

Grid-based Realtime Fluid Dynamics

Stefan Marek 0026455 *

Institute of Computer Graphics and Algorithms, TU Vienna, Austria

Abstract

Fluid dynamics is a well discussed topic in nature science and has many applications in the fields of computer graphics. A detailed overview of the grid based solutions for physically based fluid dynamics used by computational simulations is given in the following report. One of the most important way to describe fluid like dynamics in a natural way is with Navier and Stokes equations. Joe Stam gives a very good overview in the paper "Stable Fluids" [Stam 1999]. He describes a numerical solution of the Navier-Stokes equations which is used in many approaches based on computational fluid dynamics. The reader will get an understanding about the physics, the mathematics and algorithms which are used in this field of research. The physical descriptions will be used as supporting medium as one can find in the referenced papers and will not give a sufficient explanation to the used formulas and terms. The referenced authors also advice readers of their works to use books and papers written by physicians for a deeper understanding.

Keywords: computational fluid dynamics, partial differential equation, Navier-Stokes equations.

1 Introduction

Fluid dynamics is a physical term used to describe the motion and interaction of gas or liquid substances with natural environments especially forces. There are many physically based descriptions in terms of partial differential equations for this phenomenons but not all of them are usefull for computational simulations. One of the biggest problems is to find numerical stable solutions which are based on this physical descriptions even if they are simplified by making idealised presumptions. Whereas numerical stability is one of the most important part for any computational simulation especially if accuracy counts.

The classical solution for such a solution is to use numerical integration like Euler integration or better versions based on Euler for the numerical solution of the analytic formula. Using Euler integration will give you a great overview of how a solution could look like but with the cost of numerical stability and accuracy. It is often used in grid based solution which uses nice looking approximations of fluid dynamics and is prefered in approaches which uses key frame and rigid body animation. The online Siggraph course [Witkin and Baraff 1997] shows how to use numerical integration in combination with key frame and rigid body animation.

*e-mail: stefan.marek@gmx.net

1.1 Related Work

This technical review is mostly influenced by the work of [Stam 1999] because his "Stable Fluids" method shows a technique to solve the Navier-Stokes equations in a numerical stable way. It is the most important technique used in computational fluid dynamics (CFD). The basics to this stable solution is given by [Foster and Metaxas 1996] and [Foster and Metaxas 1997b]. Their work is also often referenced and gives a more fundamental solution where numerical stability is not the priority but a physically based solution with the Navier-Stokes equations. There are also several works about CFD which are not based on this kind of physical description. Because of the importance of this technique for grid based solutions this review concentrates at solutions which are based on the Navier-Stokes equations.

1.2 Navier-Stokes Equations

Claude-Louis Navier and George Gabriel Stokes invented in the 19th century a physically based mathematical discription for simulating fluids in motion. Especially their definitions and formulas for incompressible fluid gives the direction for the field or research which is called computational fluid dynamics. The following equation shows the Navier-Stokes equation for incompressible fluids. It allows to simulate the flow of fluid over a time step and is very usefull for any numerical solution.

$$\frac{\partial \mathbf{u}}{\partial t} = - \underbrace{(\mathbf{u} \cdot \nabla) \mathbf{u}}_{\text{advection}} - \underbrace{\frac{1}{\rho} \nabla p}_{\text{pressure}} + \underbrace{\nu \nabla^2 \mathbf{u}}_{\text{diffusion}} + \underbrace{\mathbf{f}}_{\text{external force}} \quad (1)$$

This equation shows a partial differential equation in its vector form which allows to calculate the changing of the velocity field over a time step $\frac{\partial \mathbf{u}}{\partial t}$. Whereas \mathbf{u} stands for the velocity field $\mathbf{u}(\mathbf{x}, t)$ with the spatial coordinates $\mathbf{x} = (x, y)$. The term p describes the pressure field $p(\mathbf{x}, t)$ and t is the time term. The constant value ρ is the density of the fluid. The fluid density is a constant because we want to have a homogeneous fluid, which assures together with the compressibility assumption, that the density is constant in time and space. The constant value ν stands for the kinematic viscosity. The term $\mathbf{f} = (f_x, f_y)$ in case of a two dimensions or $\mathbf{f} = (f_x, f_y, f_z)$ in case of a three dimensional flow represents external forces which are interacting with the flow of the fluid. The only assumption made for this formula is that it has to be incompressible [Harris 2004]. Which actually means that masses should always be conserved. There are other restrictions like boundary conditions which will be discussed in section 2. The terms in equation (1) have an important physical meaning therefore they need to be explained in detail which is done in section 1.3 "Physical Description".

The following equation is a condition to the first equation (1) and proves that the divergence of the vector field (which is actually a scalar field) has to be zero which implies divergence free fields. This condition ensures that the fluid is incompressible and mass conserving. It is also called continuity equation.

$$\nabla \cdot \mathbf{u} = 0 \quad (2)$$

The vector \mathbf{u} stands for the velocity field and nabla (∇) for the gradient operator. According to this condition any mass which enters

a fluid will leave the fluid and doesn't get lost or even more. In section I.3.1 "Incompressible Fluid" there is a more detailed description of what this physically means.

There is also a similar description for the particle density field which could be interpreted as the masses which are moved by the velocity field over a time step ($\frac{\partial \mathbf{d}}{\partial t}$).

$$\frac{\partial \mathbf{d}}{\partial t} = - \underbrace{(\mathbf{u} \bullet \nabla) \mathbf{d}}_{\text{advection}} + \underbrace{\kappa \nabla^2 \mathbf{d}}_{\text{diffusion}} - \underbrace{\alpha \nabla d}_{\text{dissipation}} + \underbrace{\mathbf{s}}_{\text{externalsource}} \quad (3)$$

The advection and diffusion (κ is the diffusion constant) term can be solved in the same way as in equation 1. Even the external source term can be solved like the external force term. The difference is in the third term which is called dissipation. The dissipation rate α is a constant value and gives the rate at which the masses dissipate or scatter inside the flow. This term could be disregarded by setting the dissipation rate to zero ($\alpha = 0$), which leads to a simpler equation for the density field.

$$\frac{\partial \mathbf{d}}{\partial t} = - \underbrace{(\mathbf{u} \bullet \nabla) \mathbf{d}}_{\text{advection}} + \underbrace{\kappa \nabla^2 \mathbf{d}}_{\text{diffusion}} + \underbrace{\mathbf{s}}_{\text{externalsource}} \quad (4)$$

This form is now nearly equal to the velocity field description and can be solved in exactly the same way. The decision whether a solution needs a dissipation rate or not depends on what natural phenomenon will be simulated.

If we define a fluid simulation in a two dimensional environment than the equation (1) is the general vector oriented form of actually two equations (three in the case of a three dimensional environment). That means if we want a description of the Navier-Stokes equations with its scalar components in two dimensions we get the following equations:

$$\frac{\partial u}{\partial t} = -(\mathbf{u} \bullet \nabla)u - \frac{1}{\rho} \nabla p + \nu \nabla^2 u + f_x, \quad (5)$$

$$\frac{\partial v}{\partial t} = -(\mathbf{u} \bullet \nabla)v - \frac{1}{\rho} \nabla p + \nu \nabla^2 v + f_y, \quad (6)$$

As a result we get a system of $n+1$ equations with $n+1$ unknown values whereas n stands for the dimension of the vector field where the flow (velocity field) exists. In this case a system of the two equations (5) and (6) plus the conditional equation (2) with three unknown values u , v and p . The term ∇p is the gradient, $\nabla \bullet \mathbf{u}$ is the divergence and $\mathbf{u} \bullet \nabla$ is the advection operator. The term $\nabla^2 p$ stands for the laplacian operator which is the short form for $\nabla^2 p = \nabla \bullet \nabla$. The sign \bullet shows the operator for the inner product. In section (2) about the finite difference equation the used operators will be described in more detail.

1.3 Physical Description

1.3.1 Incompressible Fluid

As mentioned before the condition that the fluid has to be incompressible actually means that masses have to be conserved and is mathematically described by equation (2). What this means in nature can be explained with easy to understand words. Whenever a substance flows into a region of the fluid then this condition assures that the substance exits the region with the same mass. Furthermore this masses do not vanish or even get more. In terms of solids this means whenever particles enter a fluid they will leave it and don't

get lost or more particles. To make a long story short whatever flows into comes out of the fluid.

Restrictions This restriction also means that phenomenons where two fluid like substances (two phase flow) intermixture and chemically become one can't be simulated with this approach. It just allows to simulate how substances interacts with fluids. As a consequence surface tension, interface tension and effects which are based on it are also not in this concept. Where interface tension is the surface tension between liquids. Anyway [Foster and Metaxas 1997a] shows a method which allows to approximate the behaviour of surface tension within incompressible fluids.

1.3.2 Terms

For a deeper understanding of what the four terms on the right side of the Navier-Stoke equation are meaning there is another description which can be found in many papers especially those which are based at the work of [Stam 1999]. Especially [Harris 2004] describes the terms of the Navier-Stokes equation in a very understanding way.

Advection The first term on the right side of the equation (1) is called the advection term. Small particles, for example dust particles in air, are transported along the given velocity field of the flow. This velocity field also carries itself within every time step. This self carrying is called advection or self advection and shows the motion by the velocity of the fluid flow.

Pressure The second term on the right side of the equation (1) is called the pressure term. Pressure is force per unit area which means that each particle and the fluid itself is accelerated by it. This is easy to understand if Newton's second law $\mathbf{F} = m\mathbf{a}$ is considered. Pressure is the cause for acceleration in the fluid flow.

Diffusion The third term on the right side of the equation (1) is called the diffusion term. Moving fluid is not a uniform process which means that regions inside the flow are moving faster than others. Imagine ink that drops into a bowl filled with water. The dispersion of the ink is not an uniform process in fact the distribution of the dye will show swirling effects. This phenomenon is called diffusion and exists because of the viscosity of fluid. Viscosity is a physical description of how resistive is the fluid at a specific point or region to be moved. This resistance influences the velocity which will be seen as vortex or swirl in the fluid.

1.3.3 External Forces

The fourth term on the right side of the equation (1) is called the external forces term. The Navier-Stokes equations takes into account that there can be forces which influences the flow but exists outside of it. That means if for example wind or gravity should influence the flow in a specific simulation it can easy be handled by adding this force to the other three parts of the equation. Adding an external force means accelerating the flow. The force term is of great importance because it allows to simulate natural phenomenons which influence the fluid. The following terms are the cause for many of this phenomenons.

Newtonian Gravity Gravity is an important and easy to simulate natural effect which occurs in any physical environment where masses are taken into account. The following equation shows the vector form of Newton's law of universal gravitation [Wikipedia 2007a].

$$f = -g \frac{m_1 m_2}{r^2} r_{12} \quad (7)$$

Where f is the force applied on an object 2 due to $d1$, g is the gravitational constant, r is the euclidean distance between object 1 and 2, m_1 and m_2 are the masses of object 1 and 2 and r_{12} is the unit vector from object 1 to object2. Without taking gravity into account the particles are in free fly which actually means that they will never fall to the ground even not if there is no movement. Gravity is the cause of accelerating a mass in the direction of the mass it is connected.

Buoyancy and Convection Buoyancy is a force that causes convection in gravity fields and is one of the major modes of heat and mass transfer within fluids. It allows to simulate various effects like heat and smoke which are based on convection.

Convection currents are caused by the changes in density associated with temperature changes. [Harris 2004]

This physical transfer is an important part in any fluid because it makes the flow faster where the temperature is high and slower where the temperature is low.

Smoke and Heat effects are based on convection. To simulate this effects [Harris 2004] or more detailed [Ronald Fedkiw and Jensen 2001] introduces a simple mathematical description by using buoyancy force. For effects like fire which are based on heat the following equation leads to a solution.

$$f_{buoy} = \sigma(T - T_0) \hat{\mathbf{j}} \quad (8)$$

Where the constant value σ is a scale factor, T is a scalar field for the temperatur, T_0 the amient temperatur and $\hat{\mathbf{j}}$ is a vector which points in the upward direction. Wherever a temperatur is different from the given ambient temperatur T_0 the buoyoncy force will be added to the velocity field of the flow as an external force. This equation can be enhanced for simulating smoke effects by using the following form.

$$f_{buoy} = (-\kappa \mathbf{d} + \sigma(T - T_0)) \hat{\mathbf{j}} \quad (9)$$

The only difference to the equation above is the constant κ which is a scale factor for the smoke paricle mass and the scalar field \mathbf{d} which gives the smoke density. This solution can also be used for simulating clouds.

Hardening and Melting Melting is a physical process which transforms solids into fluids after the solid is heated over its melting point which is given by the material properties of the solid. Hardening is the inverse process of melting. [Mark Carlson and Turk 2002] show a way to simulate this effect. Therefore a reformulation of the diffusion term is necessary which additionally has to solve the melting part.

Wind is moved air which is itself a gas and can therefore be evaluated by the Navier-Stokes equation. Using the Navier-Stokes equation may be an overkill and often just a common direction and

the constant pressure of the wind is enough for a simulation. Imagine a simulation where wind is used to move clouds in a given direction. Anyway if a more complex movement of fluid by wind force is needed, an appropriate description of the force has to be used. [Treuille and Stam 2003] gives a way to simulate wind forces which can be controlled during the whole simulation process.

2 Numerical Stable Fluid

Formulating the partial differential equation explicitly and using finite difference equations is probably the easiest way to find a numerical solution which can be used for a computational simulation. This is not always the best strategy because this solutions often lack of numerical stability. Numerical instability leads to inaccuracy in the calculations and therefore the whole simulation provides wrong results. This is a tricky problem because the results look right at the beginning of a simulation and probably they are but after a while the results get wronger and wronger. Therefore numerical stability is a crucial condition for any physically based simulation. [Stam 1999] introduces in his work about "Stable Fluids" a numerical stable solution which is based on the Helmholtz-Hodge decomposition. The Helmholtz-Hodge decomposition shows that any vector field can be decomposed into the following form:

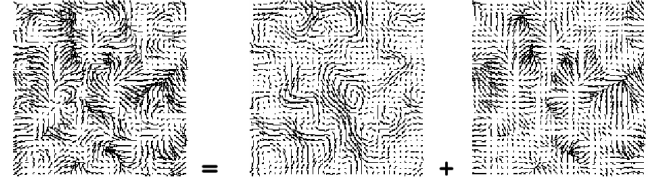


Figure 1: Any divergent vector field can be decomposed into a divergence free vector field and a gradient field. [Stam 2003]

$$\mathbf{w} = \mathbf{u} + \nabla q \quad (10)$$

Whereas \mathbf{w} is the given vector field which should be decomposed into another divergence free vector field \mathbf{u} and a scalar field q . The nabla (∇) term stands for the gradient operator and defines together with the scalar field q a gradient field. The vector field \mathbf{u} is a divergence free field which is validated by the equation $\nabla \bullet \mathbf{u} = 0$. This Helmholtz-Hodge decomposition can be interpreted physically in that form that any vector field \mathbf{w} is the sum of a mass conserving (divergence free) vector field \mathbf{u} and a gradient field ∇q . With this equation a projection operator \mathbf{P} could be defined as following:

$$\mathbf{P}(\mathbf{w}) = \mathbf{P}(\mathbf{u}) + \mathbf{P}(\nabla q) \quad (11)$$

with the characteristics $\mathbf{P}(\mathbf{w}) = \mathbf{P}(\mathbf{u}) = \mathbf{u}$ and $\mathbf{P}(\nabla q) = 0$. This projection is an operator which projects any vector field \mathbf{w} into the divergence free vector field \mathbf{u} . What is required is the divergence free vector field part therefore equation (10) has to be transformed:

$$\mathbf{u} = \mathbf{P}(\mathbf{w}) = \mathbf{w} - \nabla q \quad (12)$$

This projection operator can now be used to find one single equation for a numerical solution by applying it to the Navier-Stokes equation (2). Which leads us to the following equation:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{P}(-(\mathbf{u} \bullet \nabla) \mathbf{u} - \frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}) \quad (13)$$

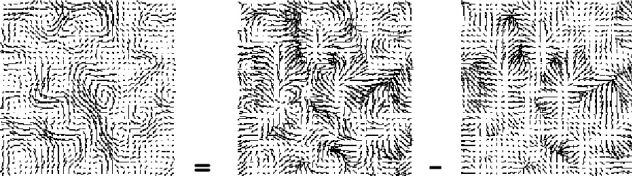


Figure 2: Any divergence free vector field is given by the difference between a divergent vector field and a gradient field. [Stam 2003]

This equation is the basis [Stam 1999] used for his numerical solution. An advantage of this transformation is that we can skip the pressure term because of the Helmholtz-Hodge decomposition which requires a divergence free velocity field ($\nabla \cdot \mathbf{u} = 0$) and so the whole pressure term becomes zero because of the feature $P(\nabla q) = 0$. Thus equation (7) can be written in the form:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{P}(-(\mathbf{u} \cdot \nabla) \mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{f}) \quad (14)$$

and together with equation (3) we have all formulas we need for a solution. Now the equation can be solved within four steps. Where the starting point is the evaluation of $\mathbf{w}_0 = \mathbf{u}(\mathbf{x}, t)$ which means that we need the result of the previous time step to calculate the actual time step (point in time). According to the comments from [Stam 1999] and [Andersson 2004] there is a symbolic description of this solution which can be seen as concatenation of the following four steps.

$$\mathbf{w}_0(\mathbf{x}) \xrightarrow{\text{addforce}} \mathbf{w}_1(\mathbf{x}) \xrightarrow{\text{advect}} \mathbf{w}_2(\mathbf{x}) \xrightarrow{\text{diffuse}} \mathbf{w}_3(\mathbf{x}) \xrightarrow{\text{project}} \mathbf{w}_4(\mathbf{x}) \quad (15)$$

Each step needs a velocity field as input and delivers a divergent velocity field as output. This means that the velocity field is not divergence free before the last step, which is why a projection step has to be executed at the end.

$$\text{Add Force } \mathbf{w}_0(\mathbf{x}) \longrightarrow \mathbf{w}_1(\mathbf{x}) \iff \mathbf{w}_1(\mathbf{x}) = \mathbf{w}_0(\mathbf{x}) + \Delta t \mathbf{f}(\mathbf{x}, t)$$

The first step is to add the external force to our solution which means that we have to evaluate the following equation with the given time step Δt :

$$\mathbf{w}_1(\mathbf{x}) = \mathbf{w}_0(\mathbf{x}) + \Delta t \mathbf{f}(\mathbf{x}, t) \quad (16)$$

Which can be realized with an explicit numerical integration technique like Euler integration. If the simulation depends on more than one force than this can be done by summing this forces and that add it to the initial velocity field.

$$\text{Advection } \mathbf{w}_1(\mathbf{x}) \longrightarrow \mathbf{w}_2(\mathbf{x}) \iff \mathbf{w}_2(\mathbf{x}) = \mathbf{w}_1(\mathbf{p}(\mathbf{x}, -\Delta t))$$

The second step encapsulates the advection term and has to solve the expression $-(\mathbf{u} \cdot \nabla) \mathbf{u}$. This step is one of the hardest problem in this kind of solution because of its non linear nature and the lack of numerical stable methods. The varying solutions for this problem differ especially in this step. [Stam 1999] used a numerical stable technique called method of characteristics which can be used to solve partial differential equations. The following equation evaluates a fluid particle in the velocity field at a spatial point to a given time step by following the streamline from the previous time step to the actual time step.

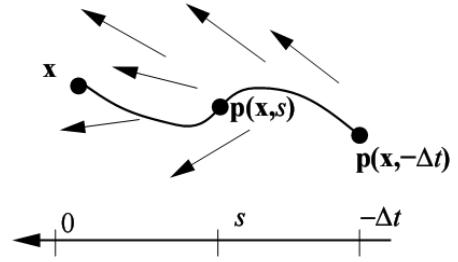


Figure 3: shows the evaluation of a fluid particle $p(\mathbf{x}, s)$ along a stream line of the velocity field from the previous time step Δt at the point $p(\mathbf{x}, -\Delta t)$ to the actual point \mathbf{x} . [Stam 1999]

$$\mathbf{w}_2(\mathbf{x}) = \mathbf{w}_1(\mathbf{p}(\mathbf{x}, -\Delta t)) \quad (17)$$

A numerical realisation of this method can be done by using linear interpolation and a computational system for particle tracing. There is also a lot more to say about the used mathematics and why this is a valid solution which can be found in the work of [Stam 1999].

$$\text{Diffusion } \mathbf{w}_2(\mathbf{x}) \longrightarrow \mathbf{w}_3(\mathbf{x}) \iff (\mathbf{I} - \nu \Delta t \nabla^2) \mathbf{w}_3(\mathbf{x}) = \mathbf{w}_2(\mathbf{x})$$

The third step solves the diffuse term $\nu \nabla^2 \mathbf{u}$. [Stam 1999] gives the advice to use a numerical stable method as described in the following equation:

$$(\mathbf{I} - \nu \Delta t \nabla^2) \mathbf{w}_3(\mathbf{x}) = \mathbf{w}_2(\mathbf{x}) \quad (18)$$

Where \mathbf{I} is the identity operator. This leads to a sparse linear system $Ax = b$ for the unknown velocity field \mathbf{w}_3 . Which can be solved in a numerical stable way by using an iterative implicit scheme called Gauss-Seidel relaxation [Black and Moore 2003].

$$\text{Projection } \mathbf{w}_3(\mathbf{x}) \longrightarrow \mathbf{w}_4(\mathbf{x}) \iff \mathbf{w}_4(\mathbf{x}) = \mathbf{w}_3(\mathbf{x}) - \nabla q$$

The last step projects the velocity field to make it divergence free. Therefore, a poisson equation ($\nabla^2 q = b$) as a result of transformation of the equation $\mathbf{w} = \mathbf{u} + \nabla q$ into the form

$$\mathbf{w}_4(\mathbf{x}) = \mathbf{w}_3(\mathbf{x}) - \nabla q \quad (19)$$

has to be solved. This poisson equation can be achieved by multiplying both sides of the equation (5) with the nabla (∇) operator ($\nabla^2 q = \nabla \cdot \mathbf{w}_3$) under the condition $P(\nabla q) = 0$ as it is given by equation (7). A numerical solution of this equations can be realised with the method of characteristics which leads to a linear system of the form $Ax = b$ as it is used in step two. Where A is a Matrix which is given by the laplacian operator ∇^2 and b is a vector of constant values. The vector x is used to solve the velocity field \mathbf{u} and the pressure field q . [Harris 2004] uses Jacobi iteration to solve this Poisson equation which is stable and easy to implement but methods like conjugate gradient or multigrid methods achieve the same goal with linear complexity.

As a result we get a divergence free velocity field as it is postulated by Navier-Stokes equation (1). For a proof and more detailed descriptions take a look at the work of [Stam 1999], [Foster and Metaxas 1996] and [Harris 2004].

Dissipation As mentioned before equation (4) could be solved like the numerical solution for the advection, diffusion and force above. If the application needs the dissipation of the masses then there is also a numerical solution for that problem:

$$(1 + \Delta t \alpha) \mathbf{d}(\mathbf{x}, t + \Delta t) = \mathbf{d}(\mathbf{x}, t) \quad (20)$$

2.1 Comparison

[Foster and Metaxas 1996] use an explicitly formulated solution of the Navier-Stokes equations which is discretised by using finite difference equations and numerical integration. Their approach is not numerical stable but can be implemented straightforward.

In choosing an explicit formulation we have made a tradeoff between ease of implementation and the maximum stable time step for numerical integration. [Foster and Metaxas 1996]

[Stam 1999] describes that his numerically stable solver has the problem that the rate at which the fluid will be dampened is too high. A solution of the problem is a constant update of the velocity field by an external force which forces vibrations.

2.2 General Algorithm

This numerical solution of the Navier-Stokes equations leads to an algorithmic description as it can be found in [Stam 1999].

Algorithm 1 Fluid Velocity Field

Require: velocity field \mathbf{w}_0

Ensure: divergence free velocity field $\nabla \bullet \mathbf{w}_4 = 0$

- 1: $\mathbf{w}_1 = \text{force}(\mathbf{w}_0) \{ \mathbf{w}_1(\mathbf{x}) = \mathbf{w}_0(\mathbf{x}) + \Delta t \mathbf{f}(\mathbf{x}, t) \}$
 - 2: $\mathbf{w}_2 = \text{advect}(\mathbf{w}_1) \{ \mathbf{w}_2(\mathbf{x}) = \mathbf{w}_1(\mathbf{p}(\mathbf{x}, -\Delta t)) \}$
 - 3: $\mathbf{w}_3 = \text{diffuse}(\mathbf{w}_2) \{ (\mathbf{I} - \nu \Delta t \nabla^2) \mathbf{w}_3(\mathbf{x}) = \mathbf{w}_2(\mathbf{x}) \}$
 - 4: $\mathbf{w}_4 = \text{project}(\mathbf{w}_3) \{ \mathbf{w}_4(\mathbf{x}) = \mathbf{w}_3(\mathbf{x}) - \nabla q \}$
-

This can easily be simplified in a less memory consuming version by using just two velocity field variables \mathbf{w} and \mathbf{w}_0 . Therefore before each step the variables have to be swapped. Algorithmic descriptions mostly differ in the execution order of the force step, the advect step and the diffuse step concerning which numerical solution will be used for an implementation. This algorithm shows how to calculate the velocity field computational but not particles which are carried by this fluid flow. For a visualisation of particles like dye smoke or other phenomena an enhanced version of this algorithm is needed. This enhanced algorithm needs to define and calculate a density field. The density values represent the particles which are moved by the velocity field. How this density field is calculated depends on what will be shown and furthermore on the implementation. Realisation of those density fields are described in the implementation section. [Stam 2003] shows a general version for realtime purposes of such an algorithm where densities are moved within a velocity field.

Algorithm 2 Fluid Density Field

Require: density field \mathbf{d}_0 , velocity field \mathbf{w}

Ensure: moved density field \mathbf{d}_3

- 1: $\mathbf{d}_1 = \text{force}(\mathbf{d}_0) \{ \mathbf{d}_1(\mathbf{x}) = \mathbf{d}_0(\mathbf{x}) + \Delta t \mathbf{s}(\mathbf{x}, t) \}$
 - 2: $\mathbf{d}_2 = \text{advect}(\mathbf{d}_1, \mathbf{w}) \{ \mathbf{d}_2(\mathbf{x}) = \mathbf{d}_1(\mathbf{p}(\mathbf{x}, -\Delta t)) \}$
 - 3: $\mathbf{d}_3 = \text{diffuse}(\mathbf{d}_2, \mathbf{w}) \{ (\mathbf{I} - \kappa \Delta t \nabla^2) \mathbf{d}_3(\mathbf{x}) = \mathbf{d}_2(\mathbf{x}) \}$
 - 4: $\mathbf{d}_4 = \text{dissipate}(\mathbf{d}_3) \{ (1 + \Delta t \alpha) \mathbf{d}_4(\mathbf{x}, t + \Delta t) = \mathbf{d}_3(\mathbf{x}, t) \}$
-

As mentioned before algorithm 1 and algorithm 2 are nearly the same which automatically leads to a simplified implementation. Techniques for an evaluation of the force, advect and diffuse steps are explained in the section 3 "Grid Based Solutions".

Algorithm 3 Fluid Simulator

Require: velocity field \mathbf{w} , density field \mathbf{d}

Ensure: divergence free velocity field $\nabla \bullet \mathbf{w} = 0$, moved density field \mathbf{d}

- 1: **while** simulating **do**
 - 2: Get force and source emitter from user interaction
 - 3: Swap(\mathbf{w}_0, \mathbf{w}) {swapping velocity \mathbf{w}_0 with velocity \mathbf{w} from previous time step}
 - 4: Swap(\mathbf{d}_0, \mathbf{d}), {swapping velocity \mathbf{d}_0 with velocity \mathbf{d} from previous time step}
 - 5: $\mathbf{w} = \text{Fluid Velocity Field}(\mathbf{w}_0)$ {evaluate actual velocity field \mathbf{w} using Algorithm 1}
 - 6: $\mathbf{d} = \text{Fluid Density Field}(\mathbf{w}_0, \mathbf{d}_0)$ {evaluate actual density field \mathbf{d} using Algorithm 2}
 - 7: Draw Density Field (\mathbf{d}) \mathbf{d} {visualise the actual density field \mathbf{d} using a rendering routine}
 - 8: **end while**
-

Code line 2 means that the user can interact with the fluid by selecting a spatial location of the grid. This input will be used as particle emitter given by the source term in equation 3 which allows a visualisation of the fluid by moving densities with the flow. It can also be used to change the flow itself by applying a force (force term equation 1).

2.3 Boundary Condition

The decision which boundary condition one can use for your application are crucial for the accuracy. There are two possible types of conditions

1. periodic boundary condition and
2. fixed boundary condition.

The usage of this types of conditions for grid based solutions (section 3) will be explained here.

2.3.1 Periodic

If a specific simulation can be formulated as periodic function then this problem can be treated as a periodic boundary condition problem. A realisation can be to define a box where the velocity field exists and then repeat this box within its periodicity.

One possible way is to use a Fourier domain and solve the problem with the fast Fourier transformation (FFT) [Stam 1999]. This allows a simplification of the used mathematics in the following way. The gradient operator in the Fourier domain is equivalent to the multiplication by the imaginary part ik ($i = \sqrt{-1}$). When this gradient operator displacement is substituted by the gradient operator ∇ in the previously used formulas the diffusion term simplifies to

$$I - \nu \Delta t \nabla^2 \iff 1 + \nu \Delta t k^2 \quad (21)$$

and the projection term to

$$P(w) \iff \hat{P}(\hat{w}(k)) = \hat{w}(k) - \frac{1}{k^2} (k \bullet \hat{w}(k)) k \quad (22)$$

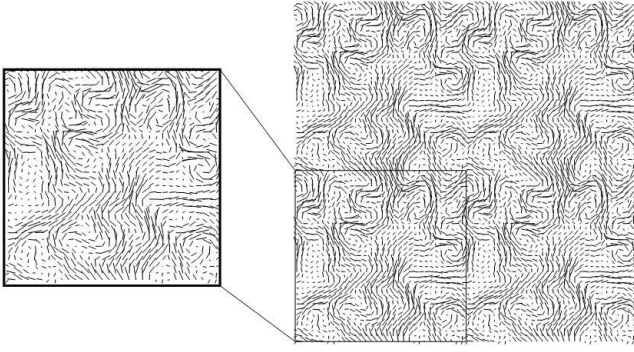


Figure 4: shows periodic tiling of a generated box (left) on a plane (right). [Stam 2001]

where $k = |k|$. Following the declarations of the Fourier domain the diffusion term can be interpreted as a low pass filter. The projection operator \hat{P} maps the vector field $\hat{w}(k)$ to that plane which is normal to the wave number k [Stam 1999]. The following algorithmic description shows such a fluid field simulation with periodic boundary conditions.

Algorithm 4 FFT Fluid Velocity Field Simulator

Require: velocity field w_0

Ensure: divergence free velocity field $\nabla \bullet w_5 = 0$

- 1: $w_1 = \text{force}(w_0) \{ w_1 = w_0 + \Delta t f \}$
 - 2: $w_2 = \text{advect}(w_1) \{ w_2 = w_1(p(x), -\Delta t) \}$
 - 3: $\hat{w}_2 = \text{transform}(w_2) \{ \hat{w}_2 = FFT(w_2) \}$
 - 4: $\hat{w}_3 = \text{diffuse}(\hat{w}_2) \{ \hat{w}_3 = \hat{w}_2(k) / (1 + \nu \Delta t k^2) \}$
 - 5: $w_4 = \text{project}(\hat{w}_3) \{ w_4 = \hat{P}(\hat{w}_3) \}$
 - 6: $w_5 = \text{transform}(w_4) \{ w_5 = FFT^{-1}(w_4) \}$
-

Given that the Fourier transformation is of complexity $N \log N$ the whole solution is of that complexity. An implementation of this technique can be used for texture generations.

2.3.2 Fixed

If a simulation of a fluid field bounded by a box is needed then there have to be found fixed boundary conditions for the borders of the box.

- **Clamping:** The simplest sort of boundary condition is to clamp the fluid at its borders which means that no fluid exits the box at its borders.
- **Continuing:** An enhanced version uses bounding conditions that let the fluid flow along the borders instead of just clamping it.
- **Repeating:** Another version could be that the fluid which leaves the border at one side will exit at the other side. This simply means that the fluid is repeated inside the box.
- **Wind Tunnel:** If we want to define our flow inside a wind tunnel then there could be a fluid emitter on one border side and a clamping strategy on all other borders.

This list could be repeated in many ways because boundary conditions are very application dependent as the example of the wind tunnel shows. Therefore a more mathematically based differentiation of the boundary condition problem is useful. The following

two types of boundary condition are defined for special types of partial differential equations like the Navier-Stokes equations.

- **Dirichlet Boundary Condition:** Clamping actually means that the implementation follows a Dirichlet boundary condition. This condition says that the flow only exists within a given region and not outside of it.
- **Neumann Boundary Condition:** Continuing follows the Neumann boundary condition which says that there has to be a predefined value of the first derivation for any border cell. For example the velocity becomes zero at the border which means that the flow disappears when it reaches the grid borders. Another continuing strategy is to set the horizontal components of the velocity field zero where the flow meets the vertical borders. At the horizontal borders the vertical velocity component values are also set to zero. An implementation of this bounding condition shows the fluid flowing along the borders as it is used by [Stam 2003].

For a realisation of the fixed boundary condition in a simulation, all steps (advection, diffusion, force and projection) has to be extended with functionality that implements the bounding of the fluid flow within the given borders after the evaluation. Border cells do not only appear on the hull of the grid. They can also appear inside the grid for example as a wall or any other solid. Whenever the fluid interacts with this solid the given border condition is taken into account.

The following algorithmic [Harris 2004] description shows a way to implement a fluid field simulator in realtime with fixed boundary conditions using a texture rendering technique. [Harris 2004] implementation of the fixed boundary condition is to left borders of the grid free and fill this borders with the values of the neighbouring cells.

Algorithm 5 Box Fluid Velocity Field Simulator

Require: velocity field w_0

Ensure: divergence free velocity field $\nabla \bullet w_4 = 0$

- 1: $w_1 = \text{advect}(w_0) \{ w_1(x) = w_0(p(x), -\Delta t) \}$
 - 2: $w_2 = \text{diffuse}(w_1) \{ (I - \nu \Delta t \nabla^2) w_2(x) = w_1(x) \}$
 - 3: $w_3 = \text{force}(w_2) \{ w_3(x) = w_2(x) + \Delta t f(x, t) \}$
 - 4: $\text{project}(w_3) \{ \text{The projection step is divided into the two following steps} \}$
 - 5: $\text{project} \rightarrow p_0 = \text{compute pressure}(w_3) \{ p_0 = \nabla q \}$
 - 6: $\text{project} \rightarrow w_4 = \text{subtract pressure gradient}(w_3, p_0) \{ w_4(x) = w_3(x) - p_0 \}$
-

Another algorithmic description is given by [Stam 2003] where he uses his idea [Stam 1999] for interactive and realtime purposes. Algorithm 3 stays the same but there is a slight difference in algorithm 1 and 2. The additional projection step after the advection step leads to a more accurate evaluation because the diffusion step gets a velocity field with conserved masses (divergence free). There is also a difference in the order of execution of the diffusion and advection step. [Stam 2003] describes that a fixed continuing boundary condition strategy is used in his implementation but any other could be used.

Algorithm 6 Fluid Velocity Field 2

Require: velocity field \mathbf{w}_0 **Ensure:** divergence free velocity field $\nabla \bullet \mathbf{w}_4 = 0$

- 1: $\mathbf{w}_1 = \text{force}(\mathbf{w}_0) \{ \mathbf{w}_1(\mathbf{x}) = \mathbf{w}_0(\mathbf{x}) + \Delta t \mathbf{f}(\mathbf{x}, t) \}$
 - 2: $\mathbf{w}_2 = \text{diffuse}(\mathbf{w}_1) \{ (\mathbf{I} - \nu \Delta t \nabla^2) \mathbf{w}_2(\mathbf{x}) = \mathbf{w}_1(\mathbf{x}) \}$
 - 3: $\mathbf{w}_3 = \text{project}(\mathbf{w}_2) \{ \mathbf{w}_3(\mathbf{x}) = \mathbf{w}_2(\mathbf{x}) - \nabla q \}$
 - 4: $\mathbf{w}_4 = \text{advect}(\mathbf{w}_3) \{ \mathbf{w}_4(\mathbf{x}) = \mathbf{w}_3(\mathbf{p}(\mathbf{x}, -\Delta t)) \}$
 - 5: $\mathbf{w}_5 = \text{project}(\mathbf{w}_4) \{ \mathbf{w}_5(\mathbf{x}) = \mathbf{w}_4(\mathbf{x}) - \nabla q \}$
-

Algorithm has to be changed by exchanging the advection and diffusion steps.

Algorithm 7 Fluid Density Field 2

Require: density field \mathbf{d}_0 , velocity field \mathbf{w} **Ensure:** moved density field \mathbf{d}_3

- 1: $\mathbf{d}_1 = \text{force}(\mathbf{d}_0) \{ \mathbf{d}_1(\mathbf{x}) = \mathbf{d}_0(\mathbf{x}) + \Delta t \mathbf{s}(\mathbf{x}, t) \}$
 - 2: $\mathbf{d}_2 = \text{diffuse}(\mathbf{d}_1, \mathbf{w}) \{ (\mathbf{I} - \kappa \Delta t \nabla^2) \mathbf{d}_3(\mathbf{x}) = \mathbf{d}_2(\mathbf{x}) \}$
 - 3: $\mathbf{d}_3 = \text{advect}(\mathbf{d}_2, \mathbf{w}) \{ \mathbf{d}_2(\mathbf{x}) = \mathbf{d}_1(\mathbf{p}(\mathbf{x}, -\Delta t)) \}$
-

3 Grid Based Solution

The following section shows how to use the Navier Stokes equations (1) and (2) for grid based solutions. Grids are widely used and well understood in computer graphics. Therefore this review gives just a short introduction with a more detailed explanation of how to use the stable fluid technique in conjunction with grids.

3.1 Grid Spacing

Grids are spatial data structures which divides a given space into cells. The cell spacing has to be predefined for any given direction. So that it is clear which length, width and depth each cell has. This leads to a spatial partitioning which can be used for discretised numerical solutions. Mostly this grid cells possess information like density values at the cross over points. Density values inside the cell can be calculated by interpolation techniques. Based on this spatial partitioning data structure there are many algorithms which simulate behaviour and visual appearance of the density value field. Another method is to store the values in the cell centers as it is given in textures. A combination of this two methods can be used for three dimensional grids. Where each of the six cell faces can be used to store the velocity in the center and the cross over points can be used to store density values (Figure 5).

Two dimensional grid based solutions are often used for texture rendering techniques and three dimensional solutions are preferred in volume rendering techniques.

3.2 Finite Difference Equation

For implementations, any numerical solution which is based on partial differential equations has to be discretised which can be achieved by translating it into a finite difference equation. For the gradient, divergence and laplacian operators there exists finite difference forms which can be found in [Harris 2004] and [Carlson 2004] or any book about numerical mathematics. The following terms δx and δy will be used as grid spacing and the indices i and j as grid positions.

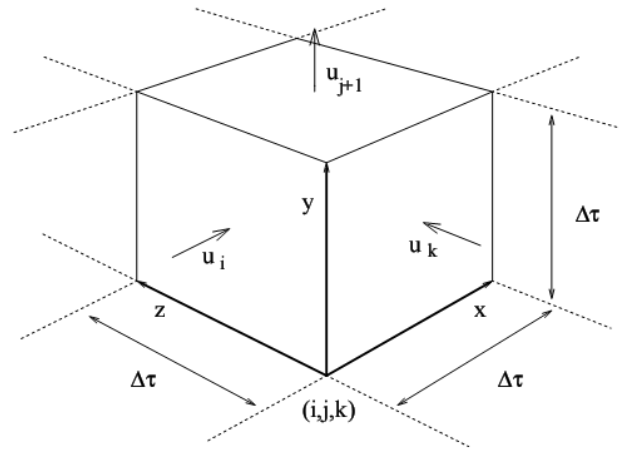


Figure 5: shows one grid cell where velocity values are stored at the cell surfaces and densities at the cross over points. [Foster and Fedkiw 2001]

Gradient A two dimensional gradient operator is mathematically defined as $\nabla p = (\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y})$. The gradient of a scalar field p is a vector field and the component values of the vector is given by the partial derivation of the scalar field p . The magnitude which means the length of the vector can physically be seen as the velocity which points into the direction of the gradient vector. The finite difference form of the gradient for a two dimensional cartesian grid can be written as:

$$\nabla p = (\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}) = (\frac{p_{i+1,j} - p_{i-1,j}}{2\delta x}, \frac{p_{i,j+1} - p_{i,j-1}}{2\delta y}) \quad (23)$$

Divergence A two dimensional divergence operator is mathematically defined as $\nabla \bullet \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}$. In physics the divergence gives the rate at which a given density exits a given region of space [Harris 2004]. The finite difference form of the divergence for a two dimensional cartesian grid could be written as:

$$\nabla \bullet \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = \frac{u_{i+1,j} - u_{i-1,j}}{2\delta x} + \frac{u_{i,j+1} - u_{i,j-1}}{2\delta y} \quad (24)$$

Laplacian operator A two dimensional laplacian operator is mathematically defined as $\nabla^2 p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}$. The finite difference form of the laplacian for a two dimensional cartesian grid could be written as:

$$\nabla^2 p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = \frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{(\delta x)^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{(\delta y)^2} \quad (25)$$

This expression can be shortened to

$$\nabla^2 p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = \frac{p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} + 4p_{i,j}}{(\delta x)^2} \quad (26)$$

, if the grid cells are square, which means that the step size in x and y direction is equal $\delta x = \delta y$.

Poisson equation are of the form $\nabla^2 x = b$. If b in the Poisson equation is zero then this equation is called Laplace's equation

$\nabla^2 x = 0$. The Laplace equation is the origin of the Laplacian operator. In the case of a squared two dimensional grid [Harris 2004] shows a solution of the Poisson equation $\nabla^2 x = b$ with an iterative method called Jacobi iteration. Therefore the following finite difference equation will be used:

$$x_{i,j}^{(k+1)} = \frac{x_{i-1,j}^{(k)} + x_{i+1,j}^{(k)} + x_{i,j-1}^{(k)} + x_{i,j+1}^{(k)} + \alpha b_{i,j}}{\beta} \quad (27)$$

where α and β are constant values. For the calculation of the pressure field the constant values are $\alpha = -(\delta x)^2$ and $\beta = 4$ where the value x stands for the pressure p and $b = \nabla \bullet \mathbf{w}$. To solve the diffusion equation the values $\alpha = \frac{(\delta x)^2}{\nu \delta t}$ and $\beta = 4 + \alpha$ can be used where x and b stands for the velocity \mathbf{u} .

There are also three dimensional finite difference forms for this operators which can be found in [Carlson 2004] and [Ronald Fedkiw and Jensen 2001]. In addition there exists better approximation strategies for the numerical solutions which can cost more execution time but are smoother or more accurate.

3.3 Common Grids

Common grids are defined as arrays which accomodates any finite difference equation because of their similar indexing. This leads to a implementation which is straightforward. In the following paragraphs the used Navier-Stokes terms are discussed on such grids.

Fluid Simulation [Stam 2003] gives a self-explanatory description of a grid based implementation which will be used in this and the following three paragraphs. Each of the steps given by algorithm 6 have to be evaluated and the resulting velocity field is stored into an array which is defined as grid. After the calculation of the fluid all substances which are moved within the flow have to be simulated.

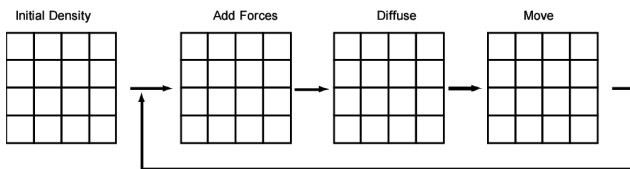


Figure 6: shows the three update stages of the density field solver. [Stam 2003]

3.3.1 Add Forces

First the forces have to be added to the initial grid of the density field. Sources are given by predefined emitters or by user interaction which fills a source array with the same dimensions as the density grid. The implementation is simple because the values just have to be added to the density grid by evaluating the actual time step multiplied by the source array ($x[i] = \Delta t * s[i]$).

3.3.2 Diffusion

At the diffusion stage velocity and density values are moved by the given turbulence of the velocity field. Thus the density values have to be dispersed (spreaded) within the given rate of diffusion. This means that the density values of the grid cells have to be moved to

the neighbouring grid cells by the given diffusion of the velocity field.

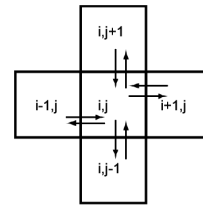


Figure 7: shows the exchange of velocity and density values with the neighbouring cells. [Stam 2001]

3.3.3 Advection

At the advection stage substances which are placed into the fluid where transported along the velocity field. This means that the density values where moved along the given streamlines of the actual time step. For a realisation the density value of a grid position is copied to the grid position which is evaluated by the advection term.

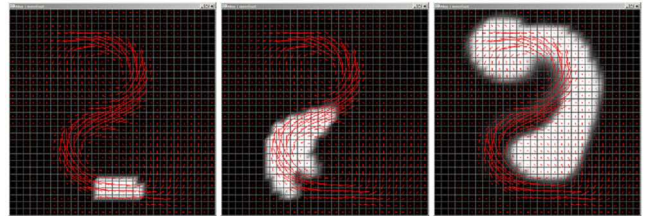


Figure 8: shows particles moved along the velocity field. [Stam 2001]

3.4 Hierarchical Grids

Grids can also be seen as hierarchical datastructure as it is given by trees. Hierarchical datastructures have the advantage that they subdivide the given space in form of a tree. Which leads to faster searching and manipulation strategies and furthermore to an acceleration of the whole solution.

Octree and Quadtree One of those hierarchical datastructure is called Octree for three dimensional and Quadtree for two dimensional grids. The octree is a data structure called tree with which the given space is subdivided by bounding boxes. At any tree level a bounding box carries itself bounding boxes or leave nodes which are cells of the grid. Especially searching of a grid cell is speeded up because traversal of the tree nodes has a run time of $n \log n$. This technique is therefore a very usefull strategy to accelerate any grid based solution. Especially those using collisions of solids with the fluid as the following picture shows [Frank Lossaso and Fedkiw 2004].

[Frank Lossaso and Fedkiw 2004] use a special mathematical description of the Navier-Stokes equation for the octree data structure which is slightly different to the mentioned solution.

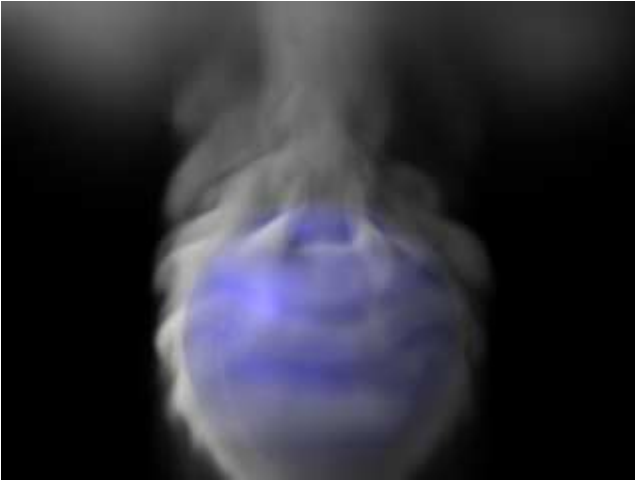


Figure 9: shows smoke colliding with a sphere. [Frank Lossaso and Fedkiw 2004]

3.5 Grid Sampling

One of the main problems when working with grids is the used resolution. The resolution of the grid is given by the number of cells which are used for the partitioning of a domain. Subdividing a space into pieces is a discretisation process which leads to discontinuity. Whenever grids are used for simulations the sampling rate at which the used discretised formulas are evaluated is important for the accuracy of the calculations. The evaluated values are stored within cells and the reconstruction of the values depends on the grid spacing because it defines how many values can be stored. If the spacing is too large the simulation suffer from numerical inaccuracy and if it is too small the whole simulation takes too long for a result.

Vorticity Confinement A specific problem when using coarse grids is that rotational parts of the flow will be dampened out. This rotational parts causes vortex and swirling effects in the simulation of liquids. Therefore the vorticity has to be estimated by solving the equation:

$$\omega = \nabla \times \mathbf{u} \quad (28)$$

where \mathbf{u} is the velocity field and ω is the resulting vorticity vector. The sign \times stands for the cross product operator. Then the vorticity vector field is given by the following normalisation process:

$$\psi = \frac{\eta}{|\eta|} \quad (29)$$

where $\eta = \nabla|\omega|$ and ψ is the resulting vector field. The force can now be calculated by:

$$\mathbf{f} = \varepsilon(\psi \times \omega)\delta x. \quad (30)$$

, where the ε is a constant which allows scaling of the force and δx gives the grid spacing. This approach can now be used by evaluating the vorticity confinement computationally and add it to the current velocity during the force step. Further descriptions of this technique can be found in [Harris 2004] and [Ronald Fedkiw and Jensen 2001].

Adaptive Time Step One crucial point in using grids with discrete approximations of fluid simulations is to find an appropriate time step Δt for the update process. This requirement can be achieved by the Courant-Friedrichs-Lewy (CFL) condition $1 > \max(u \frac{\Delta t}{\delta x}, v \frac{\Delta t}{\delta y})$ [Carlson 2004]. Where u and v are the scalar value components of the velocity vector and x, y are the step length and step width of the grid. This condition is also valid for three dimensional purposes ($1 > \max(u \frac{\Delta t}{\delta x}, v \frac{\Delta t}{\delta y}, w \frac{\Delta t}{\delta z})$).

This condition states that the time step must be small enough to make sure information does not travel across more than one cell at a time. [Carlson 2004]

This technique is usefull for any solution which is numerically unstable as it is given by [Carlson 2004] or [Foster and Metaxas 1996].

4 Rendering

Rendering of grid based solutions is a widely discussed topic in computer graphics scene. There are many rendering techniques which can be used. A few of them are described in the following paragraphs.

Texture Rendering Two dimensional grids and textures are quite the same. Fluid simulators in two dimensions often store their values direct on texture units of the graphics hardware. This strategy has many advantages. To render a texture is one of the most basic features of any graphic card which leads to a huge acceleration boost in comparison with other rendering strategies. This rendering strategy is predestinated for realtime purposes. There are restrictions when using texture based rendering for fluid simulations which leads to simplifications and further to a lack of accuracy [Harris 2004].



Figure 10: shows fluid rendered to texture and mapped onto vase. [Stam 2001]

Height Field Rendering There are also much simpler techniques like height field rendering. This technique just needs a two dimensional grid and an array in which the height values are stored for each of the two dimensional grid positions. The implementation is now quite simple because the three dimensional spatial coordinates for the surface are given by the two dimensional grid and the height field values. This rendering strategy can be used for a visualisation of liquids like water.

Surface Rendering Another method to visualise a three dimensional fluid is to use an isosurface [Wikipedia 2007b]. An Isosurface connects scalar values with similar characteristic (pressure, temperature, velocity, density) within a volume. This can be used for reconstructing surfaces in a volumetric grid. For this purpose the marching cube algorithm is used [Lorensen and Cline 1987]. This algorithm generates triangle models of the isosurface by iteratively visiting each grid cell. Any of the volumetric grid cell is a

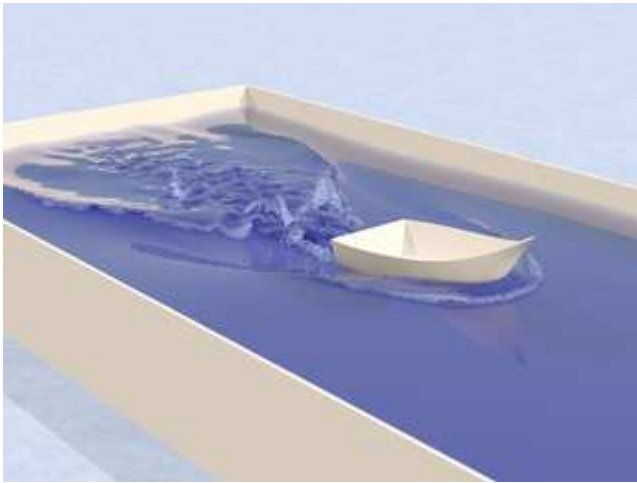


Figure 11: shows water rendered with height field rendering. [Frank Lossaso and Fedkiw 2004]

cube and also called voxel. The eight voxel corners are now used to reconstruct the surface patch cutting the cube. This is done by using surface shapes for any of the $2^8 = 256$ occurring cases where a surface can intersect a cube. The result is a reconstruction of the surface given by the volume.

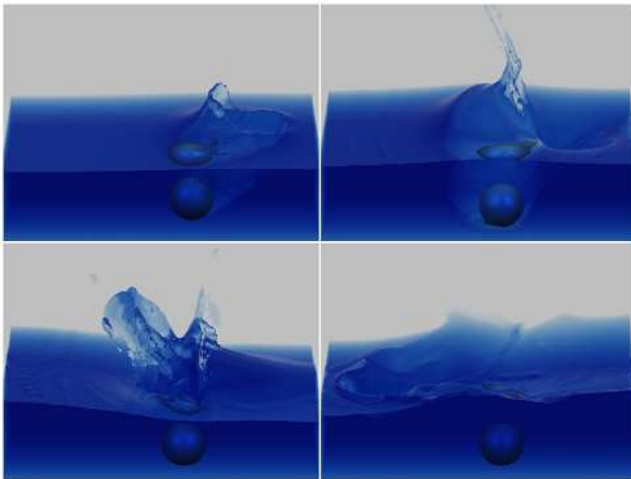


Figure 12: shows water rendered via isosurface. [Foster and Fedkiw 2001]

Volume Rendering The two dimensional grid solution can easily be enhanced to a three dimensional grid based solution, because the mentioned solutions are not restricted on two dimensions. Three dimensional solutions can be used with volume rendering techniques as it is described in the work of [Levoy 1988]. Volume Rendering techniques allow to show semi transparent fluid. This means natural phenomena like smoke, fire or other gases can be visualised as volume and not just their surface. Therefore many fluid simulations are using this strategy for their renderings.

[Ronald Fedkiw and Jensen 2001] shows a way to enhance the work of [Stam 1999] with visual output.



Figure 13: is created with MAYA Fluid Effects and shows volume rendered nebula and fire. [Stam 2001]

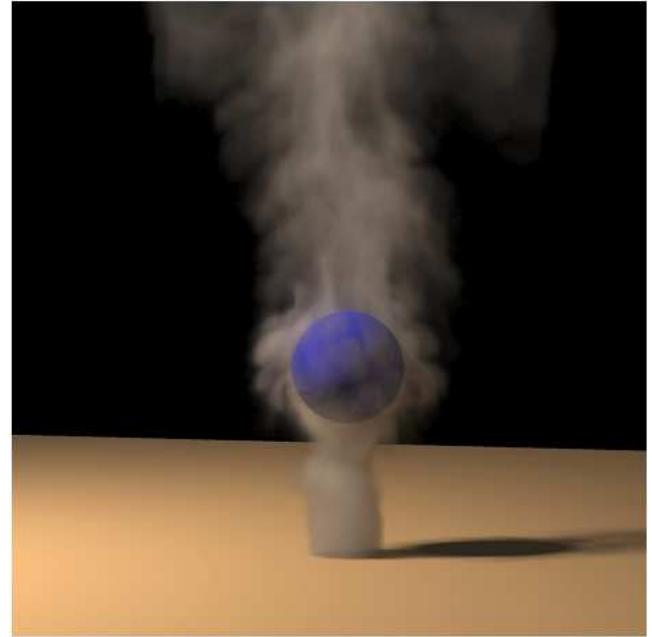


Figure 14: shows volume rendered mapped smoke using photon mapping as lighting technique. [Ronald Fedkiw and Jensen 2001]

5 Conclusion

The physically based Navier-Stokes equations allow a very realistic simulation of natural phenomena. This solution can easily be implemented for realtime purposes with the cost of accuracy because of the use of small resolution and simplification in the physical descriptions. The lack of accuracy in realtime applications is therefore the biggest con. Anyway the approach can be implemented numerically stable and it can be extended with other physical descriptions. Maya has implemented the numerically stable fluid simulation approach given by [Stam 1999] in their application MAYA Fluid Effects (Figure 13).

6 Web Information

The full source code of Joe Stam's implementation is available at the following web page: <http://www.acm.org/jgt/papers/Stam01>. An exemplary application based on this approach is available at the web page: www.plasmapong.com.

References

- ANDERSSON, L. 2004. *Real-Time Fluid Dynamics for Virtual Surgery*. Master's thesis, Chalmers University of Technology, Gothenburg.
- BLACK, N., AND MOORE, S., 2003. Gauss-seidel method. From MathWorld—A Wolfram Web Resource, created by Eric W. Weisstein. <http://mathworld.wolfram.com/Gauss-SeidelMethod.html>.
- CARLSON, M. 2004. *Rigid, Melting, and Flowing Fluid*. PhD thesis, Georgia Institute of Technology, Atlanta.
- FOSTER, N., AND FEDKIW, R., 2001. Practical animation of liquids. In Proceedings of the 28th Annual Conference on Computer Graphics and interactive Techniques SIGGRAPH.
- FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids (extended). In *Graphical Models and Image Processing*, 471483.
- FOSTER, N., AND METAXAS, D., 1997. Controlling fluid animation. In Proceedings of the 1997 Conference on Computer Graphics international.
- FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of hot, turbulent gas. *Computer Graphics Forum 16*, 3, 181188.
- FRANK LOSSASO, F. G., AND FEDKIW, R., 2004. Simulating water and smoke with an octree data structure. ACM SIGGRAPH 2004 Papers.
- HARRIS. 2004. Fast fluid dynamics simulation on the gpu. In *GPU GEMS Programming Techniques, Tips, and Tricks for Real-Time Graphics*, 637–665.
- IRVING, G., G. E. L. F., AND FEDKIW, R., 2006. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Transactions on Graphics*.
- LEVOY. 1988. Display of surfaces from volume data. *IEEE Computer Graphics and Applications 8*, 3, 2937.
- LORENSEN, W., AND CLINE, H. 1987. Marching cubes: a high resolution 3d surface reconstruction algorithm. *Proceedings of Siggraph 21*, 4, 163169.
- MARK CARLSON, PETER J. MUCHA, R. B. V. H. I., AND TURK, G., 2002. Melting and flowing. ACM SIGGRAPH Symposium on Computer Animation, July.
- RONALD FEDKIW, J. S., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Graphical Models and Image Processing*, 1522.
- STAM, J. 1997. Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Computer Graphics Forum 16*, 3, 159–164.
- STAM, J., 1999. Stable fluids. In SIGGRAPH 99 Conference Proceedings, Annual Conference Series, August.
- STAM, J. 2001. A simple fluid solver based on the fft. *journal of graphics tools 6*, 2, 43–52.
- STAM, J., 2003. Real-time fluid dynamics for games. Proceedings of the Game Developer Conference, March.
- TREUILLE, A., M. A. P. Z., AND STAM. 2003. eyframe control of smoke simulations. In *ACM Transactions on Graphics*, 716–723.
- WIKIPEDIA, 2007. gravity. From Wikipedia—Web Resource <http://en.wikipedia.org/wiki/Newton>.
- WIKIPEDIA, 2007. Isosurface. From Wikipedia—Web Resource <http://en.wikipedia.org/wiki/Isosurface>.
- WITKIN, A., AND BARAFF, D., 1997. Physically based modeling: Principles and practice. Online Siggraph '97 Course notes.