^b$. □

The above proof is specially important because it provides a method to perform the Helmholtz-Hodge decomposition of a vector field through the resolution of a *Poisson problem* with *Neumann boundary conditions*. This method will be useful later to the design of numerical simulation algorithms.

The field $\mathbf{u}_s$ is called the *divergence-free* (or *divergenceless*) component of $\mathbf{u}$, while $\mathbf{u}_p$ is known as its *potential* (or *gradient*) part. With this result, we are able to define an orthogonal projection operator which maps a given vector field onto its *divergence-free* (*incompressible*) component.

Definition 10 (projection operator)**.** *The* projection operator $\mathbb{P}$ *acts on vector fields for which the* Helmholtz-Hodge Decomposition Theorem *holds and is defined as* $\mathbb{P}\,\mathbf{u} := \mathbf{u}_s$*, where* $\mathbf{u}_s$ *is the divergence-free component of* $\mathbf{u}$*.*

The importance of this operator may be better appreciated through its action on both Euler's and the Navier-Stokes equations for homogeneous incompressible flows (3.4) and (3.8), which take, respectively, the forms:

$$\boxed{\mathbf{u}_t = \mathbb{P}\Big(-\mathbf{u}\cdot\nabla\mathbf{u} + \mathbf{b}\Big), \quad \operatorname{div}\mathbf{u} = 0, \quad \mathbf{u} \parallel \partial\mathcal{W}}$$

$$\boxed{\mathbf{u}_t = \mathbb{P}\Big(-\mathbf{u}\cdot\nabla\mathbf{u} + \nu\Delta\mathbf{u} + \mathbf{b}\Big), \quad \operatorname{div}\mathbf{u} = 0, \quad \mathbf{u}|_{\partial\mathcal{W}} = 0}$$

These equations shed some light on the role of pressure in incompressible fluid flows. From them, one can think about pressure as whatever it takes to keep flow incompressibility [BMF07].[28] Other interesting implication of the above equations is the "blindness" of the fluid flow to conservative body forces $\big($i.e. $\exists U : \overline{W} \to \mathbb{R}$ such that $\mathbf{b} = -\nabla U\big)$, since this implies $\mathbb{P}\big(\mathbf{b}\big) \equiv 0$.

Such forms of the governing equations not only allow us to give an interpretation for the role of pressure in incompressible fluid flows, but also they are at the heart of some of the most commonly used numerical schemes for the animation of fluid flows in the computer graphics community. The exploitation of the projection operator to design a numerical scheme based on the Eulerian description of fluid motion is discussed in the next chapter.

[28] Such a loose interpretation for pressure is often exploited by fluid modelers to enforce some properties at the fluid's boundary (e.g. *the pressure jump in two-phase fluids* [HK05], *the coupling of fluids and deformable bodies* [GSLF05, CGFO06]).

Another way of thinking about pressure comes from interpreting it as a *Lagrange multiplier* of a fluid dynamics constrained by the incompressibility condition (i.e. the *equality constraint* $\operatorname{div}\mathbf{u} \equiv 0$) [BMF07].

3.8 Rotation and Vorticity

In this section, we study the rotational motion of fluids. Such a study is of foremost importance due to the key influence of turbulent and rotational behavior of fluids in the visual perception of fluid motion. To this end, we introduce the concept of *vorticity* as closely related to the local angular velocity of the flow.

Definition 11 (vorticity)**.** *The* vorticity field $\boldsymbol{\omega} : \overline{\mathcal{D}} \times \mathcal{I} \to \mathbb{R}^n$ *of a fluid flow is defined as the curl of the flow's velocity field, i.e.* $\boldsymbol{\omega}(\mathbf{x}, t) := \nabla \times \mathbf{u}(\mathbf{x}, t)$.

As the curl of the velocity field $\mathbf{u}$, the vorticity quantifies the local *spinning* speed and direction induced by the fluid flow map. Such an interpretation can also be supported by Batchelor's definition of vorticity as *twice the local angular velocity* [Bat99].

An interesting result regarding the local structure of a vector field states,

Proposition 1. *Consider a stationary flow*[29] *around a small neighborhood of a given point* $\mathbf{x}_0 \in \mathcal{D}$*. Let* $\mathbf{u}_0 := \mathbf{u}(\mathbf{x}_0)$, $\mathbf{D}_0 := \frac{1}{2}\Big(\nabla\mathbf{u}(\mathbf{x}_0) + \big(\nabla\mathbf{u}(\mathbf{x}_0)\big)^T\Big)$ *and* $\boldsymbol{\omega}_0 := \boldsymbol{\omega}(\mathbf{x}_0)$*, then, for sufficiently small* $\|\mathbf{h}\|$*,*[30]

$$\mathbf{u}(\mathbf{x}_0 + \mathbf{h}) \approx \mathbf{u}_0 + \mathbf{D}_0 \cdot \mathbf{h} + \frac{1}{2}\boldsymbol{\omega}_0 \times \mathbf{h}$$

Proof. This proposition follows imediately from *Taylor's formula* and the fact that $\boldsymbol{\omega}(\mathbf{x})$ equals twice the anti-symmetric part of $\nabla\mathbf{u}(\mathbf{x})$ (property readly verifiable by simple hand calculation). □

This proposition can be physically interpreted as: *a fluid flow's velocity field induces, locally and approximately, the superposition of translation, deformation and rotational effects.* From it, we can appreciate the role of vorticity in the rotational behavior of the fluid flow and a reason for Batchelor's definition of $\boldsymbol{\omega}$.

We now introduce the notion of *circulation*, which may be understood as the total amount of "spinning" around a closed contour.

[29] A fluid flow is said *stationary* if its velocity field is time-independent, i.e. $\frac{\partial \mathbf{u}}{\partial t} \equiv 0$. Notice this **doesn't** imply that $\mathbf{u} \equiv 0$.

[30] Somewhat more precisely stated: *for every* $\mathbf{h} \in \mathbb{R}^n$ *such that* $(\mathbf{x}_0 + \mathbf{h}) \in \mathcal{D}$*,*

$$\mathbf{u}(\mathbf{x}_0 + \mathbf{h}) = \mathbf{u}_0 + \mathbf{D}_0 \cdot \mathbf{h} + \boldsymbol{\omega}_0 \times \mathbf{h} + o\big(\|\mathbf{h}\|\big).$$

Definition 12 (circulation)**.** *Let $\mathcal{C}_0 \subset \Omega_0$ be a simple closed contour in the reference configuration Ω_0 at time $t = t_0$ and $\mathcal{C}_t := \varphi(\mathcal{C}, t)$ be the contour carried along by the flow. The* circulation $\Gamma_{\mathcal{C}_t}$ *around the curve $\mathcal{C}_t$ is defined to be the line integral $\Gamma_{\mathcal{C}_t} = \oint_{\mathcal{C}_t} \mathbf{u} \cdot d\mathbf{s}$, where $d\mathbf{s}$ is the* line element *on $\mathcal{C}_t$.*

This definition relates, through *Stokes's Theorem*, the circulation around a contour to the flux of vorticity across a surface bounded by this curve.

By now, we are ready to prove some theorems regarding properties of the vorticity field induced by a fluid flow and the circulation around advected contours. The first of those, is a lemma that resembles the transport theorem.

Lemma 2. $\frac{d}{dt} \oint_{\mathcal{C}_t} \mathbf{u} \cdot d\mathbf{s} = \oint_{\mathcal{C}_t} \frac{D\mathbf{u}}{Dt} \cdot d\mathbf{s}$

Proof. Let $\mathbf{x}(s)$ be a parametrization of the loop $\mathcal{C}_0$, $s \in [0,1]$. Then a parametrization of $\mathcal{C}_t$ is $\varphi\big(\mathbf{x}(s), t\big)$, $s \in [0,1]$. Thus, by definition of the line integral and the material derivative,

$$
\begin{aligned}
\frac{d}{dt} \oint_{\mathcal{C}_t} \mathbf{u} \cdot d\mathbf{s} &= \frac{d}{dt} \int_0^1 \mathbf{u}\big(\varphi\big(\mathbf{x}(s), t\big), t\big) \cdot \frac{\partial}{\partial s} \varphi\big(\mathbf{x}(s), t\big)\, ds \\
&= \int_0^1 \frac{D\mathbf{u}}{Dt}\big(\varphi\big(\mathbf{x}(s), t\big), t\big) \cdot \frac{\partial}{\partial s} \varphi\big(\mathbf{x}(s), t\big)\, ds \\
&+ \int_0^1 \mathbf{u}\big(\varphi\big(\mathbf{x}(s), t\big), t\big) \cdot \frac{\partial}{\partial t}\frac{\partial}{\partial s} \varphi\big(\mathbf{x}(s), t\big)\, ds \\
&= \oint_{\mathcal{C}_t} \frac{D\mathbf{u}}{Dt} \cdot d\mathbf{s} + \int_0^1 \mathbf{u}\big(\varphi\big(\mathbf{x}(s), t\big), t\big) \cdot \frac{\partial}{\partial s} \mathbf{u}\big(\varphi\big(\mathbf{x}(s), t\big), t\big)\, ds \\
&= \oint_{\mathcal{C}_t} \frac{D\mathbf{u}}{Dt} \cdot d\mathbf{s} + \frac{1}{2} \int_0^1 \frac{\partial}{\partial s}\big(\mathbf{u} \cdot \mathbf{u}\big)\big(\varphi\big(\mathbf{x}(s), t\big), t\big)\, ds \\
&= \oint_{\mathcal{C}_t} \frac{D\mathbf{u}}{Dt} \cdot d\mathbf{s}, \quad \text{since } \mathcal{C}_t \text{ is a simple closed contour.}
\end{aligned}
$$

□

With this result, we can prove that the circulation around any reducible closed curve that is advected by an ideal flow is an invariant quantity.

Theorem 3 (Kelvin's Circulation Theorem)**.** *For an inviscid incompressible homogeneous fluid flow subject to the action of conservative external forces,* $\Gamma_{\mathcal{C}_t}$ *is constant in time, i.e.* $\forall t \in \mathcal{I}$, $\Gamma_{\mathcal{C}_t} = \Gamma_{\mathcal{C}_0} \in \mathbb{R}$.

Proof. From $\frac{D\mathbf{u}}{Dt} = -\nabla\tilde{p} - \nabla U = -\nabla(\tilde{p} + U)$ and the preceding lemma,

$$\frac{d}{dt}\Gamma_{\mathcal{C}_t} = \frac{d}{dt}\oint_{\mathcal{C}_t} \mathbf{u} \cdot d\mathbf{s} = \oint_{\mathcal{C}_t} \frac{D\mathbf{u}}{Dt} \cdot d\mathbf{s} = -\oint_{\mathcal{C}_t} \nabla(\tilde{p} + U) \cdot d\mathbf{s} = 0$$

where the last equality follows from the fact that $\mathcal{C}_t$ is a closed contour. □

Such a result has a different interpretation regarding the flux of vorticity across an advected surface, which has recently been exploited to numerically simulate fluid motion for computer graphics applications [ETK+07].

Corollary 3. *The flux of vorticity across a surface moving with an inviscid incompressible homogeneous fluid under potential forces is constant in time.*

Proof. The flux of vorticity across the surface, by *Stokes' Theorem*, equals the circulation across its bounding curve. The corollary follows by applying *Kelvin's Circulation Theorem* to this curve. □

Another very useful result deals with the dynamics of the vorticity field.

Proposition 2. *Consider an incompressible homogeneous fluid flow under conservative body forces, then we have*

(i) in the inviscid case *(modeled by Euler's equations 3.4):*

$$\frac{D\boldsymbol{\omega}}{Dt} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u}$$

(ii) in the viscous case *(modeled by the Navier-Stokes equations 3.8):*

$$\frac{D\boldsymbol{\omega}}{Dt} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \nu\Delta\boldsymbol{\omega}$$

Proof. Both items can be verified by taking the curl of the respective momentum equation, using the vector identities $\frac{1}{2}\nabla(\mathbf{u}\cdot\mathbf{u}) = \mathbf{u}\times(\nabla\times\mathbf{u}) + (\mathbf{u}\cdot\nabla)\mathbf{u}$, $\Delta\mathbf{F} = \nabla(\nabla\cdot\mathbf{F}) - \nabla\times(\nabla\times\mathbf{F})$, $\nabla\times(\nabla f) = 0$, $\nabla\times(\mathbf{F}\times\mathbf{G}) = \mathbf{F}(\nabla\cdot\ \mathbf{G}) - \mathbf{G}(\nabla\cdot\ \mathbf{F}) + (\mathbf{G}\cdot\nabla)\mathbf{F} - (\mathbf{F}\cdot\nabla)\mathbf{G}$ and $\nabla\cdot(\nabla\times\mathbf{F}) = 0$, and from the incompressibility condition.[31] □

[31] The identity for $\Delta\mathbf{F}$ is used only in the viscous case (for $\mathbf{F} = \mathbf{u}$ and again for $\mathbf{F} = \boldsymbol{\omega}$). A lot of such useful vector calculus' identities can be found in the last pages of [CM93].

Corollary 4. *If $n = 2$ (two-dimensional flow) and the fluid is incompressible and homogeneous (under conservative body forces), then $\frac{D\boldsymbol{\omega}}{Dt} = 0$ (in the inviscid case) and $\frac{D\boldsymbol{\omega}}{Dt} = \nu\Delta\boldsymbol{\omega}$ (in the viscous case).*

Proof. Follows from the preceding proposition and that, for $n = 2$, $\boldsymbol{\omega}$ is orthogonal to the velocity field, which is constrained to the plane $z = 0$. □

This corollary ensures the time invariance of vorticity along particle trajectories of bidimensional fluid flows. Such property has also been used in computer graphics animations [GLG95].

Some interesting objects of study consist of special curves and surfaces embedded in the fluid domain which are everywhere tangent to the vorticity distribution.

Definition 13 (vortex line/sheet)**.** *Let $\mathcal{C}$ ($\mathcal{S}$) be a regular curve (surface) in $\mathcal{D}$. We call $\mathcal{C}$ ($\mathcal{S}$) a* vortex line (sheet) *if, for each time instant $t \in \mathcal{I}$, it is tangent to the vorticity field $\boldsymbol{\omega}$, i.e. $\forall \mathbf{x} \in \mathcal{C}$ ($\in \mathcal{S}$), $\boldsymbol{\omega}(\mathbf{x}, t) \in T_{\mathbf{x}}\mathcal{C}$ ($\in T_{\mathbf{x}}\mathcal{S}$).*

These concepts model situations where there is a structured distribution of vorticity along a curve or a surface. For example, in a distant scale, a tornado could be mathematically modeled as a vortex line. In a closer scale a different notion would be necessary,

Definition 14 (vortex tube)**.** *Let $\mathcal{S} \subset \mathcal{D}$ be a two-dimensional surface (diffeomorphic to a disc) that is nowhere tangent to $\boldsymbol{\xi}$ at a time t and $\mathcal{C} := \partial\mathcal{S}$ its bounding curve. We call $\mathcal{T} := \bigcup_{\mathbf{x}\in\mathcal{C}} \{$integral line of $\boldsymbol{\omega}$ through $\mathbf{x}$ at time $t\}$ a* vortex tube.

Notice that *every vortex tube is a vortex sheet*, but the converse is not true in general. Intuitively, it is like a vortex sheet that has been twisted to the shape of a cylinder, but the vorticity field is kept tangent to this cylinder's wall (i.e. the vortex tube). This provides a simplified model for a tornado built-up by rising air-currents in the atmosphere. An invariance property also holds for ideally advected vortex structures.

Proposition 3. *If a surface (or curve) moves with the flow on an inviscid incompressible homogeneous fluid under conservative body forces and is a vortex sheet (line) at $t = t_0$, then it remains so for all time.*

Proof. Let $\mathcal{P} \subseteq \mathcal{S}$ be an arbitrary patch (regular surface diffeomorphic to a disc) of $\mathcal{S}$. From the definition of a vortex sheet, $\boldsymbol{\omega} \cdot \mathbf{n} = 0$ at time t_0 (where $\mathbf{n}$ is the unit normal to $\mathcal{S}$). By the preceding corollary, the flux of vorticity across $\mathcal{P}$ is zero for all time. Since $\mathcal{P}$ is arbitrary, by continuity, $\boldsymbol{\omega} \cdot \mathbf{n} = 0$ for all time in $\mathcal{S}$. Then, $\mathcal{S}$ will always remain a vortex sheet.

For a vortex line $\mathcal{C}$, we use the *Implicit Function Theorem* (under the assumption that $\boldsymbol{\omega}$ is non-zero at every point of $\mathcal{C}$ for all time) to describe $\mathcal{C}$ as (locally) the intersection of two vortex sheets and use the surface case. □

We also have the notion of *strength* of a vortex tube and an invariance theorem for this quantity as the tube evolves along the flow.

Theorem 4 (Helmholtz's Theorem)**.** *Consider an inviscid incompressible homogeneous fluid flow under conservative body forces. Then*

(i) If $\mathcal{C}_1$ and $\mathcal{C}_2$ are two arbitrary curves encircling the same vortex tube, then

$$\Gamma_{\mathcal{C}_1} = \oint_{\mathcal{C}_1} \mathbf{u} \cdot d\mathbf{s} = \oint_{\mathcal{C}_2} \mathbf{u} \cdot d\mathbf{s} = \Gamma_{\mathcal{C}_2}.$$

This common value is called the strength *of the vortex tube.*

(ii) The strength of a vortex tube is constant in time, as the tube moves with the fluid.

Proof. Let $\mathcal{S} \subset \mathcal{T}$ be the segment of the vortex tube $\mathcal{T}$ diffeomorphic to a bounded (open) cylinder whose bounding countours are the loops $\mathcal{C}_1$ and $\mathcal{C}_2$ (supposed non-intersecting). Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two arbitrary regular surfaces diffeomorphic to a disc whose bounding curves are given by $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. Let $\Sigma := \overline{\mathcal{S}_1 \cup \mathcal{S} \cup \mathcal{S}_2}$ be the boundary of the bounded fluid region $\mathcal{V}$ (which is "inside the vortex tube"). By *Gauss' Theorem*, the vector calculus identity $\nabla \cdot (\nabla \times \mathbf{F}) = 0$ and the definition of the vorticity field $\boldsymbol{\omega}$, we have

$$\begin{aligned} 0 &= \int_{\mathcal{V}} \nabla \cdot \boldsymbol{\omega}\, dV = \int_{\Sigma} \boldsymbol{\omega} \cdot \mathbf{n}\, dA = \int_{\mathcal{S}_1 \cup \mathcal{S}_2} \boldsymbol{\omega} \cdot \mathbf{n}\, dA + \int_{\mathcal{S}} \boldsymbol{\omega} \cdot \mathbf{n}\, dA \\ &= \int_{\mathcal{S}_1 \cup \mathcal{S}_2} \boldsymbol{\omega} \cdot \mathbf{n}\, dA = \int_{\mathcal{S}_1} \boldsymbol{\omega} \cdot \mathbf{n}\, dA + \int_{\mathcal{S}_2} \boldsymbol{\omega} \cdot \mathbf{n}\, dA = \oint_{\mathcal{C}_1} \mathbf{u} \cdot d\mathbf{s} - \oint_{\mathcal{C}_2} \mathbf{u} \cdot d\mathbf{s} \end{aligned}$$

where the last equality follows from *Stokes' Theorem* and, without loss of generality, the assumption that $\mathcal{C}_2$ is oriented clockwise with respect to Σ's unit (outward) normal field $\mathbf{n}$. Thus ending the proof of item *(i)*.

Having proved *(i)*, item *(ii)* now follows from the circulation theorem. □

This classic theorem allows us to observe that, if a vortex tube gets stretched and its cross-sectional area decreases, then the magnitude of $\boldsymbol{\omega}$ must increase. Thus, the stretching of vortex tubes can increase vorticity, but it cannot create it. This also implies that, with the increase of vorticity, the particles near the stretched region spin faster than, a phenomenon similar to the increased angular velocity of a figure skater when, once in rotation, she closes her arms to spin faster.[32]

3.9 Comments and Further Reading

In this chapter, we presented the basics of mathematical fluid dynamics. We tried to provide a coherent development of the main concepts and results used by the computer graphics community in the design of novel fluid animation tools and techniques.

The mathematical theory of fluid dynamics is a very broad subject, to *begin* a more in-depth study, we would recommend the textbooks [Bat99, CM93, Mey71].[33] Other classic texts on this theory include [Lam93, CF76].

Many interesting topics were completely ignored in the main text, even though they have been investigated for computer graphics applications (e.g. shallow-waters/long-waves equations [Whi99, KM90, LvdP02], wave equation [Eva98, Joh91, BMF07, YHK07], vorticity dynamics formulation [CM93, GLG95, PK05, ETK+07], cellular-automata and lattice-Boltzmann models [WG00, Thu07, HCSL02, WLMK04, TRS06]).

It is interesting to observe that our first notions of motion and the results related to mass conservation (and some others) are mostly grounded on the basic assumptions of continnum mechanics. As such, they would also be present in an account on the mathematical theory of elastic solids [MH94, Lov44, TPBF87, TF88, NMK+06], motivating this chapter as well for those also interested in the animation of elastically deformable objects.

[32] This effect is due to the *law of conservation of angular momentum*.

[33] Batchelor's book is a good source of physical interpretations for the results, Chorin's is a classic for those mathematically savvy and Meyer's text provides a very nice balance of physical motivation and mathematical rigour in a clear and well written text.

Chapter 4

Numerical Simulation of Fluids for Animation

In this chapter, we present two representatives of the numerical schemes most commonly used by the computer graphics community for the simulation of fluid flows with the purpose of animation, namely, the *Stable Fluids* and the *Smoothed Particle Hydrodynamics* methods.

In *Stable Fluids*, the Eulerian description of fluid motion is adopted and the fluid domain is discretized as a computational grid, in which the field quantities are stored and their differential operators are approximated by *finite differences* schemes. *Operator splitting / fractional-step* methods are used to decompose the governing partial differential equations into a series of smaller (and simpler to solve) equations connected by their respective initial conditions. To numerically solve each subproblem, efficient and stable methods are chosen from the *computational fluid dynamics* literature and adequately adapted to the performance and robustness requirements demanded by graphics applications.

The *Smoothed Particle Hydrodynamics* scheme is based on the Lagrangian viewpoint of fluid dynamics and relies on a discretization of the fluid mass in a finite number of blob-like moving particles. To represent fluid properties (e.g. velocities, density) and their differentials, this method employs tools from approximation theory and scattered data interpolation to *smooth* these quantities from particles' positions to the whole fluid domain. Such a particle-based description of the fluid dynamics reduces the governing PDE's to a coupled system of ordinary differential equations whose solution can be approximated by classical time-stepping methods.

4.1 Stable Fluids

In this section, we describe the basic stable fluids scheme as presented by Jos Stam in [Sta99], with a minor modification to add "vortex detail" through *vorticity confinement* forces, as was introduced by Fedkiw et al. in [FSJ01].

To keep clarity of presentation and focus on the salient features of this method, we restrict our attention to a simple flow regime and to the use of basic discretization and numerical schemes. At the end of this chapter, we provide further references to other works devoted to both extend and fix some problems encountered in the basic stable fluids approach.

4.1.1 Flow regime and governing equations

The flow regime with which we present the stable fluids scheme comprehends an wall-bounded square domain inside which an incompressible homogeneous viscous Newtonian fluid flows subject to external (user-applied) body forces. In these conditions, we model the fluid flow with the Navier-Stokes equations as presented in the previous chapter (the projected version of the NSE):[1]

$$\boxed{\mathbf{u}_t = \mathbb{P}\Big(-\mathbf{u}\cdot\nabla\mathbf{u} + \nu\Delta\mathbf{u} + \mathbf{b}\Big), \quad \operatorname{div}\mathbf{u} = 0, \quad \mathbf{u}|_{\partial\mathcal{W}} = 0} \tag{4.1}$$

where $\mathcal{W} = (0,1)^n \subset \mathbb{R}^n$ is the open unit hypercube (i.e. square or cube).[2]

4.1.2 Field representation and *spatial* discretization

In order to approximate candidate solutions of 4.1[3], we first need to represent the unknown field quantities involved $\big($e.g. $\mathbf{u} : \overline{\mathcal{W}} \times [0,+\infty) \to \mathbb{R}^n\big)$.

To represent a scalar (or vector) field $f : \overline{\mathcal{W}} \to \mathbb{R}$, we discretize its domain $\overline{\mathcal{W}}$ into a N^n grid, where $N \in \mathbb{N}$ is the grid's resolution along each dimension,

[1] The reasons for this choice will be clearer when we present the time-stepping stategy adopted in stable fluids.

[2] Notice that, although not everywhere smooth, $\partial\mathcal{W}$ is smooth enough (indeed piecewise $\mathcal{C}^\infty$) to hold the theory we developed in the previous chapter.

[3] It is still an open problem both the existence and uniqueness of solutions to 4.1 in dimension three. In fact, the *Clay Mathematics Institute* offers a prize of 1 million dollars to anyone that (dis)proves existence and uniqueness of solutions to the *Navier-Stokes equations*. For a detailed statement of this *Prize Problem* see http://www.claymath.org/millennium/Navier-Stokes_Equations/.

and store a sample of this function's value at the middle of each grid's voxel. This strategy to discretize a function at cell-centered samples in a regular grid is known as *collocation* and the structure is called a *collocated grid*. It is one of the classical representations used by *finite differences schemes*, which approximate the continuous differential equations on bounded domains by discrete difference equations on finite regular point sets [Str04, LeV07, Tre96].

With this domain discretization, we use the finite differences method to approximate both the spatial and temporal differential operators by difference equations. The most common of these difference approximations are (for a $1D$ scalar field):

$$\begin{aligned} \textit{forward differences:} \qquad & \frac{df}{dx}(x_i) \approx \frac{f_{i+1} - f_i}{h} \\ \textit{backward differences:} \qquad & \frac{df}{dx}(x_i) \approx \frac{f_i - f_{i-1}}{h} \\ \textit{central differences:} \qquad & \frac{df}{dx}(x_i) \approx \frac{f_{i+1} - f_{i-1}}{2h} \end{aligned}$$

where $h = \frac{1}{N}$ is the *space step*. These schemes to approximate the differential operator $\frac{d}{dx}$ provide the most basic building blocks with which we will search approximate solutions to differential equations. For example, assuming $n = 2$, the differential operators appearing in 4.1 could be discretized as:

$$\begin{aligned} \mathbf{u}_t\left(x_i, y_j, t_n\right) &\approx \frac{\mathbf{u}_{i,j}^{n+1} - \mathbf{u}_{i,j}^{n}}{k} \\ \nabla \mathbf{u}\left(x_i, y_j, t_n\right) &\approx \begin{pmatrix} \frac{u_{i+1,j}^n - u_{i-1,j}^n}{2h} & \frac{v_{i+1,j}^n - v_{i-1,j}^n}{2h} \\ \frac{u_{i,j+1}^n - u_{i,j-1}^n}{2h} & \frac{v_{i,j+1}^n - v_{i,j-1}^n}{2h} \end{pmatrix} \\ \Delta \mathbf{u}\left(x_i, y_j, t_n\right) &\approx \frac{\mathbf{u}_{i+1,j}^n - 2\mathbf{u}_{i,j}^n + \mathbf{u}_{i-1,j}^n}{h^2} + \frac{\mathbf{u}_{i,j+1}^n - 2\mathbf{u}_{i,j}^n + \mathbf{u}_{i,j-1}^n}{h^2} \\ \operatorname{div} \mathbf{u}\left(x_i, y_j, t_n\right) &\approx \frac{u_{i+1,j}^n - u_{i-1,j}^n}{2h} + \frac{v_{i,j+1}^n - v_{i,j-1}^n}{2h} \end{aligned}$$

where $k > 0$ is the *time step* and $\mathbf{u}\left(x, y, t\right) = \big(u\left(x, y, t\right), v\left(x, y, t\right)\big)$.[4]

> **Remark:** Various choices for combining spatial-temporal finite differences approximations are discussed in [Str04, LeV07] as well

[4] In these equations, the superscript ^{n} and the subscripts $_{i,j}$ denote a value sampled at time t_n and at the point $\mathbf{x}_{i,j} = (x_i, y_j)$, respectively.

> as their implications. We have decided not to discuss issues related to concepts as *consistency*, *order*, *stability*, or *accuracy* of approximations in this text. The interested reader is directed to the cited references for in-depth treatments of these subjects.[5]

A note is needed regarding this discretization scheme with respect to the treatment of boundary conditions. Whenever we need to approximate the differential at a boundary cell, the value at an inexistent adjacent cell is needed. To deal with this problem, an additional layer of cells is appended adjacently to $\partial\mathcal{W}$ (in the "outer side") and their values are determined by both the values at the "inner" boundary cells and the boundary conditions.

As an example, we deal with the homogeneous Dirichlet boundary conditions of 4.1 ($\mathbf{u}|_{\partial\mathcal{W}} = 0$) by assigning to an outer cell the negated value of its adjacent inner cell, as the averaged value at the boundary point between them thus satisfies the *no-slip boundary condition*. Homogeneous Neumann conditions are dealt with in the same way, except that, rather than assigning the negation of the value in the adjacent cell, the assigned value is the same stored at the adjacent inner cell. In this way, the normal derivative at the boundary point is zeroed, thus obeying the homogeneous Neumann boundary condition $\left(\frac{d\mathbf{u}}{d\mathbf{n}} = 0, \text{ at } \partial\mathcal{W}\right)$. This kind of condition appears both in the simulation of inviscid flows and in the determination of pressures between the fluid and solid obstacles (a stationary solid wall in our flow regime). Additional material regarding common boundary conditions in graphics applications can be found in [FM96a, BMF07].

4.1.3 Operator splitting/fractional-stepping

For simulating unsteady flows, we need a time-stepping scheme to advance in time the field quantities that represent our fluid. To this end, we adopt an *operator splitting/fractional-stepping* method [PTVF92, MQ02] to decompose 4.1 in a sequence of simpler subproblems (each one connected to the previous by its initial condition), which are solved by specialized schemes.

[5] An advice, if you decide to study finite difference schemes by the first edition of Strikwerda's book, manage to download its 13-pages errata!

Our splitting of 4.1 consists in the following sequence of subproblems[6]:

- *Convection:* which describes the "self-transport" in fluid flows.

$$\mathbf{u}_t = -\mathbf{u} \cdot \nabla \mathbf{u} \quad \Longleftrightarrow \quad \frac{D\mathbf{u}}{Dt} = 0$$

- *Forcing:* which accounts for external/body forces acting on the fluid.

$$\mathbf{u}_t = \mathbf{b}$$

- *Diffusion:* which propagates viscous stresses across the fluid domain.

$$\mathbf{u}_t = \nu \Delta \mathbf{u}$$

- *Projection:* which enforces incompressibility by applying pressure forces.

$$\mathbf{u}_t = -\nabla \tilde{p}, \quad \operatorname{div} \mathbf{u} = 0$$

where each of these time-dependent PDE's has as initial condition a solution of the previous equation. This means that, to advance a complete time step from $\mathbf{u}_{i,j}^n$ to $\mathbf{u}_{i,j}^{n+1}$, the discretized field $\mathbf{u}_{i,j}^n$ (assumed to be divergence-free) is used as the initial condition for numerical integration of the convection equation along a time interval of lenght k and the resulting (convected) field $\mathbf{u}_{i,j}^{n,c}$ is used as initial condition for the subsequent step (in this case, *forcing*). In this way, the sequence of fields $\mathbf{u}_{i,j}^{n,c}$, $\mathbf{u}_{i,j}^{n,f}$, $\mathbf{u}_{i,j}^{n,d}$ and $\mathbf{u}_{i,j}^{n,p}$ (respectively *convected*, *forced*, *diffused* and the final *projected* velocity fields) is generated from $\mathbf{u}_{i,j}^n$ and $\mathbf{u}_{i,j}^{n+1}$ is taken as the last of these intermediate fields.[7]

The processing of a time step by this splitting can be summarized as:

$$\mathbf{u}_{i,j}^n \xrightarrow{convection} \mathbf{u}_{i,j}^{n,c} \xrightarrow{forcing} \mathbf{u}_{i,j}^{n,f} \xrightarrow{diffusion} \mathbf{u}_{i,j}^{n,d} \xrightarrow{projection} \mathbf{u}_{i,j}^{n,p} \rightarrow \mathbf{u}_{i,j}^{n+1}$$

In the following subsections, we describe the methods chosen to solve each subproblem of the operator splitting scheme adopted by *Stable Fluids*.

[6] Respectively augmented by appropriate boundary conditions. Notice that, to simulate inviscid flows governed by *Euler's equations*, the diffusion step should be ommited.

[7] Notice that $\mathbf{u}_{i,j}^{n+1} = \mathbf{u}_{i,j}^{n,p}$ is a divergence-less velocity field, so that it is ready to be used as initial condition for integration into $\mathbf{u}_{i,j}^{n+2}$.

4.1.4 Semi-Lagrangian advection

The numerical integration of the convection equation $0 = \mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u} = \frac{D\mathbf{u}}{Dt}$, also known as the *(inviscid) Burgers' equation*, will be performed by approximating the solution of this nonlinear PDE by a sequence of solutions to the *advection/transport equation* $0 = f_t + \mathbf{u}\cdot\nabla f = \frac{Df}{Dt}$, a linear PDE (on f).

Such an approach to this problem is motivated by the interpretation of the material derivative operator $\frac{D}{Dt} = \partial_t + \mathbf{u}\cdot\nabla$ as the rate in which a quantity changes along a material trajectory. So the equation $\frac{Df}{Dt} = 0$ means that $f(\mathbf{x}(t), t)$ is constant in time, where $\mathbf{x}(t)$ denotes a material curve/trajectory (hence an integral line of $\mathbf{u}$, if it is assumed steady, i.e. $\mathbf{u}_t \equiv 0$).

This is the key reasoning behind the *semi-Lagrangian methodology for advection*, where the goal is to transport a field through a divergence-free steady velocity/vector field during a given time step $k > 0$ [SC91]. To this end, a semi-Lagrangian solver takes a velocity field $\mathbf{u}$, an initial (scalar, vector, tensor etc.) field f^n and outputs a field f^{n+1} as the solution of $f_t + \mathbf{u}\cdot\nabla f = 0$ (with initial condition f^n) evaluated at time t^{n+1}. Such a task is performed by assigning to $f^{n+1}_{i,j}$ the value from f^n at the point $\mathbf{x}(k) \in \mathcal{W}$, where $\mathbf{x}$ is the solution of the following ODE:

$$\dot{\mathbf{x}} = -\mathbf{u}(\mathbf{x}), \qquad \mathbf{x}(0) = \mathbf{x}_{i,j}.$$

Intuitively, this means that the value attributed to $f^{n+1}_{i,j}$ (located at $\mathbf{x}_{i,j}$) is the same as that from the point where the particle positioned at $\mathbf{x}_{i,j}$ at time t^{n+1} was at time $t^n = t^{n+1} - k$, i.e. to set $f^{n+1}_{i,j}$, we *backtrace* $\mathbf{x}_{i,j}$ along the vector field $\mathbf{u}$ during a time step k and evaluate f^n (which, for a discretized representation $f^n_{i,j}$, involves an interpolation procedure).

It is with the semi-Lagrangian method to solve advection equations, we approximate a solution to the convection equation. This is performed by advecting $\mathbf{u}^n_{i,j}$ along itself, i.e. we solve a vector advection equation where $\mathbf{u}^n_{i,j}$ is taken as both the steady velocity field (along which the transported field flows) and as the initial condition for its solution. Therefore, the value atributed to $\mathbf{u}^{n,c}_{i,j}$ is computed by tracing *backward in time* a massless particle at $\mathbf{x}_{i,j}$ along the steady vector field $\mathbf{u}^n_{i,j}$ (using some ODE time-stepper, e.g. Euler scheme) and interpolating $\mathbf{u}^n_{i,j}$ at the resulting position (e.g. using *bilinear interpolation*), the vector value thus found in assigned to $\mathbf{u}^{n,c}_{i,j}$.

4.1.5 Explicit forcing and vorticity confinement

To integrate the body forcing equation $\mathbf{u}_t = \mathbf{b}$ during a time step $k > 0$, we assume $\frac{\partial \mathbf{b}}{\partial t} = 0$ in $(t^n, t^{n+1}) \times \mathcal{W}$, effectively approximating $\mathbf{b}$ by a vector field which varies piecewise-constantly in time. With this approximation, we have $\mathbf{u}_{i,j}^{n,f} = \mathbf{u}_{i,j}^{n,c} + k\mathbf{b}_{i,j}^n$, which is equivalent to applying the forward Euler scheme to the ODE $\frac{d}{dt}\mathbf{u}_{i,j}(t) = \mathbf{b}_{i,j}^n$ and advancing from $\mathbf{u}_{i,j}^{n,c}$ to $\mathbf{u}_{i,j}^{n,f}$.

Vorticity confinement. A particular kind of body force was introduced by [FSJ01] with the purpose of adding "vortical detail" into fluid simulations for computer graphics, the *vorticity confinement forces* $\mathbf{b}_{vc}$. Their analytical expression $\mathbf{b}_{vc} = \epsilon h \left(\frac{\boldsymbol{\eta}}{\|\boldsymbol{\eta}\|} \times \boldsymbol{\omega} \right)$, where $\epsilon > 0$ is used to control the amount of detail added into the flow, $h > 0$ is the grid spacing, $\boldsymbol{\eta} = \nabla \|\boldsymbol{\omega}\|$ and $\boldsymbol{\omega} = \nabla \times \mathbf{u}$ is the vorticity as defined in the previous chapter, reflects the intended effect of "slapping" the vortical regions and augmenting the overall rotational motion present in the flow. Vorticity confinement forces have been both widely used and extended for simulation on coarse grids as those used in computer graphics applications [FSJ01, SRF05, SU94].

4.1.6 Implicit diffusion

The diffusion step in our splitting scheme is the classical *heat equation* with *Dirichlet boundary conditions* (the *no-slip* condition, $\mathbf{u}|_{\partial\mathcal{W}} = 0$) [Eva98, Joh91]. To numerically solve the diffusion equation, we adopt a fully implicit scheme based on a *backward in time and central in space* discretization scheme [PTVF92, Str04, LeV07, Tre96]. In such an approximation, the spatial differential operator (in this case, the laplacian Δ) is discretized using central differences yielding a coupled linear system of ODE's on the unknown variables. Then, the backward Euler scheme is applied to this resulting ODE:

$$\mathbf{u}_t = \nu\Delta\mathbf{u} \quad \longrightarrow \quad \frac{d}{dt}\tilde{\mathbf{u}} = \frac{\nu}{h^2}\mathbf{L}\tilde{\mathbf{u}} \quad \longrightarrow \quad \frac{\tilde{\mathbf{u}}^{n+1} - \tilde{\mathbf{u}}^n}{k} = \frac{\nu}{h^2}\mathbf{L}\tilde{\mathbf{u}}^{n+1}$$

where $\tilde{\mathbf{u}}$ groups the unknown grid values $\mathbf{u}_{i,j}$ in a vector of dimension N^2 and $\mathbf{L}$ is the finite differences matrix representing the laplacian operator.[8]

[8] A component-wise version of the above differential/difference equations would read:

$$\frac{d}{dt}\mathbf{u}_{i,j} = \nu \left(\frac{\mathbf{u}_{i+1,j} - 2\mathbf{u}_{i,j} + \mathbf{u}_{i-1,j}}{h^2} + \frac{\mathbf{u}_{i,j+1} - 2\mathbf{u}_{i,j} + \mathbf{u}_{i,j-1}}{h^2} \right) \xleftarrow[condition]{initial} \mathbf{u}_{i,j}^0 = \mathbf{u}_{i,j}^{n,f}$$

Notice that the last equation from the backward Euler discretization is equivalent to solving the linear system of equations $\left(\mathbf{I} - \nu\frac{k}{h^2}\mathbf{L}\right)\tilde{\mathbf{u}}^{n+1} = \tilde{\mathbf{u}}^n$, where $\tilde{\mathbf{u}}^{n+1}$ corresponds to $\mathbf{u}_{i,j}^{n,d}$ and $\tilde{\mathbf{u}}^n$ represents $\mathbf{u}_{i,j}^{n,f}$. It is interesting to notice that the matrix on the right-hand side of this system has good numerical properties: it is *symmetric*, *positive definite* and *sparse*. So specialized (and fast) methods can be applied to approximately solve these equations (e.g. *preconditioned conjugate gradients* and *multigrid* methods [Dem97]).[9]

Another very important property of the above discretization scheme is that $\left(\mathbf{I} - \nu\frac{k}{h^2}\mathbf{L}\right)$ has eigenvalues stricly greater than 1, an implication of this fact (and a very instructive exercise in numerical analysis of ODE's) is that, no matter how large the time step k, the numerical integration conserves the asymtoptic behaviour of the original ODE (as $t \to +\infty$, the solution converges to the origin). This means that we can take arbitrarily large time steps without suffering with stability concerns, i.e. the simulation never "blows up" as a consequence of diffusion, a very important feature for simulators designed for computer graphics applications [Sta99, CMIT02].

4.1.7 Pressure projection

After performed the diffusion step, the resulting velocity field $\mathbf{u}_{i,j}^{n,d}$ is not divergence-free, so doesn't respect the law of conservation of mass. Hence we need to solve for pressure, since it has an incompressibility enforcement role (as noticed in the previous chapter), and complete our splitting pipeline.

After discretizing in time the projection equation we have that:

$$\mathbf{u}_t = -\nabla\tilde{p},\ \operatorname{div}\mathbf{u} = 0 \longrightarrow \frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{k} = -\nabla\tilde{p}^{n+1},\ \operatorname{div}\mathbf{u}^{n+1} = 0$$

Taking the divergence on both sides of the discretized equation yields:

$$-\Delta\tilde{p}^{n+1} = -\frac{1}{k}\nabla\cdot\mathbf{u}^n,\ in\ \mathcal{W}$$

$$\frac{\mathbf{u}_{i,j}^{n,d} - \mathbf{u}_{i,j}^{n,f}}{k} = \nu\left(\frac{\mathbf{u}_{i+1,j}^{n,d} - 2\mathbf{u}_{i,j}^{n,d} + \mathbf{u}_{i-1,j}^{n,d}}{h^2} + \frac{\mathbf{u}_{i,j+1}^{n,d} - 2\mathbf{u}_{i,j}^{n,d} + \mathbf{u}_{i,j-1}^{n,d}}{h^2}\right)$$

Notice that, actually, $\tilde{\mathbf{u}}$ would have dimension $(2N)^2$, because $\mathbf{u}$ is a vector field in $\mathbb{R}^2$, but here it is treated as a scalar field, since the laplacian operator is applied separately for each coordinate field u, v of $\mathbf{u} = (u, v)$.

[9] The implementation of *Stable Fluids* we have is based on an iterative Gauss-Seidel method for simplicity reasons.

where the no-slip condition (in fact, the kinematic condition that $\mathbf{u} \parallel \partial\mathcal{W}$) and the original partial differential equation give a homogeneous Neumann condition on $\tilde{p}^{n+1}$, i.e. $\frac{d\tilde{p}^{n+1}}{d\mathbf{n}} = 0$ on $\partial\mathcal{W}$. Solved this *Poisson problem with Neumann boundary conditions*, we take $\mathbf{u}^{n+1} = \mathbf{u}^n - k\nabla\tilde{p}^{n+1}$. The interesting fact behind all these calculations is that we have in fact taken $\mathbf{u}^{n+1}$ as the projection of $\mathbf{u}^n$ by $\mathbb{P}$, what can be verified by projecting the last equation:

$$\mathbf{u}^{n+1} = \mathbb{P}\left(\mathbf{u}^{n+1}\right) = \mathbb{P}\left(\mathbf{u}^n - k\nabla\tilde{p}^{n+1}\right) = \mathbb{P}\left(\mathbf{u}^n\right)$$

This configures a remarkable application of the projection operator $\mathbb{P}$ defined through the *Helmholtz-Hodge Decomposition Theorem*. It provides a natural connection between the abstraction from the theory and a tool in the practicalities of numerical computation. Such a view for incompressibility enforcement has been extensively exploited and extended by others both to advance the theory and to design new simulation techniques that model the coupling of multiple fluids and solids [CM93, HK05, LSSF06, CMT04, GSLF05, FOKG05, CGFO06, BBB07].

With these considerations we are able to design a numerical scheme to enforce flow incompressibility and solve the final step of our splitting method. To apply the above procedure for advancing from $\mathbf{u}_{i,j}^{n,d}$ to $\mathbf{u}_{i,j}^{n,p}$, all we need is to solve the Poisson problem described before. This is done by discretizing in space the laplacian operator in the very same way as done for diffusion[10], with the sole observation regarding the change from Dirichlet to Neumann boundary conditions as noted in the subsection on spatial discretization. The discretization procedure yields a linear system of equations:

$$-\Delta\tilde{p} = f \quad \longrightarrow \quad -\frac{1}{h^2}\mathbf{L}\tilde{\mathbf{p}} = -\frac{1}{2hk}\mathbf{D}\tilde{\mathbf{u}} \quad \Longleftrightarrow \quad (-\mathbf{L})\,\tilde{\mathbf{p}} = -\frac{h}{2k}\mathbf{D}\tilde{\mathbf{u}}$$

where $\mathbf{L}$ is as before, $\tilde{\mathbf{p}}$ groups the unknown grid values $\tilde{p}_{i,j}^n$ in a vector of dimension N^2, $\mathbf{D}$ denotes the finite differences matrix of the divergence operator and $\tilde{\mathbf{u}}$ groups the *known* grid values $\mathbf{u}_{i,j}^{n,d}$ (now really in a vector of dimension $(2N)^2$, since the divergence operator needs derivatives for both u and v of $\mathbf{u} = (u, v)$).[11] Where $(-\mathbf{L})$ shares the same properties commented about the matrix derived for the diffusion step, hence the same methods used to solve numerically that problem can be reused in the projection step.

[10] Just notice that, for diffusion, the laplacian operator was applied for each u, v of $\mathbf{u} = (u, v)$. Therefore, we will use the same notation for the finite differences matrix $\mathbf{L}$ used to solve for pressure.

[11] The above equations in matrix/vector-form comprise just a compact version of the

After solving for $\tilde{p}^n_{i,j}$, we finally compute $\mathbf{u}^{n,p}_{i,j} = \mathbf{u}^{n,d}_{i,j} - k\nabla\tilde{p}^n_{i,j}$ by approximating the gradient operator by central differences, yielding the update

$$\mathbf{u}^{n,p}_{i,j} = \mathbf{u}^{n,d}_{i,j} - \frac{k}{2h}\begin{pmatrix}\tilde{p}^n_{i+1,j} - \tilde{p}^n_{i-1,j} \\ \tilde{p}^n_{i,j+1} - \tilde{p}^n_{i,j-1}\end{pmatrix},$$

finishing the pressure projection step and one step of our splitting scheme.

4.1.8 Experiments

As an illustration for our description of the basic *Stable Fluids* method, we implemented a simple fluid simulator in the `C` language for the flow regime described in 4.1.1. In this subsection, we present some results generated with that solver and some issues which appeared during its development. Appendix A presents the core code of our simple *Stable Fluids* solver as a more concrete example of the method described.

Flow visualization. For our basic *Stable Fluids* solver, we implemented a couple simple visualizations to inspect some concepts discussed in Chapter 3:

- *scalar fields* advected and diffused along the fluid flow as marker dye fields (*top left* of Figure 4.1), evolving according to $\frac{Df}{Dt} = \kappa\Delta f + \mathbf{f}$;
- *velocity fields* depicted by small vectors originating at computational cell centers (*top right* of Figure 4.1);
- *scalar fields* derived from other fields (e.g. the velocity and vorticity magnitude fields $\|\mathbf{u}\|$ and $\|\boldsymbol{\omega}\|$, at the *middle* and *bottom* of Figure 4.1).

These visualizations were implemented with `OpenGL` with simple user interactions using `GLUT` (e.g. mouse-driven *body forcing* and *dye sourcing*).

---

linear system formed from:

$$-\left(\frac{\tilde{p}^n_{i+1,j} - 2\tilde{p}^n_{i,j} + \tilde{p}^n_{i-1,j}}{h^2} + \frac{\tilde{p}^n_{i,j+1} - 2\tilde{p}^n_{i,j} + \tilde{p}^n_{i,j-1}}{h^2}\right) = -\frac{1}{k}\left(\frac{u^{n,d}_{i+1,j} - u^{n,d}_{i-1,j}}{2h} + \frac{v^{n,d}_{i,j+1} - v^{n,d}_{i,j-1}}{2h}\right)$$

Some results. The solver presented in Appendix A is able to simulate both inviscid and viscous flows. We comment a couple results generated using it.

Figure 4.1 contains some visualizations of a simple flow in which the fluid was acted by localized mouse-driven body forces pushing the fluid up from bottom. This motion induced the generation of two main vortices which were advected along the inviscid flow and transported the dye scalar field.

Figure 4.2 depicts a later frame in a more complex viscous flow animation, in which there is plenty of rotational behaviour and localized vortices kept "alive" by vorticity confinement forces.

Figure 4.3 illustrates the generation of vorticity along boundaries in viscous Navier-Stokes flows (left) and the sole transport of vorticity along with inviscid Euler flows (right), in accordance with Corollary 4.

Implementation issues. The development of a fluid simulator is a delicate problem, many little details must be kept in mind when implementing such a code. Care must be taken in enforcing the boundary conditions at each stage during simulation and in performing interpolations at the advection stage. Other issue is solving the linear systems which arise in the finite difference discretization of the linear PDE's, often it is not needed to build the system matrix to apply an iterative method [Dem97], we employed a simple Gauss-Seidel method, but other methods (e.g. Krylov subspace methods) only require the programmer to provide a routine which returns the result of multiplying a given vector by the system matrix and have better convergence rate properties (e.g. preconditioned conjugate gradients) [Saa03]. These methods avoid keeping large (and sparse) matrices saving memory, time and implementation/maintenance efforts [BBC+94] of both diffusion and projection stages (we didn't use them in our code to keep it simple). For other details on developing more complex simulators, we recommend the practical approach taken in the lecture notes [BMF07].

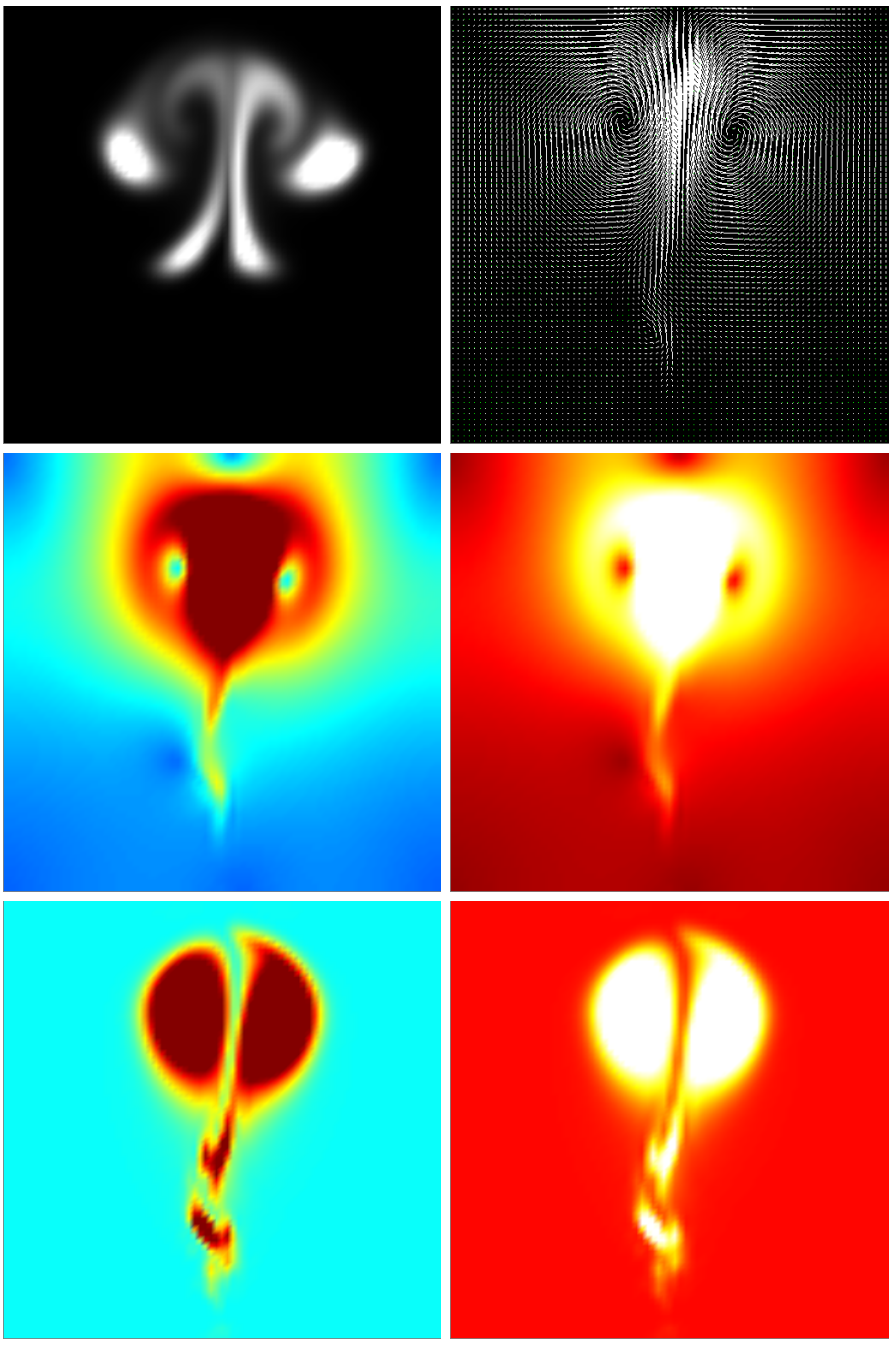

Figure 4.1: A "simple" flow example; *Top left:* scalar field advected/diffused along the velocity field depicted in the *top right* figure; *Middle:* magnitudes of the velocity field; *Bottom:* magnitudes of the vorticity field.

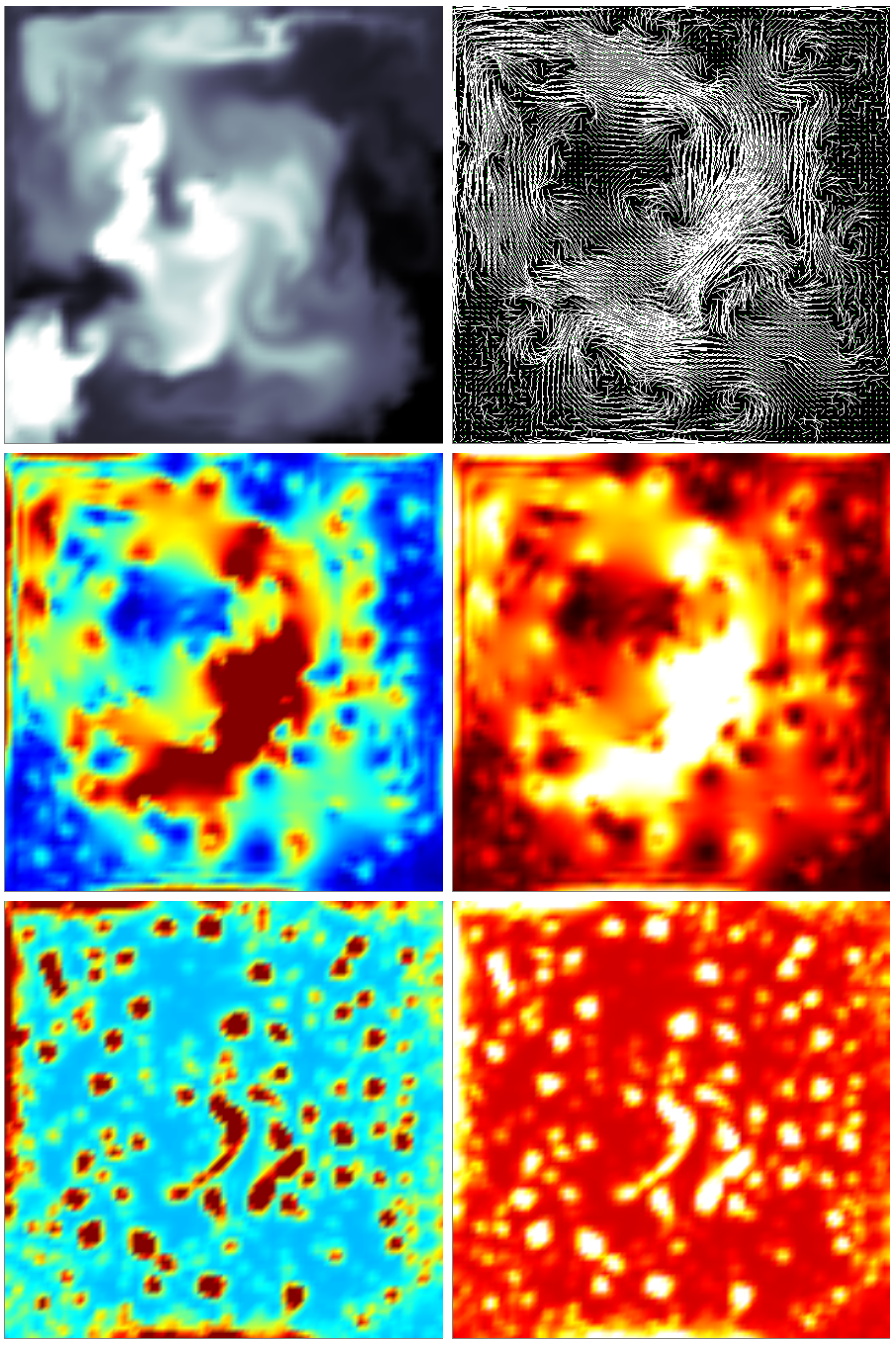

Figure 4.2: A more complex flow example; *Top left:* scalar field advected/diffused along the velocity field depicted in *top right* figure; *Middle:* magnitudes of the velocity field; *Bottom:* magnitudes of the vorticity field.

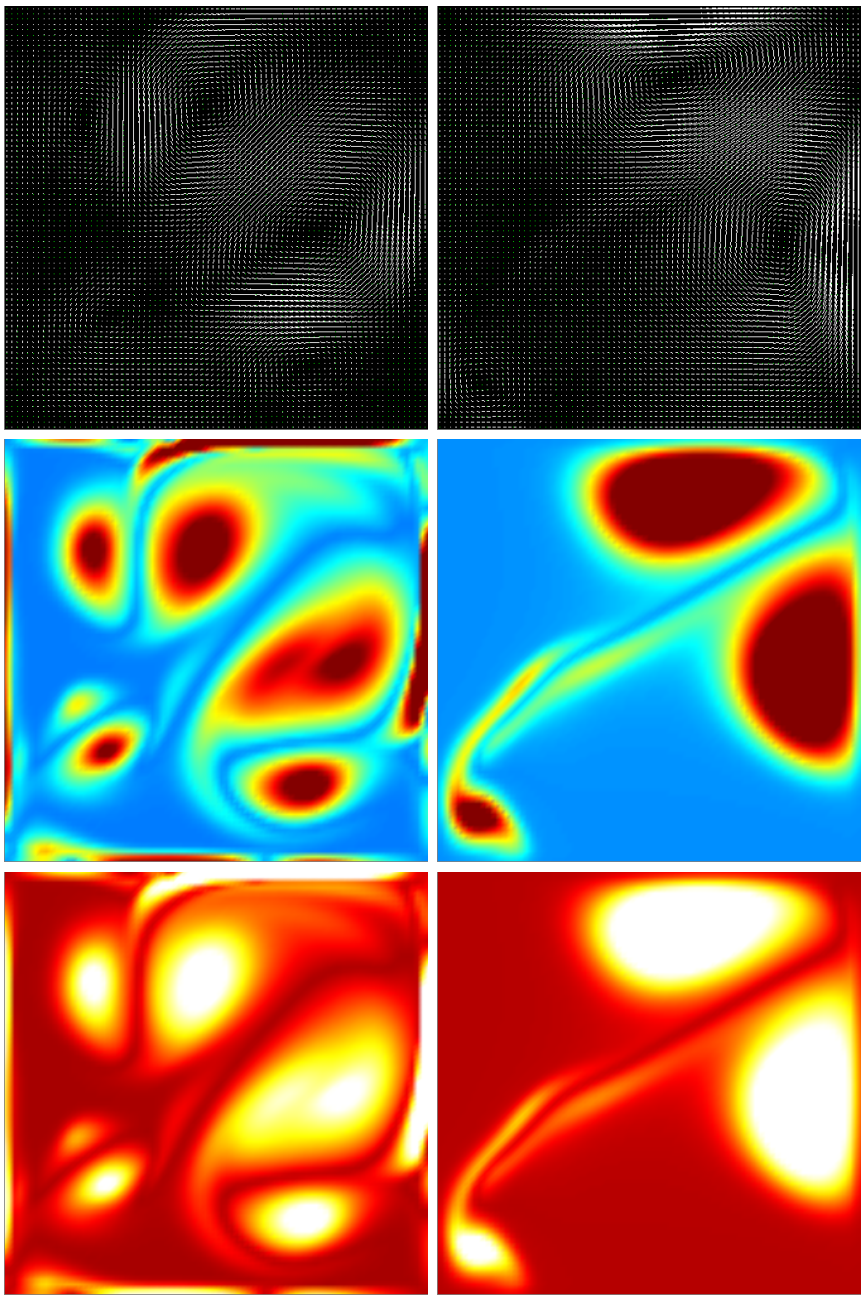

Figure 4.3: In viscous flows governed by the Navier-Stokes' equations, vorticity is generated along boundaries. *Left:* a viscous flow animated using the Navier-Stokes' equations and its respective vorticity field. *Right:* an inviscid flow governed by Euler's equations (where vorticity slips along boundaries).

4.2 Smoothed Particle Hydrodynamics

This section is devoted to describe the *Smoothed Particle Hydrodynamics* (SPH) methodology to simulate fluid flows. Our presentation is roughly based on the works of Müller et al. [MCG03], which employed it in the simulation of fluids for interactive graphics applications, and Paiva Neto [Net07], who extended SPH in his Ph.D. thesis to simulate non-Newtonian viscoplastic and multiphase flows also for computer graphics applications.[12]

As done before, our focus is on clarity and simplicity of presentation. Although our considerations are restricted to a simple flow regime, at the end of this chapter, we provide further references to works concerned with deeper descriptions of the SPH framework and its implementation issues, both for computer graphics and its "more serious" original applications [GM77, Luc77].

4.2.1 Flow regime and governing equations

Our description of the smoothed particle hydrodynamics method for flow simulation is based on a regime where the fluid has uniform viscosity and the pressure obeys an *equation of state* of the type $p(\mathbf{x},t) = f(\rho(\mathbf{x},t))$. Under these assumptions, the governing equations for such a fluid flow are:[13]

$$\textit{continuity equation:} \qquad \frac{D\rho}{Dt} = -\rho\,(\nabla\cdot\mathbf{u})$$

$$\textit{momentum equation:} \qquad \rho\frac{D\mathbf{u}}{Dt} = -\nabla p + (\lambda+\mu)\,\nabla(\nabla\cdot\mathbf{u}) + \mu\Delta\mathbf{u} + \rho\,\mathbf{b}$$

The Lagrangian formulation used to state the governing equations is due to the discretization strategy adopted by SPH (to be explained later), which also makes use of the identity $\mathbf{a} = \frac{D\mathbf{u}}{Dt}$ (where $\mathbf{a}$ stands for acceleration).

As noted in the previous chapter, the explicit use of an equation of state doesn't imply the incompressibility condition $\nabla\cdot\mathbf{u} \equiv 0$. Therefore, it is conceptually incorrect (at best) to adopt an equation of state *and* assume $\nabla\cdot\mathbf{u} \equiv 0$ to simplify the governing equations, even if this equation of state is designed to direct the system to a *quasi-incompressible* state. This was done by Müller et al. in [MCG03], when they adopted $\rho\frac{D\mathbf{u}}{Dt} = -\nabla p + \mu\Delta\mathbf{u} + \rho\,\mathbf{b}$ as their momentum equation and $p = c^2(\rho-\rho_0)$ as their equation of state.

[12] Before [MCG03] use SPH to simulate liquids, the *Smoothed Particle Hydrodynamics* framework has been exploited by computer graphics both to depict fire and gaseous phenomena [SF95] and to animate the soft deformations of elastic solids [DG96].

[13] Already derived in the previous chapter.

4.2.2 Field representation and *fluid* discretization

Instead of representing the field quantities by regularly sampled values and their differentials by difference equations (as done in stable fluids), SPH relies on scattered data approximation schemes and analytical differentiation of its approximations. This is accomplished by exploting the integral representation of a function $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$ as a convolution with *Dirac's delta distribution* δ and that this distribution can be defined as a generalized limit of certain smooth functions W_h, i.e. $h \to 0 \Rightarrow W_h \to \delta$ [Net07, II01]:

$$f(\mathbf{x}) = \int_\Omega f(\mathbf{x}')\,\delta(\mathbf{x}-\mathbf{x}')d\mathbf{x}' = \lim_{h\to 0}\left\{\int_\Omega f(\mathbf{x}')\,W_h(\mathbf{x}-\mathbf{x}')d\mathbf{x}'\right\} \tag{4.2}$$

Motivated by 4.2, the *SPH approximation* $\langle f\rangle$ to the field f is defined by a given family W_h of *smooth kernel functions* and a fixed $h > 0$ as

$$\langle f(\mathbf{x})\rangle := \int_\Omega f(\mathbf{x}')\,W_h(\mathbf{x}-\mathbf{x}')d\mathbf{x}' \tag{4.3}$$

which is numerically discretized by the quadrature formula:

$$\langle f(\mathbf{x})\rangle \approx \sum_j f(\mathbf{x}_j)\,W_h(\mathbf{x}-\mathbf{x}_j)\Delta V_j = \sum_j f_j \frac{m_j}{\rho_j} W_h(\mathbf{x}-\mathbf{x}_j) \tag{4.4}$$

where the weight $\Delta V_j = \frac{m_j}{\rho_j}$ corresponds to the volume associated to the j-th fluid particle (located at the quadrature point $\mathbf{x}_j \in \Omega$), m_j to its mass and ρ_j to its associated *specific mass* $\big($notice that f_j is an abbreviation to $f(\mathbf{x}_j)\big)$. This quadrature form for the integral SPH approximation is interpreted as discretizing the *fluid mass* into a finite number of particles which evolve according to the governing equations, that describe the system dynamics by a coupled set of nonlinear ODE's (a topic treated in the next subsection).

Inspecting 4.4, we notice that whenever the family W_h is composed by compactly supported kernels, with *influence radius* say κh, $\big($i.e. $\exists\kappa > 0$ such that $\|\mathbf{x}-\mathbf{x}_j\| \geq \kappa h \Rightarrow W_h(\mathbf{x}-\mathbf{x}_j) = 0\big)$ the sum in 4.4 effectively only takes place for those particles which are less than κh away from $\mathbf{x}$. Defining the neighboring particles of a point $\mathbf{x} \in \mathcal{W}$ as $\mathcal{N}(\mathbf{x}) := \{\, j \in \mathbb{N} \,|\, \|\mathbf{x}-\mathbf{x}_j\| < \kappa h \,\}$, we can rewrite the *discrete SPH approximation* 4.4 as [Net07]:

$$\langle f(\mathbf{x})\rangle \approx \sum_j f_j \frac{m_j}{\rho_j} W_h(\mathbf{x}-\mathbf{x}_j) = \sum_{j\in\mathcal{N}(\mathbf{x})} f_j \frac{m_j}{\rho_j} W_h(\mathbf{x}-\mathbf{x}_j) \tag{4.5}$$

To simplify notation, we adopt $\langle f(\mathbf{x})\rangle$ to represent the discrete SPH approximation in 4.5, since we won't use its integral form in this text anymore.

The compact support property is just one of a few useful (hence desired) properties for choosing the family of smooth kernels $W_h : \mathbb{R}^n \to \mathbb{R}$, [14]

- *Even:* $\forall h > 0\ \forall \mathbf{x} \in \mathbb{R}^n,\ W_h(\mathbf{x}) = W_h(-\mathbf{x})$
- *Non-negative:* $\forall h > 0\ \forall \mathbf{x} \in \mathbb{R}^n,\ W_h(\mathbf{x}) \geq 0$
- *Smooth:* $\forall h > 0,\ W_h \in C^k(\mathbb{R}^n)$, where $k > 1$
- *Partition of unity:* $\forall h > 0,\ \int_\Omega W_h(\mathbf{x})\ d\mathbf{x} = 1$
- *Compact support:* $\exists \kappa > 0,\ \forall \mathbf{x} \in \mathbb{R}^n,\ \|\mathbf{x} - \mathbf{x}_j\| \geq \kappa h \Rightarrow W_h(\mathbf{x}) = 0$
- *Convergence:* $h \to 0 \Longrightarrow W_h \to \delta$

An important class of radial compactly supported piecewise-smooth kernels can be derived from splines functions, examples of those are the cubic and quintic splines defined by [Mon05, Net07]:

$$W_h(\mathbf{x}) = \frac{\alpha_n}{h^n}\ w_3\left(\frac{\|\mathbf{x}\|}{h}\right) \quad \text{and} \quad W_h(\mathbf{x}) = \frac{\beta_n}{h^n}\ w_5\left(\frac{\|\mathbf{x}\|}{h}\right)$$

where $\alpha_1 = \frac{1}{6}$, $\alpha_2 = \frac{15}{14\pi}$, $\alpha_3 = \frac{1}{4\pi}$, $\beta_1 = 120$, $\beta_2 = \frac{7}{478\pi}$, $\beta_3 = \frac{3}{359\pi}$ and

$$w_3(q) = \begin{cases} (2-q)^3 - 4(1-q)^3, & \text{for } 0 \leq q \leq 1 \\ (2-q)^3, & \text{for } 1 \leq q \leq 2 \\ 0, & \text{for } q > 2 \end{cases}$$

$$w_5(q) = \begin{cases} (3-q)^5 - 6(2-q)^5 + 15(1-q)^5, & \text{for } 0 \leq q \leq 1 \\ (3-q)^5 - 6(2-q)^5, & \text{for } 1 \leq q \leq 2 \\ (3-q)^5, & \text{for } 2 \leq q \leq 3 \\ 0, & \text{for } q > 3 \end{cases}$$

Altought the Gaussian is the most common kernel, it doesn't have compact support (even thought it has a very fast decay), what is not a good property for computational implementations. Nevertheless, for its simplicity

[14] Notice that some of these properties are *necessary* to allow the manipulations that lead to the integral SPH approximation [II01].

(what is a key goal of our presentation), we employ the 2D Gaussian function in our simple example implementation (Appendix B).

After provided an appropriate SPH approximation for a field quantity, its spatial derivatives can be analytically computed from the discrete form:

$$\begin{aligned}
\nabla \langle f(\mathbf{x}) \rangle &= \sum_{j \in \mathcal{N}(\mathbf{x})} \frac{m_j}{\rho_j} f_j \nabla W_h(\mathbf{x} - \mathbf{x}_j) \\
\Delta \langle f(\mathbf{x}) \rangle &= \sum_{j \in \mathcal{N}(\mathbf{x})} \frac{m_j}{\rho_j} f_j \Delta W_h(\mathbf{x} - \mathbf{x}_j) \\
\nabla \cdot \langle \mathbf{f}(\mathbf{x}) \rangle &= \sum_{j \in \mathcal{N}(\mathbf{x})} \frac{m_j}{\rho_j} \mathbf{f}_j \cdot \nabla W_h(\mathbf{x} - \mathbf{x}_j) \\
\nabla \times \langle \mathbf{f}(\mathbf{x}) \rangle &= \sum_{j \in \mathcal{N}(\mathbf{x})} \frac{m_j}{\rho_j} \mathbf{f}_j \times \nabla W_h(\mathbf{x} - \mathbf{x}_j)
\end{aligned}$$

Even though these approximations to differentials of fields are natural, in practice they suffer from a number of problems related both to accumulation of numerical errors and to bad representation of physical properties (e.g. symmetry in Newton's Third Law of motion). To overcome these problems, identities from calculus are exploited to both symmetrize and amortize numerical imprecisions. For example, better numerically conditioned rules can be derived by subtracting from the above equations what should be a null constant $\big($e.g. $\nabla(1)$ or $\nabla \times (\mathbf{1})$, approximated by $\nabla \langle 1 \rangle$ and $\mathbf{1} \cdot \nabla \langle 1 \rangle\big)$:

$$\begin{aligned}
\langle \nabla f(\mathbf{x}_i) \rangle &\approx \nabla \big(\langle f(\mathbf{x}_i) \rangle - f_i \langle 1 \rangle \big) = \sum_{j \in \mathcal{N}(\mathbf{x}_i)} \frac{m_j}{\rho_j} (f_j - f_i) \nabla W_h(\mathbf{x}_i - \mathbf{x}_j) \\
\langle \Delta f(\mathbf{x}_i) \rangle &\approx \Delta \big(\langle f(\mathbf{x}_i) \rangle - f_i \langle 1 \rangle \big) = \sum_{j \in \mathcal{N}(\mathbf{x}_i)} \frac{m_j}{\rho_j} (f_j - f_i) \Delta W_h(\mathbf{x}_i - \mathbf{x}_j) \\
\langle \nabla \cdot \mathbf{f}(\mathbf{x}_i) \rangle &\approx \nabla \cdot \big(\langle \mathbf{f}(\mathbf{x}_i) \rangle - \mathbf{f}_i \langle 1 \rangle \big) = \sum_{j \in \mathcal{N}(\mathbf{x}_i)} \frac{m_j}{\rho_j} (\mathbf{f}_j - \mathbf{f}_i) \cdot \nabla W_h(\mathbf{x}_i - \mathbf{x}_j) \\
\langle \nabla \times \mathbf{f}(\mathbf{x}_i) \rangle &\approx \nabla \times \big(\langle \mathbf{f}(\mathbf{x}_i) \rangle - \mathbf{f}_i \langle 1 \rangle \big) = \sum_{j \in \mathcal{N}(\mathbf{x}_i)} \frac{m_j}{\rho_j} (\mathbf{f}_j - \mathbf{f}_i) \times \nabla W_h(\mathbf{x}_i - \mathbf{x}_j)
\end{aligned}$$

Altought these rules can be numerically more suitable for approximating the differential operators, they are not suitable to use in physical simulations, since they do not respect the Third Law of motion (what can be verified by analyzing just the interaction of two particles with different masses).

Symmetric approximations which work fairly well in practice [Net07] can be derived by considering the following identity:

$$\frac{1}{g}\nabla f = \frac{f}{g^k}\nabla\left(\frac{1}{g^{1-k}}\right) + \frac{1}{g^{2-k}}\nabla\left(\frac{f}{g^{k-1}}\right), \quad \text{where } k \in \mathbb{Z} \tag{4.6}$$

By means of identity 4.6, we can establish for $\frac{1}{\rho}\nabla f$ [Net07]:

$$\left\langle \frac{\nabla f}{\rho}(\mathbf{x}_i) \right\rangle = \sum_{j\in\mathcal{N}(\mathbf{x}_i)} m_j \left(\frac{f_j}{\rho_i^{2-k}\rho_j^k} - \frac{f_i}{\rho_i^k \rho_j^{2-k}}\right) \nabla W_h(\mathbf{x}_i - \mathbf{x}_j)$$

which give rise to symmetric approximations as, by taking $k = 1$:

$$\langle \nabla f(\mathbf{x}_i) \rangle \approx \sum_{j\in\mathcal{N}(\mathbf{x}_i)} \frac{m_j}{\rho_j}(f_j + f_i)\nabla W_h(\mathbf{x}_i - \mathbf{x}_j)$$
$$\langle \nabla \cdot \mathbf{f}(\mathbf{x}_i) \rangle \approx \sum_{j\in\mathcal{N}(\mathbf{x}_i)} \frac{m_j}{\rho_j}(\mathbf{f}_j + \mathbf{f}_i)\cdot\nabla W_h(\mathbf{x}_i - \mathbf{x}_j)$$

and, by taking $k = 2$:

$$\left\langle \frac{\nabla f}{\rho}(\mathbf{x}_i) \right\rangle \approx \sum_{j\in\mathcal{N}(\mathbf{x}_i)} m_j \left(\frac{f_j}{\rho_j^2} + \frac{f_i}{\rho_i^2}\right) \nabla W_h(\mathbf{x}_i - \mathbf{x}_j)$$
$$\left\langle \frac{\nabla \cdot \mathbf{f}}{\rho}(\mathbf{x}_i) \right\rangle \approx \sum_{j\in\mathcal{N}(\mathbf{x}_i)} m_j \left(\frac{\mathbf{f}_j}{\rho_j^2} + \frac{\mathbf{f}_i}{\rho_i^2}\right) \cdot \nabla W_h(\mathbf{x}_i - \mathbf{x}_j)$$

After deriving a systematic strategy to discretely represent field quantities and their differentials, we have enough material to numerically solve the governing equations. However, a note on how to deal with boundaries is appropriate at this point. The classical approach to boundary conditions in SPH is to approximate the solid surface by a sample of *boundary particles* (or *ghost particles*) and, to each fluid particle which is within the radius of influence of some boundary particle, to apply a compactly supported repulsive force (as those derived from a Lennard-Jones potentials, common in computer graphics and molecular dynamics [BMF07]). Such a force (*per unit mass*) is of the form [Mon94]:

$$\frac{1}{m_i}\mathbf{f}(\mathbf{r}_{ib}) = \begin{cases} D\left(\left(\frac{r_0}{r_{ib}}\right)^{p_1} - \left(\frac{r_0}{r_{ib}}\right)^{p_2}\right)\frac{\mathbf{r}_{ib}}{r_{ib}^2}, & \text{for } r_{ib} \le r_0 \\ 0, & \text{for } r_{ib} > r_0 \end{cases}$$

where D is a problem dependent constant (of the same order of the maximum expected speed squared), $p_1 > p_2$ are constants (usually $(p_1, p_2) = (4, 2)$ or $(p_1, p_2) = (12, 6)$), $\mathbf{r}_{ib} = \mathbf{x}_i - \mathbf{x}_b$ and $r_{ib} = \|\mathbf{x}_i - \mathbf{x}_b\|$.

This provides a physically-inspired approach to deal with stationary solid walls. However, when these obstacles have complex geometries and (possibly) move, different approaches should be used [MFZ97, Mon05, Net07].[15]

4.2.3 The discretized governing equations

The approach employed by SPH to discretize the governing equations is based on the Lagrangian formulation of the governing equations of fluid flow. Altought [MCG03] depart from the equations mentioned above, we adopt the differential form of the momentum equation 3.3 as derived in section 3.4, along with the tensor field deduced in section 3.6, namely:

$$\rho \frac{D\mathbf{u}}{Dt} = \operatorname{Div} \mathbf{T} + \rho \mathbf{b}, \text{ where } \mathbf{T} = -p\mathbf{I} + \boldsymbol{\sigma} \text{ and } \boldsymbol{\sigma} = \lambda \operatorname{tr}(\mathbf{D})\, \mathbf{I} + 2\mu \mathbf{D}$$

recalling $\mathbf{D} = \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)$, we have the following set of equations:

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \frac{1}{\rho} \operatorname{Div} \boldsymbol{\sigma} + \mathbf{b}, \text{ with } \boldsymbol{\sigma} = \lambda \operatorname{tr}(\mathbf{D})\, \mathbf{I} + 2\mu \mathbf{D}$$

Since we have discretized the fluid mass into a finite number of fluid particles, the dynamics of this particle system is dictated by how each particle's position, velocity and density vary in time. Hence, the behaviour of our system is described by how $\frac{d\mathbf{x}}{dt}$, $\frac{D\rho}{Dt}$ and $\frac{D\mathbf{u}}{Dt}$ are determined for each particle along this particle's trajectory $\mathbf{x}(t)$. Thus, for particle i, we have:[16]

$$\begin{aligned}
&\frac{d\mathbf{x}_i}{dt} = \mathbf{u}_i && p_i = f(\rho_i) \\
&\frac{d\rho_i}{dt} = -\rho_i \langle \nabla \cdot \mathbf{u}_i \rangle && \mathbf{D}_i = \frac{1}{2}\left(\langle \nabla \mathbf{u}_i \rangle + \langle \nabla \mathbf{u}_i \rangle^T\right) \\
&\frac{d\mathbf{u}_i}{dt} = -\left\langle \frac{1}{\rho_i}\nabla p_i \right\rangle + \left\langle \frac{1}{\rho_i} \operatorname{Div} \boldsymbol{\sigma}_i \right\rangle + \langle \mathbf{b}_i \rangle && \boldsymbol{\sigma}_i = \lambda \operatorname{tr}(\mathbf{D}_i)\, \mathbf{I} + 2\mu \mathbf{D}_i
\end{aligned}$$

[15] Altought our simple solver implements a flow regime in which there are only stationary walls, the boundaries were enforced geometrically using a simple particle-wall collision response model similar to that presented in [Net07].

[16] Notice the change of notation from $\frac{D}{Dt}$ to $\frac{d}{dt}$, since we are talking about the rates of change of a particle's properties instead of derivatives of field quantities along trajectories.

where each spatial derivative is approximated by the symmetric rules we derived before (to ensure conformance with Newton's Third Law). Now, we present discretizations for each of the terms in the above equations.

The continuity equation

Since the equation of continuity is an expression which enforces the law of conservation of mass, we have the possibility of simply ignore it, in view of the explicit conservation of the total mass in our particle system [MCG03]. In this case, the particles' densities are computed at each time-step through the SPH approximation (derived by considering ρ as an ordinary scalar field):

$$\rho_i := \langle \rho(\mathbf{x}_i) \rangle = \sum_{j \in \mathcal{N}(\mathbf{x}_i)} m_j W_h(\mathbf{x}_i - \mathbf{x}_j)$$

Altought our implementation makes use of this expression for simplicity, some authors prefer to directly discretize the continuity equation to reuse certain computations and to not oversmooth the density field [Mon92, Mon94]. In this case, the discretization recommended is given by [Mon05, Net07]:

$$\frac{d\rho_i}{dt} = -\rho_i \langle \nabla \cdot \mathbf{u}_i \rangle = -\rho_i \sum_{j \in \mathcal{N}(\mathbf{x}_i)} \frac{m_j}{\rho_j} (\mathbf{u}_j - \mathbf{u}_i) \cdot \nabla W_h(\mathbf{x}_i - \mathbf{x}_j)$$

The momentum equation

The discretization of the momentum equation is performed on each term.

Pressure term. To compute a discretization for the pressure term, we make use of the symmetric rules derived in the later section:

$$\left\langle \frac{1}{\rho_i} \nabla p_i \right\rangle = \sum_{j \in \mathcal{N}(\mathbf{x}_i)} m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2} \right) \nabla W_h(\mathbf{x}_i - \mathbf{x}_j)$$

where the pressures p_k are given by the chosen equation of state $f(\rho_k)$.

Viscous term. The viscous term requires the computation of the deformation tensor $\mathbf{D}_i$, which is given by the numerically better rule:

$$\langle \nabla \mathbf{u}_i \rangle = \sum_{j \in \mathcal{N}(\mathbf{x}_i)} \frac{m_j}{\rho_j} (\mathbf{u}_j - \mathbf{u}_i) \otimes \nabla W_h (\mathbf{x}_i - \mathbf{x}_j)$$

$$\mathbf{D}_i = \frac{1}{2} \left(\langle \nabla \mathbf{u}_i \rangle + \langle \nabla \mathbf{u}_i \rangle^T \right), \text{ where } \mathbf{v} \otimes \mathbf{w} = \mathbf{v}\mathbf{w}^T.$$

With the deformation tensor calculated, the viscous term in computed by a rule from the same family used in the pressure term (adapted to tensors):

$$\left\langle \frac{1}{\rho_i} \operatorname{Div} \boldsymbol{\sigma}_i \right\rangle = \sum_{j \in \mathcal{N}(\mathbf{x}_i)} \frac{m_j}{\rho_j \rho_i} (\boldsymbol{\sigma}_j + \boldsymbol{\sigma}_i) \nabla W_h (\mathbf{x}_i - \mathbf{x}_j)$$

where $\boldsymbol{\sigma}_k = \lambda \operatorname{tr}(\mathbf{D}_k) \mathbf{I} + 2\mu \mathbf{D}_k$ as before. Notice that more general expressions for the viscous stress tensor $\boldsymbol{\sigma}$ could be adopted with this scheme and that no second-order derivatives are required. This approximation was used, with another $\boldsymbol{\sigma}$, to simulate non-Newtonian flows in [PPLT06] and [Net07].

Body forces. The forcing term in the momentum equation accounts for external body forces and, in an SPH implementation adopting repulsive ghost particles, the repulsive boundary forces as defined above. To compute $\mathbf{b}_i$, the discrete SPH approximation scheme can be applied directly, resulting in:

$$\langle \mathbf{b}_i \rangle = \sum_{j \in \mathcal{N}(\mathbf{x}_i)} \frac{m_j}{\rho_j} \mathbf{b}_j W_h (\mathbf{x}_i - \mathbf{x}_j)$$

This allows a force applied on a particle to spread to others inside its support.

Time-stepping

After discretizing the governing equations by SPH approximations, a coupled system of nonlinear ordinary differential equations result. The numerical integration of this system can be performed by employing any standard time-stepper designed for ODE's [PTVF92, SB02, Ise96, AP98, LeV07].

In our implementation, we employed a simple explicit symplectic Euler method [SD06, KYT+06]. The integration rule of this method is very similar to the classical forward Euler scheme, applied to the ODE, $\ddot{\mathbf{x}} = f(\mathbf{x})$:

forward Euler scheme:	*symplectic Euler scheme:*
$\mathbf{v}^{n+1} = \mathbf{v}^n + kf\left(\mathbf{x}^n\right)$ $\mathbf{x}^{n+1} = \mathbf{x}^n + k\mathbf{v}^n$	$\mathbf{v}^{n+1} = \mathbf{v}^n + kf\left(\mathbf{x}^n\right)$ $\mathbf{x}^{n+1} = \mathbf{x}^n + k\mathbf{v}^{n+1}$

where $\ddot{\mathbf{x}} = f\left(\mathbf{x}\right)$ was transformed to the first-order ODE, $\begin{pmatrix}\dot{\mathbf{x}}\\ \dot{\mathbf{v}}\end{pmatrix} = \begin{pmatrix}\mathbf{v}\\ f\left(\mathbf{x}\right)\end{pmatrix}$.

A note is required regarding the choice of the time step k. In an implementation of SPH with more than didatic goals, the integration of the discretized governing equations by an explicit method during a time k is performed by a sequence of iterations with smaller time steps in such a way to guarantee that the simulation doesn't blow up. This means that, after each substep, a new time step $\hat{k}$ is chosen in such a manner that some stability condition is satisfied (e.g. the *Courant-Friedrichs-Lewy* (CFL) condition for PDE's [CFL67]). The form in which these conditions restrict the time step used in each substep is dependent of the discretization performed. Stability conditions for discretizations similar to ours can be found in [MFZ97, Net07]. Since we don't choose our substeps by a stability condition in our implementation, our time step k has to be hand-tuned for every simulation performed.

4.2.4 Experiments

As an illustration for our description of a simple *Smoothed Particle Hydrodynamics* scheme, we implemented a simple fluid simulator in the `C` language for the governing equations as described in 4.2.1. In this subsection, we present some results generated with that solver and some issues which appeared during its development. Appendix B presents the core code of our simple *SPH* solver as a more concrete example of the method described.

A simple breaking dam. Figure 4.4 presents six frames from a simple two-dimensional breaking dam animation. The fluid is confined inside a square bounded domain, in which a vertical wall is fixed at the middle of the ceiling in this chamber going down to $\frac{1}{8}$ of the chamber's height above its floor. The simulation is then initialized with half the left-side of the chamber filled with a fluid under the action of gravity. Those frames were captured little after the simulation began and are spaced roughly regularly in time (they are intended just to illustrate the resulting animation).

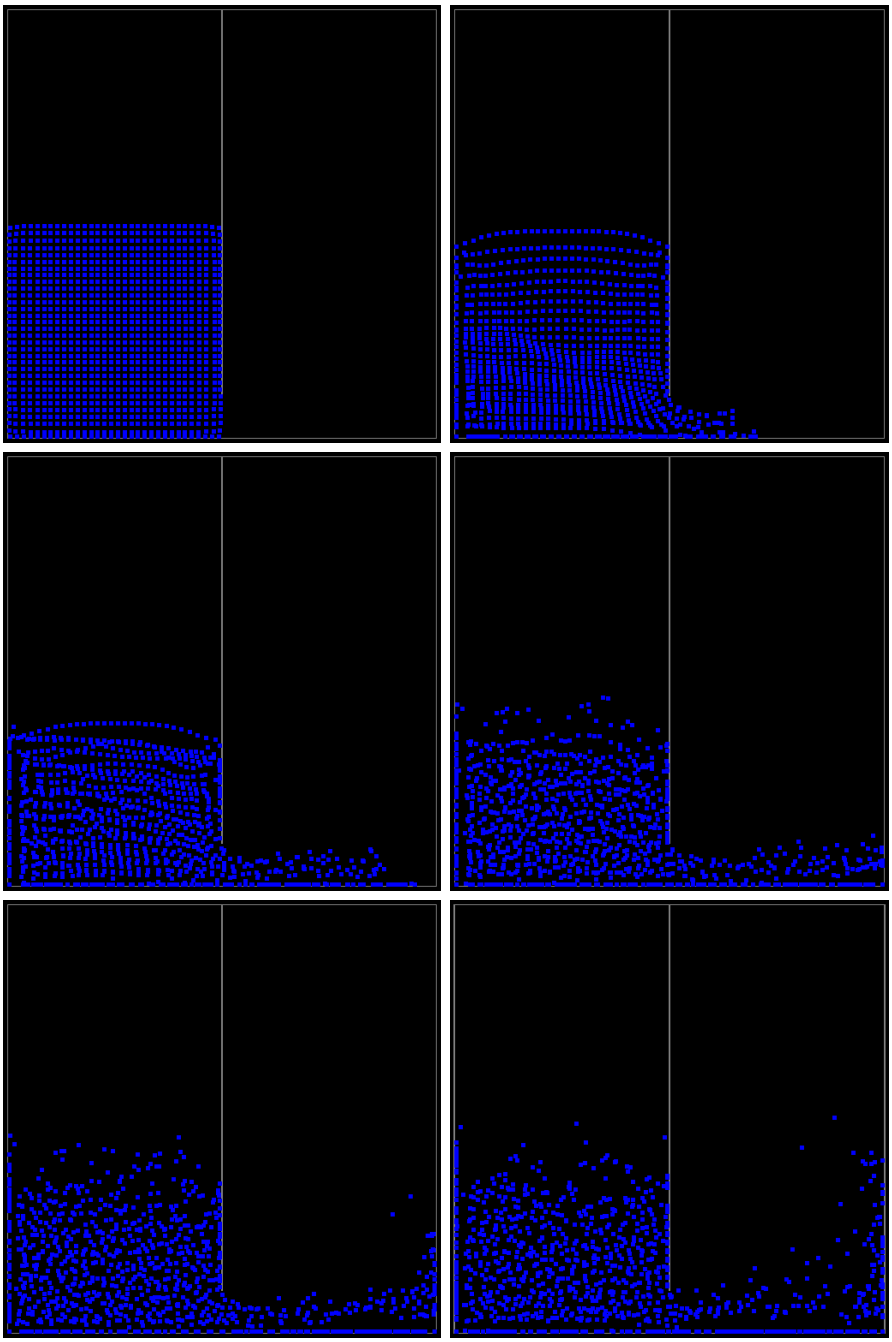

Figure 4.4: Frames from an animation of a breaking dam (1024 particles).

To avoid excessive compression and blow up of the simulation, we needed to take very small time steps. Combined with the fact that we did not use any data structure to accelerate neighbor queries, the simulation ran annoyingly slowly, at least compared to our simple *Stable Fluids* implementation, even using just 1024 fluid particles.

Implementation issues. As before, we experienced some problems in dealing with boundaries. We first implemented the ghost particle approach outlined in 4.2.2 but they introduced too much stiffness in the resulting ODE's, which required time steps prohibitively small for interactive simulation and avoiding blow up. For this reason, we changed paradigm and employed a collision response scheme similar to those used in rigid body animations [Net07]. This change allowed us to take time steps large enough to keep interactivity without blow up. Improvements in stability can be made using particle-dependent time steps by "freezing" the acceleration field during the time step and evolving each particle independently using the interpolated accelerations. Such an approach is adaptive both in time and space and improvements are reported in the simulation time complexity [HK89].

4.3 Comments and Further Reading

In this chapter, we presented the two most basic and commonly used methods to simulate the governing equations of fluid flows in computer graphics. Although they already produce very nice (and physically-plausible) results, many other methods and extensions have been developed to overcome problems and limitations encountered in their basic formulation.

Stable Fluids and variants. The stable fluids method was introduced by Stam [Sta99] as an alternative to the work of Foster and Metaxas [FM96a] better suited to computer graphics, because of the stringent restrictions on the time step imposed by the techniques they used to numerically solve the Navier-Stokes equations, which was strongly based on the seminal work of Harlow and Welch [HW65], that introduced both the *Marker And Cell* method and the use of a staggered grid to represent the velocity field. After [Sta99], some works proposed modifications to the basic stable fluids scheme in order to reduce effects of vortical diffusion [FSJ01, SRF05], mass dissipation in liquid simulation [FF01, EMF02] and to speed-up pressure projection

[LGF04]. Other works proposed extensions to the basic approach (and to other extensions) to be able to animate fire [NFJ02, RNGF03], highly viscous and viscoelastic fluids [CMIT02, GBO04], flows on surfaces [Sta03], coupling of fluids and solids [CMT04, GSLF05, CGFO06, BBB07], hybrid, deforming and dynamic meshes [FOK05, FOKG05, KFCO06] and GPU implementations [Har04, CLT07, Bor06] (just to name a *few*).

Smoothed Particle Hydrodynamics and variants. SPH was proposed independently by Gingold and Monaghan [GM77] and Lucy [Luc77] in 1977 with the purpose of simulating stellar flows in astronomy and astrophysics. Since then, much work has been done in improving the method both for its original applications in astrophysics and to other fields [Mon92, Mon05]. In computer graphics, it was introduced by Stam to depict gaseous phenomena [SF95] and used later by Desbrun and Cani to simulate deformable solids [DG96]. However, it was just after the work of Müller et al. [MCG03] on interactive simulation of liquids that SPH has caught widespread attention from the computer graphics community. Since its introduction, many authors have been exploting the SPH methodology to animate systems as complex as: weakly compressible fluids [BT07], non-Newtonian fluids [PPLT06, Net07], bubbling and frothing [CPPK07], coupling of solids and fluids [KAG+05], adaptive fluids [APKG07], fluids in GPU's [Nak07], elastic, plastic and melting objects [MKN+04] and even hair [HMT01, BCN03].

Other methods. The computational fluid dynamics community is very prolific, this may be a reason for the many different techniques recently introduced into the computer animation community to simulate fluid flows, a few of them are [SCC04, SCC05, ZB05, ETK+07, WBOL07, BWHT07, GN07, GLG95, PK05, CACC06, AN05, ANSN06].

Chapter 5

Final Remarks

After an overview of the fundamentals of mathematical fluid dynamics employed in the simulation of fluid flows for computer graphics, and the basic methods used by the CG community, we have provided some of the foundations needed by the interested (and mathematically educated) reader to pursue a more in-depth understanding of the techniques which have been published in the computer animation literature regarding the animation of fluid motion. As a complement to this introductory text, we provided some guidance for further reading of relevant references. In the following, we present some efforts which have been (and, in near future, are intended to be) made.

Work in Progress. Both our implementations of *Stable Fluids* and *SPH* are still very basic. Currently, we are working on an extension of these solvers to handle *moving boundaries* (*internal* and *external*) [CMT04], *high-viscosity* and *non-Newtonian fluids* [CMIT02, PPLT06], *liquids* (using *particle level-sets* in stable fluids and *contouring* in rendering SPH simulations) [FF01, EMF02, LLVT03], and on optimizing the SPH code with data-structures for *hierarchical range queries* [HK89]. Since we are also interested in simulating deformable objects, we are currently studying the foundations of *mathematical elasticity* [MH94, Lov44, Pat87] and the recent works on *discrete differential geometry* [Gri06], which have found interesting applications in the simulation of fluids [ETK+07].

Future Work. In (a not so far) future, we intend to experiment with recently developed (and promising) techniques to animate fluids for computer graphics. These include the work of Elcott et al. [ETK+07], which applies the

tools of discrete differential geometry to stably simulate fluids avoiding the "correction" procedure that projects an intermediary vector field onto its divergence-free part; the dynamic/adaptive remeshing method of Klingner et al. [KFCO06] which conforms the mesh to fluid features attaining detailed simulations without over-subdividing the computational domain (and impacting performance); the variational/Eulerian approach of Mullen et al. [MMTD07] to represent and advect field quantities; the hierarchical run-lenght encoded level-set structure of Houston et al. [HNB$^+$06] to compactly represent highly detailed level-sets; and the lattice Boltzmann method employed by Thuerey [Thu07] in the simulation of fluid motion by means of cellular automata.

Some Ideas and more Future Work. During the preparation of this text, some ideas have shown up as interesting avenues for future research. We believe that proposition 1 — describing the local behaviour of the fluid flow — might be exploited to design some criteria for adaptive subdivision dynamic Eulerian simulations [KFCO06], adaptively sampled particle-based simulations [APKG07], and anisotropic rendering of the free-surface in liquid/fire animations [AA06] (possibly avoiding the blobby artifacts common when using few particles). Another possibility regards the design of velocity/vector fields [AN05, ANSN06] as a "cheap" alternative to a full simulation of the Navier-Stokes equations. Maybe an intuitive modeling metaphor can be provided by exploiting the concepts involved in the definitions of vortex lines/sheets/tubes and in theorem 4 (Helmholtz's Theorem). Recently, a model-reduced simulation method has been introduced to computer graphics for real-time fluid simulation [TLP06], we believe this technique (or a similar one using machine learning algorithms other than principal component analysis) could be exploited to perform faster optimization-based fluid control methods [TMPS03, MTPS04]. Another intesting possibility, would be to evaluate the feasibility of employing domain decomposition methods (spatial splitting methods) [Bor07] to parallelize computations, both in distributed systems and GPU's. At last, further study of the variational principles underlying fluid motions might be used to perform numerical simulations which better preserve invariants of flows (e.g. kinetic energy in unforced ideal fluids), some recent works have benefited from an understanding of such variational principles [BBB07, KYT$^+$06] and we think these properties could be more exploited in the design of better solvers [LR04].

Appendix A

A Simple *Stable Fluids* Solver

In this appendix, we present our implementation of the basic *Stable Fluids* scheme as described in [Sta99]. This code also contains a simple implementation of the *vorticity confinement* forces as introduced in [FSJ01]. We stress that this code is ***neither*** optimized for performance ***nor*** memory consumption, it was written for **didatic** reasons (i.e. designed to be *almost* readable).

```
#include <math.h>
#include <string.h>

#define INDEX(i,j)      ((i)+((j)*(N+2)))
#define FOR_EACH_CELL   for(i=1;i<=N;++i){for(j=1;j<=N;++j){
#define END_FOR_EACH    }}
#define MAX_ITERATIONS  25
#define CLAMP(x,m,M)    (((x)<(m))?(m):(((x)>(M))?(M):(x)))
#define EPSILON         (1e-12)

typedef enum boundary_e {
  DIRICHLET,
  NEUMANN
} boundary_t;

typedef enum boolean_e {
  FALSE = 0,
  TRUE  = (!0)
} boolean_t;
```

```
void setBoundaryCondition( boundary_t bx, boundary_t by,
    unsigned int N, float* f ) {
  register unsigned int i;

  for ( i = 1 ; i <= N ; ++i ) {
    f[INDEX(  0,  i)]=((bx==DIRICHLET)?-f[INDEX(1,i)]:f[INDEX(1,i)]);
    f[INDEX(N+1,  i)]=((bx==DIRICHLET)?-f[INDEX(N,i)]:f[INDEX(N,i)]);
    f[INDEX(  i,  0)]=((by==DIRICHLET)?-f[INDEX(i,1)]:f[INDEX(i,1)]);
    f[INDEX(  i,N+1)]=((by==DIRICHLET)?-f[INDEX(i,N)]:f[INDEX(i,N)]);
  }

  f[INDEX(  0,  0)] = 0.5f * (f[INDEX(  0,1)] + f[INDEX(1,  0)]);
  f[INDEX(N+1,  0)] = 0.5f * (f[INDEX(N+1,1)] + f[INDEX(N,  0)]);
  f[INDEX(  0,N+1)] = 0.5f * (f[INDEX(  0,N)] + f[INDEX(1,N+1)]);
  f[INDEX(N+1,N+1)] = 0.5f * (f[INDEX(N+1,N)] + f[INDEX(N,N+1)]);
}

/*
 * Solves (\alpha I - \beta\Delta)u = f on the unit square,
 * with boundary conditions 'b' using the Gauss-Seidel iteration
 */
void solve( boundary_t bx, boundary_t by, float alpha, float beta,
    unsigned int N, float* f, float* u ) {
  register unsigned int i, j, k;
  float n = beta * N * N;
  float d = alpha + 4.0f * n;
  float left, right, down, up;

  for ( k = 0 ; k < MAX_ITERATIONS ; ++k ) {
    FOR_EACH_CELL
      left  = u[INDEX(i-1,  j)];
      right = u[INDEX(i+1,  j)];
      down  = u[INDEX(  i,j-1)];
      up    = u[INDEX(  i,j+1)];
      u[INDEX(i,j)] = (f[INDEX(i,j)]+n*(left+right+down+up))/d;
    END_FOR_EACH

    setBoundaryCondition( bx, by, N, u );
  }

}
```

```
void advect( boolean_t advectAndForce, boundary_t bx, boundary_t by,
    float dt, unsigned int N, float* u, float* v, float* f0, float* f ) {
  register unsigned int i, j;
  float x0, y0, x, y, xi, yj, fxi, fyj, cxi, cyj, h = 1.0f/N;
  float s, t, ld, rd, lu, ru, c;

  FOR_EACH_CELL
    x0 = (i - 0.5f) * h;
    y0 = (j - 0.5f) * h;

    x = x0 - dt * u[INDEX(i,j)];
    y = y0 - dt * v[INDEX(i,j)];

    x = CLAMP(x,0.0f,1.0f);
    y = CLAMP(y,0.0f,1.0f);

    xi = x * N + 0.5f;
    yj = y * N + 0.5f;

    fxi = floor(xi);
    cxi = ceil(xi);

    if ( fxi != cxi ) {
      s = (xi - fxi) / (cxi - fxi);
    } else {
      s = 0.0f;
    }

    fyj = floor(yj);
    cyj = ceil(yj);

    if ( fyj != cyj ) {
      t = (yj - fyj) / (cyj - fyj);
    } else {
      t = 0.0f;
    }

    ld = f0[INDEX((int)fxi,(int)fyj)];
    rd = f0[INDEX((int)cxi,(int)fyj)];
    lu = f0[INDEX((int)fxi,(int)cyj)];
    ru = f0[INDEX((int)cxi,(int)cyj)];

    c = ((1.0f-t)*((1.0f-s)*ld+s*rd)+t*((1.0f-s)*lu+s*ru));
```

```
    if ( advectAndForce == TRUE ) {
      f[INDEX(i,j)] *= dt;
      f[INDEX(i,j)] += c ;
    } else {
      f[INDEX(i,j)]  = c;
    }

  END_FOR_EACH

  setBoundaryCondition( bx, by, N, f );
}

void force( boundary_t bx, boundary_t by, float dt,
    unsigned int N, float* f, float* u ) {
  register unsigned int i, j;

  FOR_EACH_CELL
    u[INDEX(i,j)] += dt * f[INDEX(i,j)];
  END_FOR_EACH

  setBoundaryCondition( bx, by, N, u );
}

void confineVorticity ( boundary_t ubx, boundary_t uby,
    boundary_t vbx, boundary_t vby, float dt, float epsilon,
    unsigned int N, float* u, float* v,
    float* hOmega, float* hAbsOmega ) {

  if ( epsilon < EPSILON ) {
    return;
  }

  register unsigned int i, j;
  float left, right, down, up;
  float dx, dy, norm, alpha = dt * epsilon;

  FOR_EACH_CELL
    left  = v[INDEX(i-1,  j)];
    right = v[INDEX(i+1,  j)];
    down  = u[INDEX(  i,j-1)];
    up    = u[INDEX(  i,j+1)];
    hOmega[INDEX(i,j)]    = ( right - left ) - ( up - down );
    hAbsOmega[INDEX(i,j)] = (float) fabs( hOmega[INDEX(i,j)] );
  END_FOR_EACH
```

```
  setBoundaryCondition( NEUMANN, NEUMANN, N,    hOmega );
  setBoundaryCondition( NEUMANN, NEUMANN, N, hAbsOmega );

  FOR_EACH_CELL
    left  = hAbsOmega[INDEX(i-1,  j)];
    right = hAbsOmega[INDEX(i+1,  j)];
    down  = hAbsOmega[INDEX(  i,j-1)];
    up    = hAbsOmega[INDEX(  i,j+1)];
    dx    = right - left;
    dy    =    up - down;
    norm  = sqrt( dx * dx + dy * dy );

    if ( norm > EPSILON ) {
      u[INDEX(i,j)] += alpha * hOmega[INDEX(i,j)] *   dy  / norm;
      v[INDEX(i,j)] += alpha * hOmega[INDEX(i,j)] * (-dx) / norm;
    }

  END_FOR_EACH

  setBoundaryCondition( ubx, uby, N, u );
  setBoundaryCondition( vbx, vby, N, v );
}

void diffuse( boundary_t bx, boundary_t by, float nu, float dt,
    unsigned int N, float* u0, float* u ) {
  solve( bx, by, 1.0f, nu * dt, N, u0, u );
}

/*
 *  Solve -\Delta\phi = \nabla\cdot U, using Neumann boundary conditions
 *  for \phi, and assigns U <- U + \nabla\phi (U <- IP(U))
 */
void project( unsigned int N, float* u, float* v, float* divU, float* phi ) {
  register unsigned int i, j;
  float n = 0.5f * N;
  float left, right, down, up;

  FOR_EACH_CELL
    left  = u[INDEX(i-1,  j)];
    right = u[INDEX(i+1,  j)];
    down  = v[INDEX(  i,j-1)];
    up    = v[INDEX(  i,j+1)];
    divU[INDEX(i,j)] = n * ( ( right - left ) + ( up - down ) );
    phi[INDEX(i,j)]  = 0.0f;
  END_FOR_EACH
```

```
  setBoundaryCondition( NEUMANN, NEUMANN, N, divU );
  setBoundaryCondition( NEUMANN, NEUMANN, N,  phi );

  solve( NEUMANN, NEUMANN, 0.0f, 1.0f, N, divU, phi );

  FOR_EACH_CELL
    left  = phi[INDEX(i-1,  j)];
    right = phi[INDEX(i+1,  j)];
    down  = phi[INDEX(  i,j-1)];
    up    = phi[INDEX(  i,j+1)];
    u[INDEX(i,j)] += n * ( right - left );
    v[INDEX(i,j)] += n * ( up    - down );
  END_FOR_EACH

  setBoundaryCondition( DIRICHLET,   NEUMANN, N, u );
  setBoundaryCondition(   NEUMANN, DIRICHLET, N, v );
}

void stepNavierStokes( float dt, float nu, float epsilon,
    unsigned int N, float* bx, float* by, float* u, float* v ) {
  /*
   * Advect -> Force -> Diffuse -> Project
   */
  advect( TRUE, DIRICHLET, DIRICHLET, dt, N, u, v, u, bx );
  advect( TRUE, DIRICHLET, DIRICHLET, dt, N, u, v, v, by );
  confineVorticity( DIRICHLET, DIRICHLET, DIRICHLET, DIRICHLET,
                    dt, epsilon, N, bx, by, u, v );
  diffuse( DIRICHLET, DIRICHLET, nu, dt, N, bx, u );
  diffuse( DIRICHLET, DIRICHLET, nu, dt, N, by, v );
  project( N, u, v, bx, by );
  setBoundaryCondition( DIRICHLET, DIRICHLET, N, u );
  setBoundaryCondition( DIRICHLET, DIRICHLET, N, v );
}

void stepScalarField( float dt, float kappa, unsigned int N,
    float* u, float* v, float* f, float* rho ) {
  /*
   * Force -> Advect -> Diffuse
   */
  force( NEUMANN, NEUMANN, dt, N, f, rho );
  advect( FALSE, NEUMANN, NEUMANN, dt, N, u, v, rho, f );
  diffuse( NEUMANN, NEUMANN, kappa, dt, N, f, rho );
}
```

```
void stepEuler( float dt, float epsilon, unsigned int N,
    float* bx, float* by, float* u, float* v ) {
  /*
   * Advect -> Force -> Project
   */
  advect( TRUE, DIRICHLET,   NEUMANN, dt, N, u, v, u, bx );
  advect( TRUE,   NEUMANN, DIRICHLET, dt, N, u, v, v, by );
  confineVorticity( DIRICHLET, NEUMANN, NEUMANN, DIRICHLET,
                    dt, epsilon, N, bx, by, u, v );
  project( N, bx, by, u, v );
  memcpy( u, bx, (N + 2) * (N + 2) * sizeof(float) );
  memcpy( v, by, (N + 2) * (N + 2) * sizeof(float) );
}
```

"I think that the most important result of the computer graphics revolution is that it has helped heal the gulf between art and science."
Jim Blinn

Appendix B

A Simple *Smoothed Particle Hydrodynamics* Solver

In this appendix, we present our implementation of a **simple** *SPH* scheme as outlined in section 4.2. We stress that this code is ***neither*** optimized for performance ***nor*** memory consumption, like our *Stable Fluids* solver, it was written for the purpose of **illustrating** our considerations in chapter 4.

```
#include <math.h>
#include <string.h>

#define FOR_EACH_PARTICLE       for(i=0;i<N;++i){
#define END_FOR_EACH_PARTICLE   }

#define FOR_EACH_PAIR       for(i=0;i<(N-1);++i){for(j=i+1;j<N;++j){
#define END_FOR_EACH_PAIR   }}

float W(float h, float dx, float dy) {
  static const float r2pi = 0.15915494309189535f;
  float rh2 = 1.0f / ( h * h );
  return r2pi * rh2 * exp( - 0.5f * rh2 * ( dx * dx + dy * dy ) );
}

void gradW(float h, float dx, float dy, float* gradWx, float* gradWy) {
  float scale  = - W(h, dx, dy) / ( h * h );
  *gradWx = scale * dx;
  *gradWy = scale * dy;
}
```

```
void wGradW(float h, float dx, float dy,
    float* w, float* gradWx, float* gradWy) {
  *w = W(h, dx, dy);
  float scale  = - (*w) / ( h * h );
  *gradWx = scale * dx;
  *gradWy = scale * dy;
}

void computeDensities( float h, float m, unsigned int N,
    float* x, float* y, float* rho ) {
  register unsigned int i, j;
  float Wh = W(h, 0.0f, 0.0f);

  FOR_EACH_PARTICLE
    rho[i] = Wh;
  END_FOR_EACH_PARTICLE

  FOR_EACH_PAIR
    Wh = W(h, x[i]-x[j], y[i]-y[j]);
    rho[i] += Wh;
    rho[j] += Wh;
  END_FOR_EACH_PAIR

  FOR_EACH_PARTICLE
    rho[i] *= m;
  END_FOR_EACH_PARTICLE
}

void computeStressTensors( float h, float m, float lambda, float mu,
    unsigned int N, float* rho, float* x, float* y, float* u, float* v,
    float* s11, float* s12, float* s22 ) {
  register unsigned int i, j;
  float twomu = mu+mu, gradWx, gradWy, du, dv;
  float d11, twod12, d22, lambdaTrD;
  float s11rho, s12rho, s22rho, rrho;

  memset(s11, 0, N * sizeof(float));
  memset(s12, 0, N * sizeof(float));
  memset(s22, 0, N * sizeof(float));

  FOR_EACH_PAIR
    gradW(h, x[i]-x[j], y[i]-y[j], &gradWx, &gradWy);

    du = u[j]-u[i];
    dv = v[j]-v[i];
```

```
    d11       = du * gradWx;
    twod12    = du * gradWy + dv * gradWx;
    d22       = dv * gradWy;
    lambdaTrD = lambda * ( d11 + d22 );

    s11rho = lambdaTrD + twomu * d11   ;
    s12rho =                mu * twod12;
    s22rho = lambdaTrD + twomu * d22   ;

    rrho    = 1.0f / rho[j];
    s11[i] += s11rho * rrho;
    s12[i] += s12rho * rrho;
    s22[i] += s22rho * rrho;

    rrho    = 1.0f / rho[i];
    s11[j] += s11rho * rrho;
    s12[j] += s12rho * rrho;
    s22[j] += s22rho * rrho;
  END_FOR_EACH_PAIR

  FOR_EACH_PARTICLE
    s11[i] *= m;
    s12[i] *= m;
    s22[i] *= m;
  END_FOR_EACH_PARTICLE
}

void addAccelerations( float h, float m, float B, float n, float rho0,
    unsigned int N, float* rho, float* x, float* y,
    float* s11, float* s12, float* s22, float* f, float* g ) {
  register unsigned int i, j;
  float Wh, gradWx, gradWy, rrhoi, rrhoj, pi, pj, scale, a, b;

  FOR_EACH_PAIR
    wGradW(h, x[i]-x[j], y[i]-y[j], &Wh, &gradWx, &gradWy);

    rrhoi = 1.0f / rho[i];
    rrhoj = 1.0f / rho[j];

    /* Equation of state */
    pi = (1.0f+B)*pow(rho[i]/rho0,n)-B;
    pj = (1.0f+B)*pow(rho[j]/rho0,n)-B;
```

```
    scale = m * ( pi * rrhoi * rrhoi + pj * rrhoj * rrhoj );

    a = - scale * gradWx;
    b = - scale * gradWy;

    f[i] += a; f[j] += -a;
    g[i] += b; g[j] += -b;

    scale = m * rrhoi * rrhoj;
    a = scale * ((s11[i]+s11[j])*gradWx + (s12[i]+s12[j])*gradWy);
    b = scale * ((s12[i]+s12[j])*gradWx + (s22[i]+s22[j])*gradWy);

    f[i] += a; f[j] += -a;
    g[i] += b; g[j] += -b;
  END_FOR_EACH_PAIR
}

void enforceBoundaryConditions( float dt, float epsilon, float h,
    float* x, float* y, float* u, float* v ) {
  float x0 = (*x) - dt * (*u);
  float y0 = (*y) - dt * (*v);
  float gap = 0.75f * h;

  if ( (*x) < 0.0f+gap ) {
    *x = 0.0f+gap;
    *u = -epsilon * ((*x) - x0) / dt;
  }

  if ( 1.0f-gap < (*x) ) {
    *x = 1.0f-gap;
    *u = -epsilon * ((*x) - x0) / dt;
  }

  if ( (*y) < 0.0f+gap ) {
    *y = 0.0f+gap;
    *v = -epsilon * epsilon * ((*y) - y0) / dt;
  }

  if ( 1.0f-gap < (*y) ) {
    *y = 1.0f-gap;
    *v = -epsilon * epsilon * ((*y) - y0) / dt;
  }
```

```
  if ( x0 <= 0.5f && 0.5f < *x+gap && 0.125f < y0+gap ) {
    *x = 0.5f-gap;
    *u = -epsilon * ((*x) - x0) / dt;
  } else if ( 0.5f <= x0 && *x-gap < 0.5f && 0.125f < y0+gap ) {
    *x = 0.5f+gap;
    *u = -epsilon * ((*x) - x0) / dt;
  }

}

void step( float dt, float h, float m, float lambda, float mu, float epsilon,
    float B, float n, float rho0, unsigned int N, float* rho,
    float* x, float* y, float* u, float* v, float* f, float* g,
    float* s11, float* s12, float* s22 ) {
  computeDensities( h, m, N, x, y, rho );
  computeStressTensors( h, m, lambda, mu, N, rho,
                        x, y, u, v, s11, s12, s22 );
  addAccelerations( h, m, B, n, rho0, N, rho,
                    x, y, s11, s12, s22, f, g );

  register unsigned int i;

  FOR_EACH_PARTICLE
    u[i] += dt * f[i];
    v[i] += dt * g[i];

    x[i] += dt * u[i];
    y[i] += dt * v[i];

    enforceBoundaryConditions( dt, epsilon, h, &x[i], &y[i], &u[i], &v[i] );
  END_FOR_EACH_PARTICLE
}
```

"Beware of bugs in the above code;
I have only proved it correct, not tried it."
Donald Ervin Knuth

Bibliography

[AA06] Anders Adamson and Marc Alexa. Anisotropic point set surfaces. In *Afrigaph '06: Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 7–13, New York, NY, USA, 2006. ACM.

[AH70] A. A. Amsden and F. H. Harlow. The smac method: A numerical technique for calculating incompressible fluid flows. *The SMAC Method: A Numerical Technique for Calculating Incompressible Fluid Flows*, 1970.

[AN05] Alexis Angelidis and Fabrice Neyret. Simulation of smoke based on vortex filament primitives. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 87–96, New York, NY, USA, 2005. ACM Press.

[ANSN06] Alexis Angelidis, Fabrice Neyret, Karan Singh, and Derek Nowrouzezahrai. A controllable, fast and stable basis for vortex based smoke simulation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 25–32, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[AP98] Uri M. Ascher and Linda R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998.

[APKG07] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. Adaptively sampled particle fluids. In *SIGGRAPH '07:*

ACM SIGGRAPH 2007 papers, page 48, New York, NY, USA, 2007. ACM Press.

[Bat99] G. K. Batchelor. *An introduction to fluid dynamics*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, paperback edition, 1999.

[BBB07] Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph.*, 26(3):100, 2007.

[BBC+94] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[BCN03] Yosuke Bando, Bing-Yu Chen, and Tomoyuki Nishita. Animating hair with loosely connected particles. *Computer Graphics Forum*, 22(3):411–418, 2003.

[BHW96] Ronen Barzel, John F. Hughes, and Daniel Wood. Plausible motion simulation for computer animation. In *EGCAS '96: Seventh International Workshop on Computer Animation and Simulation*, 1996.

[BMF07] Robert Bridson and Matthias Müller-Fischer. Fluid simulation. In *SIGGRAPH'07: ACM SIGGRAPH 2007 Courses*, pages 1–81, New York, NY, USA, 2007. ACM Press.

[Bor06] Alex Laier Bordignon. Navier-stokes em gpu. Master's thesis, Pontifícia Universidade Católica do Rio de Janeiro, 2006.

[Bor07] Carlos E. C. Borges. Coarse grid correction operator splitting for parabolic partial differential equations. Master's thesis, Instituto Nacional de Matemática Pura e Aplicada, 2007.

[BT07] Markus Becker and Matthias Teschner. Weakly compressible sph for free surface flows. In *SCA '07: Proceedings of the 2007*

ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 209–217, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[BWHT07] Adam W. Bargteil, Chris Wojtán, Jessica K. Hodgins, and Greg Turk. A finite element method for animating large viscoplastic flow. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 16, New York, NY, USA, 2007. ACM Press.

[CACC06] Mathieu Coquerelle, Jeremie Allard, Georges-Henri Cottet, and Marie-Paule Cani. A vortex method for bi-phasic fluids interacting with rigid bodies, 2006. Available at http://arxiv.org/abs/math/0607597.

[CdVL95] Jim X. Chen and Niels da Vitoria Lobo. Toward interactive-rate simulation of fluids with moving obstacles using navier-stokes equations. *Graph. Models Image Process.*, 57(2):107–116, 1995.

[CF76] R. Courant and K. O. Friedrichs. *Supersonic flow and shock waves*. Springer-Verlag, New York, 1976. Reprinting of the 1948 original, Applied Mathematical Sciences, Vol. 21.

[CFL67] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM J. Res. Develop.*, 11:215–234, 1967.

[CFL+07] Nuttapong Chentanez, Bryan E. Feldman, François Labelle, James F. O'Brien, and Jonathan R. Shewchuk. Liquid simulation on lattice-based tetrahedral meshes. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 219–228, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[CGFO06] Nuttapong Chentanez, Tolga G. Goktekin, Bryan E. Feldman, and James F. O'Brien. Simultaneous coupling of fluids and deformable bodies. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 83–89, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[Cho68] Alexandre Joel Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.

[CLT07] Keenan Crane, Ignacio Llamas, and Sarah Tariq. *GPU Gems 3 – Chapter 30: Real-Time Simulation and Rendering of 3D Fluids*. Addison-Wesley Professional, 2007.

[CM93] Alexandre J. Chorin and Jerrold E. Marsden. *A mathematical introduction to fluid mechanics*, volume 4 of *Texts in Applied Mathematics*. Springer-Verlag, New York, third edition, 1993.

[CMIT02] Mark Carlson, Peter J. Mucha, R. Brooks Van Horn III, and Greg Turk. Melting and flowing. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 167–174, New York, NY, USA, 2002. ACM Press.

[CMT04] Mark Carlson, Peter J. Mucha, and Greg Turk. Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph.*, 23(3):377–384, 2004.

[CPPK07] Paul W. Cleary, Soon Hyoung Pyo, Mahesh Prakash, and Bon Ki Koo. Bubbling and frothing liquids. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 97, New York, NY, USA, 2007. ACM Press.

[DEF+04] Oliver Deusen, David S. Ebert, Ron Fedkiw, F. Kenton Musgrave, Przemyslaw Prusinkiewicz, Doug Roble, Jos Stam, and Jerry Tessendorf. The elements of nature: interactive and realistic techniques. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes*, page 32, New York, NY, USA, 2004. ACM Press.

[Dem97] James W. Demmel. *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[DG96] M. Desbrun and M. P. Gascuel. Smoothed particles: a new paradigm for animating highly deformable bodies. In *EGCAS '96: Seventh International Workshop on Computer Animation and Simulation*, 1996.

[EMF02] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 736–744, New York, NY, USA, 2002. ACM Press.

[ETK+07] Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph.*, 26(1):4, 2007.

[Eva98] Lawrence C. Evans. *Partial differential equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 1998.

[FF01] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 23–30, August 2001.

[FL04] Raanan Fattal and Dani Lischinski. Target-driven smoke animation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 441–448, New York, NY, USA, 2004. ACM.

[FM96a] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. In *GI '96: Proceedings of the conference on Graphics interface '96*, pages 204–212, Toronto, Ont., Canada, Canada, 1996. Canadian Information Processing Society.

[FM96b] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.

[FM97a] Nick Foster and Dimitris Metaxas. Controlling fluid animation. In *CGI '97: Proceedings of the 1997 Conference on Computer Graphics International*, page 178, Washington, DC, USA, 1997. IEEE Computer Society.

[FM97b] Nick Foster and Dimitris Metaxas. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 181–188, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[FM00] Nick Foster and Dimitris Metaxas. Modeling water for computer animation. *Communications of the ACM*, 43(7):60–67, 2000.

[FOA03] Bryan E. Feldman, James F. O'Brien, and Okan Arikan. Animating suspended particle explosions. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 708–715, New York, NY, USA, 2003. ACM.

[FOK05] Bryan E. Feldman, James F. O'Brien, and Bryan M. Klingner. Animating gases with hybrid meshes. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 904–909, New York, NY, USA, 2005. ACM Press.

[FOKG05] Bryan E. Feldman, James F. O'Brien, Bryan M. Klingner, and Tolga G. Goktekin. Fluids in deforming meshes. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 255–259, New York, NY, USA, 2005. ACM Press.

[FSJ01] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22, New York, NY, USA, 2001. ACM Press.

[GBO04] Tolga G. Goktekin, Adam W. Bargteil, and James F. O'Brien. A method for animating viscoelastic fluids. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 463–468, New York, NY, USA, 2004. ACM Press.

[GLG95] Manuel Noronha Gamito, Pedro Faria Lopes, and Mário Rui Gomes. Two-dimensional simulation of gaseous phenomena using vortex particles. In Demetri Terzopoulos and Daniel Thalmann, editors, *Computer Animation and Simulation '95*, pages 2–15. Eurographics, Springer-Verlag, September 1995.

[GM77] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics - Theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, November 1977.

[GN07] Mohit Gupta and Srinivasa G. Narasimhan. Legendre fluids: a unified framework for analytic reduced space modeling and rendering of participating media. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 17–25, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[Gri06] Eitan Grinspun. Discrete differential geometry: An applied introduction. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, New York, NY, USA, 2006. ACM.

[GSLF05] Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph.*, 24(3):973–981, 2005.

[Har04] Mark Harris. *GPU Gems – Chapter 38: Fast Fluid Dynamics Simulation on the GPU*. Addison-Wesley Professional, 2004.

[HCSL02] Mark J. Harris, Greg Coombe, Thorsten Scheuermann, and Anselmo Lastra. Physically-based visual simulation on graphics hardware. In *Graphics Hardware 2002*, pages 109–118, September 2002.

[HK89] L. Hernquist and N. Katz. TREESPH - A unification of SPH with the hierarchical tree method. *The Astrophysical Journal Supplement Series*, 70:419–446, June 1989.

[HK05] Jeong-Mo Hong and Chang-Hun Kim. Discontinuous fluids. *ACM Trans. Graph.*, 24(3):915–920, 2005.

[HM76] Thomas J. R. Hughes and Jerrold E. Marsden. *A short course in fluid mechanics*. Publish or Perish Inc., Boston, Mass., 1976. Mathematics Lecture Series, No. 6.

[HMT01] Sunil Hadap and Nadia Magnenat-Thalmann. Modeling dynamic hair as a continuum. *Computer Graphics Forum*, 20(3):329–338, 2001.

[HNB+06] Ben Houston, Michael B. Nielsen, Christopher Batty, Ola Nilsson, and Ken Museth. Hierarchical rle level set: A compact

and versatile deformable surface representation. *ACM Trans. Graph.*, 25(1):151–175, 2006.

[HW65] Francis H. Harlow and J. Eddie Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12):2182–2189, 1965.

[Igl04] A. Iglesias. Computer graphics for water modeling and rendering: A survey. *Future Generation Computer Systems*, 20(8):1355–1374, 2004.

[II01] Rafael José Iorio, Jr. and Valéria de Magalhães Iorio. *Fourier analysis and partial differential equations*, volume 70 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2001.

[Ise96] Arieh Iserles. *A first course in the numerical analysis of differential equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, 1996.

[Joh91] Fritz John. *Partial differential equations*, volume 1 of *Applied Mathematical Sciences*. Springer-Verlag, New York, fourth edition, 1991.

[KAG+05] R. Keiser, B. Adams, D. Gasser, P. Bazzi, P. Dutre, and M. Gross. A unified lagrangian approach to solid-fluid animation. pages 125–148, 2005.

[KFCO06] Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. Fluid animation with dynamic meshes. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 820–825, New York, NY, USA, 2006. ACM Press.

[KM90] Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, pages 49–57, August 1990.

[KMT06] Yootai Kim, Raghu Machiraju, and David Thompson. Path-based control of smoke simulations. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on*

Computer animation, pages 33–42, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[KYT+06] L. Kharevych, Weiwei Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun. Geometric, variational integrators for computer animation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 43–51, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[Lam93] Horace Lamb. *Hydrodynamics*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, sixth edition, 1993. With a foreword by R. A. Caflisch [Russel E. Caflisch].

[LeV07] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2007.

[LGF04] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 457–462, New York, NY, USA, 2004. ACM Press.

[LLVT03] Thomas Lewiner, Hélio Lopes, Antônio Wilson Vieira, and Geovan Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003.

[Lov44] A. E. H. Love. *A treatise on the Mathematical Theory of Elasticity*. Dover Publications, New York, 1944. Fourth Ed.

[LR04] Benedict Leimkuhler and Sebastian Reich. *Simulating Hamiltonian dynamics*, volume 14 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2004.

[LSSF06] Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. Multiple interacting liquids. *ACM Trans. Graph.*, 25(3):812–819, 2006.

[Luc77] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82:1013–1024, December 1977.

[LvdP02] Anita T. Layton and Michiel van de Panne. A numerically efficient and stable algorithm for animating water waves. *The Visual Computer*, 18(1):41–53, 2002.

[MCG03] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Mei07] Chiang C. Mei. Lecture notes on fluid dynamics. Available at http://web.mit.edu/1.63/www/, 2007.

[Mey71] Richard E. Meyer. *Introduction to mathematical fluid dynamics*. Wiley-Interscience [A division of John Wiley & Sons, Inc.], New York-London-Sydney, 1971. Pure and Applied Mathematics, Vol. XXIV.

[Mey82] Richard E. Meyer. *Introduction to mathematical fluid dynamics*. Dover Publications Inc., New York, 1982. Corrected reprint of the 1971 original.

[MFZ97] Joseph P. Morris, Patrick J. Fox, and Yi Zhu. Modeling low reynolds number incompressible flows using sph. *Journal of Computational Physics*, 136(1):214–226, 1997.

[MH94] Jerrold E. Marsden and Thomas J. R. Hughes. *Mathematical foundations of elasticity*. Dover Publications Inc., New York, 1994. Corrected reprint of the 1983 original.

[MKN+04] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[MMTD07] Patrick Mullen, Alexander McKenzie, Yiying Tong, and Mathieu Desbrun. A variational approach to eulerian geometry processing. *ACM Trans. Graph.*, 26(3):66, 2007.

[MN91] Severiano Toscano Melo and Francisco Moura Neto. *Mecânica dos Fluidos e Equações Diferenciais*. 18º Colóquio Brasileiro de Matemática, Instituto Nacional de Matemática Pura e Aplicada, 1991.

[Mon92] J. J. Monaghan. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30:543–574, 1992.

[Mon94] J. J. Monaghan. Simulating free surface flows with sph. *J. Comput. Phys.*, 110(2):399–406, 1994.

[Mon05] J. J. Monaghan. Smoothed particle hydrodynamics. *Rep. Progr. Phys.*, 68(8):1703–1759, 2005.

[MQ02] Robert I. McLachlan and G. Reinout W. Quispel. Splitting methods. *Acta Numer.*, 11:341–434, 2002.

[MSKG05] Matthias Müller, Barbara Solenthaler, Richard Keiser, and Markus Gross. Particle-based fluid-fluid interaction. In *2005 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 237–244, July 2005.

[MST+04] Matthias Müller, Simon Schirm, Matthias Teschner, Bruno Heidelberger, and Markus Gross. Interaction of fluids with deformable solids. *Computer Animation and Virtual Worlds*, 15(3-4):159–171, 2004.

[MTPS04] Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 449–456, New York, NY, USA, 2004. ACM.

[Nac01] André Nachbin. *Aspectos de Modelagem Matemática em Dinâmica dos Fluidos*. 23º Colóquio Brasileiro de Matemática, Instituto Nacional de Matemática Pura e Aplicada, 2001.

[Nac07] André Nachbin. Dinâmica dos fluidos. Lecture Notes, 2007.

[Nak07] Fábio Issao Nakamura. Animação interativa de fluido baseada em partículas pelo método sph. Master's thesis, Pontifícia Universidade Católica do Rio de Janeiro, 2007.

[Net07] Afonso Paiva Neto. *Uma abordagem Lagrangeana para simulação de escoamentos de fluidos viscoplásticos e multifásicos*. PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro, 2007.

[NFJ02] Duc Quang Nguyen, Ronald Fedkiw, and Henrik Wann Jensen. Physically based modeling and animation of fire. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 721–728, New York, NY, USA, 2002. ACM Press.

[NMK+06] Andrew Nealen, Matthias Muller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[OD01] Carol O'Sullivan and John Dingliana. Collisions and perception. *ACM Trans. Graph.*, 20(3):151–168, 2001.

[ODGK03] Carol O'Sullivan, John Dingliana, Thanh Giang, and Mary K. Kaiser. Evaluating the visual fidelity of physically based animations. *ACM Trans. Graph.*, 22(3):527–536, 2003.

[OF03] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2003.

[O'S05] Carol O'Sullivan. Collisions and attention. *ACM Trans. Appl. Percept.*, 2(3):309–321, y 05.

[Pat87] Jorge Patiño. *Introdução à Teoria da Elasticidade*. 16º Colóquio Brasileiro de Matemática, Instituto Nacional de Matemática Pura e Aplicada, 1987.

[PK05] Sang Il Park and Myoung Jun Kim. Vortex fluid for gaseous phenomena. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 261–270, New York, NY, USA, 2005. ACM Press.

[PPLT06] Afonso Paiva, Fabiano Petronetto, Thomas Lewiner, and Geovan Tavares. Particle-based non-newtonian fluid animation for melting objects. *SIBGRAPI*, 0:78–85, 2006.

[PTB+03] Simon Premoze, Tolga Tasdizen, James Bigler, Aaron Lefohn, and Ross T. Whitaker. Particle-based simulation of fluids. *Computer Graphics Forum*, 22(3):401–410, September 2003.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C*. Cambridge University Press, Cambridge, second edition, 1992. The art of scientific computing.

[REN+04] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 193–202, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[RNGF03] Nick Rasmussen, Duc Quang Nguyen, Willi Geiger, and Ronald Fedkiw. Smoke simulation for large scale phenomena. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 703–707, New York, NY, USA, 2003. ACM Press.

[Saa03] Yousef Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[SB02] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*, volume 12 of *Texts in Applied Mathematics*. Springer-Verlag, New York, third edition, 2002. Translated from the German by R. Bartels, W. Gautschi and C. Witzgall.

[SC91] Andrew Staniforth and Jean Côté. Semi-lagrangian integration schemes for atmospheric models — a review. *Monthly Weather Review*, 119(9):2206–2223, 1991.

[SCC04] C. Scheidegger, J. Comba, and R. Cunha. Navier-stokes on programmable graphics hardware using smac. In *Proceedings of XVII SIBGRAPI – II SIACG*, pages 300–307. IEEE Press, 2004.

[SCC05] C. Scheidegger, J. Comba, and R. Cunha. Practical cfd simulations on the gpu using smac. *Computer Graphics Forum*, 24(4):715–728, 2005.

[SCP+04] Maurya Shah, Jonathan M. Cohen, Sanjit Patel, Penne Lee, and Frédéric Pighin. Extended galilean invariance for adaptive fluid simulation. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 213–221, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[SD06] Ari Stern and Mathieu Desbrun. Discrete geometric mechanics for variational time integrators. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 75–80, New York, NY, USA, 2006. ACM Press.

[SF93] Jos Stam and Eugene Fiume. Turbulent wind fields for gaseous phenomena. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 369–376, New York, NY, USA, 1993. ACM.

[SF95] Jos Stam and Eugene Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 129–136, New York, NY, USA, 1995. ACM Press.

[SMML07] Jason Sewall, Paul Mecklenburg, Sorin Mitran, and Ming Lin. Fast fluid simulation using residual distribution schemes. In *Proceedings of the Eurographics Workshop on Natural Phenomena*, 2007.

[SRF05] Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph.*, 24(3):910–914, 2005.

[Sta99] Jos Stam. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[Sta00] Jos Stam. Interacting with smoke and fire in real time. *Communications of the ACM*, 43(7):76–83, 2000.

[Sta01] Jos Stam. A simple fluid solver based on the fft. *J. Graph. Tools*, 6(2):43–52, 2001.

[Sta03] Jos Stam. Flows on surfaces of arbitrary topology. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 724–731, New York, NY, USA, 2003. ACM Press.

[Str80] Gilbert Strang. *Linear algebra and its applications*. Academic Press [Harcourt Brace Jovanovich Publishers], New York, second edition, 1980.

[Str04] John C. Strikwerda. *Finite difference schemes and partial differential equations*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2004.

[SU94] John Steinhoff and David Underhill. Modification of the euler equations for "vorticity confinement": Application to the computation of interacting vortex rings. *Physics of Fluids*, 6(8):2738–2744, 1994.

[SY05a] Lin Shi and Yizhou Yu. Controllable smoke animation with guiding objects. *ACM Trans. Graph.*, 24(1):140–164, 2005.

[SY05b] Lin Shi and Yizhou Yu. Taming liquids for rapidly changing targets. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 229–236, New York, NY, USA, 2005. ACM.

[TB97] Lloyd N. Trefethen and David Bau, III. *Numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[TDM96] M. F. Tomé, B. Duffy, and S. McKee. A numerical technique for solving unsteady non-newtonian free surface flows. *Journal of Non-Newtonian Fluid Mechanics*, 62(1):9–34, 1996.

[TF88] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Computer*

Graphics (Proceedings of SIGGRAPH 88), pages 269–278, August 1988.

[TFC+01] M. F. Tomé, A. C. Filho, J. A. Cuminato, N. Mangiavacchi, and S. McKee. Gensmac3d: A numerical method for solving unsteady three-dimensional free surface flows. *International Journal for Numerical Methods in Fluids*, 37(7):747–796, 2001.

[TGC+04] M. F. Tomé, L. Grossi, A. Castelo, J. A. Cuminato, N. Mangiavacchi, V. G. Ferreira, F. S. deSousa, and S. McKee. A numerical method for solving three-dimensional generalized newtonian free surface flows. *Journal of Non-Newtonian Fluid Mechanics*, 123(2-3):85–103, 2004.

[Thu07] Nils Thuerey. *Physically based Animation of Free Surface Flows with the Lattice Boltzmann Method*. PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, Institut für Informatik, 2007.

[TKPR06] N. Thüerey, R. Keiser, M. Pauly, and U. Rüde. Detail-preserving fluid control. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 7–12, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[TLP06] Adrien Treuille, Andrew Lewis, and Zoran Popović. Model reduction for real-time fluids. *ACM Trans. Graph.*, 25(3):826–834, 2006.

[TM94] Murilo F. Tomé and Sean McKee. Gensmac: A computational marker and cell method for free surface flows in general domains. *Journal of Computational Physics*, 110(1):171–186, 1994.

[TMPS03] Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. Keyframe control of smoke simulations. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 716–723, New York, NY, USA, 2003. ACM.

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, pages 205–214, July 1987.

[Tre96] Lloyd N. Trefethen. Finite difference and spectral methods for ordinary and partial differential equations. Unpublished manuscript avaliable at http://web.comlab.ox.ac.uk/oucl/work/nick.trefethen/pdetext.html, 1996.

[TRS06] Nils Thürey, Ulrich Rüde, and Marc Stamminger. Animation of open water phenomena with coupled shallow water and free surface simulations. In *2006 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 157–166, September 2006.

[WB01] Andrew Witkin and David Baraff. Physically based modeling. In *SIGGRAPH'01: ACM SIGGRAPH 2001 Courses*, 2001. Available at http://www.pixar.com/companyinfo/research/pbm2001/.

[WBOL07] Jeremy D. Wendt, William Baxter, Ipek Oguz, and Ming C. Lin. Finite volume flow simulations on arbitrary domains. *Graph. Models*, 69(1):19–32, 2007.

[WG00] Dieter A. Wolf-Gladrow. *Lattice-gas cellular automata and lattice Boltzmann models*, volume 1725 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2000. An introduction.

[WH91] Jakub Wejchert and David Haumann. Animation aerodynamics. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 19–22, New York, NY, USA, 1991. ACM.

[Whi99] G. B. Whitham. *Linear and nonlinear waves*. Pure and Applied Mathematics (New York). John Wiley & Sons Inc., New York, 1999. Reprint of the 1974 original, A Wiley-Interscience Publication.

[WLMK04] Xiaoming Wei, Wei Li, Klaus Mueller, and Arie E. Kaufman. The lattice-boltzmann method for simulating gaseous phenomena. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):164–176, March/April 2004.

[YHK07] Cem Yuksel, Donald H. House, and John Keyser. Wave particles. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):to appear, 2007.

[ZB05] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Trans. Graph.*, 24(3):965–972, 2005.