Equation of State Solvers for Smoothed Particle Hydrodynamics

Julian Karrer

$$\frac{D\vec{\boldsymbol{v}}}{Dt} = -\frac{1}{\rho}\nabla p + \nu\,\nabla^2\vec{\boldsymbol{v}} + \vec{\boldsymbol{b}}^{ext}$$

Figure 1: The Navier-Stokes momentum equation for incompressible flow. This title page itself is used as a simulation domain in which this equation is solved, highlighting the solver's ability to handle complex boundary conditions and resolve details while maintaining low levels of compression (here: $\rho_{err}^{max} < 0.1\%$ for $N > 250k$ particles).

Lab Course
Master of Science in Computer Science

Faculty of Engineering
Department of Computer Science
Supervised by Prof. Dr.-Ing. Matthias Teschner

universität freiburg

Contents

Chapter 1

Introduction

Smoothed Particle Hydrodynamics, or SPH for short, has established itself as a particle-based method of fluid simulation in Computer Graphics for its ability to handle free surface flows with complex boundary conditions and its versatility in the materials that can be simulated using a unified framework, amongst other benefits.

This report documents not only the implementation of an equation-of-state based solver and variations thereof, but attempts to frame the problem of fluid simulation from the more general perspective of numerically solving a partial differential equation by covering the main aspects common to such problems: understanding the underlying governing equations, the initial conditions, boundary conditions and the numerical scheme used to discretize the equations and propagate a solution forwards in time. This is reflected in the structure of the report, which first derives the Navier-Stokes equations that underpin the fluid simulation in chapter 2, stripping the governing equations of their mystery. Smoothed Particle Hydrodynamics is then discussed as a possible discretization of the equations in chapter 3, yielding three variations of a fluid solver presented in chapter 4, which build up from a simple solver to an iterative version that can simulate large scenes with low compressions robustly, as demonstrated by the title page and Figure 1. The boundary conditions and initial conditions are discussed in chapter 5, with a focus on the often neglected aspect of how the geometry of the initial sampling of a field can influence stability and isotropy. These aspects, as well as the differing behaviour of the presented variants of solvers, are then analysed in chapter 6, yielding some insight into the effects that certain choices of parameters can have on the simulation.

This report attempts to repeatedly ask *why* an aspect of fluid simulation with SPH is implemented the way it is, instead of solely focusing on *how* it is implemented. Lagrangian methods are contrasted with Eulerian approaches to explain why an especially convenient form of the governing equations can be obtained using this mesh-free method. By deriving the Navier-Stokes equations, the simplifying assumptions they rely on are revealed, both building an understanding as to why the equations consist of their respective terms and how they might be adapted when such assumptions are broken. The question of 'why' in regard to the discretization of fields using SPH is answered by considering what it means to sample a continuum at discrete locations and how a Gaussian-like kernel function with spherical symmetry might be a particularly natural and in some regards optimal choice for this problem. In building up to an iterative solver step by step, the constituent components can be understood and the question of why the pressure computation is split off and iterated upon can be answered. When considering boundary conditions, simplifying assumptions that allow for single-layer, non-uniform samplings are motivated by again building up from simpler, less powerful methods of boundary representation and considering their respective flaws. It is then asked why field are initially sampled in one lattice or another, why samplings should be regular or pseudorandom and why uniform masses or uniform densities might be preferred when initializing a system, when both cannot easily be obtained in conjunction. Lastly, the analysis attempts to shine a light on why certain parameters affect the fluid simulation as they do, using techniques that analyse the structure in space, behaviour in the frequency domain and shape in parameter-space that each of the solvers and initial conditions result in.

Chapter 2

Governing Equations of Fluid Flow

In an attempt to create a numerical solver for fluid dynamics problems, the governing equations of the underlying physical process must first be understood and formulated. Only then can an appropriate discretization be applied to numerically solve for desired properties of a system. In this chapter, the abstractions of continuum mechanics are used as a framework to describe incompressible flow. Physical principles such as conservation of mass and momentum are used to derive the continuity and momentum equations which encode them, then augmented by constitutive relations which describe properties of Newtonian fluids to finally yield the Navier-Stokes equations as governing equations.[1][2]

The particular form of these equations will favour a Lagrangian view of the system, in which the frame of reference in which quantities are described is advected along with the flow of the fluid itself, which will seamlessly integrate with the discretization scheme later used to derive workable numerical algorithms.

2.1 Lagrangian and Eulerian Continuum Mechanics

The purpose of our mathematical modelling of fluids is to simulate fluid dynamics at macroscopic scales with numerical methods. We know that fluids consist of innumerable molecules, and smaller particles yet, interacting in complex ways, which give rise to emergent properties that we observe on a macroscopic scale. Instead of resolving all scales and simulating from quantum mechanical principles up, we content with modelling the emergent properties themselves, focusing on the question of how fluids behave on our desired scales instead of what gives rise to such effects. Our macroscopic scale is so many orders of magnitude larger than the quantized world of particles, that we can reasonably assume quantities describing the fluid to be continuous and tackle them with the tools of calculus. This gives rise to the field of **Continuum Mechanics**.
In the following derivations, two major points of view can be taken, which produce different but equivalent forms of equations: the Eulerian or conservation forms, and the Lagrangian or nonconservation forms of the equations[1].

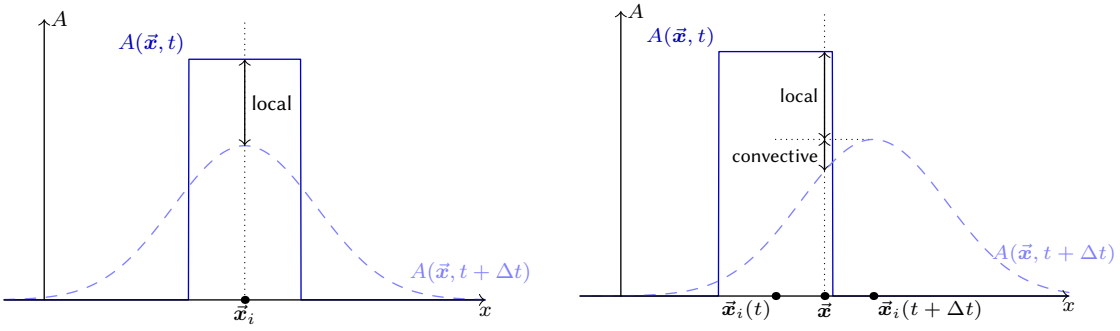
Using the assumption from continuum mechanics that quantities of our fluid are continuously distributed in space and asserting that they be differentiable, we can define derivatives on them. The two major forms of equations arise from a different interpretation of the so-called substantial derivative[1] or material derivative[2] $\frac{D}{Dt}$. This operator describes the instantaneous time rate of change of a quantity of a continuum element as it moves through space[1]. This movement through space however can be observed from different frames of reference:

- a frame that is advected along with the flow of the fluid, in which the continuum element observed appears unmoving
- a frame that is unmoving in space and located at a fixed point, observing the flow of the fluid as continuum elements move through it

For both frames of reference, it can be derived that the material derivative in vector notation is[1]:

$$\frac{D}{Dt} = \underbrace{\frac{\partial}{\partial t}}_{\text{local derivative}} + \underbrace{(\vec{\boldsymbol{v}} \cdot \nabla)}_{\text{convective derivative}} \tag{2.1}$$

where $\vec{\boldsymbol{v}}$ is the velocity of the element and ∇ denotes the differential operator $\left(\frac{\partial}{\partial x_0}, \frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n}\right)^T$ in n dimensions[1]. If an Eulerian view is chosen, there is an additional term for the convective derivate, which describes a rate of change of a quantity at a fixed point due to movement of the fluid. If a Lagrangian view



(a) Lagrangian view (b) Eulerian view

Figure 2.1: A field quantity A in one dimension is shown at some time t and a later time $t + \Delta t$ where the distribution of A has changed due to some diffusive process and the quantity was advected in positive x-direction. In the Lagrangian view, changes in a quantity A are evaluated at the advected position $\vec{x}_i$ at any point in time and only the local derivative is needed to describe the change in A at $\vec{x}$. In the Eulerian view there are two reasons for A at a point $\vec{x}$ fixed in space to change: the local derivative due to the diffusive process and the convective derivative due to the advection of the quantity with the velocity field.

is taken, the reference frame is advected with the velocity $\vec{v}$, precisely such that the convective derivative is zero and the material derivative simply becomes the total time derivative of a quantity. The difference between the two views is illustrated in Figure 2.1. In a way, the Lagrangian frame of reference is chosen precisely such that only local derivatives suffice to describe material derivatives by using a coordinate transformation defined by the velocity field.

How this simpler, Lagrangian form can be used largely depends on the later choice of discretization: discretizing space and tracking the fluid that moves through it favours an Eulerian framework, while discretizing the continuum into particles and sampling quantities only at advected particle positions makes the Lagrangian view convenient.
As is common for SPH discretizations, we will elect the Lagrangian view since it not only harmonizes well with particle-based discretizations but holds additional desirable properties such as making conservation of mass trivial to implement and enabling solving the Navier-Stokes equations for primitive quantities instead of flux quantities that may cause drift instead of oscillations due to numerical inaccuracy. We state all following equations in the Lagrangian, nonconservation form.

2.2 The Continuity Equation

Using the Lagrangian view of continuum mechanics, we can apply laws of conservation to derive equations that express invariants of each fluid element with respect to time, which is an important step towards describing the dynamics of the system as time evolves. One such equation is the **CONTINUITY EQUATION**, which expresses conservation of mass:
Consider an infinitesimally small volume element $\delta\mathcal{V}$ with density ρ. The mass of the volume δm is simply[1]:

$$\delta m = \rho \delta \mathcal{V} \tag{2.2}$$

and is invariant under the material derivative in the Lagrangian reference frame[1]:

$$\frac{D\delta m}{Dt} = 0 \qquad \textit{conservation of mass} \tag{2.3}$$

$$= \frac{D\rho\delta\mathcal{V}}{Dt} \qquad \textit{identity 2.2} \tag{2.4}$$

$$= \delta\mathcal{V}\frac{D\rho}{Dt} + \rho\frac{D\delta\mathcal{V}}{Dt} \qquad \textit{product rule of calculus} \tag{2.5}$$

$$= \frac{D\rho}{Dt} + \rho\left(\frac{1}{\delta\mathcal{V}}\frac{D\delta\mathcal{V}}{Dt}\right) \qquad \textit{divide by } \delta\mathcal{V} \tag{2.6}$$

We can now apply the **DIVERGENCE THEOREM** to relate $\frac{D\mathcal{V}}{Dt}$ to the divergence of the velocity across the volume of the element, where $\partial\mathcal{V}$ is its surface and $\vec{n}$ the corresponding unit normal vector[1]:

$$\frac{D\mathcal{V}}{Dt} = \oint_{\partial\mathcal{V}} \vec{v}\cdot\vec{n}\, dS = \int_{\mathcal{V}} (\nabla\cdot\vec{v})\, d\mathcal{V} \tag{2.7}$$

As the volume $\mathcal{V}$ approaches the infinitesimal volume element $\delta\mathcal{V}$ of interest, the velocity in the volume becomes constant, the integral vanishes, and it holds that[1]:

$$\frac{D(\delta\mathcal{V})}{Dt} = (\nabla \cdot \vec{v})\,\delta\mathcal{V} \tag{2.8}$$

Substituting Equation 2.8 into Equation 2.6 we finally obtain the continuity equation:

$$\boxed{\frac{D\rho}{Dt} + \rho\,(\nabla \cdot \vec{\boldsymbol{v}}) = 0} \tag{2.9}$$

This is one of the Navier-Stokes equations in its derivative form, as opposed to the more general integral form[1]. When we additionally assume that the fluid is incompressible across a wide range of pressures, as is often done when simulating hydrodynamics, we can assert that the density of the fluid element in a Lagrangian reference frame is constant, meaning:

$$\frac{D\rho}{Dt} = 0 \tag{2.10}$$

and therefore the velocity field of the flow for constant density is divergence-free[3]:

$$\nabla \cdot \vec{\boldsymbol{v}} = 0 \tag{2.11}$$

In the following sections, the fluid will generally be assumed to be incompressible.

An alternative derivation of the continuity equation uses the **Reynolds Transport Theorem**, which describes the material derivative of a scalar or tensor quantity $q(\vec{\boldsymbol{x}}, t)$ integrated over a volume as the sum of its time rate of change within the volume and the flux of the quantity through the volume's surface[3]:

$$\frac{D}{Dt}\int_{\mathcal{V}} q(\vec{\boldsymbol{x}}, t)\,dV = \int_{\mathcal{V}} \frac{\partial q(\vec{\boldsymbol{x}}, t)}{\partial t}\,dV + \oint_{\partial\mathcal{V}} q(\vec{\boldsymbol{x}}, t)(\vec{\boldsymbol{v}} \cdot \vec{\boldsymbol{n}})\,dS \tag{2.12}$$

This derivation goes as follows[3]:

$$0 = \frac{D}{Dt}\int_{\mathcal{V}} \rho\,dV \qquad \textit{conservation of mass} \tag{2.13}$$

$$= \int_{\mathcal{V}} \frac{\partial\rho}{\partial t}\,dV + \oint_{\partial\mathcal{V}} \rho(\vec{\boldsymbol{v}} \cdot \vec{\boldsymbol{n}})\,dS \qquad \textit{Reynolds Transport Theorem} \tag{2.14}$$

$$= \int_{\mathcal{V}} \frac{\partial\rho}{\partial t}\,dV + \int_{\mathcal{V}} \nabla \cdot (\rho\vec{\boldsymbol{v}})\,dV \qquad \textit{Divergence Theorem} \tag{2.15}$$

$$= \int_{\mathcal{V}} \left(\frac{\partial\rho}{\partial t} + \nabla \cdot (\rho\vec{\boldsymbol{v}})\right) dV \qquad \textit{combine integrals} \tag{2.16}$$

$$= \int_{\mathcal{V}} \left(\frac{D\rho}{Dt} + \rho\nabla \cdot \vec{\boldsymbol{v}}\right) dV \qquad \textit{constant density, Lagrangian framework} \tag{2.17}$$

$$\overset{\forall\mathcal{V}}{\Longrightarrow} \frac{D\rho}{Dt} + \rho\,(\nabla \cdot \vec{\boldsymbol{v}}) = 0 \qquad \textit{integral holds for all } \mathcal{V} \tag{2.18}$$

This use of the Reynolds Transport Theorem is very similar to the derivation that follows in section 2.3, which is why this alternative derivation was stated.

2.3 The Cauchy Momentum Equation

Mass is not the only conserved quantity that can be formulated in terms of a volume integral which can be transformed into a more convenient form using Reynolds Transport Theorem: a vital step in the derivation of the Navier-Stokes equations comes from applying the same concept to the conservation of momentum. In fact, the **Cauchy momentum equation**, which is the general case of the more specific momentum equation used in the Navier-Stokes equations, can be derived similarly to section 2.2, additionally using the continuity equation itself and Newton's second law.

We begin by observing that the change of momentum of a fluid volume $\mathcal{V}$ can be defined as the material derivative of the momentum $\int_{\mathcal{V}}(\rho\vec{\boldsymbol{v}})\,dV$ and simplify the resultant expression[3]:

$$\frac{D}{Dt}\int_{\mathcal{V}}(\rho\vec{\boldsymbol{v}})\,dV \qquad \textit{define change in momentum} \tag{2.19}$$

$$=\int_{\mathcal{V}}\frac{\partial(\rho\vec{\boldsymbol{v}})}{\partial t}\,dV+\oint_{\partial\mathcal{V}}\rho\vec{\boldsymbol{v}}(\vec{\boldsymbol{v}}\cdot\vec{\boldsymbol{n}})\,dS \qquad \textit{Reynolds Transport Theorem 2.12} \tag{2.20}$$

$$=\int_{\mathcal{V}}\frac{D}{Dt}(\rho\vec{\boldsymbol{v}})\,dV+\int_{\mathcal{V}}(\rho\vec{\boldsymbol{v}})\nabla\cdot\vec{\boldsymbol{v}}\,dV \qquad \textit{Divergence Theorem} \tag{2.21}$$

$$=\int_{\mathcal{V}}\rho\frac{D\vec{\boldsymbol{v}}}{Dt}+\vec{\boldsymbol{v}}\frac{D\rho}{Dt}+(\rho\vec{\boldsymbol{v}})\nabla\cdot\vec{\boldsymbol{v}}\,dV \qquad \textit{product rule on first integral} \tag{2.22}$$

$$=\int_{\mathcal{V}}\rho\frac{D\vec{\boldsymbol{v}}}{Dt}+\vec{\boldsymbol{v}}\underbrace{\left(\frac{D\rho}{Dt}+\rho\nabla\cdot\vec{\boldsymbol{v}}\right)}_{\text{continuity equation}=0}dV \qquad \textit{factor out } \vec{\boldsymbol{v}} \tag{2.23}$$

$$=\int_{\mathcal{V}}\rho\frac{D\vec{\boldsymbol{v}}}{Dt}\,dV \tag{2.24}$$

Then, we use Newton's second law, best known in its form $F=m\vec{\boldsymbol{a}}$, to assert that this change in momentum $m\vec{\boldsymbol{a}}$ is equal to the sum of forces exerted on the fluid volume, which can be decomposed into body forces $\vec{\boldsymbol{b}}^{ext}$ per unit mass[3] that act on the entire fluid mass homogeneously 'at a distance'[1], like gravity for example, and into surface forces described by stress vectors $\vec{\boldsymbol{t}}$ integrated over the fluid element's surface[3]:

$$\int_{\mathcal{V}}\rho\frac{D\vec{\boldsymbol{v}}}{Dt}\,dV=\oint_{\partial\mathcal{V}}\vec{\boldsymbol{t}}\,dS+\rho\vec{\boldsymbol{b}}^{ext} \tag{2.25}$$

One can define the **Cauchy stress tensor** $\mathbb{T}$ (sometimes referred to as σ) for the material such that it satisfies[3] $\mathbb{T}\vec{\boldsymbol{n}}=\vec{\boldsymbol{t}}$. Then, the divergence theorem may be applied again and the total forces acting on the fluid element can be written as:

$$\int_{\mathcal{V}}\nabla\cdot\mathbb{T}\,dV+\rho\vec{\boldsymbol{b}}^{ext} \tag{2.26}$$

Setting the expressions for total force in Equation 2.26 and total change of momentum in Equation 2.24 equal according to Newton's Law, we obtain:

$$\int_{\mathcal{V}}\rho\frac{D\vec{\boldsymbol{v}}}{Dt}-\nabla\cdot\mathbb{T}-\rho\vec{\boldsymbol{b}}^{ext}\,dV=0 \tag{2.27}$$

From this, we have obtained the **Cauchy Momentum Equation** as our equation of motion[2]:

$$\boxed{\rho\frac{D\vec{\boldsymbol{v}}}{Dt}=\nabla\cdot\mathbb{T}+\rho\vec{\boldsymbol{b}}^{ext}} \tag{2.28}$$

2.4 The Lagrangian Navier-Stokes Equations

With the Cauchy momentum equation we have reached the end of what can be modelled using general physical principles and continuum mechanics and is valid for a range of materials. To close the system of equations for fluid flow, generality must be given up and specific assumptions about the behaviour of fluids must be used to model the specific stress tensor $\mathbb{T}$ representing incompressible, linearly viscous or Newtonian fluids. In order to derive the form of the tensor, we make the further assumptions about the fluid that will later be clarified:

1. Fluids cannot sustain shear stresses when in rigid body motion.
2. Viscosity depends on the symmetric component of the gradient of velocity, it is linearly proportional to the rate of deformation tensor.

All remaining terms of the Cauchy momentum equation are clear, only the stress tensor $\mathbb{T}$ needs to be elaborated upon. First, it can be noted that $\mathbb{T}$ is a linear transformation[3] and that the tensor is symmetric[3], as in equal to its transpose $\mathbb{T}^T=\mathbb{T}$ or $\mathbb{T}_{ij}=\mathbb{T}_{ji}$. This means that in three dimensions for example, only six degrees of freedom actually exist in this tensor[4].

The element $\mathbb{T}_{ij}$ expresses a stress along some axis $\vec{e_i}$ acting on a plane perpendicular to $\vec{e_j}$, which means that the diagonal elements $\mathbb{T}_{ii}$ are normal stresses called *tensile stresses* for negative values and *compressive stresses* for positive values of $\mathbb{T}_{ii}$[3], while $\forall i \neq j : \mathbb{T}_{ij}$ refer to *shear stresses*[1].

To make this tensor more tractable, it can be assumed that a fluid is a material which cannot sustain shear stresses when in rigid body motion, including rest[3] (assumption 1) - this means that when in rigid body motion, the stress vector on any plane is normal to that plane[3], the stress is therefore isotropic and $\mathbb{T}$ must be represented by the only isotropic second order tensor $\lambda\mathbb{1}$ or $\lambda\delta_{ij}$ for some $\lambda \in \mathbb{R}$ where δ_{ij} is the Kronecker delta[5]. This motivates a decomposition of $\mathbb{T}$ for any general motion into a sum of an isotropic tensor describing *volumetric stress* caused by pressure forces and the *deviatoric stress* $\mathbb{V}$ which simply describes deviation of the total stress $\mathbb{T}$ from the volumetric stress[6]:

$$\mathbb{T} = \mathbb{V} - p\mathbb{1} \tag{2.29}$$

Conventionally, the pressure p is defined such that a positive pressure causes a negative stress, meaning the pressure acts normal to the surface and is directed into the fluid volume $\mathcal{V}$[4]. For a fluid at rest $\mathbb{V}_{ij} = 0$ holds and the normal stress is isotropically $-p$ according to *Pascal's law*[5]. Equation 2.29 decomposes stresses into a part caused by pressure and one caused by viscosity, which is why $\mathbb{V}$ is sometimes referred to as the *viscous stress tensor*[4]. Viscosity can be thought of as internal friction in a fluid or its resistance to deformation.

The remaining term $\mathbb{V}$ is caused by viscosity and modelled according to assumption 2 in terms of the gradient of the velocity. This makes intuitive sense: where the velocity is homogeneous, and the gradient is zero, there is no friction between fluid elements - where the velocity differs greatly, there is more friction. Since velocity is a vector quantity, the gradient $\nabla\vec{v}$ is a tensor[4]:

$$(\nabla\vec{v})_{ij} = \partial_j v_i = \frac{\partial v_i}{\partial x_j} \tag{2.30}$$

As always, we can decompose this tensor $\mathbb{L} := \nabla\vec{v}$ into a sum of a symmetric and an antisymmetric part[3]:

$$\mathbb{L} = \mathbb{D} + \mathbb{W} \tag{2.31}$$

$$\mathbb{D} = \frac{1}{2}\left(\mathbb{L} + \mathbb{L}^T\right) \tag{2.32}$$

$$\mathbb{W} = \frac{1}{2}\left(\mathbb{L} - \mathbb{L}^T\right) \tag{2.33}$$

$$\tag{2.34}$$

$\mathbb{D}$ is referred to as the **RATE OF DEFORMATION TENSOR** and $\mathbb{W}$ is called the **SPIN TENSOR**.

This decomposition is convenient since the spin tensor does not contribute to viscosity and only the rate of deformation tensor may be focused on. Note that since the deviatoric stress $\mathbb{V}$ we are trying to approximate is symmetric, and it only makes sense to use the symmetric component of the velocity gradient to model it.

Intuitively, the spin tensor encodes the rotational component of the velocity gradient, and a steadily rotating fluid (where $\mathbb{D}_{ij} = 0$) is like a rigid body rotation: the relative positions of the fluid elements do not change, only their orientation with respect to a fixed reference frame, and therefore there is no friction. There is a vector $\vec{\omega}$ such that for any $\vec{v}$ it holds that $\mathbb{W}\vec{v} = \vec{\omega} \times \vec{v}$, where $\vec{\omega}$ points in the axis of rotation with a length of the angular velocity[3]. This is why the spin tensor is closely related to the vorticity tensor $2\mathbb{W}$[3]. In fact, enforcing that viscosity shall not affect the rotational component of velocity gradients and preserving accurate vorticity is key to accurately simulating turbulences in incompressible flows and conserving angular momentum[7].

Focusing further on the rate of deformation tensor, assumption 2 can now fully be appreciated. One defining characteristic of Newtonian fluids is the assumption dating back to Isaac Newton that viscosity depends *linearly* on the rate of deformation tensor[1]. This means that terms of an order higher than linear may be neglected for small velocity gradients[4] and constant terms cannot occur since shear stress is only proportional to the rate of deformation, not deformation itself[3]: if a shear stress is applied to a fluid it will eventually continuously deform at some non-zero rate but will remain in that deformed state if the

stress is removed, unlike purely elastic materials[3]. In other words $\mathbb{V}$ must vanish when the velocity is homogeneous since there is no friction in that case[4].

We now know that for incompressible fluids $\mathbb{V}$ is of the form[4]:

$$\mathbb{V} = 2\mu\mathbb{D} + \lambda(\nabla\cdot\vec{\boldsymbol{v}})\mathbb{1} \tag{2.35}$$

$$= \frac{2\mu}{2}\left((\nabla\vec{\boldsymbol{v}}) + (\nabla\vec{\boldsymbol{v}})^T\right) + \underbrace{\lambda(\nabla\cdot\vec{\boldsymbol{v}})\mathbb{1}}_{\text{incompressibility} = 0} \tag{2.36}$$

$$= \mu\left((\nabla\vec{\boldsymbol{v}}) + (\nabla\vec{\boldsymbol{v}})^T\right) \tag{2.37}$$

where μ is the dynamic viscosity[1] or first-order viscosity[4]. A second-order viscosity λ exists for compressible flows[1], but can be neglected here.

Combining the deviatoric stress with the volumetric stress, the **CONSTITUTIVE RELATION** for the stress tensor $\mathbb{T}$ of an incompressible, Newtonian fluid is finally obtained[2]:

$$\boxed{\mathbb{T} = -p\mathbb{1} + \mu\left((\nabla\vec{\boldsymbol{v}}) + (\nabla\vec{\boldsymbol{v}})^T\right)} \tag{2.38}$$

With the constitutive relation in hand, the Cauchy momentum equation can be revisited, and Equation 2.38 can be inserted into Equation 2.28:

$$\rho\frac{D\vec{\boldsymbol{v}}}{Dt} = \nabla\cdot\left(-p\mathbb{1} + \mu\left((\nabla\vec{\boldsymbol{v}}) + (\nabla\vec{\boldsymbol{v}})^T\right)\right) + \rho\vec{\boldsymbol{b}}^{ext} \qquad \textit{insert Eq. 2.38 into Eq. 2.28} \tag{2.39}$$

$$\rho\frac{D\vec{\boldsymbol{v}}}{Dt} = \nabla\cdot(-p\mathbb{1}) + \mu\nabla\cdot\left((\nabla\vec{\boldsymbol{v}}) + (\nabla\vec{\boldsymbol{v}})^T\right) + \rho\vec{\boldsymbol{b}}^{ext} \qquad \nabla\cdot \textit{ is linear} \tag{2.40}$$

$$\frac{D\vec{\boldsymbol{v}}}{Dt} = -\frac{1}{\rho}\nabla p + \nu\nabla\cdot\left((\nabla\vec{\boldsymbol{v}}) + (\nabla\vec{\boldsymbol{v}})^T\right) + \vec{\boldsymbol{b}}^{ext} \qquad \nabla\cdot(-p\mathbb{1}) = -\nabla p\textit{, divide by } \rho \tag{2.41}$$

$$\frac{D\vec{\boldsymbol{v}}}{Dt} = -\frac{1}{\rho}\nabla p + \nu\left(\underbrace{\nabla\cdot(\nabla\vec{\boldsymbol{v}})}_{=\nabla^2\vec{\boldsymbol{v}}} + \underbrace{\nabla\cdot(\nabla\vec{\boldsymbol{v}})^T}_{=0}\right) + \vec{\boldsymbol{b}}^{ext} \qquad \nabla\cdot \textit{ is linear} \tag{2.42}$$

$$\frac{D\vec{\boldsymbol{v}}}{Dt} = -\frac{1}{\rho}\nabla p + \nu\,\nabla^2\vec{\boldsymbol{v}} + \vec{\boldsymbol{b}}^{ext} \tag{2.43}$$

A few things of note happen in this derivation:

- The kinematic viscosity ν is defined as $\frac{\mu}{\rho}$ and inserted in Equation 2.41
- The identity $\nabla\cdot(-p\mathbb{1}) = -\nabla\cdot\begin{bmatrix}p & 0 & 0\\ 0 & p & 0\\ 0 & 0 & p\end{bmatrix} = -\begin{bmatrix}\partial p/\partial x\\ \partial p/\partial y\\ \partial p/\partial y\end{bmatrix} = -\nabla p$ is used in Equation 2.41.
- For sufficiently smooth $\vec{\boldsymbol{v}}$ and $\nabla\cdot\vec{\boldsymbol{v}} = 0$ one can show using the Theorem of Schwarz that $\nabla\cdot(\nabla\vec{\boldsymbol{v}}) = \nabla^2\vec{\boldsymbol{v}}$ as annotated in Equation 2.42[4].
- Similarly, in Equation 2.42 $\nabla\cdot(\nabla\vec{\boldsymbol{v}}^T) = \nabla(\nabla\cdot\vec{\boldsymbol{v}}) = 0$ is used[4], since the continuity equation for fluids of homogeneous density implies $\nabla\cdot\vec{\boldsymbol{v}} = 0$.

With all this, the final Navier-Stokes momentum equation for incompressible Newtonian fluids in Lagrangian form is obtained in step 2.43:

$$\boxed{\frac{D\vec{\boldsymbol{v}}}{Dt} = -\frac{1}{\rho}\nabla p + \nu\,\nabla^2\vec{\boldsymbol{v}} + \vec{\boldsymbol{b}}^{ext}} \tag{2.44}$$

2.5 Equations of State

Although the momentum equation typically takes centre stage when discussing the Navier-Stokes equations, it is important to realize that the complete Navier-Stokes equations actually refer to a set of equations and the momentum equation cannot function on its own. At the very least, the continuity equation should be included, sometimes accompanied by an energy balance equation which is crucial to heat transport problems and formulates conservation of energy in viscous flows[1]. For the dynamics of the systems

considered here, the continuity and momentum equations appear to be sufficient, but one field quantity remains elusive until now: We have yet to discuss how to compute pressure.

When incompressibility is strongly enforced, the continuity equation is a constraint on the momentum equation that p can be chosen to fulfil, making it a Lagrange multiplier to the equation[2]. Since strongly enforced incompressibility generally requires solving a system to solve the Poisson equation for pressure and can be more involved, a more straightforward first approach is to employ an **Equation of State** to couple pressure to known quantities. Such an equation of state can be thought of intuitively as relating strain and stress, or a deformation of a material and the potential caused by this deformation, the negative gradient of which is a force. In this case a deviation of the fluid from its rest density ρ_0 or volume V_0 causes a pressure potential, the negative gradient of which is a pressure force that counteracts the deformation, in this case compression, as demonstrated in the hydrostatic case.

There are many such equations of state to choose from. While this choice indeed encodes different physical assumption about the fluid, the choice often appears to be motivated by implementation details and practical reasons rather than general physical principles. The equation of state should be chosen with the goal of low compressibility and well-behavedness of the system in mind and options include:

1. $p = k\rho$ or $p = k(\rho - \rho_0)$ from the ideal gas equation[8]
2. $p = k\left(\frac{\rho}{\rho_0} - 1\right)^\gamma$ which is sometimes referred to as Tait's equation[8], while others critique this attribution to Tait[9] and some refer to it as a form of the Murnaghan equation of state[10] or attribute its first use to Kirkwood[10]
3. $p = \max\left(0, k(\rho - \rho_0)\right)$ or $p = \max\left(0, k\left(\frac{\rho}{\rho_0} - 1\right)^\gamma\right)$ to prevent negative pressure values[2]

Option 3 in particular is used to penalize relative deviations from rest density while preventing negative pressure values that may cause undesired clumping artefacts when using SPH discretizations. In the implementation used for this report, the equation:

$$p = \max\left(0, k\left(\frac{\rho}{\rho_0} - 1\right)\right) \tag{2.45}$$

was chosen by inserting $\gamma = 1$ into option 2 and applying clamping.

While the equation of state does allow the computation of the unknown pressure, it does not appear to help close the Navier-Stokes equations, since the problem was only pushed back to the seemingly arbitrary choice or some parameter k representing stiffness. It is important to note that this parameter does not however govern the magnitude of pressure per se, but only the compressibility of the fluid[2], where larger values of k yield higher incompressibility but also higher pressure accelerations and therefore demand a higher resolution of time discretization to remain numerically stable[2] and satisfy the Courant-Friedrichs-Lewy condition[11], decreasing computational efficiency. In order to ensure that the assumption of incompressibility made in the derivation of the Navier-Stokes momentum equations hold, a sufficiently large stiffness k should be chosen for a given setting, such that compressibility becomes negligible.

Approximations for the choice of k in Murnaghan's equation such as $k \approx \frac{\rho_0 c_s}{\gamma}$ exist, where c_s is the speed of sound and relates to the speed of flow $\vec{v}_f$[8]. Other methods such as Predictive-Corrective Incompressible SPH (PCISPH for short) approximate a globally constant k specifically for SPH discretizations such that a more optimal trade-off of incompressibility and time step size might be realized. Generally, k might need to be tuned depending on the simulated scenario in a fluid solver employing an equation of state.

Chapter 3

Smoothed Particle Hydrodynamics

In chapter 2, the governing equations of fluid flow were derived in their Lagrangian differential form for continuous field quantities. To make the simulation of fluids tractable, these equations must now be discretized in space and time so that the evolution of the system can be numerically calculated.

The temporal domain is commonly discretized into global time steps Δt that propagate the solution of the system into the future. Numerically integrating the acceleration $\vec{\boldsymbol{a}}_i(t) = \frac{D\vec{\boldsymbol{v}}_i(t)}{Dt}$ from the left-hand side of the momentum equation (Equation 2.44) twice with respect to time yields a change in position $\Delta\vec{\boldsymbol{x}}$ that can be used to advect quantities. Symplectic Euler time integration (also referred to as semi-implicit Euler or Euler-Cromer) is very commonly used to achieve this[2]:

$$\vec{\boldsymbol{v}}_i(t + \Delta t) = \vec{\boldsymbol{v}}_i(t) + \Delta t\vec{\boldsymbol{a}}_i(t) \tag{3.1}$$

$$\vec{\boldsymbol{x}}_i(t + \Delta t) = \vec{\boldsymbol{x}}_i(t) + \Delta t\vec{\boldsymbol{v}}_i(t + \Delta t) \tag{3.2}$$

The subscript i in these equations indicates that quantities are evaluated at respective particle positions $\vec{\boldsymbol{x}}_i$, which are advected with the velocity field. This is why the Lagrangian form is applicable, and the material derivative can be implemented as a total derivative with respect to time.

The spatial discretization of the problem is less straightforward and yields different methods depending on the scheme chosen.

3.1 Spatial Discretization of the Continuum

The discretization chosen here makes use of **Smoothed Particle Hydrodynamics** or **SPH** for short, which was devised independently by Lucy[12] as well as Gingold and Monaghan[13] in 1977. Despite its name, this scheme has little to do with Hydrodynamics per se and does not even strictly require a particle representation of quantities to work, but rather is a general framework for the interpolation of field quantities stored at discrete locations to obtain a smooth function that can be evaluated at any location.

Since the Lagrangian framework tends to favour discretizing the continuum itself over the space it exists within, regions of the continuum are here represented by so-called particles with a singular position that represent some volume or, equivalently in the incompressible case with homogeneous density, mass. It is important to keep in mind that the word *particle* in this context refers not to a physical, elementary particle or a spherical object, but rather an abstract representation of a discrete, shapeless parcel of the continuum.

SPH can be derived by considering that these particles represent a sampling of the continuous fluid domain at singular points and can be expressed as Dirac-δ distributions weighted by some quantity. The δ-distribution can be defined as being normalized:

$$\int \delta(\vec{\boldsymbol{x}})\, dV = 1 \tag{3.3}$$

and obeying $\vec{\boldsymbol{x}} \neq \vec{\boldsymbol{0}} \implies \delta(\vec{\boldsymbol{x}}) = 0$. This results in a distribution that is zero everywhere but at a singularity at the origin, where a spike of undefined height shoots up and only the integral of the distribution across that spike is a well-defined function (δ itself is a *generalized function*, not one in the analytic sense[14]). The Dirac-δ can be thought of as the limit of a Gaussian distribution as the variance approaches zero and the distribution becomes ever higher and narrower[2].
For this distribution representing the particles, the identity holds that for any continuous, compactly supported function $A(\vec{\boldsymbol{x}})$[2]:

$$A(\vec{\boldsymbol{x}}) = (A * \delta)(\vec{\boldsymbol{x}}) = \int A(\vec{\boldsymbol{x}}')\delta(\vec{\boldsymbol{x}} - \vec{\boldsymbol{x}}')dV' \tag{3.4}$$

or the convolution of A with δ is A itself. This identity can be explained from the perspective of Fourier analysis, where the δ can be defined as the constant unit function in Fourier space and therefore $\delta = \mathcal{F}^{-1}(1)$. Since the convolution theorem applies, it then holds that a convolution in the spatial domain is equivalent to a multiplication in the transformed domain and vice-versa, resulting in a multiplication by one in the case of the convolution with a δ-distribution real space, and therefore an identity.

The key insight to SPH is that the δ-distribution can be approximated by a more well-behaved function with desirable properties such as smoothness, while approximately retaining the above identity. Such a function is referred to in SPH as a **KERNEL FUNCTION** W, *smoothing kernel*[2] or *broadening function*[12], since it broadens and smooths out the Dirac-δ distribution. With this, one can then derive[2]:

$$A(\vec{\boldsymbol{x}}) = (A * \delta)(\vec{\boldsymbol{x}}) \qquad \text{Equation 3.4} \tag{3.5}$$

$$= \int A(\vec{\boldsymbol{x}}')\delta(\vec{\boldsymbol{x}} - \vec{\boldsymbol{x}}')dV' \qquad \text{Def. of convolution, } \textit{sifting property} \text{ of } \delta^{14} \tag{3.6}$$

$$\approx \int A(\vec{\boldsymbol{x}}')W(\vec{\boldsymbol{x}} - \vec{\boldsymbol{x}}')dV' \qquad \text{approximate } \delta \text{ by } W \tag{3.7}$$

$$= \int \frac{A(\vec{\boldsymbol{x}}')}{\rho(\vec{\boldsymbol{x}}')}W(\vec{\boldsymbol{x}} - \vec{\boldsymbol{x}}')\underbrace{\rho(\vec{\boldsymbol{x}}')dV'}_{=dm'} \qquad \text{multiply by } \frac{\rho(\vec{\boldsymbol{x}}')}{\rho(\vec{\boldsymbol{x}}')} = 1 \tag{3.8}$$

$$\approx \sum_{\vec{\boldsymbol{x}}_j \in \mathcal{S}} A_j \frac{m_j}{\rho_j} W(\vec{\boldsymbol{x}} - \vec{\boldsymbol{x}}_j) \qquad \text{approximate Integral with discrete samples} \tag{3.9}$$

where subscripts denote the position where a quantity is evaluated as in $A_j := A(\vec{\boldsymbol{x}}_j)$ and $\mathcal{S}$ is a set of fluid samples. This leads to the general SPH approximation for any field quantity[2] A:

$$A_i = \sum_j A_j \frac{m_j}{\rho_j} W_{ij} \tag{3.10}$$

where the sample set $\mathcal{S}$ is implicit in the notation and $W_{ij} := W(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)$. The term $\frac{m_j}{\rho_j} = V_j$ can be seen as the fluid volume that sample j represents.

Note in particular that the mass density:

$$\rho_i = \sum_j m_j W_{ij} \tag{3.11}$$

is simply a sum over kernel functions weighted by the respective mass of samples[2]. Since mass can be perfectly conserved in a Lagrangian framework, this lends itself to fluid solvers that enforce density invariance as opposed to minimizing velocity divergence and the errors of which therefore result in volume oscillations rather than loss of volume and drift - this trade-off might be desirable but is not required by the SPH scheme in general.

As briefly mentioned before, SPH simply employs the kernel function W to perform a smoothing, thereby interpolating discrete samples, and does not necessarily have to be applied only to locations that coincide with particle positions, although finding the value of field quantities at a particle position is certainly desirable in a Lagrangian fluid simulation.

Further, note that since the gradient is a linear operator it can be pulled into the sum in Equation 3.10, resulting in[2]:

$$\nabla A_i \approx \sum_j A_j \frac{m_j}{\rho_j} \nabla W_{ij} \tag{3.12}$$

such that the gradient of a field can conveniently be computed simply by evaluating the function ∇W instead of W.

3.2 Kernel Functions and Properties

So far it has been left unspecified what form exactly the kernel function W takes, although some of its required properties were alluded to. Furthermore, W is often parameterized in its support radius $\hbar$ and smoothing length, which we will assume to be equal in the following, yielding $W(\vec{\boldsymbol{x}}_{ij}, \hbar)$, where $\vec{\boldsymbol{x}}_{ij} = \vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j$. Properties of this function shall be enumerated in the following[2]:

Normalization $\int_{\mathcal{V}} W(\vec{\boldsymbol{x}}_{ij}, \hbar) d\vec{\boldsymbol{x}}_j = 1$
is required for the approximation to be consistent.

Dirac-δ Condition $\lim_{\hbar \to 0} W(\vec{\boldsymbol{x}}_{ij}, \hbar) = \delta(\vec{\boldsymbol{x}}_{ij})$
is the motivation for the scheme in the first place and required for $A = (A * \delta) = (A * W)$ to hold in the limit.

Compact Support $\forall \, ||\vec{\boldsymbol{x}}_{ij}|| > \hbar : W(\vec{\boldsymbol{x}}_{ij}, \hbar) = 0$
reduces the SPH sum from $\mathcal{O}(n^2)$-complexity in n particles to potentially $\mathcal{O}(n)$

Sufficient Smoothness $W \in C^n, n \geq 2$
it is desirable for the first few derivatives of W to be continuous for discretizations such as in Equation 3.12 to be viable and for second order partial differential equations to be handled with ease[2]

Positivity $\forall \vec{\boldsymbol{x}}_{ij} : W(\vec{\boldsymbol{x}}_{ij}, \hbar) \geq 0$
while negative values of the kernel are permitted[12] and even desired in some cases such as when modelling surface tension[15], they are typically avoided since they might yield unphysical results at suboptimal sampling for quantities that should not be negative like mass, density, volume etc.

Symmetry $W(\vec{\boldsymbol{x}}_{ij}, \hbar) = W(\vec{\boldsymbol{x}}_{ji}, \hbar)$
is typically desired, even if just for lack of better assumptions about the structure of the interpolated field - indeed most kernels are spherically symmetric and only depend on the distance $||\vec{\boldsymbol{x}}_{ij}||$ between two points, which is especially valid when the interpolated field is assumed to be approximately isotropic on scales of the order of $\hbar$.

Note that the two required properties in this case, the normalization and δ-condition, correspond to the two approximations made in Equation 3.7 and Equation 3.9 of the derivation of SPH: the approximation is valid as the number of samples in the kernel support goes to infinity, making the discretization of the integral exact, and as the kernel support goes to zero, making the convolution with the kernel function an exact identity[12]. It is sometimes noted that SPH can not guarantee 0-th order consistency for arbitrary samplings, however 0-th and 1st order consistency are in fact achieved if the conditions[2]:

$$\sum_j \frac{m_j}{\rho_j} W_{ij} = 1 \qquad\qquad \sum_j \frac{m_j}{\rho_j} (\vec{\boldsymbol{x}}_j - \vec{\boldsymbol{x}}_i) W_{ij} = 1 \tag{3.13}$$

hold, which can be enforced if desired by a normalization and a matrix inversion respectively[16], although this is often not required for plausible results.

Further, note that if spherical symmetry is not enforced, a kernel may be constructed that linearly interpolates quantities along the Cartesian coordinate axes, compact to not just a sphere but a box within that sphere, and that this kernel may be evaluated on a regular grid, yielding the finite difference method. Whether SPH is therefore a generalization of grid-based methods or if the lack of the symmetry condition reduces the definition of 'SPH' to meaninglessness is a purely taxonomic question, but it underlines the expressivity of the SPH framework.

A very typical choice for a kernel function is one that is similar to a Gaussian distribution in shape but has compact support, as demanded above. There are a few intuitions as to why a Gaussian-like kernel is a very natural choice for this problem. From the perspective of signal theory, it is common[17] and natural to apply a Gaussian filter to a signal in order to smoothen it and reduce high-frequency noise, allowing for better interpolation. The Gaussian is special in the sense that it is one of the eigenfunctions of the Fourier transform, yielding a Gaussian again when transformed[18]. More specifically, an isotropic (spherically symmetric) Gaussian filter can be thought of as the optimal way to filter a signal in many regards:

- it does not overshoot when approximating step functions, being the unique function of a [19, useful class of functions] that does not create or change local extrema[17]
- it does not create new zero-crossings in the second derivative[17], which is crucial for fluid dynamics where zero-crossings of Laplacians are of interest (e.g. the pressure Poisson equation)
- it has optimal locality in space and frequency, minimizing the Heisenberg-Weyl inequality[17]. Intuitively, spatial locality has the benefit that the smoothed signal swiftly follows the original signal with minimal delay and high fidelity, while frequency locality means that the result is optimally smooth, since the Gaussian does not extend to higher frequencies and implements a stricter low-pass filter.

The last property is an interesting connection to the uncertainty principle, which is the same inequality but typically applied to momentum and locations in physics: the product of variances of a function in the spatial and frequency domains has a lower bound, which results in uncertainty when measuring in either domain, and the lowest bound is reached only by a Gaussian distribution.

Another perspective on the usefulness of the Gaussian and SPH in general is given by a probabilistic perspective on the problem. Interestingly, the original authors of SPH independently derive it from a stochastic point of view[12][13], both groups even referencing the same book on Monte Carlo techniques[20].

Suppose fluid samples representing equal masses are independently sampled from a distribution proportional to the mass density of a fluid of homogeneous density. It then holds that the mass density can be approximated by counting the number of samples within some volume $\mathcal{V}$ around a point of interest $\vec{\boldsymbol{x}}$ and normalizing the result[21]. It is equally valid to define certain compactly supported kernel functions to weigh the samples by and sum those weighted contributions instead of simply counting the samples in $\mathcal{V}$ with an indicator function, such that the weight functions potentially smooth out the approximated field[21]. The resulting integral[13] $\rho_{est}(\vec{\boldsymbol{x}}) = \int_{\vec{\boldsymbol{x}}_j \in \mathcal{V}} W(\vec{\boldsymbol{x}} - \vec{\boldsymbol{x}_j})\rho(\vec{\boldsymbol{x}}_j)\, d\vec{\boldsymbol{x}}_j$ can then be approximated by a Monte-Carlo estimator over discrete samples $\vec{\boldsymbol{x}}_j$ drawn from a distribution $\rho' \propto \rho$, yielding the SPH method[13]. Again, it is natural to model the kernel W to a Gaussian by invoking the central limit theorem to argue that the measured number of samples in the volume, and by proxy the estimated density, is distributed normally around the true density as this process is repeated.

A very commonly used kernel that mimics the Gaussian in shape but has compact support and is fast to evaluate by virtue of being a polynomial is the **CUBIC SPLINE KERNEL** or M_4 Schoenberg B-spline[16][22]:

$$W(\vec{\boldsymbol{x}}, \hbar) = \frac{\alpha}{4h^d} \begin{cases} (2-q)^3 - 4(1-q)^3 & 0 \leq q < 1 \\ (2-q)^3 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases} \tag{3.14}$$

$$\nabla W(\vec{\boldsymbol{x}}, \hbar) = \frac{\alpha}{4h^d} \frac{\vec{\boldsymbol{x}}}{||\vec{\boldsymbol{x}}||\, h} \begin{cases} -3(2-q)^2 + 12(1-q)^2 & 0 \leq q < 1 \\ -3(2-q)^2 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases} \tag{3.15}$$

where $q := \frac{||\vec{\boldsymbol{x}}||}{h}$ is the distance normalized to a particle spacing h, it holds that $\hbar = 2h$, d is the number of dimensions and α is a dimensionality-dependent constant which is $\frac{2}{3}$ for 1D, $\frac{10}{7\pi}$ for 2D and $\frac{1}{\pi}$ for 3D[16]. The kernel can also be written branchlessly (and implemented as such) as[22]:

$$W(\vec{\boldsymbol{x}}, \hbar) = \alpha \left[\max(0\,,\, 2-q)^3 - 4\max(0\,,\, 1-q)^3 \right] \tag{3.16}$$

$$\nabla W(\vec{\boldsymbol{x}}, \hbar) = \alpha \frac{\vec{\boldsymbol{x}}}{||\vec{\boldsymbol{x}}||\, h} \left[-3\max(0\,,\, 2-q)^2 + 12\max(0\,,\, 1-q)^2 \right] \tag{3.17}$$

This kernel function and its first derivative are illustrated in Figure 3.1.

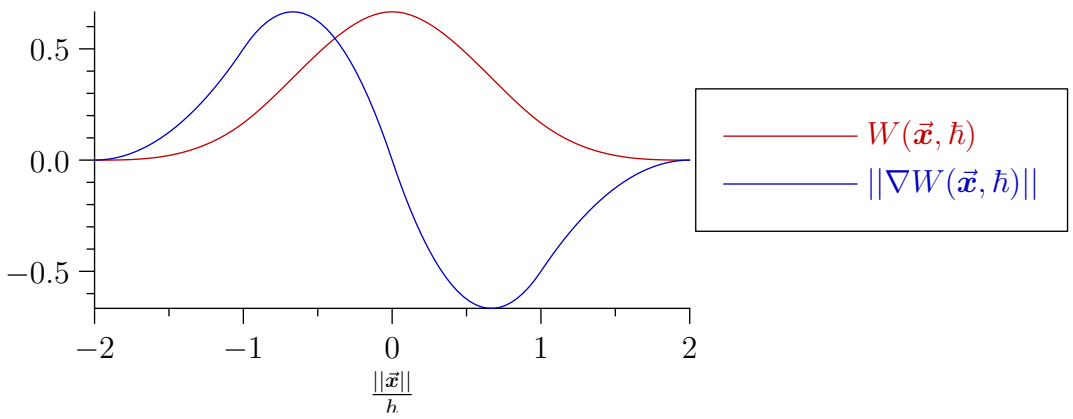


Figure 3.1: The kernel function in 1D and the magnitude of its derivative are shown with respect to the length of $\vec{x}$ normalized by the particle spacing h. It can be seen that the kernel support for this function is $\hbar = 2h$. W can also be written as parameterized by a scalar distance and is therefore spherically symmetric around the origin, while ∇W depends on the direction of $\vec{x}$ as seen in Equation 3.15 and is antisymmetric.

CHAPTER 4

SOLVING THE NAVIER-STOKES EQUATIONS

Having derived the Navier-Stokes equations as the governing equations of fluid flow in chapter 2 and a method for discretizing these equations in chapter 3, we can go about actually implementing a numerical solver for the equations. Two types of solvers in particular will be discussed here: a simple solver using the equation of state to simulate weakly compressible flow and an iterative solver that uses a nearly identical formulation but can strongly enforce incompressibility. The concept of operator splitting will be key to turning a weakly compressible SPH formulation into a scheme that can be iterated to approximately solve the pressure Poisson equation and yield incompressible flow - a task that appears rather non-trivial - using an implementation that is not much more complex than a regular state equation solver.

4.1 Equation of State SPH Solver

To construct a solver for the Navier-Stokes equation, we assume for now that an initial state and boundary conditions are given and focus on propagating the time-dependent solution into the future - details on boundaries and initialization will follow in chapter 5. Suffice it for now to know that in the approach chosen, boundaries are discretized in much the same way as the fluid and treated as tough they were fluid particles, only their positions are static.

In order to implement the kernel sum in Equation 3.10 and similar SPH sums, one needs to iterate over a set of samples denoted as subscript j. Since a kernel with compact support was chosen in Equation 3.14, all samples with non-zero contributions to the approximation of fields at $\vec{\boldsymbol{x}}_i$ lie within a radius of $\hbar$ around $\vec{\boldsymbol{x}}_i$, with $\hbar = 2h$ being a global constant since the fluid is chosen to be sampled at uniform resolution in this case. The sum $\sum_j$ therefore only has to iterate over the set of neighbouring fluid particles $\mathcal{N}_f(\vec{\boldsymbol{x}}_i) = \{\vec{\boldsymbol{x}}_j : ||\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j|| \leq \hbar\}$ and similarly for boundary particles that will be denoted with subscript k, making the computation of the sum an instance of a fixed-radius near neighbour problem[2].

A uniform grid with a cell side-length of $\hbar$ is chosen to compute this set in linear time, since only a constant number of cells (3^d in d dimensions) must be searched to find all possible neighbours of $\vec{\boldsymbol{x}}_i$. An implicit memory representation of such a grid is can be obtained by:

1. Computing a cell index per particle that uniquely identifies the cell containing the particle. Space-filling curves such as the Z-curve are popular for computing these indices since they rely on little prior information and yield good memory coherence[23].

2. Creating a list of handles which each store the particle's cell index together with its index in attribute buffers (positions, velocities, masses, etc.)

3. Sorting the list of handles with respect to the cell index. In this instance, a multithreaded, parallel radix sort is chosen for its linear runtime complexity on the discrete indices

Then, the particles in the desired cell can be looked up by performing a search for the first handle with the cell's index in the sorted array and including subsequent particles until the cell index no longer matches. Since the array is sorted, a binary search can be used to find particles in a given cell with logarithmic time complexity. For details and more optimized implementations, we refer to corresponding [24, Literature].

With the technicalities of computing sums over neighbours out of the way, the discrete versions of the governing equations can be formalized. The Navier-Stokes momentum equation as stated in Equation 2.44 reads as follows, annotated for a particle of interest i:

$$\underbrace{\frac{D\vec{\boldsymbol{v}}}{Dt}}_{\text{total acceleration } \vec{\boldsymbol{a}}_i} = \underbrace{-\frac{1}{\rho}\nabla p}_{\text{pressure acceleration } \vec{\boldsymbol{a}}_i^p} + \underbrace{\nu\nabla^2\vec{\boldsymbol{v}}}_{\text{viscous acceleration } \vec{\boldsymbol{a}}_i^{vis}} + \underbrace{\vec{\boldsymbol{b}}^{ext}}_{\text{external accelerations } \vec{\boldsymbol{a}}_i^{ext}} \tag{4.1}$$

- Firstly, the only **EXTERNAL BODY FORCE** $\vec{\boldsymbol{b}}_i^{ext}$ per unit mass acting on the fluid is gravity, which is equal to the gravitational acceleration $\vec{\boldsymbol{g}} \approx (0, -9.81)^T$, where a 2D setting, SI units and a y-axis facing up in the positive direction are assumed in the following.

- Secondly, the **VISCOUS ACCELERATION** may be discretized. For this, an SPH approximation of the Laplacian is required, but using the less smooth and more detailed second derivative of the kernel function directly to implement this can lead to inaccurate results when sampling quality happens to be suboptimal. Instead, operating on the kernel gradient in a manner similar to a finite difference, the following discretization can be derived[2]:

$$\nabla^2 \vec{\boldsymbol{v}} \approx 2(d+2) \sum_j \frac{m_j}{\rho_j} \frac{\vec{\boldsymbol{v}}_{ij} \cdot \vec{\boldsymbol{x}}_{ij}}{||\vec{\boldsymbol{x}}_{ij}||^2 + 0.01h^2} \nabla W_{ij} \tag{4.2}$$

  where $d = 2$ is the number of dimensions and a double subscript indicates a difference as in $A_{ij} = A_j - A_i$, while the small value $0.01h^2$ is in place purely for avoiding divisions by zero and divergences if particle positions coincide[16].

  From Equation 4.2 it is apparent that pairwise viscous accelerations are modelled to align with the axis spanned by the two positions of the pair and that they are symmetric, since all but the scalar quantities are projected onto $\vec{\boldsymbol{x}}_{ij}$ by virtue of the dot product and ∇W_{ij} being a scalar multiple of $\vec{\boldsymbol{x}}_{ij}$, which gives an intuition for why this formulation conserves momentum[2]. The masses m_j used in the equation are set when initializing the system and remain constant as previously mentioned, the current density ρ_j however should be calculated as outlined in Equation 3.11:

$$\rho_i \approx \sum_j m_j W_{ij} \tag{4.3}$$

- Lastly, the **PRESSURE ACCELERATION** must be discretized. For this, a symmetric formula that also conserves linear and angular momentum is chosen, since these properties are critical for robust simulations[2]:

$$-\nabla p \approx -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \tag{4.4}$$

  To compute the pressures p_i, p_j, we use the equation of state from Equation 2.45:

$$p_i = \max\left(0, k\left(\frac{\rho_i}{\rho_0} - 1\right)\right) \tag{4.5}$$

  for some uniform rest density ρ_0 of the fluid, a stiffness parameter k that will be discussed later and the calculated densities ρ_i.

Having discretized the right-hand side of the Navier-Stokes momentum equation and weakly enforced incompressibility by linking pressure accelerations to the density deviations they are correcting through the equation of state, a procedure for calculating accelerations at some point in time t is obtained. With this, Newton's second law can be solved for the updated position $\vec{\boldsymbol{x}}_i(t + \Delta t)$ of each particle, yielding an equation of motion that is discretized in time and solved using the symplectic Euler scheme as outlined in Equation 3.1:

$$\vec{\boldsymbol{v}}_i(t + \Delta t) = \vec{\boldsymbol{v}}_i(t) + \Delta t \vec{\boldsymbol{a}}_i(t) \tag{4.6}$$

$$\vec{\boldsymbol{x}}_i(t + \Delta t) = \vec{\boldsymbol{x}}_i(t) + \Delta t \vec{\boldsymbol{v}}_i(t + \Delta t) \tag{4.7}$$

The time step Δt must be chosen to ensure a temporal resolution fine enough to resolve processes that happen on a length scale of h, which motivates the **COURANT-FRIEDRICHS-LEWY** condition or *CFL condition* for short: a particle shall not move further than its radius h in one time step, or:

$$||\vec{\boldsymbol{x}}_i(t + \Delta t) - \vec{\boldsymbol{x}}_i(t)|| \leq \lambda h \tag{4.8}$$

where $0 < \lambda \leq 1$ describes the time step size relative to the maximum time step allowed by the condition. Since the same time step is used for all particles for simplicity, Δt must be estimated conservatively by approximating the distance the fastest particle might move in the current time step, based on the highest velocity observed in the previous time step. Further, including a maximum time step that avoids

divergences when the fastest velocity is very small, such as when the simulation is initialized with zero velocities, the formulation used in this report is:

$$\Delta t = \min\left(\Delta t_{max}\ ,\ \lambda \frac{h}{\max_i ||\vec{\boldsymbol{v}}_i||}\right) \tag{4.9}$$

This completes the simulation loop, allowing solutions to be propagated through time and solving the dynamics of the system. A summary of the algorithm is given in algorithm 1.

Algorithm 1 Equation of State SPH Fluid Solver *EOSSPH*

function EOSSPH($\langle \vec{\boldsymbol{x}}_i(t) \rangle , \langle \vec{\boldsymbol{v}}_i(t) \rangle , \langle m_i \rangle , \vec{\boldsymbol{g}}, \nu, k, \lambda, \rho_0$)

Step 1 – Fixed Radius Neighbour Search

$\mathcal{N}_f(\vec{\boldsymbol{x}}_i) \leftarrow \{\vec{\boldsymbol{x}}_j : ||\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j|| \leq \hbar\}$

Step 2 – Iteration: compute quantities with dependency on inputs

$\rho_i \leftarrow \sum_j m_j W_{ij}$ ▷ update density Equation 3.11

$a_i^{ext} \leftarrow \vec{\boldsymbol{g}}$ ▷ external body forces

Step 3 – Iteration: compute quantities with dependency on $\langle \rho_i \rangle$

$a_i^{vis} \leftarrow 2\nu(d+2)\sum_j \frac{m_j}{\rho_j} \frac{\vec{\boldsymbol{v}}_{ij} \cdot \vec{\boldsymbol{x}}_{ij}}{||\vec{\boldsymbol{x}}_{ij}||^2 + 0.01h^2} \nabla W_{ij}$ ▷ viscous acceleration Equation 4.2

$p_i \leftarrow \max\left(0, k\left(\frac{\rho_i}{\rho_0} - 1\right)\right)$ ▷ update pressure Equation 2.45

$a_i^p \leftarrow -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}$ ▷ pressure acceleration Equation 4.4

Step 4 – Numerical Time Integration

$\Delta t \leftarrow \min\left(\Delta t_{max}\ ,\ \lambda \frac{h}{\max_i ||\vec{\boldsymbol{v}}_i||}\right)$ ▷ CFL condition Equation 4.9

$\vec{\boldsymbol{v}}_i(t + \Delta t) \leftarrow \vec{\boldsymbol{v}}_i(t) + \Delta t \left(a_i^{ext} + a_i^{vis} + a_i^p\right)$ ▷ explicit velocity update Equation 3.1

$\vec{\boldsymbol{x}}_i(t + \Delta t) \leftarrow \vec{\boldsymbol{x}}_i(t) + \Delta t \vec{\boldsymbol{v}}_i(t + \Delta t)$ ▷ implicit position update Equation 3.1

return $\langle \vec{\boldsymbol{x}}_i(t + \Delta t) \rangle \langle \vec{\boldsymbol{v}}_i(t + \Delta t) \rangle$

end function

4.2 Operator Splitting

In order to move towards a solver that more strongly enforces incompressibility, the update to the velocity in algorithm 1 must be inspected more closely. A main focus of fluid solvers that simulate hydrodynamics is the pressure force, since viscosity is not typically dominant and pressure forces are critical to ensure incompressibility[2]. These pressure forces are functions of the current position of all particles, which yield some density and pressure - the more accurate the estimated positions are, the more accurately can the pressure force be calculated. Note how the update to the velocity in algorithm 1 integrates a sum of accelerations to a sum of velocities that are added to the current velocity, with one component of the sum being caused by pressure and two by non-pressure forces. One can equivalently compute these parts of the sum in sequence, as in:

$$\vec{\boldsymbol{v}}_i^*(t) \leftarrow \vec{\boldsymbol{v}}_i(t) + \Delta t \left(a_i^{ext} + a_i^{vis}\right) \tag{4.10}$$

$$\vec{\boldsymbol{v}}_i(t+\Delta t) \leftarrow \vec{\boldsymbol{v}}_i^*(t) + \Delta t a_i^p \tag{4.11}$$

So far, nothing is lost and nothing is gained through this transformation, but things change if the pressure can be formulated in direct functional dependence of the current velocity: by the time that pressure accelerations are computed, a better estimate for the velocity $\vec{\boldsymbol{v}}_i(t+\Delta t)$ is available in the form of $\vec{\boldsymbol{v}}_i^*(t)$, which incorporates how gravity and viscous forces will affect the particle movement in the current time step. In order to make use of this additional knowledge, positions could be updated to yield $\vec{\boldsymbol{x}}_i^*(t)$, a neighbour search could be executed and updated densities and pressure could be found as a result, however this can be inefficient, since the neighbour search already occupies a significant, if not dominant, part of the computation time of each simulation step.

Instead, an approximation can be applied at the current position that estimates a predicted density ρ_i^* using the updated velocities $\vec{\boldsymbol{v}}_i^*$ without actually advecting the particles yet. The velocities are used to approximate the time rate of change of density $\frac{D\rho_i}{Dt}$ due to the viscous and external forces, which can then be multiplied by the time step size Δt to integrate it with respect to time and obtain a change in density $\Delta\rho$[22]:

$$\rho_i^* = \rho_i + \Delta\rho \tag{4.12}$$

$$\approx \rho_i + \Delta t \frac{D}{Dt}\rho_i \tag{4.13}$$

$$\approx \sum_j m_j W_{ij} + \Delta t \sum_j m_j \vec{\boldsymbol{v}}_{ij}^* \cdot \nabla W_{ij} \tag{4.14}$$

The second term can be interpreted as a change in density caused by velocity divergence which is caused by the viscous and external accelerations - it can be derived by applying one possible SPH discretization of the divergence operator for a vector quantity $\vec{\boldsymbol{a}}$[23]:

$$\nabla \cdot \vec{\boldsymbol{a}}_i = -\frac{1}{\rho}\sum_j m_j \vec{\boldsymbol{a}}_{ij} \cdot \nabla W_{ij} \tag{4.15}$$

to the $\frac{D\rho}{Dt}$ term in the continuity equation as seen in 2.9:

$$\frac{D\rho_i}{Dt} = -\rho_i(\nabla \cdot \vec{\boldsymbol{v}}_i) \qquad \textit{Continuity, Equation 2.9} \tag{4.16}$$

$$\approx -\rho_i \left(-\frac{1}{\rho_i}\sum_j m_j \vec{\boldsymbol{v}}_{ij} \cdot \nabla W_{ij}\right) \qquad \textit{SPH divergence, Equation 4.15} \tag{4.17}$$

$$= \sum_j m_j \vec{\boldsymbol{v}}_{ij} \cdot \nabla W_{ij} \qquad \textit{simplify} \tag{4.18}$$

With this approximation, the updated velocity $\vec{\boldsymbol{v}}^*$ can be used, resulting in a technique referred to as **OPERATOR SPLITTING**[2]. The original partial differential equations are split up into a sequence of subproblems: one that solves for accelerations caused by non-pressure forces and second one that uses the result of the first problem to solve for pressure accelerations which attempt to enforce incompressibility[2]. The first problem updates velocities to $\vec{\boldsymbol{v}}_i^*$ in an explicit manner, while the second problem performs a somewhat 'implicit' update using the updated $\vec{\boldsymbol{v}}^*$, in the hopes of improving stability[2]. This may be thought of as akin to the semi-implicit Euler update for time integration, in which an explicit velocity

update allows for an implicit update to positions, in hopes of improving stability and accuracy. Stiffer sub-problems are computed at a later point in the sequence, such that they can make use of better approximations[2].

For an incompressible fluid, the time rate of change of density in a Lagrangian frame of reference ought to be zero in accordance with Equation 2.10. This means that in order to enforce incompressibility, pressures should be computed such that $\frac{D\rho}{Dt} = 0$ and $\rho_i = \rho_0$ or in other words the predicted density ρ_i^* should be ρ_0. This insight will become the connection from operator splitting to an SPH formulation that solves a Pressure Poisson Equation in section 4.3.

An updated algorithm that includes operator splitting is shown in algorithm 2. What changes in comparison to algorithm 1 is the density estimation before pressure accelerations are calculated and the fact that the time step size must be fixed at that point. The numerical time integration that was previously an appendix to the actual computation is now a relevant part of the density estimation, since the predicted density depends on the method with which the predicted velocity $\vec{\boldsymbol{v}}^*$ is integrated, and since the integration method across now split-up summands of the total acceleration is to be consistent.

Algorithm 2 Equation of State SPH Fluid Solver with Operator Splitting *SplitSPH*

function SPLITSPH($\langle \vec{\boldsymbol{x}}_i(t) \rangle, \langle \vec{\boldsymbol{v}}_i(t) \rangle, \langle m_i \rangle, \vec{\boldsymbol{g}}, \nu, k, \lambda, \rho_0$)

Step 1 – Fixed Radius Neighbour Search

$\mathcal{N}_f(\vec{\boldsymbol{x}}_i) \leftarrow \{\vec{\boldsymbol{x}}_j : ||\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j|| \leq \hbar\}$

Step 2 – Compute density ρ

$\rho_i \leftarrow \sum_j m_j W_{ij}$ ▷ update density Equation 3.11

Step 3 – Compute non-pressure accelerations

$a_i^{ext} \leftarrow \vec{\boldsymbol{g}}$ ▷ external body forces

$a_i^{vis} \leftarrow 2\nu(d+2)\sum_j \frac{m_j}{\rho_j} \frac{\vec{\boldsymbol{v}}_{ij} \cdot \vec{\boldsymbol{x}}_{ij}}{||\vec{\boldsymbol{x}}_{ij}||^2 + 0.01h^2} \nabla W_{ij}$ ▷ viscous acceleration Equation 4.2

Step 4 – Predict densities ρ_i^* that take non-pressure accelerations into account

$\Delta t \leftarrow \min\left(\Delta t_{max}, \lambda \frac{h}{\max_i ||\vec{\boldsymbol{v}}_i||}\right)$ ▷ CFL condition Equation 4.9

$\vec{\boldsymbol{v}}_i^* \leftarrow \vec{\boldsymbol{v}}_i(t) + \Delta t \left(a_i^{ext} + a_i^{vis}\right)$ ▷ estimated velocity Equation 3.1

$\rho_i^* \leftarrow \sum_j m_j W_{ij} + \Delta t \sum_j m_j \vec{\boldsymbol{v}}_{ij}^* \cdot \nabla W_{ij}$ ▷ predicted density Equation 4.12

Step 5 – Compute pressure accelerations

$p_i \leftarrow \max\left(0, k\left(\frac{\rho_i^*}{\rho_0} - 1\right)\right)$ ▷ update pressureEquation 2.45

$a_i^p \leftarrow -\sum_j m_j \left(\frac{p_i}{(\rho_i^*)^2} + \frac{p_j}{(\rho_j^*)^2}\right) \nabla W_{ij}$ ▷ pressure acceleration Equation 4.4

Step 6 – Numerical Time Integration

$\vec{\boldsymbol{v}}_i(t + \Delta t) \leftarrow \vec{\boldsymbol{v}}_i(t) + \Delta t \left(a_i^{ext} + a_i^{vis} + a_i^p\right)$ ▷ explicit velocity update Equation 3.1

$\vec{\boldsymbol{x}}_i(t + \Delta t) \leftarrow \vec{\boldsymbol{x}}_i(t) + \Delta t \vec{\boldsymbol{v}}_i(t + \Delta t)$ ▷ implicit position update Equation 3.1

return $\langle \vec{\boldsymbol{x}}_i(t + \Delta t) \rangle \langle \vec{\boldsymbol{v}}_i(t + \Delta t) \rangle$

end function

4.3 Incompressible SPH Solver

Having derived a solver that uses both the Equation of State and operator splitting, the step towards a solver that enforces incompressibility is surprisingly simple: instead of predicting densities and computing pressures accelerations once, the same operations can be iterated and $\vec{\boldsymbol{v}}^*$ refined until the predicted density is the rest density and the fluid is under as little compression as desired. Each iteration of computing pressure accelerations that minimize the density error can be interpreted as an attempt at projection of the predicted velocity field onto a divergence-free state[23]. This projection is actualized if the **pressure Poisson equation** is solved, which, depending on whether density invariance or zero-divergence of velocity is chosen as a source term, can read[23]:

$$\nabla^2 p_i^2 = \frac{1}{\Delta t}\rho_0 \nabla \cdot \vec{\boldsymbol{v}}_i^* \qquad \textit{divergence-free source term} \tag{4.19}$$

$$\nabla^2 p_i^2 = \frac{1}{\Delta t^2}\left(\rho_0 - \rho_i^*\right) \qquad \textit{density invariant source term} \tag{4.20}$$

Either way, this can be read as a system of N equations in N unknown pressure values that realizes a Poisson equation $\nabla^2 a = s$ for some unknown quantity a and known source term s, since $\rho_i^*, \rho_0, \vec{\boldsymbol{v}}_i^*$ are known at this point. Instead of simply evaluating an equation of state once to explicitly obtain pressures, pressures are now chosen such that the pressure accelerations resulting from them cause the fluid to remain in an uncompressed or divergence-free state.

Many methods to solve for these pressures and achieve incompressible SPH exist, varying in aspects such as:

- Whether they use a solver like relaxed Jacobi or Conjugate Gradients to compute solutions to the global system of equations, as is done in implicit incompressible SPH *(IISPH)*[2325], or if they iteratively solve for pressures using a specialized equation of state, such as in local Poisson SPH *(LPSPH)*[26] or predictive-corrective SPH *(PCISPH)*[11]
- In what variable they accumulate the changes computed in each iteration. PCISPH[11] and IISPH[25] accumulate pressures, while LPSPH refines predicted velocities and positions[26]

In any case, these methods typically employ an optimized scheme to compute a relation between density error and pressure that is hoped to lead to faster convergence and therefore fewer solver iterations, instead of relying on a hand-tuned and user-defined parameter k. Instead, only the convergence criterion is specified by the user, often in terms of a maximum predicted average density error η_{avg}[112625].

While it can be shown that PCISPH and IISPH are 'essentially equal'[2], suggesting they belong to a similar class of solvers, they might be differentiated by the fact that PCISPH computes a single, global stiffness constant k that is motivated by the geometric setting of a template particle, while IISPH computes an optimized stiffness k at each particle individually (which can be used to implement a relaxed Jacobi iteration etc.), such that faster convergence may be achieved[2].

In this report, the simplest option for an iterative solver is chosen and a single, global stiffness constant k is left open as a parameter and analysed in chapter 6. We choose to accumulate changes between iterations in the predicted velocities $\vec{\boldsymbol{v}}_i^*$, reusing the greatest part of algorithm 2 and following [22, this description], which loosely resembles the LPSPH[26], except it does not perform a neighbour search in every iteration and uses a different equation of state to compute pressures, namely Equation 2.45. This results in algorithm 3, which implements a density invariant source term but approximates the predicted density ρ^* within it using the divergence of the estimated velocity $\vec{\boldsymbol{v}}_i^*$.

Algorithm 3 Iterative SPH Fluid Solver using Operator Splitting *IterSPH*

function IterSPH($\langle \vec{\boldsymbol{x}}_i(t) \rangle, \langle \vec{\boldsymbol{v}}_i(t) \rangle, \langle m_i \rangle, \vec{\boldsymbol{g}}, \nu, k, \lambda, \rho_0, \eta$)

Step 1 – Fixed Radius Neighbour Search

$\mathcal{N}_f(\vec{\boldsymbol{x}}_i) \leftarrow \{\vec{\boldsymbol{x}}_j : ||\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j|| \leq \hbar\}$

Step 2 – Compute density ρ

$\rho_i \leftarrow \sum_j m_j W_{ij}$ ▷ update density Equation 3.11

Step 3 – Compute non-pressure accelerations and determine time step

$a_i^{ext} \leftarrow \vec{\boldsymbol{g}}$ ▷ external body forces

$a_i^{vis} \leftarrow 2\nu(d+2)\sum_j \frac{m_j}{\rho_j} \frac{\vec{\boldsymbol{v}}_{ij} \cdot \vec{\boldsymbol{x}}_{ij}}{||\vec{\boldsymbol{x}}_{ij}||^2 + 0.01h^2} \nabla W_{ij}$ ▷ viscous acceleration Equation 4.2

$\Delta t \leftarrow \min\left(\Delta t_{max}\,,\, \lambda \frac{h}{\max_i ||\vec{\boldsymbol{v}}_i||}\right)$ ▷ CFL condition Equation 4.9

Step 4 – Iteratively refine $\vec{\boldsymbol{v}}_i^*$ based on ρ_i^* after applying pressure forces

$\vec{\boldsymbol{a}}_i^* \leftarrow \vec{\boldsymbol{a}}_i^{ext} + \vec{\boldsymbol{a}}_i^{vis}$

$\vec{\boldsymbol{v}}_i^* \leftarrow \vec{\boldsymbol{v}}_i(t)$

$\rho_{avg}^{err} \leftarrow 0$

$l \leftarrow 0$

while $l < 300 \wedge \left((l < 5) \vee \left(\rho_{avg}^{err} \geq \eta_{avg}\right)\right)$ **do**

  $\vec{\boldsymbol{v}}_i^* \leftarrow \vec{\boldsymbol{v}}_i^*(t) + \Delta t \vec{\boldsymbol{a}}_i^*$ ▷ refine estimated velocity Equation 3.1

  $\rho_i^* \leftarrow \sum_j m_j W_{ij} + \Delta t \sum_j m_j \vec{\boldsymbol{v}}_{ij}^* \cdot \nabla W_{ij}$ ▷ predicted density Equation 4.12

  $p_i \leftarrow \max\left(0, k\left(\frac{\rho_i^*}{\rho_0} - 1\right)\right)$ ▷ update pressure Equation 2.45

  $\vec{\boldsymbol{a}}_i^* \leftarrow -\sum_j m_j \left(\frac{p_i}{(\rho_i^*)^2} + \frac{p_j}{(\rho_j^*)^2}\right) \nabla W_{ij}$ ▷ pressure acceleration Equation 4.4

  $\rho_{avg}^{err} \leftarrow \max\left[0\,,\, \left(\frac{1}{N}\sum_{i=1}^{N} \rho_i^*\right) - \rho_0\right]$ ▷ compute *estimated* (!) density error

  $l \leftarrow l + 1$

end while

Step 5 – Numerical Time Integration

$\vec{\boldsymbol{v}}_i(t + \Delta t) \leftarrow \vec{\boldsymbol{v}}_i^* + \Delta t \vec{\boldsymbol{a}}_i^*$ ▷ explicit velocity update Equation 3.1

$\vec{\boldsymbol{x}}_i(t + \Delta t) \leftarrow \vec{\boldsymbol{x}}_i(t) + \Delta t \vec{\boldsymbol{v}}_i(t + \Delta t)$ ▷ implicit position update Equation 3.1

return $\langle \vec{\boldsymbol{x}}_i(t + \Delta t) \rangle \langle \vec{\boldsymbol{v}}_i(t + \Delta t) \rangle$

end function

Chapter 5

Boundary and Initial Conditions

The governing equations of fluid flow, their discretization and the forwards propagation in time of a solution have been discussed so far, but two more crucial elements that determine the problem have been neglected: initial conditions and boundary conditions of the system.

Without an initial condition to go on, most problems, especially if they don't solve for a steady-state solution, are undetermined. The solvers in chapter 4 calculate positions and velocities at a future time $t + \Delta t$ from the positions and velocities at the current time t - seen backwards, this recursion must have an origin at some t_0 where positions and velocities are known. Constructing these positions and velocities for a given scenario that we wish to compute the dynamics of involves the act of discretizing a continuous field and might be thought of as trivial at a glance but actually has some nuances that determine the quality and stability in the first few seconds of the simulation - a timespan that can be of significant importance.

Boundary conditions are equally essential to the description of many problems, especially real-world problems that are full of complex boundary conditions and -geometry. Simulations that are actually boundless or where boundary conditions do not matter are typically not the most interesting. In some way or another, slip, no-slip, periodic or otherwise, the limits of the simulated domain must be enforced, desirably in the most robust and elegant way possible.

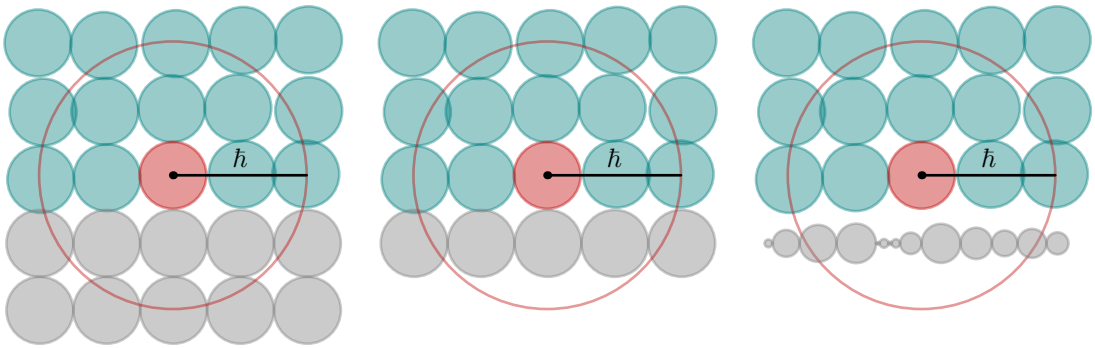
Both of these factors will be discussed in the following. In particular, a single layer of non-uniformly sampled particles will be used to represent boundaries, giving great flexibility while fitting seamlessly in the solvers developed in chapter 4. Jittering of initial positions as a way of reducing aliasing and the stability of the initial lattice in which continua are discretized will be discussed. Finally, an iterative method for enforcing a uniform initial density field will be shown.

5.1 Non-Uniform Single Layer Boundaries

While a great many boundary handling methods for SPH exist, a very common idea is to use the same type of discretization for boundaries as for fluids and represent them as particles, even going so far as to reuse the pressure solver to compute contact forces[27]. Great effort was taken to construct a solver that handles incompressibility well, so the same solver can be reused to implement boundaries if a boundary is simply imagined as being a fluid volume the positions of which remain static. This approach is referred to as **frozen particles**, as opposed to, for example, 'ghost particles' that are computed on the fly[27] or approaches based on signed distance fields[28]. With this particle-based representation, quantities at the boundary can be approximated using SPH just as before and pressure forces can be used to resolve contacts.

Multiple layers of boundary particles are conceptually required to fill the neighbourhood of a particle at the boundary and prevent the particle deficiency problem that also plagues free surfaces, where approximation quality deteriorates. This setting is illustrated in Figure 5.1a.

While this is not a problem for flat, thick boundary geometries, where the initial particle spacing h of the fluid can simply be used to regularly sample the boundary to a depth of at least the kernel support radius $\hbar$, things become less trivial when the boundary is particularly thin, curved, represents a non-manifold surface or some other inconvenient or complex geometry. To handle this, a more flexible approach is required. To derive this approach, first consider the case where the boundary is still sampled uniformly with the particle spacing h, but only a single layer of the boundary is present, as shown in Figure 5.1b. In this case, there is a particle deficiency resulting in incomplete SPH sums: the density, for example, that may be computed by iterating over fluid neighbours j and boundary neighbours k, is lacking an additional sum over missing fluid neighbours m, which is typically compensated for by linearly



(a) multiple, uniform boundary layers (b) single, uniform boundary (c) single, non-uniform boundary

Figure 5.1: A fluid resting on different particle representations of a flat, thick boundary are shown. A fluid particle at the boundary for which forces may be computed is shown in red with its kernel support radius $\hbar = 2h$ visualized as a circle. Fluid neighbours are shown in blue, while boundary particles are shown in grey. The area of the disks drawn is proportional to the volume $V_i = \frac{m_i}{\rho_i}$ of each particle. Note that the particles do not actually represent a sphere so much as a volume of unspecified shape surrounding a point. It can be observed that in Figure 5.1b particles are missing from the kernel support, while in Figure 5.1c additionally the resolution of the boundary sampling is non-uniform, which is compensated for by coefficients γ_1, γ_2.

scaling the contribution of the sum over boundary neighbours by a coefficient γ_1[2]:

$$\rho_i = \sum_j m_j W_{ij} + \sum_k m_k W_{ik} + \sum_m m_m W_{im} \tag{5.1}$$

$$\approx \sum_j m_j W_{ij} + \gamma_1 \sum_k m_k W_{ik} \tag{5.2}$$

where $m_k = m_m = m_j$ since the boundary sampling is uniform, and the volume of boundary samples is therefore equal to the volume of the fluid particles, which relative to the same reference rest density ρ_0 can also be expressed as an equal mass.

The coefficient γ_1 can be computed for a template particle with perfect sampling on a regular grid of grid size h by using the fact that for such optimal sampling, according to Equation 3.13 the kernel sum over all neighbours, including missing boundary samples, is normalized and the condition:

$$\sum_j V_j W_{ij} + \sum_k V_k W_{ik} + \sum_m V_m W_{im} = 1 \tag{5.3}$$

should hold, which means that for $V_j = V_k = V_m$[2]:

$$\sum_j W_{ij} + \gamma_1 \sum_k W_{ik} = \frac{1}{V_i} \tag{5.4}$$

$$\implies \quad \gamma_1 = \frac{\frac{1}{V_i} - \sum_j W_{ij}}{\sum_k W_{ik}} \tag{5.5}$$

This factor generally depends on the kernel support, dimensionality and kernel function used[2], but turns out to be $\gamma_1 \approx 1$ for $\hbar = 2h$ in this instance - since the kernel support ends where the second boundary layer starts for a perfect sampling, no compensation for missing samples is required in this case.

Now, the condition on the regular sampling of the single layer must be relaxed, since it is difficult to uphold in practice for previously mentioned complex geometries. Instead of assuming that $V_k = V_m = V_j$, the volume of each boundary particle is calculated and translated via the relation $V_i = \frac{m_i}{\rho_i}$ into a virtual 'mass' relative to the rest density ρ_0 of the fluid. Note that this 'mass' is not the actual mass of the boundary particle, as this would, for example, depend on the density of the boundary material in the general case of rigid-fluid coupling - it is just a re-encoding of the volume with respect to the fluid's rest density that makes it easier to apply the previously derived pressure solver to boundary handling. Using the same relation between SPH kernel sums and volumina of particles as above, the measured volume of a boundary particle denoted k' and the corresponding 'mass' relative to ρ_0 can be calculated as a sum over other boundary particles[2]:

$$V_{k'} = \frac{\gamma_1}{\sum_k W_{k'k}} \qquad\qquad m_{k'} = \rho_0 \frac{\gamma_1}{\sum_k W_{k'k}} \tag{5.6}$$

The final expression for the density of a fluid particle then becomes[27]:

$$\rho_i = \sum_j m_j W_{ij} + \sum_k m_k W_{ik} \tag{5.7}$$

With this, boundaries can not only be sampled in a more flexible way, but also more densely, as shown in Figure 5.1c. Since only the SPH sums of particles at the boundary have linearly more terms as the boundary resolution increases, and since sums from boundary particles to other boundary particles are only calculated when initializing the masses m_k as in Equation 5.6, the runtime cost for this oversampling is low while the benefit in terms of reducing the error in the direction of the pressure forces, for example, can be substantial. In this implementation, the boundary is sampled with $2.0106\ldots$ times the resolution of the fluid, where the integer multiple 2 is avoided to prevent aliasing artefacts.

With the ability to compute accurate densities of fluid particles even at boundaries, all that remains is to extend the SPH discretization of the pressure acceleration to include boundary neighbours. Recalling Equation 4.4, densities and pressures at the boundary sample are also required for this computation. A few approaches exist to estimate these quantities, including the mirroring of density and pressure from a particle i to a boundary neighbour k as in $\rho_i = \rho_k, p_i = p_k$, which may however assign inconsistent values to the same boundary particle depending on which fluid particle is referenced as i. In this implementation, we choose to mirror pressure values to boundary particles $p_k = p_i$, while assuming that boundary particles have the rest density ρ_0 of the fluid and introduce a factor γ_2 to compensate for missing samples in the sum over kernel gradients in much the same way as γ_1 was motivated, making the final pressure acceleration of a fluid particle:

$$\vec{\boldsymbol{a}}_i^p = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij} - \gamma_2 \sum_k m_k \left(\frac{p_i}{\rho_i^2} + \frac{p_i}{\rho_0^2}\right) \nabla W_{ij} \tag{5.8}$$

Where for an ideal sampling as shown in Figure 5.1a the factor γ_2 can be computed as[2]:

$$\gamma_2 = \frac{\left(\sum_j -\nabla W_{ij}\right) \cdot \left(\sum_k \nabla W_{ik}\right)}{\left|\left|\sum_k \nabla W_{ik}\right|\right|^2} \tag{5.9}$$

and is set to one in this implementation by correcting the kernel derivative by a constant factor, which is about $0.987\ldots$ for the 2D Cubic Spline kernel with $\hbar = 2h$. If, for example, improved stability is observed, the parameter γ_2 can reasonably be set to differing values, where $\gamma_2 = \frac{1}{2}$ may be chosen as a lower bound, since it is equivalent to setting the pressure at the boundary to zero.

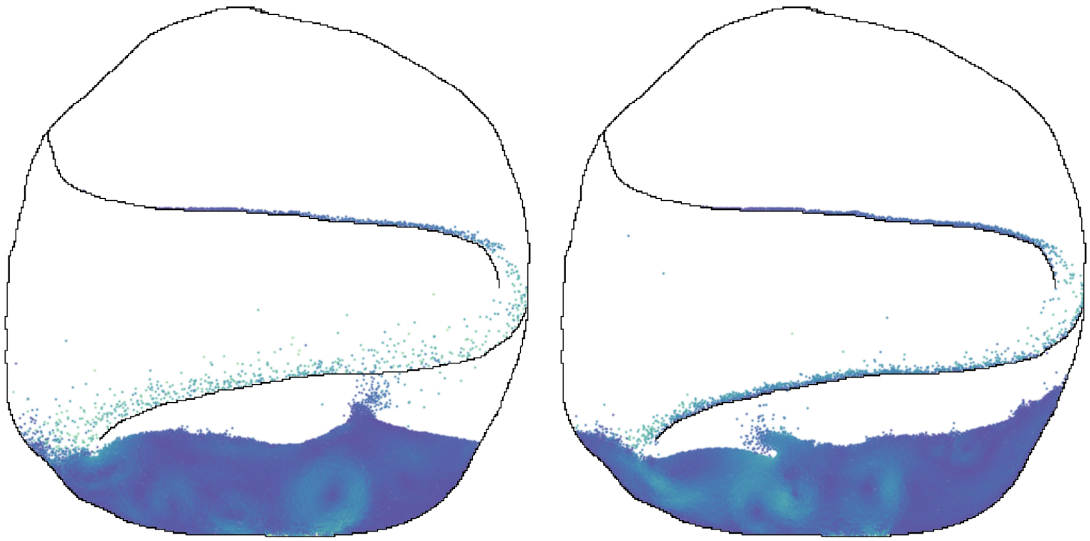
While the boundary was modelled such that the pressure solver can be used to resolve contact forces and boundary particles are handled in the same way as fluid particles, extending all previous sums over neighbours to both fluid and boundary neighbours, one exception is implemented: the SPH approximation of the viscous acceleration is split up into a viscosity caused by fluid neighbours and one caused by boundary neighbours, where the boundary is treated differently: in order to model adhesion to the boundary, the viscous force is multiplied by a separate coefficient ν_2 and projected onto the approximate boundary normal $\hat{\vec{\boldsymbol{n}}}_i$. The boundary normal can be approximated using an SPH sum over kernel gradients, sice they point straight away from the respective boundary particle:

$$\hat{\vec{\boldsymbol{n}}}_i = \begin{cases} \frac{\sum_k \nabla W_{ik}}{\left|\left|\sum_k \nabla W_{ik}\right|\right|} & \left|\left|\sum_k \nabla W_{ik}\right|\right| > 0 \\ \vec{\boldsymbol{0}} & \text{otherwise} \end{cases} \tag{5.10}$$

This results in an additional adhesion term that reads:

$$\vec{\boldsymbol{a}}^{adh} = \hat{\vec{\boldsymbol{n}}}_i \left[2\nu_2(d+2) \left(\sum_k \frac{m_k}{\rho_0} \frac{\vec{\boldsymbol{v}}_{ik} \cdot \vec{\boldsymbol{x}}_{ik}}{\left|\left|\vec{\boldsymbol{x}}_{ik}\right|\right|^2 + 0.01h^2} \nabla W_{ik} \right) \cdot \hat{\vec{\boldsymbol{n}}}_i \right] \tag{5.11}$$

which is always a multiple of the normal to the boundary if such a boundary is in the kernel support radius, therefore affecting only fluid particles at boundaries and only in the normal component of the movement. This leads not only to more stable behaviour but also dampens the impact of splashes that would otherwise bounce off the surface unrealistically in a manner similar to a perfectly elastic impact due to conservation of momentum. Note that for static boundary samples, $\vec{\boldsymbol{v}}_{ik} = \vec{\boldsymbol{v}}_i - \vec{\boldsymbol{0}} = \vec{\boldsymbol{v}}_i$. The impact of this adhesion term is shown in Figure 5.2.



(a) $\nu_2 = 0$ (b) $\nu_2 = 0.01$

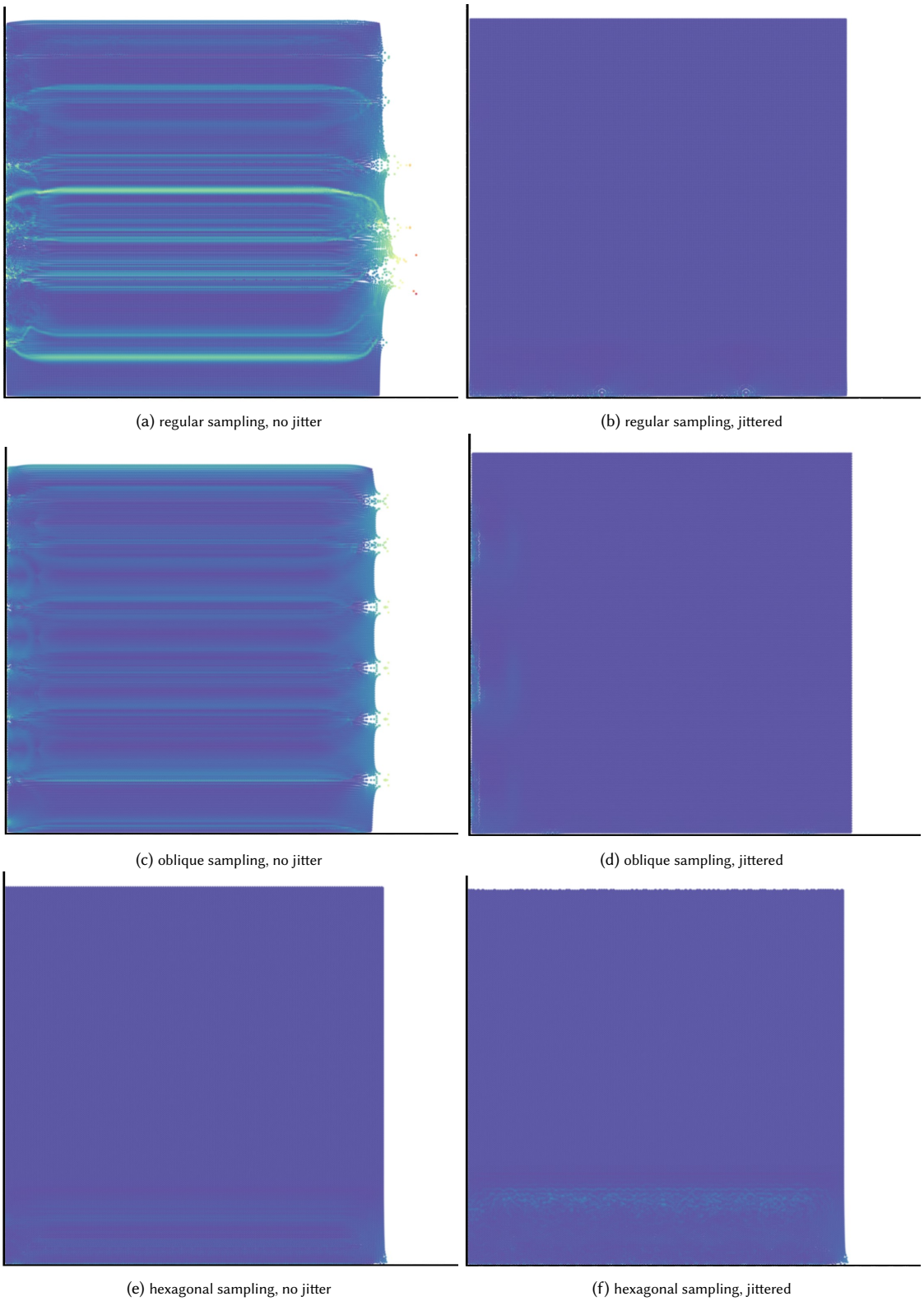
$N = 20952, h = 0.015m, \lambda = 0.1, k = 1250, \nu = 10^{-4}, \gamma_1 = \gamma_2 = 1$, SplitSPH Solver

Figure 5.2: Simulation frames showcasing complex boundary conditions and the impact of the adhesion term. While for $\nu_2 = 0$ there is unrealistic 'bouncing' of particles on the lower ramp leading to excessive spray, for $\nu_2 = 0.01$ the normal component of the fluid-boundary interaction is heavily damped, resulting in a more plausible image. The boundary itself was discretized from a hand-drawn, low resolution image to showcase the versatility of the boundary discretization into particles of arbitrary sampling, allowing for non-manifold and curved lines.

5.2 Jittered Initialization and Lattices

When initializing the simulation domain and discretizing the fields that describe the fluid into particles, it is common to simply use a regular grid with a spacing of h to sample particles. While this sampling is perfectly regular in space, which is desirable to ensure an accurate SPH approximation of the fields it discretizes, it is quite singular in the Fourier-transformed space, since the sampling along every coordinate axis is conducted at a single, constant frequency. This may lead to aliasing artefacts that can be observed in the initial stages of the simulation, where density errors and erroneous velocities are aligned with coordinate axes, although the behaviour of the fluid should ideally be isotropic, meaning it imposes no preferred directions on the behaviour of the fluid[29]. The problem appears to be often neglected in descriptions of SPH fluid solvers, as it seems to be less relevant for more elaborate, incompressible solvers and can be circumvented by discarding initial simulation frames. This circumvention is not always an option, however, warranting a closer look at the initialization.

In this implementation, a pseudo-random jitter of the initial positions is proposed, introducing noise to the initial sampling in order to smooth out the sampling in reciprocal space. Since a seeded pseudo-random number generator can be used to achieve this, the reproducibility of each simulation is kept intact. Offsets $\vec{x}_\Delta$ are drawn from a standard normal distribution $\vec{x}_\Delta \sim \mathcal{N}(0, 1)$ and scaled by a factor on the scale of $0.01h$ to obtain the initial sampling. A trade-off to consider when choosing the amplitude of the introduced noise is the effectiveness in preventing aliasing effects weighed against the regularity of the sampling: too little noise does not prevent aliasing, while too much noise can reduce the initial interpolation accuracy of the SPH approximation of fields unpredictably, especially for low kernel support radii that have fewer neighbours to rely on. Note that the choice of initial particle spacing can already mitigate much of the aliasing observed in Figure 5.3, but a jitter is still effective in the sense that with it, no restrictions are imposed on the initial particle spacing, which is desirable. In this sense, jitter only makes the initial conditions more robust to adverse parameter choices, preventing worst-case aliasing as is observed in Figure 5.3.



(a) regular sampling, no jitter

(b) regular sampling, jittered

(c) oblique sampling, no jitter

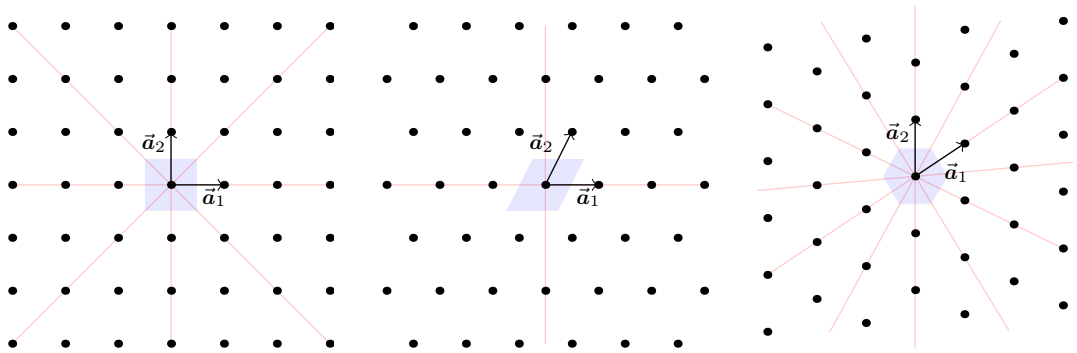
(d) oblique sampling, jittered

(e) hexagonal sampling, no jitter

(f) hexagonal sampling, jittered

$N \approx 88800, h = 0.01m, \lambda = 0.1, k = 1000, \nu = 10^{-3}, \gamma_1 = \gamma_2 = 1,$ SplitSPH Solver

Figure 5.3: Initial simulation frames at $t = 0.02s$ from a dam break scenario are compared for different initial particle samplings, where velocities are colour-coded. While the regular sampling experiences large erroneous velocities only in the vertical direction, an oblique sampling appears to result in a smaller and more evenly spread out errors, but still experiences aliasing. Using a hexagonal lattice or adding just a $\sigma^2 = (0.01h)^2$ pseudo-random jitter to the initialization very effectively reduces these artefacts. All simulations start at rest density as described in section 5.3



(a) square lattice (b) oblique lattice (c) hexagonal lattice

Figure 5.4: A square, oblique and hexagonal lattice in 2D are shown, including their respective mirror lines in red and the $A = h^2$ Voronoi cell surrounding the central lattice point in light blue.

There is another aspect that can be considered in improving the quality of the initialization: instead of a regular grid, another type of lattice can be used to discretize the fluid. In fact, comparing a regular lattice to, for example, a hexagonal close-packed lattice, it is well known that the former leads to anisotropy while the latter, being a close packing of spheres as well as a regular lattice, is more optimal in many regards, including stability against random permutation[29]. Three types of lattices $\vec{\boldsymbol{x}} = j \cdot \vec{\boldsymbol{a}}_1 + k \cdot \vec{\boldsymbol{a}}_2$ for $j, k \in \mathbb{Z}$ with the same unit cell volume of h^2 (which is crucial for a correct SPH approximation without re-normalizing the kernel function) were examined in this implementation, were h is the particle spacing of the regular grid:

Square Lattice (Regular Grid) $\vec{\boldsymbol{a}}_1 = (h, 0)^T, \vec{\boldsymbol{a}}_2 = (0, h)^T$
Appears to be particularly vulnerable to aliasing, which can be improved using a jitter as described above.

Oblique Lattice $\vec{\boldsymbol{a}}_1 = (h, 0)^T, \vec{\boldsymbol{a}}_2 = (h/2, h)^T$
Is trivial to implement by alternately adding a $\pm\frac{1}{4}h$ offset in the x-direction to each row in a regular grid. The shape seems closer to the hexagonal pattern that naturally arises, as seen in Figure 5.5.

Hexagonal Lattice $\vec{\boldsymbol{a}}_1 = \left(\frac{3}{2}a, a\right)^T, \vec{\boldsymbol{a}}_2 = \left(0, a\sqrt{3}\right)^T$ with $a = \sqrt{\frac{2h^2}{3\sqrt{3}}}$
Each hexagonal cell can be thought of as consisting of six equilateral triangles that meet at the centre, each with a side length of a. The calculation of the side length a stems from the fact that the area of each hexagonal tile should be h^2 in 2D, meaning $h^2 = \frac{3\sqrt{3}}{2}a^2$, which accounts for the hexagonal packing being denser and increases the spacing accordingly until each cell has the expected volume.

These two-dimensional lattices are illustrated in Figure 5.4.

With this, all crystal systems in two dimensions except for the rectangular lattice are covered. Comparisons of initial frames from a simulation with different lattices and jittered or regular initialization are shown in Figure 5.3. It seems that a hexagonal lattice is particularly stable even without jitter, while the other lattices can be made more stable with the introduction of even the smallest jitter, despite an unfavourable choice of initial sampling resolution. Varying the initial lattice and using more hexagonal shapes was motivated by the observation that particles naturally tend towards forming chunks of hexagonal lattices as they rest, such as in a hydrostatic case, when viscosity is high or the flow is very slow and laminar, as seen in Figure 5.5 - however one may suspect that the same behaviour might not arise quite as frequently and naturally in higher dimensions, since the hexagonal lattice is not the unique closest packed lattice in three dimensions for example.

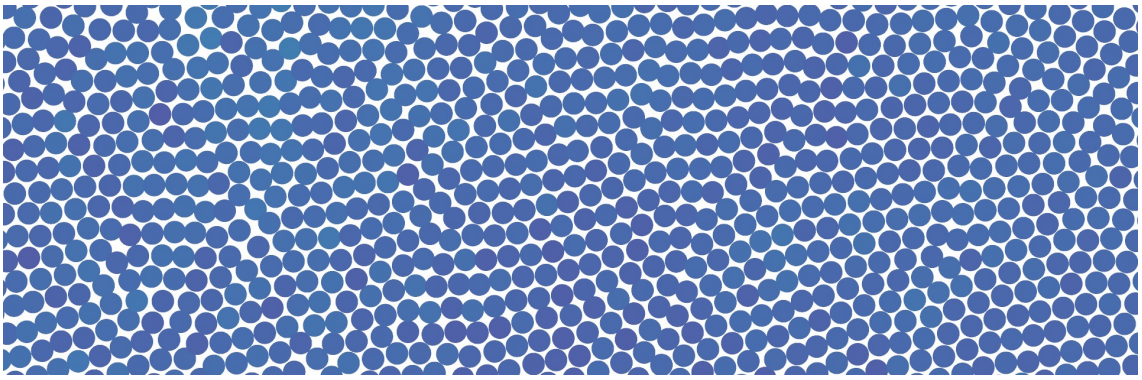


Figure 5.5: The same dambreak scenario with the same parameters as in Figure 5.3 (except $h = 0.0198m$) is simulated for $25s$ and a close-up screenshot of the now almost resting fluid taken. Despite being initialized with a regular lattice and no jitter, the particles settle into chunks of seemingly hexagonal crystal structure, which appears to be a particularly energetically optimal configuration.

5.3 Solving for Uniform Density

The solver discussed in section 4.3 uses density invariance as a source term to enforce incompressibility, where a measured density that deviates from the rest density is penalized. This can lead to many complications for the initialization of the fluid, some of which are:

1. the jitter discussed earlier, for example, would lead to random compressions and therefore random pressure forces even in a fluid that is supposed to have no relative movement of particles, like a volume initially in free fall (neglecting air drag, surface tension etc.)

2. where the fluid borders on a boundary, erroneous compressions might occur, especially since one might want to sample the boundary at a different resolution and with a different lattice as the fluid, for reasons previously discussed

3. the measured density of the fluid is smaller than the rest density ρ_0 at the free surface, where particle deficiency in the fluid particles' neighbourhoods cause the density to be underestimated

All of these complications can reduce the quality of the simulation, particularly in the initial timespan. In this implementation, it is proposed to solve this problem by not initializing every particle with the same mass $m_i = \frac{\rho_0}{h^d}$ but instead iteratively solving a system for m_i such that initially, the exact rest density is measured at every fluid particle. This elegantly solves all the problems mentioned above, while introducing some side effects that come with fluid particles having slightly different masses.

The method is similar in spirit to the derivation of γ_1, where the property of a normalized kernel function in Equation 3.13 relating a sum over neighbours' positions to the volume of a particle is used to normalize masses such that a specific condition is satisfied. To make this simpler, one can reuse the density update that incorporates the boundary particles of arbitrary sampling:

$$\rho_i = \sum_j m_j W_{ij} + \sum_k m_k W_{ik} \tag{5.12}$$

Since the rest volume of each fluid particle was fixed when designing the lattice of initial sampling positions to h^d for d dimensions and the constant rest density ρ_0 is also fixed, the required change in mass from a previous guess m_i^l can be calculated using a factor:

$$\frac{m_i^*}{m_i^l} = \frac{\rho_0}{\rho_i^l} = \frac{\rho_0}{\sum_j m_j^l W_{ij} + \sum_k m_k W_{ik}} \tag{5.13}$$

However, this does not equilibrate the system to rest density immediately, since the mass of each particle depends on the mass of its neighbours. Imagine, for example, a particle at a free surface that experiences particle deficiency and therefore needs to increase its mass to ensure rest density: its immediate neighbours inside the fluid that had their neighbourhood filled with particles of equal masses m_0 now need to reduce their masses to ensure rest density, and so on. An iterative scheme is instead chosen

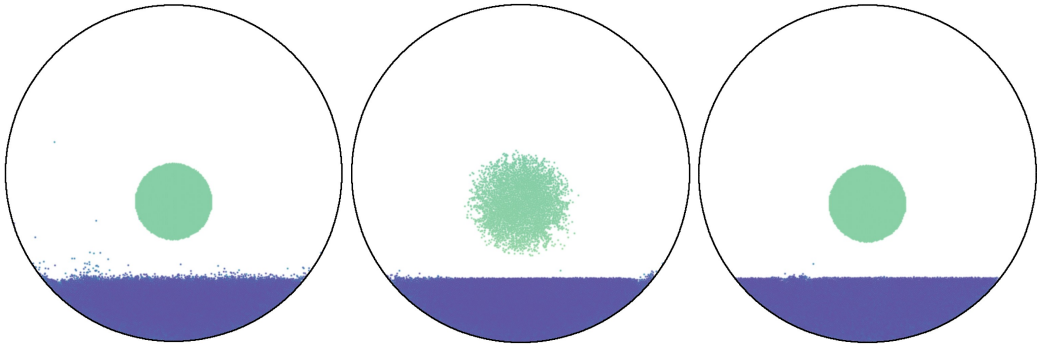
to ease the system into a state of uniform density everywhere using:

$$m_i^0 = \rho_0 h^d \tag{5.14}$$

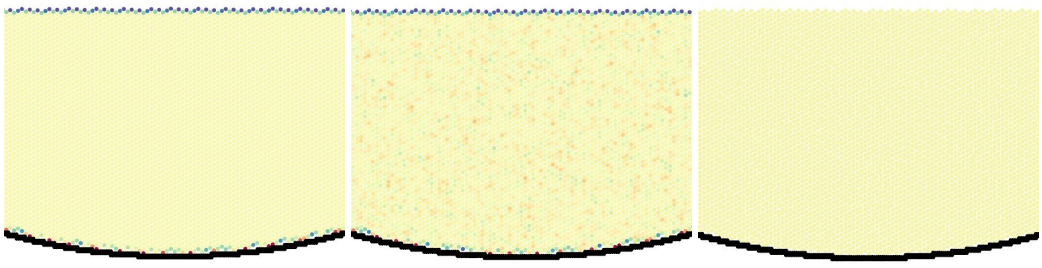
$$m_i^{l+i} = \frac{1}{2} m_i^l + \frac{1}{2} \left(m_i^l \frac{\rho_0}{\rho_i} \right) \tag{5.15}$$

where the ideal mass $\rho_0 h^d$ is the initial guess. Since this computation is a one-time cost at the beginning of the simulation and has no runtime cost, a relatively strict convergence criterion may be used here. In this implementation, a maximum absolute density deviation less than $10^{-3}\rho_0$ in $100 \leq l \leq 1000$ iterations was chosen as a stopping criterion.

A more efficient scheme could also be investigated, for example by choosing a parameter ω different from $1/2$ in the interpolation towards $m_i^l \frac{\rho_0}{\rho_i}$. Anyhow, this scheme does result in a uniform rest density at all fluid particles, irrespective of how exactly the fluid volume was discretized, while keeping the average mass of the particles around $\rho_0 h^d$ (although a slightly higher value is expected for small simulations, where the fraction of particles at surfaces that must compensate for neighbour deficiencies is higher). The effect of the uniform initial density on sampling artefacts can be observed in Figure 5.6.



(a) $t = 0.5s$, no jitter, no uniform density (b) $t = 0.5s$, jitter, no uniform density (c) $t = 0.5s$, jitter and uniform density



(d) $t = 0$, no jitter, no uniform density (e) $t = 0$, jitter, no uniform density (f) $t = 0$, jitter and uniform density

$N = 13532, h = 0.02m, \lambda = 0.1, k = 1000, \nu = 0, \gamma_1 = \gamma_2 = 1,$ SplitSPH Solver

Figure 5.6: Simulation frames of a large, round drop falling into a filled, circular basin are shown for a hexagonal initial sampling. The first row uses velocities while the zoomed-in second row uses density to colour-code particles. As can be seen in Figure 5.6d, naively initializing all particles to equal mass with no jitter can cause over- and underestimated densities at the boundaries of each fluid volume, causing artefacts as observed in Figure 5.6a. Using jittered initial positions with $\sigma = 0.025h$ actually aggravates this effect as observed in the second column, since the density inside the fluid is now also erroneously sampled. The last column shows the effect of solving a system for uniform initial densities in conjunction with jittered positions: all particles in Figure 5.6f are initially at rest density and the artefacts at $t = 0.5s$ in Figure 5.6c are reduced, despite the challenging choice of parameter $\nu = 0$.

Chapter 6

Analysis

Having derived the governing equations, discretized them, established initial conditions, boundary conditions and solvers that propagate solutions through time, the description of the implemented solvers has concluded and what is left to discuss is the impact of some design choices made along the way. In particular, the impact of non-uniform masses on the behaviour of the fluid will be discussed in the following, as well as the oscillation that results from having chosen density invariance as the source term for pressure computations. Lastly, the stability of the simulation is discussed in terms of the stiffness parameter k that has been left unspecified, as well as time step size and viscosity.

6.1 Impact of Jittered Masses on Viscosity and Structure

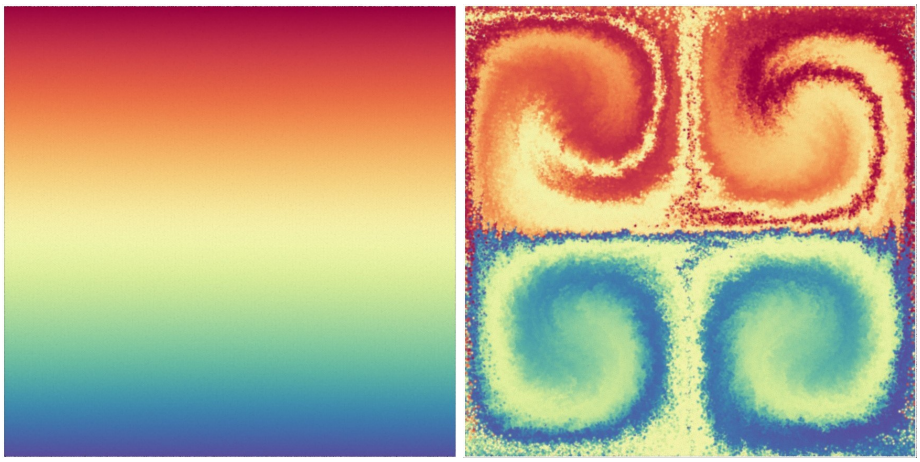
A particularly interesting observation that ties the topic of lattices, random jitter and uniform densities together is the fact that for a uniform initial density combined with a jitter, the mass of the particles is necessarily non-uniform and varies pseudo-randomly. This actually appears to lead to defects in the formation of the crystalline structures that otherwise naturally form, as seen in Figure 5.5, resulting instead in a more amorphous structure. This could have desirable effects, such as increasing isotropy or perhaps reducing undesired viscosity. Intuitively, a particle in a crystalline structure may be expected to require more energy to be moved along an arbitrary direction, than if it had settled into a less rigid structure.

To test this hypothesis, a scene very similar to the Taylor-Green vortex was used, where fluid is initialized at rest density using some lattice as described above, but given a varying degree of pseudo-random jitter in the initial positions. The fluid is contained in a box with no gravity, where all viscosities are in this case set to zero in order to measure only the undesired loss in kinetic energy inherent to the simulation method. The particles are initialized with a velocity that varies according to a sine and cosine function of initial positions $\vec{\boldsymbol{x}}_i(t_0)$ such that four opposing vortices form in each of the four quadrants of a $[0; 2\pi]^2$ domain, using in this case[30]:

$$\vec{\boldsymbol{v}}(t=0)_i = 2 \begin{pmatrix} \sin(x_i)\cos(y_i) \\ -\cos(x_i)\sin(y_i) \end{pmatrix} \tag{6.1}$$

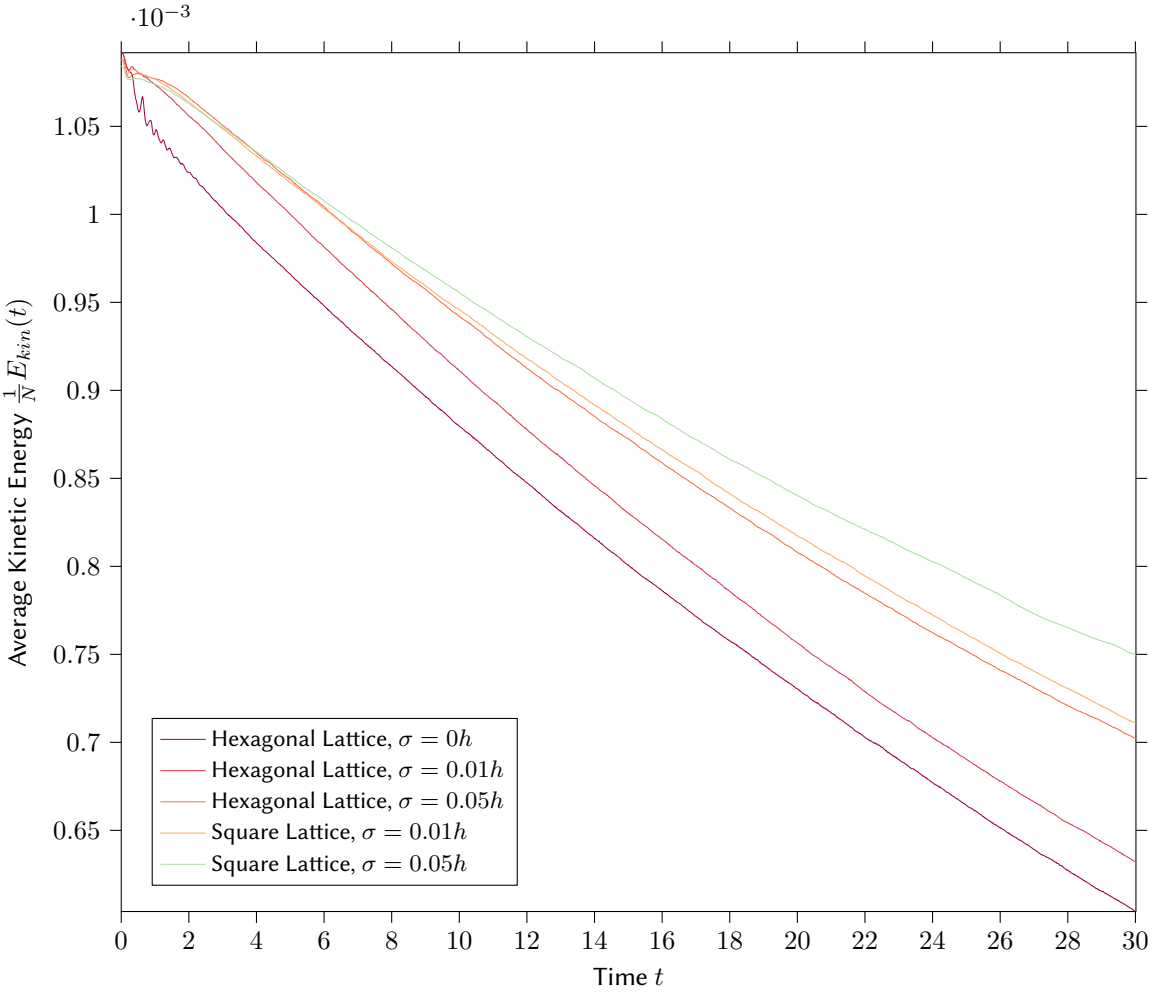
where $\vec{\boldsymbol{x}}_i(t_0) = (x_i, y_i)^T$. The setting is visualized in Figure 6.1, where colour-coded particles are advected.

The rate at which the velocity of the Taylor-Green vortex on a periodic domain decays in relation to the viscosity of a fluid is analytically known to be[30] in $\mathcal{O}\left(e^{-\nu t}\right)$. Despite the exact analytic solution not holding in this instance since periodic boundaries are not applied, the scenario still allows the comparison of decay of the average kinetic energy in the system for different initial samplings of the fluid, where a slower decay is more desirable in reaching lower effective viscosities. A $\frac{1}{N}E_{kin}(t)$ curve can be plotted for different initial sampling lattices and amounts of jitter in conjunction with uniform initial density as described in section 5.3, which is shown in Figure 6.2



(a) $t = 0s$ (b) $t = 4.7s$

Figure 6.1: 95500 colour coded particles are advected by a Taylor-Green vortex on a $[0; 2\pi]$ domain, with four vertices each spinning in directions opposite to their respectively adjacent vortices.



$\nu = \nu_2 = 0, k = 1000, h = 0.033, \lambda = 0.1, N = 90500K, \vec{x} \in [0; 2\pi]^2, v_{x_0} = 2\sin(x)\cos(y), v_{y,0} = -2\cos(x)\sin(y), \rho_0 = 1$ SplitSPH

Figure 6.2: Time evolution of $\frac{1}{N}E_{kin}$ in Taylor-Green vortex for varying amounts of initial jitter and initial sampling lattices. As the initial Jitter approaches a standard deviation of 5% of the particle spacing h, there is a continuous decrease in undesired viscosity. The hexagonal lattice being the most stable in many a sense is detrimental in this case, where to achieve low viscosities, a small jitter and an initial square-lattice sampling appear more effective.

Figure 6.2 suggests that viscosity does in fact decrease as the mass varies more intensely, at least up to a reasonable amount of initial jitter. In order to empirically examine whether this is actually due to fewer rigid, crystalline structures forming, a metric for the degree of crystallinity or amorphousness of a material, or in other words a structural order parameter, is required. For this, a metric from the study of two-dimensional melting in condensed matter physics may be borrowed: the Nelson-Halperin 2D bond orientational order parameter[31] Ψ_6. It is defined as[32]:

$$\Psi_6^k = \frac{1}{n_k} \sum_{l \in \mathcal{N}(k)} e^{6i\Theta_{k,l}} \tag{6.2}$$

where the sum is over the n_k nearest neighbours $k \neq l$ to the particle of index k determined through a Delaunay triangulation, $\Theta_{k,l}$ is the angle between particles k, l measured from an arbitrary, fixed axis and the usual particle index i was avoided to not cause confusion with the imaginary unit in the exponent[32]. Generally, the Ψ_x metric shows how close to perfect x-atic symmetry the local environment is, which is why the hexatic Ψ_6 metric was chosen for this two-dimensional case where the hexagon is the unique closest packing and six nearest neighbours are indeed the most frequent result of a Delaunay triangulation of the particle positions for all simulation runs. The value $0 \leq |\Psi_6| \leq 1$ is maximized for a hexagonal grid and decreases as the material becomes less ordered[32], where magnitude expresses regularity or how 'well-packed' the arrangement is[33], while the complex number itself also encodes a phase or orientation of the structure.

If there was a relation between jittered masses, lower viscosity and preventing crystal structures, one would expect the average magnitude of the bond orientation parameter $\Psi_6^{avg} = \frac{1}{N}\sum_i \left|\Psi_6^i\right|$ to decrease as the jitter increases and the material becomes more disorderly. To test this, Ψ_6^{avg} was measured at $t = 30$ for all the configurations in Figure 6.2, long after the initialization, with results shown in table 6.3.

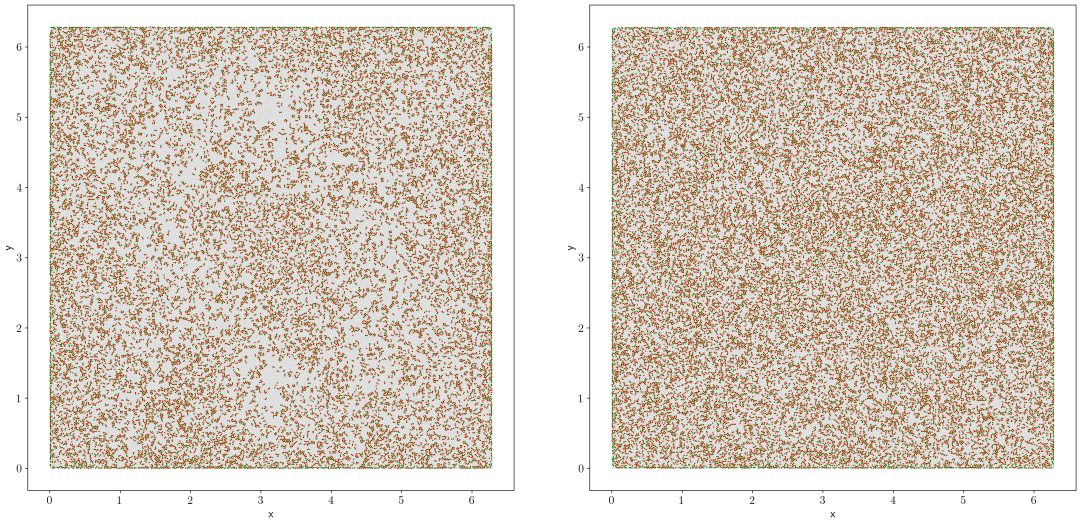
Lattice	Time	Jitter σ	Ψ_6^{avg}
Hexagonal	t=0	0	0.973
Hexagonal	t=30	0	0.522
Hexagonal	t=30	0.01h	0.511
Hexagonal	t=30	0.05h	0.459
Square	t=30	0.01h	0.478
Square	t=30	0.05h	0.444

Figure 6.3: The average bond orientational parameter Ψ_6^{avg} is shown for different initial samplings and amounts of jitter. The $t = 0$ entry highlights how a hexagonal structure maximizes this value, while increasing jitter results in lower values, even at $t = 30$.

As can be seen, irrespective of which lattice is used to sample the fluid, there are more defects, less order and therefore lower values of Ψ_6^{avg} as the jitter increases, even long after the initial conditions should have no more bearing on the behaviour of the fluid. This further suggests that the combination of jitter and solving for uniform density could indeed make the discretization more isotropic and help decrease unwelcome viscosity. On the other hand, especially for small kernel support radii, the SPH approximation quality might suffer from excessive particle disorder.

Just to further drive home the point that order decreases as particle masses vary, the Voronoi tessellations of the particle positions from the same simulation runs can be analysed instead of Ψ_6. Then, particles the Vornoi cells of which form a polygon with six vertices, more than six or less than six vertices can be coloured respectively and plotted, as seen in Figure 6.4. This again suggests that jittered masses lead to more defects in otherwise locally ordered structures.

In sources on melting of condensed matter in two dimensions, the hexatic phase is frequently examined, creating a continuous middle ground between the solid and liquid phases and bearing a surface-level resemblance to the structures seen in Figure 5.5 - it might be interesting to draw from knowledge about these physical processes in order to combat undesired viscosity in SPH fluid simulation and reach higher effective Reynolds numbers, analysing for example the time evolution of translational and orientational order throughout a simulation and how masses and kernel support radii that vary per particle or resampling of fields can influence these phenomena - however those analyses are not conducted in this report.



(a) $\sigma = 0$ (b) $\sigma = 0.05h$

Figure 6.4: The result of the Vornoi tessellation of the simulations of the Taylor-Green vortex are shown for a hexagonal initial sampling at $t = 30$ with the specified jitter (all parameters are the same as in Figure 6.2). Each particle that has a Voronoi cell with more than six vertices is coloured red, ones with fewer neighbours are coloured green. While for $\sigma = 0$, there are 61759 particles that have hexagonal Voronoi cells, for $\sigma = 0.05h$ jitter there are only 52845 such particles, as can be seen observed in the increased amount of coloured dots in the right figure. This further suggests that pseudo-randomly varying particle masses create a less ordered and possibly more isotropic structure down the line.

6.2 Oscillation Frequency and Error as a Function of Stiffness

By choosing a density invariance term in the formulation of the fluid solvers instead of the velocity divergence, the fluid can be expected to oscillate about its rest density for the non-iterative solvers `SplitSPH` and `EOSSPH` instead of exhibiting volume drift.

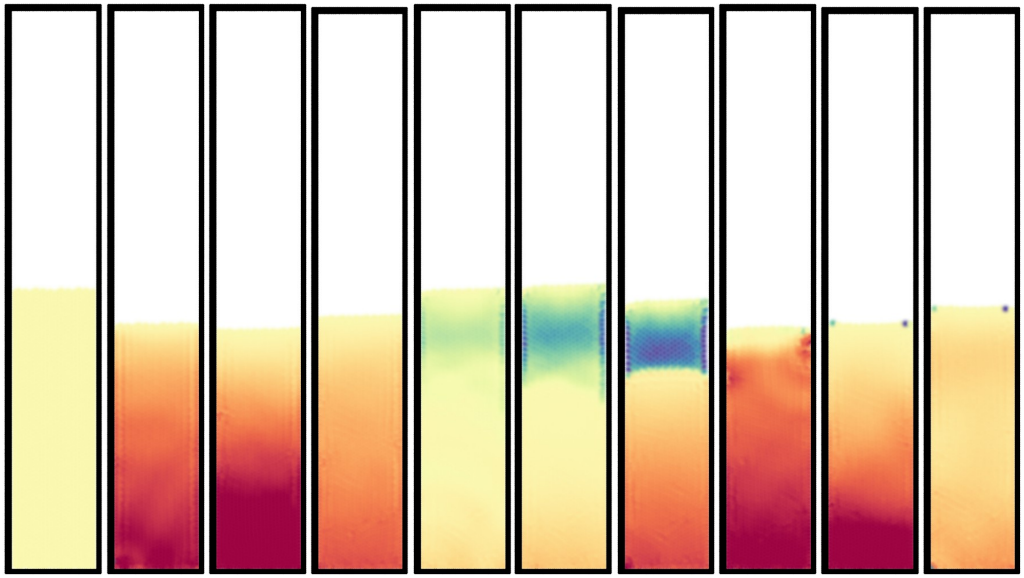
The iterative solver behaves somewhat similarly, but does not oscillate about the rest density in a sinusoidal fashion since incompressibility is more strongly enforced with a density error threshold that the average density does not significantly exceed, resulting in peaks of the oscillations in density being cut off. To make the `IterSPH` solver more comparable, a fixed amount of iterations can be used, $l = l_{min} = l_{max} = 3$ and $l = 2$ respectively in this case, which makes the oscillation regular but disables any error thresholds or guarantees.

One can analyse the behaviour of this oscillating error using a simple column of water in a tall, rectangular container, since it is known that this configuration would ideally result in a perfectly still fluid that does not oscillate. The width of the container has little bearing on the magnitude of the error, which is why a narrow and tall configuration is preferred in order to analyse the same effect using fewer particles and therefore less computation time. Both the amplitude and the frequency of this oscillation can be found to depend on the stiffness parameter k that governs incompressibility in the equation of state. In this section, the influence of k on the magnitude of the error and the frequency of the oscillation are analysed. The setting and the oscillation are shown in Figure 6.5

The effect can be thought of as an oscillation due to the fact that particles at the top of the column are still in free fall when the bottom particles first experience contact forces - the stiffer the fluid, the faster the shock within it propagates and the quicker the particles at the top of the column are accelerated back up, where gravity takes over and the cycle continues, albeit with smaller and smaller amplitude.

In order to conduct the analysis, the measured density according to Equation 3.11 can be averaged across all particles to obtain $\rho_{avg}(t) = \frac{1}{N}\sum_i \rho_i(t)$, and the rest density subtracted to obtain the average density error $\Delta\rho(t) = \rho_{avg}(t) - \rho_0$, which ideally oscillates about zero. This is done for a simulation running with a specified, fixed time step Δt; the CFL-condition is not used in favour of obtaining a regular sampling of the $\Delta\rho(t)$-Curve which can be more easily subjected to a discrete Fourier transform to analyse the frequency of the oscillation. In the following, a time step size of $\Delta t = 0.0002s$ is used, which is a conservative estimate of the Δt required by the CFL condition. This means that $\Delta\rho(t)$ is sampled at $5000Hz$ and the frequency:

$$f_{peak} = \underset{f}{\operatorname{argmax}}\, \mathcal{F}\left(\Delta\rho(t)\right) = \underset{f}{\operatorname{argmax}}\, \Delta\rho'(f) \tag{6.3}$$



(a) $t = 0.0s$ (b) $t = 0.2s$ (c) $t = 0.3s$ (d) $t = 0.4s$ (e) $t = 0.5s$ (f) $t = 0.6s$ (g) $t = 0.7s$ (h) $t = 0.8s$ (i) $t = 0.9s$ (j) $t = 1.0s$

$k = 100, \nu = 0.01, \nu_2 = 0, h = 0.02, \lambda = 0.1, N = 1572, \rho_0 = 1, \Delta t = 0.0002s$ SplitSPH

Figure 6.5: A half-filled, tall, rectangular tube is shown with the density of the fluid being colour-coded. For the very small stiffness parameter $k = 100$ shown here, major compressions occur and a clearly visible, slow oscillation sets in.

with maximum contribution to the spectrum of the oscillation is plotted against k, neglecting the zero-frequency f_0 which is plotted in a separate graph and expresses the magnitude of the total error. This is done for $2s \leq t \leq 12s$, where the first two seconds of simulated time are discarded in order to minimize the impact of the initial configuration.

The results of the frequency analysis are shown in Figure 6.6. This graph shows that as the stiffness k increases, so does the frequency of the oscillation. The graph closely resembles a square root function, which might be explained by the stiffness being linked to the **NUMERICAL SPEED OF SOUND** c, which governs the frequency of the oscillations[34], through the Newton-Laplace equation that states[35]:

$$c = \sqrt{\frac{\Delta p}{\Delta \rho}} \tag{6.4}$$

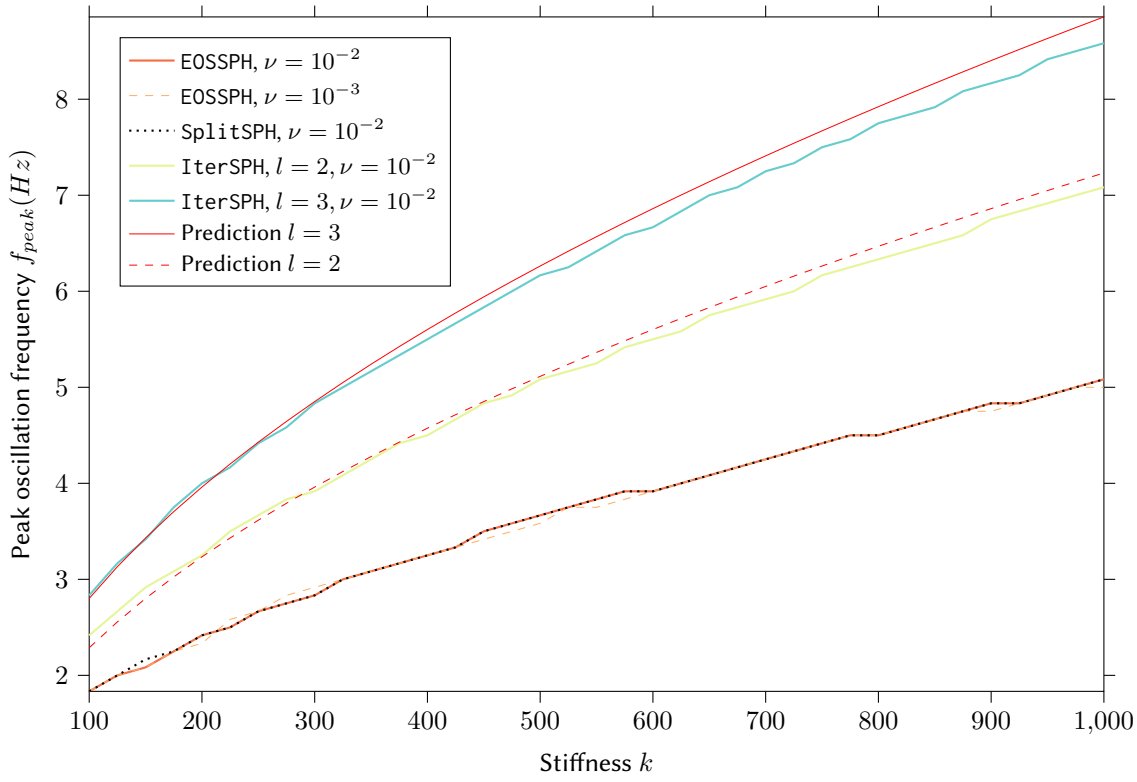
meaning that for the equation of state in Equation 2.45 for which $\Delta p \propto k\Delta\rho$ and if the oscillation frequency is indeed proportional to the numerical speed of sound ($f_{peak} \propto c$), then:

$$f_{peak} \propto c = \sqrt{\frac{\Delta p}{\Delta \rho}} \propto \sqrt{\frac{k\Delta\rho}{\Delta\rho}} = \sqrt{k} \tag{6.5}$$

and therefore $f_{peak} \propto \sqrt{k}$, which would explain the shape of the curve.

The graph also clearly shows that the oscillation is *not affected* by viscosity, with simulations that differ in ν by an order of magnitude exactly overlapping, and that operator splitting *also does not affect* the result at all. This further evidences that the oscillation frequency depends on the numerical speed of sound and not any of the other factors mentioned. The only solver with a differing curve is the iterative solver: for this solver, one would expect the numerical speed of sound to be higher, since there are multiple iterations per time step that may propagate information, or accelerations, from neighbouring particle to neighbouring particle - this makes intuitive sense, since the iterative solver achieves a higher effective incompressibility, which has been shown to be linked to the numerical speed of sound in Equation 6.4.

One could speculate that within some low bound of l_{iter}, where repeated neighbourhood calculations are not necessary, the number of iterations could have a nearly linear relationship with the stiffness k, the square root of which was assumed above to be proportional to the frequency of oscillation - if this were true, one would expect the iterative solver with $l_{iter} = 2$ to have a higher frequency of oscillation by a factor of $\sqrt{l_{iter}} = \sqrt{2}$ and similarly for $l = 3$. To test this hypothesis, the average of the four curves using a single-iteration solver as seen in Figure 6.6 was taken (EOSSPH, SplitSPH for $\nu = 10^{-2}, \nu - 10^{-3}$



$\nu_2 = 0.001, h = 0.02, \Delta t = 0.0002s, N = 1572, \rho_0 = 1, \gamma_1 = 1, \gamma_2 = 0.5$, Hexagonal sampling with $\sigma = 0.015h$

Figure 6.6: The frequency f_{peak} with maximum contribution to the oscillation of the function $\Delta\rho(t)$ is plotted against the stiffness parameter k that governs incompressibility. The frequency of oscillation continuously increases as k increases. The curves of the EOSSPH and SplitSPH simulations all overlap and so do all curves that differ only in viscosity, so only a selection of such curves was plotted.

respectively), a square root function fitted to this average using the Levenberg-Marquardt algorithm for least squares regression[1] and then multiplied by $\sqrt{l_{iter}}$ to yield the red line and dashed red line shown in Figure 6.6 as the *Prediction*. These curves describe what would be expected to happen if the hypotheses made in this section were true, meaning:

1. $f_{peak} \propto \sqrt{k}$
2. $k \propto l_{iter}$

and matches the experimental values for both the $l = 2$ and the $l = 3$ iterative solvers remarkably well, suggesting that stiffness may indeed be almost linear in the number of iterations for such solvers, at least up to some small, fixed number of iterations.

The total density error across time can be expressed by the zero-frequency f_0 of the discrete Fourier transform of $\Delta\rho(t)$ as shown in Figure 6.7, where:

$$f_0 = \int_{2s}^{10s} \Delta\rho(t)\, dt \tag{6.6}$$

Note that $\Delta\rho(t)$ should oscillate about zero, so the integral should be zero if the solver perfectly enforces incompressibility, greater than zero if some compression occurs and negative if somehow the measured average density across particles and time is below rest density.

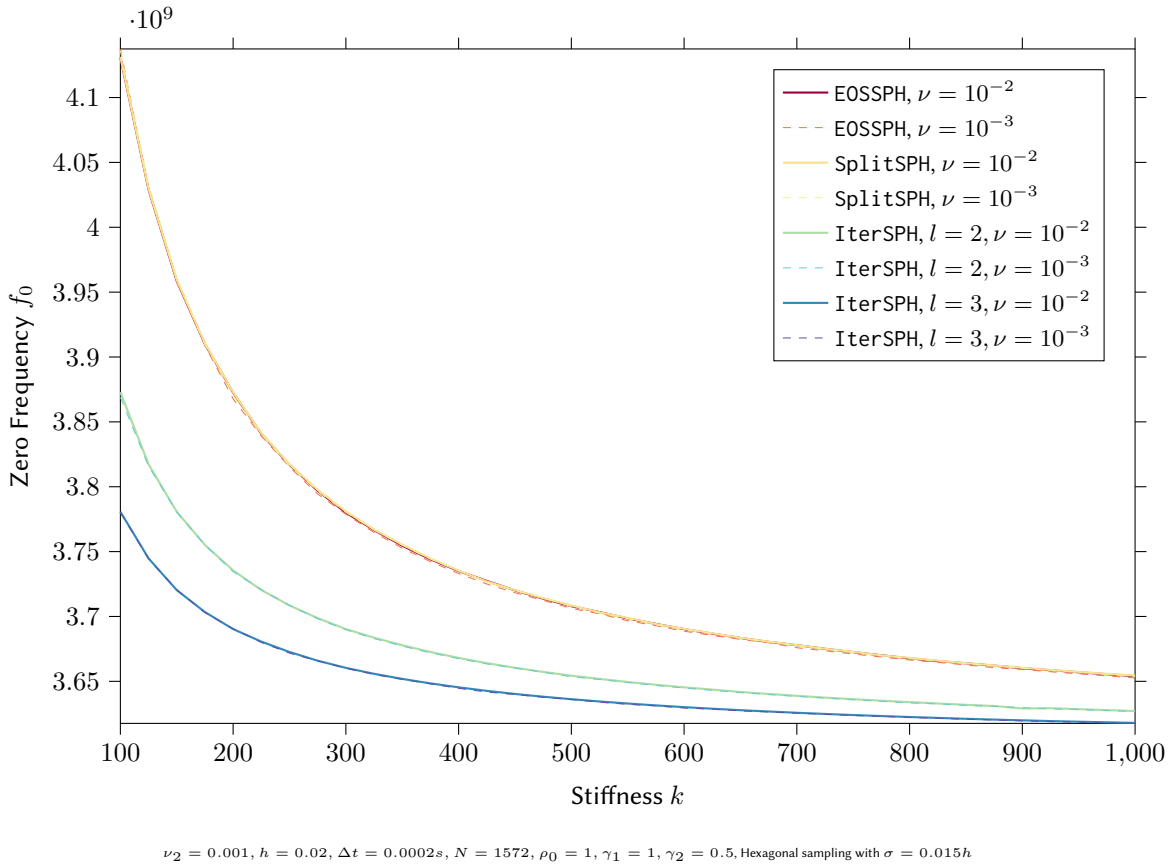
Here, the data shows that a greater stiffness coefficient k results in less density error, which makes sense since compressions for higher k result in greater pressure forces that correct the compression.

[1] Implemented in SciPy Optimize: Curve Fit https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html

The graph also shows how increasing the stiffness k yields diminishing returns, further motivating the use of iterative solvers as a way to improve simulation quality, since simply increasing the stiffness used in a single iteration per time step might become unfeasible. The iterative solver achieves a much smaller value of f_0 across all coefficients k, with higher iteration counts further decreasing the error, but once again an increase in iteration count appears to yield diminishing returns.

In this instance, the single-iteration solvers with and without operator splitting again show the same behaviour, with viscosity once more playing no role in the oscillation for the measured range of parameters. Operator splitting, while perhaps effective in reducing some kinds of errors and improving robustness and stability, does not have an impact on the type of error under consideration in this scenario.

At some point, when increasing k, instability might occur if the time step size Δt is not simultaneously decreased, making the simulation more computationally expensive - this is analysed in the following section.



$\nu_2 = 0.001, h = 0.02, \Delta t = 0.0002s, N = 1572, \rho_0 = 1, \gamma_1 = 1, \gamma_2 = 0.5$, Hexagonal sampling with $\sigma = 0.015h$

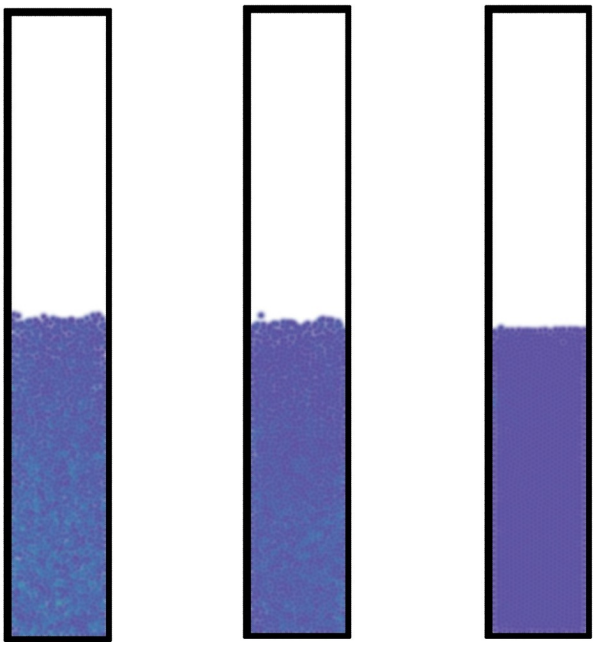
Figure 6.7: The zero frequency f_0 or integral of the function $\Delta\rho(t)$ is plotted against the stiffness parameter k that governs incompressibility. As the stiffness increases, the total error sharply decreases. The viscosity and whether operator splitting is applied has no impact on the single-iteration solvers EOSSPH and SplitSPH, the curves of which all overlap. The IterSPH solver with achieves a lower total error across all k, with higher iteration counts resulting in lower errors but seemingly yielding diminishing returns.

6.3 Stability as a Function of Viscosity, Stiffness and Time Step Size

The setting described in section 6.2, where a column of water at rest is simulated, can also be used to analyse the stability of the simulation. Once again, the fact that the water ought to be at rest and therefore the analytic solution is trivially known can be used to measure an error ϵ relative to that solution for each of the solvers presented in chapter 4. It has already been discussed that higher stiffness k can lead to better simulation accuracy, but it has not yet been shown that a choice of k that is too large for a given time step size λ can lead to unstable behaviour with erratically moving particles and unphysical behaviour. Ideally, the average kinetic energy of the fluid particles $E_{kin}(t) = \frac{1}{N}\sum_i m_i \left|\left|\vec{\boldsymbol{v}}_i(t)\right|\right|^2$ is at zero, or very close to it, for the resting water column shown in Figure 6.5. Instabilities can be detected by the kinetic energy being considerably higher than expected and repeatedly spiking, which is measured in the following using the logarithm of the average time integral of kinetic energy of the particles, or assuming a potential energy of zero, the logarithm of the average action:

$$\epsilon = \log_{10}\left(\int_{10s}^{20s} E_{kin}(t)\,dt\right) \tag{6.7}$$

where the first 10 seconds are discarded to make sure that any movement due to unfortunate initial conditions and the oscillations measured in section 6.2 have subsided, the state of the particles is somewhat equilibrated, and instabilities are responsible for any remaining, significant amounts of kinetic energy. The time integration in Equation 6.7 is implemented using the trapezoidal rule and the logarithm is taken to account for an unstable simulation potentially having multiple orders of magnitude more kinetic energy than a stable simulation in this scenario. The scenario for one set of parameters and varying solvers is shown in Figure 6.8, while the results of the analysis are shown in Figure 6.9.

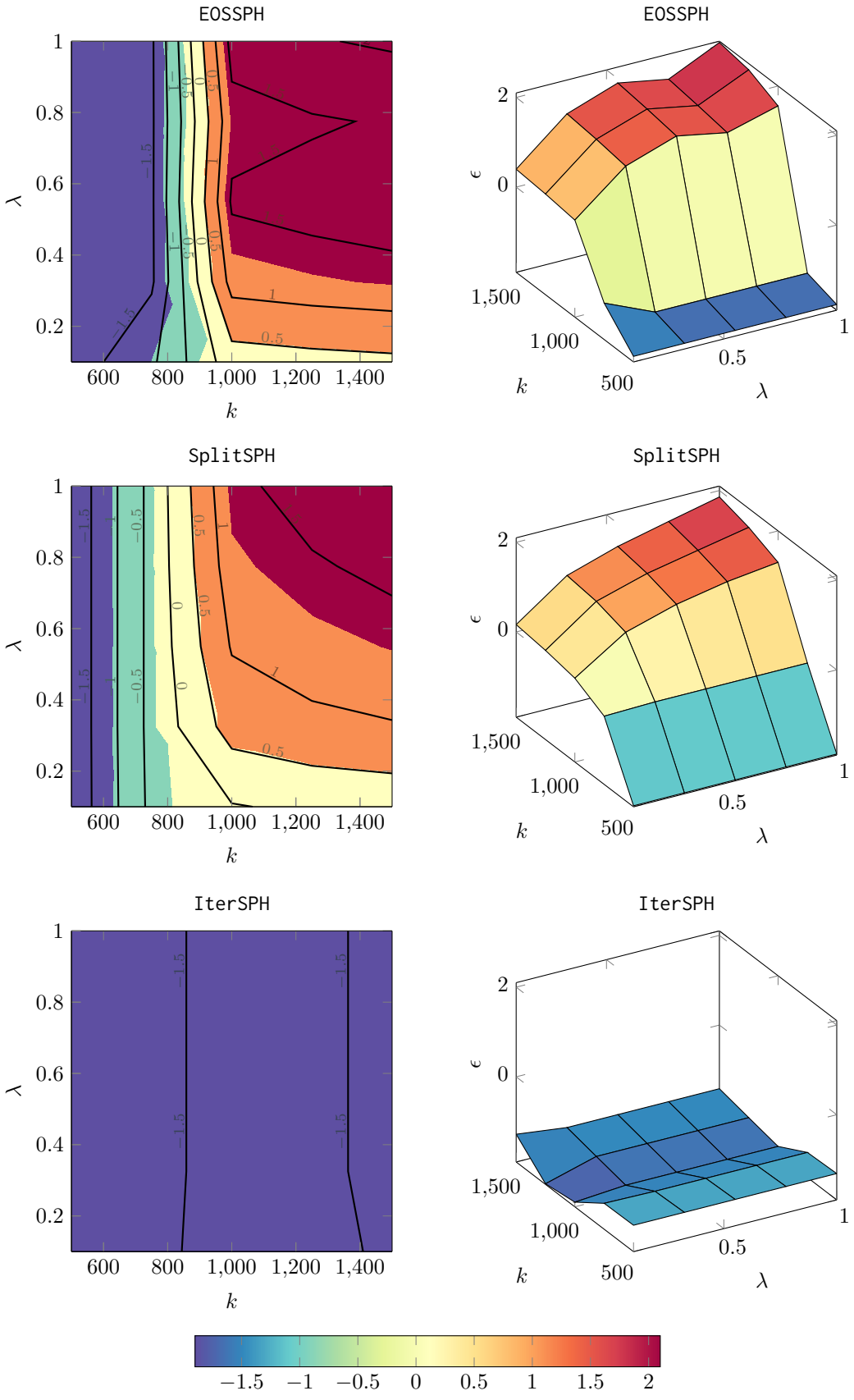


(a) EOSSPH (b) SplitSPH (c) IterSPH

$\lambda = 0.1, \nu = 0.0001, \nu_2 = 0.001, h = 0.02, N = 1572, \rho_0 = 1, \gamma_1 = 1, \gamma_2 = 0.5$, Hexagonal sampling with $\sigma = 0.015h$

Figure 6.8: The exact same setting with the same parameters is shown at $t = 5s$ for different solvers, showcasing how operator splitting can increase stability and the iterative solver further improves this. The magnitudes of velocities are colour-coded.

The figure reveals a couple of interesting facts about the solvers in the chosen range of parameters. Firstly, it can be observed that in this scenario, the iterative solver vastly outperforms all other solvers in terms of stability and robustness, being stable across the entire range of tested parameters. There appears to be trough along the $k = 1000$ line, while no discernable gradient in λ-direction can be observed. This means that all tested time step sized lead to roughly equally stable results, while one stiffness in particular results in slightly more stable simulations. This could be an indication that an optimized stiffness k, as



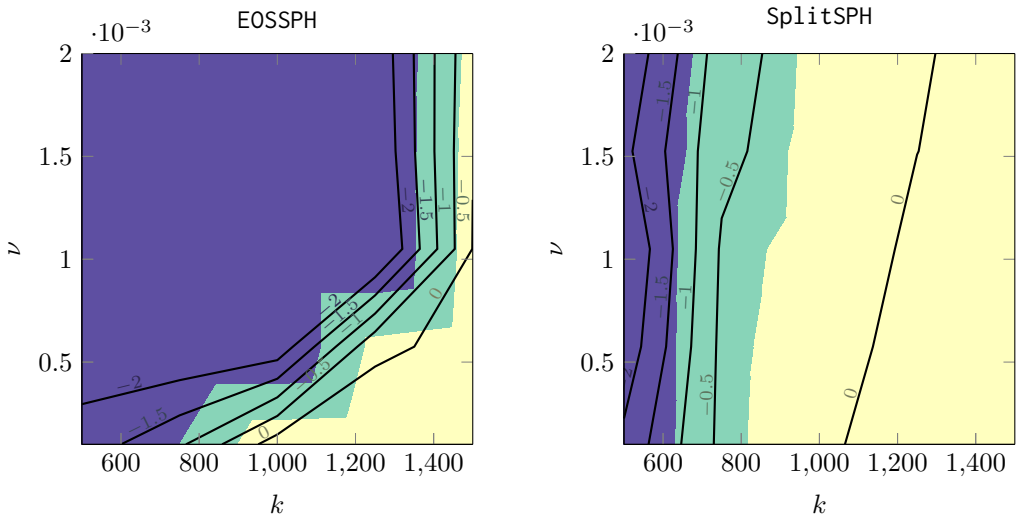
$\nu = 10^{-4}, \nu_2 = 10^{-3}, h = 0.02, N = 1572, \rho_0 = 1, \gamma_1 = 1, \gamma_2 = 0.5, l_{min} = 5, l_{max} = 300$, Hexagonal sampling with $\sigma = 0.015h$

Figure 6.9: The error ϵ (Equation 6.7) in the kinetic energy of the resting water column is plotted for different parameters of time step size $\lambda \in [0, 1]$ and stiffness $k \in [500, 1500]$ for 5×5 datapoints. Each row shows one solver, where the left column are contour plots and the right column shows the same values in 3D for better visual clarity. The mapping from values to colours is shown in the bar at the bottom.

is often derived for iterative solvers, is indeed important for optimal behaviour of the solver. Reducing the time step size on the other hand may not be as effective at improving accuracy in a scenario where relatively large time steps are tolerated, further motivating the use of a time step size closer to the largest stable value.

The EOSSPH and SplitSPH solvers behave as one might expect, with less stable behaviour as the stiffness k and the time step size λ are increased. This leads to the aforementioned conclusion that in order to achieve a desired level of both incompressibility and stability, k should be increased until the incompressibility requirement is met and λ decreased until the simulation is stable. Figure 6.9 also shows that operator splitting helps improve stability and robustness by shifting the stable region of parameters in the positive λ-direction: whereas the EOSSPH solver has a steep gradient along the k-direction, with impractically low stiffnesses being drastically more stable, a more distinct second gradient along the λ-direction can be observed for the SplitSPH solver. This means that even for large values of k a relatively small decrease in time step size can help significantly improve the stability of the simulation. For the EOSSPH solver one would expect the same to be true, but for much smaller time step sizes, which in turn increase the computational cost of the simulation.

The same analysis can be made for varying viscosity ν and stiffness k. Once again, the iterative solver produces uninteresting data since it is stable across the entire range of parameters. The results for the two remaining solvers are shown in Figure 6.10.



$\lambda = 0.1, \nu_2 = 10^{-3}, h = 0.02, N = 1572, \rho_0 = 1, \gamma_1 = 1, \gamma_2 = 0.5$, Hexagonal sampling with $\sigma = 0.015h$

Figure 6.10: The error ϵ (Equation 6.7) is plotted for different parameters of time step size $\lambda \in [0, 1]$ and viscosity $\nu \in [10^{-4}, 2 \cdot 10^{-3}]$ for 5×5 datapoints. The mapping from values to colours is the same as in Figure 6.9.

This graph reveals that for the EOSSPH solver, an increase in viscosity can improve stability, at least up to a certain value of ν, while very inviscid fluids are more challenging to simulate stably. More interestingly this does not seem to apply as much to the solver with operator splitting, which shows an error gradient exclusively in the k-direction. This might be due to the fact that the viscosity in the tested parameter range is not large enough for inaccurate viscous forces to cause instability, which leaves only the pressure forces as a possible reason for instability. These pressure forces correct a predicted density when operator splitting is used, where increased viscosity affects the predicted density, but appears to have less influence on the instabilities resulting from inaccurately correcting these predicted densities - although there are too few data points to be conclusive. It might be interesting to further examine this behaviour by checking if the opposite is true for operator splitting that integrates pressure forces first and viscosity afterwards, to check if for such a solver stiffness becomes less relevant for stability if the viscosity is sufficiently high.

Chapter 7

Conclusion and Future Work

In this report, the Navier-Stokes equations have been derived, discretized, initial conditions and boundary conditions discussed and three variations of a solver for the system of partial differential equations have been presented, each building up from a simple solver using an equation of state, through the concept of operator splitting towards an iterative solver with superior stability and incompressibility guarantees. Parameters that affect the stability, numerical viscosity and the quality of the initial sampling of the continuous fields into discrete particles have been analysed.

The generality of the SPH scheme as a method of discretizing and interpolating fields was highlighted, decoupling it from preconceptions about the more specific implementations that commonly use SPH to simulate fluids, such as the density invariant source terms and even Lagrangian frameworks not being related to SPH by necessity but rather through choice. Gaussian-like kernels in particular, which are often taken for granted in SPH literature, were appreciated and possible reasons for their widespread use were given. An uncommon form of an iterative solver that more directly shows the connection between equation-of-state based solvers and more elaborate iterative solvers commonly in use was given, proving that the hurdle towards implementing a solver that can uphold incompressibility might not be as high as it seems.

A particular emphasis was placed on discussing how a field can be discretized to yield initial conditions that prevent aliasing artefacts and how the lattice that a field is sampled with, whether particles of uniform masses or densities are sampled and if a regular or pseudorandomly jittered configuration is chosen can influence the simulation quality in conjunction. This is a section of the report in that could use a more thorough investigation into what causes numerical viscosity and what parameters could positively affect this behaviour.

The frequency of oscillations observed when using the absolute density to enforce the continuity equation was analysed and some relatively speculative claims where made that also warrant further investigation, such as the exact nature of the relation between stiffness, speed of sound, oscillation frequency and the number of iterations of a solver. This might be an area of particular interest, since unwelcome oscillations can be seen as a major flaw in many SPH-based fluid solvers.

All in all, there are many angles from which the observations laid out in this report could be more thoroughly theoretically underpinned and empirically validated or taken as motivation to investigate a perhaps less vigorously researched branch of improvements to fluid simulation with Smoothed particle hydrodynamics, and the possibilities for future work in this field seem endless.

Bibliography

[1] J. D. Anderson, *Computational Fluid Dynamics: The Basics with Applications.* McGraw-Hill series in mechanical engineering (New York, NY, May 1995).

[2] D. Koschier, J. Bender, B. Solenthaler, and M. Teschner, "Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids", Eurographics 2019 - Tutorials (2019).

[3] W. M. Lai, D. H. Rubin, D. Rubin, and E. Krempl, *Introduction to continuum mechanics*, Third (Butterworth-Heinemann, 1999).

[4] V. John, *Finite Element Methods for Incompressible Flow Problems*, Vol. 51, Springer Series in Computational Mathematics 51 (Springer, Jan. 2016).

[5] J. Pedlosky, *12.800: Fluid Dynamics of the Atmosphere and Ocean (Course Notes)*, July 2014.

[6] N. Coleman, "A Derivation of the Navier-Stokes Equations", Ball State Undergraduate Mathematics Exchange **7** (2010).

[7] J. Bender, D. Koschier, T. Kugelstadt, and M. Weiler, "Turbulent Micropolar SPH Fluids with Foam", IEEE Transactions on Visualization and Computer Graphics **25**, 2284–2295 (2019).

[8] M. Becker and M. Teschner, "Weakly compressible SPH for free surface flows", in Proceedings of the 2007 acm siggraph/eurographics symposium on computer animation, SCA '07 (2007), pp. 209–217.

[9] J. Monaghan, "Smoothed Particle Hydrodynamics and Its Diverse Applications", Annual Review of Fluid Mechanics **44**, 323–346 (2012).

[10] J. R. MacDonald, "Review of Some Experimental and Analytical Equations of State", Rev. Mod. Phys. **41**, 316–349 (1969).

[11] B. Solenthaler and R. Pajarola, "Predictive-corrective incompressible SPH", in Acm siggraph 2009 papers, SIGGRAPH '09 (2009).

[12] L. B. Lucy, "A numerical approach to the testing of the fission hypothesis.", The Astronomical Journal **82**, 1013–1024 (1977).

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

[14] A. Falaschi, *Signal Processing and Information Theory*, 1st ed. (July 2022).

[15] N. Akinci, G. Akinci, and M. Teschner, "Versatile surface tension and adhesion for SPH fluids", ACM Trans. Graph. **32** (2013).

[16] D. J. Price, "Smoothed particle hydrodynamics and magnetohydrodynamics", Journal of Computational Physics **231**, 759–794 (2012).

[17] P. Getreuer, "A Survey of Gaussian Convolution Algorithms", Image Processing On Line **3**, 286–310 (2013).

[18] P. P. Vaidyanathan, "Eigenfunctions of the Fourier Transform", IETE Journal of Education **49**, 51–58 (2008).

[19] J. Babaud, A. Witkin, M. Baudin, and R. Duda, "Uniqueness of the Gaussian Kernel for Scale-Space Filtering", Pattern Analysis and Machine Intelligence, IEEE Transactions on **8**, 26–33 (1986).

[20] J. M. Hammersley and D. C. Handscomb, *Monte Carlo methods* (Methuen, London, 1964).

[21] E. Parzen, "On Estimation of a Probability Density Function and Mode", The Annals of Mathematical Statistics **33**, 1065–1076 (1962).

[22] M. Teschner, *Simulation in Computer Graphics: Particle Fluids*, (lecture slides), July 2024.

[23] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner, "SPH Fluids in Computer Graphics", Eurographics 2014 STAR - State of The Art Report (2014).

[24] S. Band, C. Gissler, and M. Teschner, "Compressed Neighbour Lists for SPH", Computer Graphics Forum **39**, 531–542 (2020).

[25] M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath, and M. Teschner, "Implicit Incompressible SPH", IEEE Transactions on Visualization and Computer Graphics **20**, 426–435 (2014).

[26] X. He, N. Liu, S. Li, H. Wang, and G. Wang, "Local Poisson SPH For Viscous Incompressible Fluids", Computer Graphics Forum **31**, 1948–1958 (2012).

[27] N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, and M. Teschner, "Versatile rigid-fluid coupling for incompressible SPH", ACM Trans. Graph. **31** (2012).

[28] D. Koschier and J. Bender, "Density maps for improved SPH boundary handling", in Proceedings of the acm siggraph / eurographics symposium on computer animation, SCA '17 (2017).

[29] S. Diehl, G. Rockefeller, C. L. Fryer, D. Riethmiller, and T. S. Statler, "Generating Optimal Initial Conditions for Smoothed Particle Hydrodynamics Simulations", Publications of the Astronomical Society of Australia **32**, e048 (2015).

[30] X. Zhao, B. Protas, and R. Shvydkoy, "On instability of the Taylor-Green vortex in 2D Euler flows", ArXiv (2024).

[31] B. I. Halperin and D. R. Nelson, "Theory of Two-Dimensional Melting", Phys. Rev. Lett. **41**, 121–124 (1978).

[32] M. Ledesma-Motolinía, J. L. Carrillo-Estrada, and F. Donado, "Crystallisation in a two-dimensional granular system at constant temperature", Scientific Reports **11**, 10.1038/s41598-021-96099-9 (2021).

[33] G. A. Pantelopulos, T. Nagai, A. Bandara, A. Panahi, and J. E. Straub, "Critical size dependence of domain formation observed in coarse-grained simulations of bilayers composed of ternary lipid mixtures", en, J Chem Phys **147**, 095101 (2017).

[34] D. Isik and Z. He, "Effect of numerical speed of sound and density diffusion on SPH modeling of rigid body migration in plane Poiseuille flow", Computational Particle Mechanics **10**, 503–517 (2023).

[35] B. Finn, "Laplace and the Speed of Sound", Isis **55**, 7–19 (1963).