Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Irányítástechnika és Informatika Tanszék

Real time simulation and control of Newtonian fluids using the Navier-Stokes equations

Diplomaterv

Készítette

Zsolnai Károly

zsolnai@iit.bme.hu

Konzulens

Dr. Szirmay-Kalos László

szirmay@iit.bme.hu

Budapest, 2012. Szeptember 18.

Contents

TANSZÉKVEZETŐ

DIPLOMATERVEZÉSI FELADAT

Zsolnai Károly

MSc informatikus hallgató részére

Folyadékok valós idejű szimulációja és irányítása a Navier-Stokes egyenletek alkalmazásával

A folyadékok dinamikáját a Navier-Stokes egyenletek fogalmazzák meg, amelyek megoldásával adott anyagi viszonyok és peremfeltételek mellett a folyadék mozgása kiszámítható. Ennek a feladatnak az inverz problémája olyan feltételeket, pl. anyagi jellemzőket és peremfeltételeket teremt, amely esetén a folyadék az előre definiált jellegzetességeket mutatja. Az inverz feladat megoldása fontos alkalmazást kap a produkciós rendszerekben, amikor nem általában kell folyadékokat szimulálnunk, hanem azoknak valamilyen célt kell követniük. A feladat megoldásához a szimulációt, azaz a Navier-Stokes egyenlet megoldását ki kell egészíteni olyan kereső eljárásokkal, amelyek a célfüggvény felé terelik a folyadékot. A diplomaterv célja egy ilyen rendszer elméletének kidolgozása és megvalósítása.

A hallgató feladatának a következőkre kell kiterjednie:

- Foglalja össze a Navier-Stokes egyenletek numerikus megoldási módszereit, kitérve az Euler-i és Lagrange-i megközelítésekre.
- Tervezzen meg egy Euler-i szimulátort, vizsgálja meg a hierarchikus rácsok alkalmazási lehetőségeit is.
- Dolgozzon ki algoritmusokat, amelyek a sűrűséget egy előre megadott függvényhez közelítik.
- Implementálja a javasolt módszert CPU-n és GPU-n is.
- Értékelje a megvalósítást az irodalmi adatok fényében.

Tanszéki konzulens: Dr. Szirmay-Kalos László, egyetemi tanár

Budapest, 2012. március 5.

Dr. Szirmay-Kalos László
egyetemi tanár
tanszékvezető

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Control Eng. and Inf. Technology

Budapest, Magyar tudósok körútja 2., H-1117, Hungary
Web: http://www.iit.bme.hu
Tel: +36-1463-2353, Fax: +36-1463-2204,
e-mail: szirmay@iit.bme.hu

Hallgatói nyilatkozat

Alulírott Zsolnai Károly szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök, stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. A teljes szöveg közzététele dékáni engedéllyel titkosított diplomatervekre nem vonatkozik.

Budapest, 2012 Szeptember 18.

.....

szigorló hallgató

Összefoglaló

A mindennapi életünk során előforduló természeti jelenségek megfigyelése kapcsán joggal gondolhatjuk, hogy a folyadékok mozgása mögött meghúzódó törvényszerűségek megismerése közel emberfeletti nehézségekkel jár. Valamivel több, mint száz évvel ezelőtt Claude-Louis Navier és Sir George Stokes egy rövid, univerzális formulát fogalmazott meg az összenyomhatatlan folyadékok dinamikájával kapcsolatosan, mely rádöbbentett minket arra, hogy ezek a szabályszerűségek nem csak hogy megismerhetők, de csupán három összetevő kiértékelésével még a legbonyolultabb természeti áramlások is modellezhetők. Mindezek ellenére a folyadékok mozgásának szimulálására alkalmas programot készíteni nem triviális feladat, ám az erre irányuló erőfeszítéseink nagyszerű kilátásokkal kecsegtetnek, hiszen napjainkban már rengeteg fizikai és mérnöki alkalmazásban is hasznosnak bizonyultak. Ezen szakdolgozatban az eddigi kutatási eredmények felhasználásával megmutatjuk, hogy hogyan készíthető el egy olyan rendszer, mely képes hatékonyan megoldani a Navier-Stokes egyenleteket.

A dolgozatban szó esik a folyadékirányítási problémáról is, mely megoldásához a folyadék térfogatban való szimulációja mellett egy végső sűrűségeloszlás (egy tetszőleges alakzat) is adott, amihez egy időben változó, külső erőteret definiálunk azzal a célzattal, hogy a folyadék előbb-utóbb ezt a formát vegye fel. Az eddigi kutatási eredmények egyik fontos konklúziója, hogy a probléma több szinten is kiemelkedő nehézségű: elsőként, a térben vett minden anyagi pont környezetében több másik anyagi pont is található, melyekkel az szoros interakcióban van, így külön körültekintéssel kell lenni arra, hogy az egyikükre kifejtett erőhatás hogyan fog hatni annak szomszédságában. Másodszorra, egy kívánatos, de ugyanakkor szintén ambivalens követelmény is megfogalmazódott, miszerint a folyadékmozgásnak a külső erőtér hozzáadása mellett is természetesnek kell hatnia, még akkor is, ha olyan erősen valószínűtlen áramlásokat írunk le, mely során a víz egy térfogategységben egy emberi arc, vagy egy felhőkarcoló alakját veszi fel.

Az eddigi legígéretesebb kutatási eredmények a funkcionálanalízis és az irányításelmélet területén használt kifinomult matematikai módszerek alka-

lmazásával képesek a probléma kezelésére, ám ezeknek az árát képkockánként 5-7 percnyi számításigény kivárásával kell megfizetnünk. Ez gyakorlatilag kétórányi munkaidőt jelent minden egyes másodpercnyi anyag elkészítésére.

Ezen szakdolgozat hasábjain egy újszerű megközelítést képviselő algoritmus részleteit mutatjuk be, mely képes a folyadékirányítási probléma valós idejű megoldására, és az eredménye akár egy átlagos otthoni számítógép használatával demonstrálható.

Abstract

When observing natural phenomena in our everyday life involving complex fluid flows, one may feel that it ought to be immensely difficult to understand the underlying laws of fluid motion. More than a hundred years ago, Claude-Louis Navier and Sir George Stokes came up with a short universal formula that describes the motion of incompressible fluids. This made us realize that these underlying rules are surprisingly simple: even the most difficult looking flows can be understood by the evaluation of only three terms. Implementing a fluid simulation program on computer hardware based on this equation is not a trivial problem, however, it is quite rewarding: these simulation programs serve a vast array of applications in physics- and engineering-related problems. In this thesis, based on previous research work, we show a highly parallel framework that is capable of giving an effective numerical solution of the Navier-Stokes equations.

Here we also address the fluid control problem, where, alongside simulating the motion of fluids, an arbitrary density distribution (a shape of any kind) is given, and forces are exerted on the system with the intention that the fluid would sooner or later take this form. Prior research work had shown that the problem is of enormous difficulty due to multiple reasons: first, every region is tightly bound to its neighborhood, therefore a force that acts upon a point in space will also have effect on nearby regions, therefore making the controlling process unpredictable. Second, it also a desirable, but ambivalent requirement that the controlling external force field should make the fluid flow realistic, even though it is highly improbable that a given volume of water would suddenly flow exactly into a shape of a human face or a skyscraper. Utilizing highly sophisticated mathematical methods from different fields such as optimization and control theory, current state of the art methods are able to give visually pleasing results at the cost of 5 to 7 minutes of computation time per frame, effectively taking more than two hours of time for every second of video footage.

In this thesis work, we present a novel solution for the fluid control problem, making it possible to solve it in real time, even on home computers

with limited processing power.

1 Introduction

1.1 Motivation and Contributions

For the untrained eye, natural phenomena involving fluid movement may look incomprehensibly complex. Now, for a little more than 100 years, we have had the privilege of knowing the few simple rules that account for the immensely complex flows of waterfalls, pouring honey, or the air turbulence experienced during an airplane flight. The world of the motion picture industry had also shown interest the topic: in order to enhance the visual impact of movie clips, it is now common to show simulation results alongside with real life footage with physically based fluid and gas simulation programs. The realism has now reached a point where even a professional with a keen eye would be hard-pressed to distinguish the real from the unreal.

About ten years ago, equipped with all this knowledge regarding the motion of fluids, an interesting new question has emerged: what if we could not only simulate, but *control* fluid movement? What kind of forces are needed for a fluid to take a given direction or shape? *What if we could not only predict disasters such as the tsunami or hurricanes, but avoid them entirely? What if we could not only predict blood flow in the human body but prevent the development of a deadly aneurysm in the aorta?* Later, in Section 1.3 we show prior research work on how the answer could possibly have saved Albert Einstein's life. This question is what scientists call the *fluid control problem* and a novel method to solve it efficiently will be the topic for this thesis work. There are also artistic applications of the topic (Figure 2), even in Hollywood motion picture productions (discussed in Section 6).

However, solving the fluid control problem proved to be immensely difficult: we know of no real time solution that exists as of now, and even the best state-of-the-art methods take 5-7 minutes of computation time for every frame, therefore one second of video footage would take at least two hours of wait time. The wide variety of applications showed that the industrial world could take immense advantage of a real time solution to the problem. This is no different in motion picture production: no film studio is going to pay its artists to click once and then wait hours in or-

Figure 1: Andreas Burmberger's Cherry Splash scene exhibiting surface tension computation.

der to know how the last changes affects the view of the scene that's being worked on.

In this thesis, we present the following contributions:

- A review of the mathematical background of computational fluid dynamics,
- Statement of the fluid control problem and a brief overview of a state-of-the-art solution to it,
- Observations regarding the fluid control problem, where simplifications can be made to the target distribution while retaining its applicability to both industrial and artistic use,
- A novel, highly parallelizable method to simplify the optimization approach that relies only on local data, and which is able to solve the fluid control problem in real time while retaining realistic looking fluid flows.
- An implementation of the algorithm on GPU hardware that is released alongside this thesis work, which is under the GPL license and is freely accessible for everyone through the department website.

Figure 2: Demonstration of the artistic side of the fluid control problem. Image source: Yizhou Yu.

1.2 Computational Fluid Dynamics

In 1822, Claude-Louis Navier and Sir George Stokes have made an important contribution to the world of mathematics and physics by formulating partial differential equations that describe how the motion of fluids changes in time [1]. Computational fluid dynamics is a vital part of fluid mechanics which incorporates numerical methods to visualize and solve problems involving fluid flows, which, most of the cases means solving the Navier-Stokes equations with given periodic or bounded initial conditions. These equations also pose an interesting and challenging mathematical problem as it is still unknown whether smooth solutions exist for any three-dimensional system with given boundary conditions. This is one of the seven *Millennium Prize Problems*[1], and anyone who is able to come up with a solution or a counterexample is awarded $1.000.000 US dollars. The smooth solution itself, whether it always exists or not, isn't that important, but the process of solving it is expected to show new insight to the behavior and the understanding of partial differential equations.

As the problem is given in the form of a partial differential equation, which often proves to be of great difficulty or even impossible to solve for practical cases, with the use of the *finite element method* it is possible to obtain a numerical solution of remarkable accuracy by solving the equation on a sufficiently high resolution discrete grid. However, the solution is still not trivial: the evaluation of the advection term of the fluid (discussed in Section 2 and 3.2) will not result in a stable fluid flow

[1]The problem statement and details are accessible on the The Clay Mathematics Institute's Millennium Prize Problems website.

simulation. In order to overcome this, a clever treatment of Jos Stam's [2] is applied.

1.3 Applications of Fluid Dynamics

Computational fluid dynamics enjoys a wide variety of use in a number of engineering and physics problems: it is possible to visualize heat distribution in a newly designed car engine, to validate airplane design by performing wind tunnel tests with a computer software (Figure 6), simulate gas or water flow in chopper pumps (Figure 4a[2]), visualize and plan optimum air circulation in buildings, calculate the drag for accelerating vehicles of various shapes (Figure 5), or to understand the possible outcomes better when a catastrophe, such as a flood, the tsunami, or a volcanic eruption happens. There are also methods to help the medical examination of humans by detecting probable spots in the aorta for aneurysms, little bulges in the wall of blood vessels, which, under extremal pressure conditions, may explode, causing oftentimes lethal implications. This is what also caused the death of the famous physicist, *Albert Einstein*, where more in-depth knowledge of fluid dynamics may have saved his life.

In [3] an MRI reconstruction problem is discussed, which is followed by a blood flow simulation to help blood hemodynamic analysis for recognizing congenital diseases early on. The results can be viewed in Figure 3. We also show a wind flow visualization[3] around a chimney stack on in Figure 4b and measurement test results of turbulent wind flows near the Concorde airplane in Figure 6. It is also worth noting that the Navier-Stokes equations, combined with *Maxwell's* equations of electromagnetism, enable us to arrange experiments in magnetohydrodynamics, the study of magnetic fields in electrically conducting fluids.

1.4 The Eulerian and Lagrangian Approach

There are different ways to approach the numerical simulation of fluids. The Lagrangian method treats the fluid as a collection of particles that move in space and time according to their velocities and positions. The Eulerian approach is named

[2]The simulation is obtained by ANSYS CFD, and the source of the image.

[3]We show some works of Henri Werlé, which are accessible at the eFluids website and *On the Flow of Fluids Made Visible, Henri Werlé, Leonardo, Vol. 8, No. 4. (Autumn, 1975), pp. 329-331.*

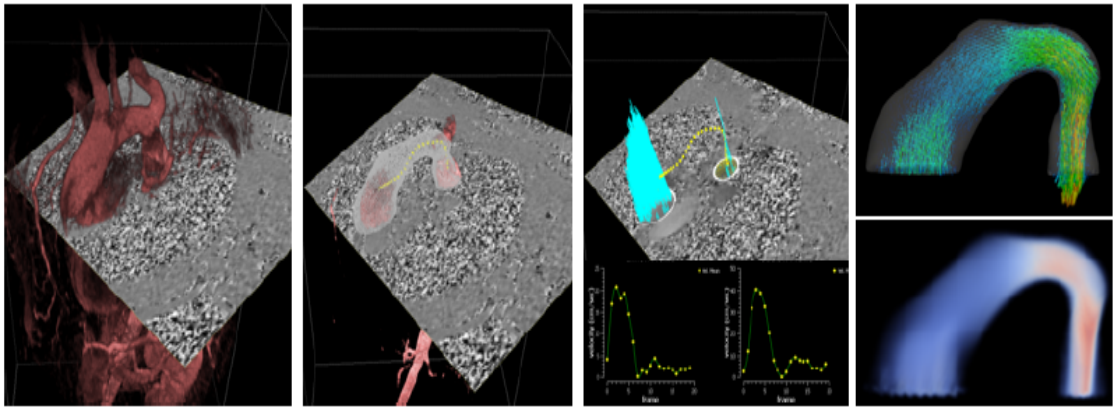
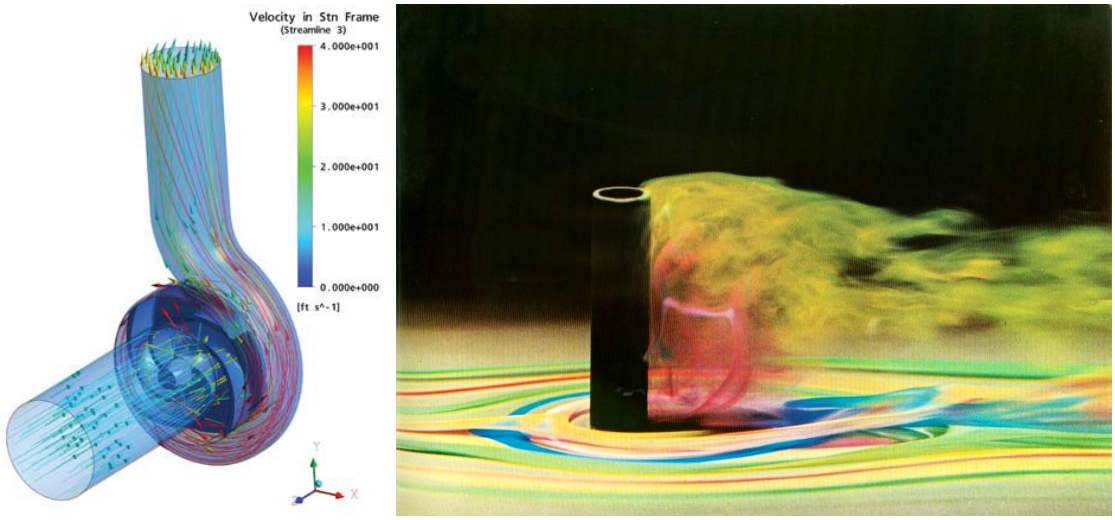


Figure 3: A sequence of images where upon obtaining the centerline of the aortic arch (second image) from the volume rendering (first image) of an MRI image, blood in- and outflow informations are gathered (third image) to simulate the blood flow of the patient. Image source: Lucian Itu [4].

after Swiss mathematician and physician Leonhard Euler, who took the problem from a different point of view: it defines a stationary grid by its points, and the relevant quantities like pressure, velocity and density are measured in these points. The fluid is expected to flow through and between these grid points. The quality of the simulation depends greatly on the resolution of the grid (i.e. how densely these points are strewn in space where we measure) and since it is impossible to compute the quantities in infinite points of the continuum, an interpolation function is used to obtain information from between the sample points. Both viewpoints utilize the techniques of the *finite element method*, which, through problem discretization, offers valuable tools to approximate the solutions of partial differential and integral equations with difficult boundary conditions.

László Szirmay-Kalos has given an intuitive example to help understanding the key difference between the two viewpoints: when collecting data for a weather report, the meteorologist can sit in an air balloon and measure quantities like temperature, wind speed and directions, humidity while floating along the flow of the air. This would be the Lagrangian approach. Someone utilizing the Eulerian viewpoint may simply set up stationary measurement devices in various places throughout the country. In Section 3.2, we discuss the background of the Eulerian approach of computational fluid dynamics.



(a) (b)

Figure 4: (a) The observation of a chopper pump in a simulated wind tunnel makes it possible to understand what shapes the curved blades have to take on the impeller to maximize the throughput of the device. (b) Henri Werlé's test visualization shows the trajectory of the smoke plume emitted by the chimney and also the well-known horseshoe vortex on the ground. The source of the images are given in the corresponding section.

2 The Navier-Stokes Equations

The Navier-Stokes equations are statements about the motion of fluid substances, which are assumed to be incompressible, homogeneous and Newtonian:

$$\frac{\partial u}{\partial t} = \underbrace{-(u \cdot \nabla)u}_{\text{advection}} - \underbrace{\frac{1}{\rho}\nabla p}_{\text{pressure}} + \underbrace{\nu \nabla^2 u}_{\text{diffusion}} + \underbrace{F_{ext}}_{\text{external forces}}, \tag{1}$$

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

where $\boldsymbol{u}$ is the traditional notation of the velocity field in fluid mechanics, $\boldsymbol{\rho}$ stands for the density, $\boldsymbol{p}$ for pressure, $\boldsymbol{\nu}$ denotes the kinematic viscosity of the fluid, and $\boldsymbol{F_{ext}}$ is the representation for the sum of all external forces.

2.1 The Momentum Equation

On the left-hand side of the first equation, called the *momentum equation* provides the answer for the most fundamental question of fluid flows: in which direction a

Figure 5: Wind tunnel tests reveal the amount of drag acting upon the caravan towed by an automobile.

given particle of the fluid will continue its path in the moment. Mathematically speaking, we would like to know how the velocity vector field changes in time. Before proceeding to further observations, perhaps it is helpful to note that this equation is equivalent to Newton's second law [5], $F = ma$ for which we show a derivation in Appendix A. The right side of the equation can be written as a sum of three simple terms: **advection**, **diffusion**, **pressure**, and if needed, an **external force field**. They all describe real life natural phenomena, therefore they all have their intuitive meanings. Let us take a brief look at them:

Advection

The motion of a fluid causes motion to any quantity that it contains, as dropping objects into a river will make them follow the flow. The terminology refers to it as *advection* or *convection*, which phenomenon is the heart, and also the bane of every fluid simulation program. However, it is not a widely known fact that the velocity field moves itself as well, not just the objects within it - the mathematical formulation for this is the directional derivative $(u \cdot \nabla)$ of the velocity field,

$$-(u \cdot \nabla)u.$$

Figure 6: Front view of the wind flow around the Concorde from a landing configuration, made by Henri Werlé.

The presence of the minus sign shows up as a result of the rearrangement in the derivation shown in Appendix A.

Pressure

In any kind of physical system, particles in higher pressure regions are pushed towards lower pressure regions. We can compute the direction of the pressure field in an arbitrary point by calculating ∇p, which is the gradient of the pressure field. Intuitively, this will point into the direction of the steepest ascent of the pressure - as mentioned above, this pushing force will lead to acceleration in the fluid. Because higher pressure regions push lower pressure regions and not the other way around, the direction of the flow is represented by $-\nabla p$ instead. This almost looks like the full pressure term, but so far we haven't taken into account the *density* of the fluid. As we need to exert more force to get a denser fluid into motion, the pressure force

is divided by ρ:

$$-\frac{1}{\rho}\nabla p.$$

Diffusion

It is a well-known property of fluids that differences in concentrations average out in time - when dripping ink into water, it will spread perfectly over time, giving the same color for the whole glass of fluid. Different types of fluids react differently to external forces - some of them are more resistant to the force we apply on them. For example, pouring honey into a cup of tea may take some more patience than doing the same with water. *Diffusion* defines this kind of motion using the Laplace operator (∇^2), which gives the sum of all the second partial derivatives of a quantity. Intuitively, applying this differential operator for a quantity gives the difference between a point and the average value between its neighboring points. For example, $\nabla^2 u = 0$ practically means that the velocity at each point is equal to the average of the nearby points. Any fluid and gas type material in nature is striving to achieve this equilibrium state. Now it is simple to understand the formal representation of *diffusion*: bigger difference of velocity and density means more motion is needed to balance it. In the formula, we also use a dampening factor, the *kinematic viscosity*, which describes the resistance of the fluids against these forces. It can be thought as the fluid's strength to fight back to prevent this kinematic stress - higher viscosity yields more resistance and therefore, slower movement (e.g. honey or different types of cocktails where multiple layers of fluids can be seen on top of each other).

$$\nu\nabla^2 u$$

External force field

This is meant to be the sum of all forces that are exerted on the fluid and has nothing to do with the fluid's internal mechanics. The source of these forces can be practically anything: gravity drag, blowing wind, or even user interaction in a simulation program. The external force field F_{ext} accelerates the velocity field at each point, therefore taking this term into consideration is as simple as adding the two vector fields together.

2.2 Incompressibility Condition

In a similar notation as Bridson used in [6] (p. 17), let Ω be an arbitrary part of the fluid volume with boundary surface $\partial\Omega$. The rate of change for this volume is measured by integrating the normal component of its velocity around the boundary, which intuitively gives the amount of density that enters or leaves the volume:

$$\iint_{\partial\Omega} u \cdot n \, d\Omega = \frac{d}{dt} vol(\Omega). \tag{3}$$

For an incompressible fluid, the volume remains constant over time, therefore the change rate of the volume is zero:

$$\iint_{\partial\Omega} u \cdot n \, d\Omega = 0. \tag{4}$$

The Divergence Theorem states that the outward flux of a vector field through a closed surface is equal to the volume integral of the divergence of the region inside the surface. An equivalent, but more intuitive interpretation of this theorem is that for a quantity, sum of all sources minus the sum of all sinks gives the net flow out of a region:

$$\iint_{\partial\Omega} u \cdot n \, d\Omega \; = \iiint_{\Omega} \nabla \cdot u \, d\Omega = 0. \tag{5}$$

This equation must hold for an arbitrary choice of Ω (meaning any possible region). Our function integrates to zero regardless of the choice of volume for integration. This implies the integrand to be zero everywhere (and vice versa):

$$\forall\Omega : \iiint_{\Omega} \nabla \cdot u \, d\Omega = 0 \quad \iff \quad \nabla \cdot u = 0. \tag{6}$$

We are now familiar with the definition of the second Navier-Stokes equation (2).

In the future we will refer to it as the *incompressibility condition* - any vector-field that satisfies this condition has to be divergence-free. A divergence-free velocity field implies mass conservation for fluids, which, in simple terms, would mean that no mass (and therefore energy [7]) is created or destroyed in the system.

Figure 7: Aaron Hill's Cornell box scene incorporates heavy use of boundary conditions and volume ray tracing techniques.

3 Numerical Solution To The Navier-Stokes Equations

As we now have a precise mathematical model for simulating fluid flows, the objective would be to give an analytical solution the Navier-Stokes equations discussed in Section 2. Unfortunately, with the exception of a few simple setups, this is impossible. This should be taken into account, especially for critical applications where complex scenarios are given, and exact solutions are required. For example, when designing a race car, it should be beyond argument that it is a matter of life and death to model every kind of force that applies to the chassis as accurately as possible, therefore very accurate (if not analytical) solutions as required. However, if our goal is to produce convincing visual footage of fluid flows, it may be well enough to give a close enough approximate solution to the equations, preferably in real time. In this chapter, we discuss solution techniques that are able to achieve both kinds of realism.

3.1 The Algorithm

As initial conditions, we have a (possibly empty) initial velocity vector field and a scalar field for both pressure and density. The simulation is broken into 4 basic steps, evaluated one after other:

1. Compute the *advection* of the velocity field,
2. Calculate *diffusion*,
3. Apply the *external force field*,
4. *Project* the velocity field to its divergence-free component.

Techniques like vorticity confinement and buoyancy force computation should also be carried out in the force field application step. These techniques will also be discussed in this section.

3.2 Evaluation of The Terms

In order to solve the Navier-Stokes equations (1) with given initial conditions, we have to evaluate several kinds of differential operators, which all require derivative information. As it is too demanding (and often not possible) to compute these derivative informations analytically, we solve the problem through discretizing it onto a 2 or 3 dimensional grid, where the much simpler *finite differential form* of the operators can be used. We show this form for the gradient operator to evaluate the pressure term and the Laplace operator to evaluate the diffusion processes in the fluid. For the advanced techniques discussed in Section 3.6, we also show the finite difference form for the divergence and the curl operators.
We also show that it is not trivial to evaluate the directional derivative in the advection term, as a regular forward projection gives results that leave much to be desired: this is an unstable method, therefore the simulation is to ''blow up'' in a matter of seconds, even when simulating time steps of little magnitude.

The finite differential form of the required operators are the following, as-

suming $\delta x, \delta y$ is the distance between two gridpoints in x, y directions:

$$\textbf{Gradient: } \nabla p = \left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}\right) \approx \left(\frac{p_{i+1,j} - p_{i-1,j}}{2\delta x}, \frac{p_{i,j+1} - p_{i,j-1}}{2\delta y}\right) \tag{7}$$

$$\textbf{Divergence: } \nabla \cdot \mathbf{u} = \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \approx \left(\frac{u_{i+1,j} - u_{i-1,j}}{2\delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\delta y}\right) \tag{8}$$

$$\begin{aligned} \textbf{Laplace: } \nabla^2 f = \left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\right) &\approx \left(\frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{2\delta x} + \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{2\delta y}\right) \\ &= \frac{f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1} - 4f_{i,j}}{\delta x^2 + \delta y^2} \end{aligned} \tag{9}$$

$$\textbf{Curl: } \nabla \times \mathbf{u} = \det \begin{bmatrix} \hat{\mathbf{i}} & \hat{\mathbf{j}} & \hat{\mathbf{k}} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ u & v & w \end{bmatrix} = \left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}\right)\hat{\mathbf{i}} + \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right)\hat{\mathbf{j}} + \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)\hat{\mathbf{k}} \tag{10}$$

Regarding the advection term $-(u \cdot \nabla)u$, the weapon of choice would be to project the velocity field forward with Δt by using the general formula

$$r(t + \Delta t) = r(t) + u(t)\Delta t, \tag{11}$$

however this would immediately violate energy and mass conservation, since nothing could ensure that the forward projection does not force the fluid to leave the piece of volume that we are simulating. To overcome this issue, we follow Jos Stam's intuitive solution [2], where instead of checking how the current density field would evolve in a Δt time step, we project the density field backwards and see where the density came from for every point on the grid:

$$r(x,\ t + \Delta t) = r(x - u(x,t)\Delta t,\ t). \tag{12}$$

Figure 8: Giulio Jiang's Picopool scene.

While this is both an effective and stable solution to the problem, the accuracy of the solution depends greatly on the choice of Δt. A time step of too great size may as well blow up the whole simulation, therefore it should be chosen wisely. In our experiments, choosing $\Delta t \leq 10^{-1}$ have always resulted in a stable simulation.

3.3 Boundary Conditions

When solving differential equations of any kind, explicit informations may be provided on how the solution should behave on the boundary of the region that is governed by the differential equation. We refer to these informations as *boundary conditions*, and give a short review on their definitions and uses in computational fluid dynamics.

Dirichlet-type boundary conditions specify the values of the solution on the boundary surface of the problem domain. They take different forms for ordinary

differential equations (ODE) and partial differential equations (PDE):

$$\begin{aligned} \textbf{ODE:} \qquad & y'' + y' + y = 0, \\ & y(a) = \psi, \;\; y(b) = \phi, \\ \textbf{PDE:} \qquad & \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \\ & u(k) = f(k) \;\; \text{on} \;\; \forall k \in \partial\Omega. \end{aligned} \tag{13}$$

Neumann-type boundary conditions specify the normal derivative of the function on a surface. It is particularly useful in fluid simulations and heat diffusion equations, where it is possible to specify zero derivatives on the Neumann-boundary, effectively meaning that no fluid or heat may enter or leave the simulation domain, therefore making it possible to simulate the interaction between fluids and rigid objects that are placed inside them.

$$\begin{aligned} \textbf{ODE:} \qquad & y'' + y' + y = 0, \\ & y'(a) = \psi, \;\; y'(b) = \phi, \\ \textbf{PDE:} \qquad & \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \\ & \frac{\partial u}{\partial n}(k) = f(k) \;\; \text{on} \;\; \forall k \in \partial\Omega. \end{aligned} \tag{14}$$

Please note that when solving the Navier-Stokes equations, the initial conditions for the velocity and density field, therefore an initial state of the vector and scalar fields should be given - most of the time, we simply initialize them to be zero everywhere, but for certain test cases, continuous functions may also be initialized.

3.4 Sparse Systems of Linear Equations

Consider a set of linear equations given in matrix form:

$$A\,\mathbf{x} = \mathbf{b} \tag{15}$$

where A is an $n \times n$ matrix and $\mathbf{x}$,$\mathbf{b} \in \mathbb{R}^n$.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{16}$$

In a sparse system, most of the elements of A are zeros, therefore it is significantly less demanding to store only the nonzero elements instead. A is usually decomposed into the sum of a diagonal and a remainder component matrix by

$$A = \underbrace{\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}}_{A_D} + \underbrace{\begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}}_{A_R}, \tag{17}$$

and therefore as the solution of the equation, we get:

$$\begin{aligned} A\,\mathbf{x} &= \mathbf{b} \\ (A_D + A_R)\,\mathbf{x} &= \mathbf{b} \\ A_D\,\mathbf{x} &= \mathbf{b} - A_R\,\mathbf{x} \\ \mathbf{x} &= A_D^{-1}(\mathbf{b} - A_R\,\mathbf{x}). \end{aligned}$$

Please note that computing A_D^{-1} is trivial as it is a diagonal matrix, therefore the inverse matrix will contain the reciprocal of the elements.

Unfortunately, iterative methods will not be convergent in every case - the requirement of convergence depends on the properties of the $A_D^{-1}A_R$ matrix. The spectral radius of a matrix is given by

$$\rho(R) = \max_j\{|\lambda_j|\} \tag{18}$$

where λ_j are the eigenvalues of the matrix. It is shown [8] that an iterative method is convergent if and only if $\rho(R) < 1$. Lower spectral radius yields better convergence speed, therefore it is desirable to find decompositions for A that have the lowest possible $\rho(R)$.

To reduce the computational error further, multigrid methods are often applied where instead of one, a hierarchy of discretizations is used, in which the final results include the solutions on a fine grid and global corrections are taken from the coarser scales. Adaptive multigrid methods exhibit mesh refinement where the resolution of the grid is increased to convey more realism for more important parts of the fluid domain.

3.4.1 Jacobi Iteration

The Jacobi iteration is far from the best linear system solving methods in terms of convergence, however, it is the most straightforward technique to implement, and also serves as a building block for other, more sophisticated methods. In the first Jacobi iteration we give an initial guess, $x^{(0)}$, which is improved in every iteration according to the following formula:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\forall j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, \ldots, n. \tag{19}$$

Note that this formula is equivalent to the derivation shown before in Section 3.4. The advantage of the method is that the elements of $x^{(k+1)}$ may be computed in any order, making it an effective option to use it on parallel computer architectures, such as the GPU.

3.4.2 Gauss-Seidel method

The convergence rate of the iterations can be further improved by using the newly computed $x^{(k+1)}$ values immediately (where available) in the same iteration

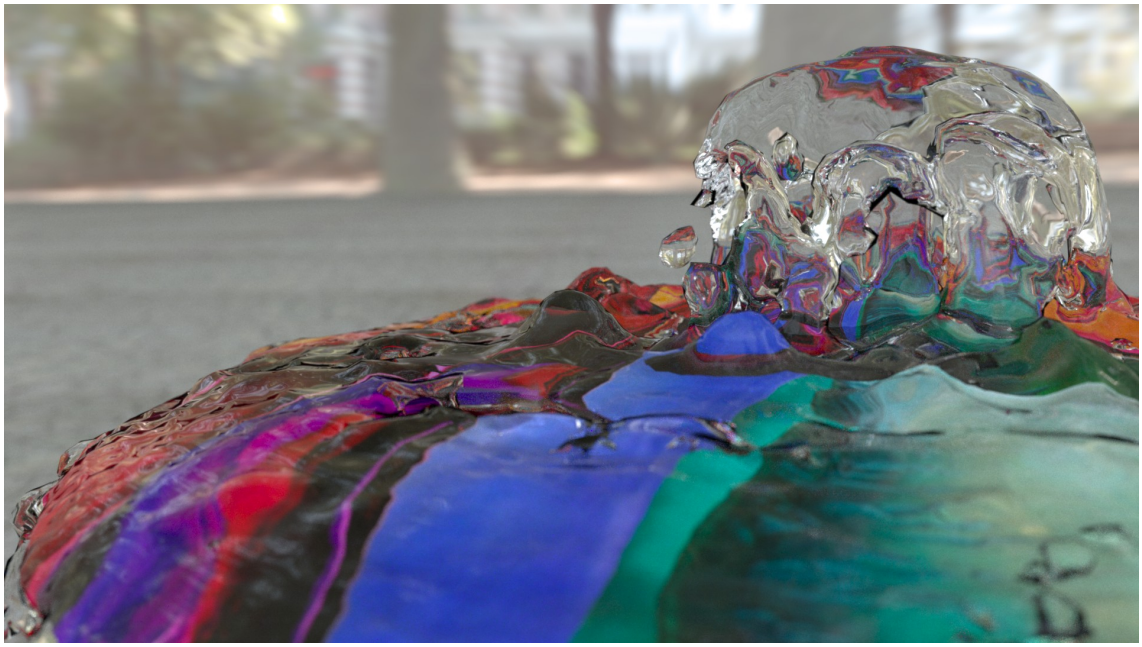


Figure 9: Giulio Jiang's Water Rainbow scene is a good example of the beauty of modeling turbulent water flows.

instead of saving them to the next iteration step.

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\forall j<i} a_{ij} x_j^{(k+1)} - \sum_{\forall j>i} a_{ij} x_j^{(k)} \right), \quad i = 1, \ldots, n. \tag{20}$$

This gives a significant improvement on the speed of convergence, and also there is no longer a need to allocate two arrays for $x^{(k)}$ and $x^{(k+1)}$, however, the order of the elements is no longer an arbitrary choice. Different strategies exist to address this problem, such as *Red-Black ordering*, or the natural ordering scheme which processes the elements row by row.

3.5 Helmholtz-Hodge decomposition

A mathematical method, known as the *Helmholtz-Hodge decomposition* ([9] p. 37, [2]) states that any $\boldsymbol{w}$ vector field can uniquely be decomposed to a sum of a divergence-free vector field and a gradient of a scalar field:

$$w = v + \nabla q, \tag{21}$$

provided that $\nabla \cdot v = 0$, and is is parallel to the boundary ($v \cdot n = 0$ on ∂D), and q is a scalar potential field. Let $\mathbb{P}(\cdot)$ be a projection operator which maps a vector field to its divergence-free part. The scalar field q is given to satisfy $\mathbb{P}(\nabla q) = 0$ and from the zero divergence property of v comes $\mathbb{P}(v) = v$, therefore $\mathbb{P}(w) = v$ will hold.

In order to maintain consistency with the incompressibility condition, (2), we can use the same idea and project both sides of the first Navier-Stokes equation (1),

$$\mathbb{P}\underbrace{\left(\frac{\partial u}{\partial t} + \frac{1}{\rho}\nabla p\right)}_{v+\nabla q} = \mathbb{P}\underbrace{\left(-(u\cdot\nabla)u + \nu\nabla^2 u + F_{ext}\right)}_{w}. \tag{22}$$

The vector field u has divergence-free property, therefore it is invariant under projection. This will also be true for $\partial u/\partial t$ if u is smooth, giving ($\mathbb{P}(\partial u/\partial t) = \partial u/\partial t$). After discarding the pressure term from the left side as $\mathbb{P}(\nabla p) = 0$, we get

$$\frac{\partial u}{\partial t} = \mathbb{P}\left(-(u\cdot\nabla)u + \nu\nabla^2 u + F_{ext}\right), \tag{23}$$

which, as discussed in Section 3.1, is the final form of the equation to be solved in the projection step of the simulation cycle.

A different kind of solution can be given by multiplying both sides of (21) with the nabla operator, ∇, where we get

$$\nabla \cdot w = \nabla \cdot v + \nabla^2 q. \tag{24}$$

As v has zero divergence, this expression simplifies to

$$\nabla \cdot w = \nabla^2 q, \tag{25}$$

which we call the Poisson equation. Following the formulation of Shi and Yu [10], the decomposition can equivalently formulated as a least squares minimization problem, where

$$\min_q \iint_\Omega ||\nabla q - w||^2 \, d\Omega, \tag{26}$$

and the solution for this optimization problem will be the solution of the Poisson equation, $\nabla^2 q = \nabla \cdot w$, which can be expressed as a sparse linear system, where sophisticated optimization algorithms, such as the conjugate gradient method or the methods discussed in Section 3.4 can be applied to solve it efficiently. A more general statement of the Helmholtz Theorem is shown in Appendix B.

Figure 10: An artistic simulation with a slightly exaggerated effect of vorticity confinement. The source of this image is found in the corresponding section.

3.6 Additional Realism

In artistic applications where a very high degree of realism is a requirement, the resolution of the simulation grid has to be raised significantly. As this operation introduces a computation time overhead of $\Theta(n^d)$ magnitude for a d-dimensional simulation, it is often implausible to increase it to the desired level. In this section we give a brief overview of advanced techniques such as *Vorticity Confinement* and *Wavelet Turbulence* to further enhance the realism of the simulation with preferably less cost than simply increasing the resolution would take. To be able to simulate the flow of gas type substances properly, the modeling of *Buoyancy and Convection* forces are discussed here.

3.6.1 Vorticity Confinement

After sufficient amount of careful observation, one may notice that the most common flow type of fluids is when particles run in parallel layers, with no disruption between them. These type of flows are called *laminar flows*, or may also be referred as *streamline flows*. When force is applied to a piece of fluid, the pressure and

velocity properties may change rapidly in both space and time, resulting in a highly irregular, chaotic flow type called *turbulent flow*. A good example for this is when low viscosity fluids contain rotational flows of varying sizes. This is what we call *vorticity*, a high frequency phenomenon that is expected to be captured in every decent simulation environment.

At this point, the advantages of the finite grid representation and a stable advection formula should be clear. Though, it should be noted that the approach has its own drawbacks - a fluid that's simulated on a coarse grid will dampen faster than it would do in reality, causing small scale vorticity to disappear from the simulation. We give a detailed overview of this phenomenon in Section 3.6.3. Fedkiw et al. [11] proposed a method to reinject the lost energy into the system, thus minimizing the error due to the nature of the solution.

The vorticity of the velocity field u is given by

$$\omega = \nabla \times u. \tag{27}$$

By computing the gradient of ω we get a vector field that points towards the center of the vortex:

$$\mu = \nabla |\, \omega|. \tag{28}$$

Because ω contains information about the size of the vortex and the direction to the center, we only wish to use the latter, therefore it is plausible to normalize μ by dividing by the vector lengths

$$\Psi = \frac{\mu}{|\mu|}, \tag{29}$$

so Ψ points to the center of the vortex and u towards the axis of the rotation. To be able to increase the strength of the rotation, their cross product will be taken, multiplied by an ε parameter that helps to adjust the weight of the vorticity confinement term. We now have the full expression

$$F_{vc} = \varepsilon \, (\Psi \times \omega), \tag{30}$$

that will be a term to be added to the external forces at any given time. Note that this method is not physically based, and is mainly used to help the modeling of high frequency turbulent flows as shown in Figure 10[4].

3.6.2 Buoyancy And Convection

It is well-known that hot and cold fluids and smoke behave in different ways - the hot smoke of a cigar rises, cold air sinks due to the fact that colder particles are heavier, causing them to be pulled downwards by the gravitational force. Temperature and density differences therefore have effect on the velocity field. These effects can be observed in natural phenomena, such as oceans, rivers, lakes, or a steaming cup of coffee, and can be modeled efficiently by replacing the gravitational force F_g with the following term:

$$F_b = (-\kappa\rho + \sigma(T - T_{amb}))\,\hat{\mathbf{j}}, \tag{31}$$

where κ and σ are constant scale factors for mass and temperature, ρ is smoke density, T and T_{amb} are the temperature of the fluid or smoke and the ambient temperature, respectively, and $\hat{\mathbf{j}}$ denotes the unit vector for the vertical direction. This term will behave as an external force field, and should be applied to the F_{ext} term of (1) by addition. The fine tuning of the scale factors will modify the behavior of the buoyant force: the first term shows how strongly heavier particles will be gravitated downwards, the second is the relative amount of lifting force that is exerted on the particles that are hotter than the environment. This lifting force term of the formula reduces to zero for $T = T_{amb}$ substitution. Please note that by plugging appropriate viscosity values into the Navier-Stokes equations and using buoyant forces, not only fluids but most gas type substances, like air, helium and nitrogen can be simulated.

3.6.3 Wavelet Turbulence

The most visually appealing effects in fluid and smoke simulations are arguably the accurate modeling of turbulent flows. However, rendering photorealistic quality video footage requires tremendous amount of resources. With the use

[4]The image is taken from a 2012 reel video, which is a property of Fusion CI Studios.

of the finite element method, the Navier-Stokes equations are solved on a discrete grid, therefore qualities like the density of the fluid are only known in these intersection points. For any grid with finite resolution, there is a Nyquist limit [12], therefore a maximum frequency value exists which the method can reconstruct. Above these frequencies no information can be extracted, making it a straightforward choice to increase the grid resolution to the highest possible value.

In a 3D case, however, the required operative memory and computation time will be of $\Theta(n^3)$ magnitude, therefore home computers are left out of the enjoyment of rendering high frequency turbulent flows, which are thus meant to be for render farms only. In [13] a novel approach is presented, exhibiting Kolmogorov's theory of turbulence and the Wavelet Transform to generate and inject turbulent flow informations into previously existing, low resolution simulations. Kolmogorov's theory [14] states that fluid flows are composed of larger and smaller swirls, which we refer to as *eddies*. Large eddies travel through the velocity field, where they get both compressed and stretched from different directions, therefore they get broken up into eddies of smaller size. We call this phenomenon *forward-scattering*. The opposite of this can also happen, where two eddies of smaller size are compressed together, disappearing, and thus forming a bigger eddy. Kolmogorov has given a probabilistic model, which, through the evolution of the fluid flow in time, is able to predict the kinetic energy distribution of the domain. This is referred to as the *"five-thirds" law*, describing that as we observe higher frequency components of the velocity field, the kinetic energy distribution will decrease obeying a slope that has $-5/3$ steepness.

In possession of the knowledge of where the fine details are lost and the cutoff point at the Nyquist frequency limit, we know exactly where and what frequency data should be synthesized. In previous research works, to synthesize new turbulent flows, the weapon of choice was the Fourier Transform, where a product is computed between the given signal and a signal of $\langle \cos(\omega t + \varphi), \sin(\omega t + \varphi) \rangle$ basis. The method excels at extracting frequency information from the signal, however, since it uses a signal of periodic basis, it lacks time localization, meaning that it is

unable to tell where the frequency components were found in the input signal. The Wavelet Transform uses a basis of $\langle\{\psi_{d,s}(t){:}d, s{\in}\mathbb{Z}\}\rangle$, different scalings and translations of a mother wavelet signal that has finite support, therefore this method is able to give an acceptable trade-off between time and frequency localization, thus making multiresolution analysis possible, giving us the opportunity to examine where exactly the extracted frequencies are found. Using the definition of the transform

$$\begin{aligned} [W_\psi f](d,s) &= \left\langle f(t), \psi_{d,s}(t) \right\rangle = \int_{-\infty}^{\infty} f(t)\, \overline{\psi_{d,s}(t)}\, dt, \\ \psi_{d,s}(t) &= \frac{1}{\sqrt{s}}\, \psi\left(\frac{t-d}{s}\right), \\ \operatorname{supp}\left(\psi_{d,s}(t)\right) &< \infty, \end{aligned} \tag{32}$$

alongside with Kolmogorov's *five-thirds law*, a new incompressible velocity field may be synthesized from a low resolution result, which will now contain details of the turbulent flows. The results are demonstrated in Figure 11.

Figure 11: The top left image shows a low resolution simulation, and on the right is the one after the wavelet reconstruction of the turbulent flows. Source: [13].

4 Higher Order Advection Methods

In Section 3.2 we have discussed a first order method to compute the advection term by tracing the velocity field back in time with a straight line by bi- or trilinear interpolation. This technique is often referred to as first order *Semi-Lagrangian advection*. As discussed in [15] and [16], a higher order approximation can be obtained by tracing back in a curved line and using higher order interpolation schemes. This approach requires more calculations, but is generally regarded a worthy trade-off, since both numerical and visual results (Figure 12) show remarkable improvement over the first order method. In this section we discuss two unconditionally stable second order methods, the *Back and Forth Error Compensation and Correction Advection* (BFECC) and the *MacCormack Advection* schemes.

Let ϕ^n denote the state of the velocity field at time step n, and $A(\cdot)$ the forward, and $A^{-1}(\cdot)$ the backwards advection operator. To calculate the error of an advection method, by using the forward advection operator we get $\hat{\phi}^{n+1} = A(\phi^n)$ and $\hat{\phi}^n = A^{-1}(\hat{\phi}^{n+1})$, then the error is obtained as $\hat{\phi}^n - \phi^n = 2e$. We can now use this valuable knowledge to adjust the initial setup as $\bar{\phi}^n = \phi^n - e$ and afterwards, the final step will be $\phi^{n+1} = A(\bar{\phi}^n)$. Both discussed methods will take advantage of this observation.

4.1 Back and Forth Error Compensation and Correction Advection

The definition of BFECC is as follows:

1. $\hat{\phi}^{n+1} = A(\phi^n)$
2. $\hat{\phi}^n = A^{-1}(\hat{\phi}^{n+1})$
3. $\bar{\phi}^n = (3\phi^n - \hat{\phi}^n)/2$
4. $\phi^{n+1} = A(\bar{\phi}^n)$

Note that this method requires three advection calculations, as opposed to one in the standard Semi-Lagrangian method. This is the price we have to pay for a higher order solution, though it is easy to see that increasing the resolution (k) and using Semi-Lagrangian method would increase the computation time in the order of $\Theta(k^2)$

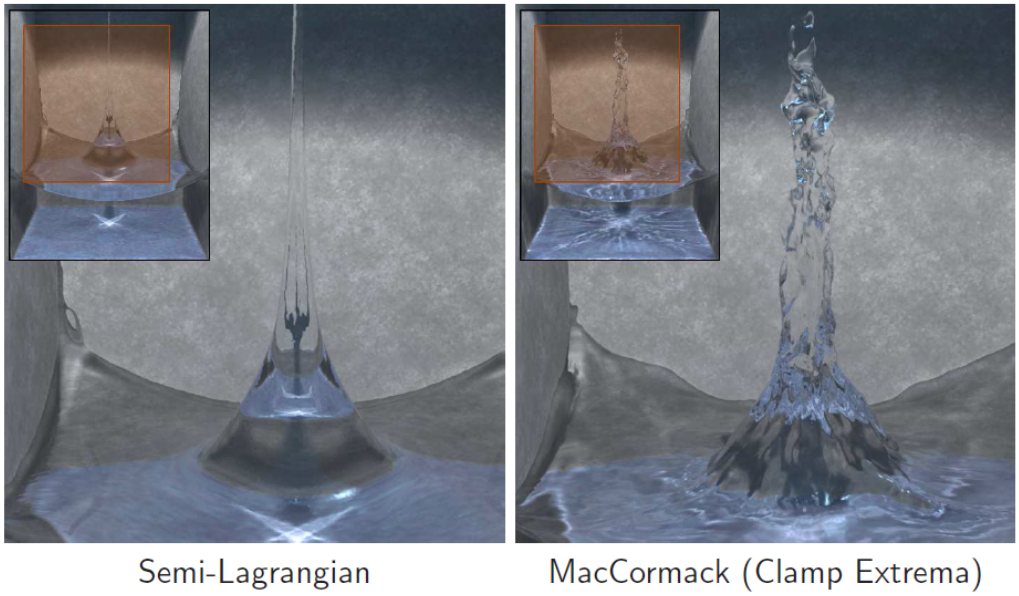


Figure 12: The image shows the striking difference between a first and a second order advection method. Source: [16].

for 2D, and $\Theta(k^3)$ for 3D simulations, while the costs of higher order advection methods only increase linearly.

4.2 MacCormack Advection

This advection scheme is defined by the series of the following steps:

1. $\hat{\phi}^{n+1} = A(\phi^n)$
2. $\hat{\phi}^n = A^{-1}(\hat{\phi}^{n+1})$
3. $\phi^{n+1} = \hat{\phi}^{n+1} + (\phi^n - \hat{\phi}^n)/2$

It is clear that only two advection operators have to be evaluated, though extreme values have to be taken care of by clamping the final velocity field between the minimum and maximum values found in the initial field, ϕ^n. This advection method is still significantly cheaper than BFECC, and as discussed in [16], it also promises notable improvement on error values.

5 Visualization of The Flow Field

In Section 1.3 we have shown a wide variety of applications of computational fluid dynamics. In this section, we discuss how 2D and 3D fluid flows are visualized. The Navier-Stokes equations give us informations on how the velocity of fluids change over time, therefore the simplest and most intuitive approach one could come up with would be to visualize the *velocity field* itself. However, we don't know of any kind of application which would make use of that directly. For instance, for artistic users almost always prefer to show the fluid itself moving in time, therefore the choice is almost always to visualize the *density field*, where the color of a pixel will be proportional to the magnitude of the density field at a given region. Intuitively, in an Eulerian simulation, it means that we have a little pool of water, in which we inject a few drops of ink and track the trajectory of these particles as they are advected by the velocity field.
Wind tunnel test personnel are indeed interested in the evolution of the *velocity field* over time, however, displaying all of it would often show too much information. The solution is often to sow a healthy number of particles onto the velocity field, simulate the path of them just like artists do, but instead representing them only with their color, speed magnitude and directional information is also often visualized. Simulations with directional information can be viewed in Figures 4a and 6, and color-coded speed magnitude is shown in Figures 4b, 5.

In a two-dimensional simulation, only one slab of data is visualized, therefore, regardless of which quantity is relevant, it can be visualized directly (Figure 13). The three-dimensional case requires special care, as if we use a grid of N^3 resolution, we can imagine it as N two-dimensional slabs put on top of each other. There is a vast amount of literature on volume rendering techniques to visualize 3D density data, including Marc Levoy's ray marching algorithm [17], where rays are cast through every pixel into the volume, sampled along their way, and shaded based on trilinear interpolation of nearby data. In order to ensure that the user only sees what is in sight of the camera, a composition phase is also applied. The composition step calculations are carried out with respect to the rendering equation [18] [19].

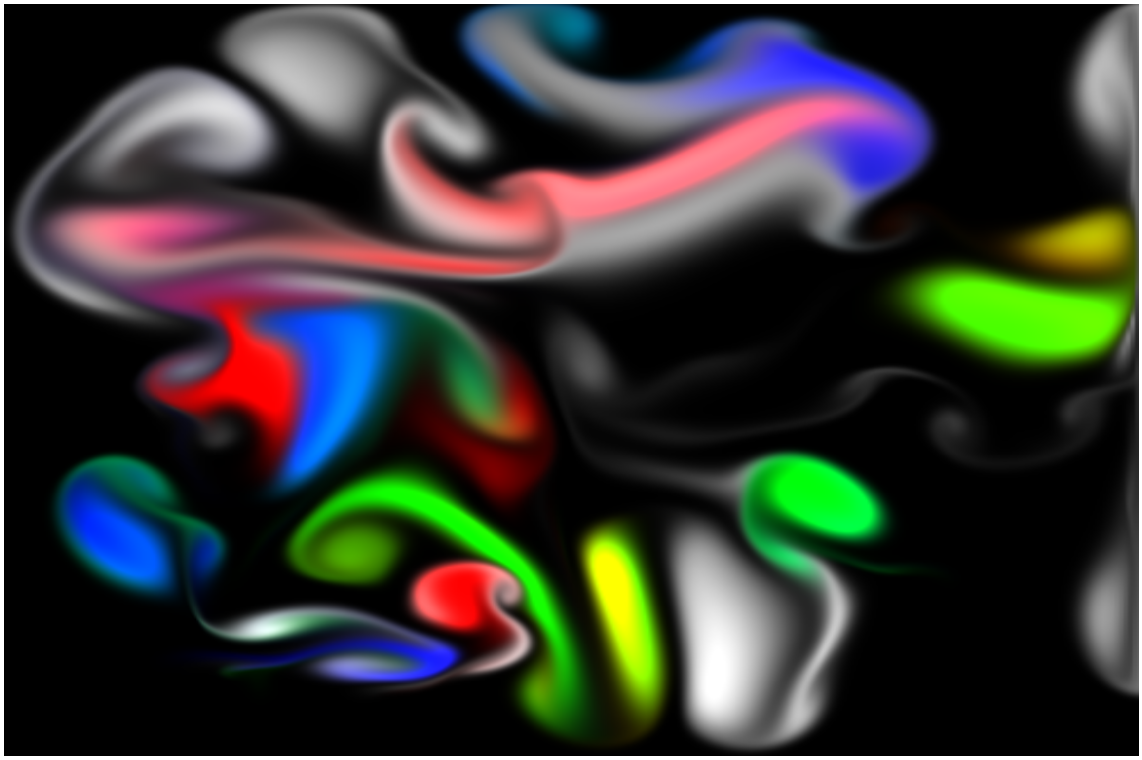


Figure 13: Visualization of the density field in two dimensions.

6 Methods for Fluid Control

It is quite common in the world of motion picture production that a scene doesn't work out as well as expected. It is also not uncommon to reshoot a scene where the actors only talk to each other, but what to do if an explosion, a gunshot, or a collapsing building just does not deliver the artist's intentions? There are material costs, deadlines are always tighter than they should be, and simply enough, not every kind of scene can be refilmed an infinite times. A building may only be blown up once, and if something goes wrong, it is impossible to retry it in the same manner as a dialogue scene. This is where visual artists come into the picture by using fluid and smoke simulation software to spice up and make good scene great, or even fix a failed attempt by adding virtual detail into previously recorded, real-life footage.

Using quality simulation software, it is possible to create a virtual scene consisting of objects with arbitrary geometry and material attributes, and observe how they would behave in reality. To make footage that looks convincingly real,

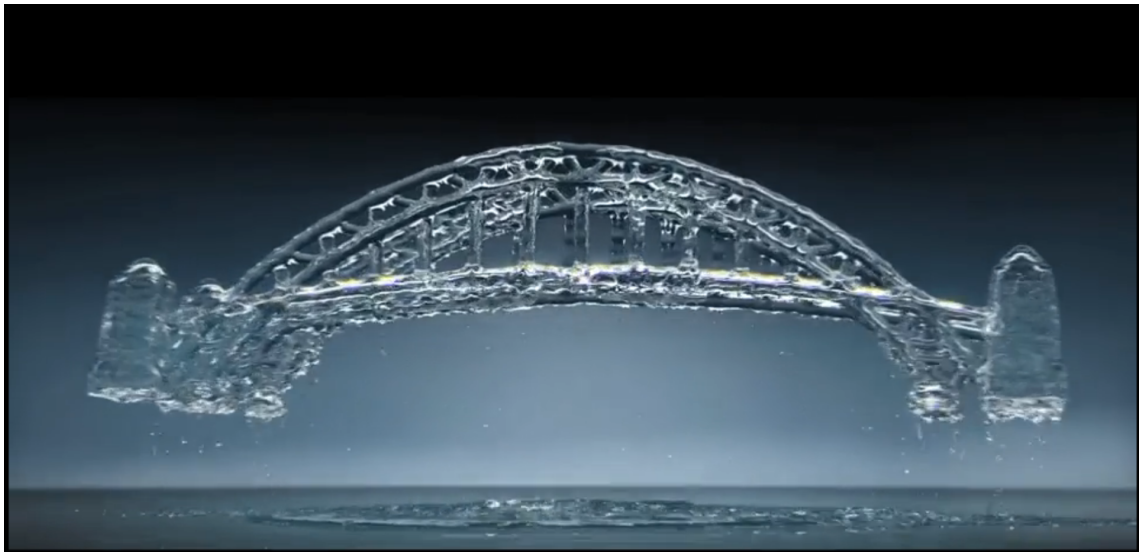


Figure 14: A creative use of fluid control in the motion picture industry. The source of this image is found in the corresponding section.

the software is expected to be based on the laws of physics. However, artists are known to have visions and images in their heads, which they would like to stick to. In *Terminator 2: Judgment Day*, Robert Patrick played the role of the famous antagonist, the infamous *T-1000*, a fictional nanomorph robot made of liquid metal, which has rapid shapeshifting abilities, being able to morph into any object. As simulation software are by principle physically based, and since probably no one has ever seen a puddle of liquid metal suddenly form a human shape by itself, it would make a quite daunting, if not impossible task to simulate such phenomena. In this case, realism is only a burden for the artists, keeping them from expressing themselves. What if the visual artist has a definite vision of what shape the flame should take when firing a flamethrower, and physical reality would look different? A similar example can be viewed in Figure 14[5].

The simplest answer would be to drop the physically based approach and move the objects in the simulation in a way that the artist finds pleasant. Though, even little changes to the natural laws results in highly unconvincing footage, from which no one would believe was real. This is quite an ambivalent problem: we require fluids and smoke to act as the laws of nature would order them to, but we

[5]The image is taken from the 2012 reel video of Fusion CI Studios.

also have our very own requirements on how they should behave. Most of the time, artists have in vision of an image, a shape, or a physical object that the viewer should be resembled. Mathematically speaking, there is a given density field at $t = 0$ that is to converge to a target density in time, while retaining only natural movement in the fluid domain. To make the fluid domain obey us, we change the laws of nature temporarily, for instance by means of using an external force field. A good intuition for this is to imagine small spoons of infinitesimal size, which we use to stir the fluid in the chosen directions, doing it several, or even thousands of places at once. To measure how good our job was done with the stirring, we measure the difference between the obtained and the target density field, and also how natural the flow of the fluid felt. The problem is that no one knows how to rigorously define ''how natural a flow feels''.

Mathematical algorithms for most problems can be objectively compared as they have their own, well-defined error metric. Comparing them is as simple as choosing the one that offers the best time-quality trade-off for our needs. However, there is no metric that would define how natural a fluid flow is, we are left to judge by what *seems* and what *feels* to be natural for us.

We have discussed a different problem in Section 3.6.3: as we compute the velocity and pressure fields on a finite grid, there is a limit for the highest frequency band it will be able to reconstruct, which depends on the resolution of the grid, and is called the Nyquist-limit. Therefore, we are unable to use our spoons arbitrarily, we are restricted to a finite band of frequency spectrum, since above the Nyquist-limit, the informations on the external force field are not just suppressed as the Nyquist-Shannon law suggests, but forward scatterings are more likely to happen, meaning that larger eddies will break into two half-sized ones, causing numerical dissipation, which will result in a very unconvincing flow footage where at different rates, density disappears into thin air.

Long-range forces

If some part of the fluid domain has excess density, meaning that ρ_j, the density at point j is higher than the target density ρ_j^t that is defined there (given by the

input distribution), the region will transport density by exerting force towards the direction of its neighborhood for those who have lower density than the target. As in physics, the exerted force weakens as the distance between the two points increases, therefore to maintain energy conservation, some kind of falloff characteristic has to be introduced. Based on the work of [10], this will be the definition of the long-range force field that is responsible for arranging the density field so it can start converging to the target on a macro level:

$$F_i^L = \sum_j \frac{\left[\rho_j - \rho_j^t\right]^+ \mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^\alpha}, \tag{33}$$

where r_{ij} is a vector that points from grid point i to j and $||\cdot||$ is on $L^2(\mathbb{R}^n)$. Most physical phenomena such as light propagation, or forces like gravity and electric force have a squared falloff with increasing distance, therefore it is not surprising that the choice of $\alpha = 3$ gives visually pleasing results. Constructing this force field following the exact definition with $O(ij)$ interactions is undoubtedly too demanding. Luckily, it isn't necessary, since as we increase the distance, every sensible choice of α will make the force decay in an at least quadratic manner, we can safely define a maximum distance that is to be computed from every point.
In our version we have also relaxed the evaluation of the long-range forces to regions that have nonzero target density,

$$F_i^L = \sum_j \frac{\left[\rho_j - \rho_j^t\right]^+ \mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^\alpha}, \quad \forall j : \ \rho_j^t > 0, \tag{34}$$

to reduce its cost to be proportional to only the volume where the target distribution is nonzero, as opposed to the original method, where it is evaluated in the full simulation domain.

Stability condition and short-range force design

The density change between two time instants is given by

$$\Delta\rho_t = \rho_{t+1} - \rho_t, \tag{35}$$

and the amount of density change we would like to apply is

$$\Delta\rho_t^a = \rho^t(x,t) - \rho(x,t), \tag{36}$$

therefore our carefully designed force field has to satisfy

$$\Delta\rho_t = \rho_{t+1} - \rho_t \quad \Rightarrow \quad \Delta\rho_t = \lambda\Delta\rho_t^a, \tag{37}$$

where λ is a constant on $[0,1)$ to obtain a stable simulation.
Now a short-range force field is to be constructed to carve out the fine details of the target distribution locally. In [10], Shi and Yu have advised that the solution of the following global optimization problem would make a desirable field:

$$\begin{aligned}\min_{F_{t+1}^S} \quad & c_1 \sum_x \left(\Delta\rho_t(x) - \lambda\Delta\rho_t^a(x)\right)^2 + c_2 \sum_x \left(DIV(x)\right)^2 \\ & -c_3 \sum_x \left(\frac{F_{t+1}(x)}{||F_{t+1}(x)||} \cdot \frac{\nabla\rho(x)}{||\nabla\rho(x)||}\right)^2 \\ & -c_4 \sum_x \sum_{y\in N(x)} \cos(\theta_{t+1}(y) - \theta_{t+1}(x)) \\ & +c_5 \sum_x \sum_{y\in N(x)} \left(\left|\left|F_{t+1}^S(y)\right|\right| - \left|\left|F_{t+1}^S(x)\right|\right|\right)^2\end{aligned} \tag{38}$$

where every function depends on x, the position given in space. The $c_1, \ldots, c_5$ coefficients are used to assign different weight values to the terms. We attempt to give an intuitive interpretation of the formula: the *first term* is responsible to ensure that the difference between the current and the target density is minimal by maximizing the amount of density change in excess density areas, and minimizing it at areas that match the target density well. The velocity field has to be divergence-free throughout the simulation and control process, therefore in the *second expression*, substituting $\nabla \cdot u_{t+1}(x)$ into $DIV(x)$ makes a plausible choice. However, measurements show that we obtain better results by using the long-range force field, evaluate the projection step afterwards to keep the divergence-free property, then giving $\nabla \cdot F_{t+1}^S(x)$ will also yield correct results. The *third term* is the dot product between the normalized direction vectors of the force field and the gradient of the density field. Minimizing this is absolutely important to achieve natural looking flows: the applied forces

will be similar to the natural flows in the fluid domain for any given time. As $\cos(\theta_{t+1}(x))$ represents the orientation of the short-range force field in point x, the *fourth and fifth expressions* are to minimize the amount of directional variance and the magnitude of the applied short-range force field - the less intrusion, the better.

The most important attribute of the new approach would be not to use F^S_{t+1} due to its computational costs, therefore omit carving out some of the fine details, but design a different, cheaper force field that is able to mobilize large amounts of density towards the target density while still retaining a realistic looking flow.

Biased diffusion

In [10], the idea of a new diffusion scheme has also been raised, where diffusion, instead of its original role of smoothing the discrepancies of the density field, would rather help the convergence of the image through acting as a stronger distributive force in regions of poor convergence. A slight change to the diffusion formula,

$$\nu\nabla^2 u \quad \Rightarrow \quad \nu\nabla^2 \left[\rho^t - \rho\right]^+, \tag{39}$$

promises improvement at carving out the fine details of the pictures. Our experience shows that using this method excels at controlling the density towards the feature points of the target, therefore it is a valuable tool to obtain convergence for complex setups, but at the cost of losing some realism on the fluid flow as the diffusion process will be conspicuously asymmetric. If target density distribution is of high variance, using biased diffusion helps convergence substantially. If maximizing the realism of the fluid flow is a more important requirement, then omitting this technique is the right decision, alongside with using low-variance target distributions.

Forces after convergence

Deciding what should happen after the target density is reached is also a very important task. While it would be a straightforward to, for example, set a very high to ν to ‘‘freeze’’ the fluid in the convergent subdomains, the results will remain

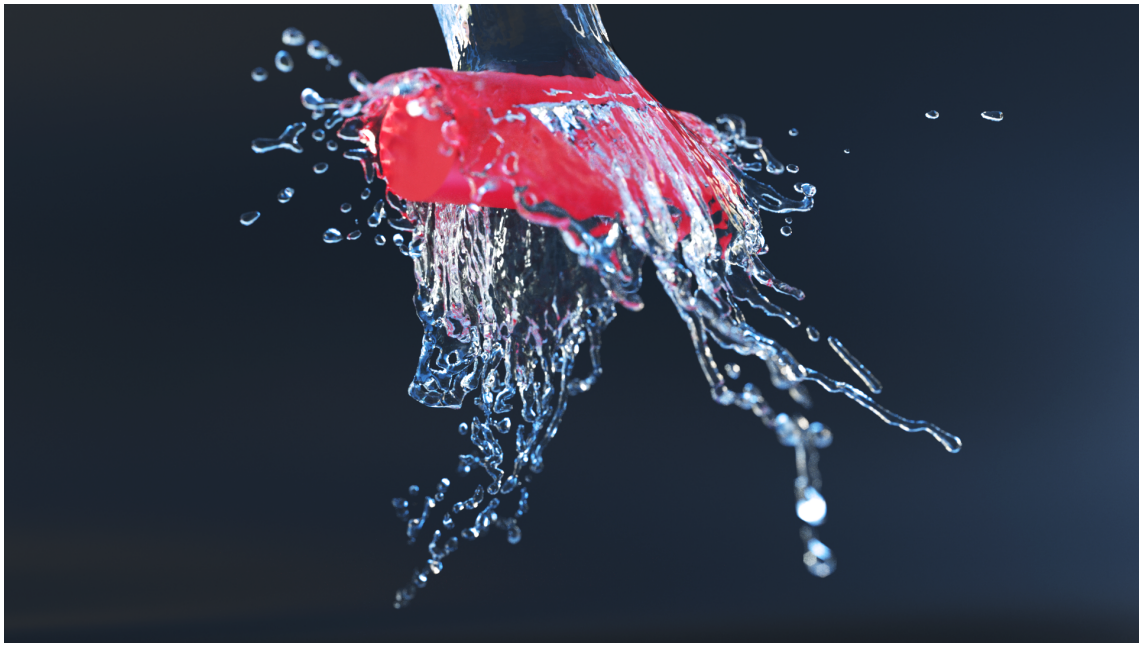


Figure 15: A scene made by Niklas van Bonn exhibits the use of boundary condition computation and the depth of field effect in the rendering pass.

correct, though can not be expected to look very lifelike, making it a simple, effective, but generally unconvincing solution.

Here we try to address the shortcomings of this approach by introducing a speedup factor ψ, which increases the velocity of the fluid at regions where convergence is reached with respect to an error threshold, ϵ. *It may sound quite counterintuitive: why speed up the fluid at regions where it already looks correct?* It would be reasonable to say that it is the exact opposite of what should happen. On a microscopic level, freezing the fluid domain would always give the best choice: if we have only a few points in space, freezing them by assigning a very high kinematic viscosity value upon reaching the target distribution will ensure that they will remain in the correct state all the time. Let's consider a simple example on a macroscopic level, where we have a fluid domain of significant size where the target distribution can be reached by going through a narrow choke point. At this region, the fluid will start freezing, therefore it will prevent any further fluid movement, making it impossible to get density through. This scenario will not be restricted only to narrow choke points: for almost every practical case on closed shapes, the closer we are to the state of convergence, the higher the probability for this to happen.

Intuitively, freezing the fluid can be associated with the ‘‘after you’re done, just stop and rest’’ behavior, as opposed to speeding up the velocity, which would mean more like ‘‘after you’re done, start helping others’’. As our results show, this approach will not just preserve fluid movements after the target density is reached, but will help increasing the speed of the convergence as well. We will use ψ in the final setting of our new Navier-Stokes equations for fluid control in (41).

Cleaning it up

It often occurs that upon reaching convergence, excess density remains around the outer side of the boundary of the target distribution. To clean up these details around the boundary, an additional force field may be applied that sucks the density back into the domain of the target distribution. The use of this cleanup force field is entirely optional, and it would consist of vectors that are oriented from points that contain density but have zero target density and are near the boundary to every other point that has nonzero target density. Formally:

$$F_i^C = \sum_{j:\rho_j^t>0} \frac{\rho_j \, \mathbf{r}_{ij}}{||\mathbf{r}_{ij}||^\alpha}, \quad \forall i : \ \rho_i^t > 0. \tag{40}$$

Depending on the target density distribution type, Neumann- or Dirichlet-type boundary conditions may be also applied.

The final form

Putting it all together, the proposed form of the Navier-Stokes equations for controlling the fluid is as follows:

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = \psi \left(-\frac{1}{\rho}\nabla p + \nu \nabla^2 u \right) + F^L + F^C + F_{ext} + cF_{vc},$$

$$\psi = \begin{cases} 1+\delta, & \text{where } \left|\rho^t - \rho\right| < \epsilon, \\ 1, & \text{elsewhere,} \end{cases} \tag{41}$$

$$\frac{\partial u}{\partial n}\Big|_{\partial\Omega} = \begin{cases} -\dfrac{\partial u}{\partial t}, & \text{inside the shape,} \\ 0, & \text{outside the shape,} \end{cases}$$

where c is a constant factor to control the magnitude of the control force field (and therefore the speed of convergence) and δ is small value to keep the fluid in motion after convergence when the actual density is equal to the target density within error threshold ϵ. This technique is capable of guiding the fluid towards the target distribution in real-time, and has the following properties:

- On an average home computer (as of 2012), both the fluid simulation and the control algorithm can be run in parallel as they take 18 and 14 milliseconds at most respectively on an 512^2 grid with 20 Jacobi iterations, therefore it is a real-time solution,
- It yields remarkably fast convergence speeds,
- It is to be used with target distributions of low-variance for a high measure of realism, or,
- It is to be used on more complex, higher-variance distributions with the aid of biased diffusion at the cost of less realism and more computational demands,
- Relying only on local data, it can be extended to simulations of any dimension with favorable amount of computational overhead.

Out test results using Neumann-type boundary conditions without control force fields show that it is unreasonable to expect a desirable degree of convergence (Figure

16). This method is also unable to control any amount of density which is not already inside the target distribution domain. The results using the new method were rendered in real time and are shown in Figure 17, which, using roughly the same amount of density, was able to achieve a high measure of convergence in the same amount of time.

7 Conclusions

In this thesis, we have given a brief overview of the mathematical background and applications of computational fluid dynamics. The main goal of the review was twofold: to show that the rules that govern the motion of fluid and gas type substances can be reduced to formula with three simple terms, and to use this knowledge to build a framework that is able to solve the Navier-Stokes equations effectively on parallel computer hardware.

We have also given a statement of the fluid control problem, where an initial state and a target distribution is given for the fluid, with the intention that the fluid would sooner or later take the form of this distribution. A solution to this problem is an external force field that is changing in time, describing the forces that have to be exerted on the fluid to take a given shape. It is desirable that this force field makes the fluid converge in a short amount of time, with the slightly ambivalent requirement of giving a realistic flow in the meantime, even if it is highly unlikely that a bowl of water would suddenly take a form of natural objects.

Current state-of-the-art methods use sophisticated mathematical methods from fields such as control theory or optimization, giving impeccable results, though, at the cost of 5 to 7 minutes of computation time per frame. We propose a new approach to control fluids, consisting of the sum of different force fields. We take advantage of the idea of using long-range force field and biased diffusion to obtain coarse- and fine-scale convergence, which are both based on prior research work. Instead of using a costly short-range force field to finish the job, we introduce a speedup factor, which, instead of freezing the fluid, raises the velocity of the

convergent subdomains to ensure that density transfer is possible between their neighborhoods. In order to make the algorithm more robust to receive incoming densities from any direction, we also introduce a new force field for cleaning up the excess density outside the domain of the target distribution.

The proposed method introduces restrictions to the target distribution that still retain usability for a variety of applications. The resulting method is able to solve the fluid control problem effectively, even on average home computer systems.

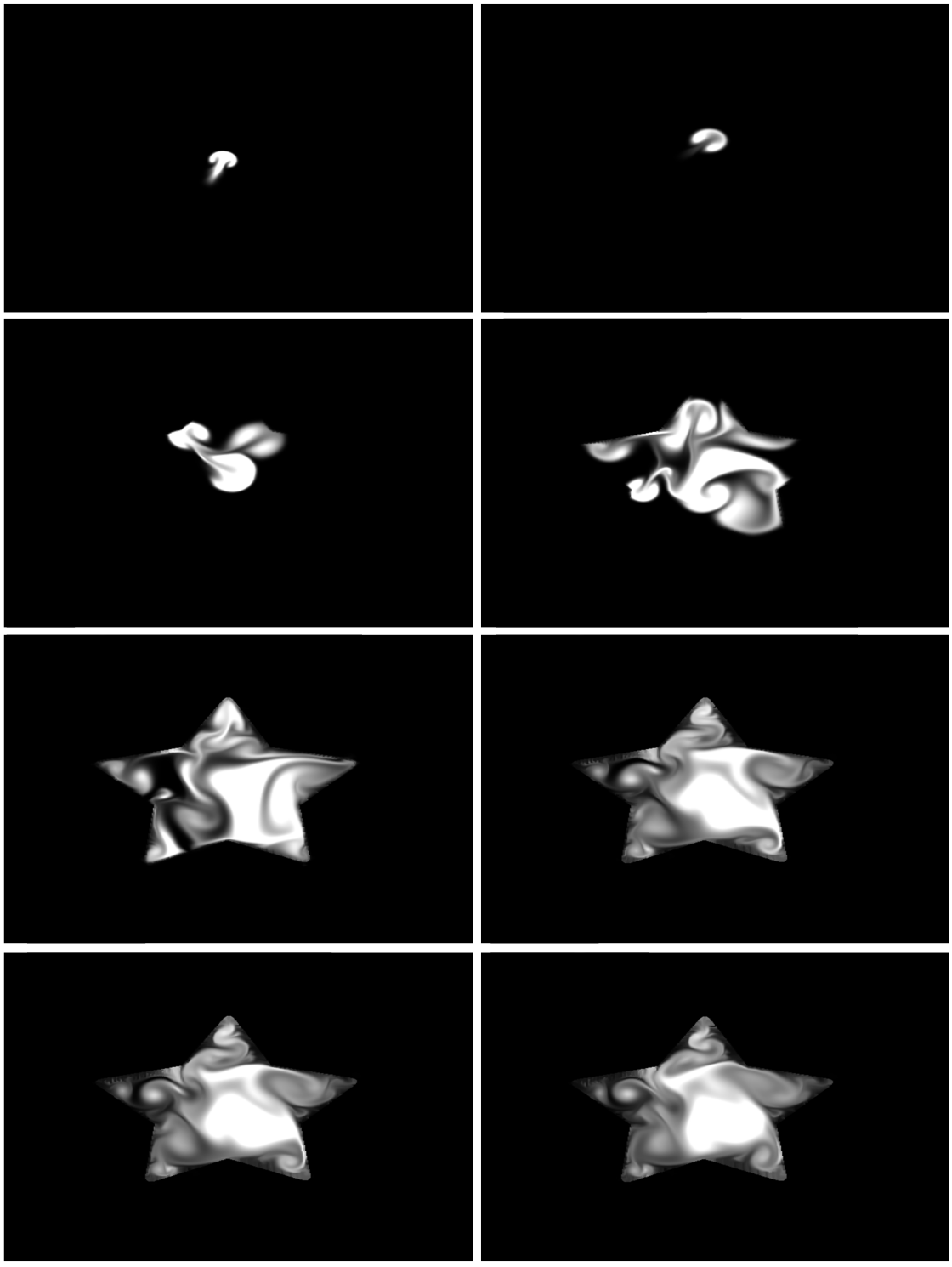


Figure 16: An imitation of the solution to the fluid control problem where no control forces, only boundary conditions are used. The example shows that even if the fluid is locked inside the domain of interest, it is highly unlikely that it would suddenly flow into the shape of a star.

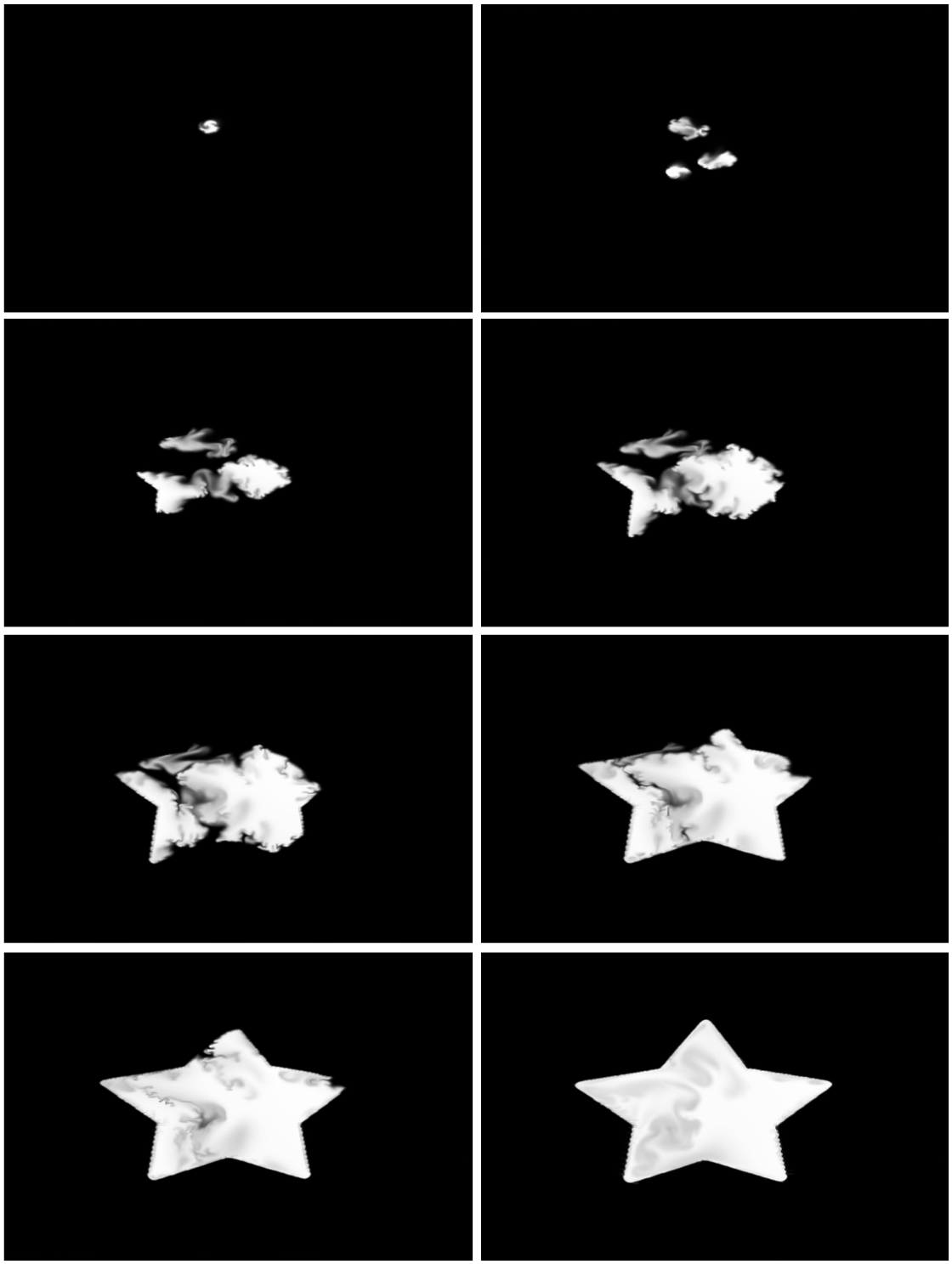


Figure 17: The proposed method runs in real time, provides good coverage of the target density, and is aware of the regions of poor convergence, which are constantly helped out by nearby regions. Roughly the same amount of density is used as in Figure 16.

References

[1] C. L. M. H. Navier, Mémoire sur les lois du mouvement des fluides. Mém. Acad. Sci. Inst. France 6, 389-440., 1822.

[2] J. Stam, ‘‘Stable fluids,’’ in Proceedings of SIGGRAPH 99, Computer Graphics Proceedings, Annual Conference Series, pp. 121--128, 1999.

[3] K. Ralovich, R. I. Ionasec, V. Mihalef, P. Sharma, B. Georgescu, A. Everett, N. Navab, and D. Comaniciu, ‘‘Computational fluid dynamics framework for large-scale simulation in pediatric cardiology,’’ Computational Biomechanics for Medicine VI (CBM6) MICCAI Workshop, 2011.

[4] L. Itu, P. Sharma, K. Ralovich, V. Mihalef, R. Ionasec, A. Everett, R. Ringel, A. Kamen, and D. Comaniciu, ‘‘Non-invasive hemodynamic assessment of aortic coarctation: validation with in vivo measurements,’’ Annals of biomedical engineering, vol. 41, no. 4, pp. 669--681, 2013.

[5] I. Newton, T. Leseur, F. Jacquier, and J. Wright, Philosophiæ naturalis principia mathematica. No. v. 1-2 in Philosophiæ naturalis principia mathematica, ex prelo academico, typis A. et J.M. Duncan, 1822.

[6] R. Bridson, Fluid Simulation. Natick, MA, USA: A. K. Peters, Ltd., 2008.

[7] A. Einstein, Zur Elektrodynamik bewegter Körper. Johann Ambrosius Barth, 1905.

[8] J. W. Demmel and M. T. H. Y, ‘‘Applied numerical linear algebra,’’ in Society for Industrial and Applied Mathematics, SIAM, 1997.

[9] A. Chorin and J. Marsden, A Mathematical Introduction to Fluid Mechanics. Texts in Applied Mathematics, Springer-Verlag, 1993.

[10] L. Shi and Y. Yu, ‘‘Inviscid and incompressible fluid simulation on triangle meshes,’’ JOURNAL OF COMPUTER ANIMATION AND VIRTUAL WORLDS, vol. 15, pp. 3--4, 2004.

[11] R. Fedkiw, J. Stam, and H. W. Jensen, ‘‘Visual simulation of smoke,’’ in ACM SIGGRAPH 2001, pp. 15--22, 2001.

[12] C. Shannon and I. of Radio Engineers, Communication in the Presence of Noise. Institute of Radio Engineers, 1949.

[13] T. Kim, N. Thürey, D. James, and M. Gross, ‘‘Wavelet turbulence for fluid simulation,’’ in ACM SIGGRAPH 2008 papers, SIGGRAPH ’08, (New York, NY, USA), pp. 50:1--50:6, ACM, 2008.

[14] A. N. Kolmogorov, ‘‘On degeneration of isotropic turbulence in an incompressible viscous liquid,’’ in Dokl. Akad. Nauk SSSR, vol. 31, pp. 538--540, 1941.

[15] T. F. Dupont and Y. Liu, ‘‘Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function,’’ J. Comput. Phys., vol. 190, pp. 311--324, Sept. 2003.

[16] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac, ‘‘An unconditionally stable maccormack method,’’ J. Sci. Comput., vol. 35, no. 2-3, pp. 350--371, 2008.

[17] M. Levoy, ‘‘Efficient ray tracing of volume data,’’ ACM Trans. Graph., vol. 9, pp. 245--261, July 1990.

[18] J. T. Kajiya and B. P. Von Herzen, ‘‘Ray tracing volume densities,’’ in Proceedings of the 11th annual conference on Computer graphics and interactive techniques, SIGGRAPH ’84, (New York, NY, USA), pp. 165--174, ACM, 1984.

[19] J. T. Kajiya, ‘‘The rendering equation,’’ in Proceedings of the 13th annual conference on Computer graphics and interactive techniques, SIGGRAPH ’86, (New York, NY, USA), pp. 143--150, ACM, 1986.

A Derivation of the Navier-Stokes equations

A few notes before the derivation takes place:

Dynamic or absolute viscosity gives the resistance of the fluids against stress, denoted by μ, and has SI unit $[Pa \cdot s]$,

Kinematic viscosity is given as $\nu = \mu/\rho$ with SI unit $\left[m^2 \cdot s^{-1}\right]$,

The material derivative gives the rate of change of a physical quantity that is placed into a velocity field that changes in time and space. Since the particles put inside a fluid volume will be carried with the velocity field, the directional derivative also has to be taken into account. Therefore, this useful tool serves as a transformation between the Eulerian and Lagrangian viewpoint and is given by the sum of the Eulerian and convective acceleration. It is also referred to as convective, hydrodynamic, Stokes, or total derivative.

Definition: $\dfrac{Du}{Dt} = \dfrac{\partial u}{\partial t} + (u \cdot \nabla)u$

We will use Newton's second law [5] as a starting point of our derivation,

$$F = ma \tag{42}$$

we first divide both sides by the volume of the region

$$\frac{F}{V} = \frac{ma}{V} \tag{43}$$

on the right side we get the definition of density, substituting

$$\frac{F}{V} = \rho a \tag{44}$$

where we give a as the material derivative

$$\frac{F}{V} = \rho \frac{Du}{Dt} \tag{45}$$

which expands to

$$\frac{F}{V} = \rho \left(\frac{\partial u}{\partial t} + (u \cdot \nabla)u \right). \tag{46}$$

Now we decompose the net force F into its components, pressure, diffusion and external force field

$$-\nabla p + \mu \nabla^2 u + \rho F_{ext} = \rho \left(\frac{\partial u}{\partial t} + (u \cdot \nabla)u \right), \tag{47}$$

therefore dividing by ρ we get

$$-\frac{1}{\rho}\nabla p + \frac{\mu \nabla^2 u}{\rho} + \frac{\rho F_{ext}}{\rho} = \frac{\partial u}{\partial t} + (u \cdot \nabla)u. \tag{48}$$

We are now able to replace the dynamic viscosity with kinematic viscosity

$$-\frac{1}{\rho}\nabla p + \nu \nabla^2 u + F_{ext} = \frac{\partial u}{\partial t} + (u \cdot \nabla)u, \tag{49}$$

and as a last step, reordering for the velocity field, we get the first Navier-Stokes equation for incompressible fluids (1),

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla)u - \frac{1}{\rho}\nabla p + \nu \nabla^2 u + F_{ext}. \tag{50}$$

B Statement of Helmholtz's Theorem

Let $F(r)$ be any continuous vector field with continuous first partial derivatives. Then $F(r)$ can be uniquely expressed in terms of a negative gradient of a scalar potential field $\phi(r)$ and the curl of a vector potential $a(r)$,

$$F(r) = -\nabla\phi(r) + \nabla \times a(r) \tag{51}$$

where $-\nabla\phi(r)$ is the longitudinal or irrotational ($\nabla \times \phi(r) = 0$) and $\nabla \times a(r)$ is the transverse or solenoidal part of the vector field ($\nabla \cdot a(r) = 0$):

$$\begin{aligned} -\nabla\phi(r) &= -\frac{1}{4\pi}\nabla \int_V \frac{\nabla' \cdot F(r')}{|r-r'|} d^3r' + \frac{1}{4\pi}\nabla \oint_S \frac{F(r')}{|r-r'|} \cdot n \; d^2r' \\ \nabla \times a(r) &= \frac{1}{4\pi}\nabla \times \int_V \frac{\nabla' \times F(r')}{|r-r'|} d^3r' + \frac{1}{4\pi}\nabla \times \oint_S \frac{F(r')}{|r-r'|} \times n \; d^2r' \end{aligned} \tag{52}$$

where ∇' denotes differentiation on the primed coordinates:

$$\nabla' = \hat{1}_x \frac{\partial}{\partial x'} + \hat{1}_y \frac{\partial}{\partial y'} + \hat{1}_z \frac{\partial}{\partial z'}. \tag{53}$$

If the surface S recedes to infinity and $F(r)$ is regular at infinity,

$$\begin{aligned} -\nabla\phi(r) &= -\frac{1}{4\pi}\nabla \int_V \frac{\nabla' \cdot F(r')}{|r-r'|} d^3r' \\ \nabla \times a(r) &= \frac{1}{4\pi}\nabla \times \int_V \frac{\nabla' \times F(r')}{|r-r'|} d^3r' \end{aligned} \tag{54}$$

will hold, giving Helmholtz's Theorem, named after German mathematician Hermann von Helmholtz.