Università Degli Studi di Catania
Dipartimento di Ingegneria Elettrica Elettronica e Informatica

Dottorato in Ingegneria dei Sistemi, Energetica, Informatica e delle Telecomunicazioni

XXXI ciclo

Vito Zago

Smoothed Particle Hydrodynamics method and flow dynamics: the case of lava numerical modeling and simulation

Coordinator:
Prof. Paolo Arena

Advisors:
Prof. Luigi Fortuna
Dr. Ciro Del Negro

Abstract

Smoothed Particle Hydrodynamics is a Lagrangian mesh-free method that has been gaining momentum in the field of Computational Fluid Dynamics. Thanks to its nature, SPH is able to deal with complex flows and their features, such as non-Newtonian rheologies, free surface, thermal dependency, phase transition, large deformations, etc. SPH shows an intrinsically parallel nature, that allows its execution on high-performance parallel computing hardware, such as modern Graphics Processing Units (GPUs), gaining advantages in terms of simulation time. In this thesis we will work on GPUSPH, an implementation of the SPH method that runs on GPUs. We will study the simulation of a very complex fluid: lava. The combination of free surface, natural topography, phase transition and the formation of structures such as levees and tunnels makes the modeling and simulation of lava flows an extremely challenging task for CFD that has an important impact in numerous fields of engineering and scientific research. We will see the introduction in GPUSPH of models and strategies that deal with the features characterizing lava

flows, including the development of a semi-implicit scheme, that allows to simulate very high-viscosity fluids ensuring robustness and reducing simulation times. The new implementation will be tested to verify its correctness and study the accuracy and the performance achieved.

Contents

List of Figures

List of Tables

Introduction

Engineering, and in general, any scientific field, including fundamental and applied research, has taken a huge advantage from the introduction of numerical simulations, expanding their limits and horizons and raising their ambition. By means of numerical simulation, many physical or technical problem that in the past needed to be solved by hand or using analogical models, with unsatisfying outcomes, can be solved with the help of a computer, obtaining better results in a shorter time. Many of the largest and innovative works, masterpieces of engineering and science, wouldn't have been as they are, or even possible, without the assist given by this tool.

One of the largest branches of numerical simulation is Computational Fluid Dynamics (CFD), that consists of the resolution of Fluid Dynamics problems by means of computers and suited algorithms. Since the inception of CFD, numerous ways of taking a physical problem into a computer and getting a solution have been developed, generating a large variety of methods for CFD. Their classification takes into account aspects like the Euerian or Lagrangian reference, the use or lack of analytical structures called meshes, the physical equations employed, the approach used for their discretization, and so on. The combination of these properties gives to each method its own properties, and defines the set of applications for which it is more suited. Recently, a class of methods based on a Lagrangian, mesh-free approach has emerged. In particular the Smoothed Particle Hydrodynamics (SPH) method has found a large number of applications in many scientific and technological fields, including oceanography, volcanology, structural engineering, nuclear physics and medicine. In SPH the problem is discretized by means of particles that carry physical information about small volumes of the fluid, and the evolution of the properties of each particle is determined by a discretized form of the physical equations: the Navier–Stokes equations for the motion, thermal equation for the temperature, and so on. The properties of the fluid at any point in space is computed by a smoothed average of the properties of the neighboring particles. Thanks to its nature, SPH easily allows to model and simulate both simple and complex fluids, simplifying the treatment of aspects that can be challenging with more traditional methods: dynamic free surfaces, large deformations, phase transition, fluid/solid interaction and complex geometries. In addition, the most common SPH formulations are favorable for implementation on high-performance parallel computing hardware, such as modern Graphics Processing Units (GPUs), gaining advantage in terms of another aspect of Numerical Simulations, the simulation time.

In this thesis we will study the application of the SPH method to one of the most complex phenomenon in Fluid Dynamics: lava flows. Lava is a complex fluid with a generalized Newtonian rheology that strongly depends on temperature and other physical and chemical aspects of the flow, which may evolve over time. The combination of the free surface, the natural topographies over which it flows, and phenomena such as phase transition and the consequent formation of levees and tunnels, makes the modeling and simulation of lava flows an extremely challenging task for CFD. Studying and simulating lava flows is important in many practical applications, such as the production of scenarios for hazard assessment [23, 13] and the planning of risk mitigation measures, as well as in scientific research to improve our understanding of the physical processes governing the dynamics of lava flow emplacement [18]. For the simulation of lava flows, SPH presents a number of advantages over traditional mesh based methods, such as the implicit tracking of the free-surface and of any internal interface (e.g. solidification fronts) and the lack of restrictions in the geometry of the fluid and of the containing structures.

We will start with an explanation of the SPH method and of the characteristic of lava flows. Then we will introduce GPUSPH, a simulation engine based on the SPH method that runs on GPUs, achieving notable simulation performance over naive implementation of the SPH method. We will extend GPUSPH with methods that improve the application to lava flows, and validate the implementation with appropriate tests. In particular we will introduce a new integration scheme, leading to better performance and more robust simulation in case of high viscosity. At the end we will test the final model, simulating some benchmark test that are common in the field of volcanology, in order to rate the accuracy and performance of the simulations.

Although the work is focused to the simulation of lava flows, the model that has been obtained maintains the identity of a simulator for general purpose CFD simulation, with the advantage of being applicable also to those fluids that share their physical properties with lava.

Chapter 1

Smoothed Particle Hydrodynamics

One of the most important branches of numerical simulation is Computational Fluid Dynamics (CFD), a discipline that concerns the simulation of Fluid Dynamics problems using computers, covering a very important role since very few problems in this field can have an analytical solution. Many ways of solving CFD problems have been introduced over time; we will focus on Smoothed Particle Hydrodynamics (SPH). To understand the essence of this method and highlight what makes it different from the others, we will give an introduction to the world of numerical simulation and the way of classifying the numerous methods that it embraces, showing in what context SPH is placed.

1.1 Numerical simulation

Numerical simulation constitutes a powerful tool at the basis of the modern progress in engineering and in almost every scientific field. Starting from a discrete interpretation of the relevant aspects that characterize a physical problem, numerical simulation allows to find a solution by means of computers, and to get information on the system or the phenomena involved, according to the purpose of the study. Before the possibility to perform numerical simulations, series of approximations and assumptions were very common in the study of a problem, leading to inaccurate and usually incomplete results, thus limiting the progress of science and technology. The introduction of computers has then settled the basis for the development of numerical simulation methods, impressing a strong momentum to this new field and to all the fields that benefit from it.

Numerical Simulation allows to make easier design, study and do preliminary experiments, without the need to perform expensive tests in the laboratory and on the field.

Numerical simulation covers an essential role in hard design [10] or test processes, where prototyping or trial and error is an expensive or dangerous operation. Think for example of the design of a nuclear plant, of an aircraft or a rocket, involving a large amount of money and human lives. Figure 1.1 shows a study performed on the Orion space capsule; On the top is reported the real setup that has been built to test the impact of the capsule with the water, and study the effect of the attack angle; the picture at the bottom shows a numerical simulation of the same problem, run on a computer with the possibility to sim-

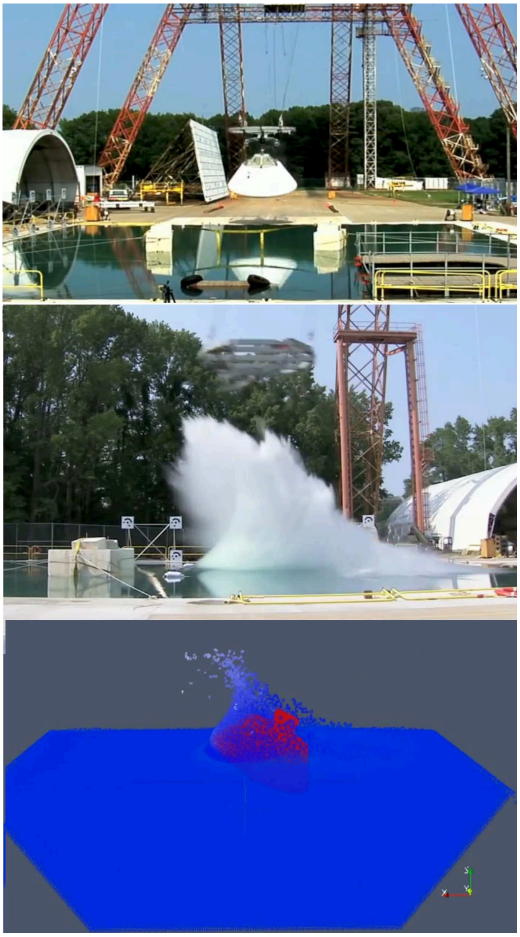


Figure 1.1: Testing the Orion space capsule. The real experiments (top and center, from the NASA Langley Research Center) and a numerical simulation (bottom) with the SPH method (proff. A. Hérault, CNAM, FR, and R. A. Dalrymple, JHU, MD, USA).

ulate numerous cases without building any expensive structure. Moreover in this last case it is possible to rapidly implement a change in the design of the capsule and immediately test its effect, or even to reproduce a failure and study it in absolute safety.

Numerical simulation also allows the extraction of information that is normally hidden or difficult to obtain. Examples are the energy dissipated by a certain chemical reaction, the force exerted by a fluid onto a body, the stresses on a mechanical structure, the velocity field of a fluid on an irregular conduit, and so on.

Numerical simulation is also a fundamental pillar of environmental studies, aimed at forecasting the behavior of a natural system, being it either the climate, a river or an erupting volcano.

1.1.1 Temporal and spatial discretization

When setting a numerical simulation we need to translate the domain into a discrete description, in order to be processed using numerical analysis methods. This discretization must be done both in the time and space domains.

To discretize in time a model describing a dynamical system a simple solution is to replace the derivatives in the differential equations with finite differences.

In space, the geometry needs to be divided into discrete components with techniques that are different for each method. We obtain then a finite set of elements, to associate to so-called interpolation nodes, that are used as representative spatial points to run computations. These nodes usually present a form of connectivity resulting into a grid, or what is called a mesh.

1.1.2 Grid based methods

Physical problems can be studied according to two description frames known as Lagrangian and Eulerian.

- In the Lagrangian frame the observer follows the single portion of the fluid, as it moves in space and time.

- In the Eulerian the observer focuses on a specific point through which the system evolves.

The first gives a material description of the problem, and is commonly used in the so called Finite Element Method (FEM), while the Eulerian description gives a spatial representation of the problem, and is typical of so called Finite Difference Methods (FDM).

These two representation frames originate two families of grid based methods, respectively Lagrangian and Eulerian.

Lagrangian grids

In Lagrangian grid based methods the grid (or mesh) is fixed to the simulated body and follows it during the whole simulation, adapting to all its deformations. In case of complex geometries the grid can be irregular, being more concentrated in the regions of higher distortion to better follow the curvatures. Since these grids are attached to specific parts of the simulated system, they can be used to impose boundary conditions on the surface or any front of a given fluid.

The code to run simulations that involve Lagrangian grids are usually simple and give quite accurate results, in fact this kind of methods has widespread usage.

However, there are difficulties when dealing with very distorted geometries and large deformations of the simulated system, like explosion, fragmentations, turbulence etc. Tracking this kind of phenomenon with a mesh is intrinsically difficult and needs the mesh to be rebuilt to better adapt to the strong ongoing deformations. This kind of methods is therefore preferred when the deformations to be simulated are contained.

Eulerian grids

In Eulerian grid based methods the mesh grid is fixed in space and the simulated system moves through it. Fluxes through cells are evaluated to describe the evolution of the physical quantities.

Here a large deformation does not cause any problem, although other issues still arise. For example, it is difficult to track the properties of a chosen volume of fluid; moreover, everything is discretized by means of fixed cells, by which it is difficult to treat complex geometries, resulting in poor tracking of the free surfaces and other interfaces. Another issue is related to the computational load, since Eulerian methods require a grid that discretizes the whole domain where the fluid can go.

Limits of grid based methods

To summarize all the drawbacks of grid-based methods, they need a complex grid that must be built in relation to the complexity of the simulation domain. Adaptive grids for complex geometries are sometimes hard to be built, both in terms of complexity and computational cost, and sometimes even more than all the other aspects of the method. The problem of the grid construction is even worse in case of strong deformations of the system, very common in fluids, where some re-meshing procedure can be required; this gives advantage in the result but constitutes an annoying and expensive process that could need to be repeated numerous times during a simulation. It is thus apparent that there methods are not suited to deal with problems involving rapid and strong deformations, like impacts or explosions. Moreover, it is easy to realize that they are not appropriate for intrinsically discrete problems, like astrophysical systems, atoms, molecules and so on.

1.1.3 Mesh-free methods

Mesh-free represent a more innovative class of methods that are very promising, expecially in the field of CFD. The idea behind mesh-free methods is to find accurate and stable numerical solution for integral equations or PDEs, with all kind of possible boundary conditions and without using any mesh that provides connectivity to the interpolation nodes, thus overcoming all the issues deriving form the use of such structures, as mentioned above [59].

A large variety of mesh free method have been introduced for the simulation of both solids and fluids. Among these, SPH has been invented in 1977 for astrophysical purposes. Since then, it is currently being developed into several variants that are also embedded in many commercial codes. Other popular mesh-free methods are for example the Diffusive Element Method (DEM), based on a moving Least Square Method (LSM); starting from SPH, another method has been developed, called Reproducing Kernel Particle Method (RKPM), able to improve the accuracy of SPH especially in proximity of boundaries. Refer to [59] for a more complete and detailed discussion on mesh-free methods.

A sub-class of mesh-free methods is constituted by **Mesh-free Particle Methods**, that use a finite set of particles to represent the state of the system and to record its movement. Examples are Molecular Dynamics (MD) introduced in 1957, DEM and SPH. In the following we will focus on the latter.

1.2 The Smoothed Particle Hydrodynamics method

The Smoothed Particle Hydrodynamic method discretizes the simulation domain by means of particles, each representative of a small part of the simulated volume, of which carries information on the physical properties, acting as interpolation nodes. All the particles are free, and move according to the governing conservation equations.

SPH was invented in 1977 by Gingold and Monagan [34], and independently by Lucy [60] to solve astronomical problems in the three-dimensional open space. The collective motion of the astronomical bodies evolved similarly to that of a fluid, so the development of the method was based on the equations of the classical Newtonian hydrodynamics. More recently, the SPH method was thus applied to the solution of fluid Dynamic problems. Other methods for CFD were already existing in a rather well developed state, but they had the drawback of relying on mesh structures, thus suffering of all the disadvantages discussed in 1.1.2. The research of an alternative method, able to face all these difficulties, has determined the adoption of the SPH method in CFD.

At the basis of SPH is the SPH interpolation, that is the way of representing a continuity between the particles, even if they are not constrained to any fixed structure, like a mesh. It relies on a kernel estimation technique. At any point of the domain, the value of a field can be reconstructed interpolating from the neighboring particles. This interpolation is done in a *Smoothed* fashion, using a weighting function, that is intended as a smoothed approximation of the Dirac's Delta distribution in its role of sampling function. This weighing function is called *Smoothing kernel*.

1.2.1 Properties of SPH and applications

After this brief introduction to the concepts of SPH and the position that this method takes in the wide set of methods for numerical simulation, it is possible to describe more into detail the principal characteristics and advantages of SPH, [66]

- Pure advection is treated exactly. For any characteristic the particles are given, once the velocity is specified, the transport of that characteristic by the particle system is exact.

- When simulating more than one material, each described by its own set of particles, the treatment of interface problem is often trivial, as they are implicitly tracked by the region occupied by the particles representing each fluid. Following this concept, the problem of free surface is trivial as well.

- As a particle method it bridges the gap between the continuum and fragmentation in a natural way, constituting the best current method for the study of brittle fracture and subsequent fragmentation in damaged solids [5, 6].

- SPH can manage adaptive resolution, that can depend on position and time. This makes the method very attractive in astrophysics and in the study of many geophysical problems.

- SPH has the computational advantage that the computation is only where the matter is, with a consequent reduction in storage and calculation.

- As for molecular dynamics, with which SPH share a lot of similarities, it is often possible to include complex physics easily [94].

- In many of its formulations, SPH has the benefit of being completely parallelizable, being quite suitable for implementation on massively parallel hardware, such as modern Graphics Processing Units [41].

1.3 SPH discretization of fields

Here we introduce how it is possible to reconstruct the value of a field at any point of the domain, using the SPH interpolation. This is based on the concept of integral representation of a function [66, 59]: consider a field f defined on a domain Ω. By definition of Dirac's delta distribution δ, the value assumed by f at any location $\bar{\mathbf{x}} \in \Omega$ can be expressed, with typical abuse of notation, as

$$f(\bar{\mathbf{x}}) = \int_\Omega f(\mathbf{x})\delta(\mathbf{x} - \bar{\mathbf{x}})\, d\mathbf{x}. \tag{1.1}$$

Consider now a family of functions $W(\cdot, h)$ (*smoothing kernels*), parametrized by a positive parameter h (*smoothing length*) which approximate Dirac's delta, i.e. such that

$$\int_\Omega W(\mathbf{x} - \bar{\mathbf{x}}, h)\, d\mathbf{x} = 1 \tag{1.2}$$

and

$$\lim_{h \to 0} W(\mathbf{x} - \bar{\mathbf{x}}, h) = \delta(\mathbf{x} - \bar{\mathbf{x}}) \tag{1.3}$$

where the limit is to be taken in the sense of distributions.

By substituting the smoothing kernel into (1.1) we get an initial approximation of $f(\bar{\mathbf{x}})$ in the form

$$f(\bar{\mathbf{x}}) \approx \int_\Omega f(\mathbf{x})W(\mathbf{x} - \bar{\mathbf{x}}, h)\, d\mathbf{x} \tag{1.4}$$

We can further approximate the integral with a summation over a finite set of points (*particles*) at positions $\mathbf{x}_1, \ldots, \mathbf{x}_\alpha, \ldots, \mathbf{x}_N$

with volume $V_\alpha = m_\alpha/\rho_\alpha$, where m_α is the particle mass, that is fixed, and ρ_α is the particle density [9]. Then:

$$f(\bar{\mathbf{x}}) \approx \sum_{\alpha=1}^{N} f(\mathbf{x}_\alpha) W(\mathbf{x}_\alpha - \bar{\mathbf{x}}, h) V_\alpha \tag{1.5}$$

where the summation is extended to all particles. To reduce computational complexity, we can additionally choose the smoothing kernels so that they have compact support, which effectively restricts the summations to the particles in a small neighborhood of $\bar{\mathbf{x}}$.

From equations (1.4) and (1.5), we observe that the SPH approximation has two main sources of error

1. approximation of the Dirac's delta with the smoothing kernels;

2. discretization of the domain by means of a finite set of particles.

The first approximation vanishes as $h \to 0$, the second error is controlled by the average inter-particle spacing Δp and vanishes for $\Delta p \to 0$. Additionally, Δp must tend to zero faster than h to ensure consistency for the method [96]. In practical applications, the ratio $h/\Delta p$ is held constant [84] typically in the range $[1.3, 1.5]$.

1.4 First order SPH spatial derivative

The SPH field discretization can be used to reconstruct a field at any point given its values on a set of particles, but the same principle can also be used to discretize spatial gradients. To this end, we will assume further that the smoothing kernels W have radial symmetry, so that they only depend on $r = |\mathbf{x} - \bar{\mathbf{x}}|$, i.e. we assume that $W = W(r, h)$.

Consider the vector field constituted by $\nabla f(\mathbf{x})$. We can approximate its value at any given point by convolution with a smoothing kernel, similarly to (1.4), obtaining

$$\nabla f(\bar{\mathbf{x}}) \approx \int_\Omega \nabla_{\mathbf{x}} f(\mathbf{x}) W(|\mathbf{x} - \bar{\mathbf{x}}|, h)\, d\mathbf{x}. \tag{1.6}$$

Applying Green's theorem gives us

$$\nabla f(\bar{\mathbf{x}}) \approx \int_\Sigma f(\mathbf{x}) W(|\mathbf{x} - \bar{\mathbf{x}}|, h)\mathbf{n}\, d\Sigma + \\ - \int_\Omega f(\mathbf{x}) \nabla_{\mathbf{x}} W(|\mathbf{x} - \bar{\mathbf{x}}|, h)\, d\mathbf{x} \tag{1.7}$$

where $\Sigma = \partial\Omega$ is the boundary of the domain and $\mathbf{n}$ its normal. Given the compact support for W, the first integral in (1.7) is zero if $\bar{\mathbf{x}}$ is far from the boundary; additionally, by symmetry of W, we have $\nabla_{\mathbf{x}} W(|\mathbf{x} - \bar{\mathbf{x}}|, h) = -\nabla_{\bar{\mathbf{x}}} W(|\mathbf{x} - \bar{\mathbf{x}}|, h)$, so that the discretized version can be written as

$$\nabla f(\bar{\mathbf{x}}) \approx \sum_\alpha^N f(\mathbf{x}_\alpha) \nabla_{\bar{\mathbf{x}}} W(|\mathbf{x}_\alpha - \bar{\mathbf{x}}|, h) V_\alpha. \tag{1.8}$$

This expression allows us to compute (an approximation of) the gradient of the field f without having its analytical expression, relying instead on the gradient of the smoothing kernel.

When applied to a particle β, the equation takes the form:

$$\nabla f(\mathbf{x}_\beta) \approx \sum_\alpha^N f(\mathbf{x}_\alpha) \nabla_\beta W_{\alpha\beta} V_\alpha \tag{1.9}$$

where $W_{\alpha\beta} = W\left(\left|\mathbf{x}_\alpha - \mathbf{x}_\beta\right|, h\right)$. This can be symmetrized (thus helping preserve conservation properties of the analytical equations) by subtracting the gradient of the function identically equal to $f(\mathbf{x}_\beta)$, obtaining:

$$\nabla f(\mathbf{x}_\beta) \approx \sum_\alpha^N f_{\alpha\beta} \nabla_\beta W_{\alpha\beta} V_\alpha \tag{1.10}$$

where $f_{\alpha\beta} = f(\mathbf{x}_\alpha) - f(\mathbf{x}_\beta)$. Finally, we observe that due to the symmetry of W, its gradient can be written as:

$$\nabla_\beta W_{\alpha\beta} = \frac{\mathbf{x}_{\alpha\beta}}{|\mathbf{x}_{\alpha\beta}|} \left.\frac{\partial W(r,h)}{\partial r}\right|_{r=|\mathbf{x}_{\alpha\beta}|} \tag{1.11}$$

where $\mathbf{x}_{\alpha\beta} = \mathbf{x}_\alpha - \mathbf{x}_\beta$. It is therefore convenient to choose a kernel such that

$$F(r) = \frac{1}{r}\frac{\partial W}{\partial r} \tag{1.12}$$

has an analytical expression, and given $F_{\alpha\beta} = F\left(|\mathbf{x}_{\alpha\beta}|\right)$, we can write $\nabla_\beta W_{\alpha\beta} = \mathbf{x}_{\alpha\beta} F_{\alpha\beta}$.

1.5 Second order SPH spatial derivative

An immediate way to obtain an SPH discretization of second order derivatives is to iterate two consecutive times the procedure that gives the discretization of the first order derivative, explained in section 1.4. However, the discretization obtained in this way has been shown to be very noisy and sensitive to particles disorder [11]. An alternative approach has been presented by [11] and [65] and gives an approximation of the second order derivative using only the first derivative of the kernel. In the following we derive it for a field f, defined in Ω, and, following [50], we consider the specific case where Ω has dimension three.

To begin, let us consider a Taylor series approximation of $f(\mathbf{x}_\alpha)$ around $\mathbf{x}_\alpha = \mathbf{x}_\beta$,

$$
\begin{aligned}
f(\mathbf{x}_\alpha) &= f(\mathbf{x}_\beta) + \nabla f\Big|_{\mathbf{x}_\beta} (\mathbf{x}_\alpha - \mathbf{x}_\beta) + \\
&+ \frac{1}{2}\frac{\partial^2 f}{\partial \mathbf{x}_s \partial \mathbf{x}_k}\Big|_{\mathbf{x}_\beta} (\mathbf{x}_\alpha - \mathbf{x}_\beta)_s (\mathbf{x}_\alpha - \mathbf{x}_\beta)_k + \mathcal{O}(\mathbf{x}_\alpha - \mathbf{x}_\beta)^3
\end{aligned}
\tag{1.13}
$$

We neglect the terms of third and higher order and we multiply by

$$
\frac{\mathbf{x}_{\beta\alpha} \nabla_\beta W(\mathbf{x}_{\beta\alpha})}{|\mathbf{x}_{\beta\alpha}|^2} \tag{1.14}
$$

where $\mathbf{x}_{\beta\alpha} = \mathbf{x}_\alpha - \mathbf{x}_\beta$ and $W_{\beta\alpha} = W(\mathbf{x}_\alpha - \mathbf{x}_\beta)$, and we integrate over all Ω. We note that for the first order term we have

$$\int \mathbf{x}_{\beta\alpha} \frac{\mathbf{x}_{\beta\alpha} \nabla_\beta W_{\beta\alpha}}{|\mathbf{x}_{\beta\alpha}|^2} d^3\mathbf{x}_\alpha = 0 \tag{1.15}$$

thanks to the spherical symmetry of the kernel, and for the second order term

$$\int (\mathbf{x}_{\beta\alpha})_s (\mathbf{x}_{\beta\alpha})_k \frac{\mathbf{x}_{\beta\alpha} \nabla_\beta W_{\beta\alpha}}{|\mathbf{x}_{\beta\alpha}|^2} d^3\mathbf{x}_\alpha = \delta_{sk} \tag{1.16}$$

Eventually, we obtain

$$\nabla^2 f \bigg|_{\mathbf{x}_\beta} \approx 2 \int \frac{f(\mathbf{x}_\beta) - f(\mathbf{x}_\alpha)}{|\mathbf{x}_{\beta\alpha}|^2} \mathbf{x}_{\beta\alpha} \nabla_\beta W_{\beta\alpha} d^3\mathbf{x}_\alpha \tag{1.17}$$

To obtain the SPH discretization we replace the integral with a sum over α and we substitute $d^3\mathbf{x}_\alpha$ with the SPH discrete form m_α/ρ_α, obtaining:

$$\langle \nabla^2 f \rangle \approx 2 \sum_\alpha \frac{m_\alpha}{\rho_\alpha} \frac{f_\beta - f_\alpha}{|\mathbf{x}_{\beta\alpha}|^2} \mathbf{x}_{\beta\alpha} \nabla_\beta W_{\beta\alpha} \tag{1.18}$$

where $f_\beta = f(\mathbf{x}_\beta)$ and $f_\alpha = f(\mathbf{x}_\alpha)$. An important property of this expression is to be zero when f is constant.

In case we are dealing with a second order derivative expressed in the form $\nabla \cdot (Q \nabla f)$, where Q may show a spatial variation, it is not difficult to show [11] that

$$\nabla \cdot (Q \nabla f) \approx -2 \int \frac{[Q_\beta + Q_\alpha][f_\alpha - f_\beta]}{|\mathbf{x}_{\beta\alpha}|^2} \mathbf{x}_{\beta\alpha} \nabla_\beta W_{\beta\alpha} d^3\mathbf{x}_\alpha \tag{1.19}$$

and then, taking the associate SPH discretization, we get

$$\langle \nabla \cdot (Q \nabla f) \rangle \approx \sum_{\alpha} \frac{m_\alpha}{\rho_\alpha} \frac{[Q_\beta + Q_\alpha][f_\alpha - f_\beta]}{|\mathbf{x}_{\beta\alpha}|^2} \mathbf{x}_{\beta\alpha} \nabla_\beta W_{\beta\alpha}. \quad (1.20)$$

Recalling (1.12), (1.20) can be finally rewritten as

$$\langle \nabla \cdot (Q \nabla f) \rangle \approx \sum_{\alpha} \frac{m_\alpha}{\rho_\alpha} [Q_\beta + Q_\alpha][f_\alpha - f_\beta] F_{\beta\alpha}. \quad (1.21)$$

In some sense this expression is the SPH discretization of a finite difference derivative.

1.6 Smoothing kernels

At the basis of the SPH method there is problem of approximate a function starting form a set of scattered points and without using a predefined grid of points [59]. This is accomplished in an integral way [58], by means of *smoothing kernel functions.* We have seen how smoothing kernels are used to interpolate over particles in 1.3. The choice of a smoothing kernel function deeply affects the results of the simulation since it gives the pattern and consistency of the function approximation.

The conditions that Smoothing Kernel functions must obey are [59]:

1. The smoothing kernel function must be normalized over its support domain, e.g

$$\int_\Omega W(r, h) dx = 1 \quad (1.22)$$

2. It should be defined on a compact support, i.e.

$$W(r,h) = 0 \quad \text{for } r > kh \tag{1.23}$$

where k specifies how the kernel is spread.

3. It must be positive

$$W(r,h) \geq 0 \tag{1.24}$$

4. It should be monotonically decreasing with the increase of distance away from the particle

5. It should approximate the Dirac's Delta function as $h \to 0$ approaches zero

$$\lim_{h\to 0} W(r,h) = \delta(r,h) \tag{1.25}$$

6. It should be an even function

7. It should be sufficiently smooth

Any function that satisfies these conditions can be used as smoothing kernel. Some functions have been found and are used as standard smoothing kernels in SPH. Here we give the expression for the two smoothing kernel that have been used in the work presented in this thesis.

1.6.1 Gaussian kernel

The Gaussian kernel is a quite smooth function, as well as its higher order derivatives. It is very stable and accurate, especially in case of disordered particles.

We define a cutoff radius δ, generally $\delta \geq 3$

$$W(q) = \frac{1}{C_{W,d}} \left(e^{-q^2} - e_\delta \right) \tag{1.26}$$

$$F(q) = -\frac{1}{C_{F,d}} e^{-q^2} \tag{1.27}$$

where F is introduced in (1.12), with $0 \leq q \leq \delta$ and $e_\delta = e^{-\delta^2}$, and the normalization constants are

$$C_{W,2} = \pi h^2 \left(1 - e_\delta (1 + \delta^2)\right) \tag{1.28}$$

$$C_{W,3} = \pi h^3 \left(\sqrt{\pi} \mathrm{Erf}(\delta) - \frac{2}{3} e_\delta \delta (3 + 2\delta^2) \right) \tag{1.29}$$

where $\mathrm{Erf}(\cdot)$ is the error function,

$$C_{F,d} = h^2 \frac{C_{W,d}}{2} \tag{1.30}$$

The radius of the kernel specifies how large is the region where the neighboring particles are considered when computing the summations that we have seen in 1.3 and 1.4. Of course the interpolation is improved as this area is increased, but this affects sensibly the computational time, since the number of particles to iterate over increases. Gaussian kernel gives good results in terms of quality of the interpolation, but its radius, usually $\delta = 3$, usually results in a quite heavy computational load.

1.6.2 Wendland Kernel

Wendland smoothing kernel is a radius two function, then leading to a lighter computational load with respect of Gaussian function.

The representation of the Wendland kernel is quite easy: [87], defined as $W(r,h) = \tilde{W}(r/h)$ and $F(r,h) = \tilde{F}(r/h)$ with

$$\tilde{W}(q) = C_W(2q+1)(1-q/2)^4 \qquad 0 \le q \le 2 \tag{1.31}$$

$$\tilde{F}(q) = C_F(q-2)^3 \tag{1.32}$$

where, working in three dimensions, $C_W = 21/16\pi h^3$ and $C_F = 5C_W/(8h^2)$.

1.7 Running the SPH method

A SPH simulation basically relies on an *Initialization* phase and then on two fundamental processes that alternate iteratively: *Force computation* and *Integration.* Though their meaning could be immediate to understand, here follows a brief description of these two processes, in order to define their content for a clearer reference in the following of this work.

1. The initialization phase creates the set of particles according to a geometrical description of the domain and other aspects, like the adopted boundary model, assigning to each particle the initial values for the state variables.

2. The computation of the forces consists on computing the derivatives of the sate variables of the system, for example the acceleration, the density derivative, the temperature derivative, and so on. Practically, it is accomplished computing the expression of the governing equations, after having been discretized according to the SPH method. As an example, in section 2 is shown a set of discretized governing equations used in fluid dynamics. The computation of the forces then takes into account all the aspects relative to boundary conditions, interactions, and some possible forms of controls, that are aimed at acting on the behavior of the system. In this phase, for each particle are performed several iterations over other particles of the domain, as required by most of aspects that have a role in the computation (For example the SPH discretization of the spatial derivatives, equations (1.10) and (1.21), and the dummy boundary conditions, equations (2.31) and (2.32)).

3. The integration phase always follows the computation of the forces, and consists of the application of an integration algorithm to compute the values of the state variables from their derivatives, obtained in the forces computation phase.

While the initialization phase is executes only once at the beginning of the simulation, the remaining two phases iteratively alternates as long as the simulation evolves.

Chapter 2

SPH and Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) is the branch of numerical analysis aimed at solving problems of Fluid Dynamics by means of computer. Recalling our introduction done in 1.1, CFD embraces all the numerical simulations involving fluids. We have seen that SPH presents all the characteristics that make it suited to work with fluids, and in fact, it nowadays represent a very powerful and promising method in the field of CFD. Here we will use the basic concept given in chapter 1 about the SPH method to see how it can be applied to the field of Fluid Dynamics.

2.1 Fluid governing equations

The governing equations of a model describe how some state variables vary in relation to the value of some other quantities. At the base of the SPH method there are the following two equations of conservation:

- conservation of mass
- conservation of momentum

then other equations can be added in order to study other physical quantities, for example the energy conservation or the thermal model. In the following we introduce the governing equations that have been used for the simulations that will be discussed in this thesis.

2.1.1 Mass continuity equation

Mass continuity without a source can be described by the equation

$$\frac{D\rho}{Dt} = -\rho\nabla \cdot \mathbf{u} \tag{2.1}$$

where ρ is the density, $\mathbf{u}$ the velocity and D/Dt the total derivative (sometimes also called the Lagrangian or material derivative), i.e. the operator:

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla \tag{2.2}$$

.

From equation (2.1), we can deduce that if the flow is incompressible (i.e. if density does not change, $D\rho/Dt = 0$), then we must have

$$\nabla \cdot \mathbf{u} = 0. \tag{2.3}$$

2.1.2 Momentum conservation equations

The conservation of momentum is expressed by the Momentum Navier–Stokes equation. The latter is a particular form of the Cauchy Momentum equation:

$$\rho \frac{D\mathbf{u}}{Dt} = \nabla \cdot \sigma + \mathbf{G} \tag{2.4}$$

σ is the Cauchy stress tensor and $\mathbf{G}$ any external force.

$$\sigma = \tau - P\mathbf{I} \tag{2.5}$$

where P is the mechanical pressure, that is intended as $P = -\frac{1}{3}\mathrm{Tr}(\sigma)$, and τ is a zero–trace tensor called *shear stress tensor*. Equation (2.4) can then be rewritten as

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla P + \nabla \cdot \tau + \mathbf{G} \tag{2.6}$$

The viscous constitutive equation

The equations introduced so far are insufficient to mathematically study the problem of the motion of a fluid. To make the problem solvable we need a *constitutive relation* that links the kinematic state of the fluid to its stress state. This relation depends on the nature of the fluid, and is described by means of *constitutive equations*.

Compressible fluids Let us consider the linear stress constitutive equation, that links the strain rate to the shear stress as follows:

$$\tau = \lambda(\nabla \cdot \mathbf{u})\mathbf{I} + 2\mu\dot{\gamma} \tag{2.7}$$

The two scalars λ and μ are the Lamé parameters, called *bulk viscosity* and *dynamic viscosity*, respectively. The strain rate, $\dot{\gamma}$, can be expressed as a function of the velocity gradient, i.e.

$$\dot{\gamma} = \frac{1}{2}\left[\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right] \tag{2.8}$$

and thus the 2.7 may be rewritten as

$$\tau = \lambda(\nabla \cdot \mathbf{u})\mathbf{I} + \mu\left[\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right] \tag{2.9}$$

The trace of the stress tensor in three dimensions is

$$tr(\tau) = (3\lambda + 2\mu)\nabla \cdot \mathbf{u} \tag{2.10}$$

Decomposing the stress tensor into its isotropic and deviatoric parts, we get

$$\tau = \left(\lambda + \frac{2}{3}\mu\right)(\nabla \cdot \mathbf{u})\mathbf{I} + \mu\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T - \frac{2}{3}(\nabla \cdot \mathbf{u})\mathbf{I}\right) \tag{2.11}$$

Since τ is a zero trace tensor, we make the likely assumption of $\lambda = -2/3\mu$, that in the (2.10) makes the second member equal to zero. This can be done since we are not dealing with sound waves and shock waves.

The (2.11) then becomes the linear stress constitutive equation, used in hydraulics for compressible fluids

$$\tau = \mu \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T - \frac{2}{3}(\nabla \cdot \mathbf{u})\mathbf{I} \right) \tag{2.12}$$

If we assume μ constant in space, and, considering the identities

$$\nabla \cdot (\nabla \mathbf{u}) = \nabla^2 \mathbf{u} \tag{2.13}$$

and

$$\nabla \cdot (\nabla \mathbf{u})^T = \nabla(\nabla \cdot \mathbf{u}), \tag{2.14}$$

we substitute (2.12) in (2.6),we can write the Navier-Stokes momentum equation for compressible fluids:

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla P + \mu \nabla^2 \mathbf{u} + \frac{1}{3}\mu \nabla(\nabla \cdot \mathbf{u}) + \mathbf{G} \tag{2.15}$$

Incompressible flow

The discussion above about compressible fluids can be repeated for the case of incompressible fluid, using the incompressibility condition (2.3). Applying the latter to (2.15), we obtain the incompressible Navier-Stokes momentum equation, i.e.

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla P + \mu \nabla^2 \mathbf{u} + \mathbf{G} \tag{2.16}$$

Non homogeneous viscosity In case the viscosity can variate over the space, when using the constitutive equation (2.12) in the Navier-Stokes equation (2.4), the viscosity coefficient cannot be moved outside the divergence operator to obtain the Laplacian operator in (2.16). A more suited form of the incompressible Navier-Stokes equation is

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla P + \nabla \cdot (\mu \nabla \mathbf{u}) + \mathbf{G} \tag{2.17}$$

This expression includes an approximated form of the viscous term. In fact, when developing the Navier-Stokes equation the (2.14) cannot be used. As a consequence, the application of the incompressibility condition (2.3) leaves a residual part, coming from $(\nabla \mathbf{u})^T$. This creates an error that can be neglected when the stresses on the transversal direction of the flow can be neglected, that is the case of quiet flows, like lava flows.

Generalized Newtonian fluids

A Newtonian fluid is characterized by a direct proportionality between the shear stress τ and the strain rate $\dot{\gamma}$, and the ratio is the dynamic viscosity μ, which may depend on local physical and chemical properties of the fluid (such as the temperature, density, composition), but not on the shear stress or strain rate themselves. A fluid where the relationship between stress and strain rate is not linear is known as non-Newtonian fluid, and many fluids of great interest in industrial applications as well as geophysics are indeed non-Newtonian.

Among non-Newtonian fluids, there is a large subset of fluids (called generalized Newtonian fluids), for which it is still possible to use the same form of the Navier–Stokes equations, provided that the constant μ is replaced by an *effective viscosity* function $\mu_{\text{eff}} = \mu_{\text{eff}}(\dot{\gamma})$ such that $\tau = \mu_{\text{eff}}(\dot{\gamma})\dot{\gamma}$.

For our model we consider the Herschel–Bulkley rheology, a generalized Newtonian rheology characterized by a *yield strength* τ_0, a *power law exponent* n and a *consistency index* k such that $\dot{\gamma} = 0 \iff |\tau| \leq \tau_0$, and

$$|\tau| = \tau_0 + k\,|\dot{\gamma}|^n \tag{2.18}$$

otherwise. As special cases of the Herschel–Bulkley rheology we obtain (see figure 2.1)

- Newtonian fluids for $\tau_0 = 0, n = 1$ (in which case k is the viscosity);
- power-law fluids for $\tau_0 = 0, n \neq 1$ (called *dilatant* if $n > 1$ and *pseudo-plastic* if $n < 1$);
- Bingham fluids for $\tau_0 \neq 0, n = 1$.

For an Herschel–Bulkley fluid, the effective viscosity can be written [95] as

$$\mu_{\text{eff}} = \mu_{\text{eff}}(\dot{\gamma}) = \frac{\tau_0}{|\dot{\gamma}|} + k\,|\dot{\gamma}|^{n-1} \tag{2.19}$$

for $|\tau| > \tau_0$ and with $|\dot{\gamma}| = \sqrt{(\dot{\gamma} : \dot{\gamma})/2}$,

$$\dot{\gamma} : \dot{\gamma} = \gamma_{xx}^2 + \gamma_{yy}^2 + \gamma_{zz}^2 + 2(\gamma_{xy}^2 + \gamma_{xz}^2 + \gamma_{yz}^2) \tag{2.20}$$

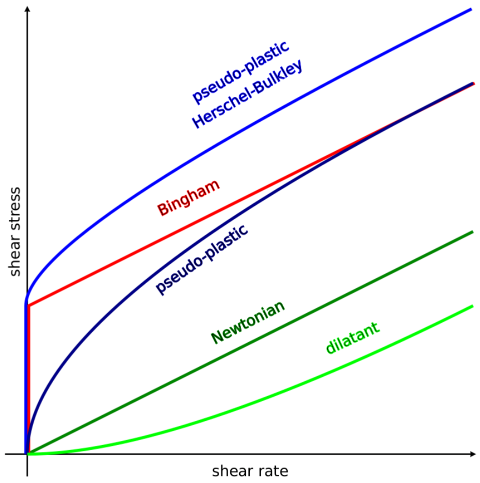


Figure 2.1: Example trends of the main rheological laws.

In common applications, the discontinuity in the stress/strain relationship when the stress is equal to the yeld strenght causes instabilities, then some regularized versions of the caonstitutive law are often used. One of the most common regularized law is proposed by [95], and will be introduced in 6.1.

2.1.3 Thermal equation

Finally, the thermal evolution is described by the heat equation

$$\rho c_p \frac{DT}{Dt} = \nabla(\kappa \nabla T) \tag{2.21}$$

where T is the temperature, c_p the specific heat at constant pressure, and κ the thermal conductivity. We model thermal radiation as described in [8], and phase transition following [8, 42, 67]. The latter is limited to the thermal effects (and particularly the constant temperature during phase transition): the dynamics of solid particles is currently the same as for fluid particles, and the aggregation of solid particles into larger bodies is not modeled.

2.2 Closure of the equations

Equations (2.1) and (2.17), obtained for incompressible fluid, constitute and open system, that need to be closed by specifying a way to determine the pressure. There are usually two ways of closing them, leading to two different SPH formulations, namely, Implicit Incompressible SPH (IISPH) and Weakly compressible SPH (WCSPH).

2.2.1 Incompressible SPH

The incompressible SPH comes from the discretization of the incompressible Navier-Stokes equations, closed by means of the Pressure Poisson equation.

For the sake of simplicity let us consider the case of uniform viscosity. The Pressure Poisson equation can be obtained starting from (2.16) and taking the divergence of both member, thus obtaining:

$$\nabla \cdot \frac{D\mathbf{u}}{Dt} = \nabla \cdot (-\nabla P + \mu \nabla^2 \mathbf{u} + \mathbf{G}) \tag{2.22}$$

If we focus on the left hand side of this equality, we can write

$$\begin{aligned}\nabla \cdot \frac{D\mathbf{u}}{Dt} = \nabla \cdot \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} \right) = \\ = \frac{\partial}{\partial t}(\nabla \cdot \mathbf{u}) + \nabla(\mathbf{u} \cdot \nabla)\mathbf{u} = \nabla(\mathbf{u} \cdot \nabla)\mathbf{u}\end{aligned} \tag{2.23}$$

where (2.3) has been used. Analogously, for the right hand side we write

$$\begin{aligned}\nabla \cdot (-\nabla P + \mu \nabla^2 \mathbf{u} + \mathbf{G}) = \\ = -\nabla^2 P + \mu \nabla^2 (\nabla \cdot \mathbf{u}) + \nabla \cdot \mathbf{G} = -\nabla^2 P + \nabla \cdot \mathbf{G}\end{aligned} \tag{2.24}$$

Eventually, equating the final form of (2.23) and (2.24) and opportunely rearranging, we get the Poisson equation

$$\nabla^2 P = \nabla \cdot [\mathbf{G} - (\mathbf{u} \cdot \nabla)\mathbf{u}] \tag{2.25}$$

The closed set of equations obtained in this way constitute non homogeneous linear system that then needs to be solved in an implicit fashion. For this reason, the method is also known as Implicit Incompressible SPH, IISPH.

2.2.2 Weakly Compressible SPH: WCSPH

Equations (2.16) and (2.17) are obtained under the assumption of incompressibility, and can be used to model the dynamic of an incompressible fluid if treated as discussed in 2.2.1, where the set of governing equations is closed by means of a Poisson equation.

An alternative approach consists of using a state equation instead of the Poisson equation, so that the pressure can be obtained directly from the density. An example of state equation is the Cole's [17, 4] law:

$$P(\rho) = c_0^2 \frac{\rho_0}{\xi} \left(\left(\frac{\rho}{\rho_0} \right)^{\xi} - 1 \right) \tag{2.26}$$

where ρ_0 is the at-rest density, ξ is the polytropic constant and c_0 is the speed of sound, that is commonly used to model compressible gases. It can be seen that the speed of sound constitutes an important connection between the density and the pressure. A larger value of c_0 makes the fluid more incompressible. A regime of *weak-compressibility* is achieved if c_0 is at least an order of magnitude higher than the maximum velocity experienced during the flow [66].

2.3 SPH Discretization

We can apply the SPH discretization to the continuity equation (2.1), obtaining

$$\frac{D\rho_\beta}{Dt} = \sum_\alpha m_\alpha \mathbf{u}_{\alpha\beta} \nabla_\beta W_{\alpha\beta}. \tag{2.27}$$

For the momentum ((2.16) or (2.17)) and thermal (2.21) equations, we additionally need a discretization for the Laplacian, for which we follow [11], [69] and [16]. The momentum equation then takes the form:

$$\begin{aligned}\frac{D\mathbf{u}_\beta}{Dt} = -\sum_\alpha \left(\frac{P_\alpha}{\rho_\alpha^2} + \frac{P_\beta}{\rho_\beta^2}\right) F_{\alpha\beta} m_\alpha \mathbf{x}_{\alpha\beta} + \\ + \sum_\alpha \frac{2\mu_{\alpha\beta}}{\rho_\alpha \rho_\beta} F_{\alpha\beta} m_\alpha \mathbf{u}_{\alpha\beta} + \mathbf{g}\end{aligned} \tag{2.28}$$

where $\mu_{\alpha\beta}$ is the harmonic mean of μ_α and μ_β, and $\mathbf{g} = \mathbf{G}/\rho$, while the thermal equation becomes:

$$\frac{DT_\beta}{Dt} = \frac{1}{c_p} \sum_\alpha \frac{\kappa_{\alpha\beta} T_{\alpha\beta}}{\rho_\alpha \rho_\beta} F_{\alpha\beta} \tag{2.29}$$

where $\kappa_{\alpha\beta}$ is the harmonic mean of κ_α and κ_β.

2.4 Kinematic boundary conditions

The role of SPH simulations is to solve physical problems by handling the corresponding Partial Differential Equations. This description of the problem requires some *boundary conditions* to be known in order to find a unique solution; this consists to assign the value of the state variables of the system over the boundary of the domain. In this section we are going to see some boundary conditions that are typical in the field of CFD, then next section will treat their implementation in an SPH context. We will start from the mechanical boundary conditions, i.e. those assigning a value to the velocity and the pressure, then we will discuss those relative to the thermal model.

2.4.1 Free-slip and no slip condition

From a mechanical point of view, at the interfaces between fluid and either walls and ground we impose a no-slip condition; this is obtained prescribing normal and tangential velocity along the analytical boundary as

$$\begin{aligned} u_n(t) &= 0 \\ u_t(t) &= v_w \end{aligned} \tag{2.30}$$

where v_w is any physical sliding velocity of the wall (in our examples, we will always have $v_w = 0$).

The opposite behavior can be obtained with a free slip condition. Conceptually, it means that the fluid is free to slip over the boundary without being held by the wall. More technically, this condition ensures a null tangential shear stress along the boundary.

SPH implementation

The SPH implementation of the mechanical boundary conditions that we have seen above has been done in several ways, giving as result a variety of boundary models, each different from the others for their behavior and geometrical implementation. Here, we are going to introduce only two boundary models, the ones that have been used in the work reported in this thesis, that are *Dynamic* [21] and *Dummy* [1] boundary models. In both models, solid boundaries are discretized some layers of particles, as many as necessary to cover a full influence radius of the smoothing kernel, rounding up. For example, for a smoothing kernel of radius 2 and a smoothing factor of 1.3, three layers of particles are necessary.

Dynamic Boundary model With dynamic boundaries, the boundary particles have a prescribed velocity, and their pressure evolves according to the standard continuity equation to better enforce the no-penetration condition.

A known issue that affects the dynamic boundary model is that the evolution of the density is essentially controlled by the motion of the fluid above it, which introduces some spurious effects near wet/dry zones. In particular, the transition from a dry to wet states leads to an increase in density that tends to ‘lift’ the fluid from the boundary, and conversely, fluid impinging on a wall and subsequently flowing away leads to a decrease in density that can reduce the stability of the simulation. This has implications both in the implementation of open boundaries, as shown momentarily, and in the accuracy of the results, particularly for more viscous fluids.

Another drawback of dummy boundary is constituted by fluid penetration, that can occur when the fluid rapidly approaches the boundary.

Dummy Boundary With dummy boundaries, the velocity of the boundary particles, is obtained by adding to the wall velocity $\mathbf{u}_w$, the opposite of the Shepard-averaged velocity of the neighboring fluid

$$\mathbf{u}_\beta = \mathbf{u}_w - \frac{\sum\limits_{\alpha\in\mathcal{F}} \mathbf{u}_\alpha W_{\alpha\beta}}{\sum\limits_{\alpha\in\mathcal{F}} W_{\alpha\beta}}, \tag{2.31}$$

(where $\mathcal{F}$ represents the set of fluid particles) while the density is computed to achieve a pressure that matches the Shepard-averaged pressure of the neighboring fluid

$$P_\beta = \frac{\sum\limits_{\alpha\in\mathcal{F}} P_\alpha W_{\beta\alpha} + \mathbf{g}\sum\limits_{\alpha\in\mathcal{F}} \rho_\alpha \mathbf{x}_{\beta\alpha} W_{\beta\alpha}}{\sum\limits_{\alpha\in\mathcal{F}} W_{\beta\alpha}}. \tag{2.32}$$

2.4.2 Periodic boundary conditions

Periodic boundary conditions are used to emulate infinitely extended domains. The actual implemented domain constitutes the base unit of a periodic domain, and is usually called *unit* *cell.* GPUSPH allows to assign the periodicity to each spatial dimension, individually, allowing to set a uni-, bi- or three- dimensional periodicity. When a particle crosses a periodic boundary it disappears and reappears on the opposite side of the unit cell, maintaining the same dynamical and thermal state.

In GPUSPH, the periodicity is acted by shifting the particle by the unit cell size every time it crosses the boundary. Having periodic boundary also implies that the neighbors search is performed also on the cells lying on the opposite face of the domain.

2.4.3 Open boundaries

Open boundaries (OB) allow us to extend the simulation domain to a virtual region that is not implemented, but that can “exchange” matter and information with the actual implemented environment. For example, if we want to model a filling tank, we don’t need to include in the simulation domain the reservoir where the fluid is coming from, but we can model just the inlet and create the new fluid as it flows inside the domain, assigning a velocity that emulates that of the real problem. Analogously, if we want to model a sink. OB can be useful if our experiment interests only a part of the whole fluid body and we need to simulate just that portion; for example, something that is floating in the ocean: we model a cube of water and we recreate the motion of the sea waves or any tide. Behind this apparent working principle there are other aspects to consider, that is the *continuity* between the internal simulated domain and the external virtual domain. To recreate this continuity, in addition to model the flow of mass across open boundary, one should also care about transmission of information, i.e. the waves propagation. This is what makes a boundary actually open: the information that reaches an open boundary needs to propagate through it and leave the domain. To achieve this, appropriate conditions for the velocity and pressure must be set, ensuring that the boundary is absorbing the mass and wave.

At a (kinematic) open boundary it is possible to impose either the velocity, or the pressure. The velocity is generally prescribed when it's necessary to impose a given flow rate, most frequently at the inlet. Conversely, pressure is most frequently prescribed at the outlet, where the fluid is free to flow out of the domain according to its own developed velocity field. In both cases, the missing information (pressure for the inlet, velocity for the outlet) is computed by the use of appropriate Generalized Riemann Invariants that are computed from the prescribed boundary conditions and the extrapolated information from the inside of the domain, to avoid spurious reflections.

SPH implementation

Since the management of open boundary conditions is an essentially Eulerian method, implementing it in a Lagrangian context requires the adoption of quite sophisticated tools [56, 62, 28]. We applied the approach proposed by [28], rearranged for the 3D case and for our boundary models, that relies on the Generalized Riemann Invariants (GRI) and using characteristic waves to model the discontinuity between the interior (the fluid) and exterior (the inlet and the virtual domain) state.

The inlet is modelled as a three-dimensional buffer of particles obtained extruding the inlet two-dimensional surface. These buffer particles move according to the imposed velocities. When fluid particles leave the inlet, a new particle is generated at the other end, keeping the buffer full. The depth of the buffer depends on the boundary model. When using dummy boundaries, the inlet depth is chosen to match the thickness of solid walls. However, in the dynamic boundary case this choice leads to large decreases in density for the boundary surrounding the inlet, since they observe fluid flowing away without additional incoming fluid. To avoid this effect, the depth of the inlet is doubled in the dynamic boundary case.

The GRI correction is then applied. In the following we consider only the case with fixed velocity, that is what we will use in this thesis. The dual problem, with fixed density, is described in [28]. The GRI will therefore be used to calculate the external state pressure from the internal state. However the type of discontinuity needs to be defined. Let us call $u_{n,ext}$ the component of the inlet fixed velocity in the direction normal to the inlet surface; similarly, $u_{n,int}$ is the component in the direction normal to the inlet surface, of the fluid velocity in the internal domain. We classify the discontinuity in

- shock wave, if $u_{n,ext} > u_{n,int}$; in this case we impose the density from the pressure evaluated using the Rankine-Hugoniot relationships:

$$P_{ext} = P_{int} + \rho_{int} u_{n,int}(u_{n,int} - u_{n,ext}) \tag{2.33}$$

- expansion wave, if $u_{n,ext} \leq u_{n,int}$; then the corresponding Riemann invariant needs to be used:

$$u_{n,ext} - \psi(\rho_{ext}) = u_{n,int} - \psi(\rho_{int}) \tag{2.34}$$

with

$$\psi = \frac{2c_0}{\xi - 1}\left(\frac{\rho}{\rho_0}\right)^{\left(\frac{\xi-1}{2}\right)} \tag{2.35}$$

To probe the internal state and compute $\mathbf{u}_{int}$ and $\rho_i nt$, we do an interpolation of the velocity in the region near to the inlet. We set a grid of marker particles attached to the inlet, having the same spacing as the domain discretization, and we perform a Shepard interpolation of the velocity and pressure of their neighboring fluid particles, just like in the dummy boundary case (equations (2.31) and (2.32)). The computed values for velocity and pressure are then extrapolated from the marker particles to the fluid particles in the inlet itself.

Hydrostatic compensation

The simulation of volcanic eruptions requires to use of vertical inlet conditions; this frequently leads to the formation of a high column of fluid on top of the inlet[91]. The constant pressure due to this column must be supported by the inlet, to prevent the particles from sinking back into the inlet. Applying the conditions seen so far, the continuity is only between the internal state and the inner layer of the buffer. The pressure is then

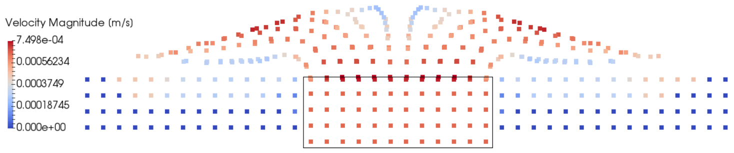


Figure 2.2: Section of a vertical inlet injecting a viscous fluid in an empty region. The rectangular box delimitates the buffer region.

projected over the buffer thickness. We need to correct this condition creating a continuity in the trend of hydrostatic pressure field. This is done adding an hydrostatic term, $P_h(d) = \rho g d$, with g the modulus of the gravity vector and the d the depth of the particle with respect to the inlet region, considered along the direction of the gravity vector.

2.5 Thermal boundary conditions

In section 2.4 we have seen the mechanical boundary conditions followed by their implementations in the SPH context. Since the main structure for the boundaries is usually defined when implementing the mechanical conditions, in this part we will see the boundary conditions and their implementation in the boundary model with the geometrical structure prescribed by the dynamic or dummy models (see 2.4.1).

2.5.1 Thermal source

The thermal source is one of the conceptually simplest thermal boundary conditions; let us indicate the boundary region with $\mathbf{S}$, chosen a temperature T_s for the source, a thermal source ensures that

$$T(\mathbf{x}, t) = T_s \quad \forall \mathbf{x} \in \mathbf{S}, \forall t \in \Re \tag{2.36}$$

and leaves free any heat exchange.

This boundary condition is implemented by fixing the temperature of the boundary particles to T_s, i.e. their temperature is not integrated, and allowing heat exchange between fluid and boundary particles according to the heat diffusion equation (2.21).

2.5.2 Adiabatic boundaries

Adiabatic boundary conditions are obtained when

$$\nabla T \cdot \mathbf{n} = 0 \tag{2.37}$$

with $\mathbf{n}$ the unit vector normal to the boundary region.

To implement this condition on SPH let us recall the discretization of the gradient of a field, expressed by equation (1.10) and let us apply it to the temperature field. If we denote by β a particle belonging to an adiabatic boundary, $\mathbf{x}_\beta$ its position and T_β its temperature, and using (1.12), we can write

$$\nabla T(\mathbf{x}_\beta) = \sum_\alpha (T_\alpha - T_\beta)(\mathbf{x}_\alpha - \mathbf{x}_\beta) \cdot \mathbf{n}\, F_{\alpha\beta} V_\alpha \tag{2.38}$$

where α denotes a fluid particle, $\mathbf{x}_\alpha$ its position and T_α its temperature. Applying (2.37) and extracting T_β we obtain

$$T_\beta = \frac{\sum_\alpha T_\alpha (\mathbf{x}_\alpha - \mathbf{x}_\beta) \cdot \mathbf{n}\, F_{\alpha\beta} V_\alpha}{\sum_\alpha (\mathbf{x}_\alpha - \mathbf{x}_\beta) \cdot \mathbf{n}\, F_{\alpha\beta} V_\alpha} \tag{2.39}$$

that is the temperature to be applied to the boundary particle β to have an adiabatic boundary condition.

2.5.3 Thermal open boundaries: the sponge layer

For the thermal model, we use absorbing boundary conditions, implemented using the *sponge layer* approach: the boundary is assumed to have a sufficiently large thickness H_s, through which heat propagates using the standard heat equation. Given a one-dimensional reference system with the origin on the boundary interface and oriented along the inwards normal $\mathbf{n}$, the conditions for the temperature $T(n,t)$ (with n the wall depth coordinate, and t time) can be described analytically as

$$\begin{aligned} T(0,0) &= T_w, \\ T(-H_s,t) &= T_w \end{aligned} \tag{2.40}$$

where T_w is the initial physical temperature of the wall [40]. Absorbing boundary conditions are achieved when the CFL-like condition:

$$4 \frac{\kappa_w}{\rho_w c_p^{(w)}} \frac{t_{end}}{H_s^2} < 1 \tag{2.41}$$

is satisfied, with t_{end} the maximum time reached in the simulation, κ_w the thermal conductivity of the wall, ρ_w its density and $c_p^{(w)}$ its specific heat at constant pressure. For the examples we show in this paper, we use for the wall the same parameters that we use for the fluid, and the thickness required by the dynamic boundary model is also sufficient to implement the absorbing conditions for the temperature.

2.6 Stability conditions for WCSPH

With a fully explicit integration scheme, CFL-like stability conditions on the time step are necessary. In the general case, separate conditions emerge from the acceleration magnitude, the sound speed, the viscous terms and the thermal equation, so that, for each particle β, we must have ([64, 68, 69] and references within):

$$\Delta t_\beta \leq \min\left\{ C_1\sqrt{\frac{h}{\|\mathbf{a}_\beta\|}}, C_2\frac{h}{c_\beta}, C_3\frac{\rho_\beta h^2}{\mu_\beta}, C_4\frac{\rho_\beta c_p h^2}{\kappa_\beta} \right\} \tag{2.42}$$

where c_β is the sound speed at density ρ_β and the C_1, C_2, C_3, C_4 are stability constants. In GPUSPH we use $C_1 = C_2 = 0.3, C_3 = 0.125$ and $C_4 = 0.1$. The maximum time-step for the whole system is then the minimum over all particles, $\Delta t = \min_\beta \Delta t_\beta$.

Informally, the conditions ensure that information propagates by an influence radius faster than a particle covering the same distance. More specifically, the first two conditions refer to the propagation of forces and pressure waves, while the other two conditions are related to diffusion of viscous and thermal changes.

Von Neumann stability analysis The stability condition on the viscous term can be derived from a Von Neumann stability analysis of the diffusion equation. Let us call $\bar{u}_i^n$ the exact solution of the difference equation, and u_i^n the computed solution, then we can write

$$u_i^n = \bar{u}_i^n + \bar{\varepsilon}_i^n \tag{2.43}$$

where $\bar{\varepsilon}_i^n$ indicates the error at time step n and point i. We can show that, since $\bar{u}_i^n$ satisfies - by definition - the equation, then the errors $\bar{\varepsilon}_i^n$ are also solution of the equation. To demonstrate this, if we indicate by N the linear equation, we have that $N(u_i^n) = 0$ and $N(\bar{u}_i^n) \equiv 0$. Therefore, by the definition of error we can write:

$$N(u_i^n) = N(\bar{u}_i^n + \bar{\varepsilon}_i^n) = N(\bar{u}_i^n) + N(\bar{\varepsilon}_i^n) = N(\bar{\varepsilon}_i^n) = 0 \tag{2.44}$$

Hence, the errors $\bar{\varepsilon}_i^n$ satisfy the equation as well as the numerical solution.

We want to study the condition within the error is bounded. If we assume that the analytical solution is bounded, then, according to (2.43) any unbounded behavior of the error will be reflected on the numerical solution. This implies that we can study indistinctly the errors $\bar{\varepsilon}_i^n$ or the numerical solution u_i^n. For simplicity, we choose the second option.

Let us consider the spatial Fourier series decomposition of the signal u_i^n, i.e. let us write it as a linear combination of sinusoidal functions of the space, with different spatial frequencies, that we write as

$$u_i^n = \sum_{j=-N}^{N} V_j^n e^{Ii\Phi} \tag{2.45}$$

where $I = \sqrt{-1}$ indicated the imaginary unit coefficient, and $\Phi = j\pi/n$ is a spatial phase.

Considering the hypothesis of linearity, we can study the boundedness of the signal u_i^n studying that of a single generic harmonic:

$$V^n e^{Ii\Phi} \tag{2.46}$$

where V^n is a value constant in the space and variable over time with n.

The Von Neumann stability requires that the amplitude of V^n does not diverge over time, then if we define an amplification factor as

$$G \doteq \frac{V^{n+1}}{V^n} \tag{2.47}$$

we want to proof that for any Φ we have

$$|G| \leq 1 \tag{2.48}$$

Stability analysis of the diffusion equation Let us consider the diffusion equation written as

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial^2 x} \tag{2.49}$$

and let us apply a central discretization,

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{1}{\Delta x^2} \left(u_{i+1}^n - 2u_i^n + u_{i-1}^n \right) \tag{2.50}$$

Let us suppose to integrate this equation using a forward Euler integrator, obtaining:

$$u_i^{n+1} = u_i^n + \frac{\alpha}{\Delta x^2} \left(u_{i+1}^n - 2u_i^n + u_{i-1}^n \right) \tag{2.51}$$

According to what said above, we can do perform study considering the generic harmonic (2.46) instead of the signal itself, thus having

$$V^{n+1}e^{Ii\Phi} = V^n e^{Ii\Phi} + \frac{\alpha\Delta t V^n}{\Delta x^2}\left(e^{I(i+1)\Phi} - 2e^{Ii\Phi} + e^{I(i-1)\Phi}\right) \tag{2.52}$$

that leads to

$$\frac{V^{n+1}}{V^n} = 1 + \frac{\alpha\Delta t}{\Delta x^2}\left(e^{I\Phi} + e^{-I\Phi} - 2\right) \tag{2.53}$$

recalling (2.47) and some trigonometric rules, we can write

$$G = 1 + 4\beta\sin^2\left(\frac{\Phi}{2}\right) \tag{2.54}$$

where $\beta = \frac{\alpha\Delta t}{\Delta x^2}$. Applying (2.48) we get as solution $\beta \leq 1/2$, then

$$\Delta t \leq \frac{1}{2}\frac{\Delta x^2}{\alpha} \tag{2.55}$$

This result comes form the assumption that SPH can be considered a central discretization; this is true only if particles are perfectly ordinate and we use the Grenier formulation [35, 36, 37]. In fact, the value of Δt is taken smaller than the limit value expressed by (2.55), also to take into account this approximation.

Chapter 3

Lava flows

Lava flows are the outpouring of molten or partially molten rock during a volcanic eruption. Close to the eruptive vent, lava is a fluid at temperatures ranging from 920 K to 1470 K. Depending on its chemical composition, temperature, effusion rate, viscosity, and on the topography, lava can flow to great distances (up to several tens of kilometers) at greatly varying speed (from a few meters to several tens of kilometers per hour) before cooling and solidifying.

As introduced, lava is a fluid showing a variety of complex behaviors, from the generalized Newtonian rheology to the phase transition and the consequent formation of structures like levees and tunnels, making its modeling and understanding an extremely challenging task.

Figure 3.1: Effusive eruption on Mount Etna. Image from INGV.

3.1 Lava flows characteristics

Lava flows are the result of volcanic eruptions. Not all eruptions are the same, according to the presence and the violence of explosions, and other factors, like the dispersion of volcanic material, they can be classified in order of growing explosivity as *Hawaiian*, *Strombolian*, *sub-Plinian*, *Plinian* and *ultra-Plinian*. All these types of eruptions fall then into two major families, that distinguish between *effusive* and *explosive*. Lava flows are the main product of effusive eruption, and usually the main aspect of interest in presence of this kind of volcanism.

3.1.1 Effusive eruptions

We call effusive all the eruptions characterized by a low explosivity that produce mainly (see figure 3.1) fluid magma, which after being erupted takes the name of lava, that flows along the flanks of the volcano, mainly driven by gravity [78]. Effusive eruptions differ from explosive eruptions, where the magma is violently fragmented and rapidly expelled from the volcano.

Lava from an effusive eruption is usually characterized by [78]

- low content of volatile substances
- high temperature: $1000 - 1200°C$
- a viscosity that can vary over many orders of magnitude.

The flow of the lava along the volcanic surface is controlled by several factors, like gravity, the viscosity, the total erupted volume,the slope of the field and the topography in general. According to [85], the main factor affecting the length of a lava flow is the effusion rate, while the viscosity mainly affects the thickness of the flow.

Effusive eruptions are not all the same. According to the type of lava they are constituted by, they can vary in behavior and type of emplacement, forming either a smooth or a very fragmented surface. In fact, effusive eruptions can be classified in:

a) **Basaltic eruptions** constitute the majority of effusive eruptions. They are characterized by basaltic lavas, i.e. lava with a low content of silica. According to the composition and eruption conditions, of basaltic eruptions can originate three types of lavas:

1. **Pahoehoe** are very fluid lava flows that form a very smooth or at most undulating surface, with lobes of some centimeters high. The viscosity of pahoehoe lava decreases over the section of the flow, and the wrinkles on the surface are created by the movement of the underlying very fluid lava that keeps moving. These lava are particularly subjected to the formation of *lava tubes*, that are formed when the flow surface solidifies or the top of the lateral levees merges. This tube structures creates a tunnel for th lava, that, subjected to a very small loss of heat, maintains its fluidity and can travel over very long distances, covering even many tens of kilometers. At the point where the lava coming from a tube is released in the environment, one talks of *ephemeral vent*.

2. **Aa** flows show a very fragmented surface, formed by sharp blocks that can reach dimensions in the order of 1 m. With respect to pahoehoe, this kind of lavas are less fluid, either due to a different chemical constitution, or due to the lower temperature. It happens that some lavas are very fluid close to the vent, assuming pahoehoe characteristics, and become aa over distance. Aa lavas are hardly subjected to the formation of lava tubes.

3. **Pillow** are produced by basaltic underwater eruptions. They form round blocks which dimensions can vary from

few centimeters to few meters and are mainly localized in correspondence of mid-oceanic ridges.

b) **Silicic eruptions** Silicic lavas are very viscous and commonly generate explosive eruptions, but they can also come out as effusive eruptions, when the magma rises slowly through the conduit. Silicic lavas usually form big block-shaped structures called *domes.*

Hawaiian volcanos are typically characterized by effusive activity, that mainly creates pahoehoe flows.

Italy, and more precisely Sicily, owns the biggest basaltic active volcano in the world, Mount Etna, which is monitored and studied by the Osservatorio Etneo of INGV. The lavas of Etna are not very fluid, usually forming aa flows. Some pahoehoe emplacements can be found in proximity of vents or ephemeral vents.

3.1.2 Hazard posed by lava flows and risk mitigation

The hazard posed by a volcanic eruption is the probability that a given area is affected by the volcanic event. Speaking of lava flows, the possibility to have different forms of eruptions, as introduced in 3.1.1, lead to a very complex process of hazard assessment. Think for example of lava flows that can be subjected to the formation of lava tubes [29]; in these cases the liquid lava can travel far away from the original vent and come out at a certain distance from an ephemeral vent, inundating a zone that was apparently not involved in the volcanic event. Crucial to lava flow hazard assessment is the timely forecasting of flow paths [13, 14, 23], flow advance rates, and final flow morphologies. The behavior of lava flows is mainly controlled by the topography, but also by a number of parameters such as effusion rate, rheology, heat loss, viscosity, velocity, and flow morphology, all of which are interconnected [85] [38]. Each of these parameters does not impact the lava emplacement in the same way; moreover the magnitude of their effect varies with the distance from the vent and also with the timescale of observations [12]. The use of models and simulators is of crucial importance to handle all of these conditions and information, and obtain very accurate estimations of the flow behavior. This requires also very frequent and accurate data on the eruptive activity, a task particularly suited for modern satellite remote sensing [81, 32, 30].

The effect that a volcanic event can have on the society is quantified by means of the volcanic *risk*. It can be defined as the product

$$\text{Risk} = \text{Value} \cdot \text{Vulnerability} \cdot \text{Hazard} \tag{3.1}$$

where the Value is the number of human lives or the value of the goods that could be lost in the case of volcanic event, and the Vulnerability is the percentage of human lives or goods that are in risk condition during the volcanic eruption. Also in this case, the possibility to run accurate simulations can be helpful, constituting a tool to plan a risk mitigation at engineering level. In fact, the study the of the interaction between a lava flow and a urban area or, more specifically, with a building, can guide the design process in order to minimize the vulnerability of the area and the structures.

3.2 Physical properties of lava

Liquid lava is a high temperature natural substance that constitutes an heterogeneous system, mainly containing liquid phase, that is generally composed by silica, a mineral solid phase and a gaseous phase. The percentages of the three phases depend the chemical composition or environmental parameters. Because of this complex nature, the state and behavior of lava can then strongly change after a variation of quantities like temperature, pressure, mechanical stress and so on. In the following we are going to describe some of these physical properties of lava flows.

3.2.1 Temperature dependence of lava and cooling mechanisms

The strongest thermal effect that lava commonly experiences is the phase transition to liquid to solid and vice versa. Magma and molten lava usually crystallize at a temperature ranging between 700 and 1250°C. Above these temperatures lava is completely liquid, while below these temperatures it becomes solid rock. The actual crystallization interval varies with some properties of the lava; the minimum and maximum crystallization temperatures are in fact higher for basic lavas. The presence of fluids decreases these two temperatures while for the lavas subjected to the lithostatic pressure the crystallization temperature is higher.

The cooling process of a lava flow occurs primarily at the surface. Here, two phenomena are mainly involved: thermal radiation and air convection. A smaller amount of heat is instead dissipated by diffusion through the ground. Thanks to surface dissipation, lava flows cool down very quickly at the initial phase when the lava is still liquid. As solid surface starts to appear, it acts as thermal shielding, insulating the liquid interior that then takes more time to cool down. In case of a thick flow, it can take even tens of years for the whole amount of lava to reach the ambient temperature.

3.2.2 Viscosity of lava

Knowing the viscosity of lava is not a simple task. Due to the extreme condition where lava flows develop (very high temperature, dangerous atmosphere, etc..) in most of the cases the viscosity of lava is not directly measurable, and, in any case, not through the whole volume of fluid. Moreover, the viscosity depends on numerous parameters and conditions of the fluid itself and is therefore continuously changing, over space and time.

As lava is a commonly known viscous fluid, its viscosity is a fundamental parameter to be known for a good study, modellization or even simulation of this fluid. A good model should then be able to reproduce the dependence of the physical properties of the flow on the viscosity of the lava. In the follow we discuss some of the main correlations.

A first, basic, contribution to the viscosity of lava is its composition. We have seen, while distinguishing between acid and basic lavas, that the silicate content plays an important role, increasing the viscosity of the system. After silicate, the major chemical contribution to the viscosity of lava comes from its content of volatile substances; firstly, water. In a silicatic composition, the silica forms tetrahedral groups that tend to give solidity to the fluid, resulting giving a positive contribution to the fluid viscosity. When water is added to the lava, the tetrahedral structure breaks, allowing the silica to bond with oxygen, giving a negative contribution to the viscosity, that thus decreases. This process is known as *depolymerization*, and affects mainly acid lavas, that have a larger silica content. The dependence of the viscosity on the water content is also linked to the temperature of the lava, becoming almost irrelevant at high temperatures.

Another quantity that deeply alters the viscosity of the lava is its temperature. We don't go into detail from a chemical or physical point of view, but we mention that viscosity decreases as the temperature increases, and this occurs in a different way from lava to lava.

In addition, the non homogeneous nature of the lava, the presence of internal polymeric structures, and many other aspects, make lava a Non-Newtonian fluid. In fact, the viscosity of lava varies with the applied mechanical stress, and the way in which the viscosity varies due to the stress specifies the *rheology* of the fluid. We have already introduced some non Newtonian rheologies in 2.1.2; although the rheology of lava is still being investigated, the Bingham model commonly the main candidate. Some times, the rheological properties of the lava are neglected, and the fluid is modelled as Newtonian, an assumption that becomes more realistic when the lava is very hot.

We limit our discussion on the viscosity of lava to these quantities, since are those that we explicitly consider in our models. Of course, many other factors contribute to the viscosity of lava, that, as we mentioned, is one of the most complex aspects of this fluid.

Chapter 4

Numerical simulation of lava flows

Lava flow modeling is important both in the scientific field, to improve our knowledge on the fluid itself, and in many practical applications, such as the simulation of potential hazard scenarios and the planning of risk mitigation measures. In fact, the growing urbanization around volcanic edifices also increases the potential risks and costs that volcanic flows represent, leading to an increasing demand for faster and more accurate predictions of flow extent, both in terms of space and time. Mathematical modeling and computer simulations can play an essential role in improving our understanding of the lava flow patterns, its morphology, and thermal evolution [24, 30].

The complex nature of the fluid, aspects such as free surface and irregular topographies, and phenomena like phase transition and the consequent formation of levees and tunnels make simulation of lava flows an extremely challenging task for Computational Fluid Dynamics (CFD). As a result, various computer codes have been developed to predict lava flow footprints and emplacement dynamics. Codes differ in their physical implementations, numerical accuracy, and computational efficiency.

Generally, a numerical code for simulation of lava flows must consider [18]:

- Topography or slope
- Eruptive input conditions: volume effusion rate, vent geometry and effusion temperature
- Thermal boundary condition at the top and the bottom of the flow.
- Physical properties of the lava: density, thermal conductivity, rheology.

In the modeling of lava flow hazards [13, 22, 23, 44], common approaches to the simulation of lava flows start by reducing the complexity with a number of different strategies, such as reduced dimensionality [27], simplified thermal or dynamic models, or the use of stochastic approaches with little or no physical modeling [19]. These simplifications allow easier implementations and higher performance, and while the results may still be useful for real-time forecasting [14], risk mitigation [79] and the production of long-term scenarios [22], they are inadequate for a more thorough study of the behavior of the fluid and the laws underlying its rheology, which require the detailed modeling of the full three-dimensional flow and its rheological aspects.

Here we will give a description of the main methods that are currently adopted for the simulation of lava flows, following a scheme introduced adopted by [18].

4.1 Channelled models

Channelled models introduces a strong simplification in the problem dimensionality, reducing it to a 1D problem. The fluids are confined and the flow advances only in one direction, downslope. In contrast to the simplified emplacement model, channelled methods can manage a quite complex thermal model, and consider aspects like the crystallization rate, and rheology models. The velocity of the flow down the channel depends on the channel dimensions [80] and on the rheology of the lava.

The main code that implements 1D channelled lava is FLOW-GO [38], [39], that models finite amounts of moving lava that is contained between stagnant levees and has no mechanically continuous roof (see figure 4.1). The top of the moving lava therefore represents a free surface open to the atmosphere, but its sides and bottom are in contact with levee walls and the emplaced flow base.

From a thermal point of view it models the heat lost by radiation and convection at the surface and conduction at the base. The rheology is computed from its crystal- and temperature-dependence. The flow along the channel is stopped when the

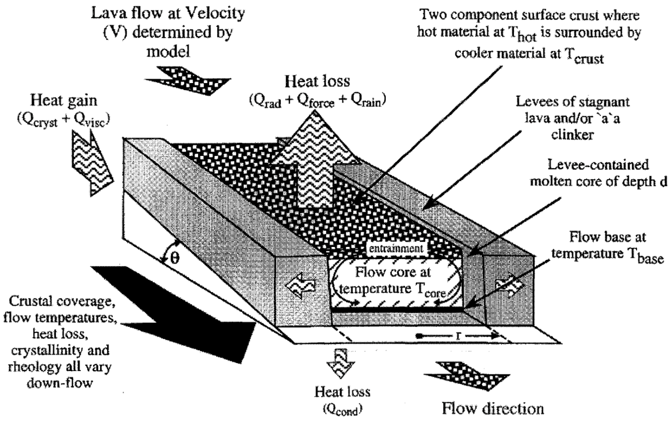

Lava flow at Velocity (V) determined by model
Two component surface crust where hot material at T_{hot} is surrounded by cooler material at T_{crust}
Heat loss ($Q_{rad} + Q_{force} + Q_{rain}$)
Heat gain ($Q_{cryst} + Q_{visc}$)
Levees of stagnant lava and/or `a`a clinker
Levee-contained molten core of depth d
Flow base at temperature T_{base}
entrainment
Flow core at temperature T_{core}
θ
r
Crustal coverage, flow temperatures, heat loss, crystallinity and rheology all vary down-flow
Heat loss (Q_{cond})
Flow direction

Figure 4.1: Scheme of the channelled flow model used in FLOWGO. Image taken from [38]

lava cools down and its rheological properties equal that of a solid.

This kind of models are very fast to run as they do not involve calculating the fluid motion. However, important limitations arise from the assumption of one-dimensionality forcing the local channel width to directly correlate with ground slope.

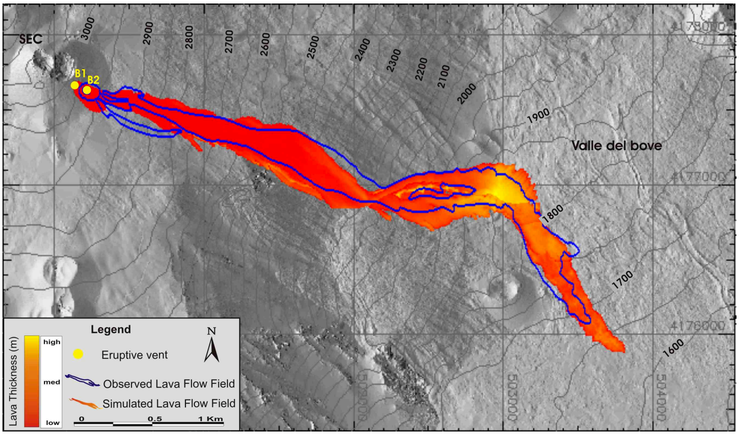


Figure 4.2: The 2006 eruption on Etna, simulated with MAGFLOW. The colors from red to yellow indicate the thickness of the simulated lava flow, while the blue line tracks the contour of the real emplacement. Image taken from [83].

4.2 Cellular Automata

The cellular automata (CA) approach is one of the most popular used to model lava flow emplacement, with codes such as MAGFLOW [82, 24, 15, 44, 32], SCIARA [20] and FLOWFRONT [88].

Using Cellular Automata, the computational domain is discretized by means of a 2D grid of cells. Each cell is characterized by properties such as altitude, lava height and temperature, and lava flow advance, and cooling is described through the evolution of cell properties. The flow of lava from one cell to another depends on the low density and on the slope, constituted by the difference of altitude between two adjacent cells. The first implementations of CA used a plastic rheology model to stop the flow, modern codes model instead the solidification by means of a solidus temperature. Figure 4.2 shows a simulation obtained using the MAGFLOW simulator.

Cellular automata models are fast to run; however, as any 2D model, their main drawback is the absence of a detailed vertical description of the lava flow, which is important in coupling surface and basal heat losses to the bulk rheology evolution, or to model the formation of phenomena like lava tubes.

4.3 Depth-averaged models

Depth-averaged methods reduce the spatial dimensionality of the problem using the shallow-water equation (SWE). Assuming that the horizontal horizontal length scale is much greater than the vertical one, SWE neglect the vertical component of the flow assuming homogeneous properties throughout the section.

Several rheology models can be implemented, from Newtonian to Bingham model, even though the assumption of a constant viscosity value along the vertical profile limits the feedback of rheology into flow dynamics.

Depth-averaged codes for volcanological applications include the model developed by G. Macedonio [61], the VOLCFLOW code [53, 54] and the implementation made in RHEOLEF [3].

Neglecting the variations along the flow depth results on a simplification of the problem with a consequent reduction of simulation times.

4.4 Nuclear based models

The name of this model comes from its first development in the field of nuclear plants. In such context, the code is used to simulate and optimize the spreading phase of melted substance in case of nuclear reactor meltdown. These codes are an evolution of the cellular automata (4.2) method where the third dimension is added, discretizing the height of each cell into several vertical cells. An example of application to simulation of lava is the code LavaSIM [74].

In nuclear based model, the mass, momentum and energy conservation are resolved for this 3D system, thus allowing the reproduction of complex emplacement features related to solidification, such as lava lobes or tubes. The limit on simulating a lava flow is in the free surface, that has not been modelled yet.

4.5 Generic 3D Computational Fluid Dynamics codes

Studying the advance of a fluid is an interest job in many fields, beside volcanology. Examples are Nuclear Engineering, Civil Engineering and metallurgy. To supply these needs, a lot of tools for CFD have been created in the past, ranging from commercial software packages to community-driven or government supported open-source libraries. Despite the final product the original design of the methods shared the same aim with volcanology, the adoption of such libraries and packages to use for lava flow is not straightforward, and can require additional capabilities and models, such as crystallization, variable effusion rates, complex rheology and the strong temperature dependence of physical properties. Here, we discuss very common CFD packages, one open-source and one commercial, usually applied to the simulation of lava flows.

4.5.1 OpenFOAM

OpenFOAM is the acronym of Open Field Operation And Manipulation (http://www.openfoam.com) [49]. It is a finite-volume-method-based open-source software package produced by OpenCFD Ltd. OpenFOAM solves problem of continuum-mechanics, including CFD problems. Among the latter, OpenFOAM can deal with complex fluids, chemical reactions, turbulence, heat transfer and so on. The code is fully parallelized using OpenMPI, and has straightforward interfaces with external meshing, and pre- and post-processing tools.

Beside the use of already embedded packages, OpenFOAM gives the users the possibility to add new equations, solvers and applications, a functionality that can be exploited to add specific feature of lava.

4.5.2 Flow3D

Flow3D is a commercial software produced and distributed by Flow Science Inc. It finds application mainly in engineering and is particularly focused to the field of the Computational Fluid Dynamics.

Flow3D is based on the Volume-of-Fluid algorithms, combined with Level Sets to obtain interface tracking feature. This last characteristic makes it suited for the simulation of free surface flows, as lava flows are.

The two main downsides of Flow3D are its slowness and its high price.

4.6 Mesh-free methods

All of the methods that have been introduced so far in this chapter rely on meshes. The main differences between mesh based and mesh-less methods has already been introduced in 1.1, then here we limit our discussion to the introduction of some mesh-free methods and their known implementations.

The two major mesh-free approaches are Smoothed Particles Hydrodynamics, **SPH** [42], and the Lattice Boltzmann methods, **LBM** [72, 73]. While the SPH relies on the Navier Stokes equations, LBM uses the Discretized Boltzmann equations. So far, only SPH has been applied to the study of lava flows, in its implementation called GPUSPH, that is one of the main objects of the work discussed in this thesis. For each of these methods, additional terms such as the Boltzmann discretization for the

LBM or kernel size limitation for the SPH inherently generate numerical diffusion, and require careful benchmarking [18].

Chapter 5

The GPUSPH particle engine

The standard weakly-compressible SPH formulation has the benefit of being completely parallelizable, and is therefore quite apt for execution on massively parallel hardware such as modern Graphics Processing Units. GPUSPH is an implementation of WCSPH running on GPUs, offering the advantages of the method at reasonable performance.

5.1 Parallelizability of the WCSPH method

So far, the SPH method has been discussed in terms of the advantages that it offers when applied to some classes of problems, showing a lot of interesting aspects. An important factor that becomes fundamental in applicative contexts is the execution time of the algorithms. A simulation usually involves thousands or even millions of particles, each requiring some time consuming operations; think for example of the iteration over the neighbors during the forces computation. It is therefore essential to manage all these processes, and relative data, in a clever way. A simple approach, employing a computer to perform all computations in a serial fashion, would in fact lead to unsustainable computational times, that for complex simulations would range in the order of months, or even years.

One of the most important aspects of the WCSPH method, introduced in 2.2.2, is the use of a state equation to close the system, instead of a Poisson equation. This gives a great advantage during the integration process (see section 1.7), making each particles independent from the others. In fact, the pressure of any particle can be directly obtained from its density, during the force computation, without the need to solve any system of equations and leaving the momentum and continuity equations uncorrelated; therefore, the problem can be solved using only direct methods.

Some interaction among the particles necessarily occurs in the computation of the forces, during the iterations over the neighbors, but in any case, this process can be independent for each particle, that only needs to access the information of its neighbors.

It is then clear that all the particles can be processed by independent threads, running in parallel the same operations at the same time. For this reason WCSPH is considered extremely parallelizable, a characteristic that allows it to be run on specific hardware, leading to a strong reduction of simulation times. Beyond the use of multi-processor computers, that would anyway give a notable contribution to the performance of the computation, we have Graphic Processing Units (GPUs), a class of hardware designed to do right this job.

5.2 General-Purpose computing on Graphic Processing Units (GPGPU)

Designed to work on digital images, constituted of millions of pixels, GPUs are able deal with large sets of elements, executing the same operation simultaneously a huge number of them. This kind of paradigm, that sees GPUs employed in other fields than graphics, is called General Purpose Computing on Graphical Processing Units (GPGPU), and is nowadays supported by different programming models, like standards such as OpenCL [55] or proprietary solutions such as NVIDIA CUDA [70].

5.2.1 Stream processing

Modern GPUs are designed around the stream processing paradigm, a simplified model for shared-memory parallel programming that sacrifices inter-unit communication in favor of higher efficiency and scalability. At an abstract level, stream processing is defined by a sequence of instructions, constituting a so called *computational kernel*, to be executed on each element of an input data stream to produce an output data stream, under the assumption that each input element can be processed independently from the others (and thus in parallel).

Computational kernels

A computational kernel is similar to a standard function in classic imperative programming languages; at run time, as many instances of the function will be executed as necessary to cover the whole input data stream. Such instances are called *work-items*, and may be dispatched in concurrent batches, running in parallel as far as the hardware allows, and the programmer is generally given very little control, if any at all, on the dispatch itself, other than being able to specify how many instances are needed in total. This choice allows the same kernel to be executed on the same data stream, adapting naturally to the characteristics of the underlying hardware, and is one of the main characteristics of stream processing.

For example, if the hardware can run $1,000$ concurrent work-items, but the input stream consists of $2,000,000$ total elements, the hardware may batch $1,000$ work-items for execution at once, and then dispatch another $1,000$ when the first batch completes execution. This continues until the entire input stream has been processed, executing $2,000$ total batches. For the same workloads, more powerful hardware able to run $100,000$ concurrent work-items may be able to complete sooner by issuing 20 total batches, in a manner completely transparent to the programmer.

This programming model fits very well the simpler workload needed in many steps of image rendering rendering process for which GPUs are designed. However, the more sophisticated requirements of general-purpose programming have led to the extension of the stream processing paradigm to provide programmers with finer control on the work-item dispatch as well as the possibility for efficient data sharing between work-items under appropriate conditions.

A modern stream processing device (typically a GPU, but may also be a multi-core CPU with vector units, a dedicated accelerator like Intel's Xeon Phi, or a special-design FPGA) is composed of one or more compute units (each being a CPU core, a GPU multiprocessor, etc) equipped with one or more processing elements (a SIMD lane on CPU, a single stream processor on GPU, etc), which are the hardware components that process the individual work-items during a kernel execution. The programming model of these devices, as OpenCL or NVIDIA CUDA, mentioned above, exposes the underlying hardware structure by allowing the programmer to specify the granularity at which work-items should be dispatched: each work-group is a collection of work-items that are guaranteed to run on a single compute unit; work-items within the same work-group can share data efficiently through dedicated (often on-chip) memory, and can synchronize with each other, ensuring correct instruction ordering. Tuning work-group size and the way work-items in the same work-group access data can have a significant impact on performance.

The GPU multiprocessors are further characterized by an additional level of work-item grouping at the hardware level, as the work-items running on a single multiprocessor are not independent from each other: instead, a single instruction pointer is shared by a fixed-width group of work-items, known as the *warp* on NVIDIA GPUs, or wavefront on AMD GPUs, corresponding in a very general sense to the vector width of SIMD instructions on modern CPUs. The subgroup structure of kernel execution influences performance in a number of ways. The most obvious way is that the size of a work-group should always be a multiple of the subgroup size: a non-integer subgroup would be fully dispatched anyway, but partially masked, leading to lower hardware usage. Additional aspects where the subgroup partitioning can influence performance are branch divergence and coalesced memory access.

Branch divergence occurs when work-items belonging to the same subgroup need to take different execution paths at a given conditional. Since the subgroup proceeds in lockstep for all intents and purposes, in such a situation the hardware must mask the work-items not satisfying either branch, execute one side of the branch, the invert the mask and execute the other side of the branch: the total runtime cost is then the sum of the runtimes of each branch. If the work-items taking different execution paths belong to separate subgroups, this cost is not incurred, because separate subgroups can execute concurrently on different code paths, leading to an overall runtime cost equal to that of the longer branch.

Coalescence in memory access is achieved when the controller of a GPU can provide data for the entire subgroup with a single memory transaction. Ensuring that this happens is one of the primary aspects of an efficient GPU implementations, and can usually be achieved that data in memory is laid out so that subgroups access memory that is consecutive and properly aligned.

5.2.2 Stream processing and particle systems

Stream processing is a natural fit for the implementation of particle systems, since the vast majority of algorithms that rely on particle systems are embarrassingly parallel in nature, with the behavior of each particle determined independently, thus providing a natural map between particles and work-items for most kernels. This allows naive implementations of particle systems to be developed very quickly, often resulting in massive performance gains over trivial serial implementations running on single-core CPUs. Such implementations will however generally fail at leveraging the full computational power of GPUs, except in the simplest of cases. Any moderately sophisticated algorithm will frequently require a violation of the natural mapping of particles to stream elements (and thus work-items), either in terms of data structure and access, or in terms of implementation logic, to be able to achieve the optimal performance on any given hardware.

5.2.3 Limitations in the use of GPUs

Programmable GPUs have brought forth a revolution in computing, making certain forms of large-scale parallel computing accessible to the masses. Many applications have seen significant benefit from a transition to the GPU as supporting hardware, and in response vendors have improved GPU architectures, making it easier to achieve better performance with less implementation effort. When choosing the GPU as preferential target platform, however, developers must take into consideration the fact that not all users may have high-end professional GPUs, and while the stream computing paradigm is largely sufficient in compensating for the difference in computational power, there are at least two significant aspects that must be explicitly handled. Memory amount is one of these issues: consumer GPUs typically only have a fraction of the total amount of RAM offered in professional or compute-dedicated devices: while the latter may feature up to 16 GB of RAM, low-end devices may have 1/4th or even 1/8th of that. Moreover, even the amount of memory available on high-end devices may be insufficient to handle larger problems. Software should therefore be designed to allow distribution of computation across multiple devices.

Another limitation of GPUs is constituted by the numerical precision. Being designed for computer graphics, GPUs typically focus on single-precision (32-bit) floating-point operations, and double-precision (64-bit) may be either not supported at all, or supported at a much lower execution rate (as low as 1:32) than single-precision, which may remove the computational advantage of using GPUs in the first place. This can be true even on high-end GPUs, as was infamously the case for the Maxwell-class Tesla GPUs from NVIDIA. Designing the software around the use of single-precision can therefore allow supporting higher performance across a wider class of devices, but it may require particular care in the handling of essential state variables in particle systems.

5.2.4 Reductions

Not all the processes that need to be done on a large amount elements are embarrassingly parallel. Reductions, for example, are operations that obtain a single value from the values of all the elements (for example, the minimum of all elements, or their sum).

While the naive approach to reductions is intrinsically sequential, *parallel reductions* are possible, by processing multiple elements with a single work-items, storing intermediate results, and then similarly processing these intermediate results until a single final result is obtained.

GPU architectures provide features that allow much faster implementation of parallel reductions, such as the *shared memory* that allows fast data exchange between work-items in the same work-group. An efficient parallel reduction on GPU can then be implemented in two passes, with the first pass producing a single result per work-group, and the second pass further reducing the partial results of the first pass. By using a "sliding window" approach, it is possible to choose the launch configuration for the first pass so that it is just enough to saturate the device, while minimizing the number of work-groups: this is achieved by having each work-item process multiple elements, by moving forward in the vector of source elements by the size of the launch grid itself, until the source elements have been exhausted.

In our applications, reductions are useful for the computation of the maximum allowed time-step, as discussed in section 2.6, and the computation of the vector norms in the computation of the CG used for the semi-implicit formulation (section 5.3.5).

5.3 GPUSPH

GPUSPH is the first implementation of WCSPH to fully run on GPUs [41], created at the TecnoLab of the Istituto Nazionale di Geofisica e Vulcanologia (INGV) in Catania. It is able to run three-dimensional SPH simulations, disposing of several embedded models for boundary, viscosity, integrators, SPH formulations and so on. In order to better handle the computational needs of lava flow simulations, GPUSPH has been extended to distribute computations across multiple GPUs [75], even across separate nodes in a cluster [76]. Exploiting the massively parallel nature of the hardware, it runs two orders of magnitude faster than a standard CPU implementation, which has allowed the application of GPUSPH to a number of practical fields ranging from oceanography [86] to industrial applications [10].

5.3.1 Structure of GPUSPH

The basis of the GPUSPH particle engine lay on the three fundamental steps of a SPH simulation, introduced in section

1.7: Initialization, forces computation and integration. Plus, GPUSPH includes additional steps and processes, that are needed for service routines or to optimize the simulation in terms of quality of the result or performance.

As for any code implementing a GPGPU program, the structure of GPUSPH has a main net division in *host* and *device* parts, relative to what is run on the CPU or the GPU, respectively.

The host part manages the initialization step and the launch of the kernels that, running on the device, will actuate the forces computation, the integration and any operation that has to be run in parallel fashion. Moreover, the host also manages the writing of the data on the the disc.

Main kernels

Going deeper into detail on force computation and integration, we can describe the main kernels that are involved [75]; we can refer to the main kernels as

- Build Neighbor list: this kernel for each particle builds a list of the neighbors;
- Forces: for each particle computes the interaction with the neighbors and its maximum allowed Δt (section 5.3.5);
- minimum Scan: selects the minimum Δt among the maxima provided by each particle;
- Euler: for each particle update the state variables integrating over the selected δt.

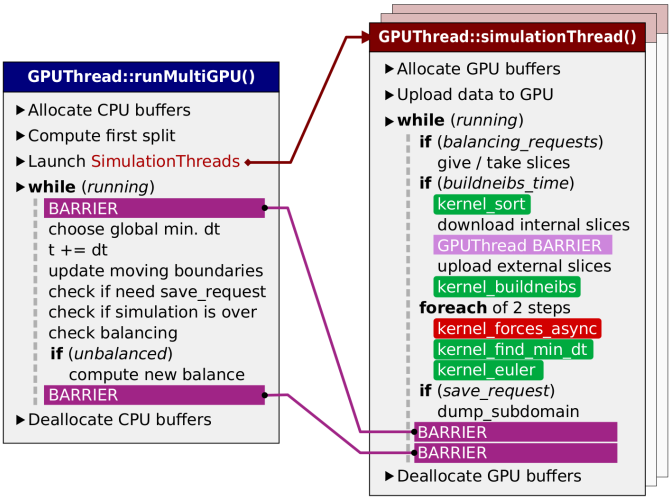


Figure 5.1: Scheme of the GPUSPH particle engine. Image taken from [75].

Multi-GPU management

In addition to the parallelism given by graphic cards, GPUSPH exploits a second level of parallelism, coming from the possibility to run a simulation on multiple devices, that in case of very large particle systems can be an essential condition to run a simulation. This may in fact be necessary simply due to the limited resources available on a single GPU: high-end GPUs currently have at most 16 GB of RAM, which may limit the particle system size to a few tens of millions, depending on the complexity of the system. Even for smaller systems, however, distribution over multiple GPUs can provide a performance boost, provided each of the devices involved is saturated, since otherwise the overhead involved in distributing the particle system will be higher than the benefits offered by the higher computational capacity.

To support this functionality, GPUSPH uses CPU threads, organized as shown in figure 5.1. Here, the sheet in blue represents the main thread, that launches all the functions executed by the CPU and coordinates the work of GPUs. The sheets in red are other CPU threads, each relative to a GPU, that we call Workers. Under request from the main thread, each worker launches the proper kernels on its corresponding device. A system fo barrier depicted in purple, articulates the communication between the master thread and the workers, ensuring the synchronization among the latter ones.

In this context we don't go more in detail in the management of a multi-device simulation, that can be found in [75].

5.3.2 Neighbors list construction

We introduced in 1.7 that the neighbors of a particle are accessed multiple times during an iteration. Searching the neighbor of each particles over the whole domain is a very expensive process, that with a naive form can be a $O(N)$ approach; it is then convenient to prepare a neighbors list once and then iterate it at any point of the iteration. To reduce the order of complexity of the neighbor search, an efficient strategy comes from the partitioning of the domain in cells. Given a neighbors search

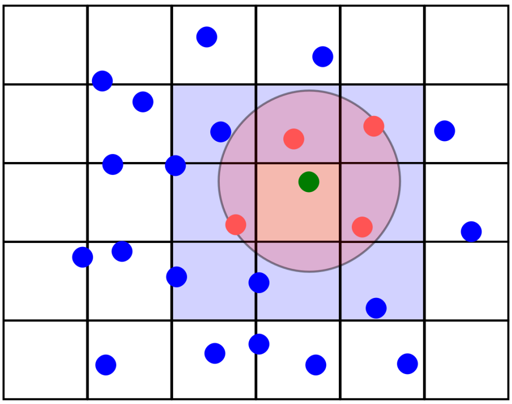


Figure 5.2: Example of partitioned 2D domain. Neighbors are sought in the blue and pink cells. Image taken from [75].

radius r, we can subdivide the domain with a regular grid where the stepping in each direction is no less than r. This guarantees that the neighbors for any particle in any given cell can only be found at most in the adjacent cells in each of the cardinal and diagonal directions, as depicted in figure 5.2

5.3.3 Numerical precision

As introduced in 5.2.3, one drawback coming from the use of GPUs is the numerical precision, that usually is limited to single precision. The effect of this limitation can be mitigated by adopting particular strategies as we discuss in the following.

Relative density

As discussed in [43], naive implementations of WCSPH are affected by numerical precision issues; these are more evident when using single precision floating-point values, which is often preferred on GPUs for performance reasons, and a number of strategies can be employed to reduce the impact of precision issues. In particular, for the density it is preferable to use the relative density difference

$$\tilde{\rho} = (\rho/\rho_0 - 1) \tag{5.1}$$

as particle property, with ρ_0 being the fluid reference density, rather than the absolute density ρ. The continuity equation can then be rewritten as

$$\frac{D\tilde{\rho}}{Dt} = -\frac{\tilde{\rho}}{\rho_0}\nabla \cdot \mathbf{u} \tag{5.2}$$

Cell relative coordinates

In SPH, as in any particle method, the actual particle position in the global reference system is almost never needed as such. Instead, we need the relative distance of the given particle to its neighbors [43]: the (global) particle position is therefore only used to be subtracted from the (global) particle position of the neighbor. Although this subtraction is done between numbers of similar magnitude, the lower precision in the representation of the (global) positions far from the origin leads to corresponding lower precision in the relative distance as well. This becomes a serious problem in case of simulation domain with high aspect ratio; in fact the need to resolve small differences over the short side, goes in conflict with the need to represent big values over the long side. An approach to reach uniform precision throughout the domain is to exploit the cell grid, introduced in the neighbors search in 5.3.2, as auxiliary data structure for the particle positions. Instead of storing the particle position in the global reference system, we store the particle position as a combination of the (integer) coordinate of the cell the particle is located in, and a (single-precision) floating-point value representing the particle position relative to the cell center. In each direction, the position is thus expressed by a pair (i, f), with i spanning the range of cell indices, and f is a single-precision value, such that $-0.5d < f < 0.5d$, where d is the length of the cell side. With this strategy, the relative position from the neighboring particles is easily computed as the difference between the floating-point values, eventually corrected by an amount equal to the cell side in the case of neighbors which are not in the same cell as the particle itself. Absolute positions are only used outside the SPH algorithm, in two places. The first place is during the initialization phase: this is done using absolute positions, computed by the host in double precision, and then translated into coordinates (i, f) before uploading them to the device. Secondly, absolute positions are used when storing the simulation results: in this case absolute positions are reconstructed, in double precision, on the host, from (i, f) coordinates.

Double single-precision for temperature

The integration of temperature becomes another delicate topic when dealing with a finite precision. In application to very hot fluids, small variations of the temperature could be neglected during the integration process. This phenomenon becomes more important as te time step is taken smaller. This is the case of lava flow: the high viscosity of the fluid, or a high speed of sound in the case of implicit integration, can determine a very small time step, easily reaching the order of 10^{-8} s; considering that the temperature can range in the order of 10^3, we get that almost any derivative of the temperature is lost during the integration process. This problem cannot be overcame using a relative quantity, as done for the density 5.3.3, since the temperature can simultaneously assume very different values over the simulation domain. On the other side, working with double precision variables on device can drastically degrade the performance, as said in 5.2.3. A solution to run double precision integrations without using double precision variables, is to use two single precision variables to store and integrate the quantity. The two variables, let's say x_I and x_D, store the integer and decimal parts of the value x, and their content is added every time the complete value is needed. The integrations are then performed on x_D, and the gained variation is reported to the integer part every time the magnitude of x_D reaches one. If one knows the highest value that can be assumed by x, in case the extension guaranteed by x_I in this context exceeds the requirements, the numerical precision can be furtherer improved by shifting the separation point between x_I and x_D to lower orders of magnitude, instead of using the unity.

Kahan summation

Another issue due to the finite precision appears when summing up several small quantities to a larger one. This is the case of the Shepard interpolation, for example, used in for the dummy boundary model (equations (2.31) and (2.32)) or, even more importantly, in the summations needed to build the matrix for the semi-implicit formulation, as we will see in section 5.3.5. Here, all contribution from the neighbors are summed up, in some cases starting from an already non-zero initial value, and while the partial sum grows in module, the less significant contribution from the incoming addends is lost. A solution to achieve a better accuracy is the adoption of a compensated algorithm, like Kahan summation [52], where a secondary variable is used to store the unrepresentable reminder in summation operations.

```
Result: s
s=0;
c=0;
for $i \leftarrow 0$ to $n$ do
    y = x[i] -1;
    t = s + y;
    c = (t-s)-y;
    s=t;
end
```

Algorithm 1: Kahan summation

For the computation of sum s, the Kahan algorithm applied in single precision performs numerically as well as a summation done in double precision, followed by a rounding to single precision of the final result.

5.3.4 Density diffusion: the Molteni and Colagrossi approach

Very often, SPH simulations are affected by the development of instabilities as well as the presence of high frequencies numerical noise on the pressure field. A typical way to face this problem is to add an artificial viscosity [66], but sometimes this approach is not sufficient. Instead, the addition of a diffusive term in the continuity equation (2.1) can avoid these drawbacks giving more reliable results [63]. An example of diffusive term proposed by [63] is

$$D_\beta = \xi h c_0 \sum_\alpha \psi_{\alpha\beta} \cdot \nabla W_{\alpha\beta} dV_\alpha \tag{5.3}$$

where

$$\psi_{\alpha\beta} = 2\left(\frac{\mathbf{u}_\beta}{\mathbf{u}_\alpha} - 1\right) \frac{\mathbf{x}_\alpha - \mathbf{x}_\beta}{\left|\mathbf{x}_\alpha - \mathbf{x}_\beta\right|^2 + \varepsilon_h h^2} \tag{5.4}$$

with $\varepsilon_h = 0.01$

5.3.5 Integrator

The integrator the part of the code that accomplish the integration step, discussed in 1.7. We refer to a particular sequence of operations that implement an integrator as the *integration scheme*. After a discussion on the way of determining the time step, we will see the two integration schemes that are implemented in GPUSPH, one fully explicit and a semi-implicit one, where only the viscous forces are integrated implicitly.

Time step

The time step is fixed in order to satisfy the stability conditions discussed in 2.6. In most inviscid or low-viscosity flows, the dominant time-stepping condition is the one given by the sound speed. However, for highly viscous flows the viscous term takes over; since this term is quadratic in h, while the sound speed term is linear, we can compute the resolution at which the viscous term becomes dominant, given by

$$h < h^\star = C_r \frac{\mu_\beta}{\rho_\beta c_\beta} \tag{5.5}$$

where $C_r = C_2/C_3$. From a different perspective, at a given resolution there is a critical viscosity beyond which the viscous CFL term becomes dominant:

$$\mu_\beta > \mu_\beta^\star = \frac{\rho_\beta c_\beta h}{C_r}. \tag{5.6}$$

Explicit integration scheme

One of the main advantages of WCSPH is that the acceleration, density derivative, thermal change etc can be computed and integrated independently for each particle. When combined with an explicit time-stepping scheme, this leads to a natural parallelization of the method, where the derivatives for each particle are computed independently in parallel, and the new position, velocity, density, etc are then integrated independently in parallel. Explicit integration schemes are therefore well-suited and often preferred in implementations of the SPH method.

GPUSPH uses a fully explicit predictor–corrector time integration scheme which can be described with the following four steps:

1. compute acceleration, density derivative and temperature derivative at instant n:

   a) $\mathbf{a}^{(n)} = \mathbf{a}(\mathbf{x}^{(n)}, \mathbf{u}^{(n)}, \rho^{(n)}, T^{(n)}, \mu^{(n)})$,
   b) $\dot{\rho}^{(n)} = \dot{\rho}(\mathbf{x}^{(n)}, \mathbf{u}^{(n)}, \rho^{(n)}, T^{(n)})$,
   c) $\dot{T}^{(n)} = \dot{T}(\mathbf{x}^{(n)}, \mathbf{u}^{(n)}, \rho^{(n)}, T^{(n)})$,

   with $\mathbf{x}$ the position, $\mathbf{u}$ the velocity, ρ the density, T the temperature, and μ the dynamic viscosity;

2. compute half-step intermediate positions, velocities, density and temperature:

   a) $\mathbf{x}^{(n\star)} = \mathbf{x}^{(n)} + \mathbf{u}^{(n)}\frac{\Delta t}{2}$,
   b) $\mathbf{u}^{(n\star)} = \mathbf{u}^{(n)} + \mathbf{a}^{(n)}\frac{\Delta t}{2}$,
   c) $\rho^{(n\star)} = \rho^{(n)} + \dot{\rho}^{(n)}\frac{\Delta t}{2}$,
   d) $T^{(n\star)} = T^{(n)} + \dot{T}^{(n)}\frac{\Delta t}{2}$,

3. compute corrected acceleration, density derivative and temperature derivative

   a) $\mathbf{a}^{(n\star)} = \mathbf{a}(\mathbf{x}^{(n\star)}, \mathbf{u}^{(n\star)}, \rho^{(n\star)}, T^{(n\star)}, \mu^{(n\star)})$,

   b) $\dot{\rho}^{(n\star)} = \dot{\rho}(\mathbf{x}^{(n\star)}, \mathbf{u}^{(n\star)}, \rho^{(n\star)}, T^{(n\star)})$,
   c) $\dot{T}^{(n\star)} = \dot{T}(\mathbf{x}^{(n\star)}, \mathbf{u}^{(n\star)}, \rho^{(n\star)}, T^{(n\star)})$,

4. compute new positions, velocities, densities and temperatures:
   a) $\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + (\mathbf{u}^{(n)} + \mathbf{a}^{(n\star)}\frac{\Delta t}{2})\Delta t$,
   b) $\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \mathbf{a}^{(n\star)}\Delta t$.
   c) $\rho^{(n+1)} = \rho^{(n)} + \dot{\rho}^{(n\star)}\Delta t$.
   d) $T^{(n+1)} = T^{(n)} + \dot{T}^{(n\star)}\Delta t$.

In this integration scheme, there are three steps in which the acceleration is integrated: step 2b, to obtain the velocity at the intermediate step, step 4a to obtain the velocity for the trapezoidal rule, and step 4b to obtain the final velocity.

This second order explicit integrator in characterized by a very light and parallelizable structure, that implemented on GPUs leads to very fast execution times.

Semi-implicit integration scheme

One main drawback of the explicit integration scheme is the dependence of the time step (5.3.5) on the stability conditions that we saw in 2.6. This becomes a problem when dealing with very high viscosity fluids, where the time step can drop to the order of 10^{-8} s or even below, leading to very long simulation times and issues of numerical resolution that can prevent the simulation to run at all. A semi implicit integration scheme [90, 93], where the viscous contribution is integrated implicitly, allows the neglect of the stability condition on the viscosity and overcomes the issues related to it.

While a fully implicit scheme would allow the use of an even larger time step, its use with the WCSPH formulation would lead to non-linear systems with most equations of state; an incompressible formulation would be better suited for a fully implicit scheme, which however have other limitations, such as difficulties imposing the appropriate boundary conditions on the free surface [48]. Our choice to only treat the viscous term implicitly aims at avoiding these issues and produce a simpler approach, particularly in combination with the appropriate boundary model.

To introduce the semi-implicit scheme, we will at first look at each of the acceleration integration phases as if it were a simple Forward Euler integration step, to show how implicitation is achieved. We will then present the full semi-implicit predictor/ corrector scheme, that maintains second order accuracy in time.

We can write the Navier–Stokes equation ((2.16) or (2.17)) in the abbreviated form:

$$\frac{D\mathbf{u}}{Dt} = \mathbf{f}_P + \mathbf{f}_v + \mathbf{g} \tag{5.7}$$

where $\mathbf{f}_p$, $\mathbf{f}_v$ stand respectively for the pressure and viscous forces per unit mass, and $\mathbf{g}$ represents the external forces (per unit mass). The same expression can be also used to write the discretized form of the momentum equation (2.28), with $\mathbf{f}_p$, $\mathbf{f}_v$ and $\mathbf{g}$ being the SPH discretization of the various contributions.

The acceleration for a single particle is thus $\mathbf{a} = \frac{D\mathbf{u}}{Dt}$, and using the superscript to denote the time step, the velocity at step $n+1$ is obtained, using the Euler's integration method, as

$$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \mathbf{a}^{(n)}\Delta t. \tag{5.8}$$

From equation (2.28), the viscous term can be written in a compact form as

$$\mathbf{f}_v = \sum_\alpha k_{\alpha\beta} F_{\alpha\beta} (\mathbf{u}_\beta - \mathbf{u}_\alpha) \tag{5.9}$$

with $k_{\alpha\beta} = \frac{2\mu_{\alpha\beta}}{\rho_\alpha \rho_\beta} m_\alpha$, and where we write the relative velocity in full form (rather than the common SPH shorthand notation) for reasons that will be apparent momentarily.

Therefore, from (5.7), (5.8) and (5.9), the temporal evolution of the particle velocity can be obtained as

$$\begin{aligned} \mathbf{u}_\beta^{(n+1)} &= \mathbf{u}_\beta^{(n)} + \Delta t(\mathbf{f}_p + \mathbf{f}_v + \mathbf{g}) = \\ &= \mathbf{u}_\beta^{(n)} + \Delta t(\mathbf{f}_P + \mathbf{g}) + \Delta t \left(\sum_\alpha k_{\alpha\beta} F_{\alpha\beta} (\mathbf{u}_\beta^{(n)} - \mathbf{u}_\alpha^{(n)}) \right) \end{aligned} \tag{5.10}$$

Equation (5.10) constitutes an explicit integration of the velocity, since the new velocity can be computed directly from the previous velocity. We introduce our semi-implicit integration scheme by computing the viscous part of (5.10) using the *new* velocity instead of the previous one, obtaining:

$$\mathbf{u}_\beta^{(n+1)} = \mathbf{u}_\beta^{(n)} + \Delta t(\mathbf{f}_P + \mathbf{g}) + \Delta t \left(\sum_\alpha k_{\alpha\beta} F_{\alpha\beta} (\mathbf{u}_\beta^{(n+1)} - \mathbf{u}_\alpha^{(n+1)}) \right) \tag{5.11}$$

and, after collecting the new velocities on the left hand side and reordering, we get:

$$\left(1 - \Delta t \sum_{\alpha} k_{\alpha\beta} F_{\alpha\beta}\right) \mathbf{u}_{\beta}^{(n+1)} + \Delta t \sum_{\alpha} k_{\alpha\beta} F_{\alpha\beta} \quad \mathbf{u}_{\alpha}^{(n+1)} = \\ = \mathbf{u}_{\beta}^{(n)} + \Delta t(\mathbf{f}_P + \mathbf{g}). \quad (5.12)$$

This can be written in vector form as a set of three linear systems, one for each of the space dimensions:

$$\begin{aligned} \mathbf{A}\mathbf{v}_x &= \mathbf{b}_x, \\ \mathbf{A}\mathbf{v}_y &= \mathbf{b}_y, \\ \mathbf{A}\mathbf{v}_z &= \mathbf{b}_z \end{aligned} \quad (5.13)$$

where

$$\mathbf{v}_x = \mathbf{u}_x^{(n+1)} = (u_{x1}^{(n+1)}, u_{x2}^{(n+1)}, u_{x3}^{(n+1)}, \ldots, u_{xN}^{(n+1)})^T, \quad (5.14)$$

with N the number of particles,

$$\mathbf{b}_x = \mathbf{u}_x^{(n)} + \Delta t(f_{xp} + g_x) = \\ = (u_{x1}^{(n)}, u_{x2}^{(n)}, u_{x3}^{(n)}, \ldots, u_{xN}^{(n)})^T + \Delta t(f_{p_x} + g_x), \quad (5.15)$$

and analogously for $\mathbf{v}_y, \mathbf{b}_y$ and $\mathbf{v}_z, \mathbf{b}_z$.

The *viscous term* matrix $\mathbf{A}$ is the same for the three systems and, under the hypotheses of compact support, its entries for a fluid particle are

$$a_{\beta\beta} = 1 - \Delta t \sum_{\alpha} k_{\alpha\beta} F_{\alpha\beta} \quad (5.16)$$

for the diagonal terms, with the summation extended to all neighbors, and

$$a_{\alpha\beta} = \begin{cases} 0, & \text{if } \alpha \text{ is not a neighbor of } \beta \ , \\ \Delta t \ k_{\alpha\beta} F_{\alpha\beta}, & \text{if } \alpha \text{ is a neighbor of } \beta. \end{cases} \tag{5.17}$$

for the off-diagonal terms.

The matrix rows corresponding to a boundary particle will have $a_{\beta\beta} = 1$, and $a_{\alpha\beta} = 0$ for $\alpha \neq \beta$. Since the velocity of boundary particles is imposed a priori from outside the linear system resolution, their contribution to the fluid particles can be kept in the right hand side of the system. The matrix is then symmetric if $k_{\alpha\beta}$ is symmetric, i.e. if m_α does not depend on α, which is the case for example if there is only a single fluid, and all discrete particles have the same mass. Multi-fluid simulations are thus not considered in this discussion. Moreover, for the current application, boundaries are always considered not moving (i.e. with zero velocity) so their contributions can be removed even from the known terms.

Linear system solvability We can show that, under the assumption that the smoothing kernel is monotonically decreasing in r (which is the case for all the choices of kernels typically employed in SPH), the coefficient matrix in our linear systems is *diagonally dominant*,

$$|a_{\beta\beta}| > \sum_{\alpha\neq\beta} |a_{\alpha\beta}|. \tag{5.18}$$

To prove this, observe that with a monotonically decreasing kernel, we have $F(\cdot) < 0$. Moreover, since the density, mass and dynamic viscosity of each particle are positive, we have $k_{\alpha\beta} > 0$, so that $k_{\alpha\beta}F_{\alpha\beta}$ is negative by construction, and $|a_{\alpha\beta}| = -a_{\alpha\beta}$ for $\alpha \neq \beta$. Additionally, F is identically zero outside the compact support, so that the right-hand side of (5.18) equals the summation in the right-hand side of (5.16).

We thus have that

$$|a_{\beta\beta}| = a_{\beta\beta} = 1 + \sum_{\alpha\neq\beta} |a_{\alpha\beta}| > \sum_{\alpha\neq\beta} |a_{\alpha\beta}|\,, \tag{5.19}$$

which proves the diagonal dominance of the matrix. Indeed, we additionally know that the diagonal elements are exactly one more than the sum of the absolute values of the off-diagonal elements. This is also true for boundary particles, for which the off-diagonal elements are zero.

We thus have a symmetric, diagonally dominant coefficient matrix in our linear systems, sufficient conditions for the solvability of the system. Since the order of the matrix is equal to the number of particles, which may be in the millions in practical applications, a numerical resolution method is generally necessary. Our choice is to use the Conjugate Gradient CG method [45], which is guaranteed to converge thanks again to the symmetry and diagonal dominance of the coefficient matrix.

5.3.6 Implementation of the semi-implicit scheme in GPUSPH

The predictor–corrector scheme in GPUSPH can be adapted to a semi-implicit scheme in the following way:

1. compute the inviscid acceleration $\bar{\mathbf{a}}$, i.e. the acceleration evaluated without considering the viscous term, at instant n, and the density and temperature derivative as done in section 5.3.5 for the explicit scheme:

   a) $\bar{\mathbf{a}}^{(n)} = \bar{\mathbf{a}}(\mathbf{x}^{(n)}, \mathbf{u}^{(n)}, \rho^{(n)}, T^{(n)})$,

   b) $\dot{\rho}^{(n)} = \dot{\rho}(\mathbf{x}^{(n)}, \mathbf{u}^{(n)}, \rho^{(n)}, T^{(n)})$,

   c) $\dot{T}^{(n)} = \dot{T}(\mathbf{x}^{(n)}, \mathbf{u}^{(n)}, \rho^{(n)}, T^{(n)})$

2. compute half-step intermediate positions, velocities, densities and temperatures:

   a) solve the three systems

   $$\mathbf{A}^{(n)}_{\frac{\Delta t}{2}} \mathbf{u}^{(n\star)} = \mathbf{u}^{(n)} + \bar{\mathbf{a}}^{(n)} \frac{\Delta t}{2},$$

   where $\mathbf{A}^{(n)}_{\frac{\Delta t}{2}}$ is the viscous term matrix built using positions and densities at step n and a time-step of $\Delta t/2$;

   b) $\mathbf{x}^{(n\star)} = \mathbf{x}^{(n)} + \mathbf{u}^{(n)} \frac{\Delta t}{2}$,

   c) $\rho^{(n\star)} = \rho^{(n)} + \dot{\rho}^{(n)} \frac{\Delta t}{2}$,

   d) $T^{(n\star)} = T^{(n)} + \dot{T}^{(n)} \frac{\Delta t}{2}$,

3. compute corrected inviscid accelerations, density and temperature derivatives
   a) $\bar{\mathbf{a}}^{(n\star)} = \bar{\mathbf{a}}(\mathbf{x}^{(n\star)}, \mathbf{u}^{(n\star)}, \rho^{(n\star)}, T^{(n\star)})$,
   b) $\dot{\rho}^{(n\star)} = \dot{\rho}(\mathbf{x}^{(n\star)}, \mathbf{u}^{(n\star)}, \rho^{(n\star)}, T^{(n\star)})$,
   c) $\dot{T}^{(n\star)} = \dot{T}(\mathbf{x}^{(n\star)}, \mathbf{u}^{(n\star)}, \rho^{(n\star)}, T^{(n\star)})$,
4. compute new positions, velocities, densities and temperatures:
   a) compute the new velocity solving:
   $$\mathbf{A}^{\star}_{\Delta t}\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \bar{\mathbf{a}}^{(n\star)}\Delta t$$
   where $\mathbf{A}^{\star}_{\Delta t}$ is the viscous term matrix computed using positions and densities at step $n\star$ and a time-step of Δt;
   b) compute the new position using the trapezoidal rule
   $$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + (\mathbf{u}^{(n)} + \mathbf{u}^{(n+1)})\Delta t/2.$$
   c) $\rho^{(n+1)} = \rho^{(n)} + \dot{\rho}^{(n\star)}\Delta t$.
   d) $T^{(n+1)} = T^{(n)} + \dot{T}^{(n\star)}\Delta t$.

Notice that the position update is done after the final velocity update, unlike in the explicit case. This is particularly important in the corrector step: in the explicit case the position uses a dedicated velocity integration (point 4a, section 5.3.5), the cost of which is very low; for the implicit case this would require an additional resolution of the linear system, and the result would not be the same as using the trapezoidal rule.

Computational issues

To solve each of the linear systems that occur during the integration time-step, we use the CG method. Some of the issues related to this aspect are discussed in what follows.

Memory consumption Due to the iterative nature of the method, it would be convenient to store both the coefficient matrix $\mathbf{A}$ and the right-hand sides of each of the three systems. However, the memory consumption of the matrix, even considering its sparseness, is of order $N \times M$, where N is the number of particles and M the maximum number of neighbors per particle, which depends on the diameter of the compact support of the kernel. (A full neighborhood would have to be allocated for each particle, even though individual particles may have less neighbors than the maximum, because this makes memory accesses much faster on GPU.)

With a typical kernel radius of 2 and a smoothing factor of 1.33, we have $M = 128$, which would require 512 additional bytes per particle to store single-precision floating-point coefficients. This would reduce the maximum number of particles in a simulation on a single device to less than half of what is currently possible.

Therefore, in GPUSPH we only precompute the right-hand side of each linear system. Since the matrix only appears in matrix/vector products, we compute its coefficients dynamically for each needed product. This has a higher computational cost, as each coefficient needs to be generated repeatedly during a resolution, but it allows us to run simulations with many more particles even on lower-end GPUs.

With the semi-implicit scheme we lose the benefit of embarrassingly parallel integration, but since this is replaced by linear algebra operations (dot products and matrix/vector products), we can still benefit from the parallel computational power of GPUs. Hence, GPUSPH implements the entire CG on GPU.

Stopping criteria for the conjugate gradient method As known from the literature, in exact arithmetic the Conjugate Gradient (CG) method converges to the exact solution in at most N iterations, N being the order of the system.

In most practical applications of SPH, N would be in the orders of millions or more, and computations are done in floating-point, potentially even single-precision, therefore it is necessary to establish a numerical stopping condition.

Since our aim is to solve $\mathbf{A}\mathbf{x}_k = \mathbf{b}_k$, we can consider the solution reached numerically when $\mathbf{A}\mathbf{x}_k$ is numerically indistinguishable from $\mathbf{b}_k$, i.e. $\|\mathbf{b}_k - \mathbf{A}\mathbf{x}_k\| < \|\mathbf{b}_k\|\varepsilon_M$ where ε_M is the machine epsilon (the special case of null $\mathbf{b}_k$ is handled separately by setting $\mathbf{x}_k$ to the null vector as well). We observe that the computational cost of checking the condition is low, since the residual $\mathbf{r}_k = \mathbf{b}_k - \mathbf{A}\mathbf{x}_k$ is computed as part of the CG algorithm.

More in detail, three stopping conditions are employed:

1. the norm of the residual drops below the threshold specified above;

2. the number of iterations, k, becomes larger than a given limit (we use for this work $k_{\max} = 200$ iterations, but a different value can be chosen at any time);

3. the residual norm stops decreasing. In this case we store the last solution until the convergent trend is restored or conditions 1 or 2 apply.

and the CG is interrupted as soon as any of the three conditions is satisfied. While condition 1 ensures that the requested level of approximation has been reached, conditions 2 and 3 stop the CG execution, leaving a residual bigger than $\mathbf{r}_k$. In this case, the GPUSPH user is notified of the event and informed of the final value of the residual. Any improvement of the result can be sought acting on k_{max}.

For each CG execution we need to solve three linear systems, one for each spatial dimension; these are solved simultaneously (thus reducing the cost of the dynamic computation of the coefficient matrix), and the halting condition must be satisfied for each dimension on the same iteration.

Conjugate gradient initial conditions The CG method is an iterative solver, and choosing the appropriate starting point can significantly reduce the number of iterations needed for convergence.

Considering that between two integration steps the configuration in the particles velocity undergoes only small variations, we take as initial condition the velocity at the previous integration step. This is particularly efficient for stationary or nearly stationary flows.

For the correction step, another option is to use an intermediate velocity between the previous and predicted one. Our tests indicate that this choice is often a good one, leading to faster convergence, but this is not always the case. Further analysis is required so that the best starting condition can be chosen programmatically.

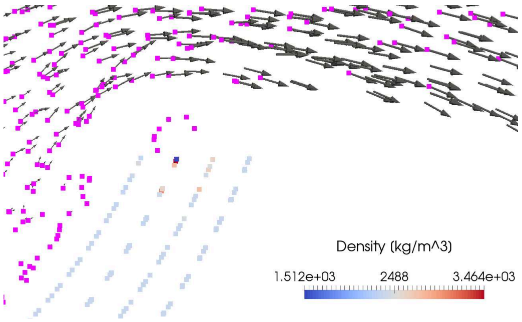
Semi-implicit scheme and time step When using our semi-implicit integration scheme, we can disregard the stability condition for the viscosity, even when in (5.5) $h < h^\star$, which allows us to use much larger time-steps for high viscosity/very fine resolutions. However, the increase in the time-step is offset by the much higher computational cost of the single time-step. In this sense, computationally, the semi-implicit scheme only becomes convenient when h is at least an order of magnitude smaller than the critical value (or conversely the viscosity is at least an order of magnitude larger than the critical viscosity).

5.3.7 Dynamic boundaries and density integration

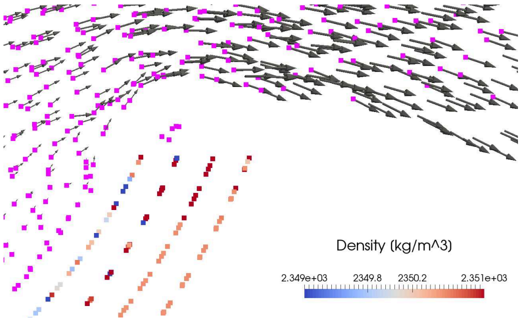
Using dynamic boundary conditions the density of boundary particles evolves as described in section 2.4.1. In the explicit integration scheme the derivative of the density is computed using the velocity at step n. Highlighting the time-step, we can then write (2.27) as

$$\frac{D\rho_\beta^{(n)}}{Dt} = \sum_\alpha m_\alpha \mathbf{u}_{\alpha\beta}^{(n)} \nabla_\beta W_{\alpha\beta} \tag{5.20}$$

In our experiments, with the increase in time step allowed by the implicit scheme, this formula for the density derivative leads



(a) Density derivative evaluated using the $\mathbf{u}^{(n)}$ velocity: some boundary particles undergo a strong pressure drop that causes excessive attraction of fluid particles, leading to instability of the simulation.



(b) Density derivative evaluated using the $\mathbf{u}^{(n\star)}$ velocity: the decrease in boundary density is mitigated and the fluid particles attraction is weaker.

Figure 5.3: Upper view of a portion of the crucible opening (sliced): using $\mathbf{u}^{(n\star)}$ rather than $\mathbf{u}^{(n)}$ helps avoiding density issues due to Dynamic boundaries. Boundary particles are colored by pressure while fluid particles are colored in magenta. Arrows indicate particles velocity.

to instabilities, as the boundary fails to react sufficiently to the approaching fluid (Fig. 5.3a).

To solve this, we compute the density derivative from the value of the velocity used to integrate the position, i.e. $\mathbf{u}^{(n\star)}$ on the predictor step, $(\mathbf{u}^{(n)} + \mathbf{u}^{(n+1)})/2$ in the corrector step, which leads to improved density evolution and a more correct response from the boundary (Fig. 5.3b).

This choice is made in analogy with summation density, a method sometimes used in SPH to compute density without resorting to integration of the mass continuity equation. Indeed, applying (1.5) to the density, we have that ρ can be computed as $\rho_\beta = \sum_\alpha m_\alpha W_{\alpha\beta}$.

While this expression is not of practical use in free-surface problems, since it underestimates the density near the free surface, it implies that the density can be derived from the geometric distribution of the particles (and their mass). Using the same velocity to integrate the position and to compute the density derivative preserves this consistency when using the mass continuity equation.

5.3.8 Semi-implicit scheme for Dummy boundaries

When using dummy boundary conditions, the velocity of the boundary particles is given by the condition:

$$\mathbf{U}_\beta = 2\mathbf{V}_\beta - \frac{\sum_\alpha W_{\alpha\beta}\mathbf{U}_\alpha}{\sum_\alpha W_{\alpha\beta}}$$

where $\mathbf{V}$ is the prescribed wall velocity, and the summations are extended only to fluid particles (i.e. boundary particles do no interact with other boundary particles). We can rewrite the condition as

$$\left(\sum_{\alpha} W_{\alpha\beta}\right)\mathbf{U}_{\beta} + \sum_{\alpha} W_{\alpha\beta}\mathbf{U}_{\alpha} = 2\left(\sum_{\alpha} W_{\alpha\beta}\right)\mathbf{V}_{\beta}$$

to better highlight the diagonal term of the matrix $(\sum_{\alpha} W_{\alpha\beta})$ and the off-diagonal terms $W_{\alpha\beta}\mathbf{U}_{\alpha}$.

Hence, the linear system $\mathbf{AU} = \mathbf{B}$ has a $\mathbf{A}$ matrix which is still symmetric, and while it is still diagonally dominant, the condition $|\mathbf{A}_{\beta\beta}| \geq \sum_{\alpha} |\mathbf{A}_{\alpha\beta}|$ is only satisfied by the equality in the rows corresponding to the boundary particles.

However, the matrix is *weakly chained diagonally dominant*, since for every row where only the equality holds, which corresponds to a boundary particle, any off-diagonal element chains with a row which is diagonally dominants, since boundary particles only have fluid particles as neighbors, and the fluid particles have diagonally dominant rows.

Solving the linear system

One major issue coming from the use of the dummy boundary model is that the matrix $\mathbf{A}$ is not symmetric. This prevents the Conjugate Gradient algorithm to be used for the resolution of the linear system. As an alternative, the Stabilized Biconjugate Gradient Method (BCGSTAB) has been used, introducing two major issues, one related to the simulation time, since the

BCGSTAB involves much more computations than CG, and another to the accuracy. In fact, we have found the BCGSTAB to be very sensitive to the numerical precision. Because of the finite precision, any numerical solver can improve the estimated solution up to a certain accuracy; after that point, any other iteration will not be able to further reduce the residual. Despite the residual descent is faster for the BCGSTAB, it stalls several orders before than the CG. We have also seen that the passing from double to single precision, the BCGSTAB has the higher loss of performance. We experienced in some cases that the residual left was too big, introducing spurious behaviors in the simulation, like the missed integration of small forces. Since in GPUs we always work in single precision, this method needs to be revised in its implementation or substituted with some others. A next candidate that is currently under examination is the Jacobi method.

Chapter 6

GPUSPH and lava flows

Lava flows are complex fluids showing a number of behavior that are not common to other flows. Looking at the properties of the SPH method, that we have discussed in section 1.2.1, and at the characteristics of lava flows, discussed in chapter 3, we can conclude that SPH can be considered the best method to be used for the simulation of this kind of phenomenon. The SPH method itself constitutes only the basis for the simulation of lava flows, to get a more complete model, other characteristic of lava and lava flows need to be taken into account. In this chapter we will see the implementation in GPUSPH of several models interpreting some characteristics or behaviors of lava flows. To prove the correct implementation we will show some simulations reproducing systems which behavior is intuitive or already known.

6.1 Non-Newtonian rheologies

To investigate and test the rheology of lava, GPUSPH has been equipped with the Herscher-Bulkley (HB) rheological model, able to reproduce the main generalized Newtonian rheologies. Because of stability purposes, the HB model is not directly adopted in GPUSPH due to its discontinuity. We rather use the regularized model proposed by [95], expressed as

$$\mu_{app}(\dot{\gamma}) = \mu_0 e^{t_1 \dot{\gamma}} + \tau_0 \frac{1 - e^{-m\dot{\gamma}}}{\dot{\gamma}} \tag{6.1}$$

Here, μ_0 is the limiting viscosity, measured in Pa s and t_1 and m are model parameters, measured in seconds. The value of m should be large enough to guarantee a large, but finite, apparent viscosity at vanishing shear rates, the limit being $\mu_0 + m\tau_0$. The value of t_1 on the other hand determines whether the fluid is shear thinning ($t_1 > 0$) or shear thickening ($t_1 < 0$).

When using Non-Newtonian rheology in GPUSPH, each particle is assigned its viscosity, that is computed using 6.1. The strain rate $\dot{\gamma}$ is obtained from the velocity fields, using the equation (2.8). The gradients of the three velocity components, involved in the (2.8), are evaluated iterating over the neighboring particles according to the SPH discretization of gradients, given by equation (1.10). The value $|\dot{\gamma}|$ is finally achieved as the second invariant of $\dot{\gamma}$, as defined by equation (2.20).

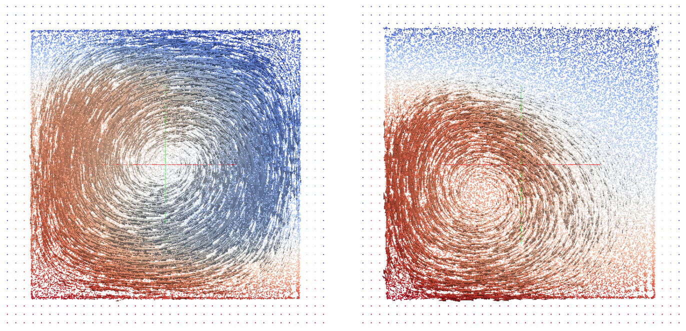
For avoiding divisions by zero, before evaluating the (6.1) a threshold is applied to $|\dot{\gamma}|$, limiting it to a minimum positive value. By default we use $|\dot{\gamma}|_{min} = 10^{-4}$.

6.2 Temperature dependent viscosity

We have seen 3.2.1 that one of the main parameters that determines the behavior of a lava flow is the viscosity. For any fluid, the viscosity depends on the temperature, and this dependence is very important when studying lava flows, since the temperatures involved span very large intervals. GPUSPH has then been equipped with a model of temperature dependent viscosities. When setting the problem we can define a law that links the kinematic viscosity of a particle to its temperature. Additionally, in case of non-Newtonian fluids, also the other rheological parameters, like the yield strength, have been given a dependence on the the temperature. The law of dependence can be defined independently in the problem.

6.2.1 The Rayleigh-Bénard problem

We test the interaction of the thermal and mechanical models by simulating a thermal convection. We consider a box containing the fluid, with adiabatic walls, a heated bottom plate and a cooled top plate. As effect of the heating, the fluids gets in motion causing the inception of a Rayleigh-Bénard convective



(a) *Rayleigh-Bénard convective cell.* (b) *Rayleigh-Bénard convection with temperature dependent viscosity.*

Figure 6.1: Rayleigh-Bénard convection cells. Particles are colored by temperature. Arrows represent particles' velocities.

cell.

The bottom plate is at temperature $T_b = 100\,\mathrm{K}$ and the top plate at $T_t = 0\,\mathrm{K}$.

For the thermal dependence we use the law

$$\nu = e^{-0.1T} \tag{6.2}$$

The kinematic viscosity at temperature $T = T_b$ is $\nu_b = 1\,\mathrm{m}^2/\mathrm{s}$ while at temperature $T = T_t$ is $\nu_t \approx 4.5^{-5}\,\mathrm{m}^2/\mathrm{s}$.

Figure 6.1 show comparison between a convective cell with temperature independent viscosity (figure 6.1a) and with the thermal dependence described above in [51], that alters the behavior of the convective cell as shown in figure 6.1b.

6.3 Thermal surface dissipation

Surface thermal dissipation plays an essential role in the cooling process of a lava flow and then on its emplacement.

Thermal dissipation is modeled both at the contact with the ground, using equation (3) and on the free surface. The latter occurs according to two phenomena:

1. **Thermal radiation**: according to Stefan-Boltzmann law, we express the radiated heat per unit surface as:

$$\frac{DT}{Dt} = \frac{K_B \kappa \epsilon}{m c_p}(T^4 - T_a^4) \tag{6.3}$$

   with K_B the Stefan-Boltzmann constant, κ the thermal conductivity, ϵ the emissivity, m the mass, c_p the specific heat at constant pressure and T_a the ambient temperature.

2. **Air convection** We do not model air particles, but we account the heat lost due to air convection by means of a convection coefficient , according to the following law, per unit surface:

$$\frac{DT}{Dt} = \frac{\eta}{m c_p}(T - T_a) \tag{6.4}$$

   where η is a convection coefficient, measured in $W/(m^2K)$, m is the mass, and T_a the ambient temperature.

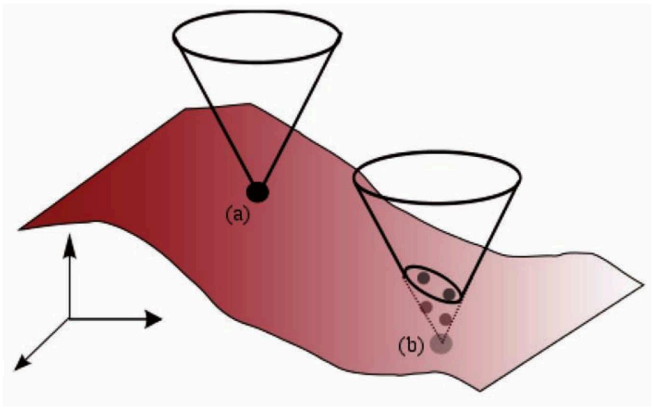


Figure 6.2: Free surface detection: particle a belongs to the surface, while particle b belongs to the interior of the flow.

To detect the particles that belong to the free surface [42, 8] for every particle α we consider the cone C_α with vertical axis and vertex on particle α; let θ be the angular aperture of the cone, then

- If C_α does not contain any other particles, then the particle α belongs to the free surface (Figure 6.2, particle a)

- If C_α does contain other particles, then the particle α does not belong to the free surface (Figure 6.2, particle b)

6.3.1 Free surface estimation

Once the free surface particles are detected, in order to apply the models for thermal dissipation ((6.3) and (6.4)) we need to compute their surface. We tried two main approaches, a first geometrical one, where the particle surface is consider to be a square with the average inter-particle spacing (Δp) as particle side. But dealing with flows that usually evolve towards under-resolved conditions, as spreading flows, we would underestimate the surface in the regions where the simulation becomes more rarefied (i.e. where we have a thin flow). To face this problem we compute the particle surface using the numerical volume [47].

$$V_\beta = \frac{1}{\sum_\alpha W_{\alpha\beta}} \tag{6.5}$$

with W the smoothing kernel, and considering a spherical particle volume and a circular particle surface.

6.4 Simulating a setup for lava experiments

The ability to reproduce the conditions that can be found in a real context involving a lava flow, sets the basis for a study aimed at understanding the properties of lava and lava flows. Figure 6.3a shows the pouring of molten lava onsloping ground, a typical set-up used to test the rheological properties of lava in analogical experiments. The simulation reproduces setup composed by a rotating crucible and a channel with semicircular section, used in a real experiment [26], partially shown in picture 6.3b.

The simulation of the rotating crucible is controlled by means od the ODE (Open Dynamic Engine) [71], and it is possible to be assigned of a well defined angular velocity trend, in order accurately reproduce any real experiment.

The fluid has been simulated with realistic lava parameters, the dynamical viscosity of the lava depends on the temperature following the Vogel- Fulcher-Tammann equation [18], with coefficients determined experimentally [H. Dietterich, private communication]

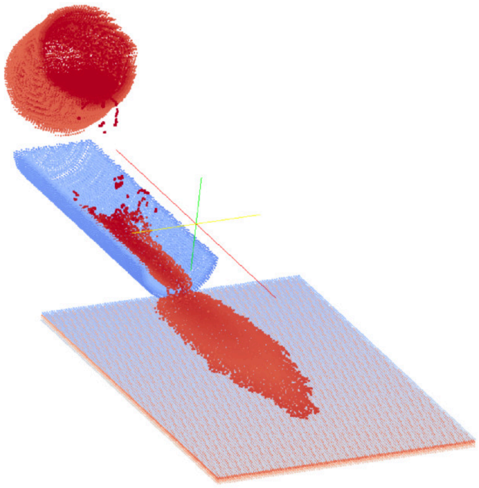
$$\log_{10}(\mu_{exp}) = 5.94 + \frac{5500}{T - 610}. \tag{6.6}$$

The density is $\rho = 2350\,\mathrm{kg/m^3}$. The thermal model has the same parameters adopted in 7.4.

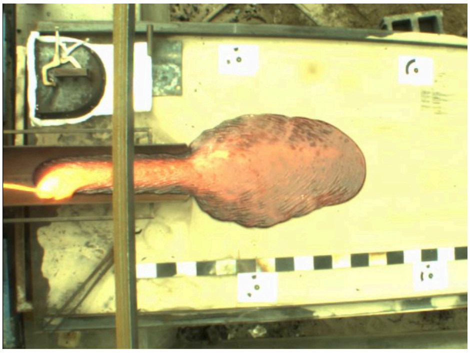
6.4.1 Interaction with structures and moving bodies

A possible study case could be the interaction of a lava flow with a structure. A fixed structure can be obtained by using the classical boundary implementation, while improved simulation quality in order to give more realistic reaction of the obstacle, can be obtained using the ODE library [71], already embedded in GPUSPH.

Here we run a numerical simulation to analyze the distribution of the force exerted by the lava onto a structure. Figure 6.4 shows a simulation of the force exerted onto a wall, an analysis



(a) The setup simulated with GPUSPH



(b) a top view of the flow obtained with the real setup (courtesy of H. Diettrich, USGS)

Figure 6.3: Setup for the study of the rheological properties of lava.

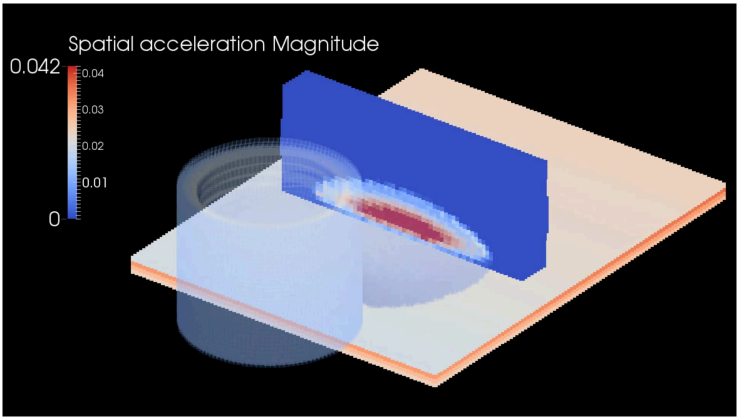
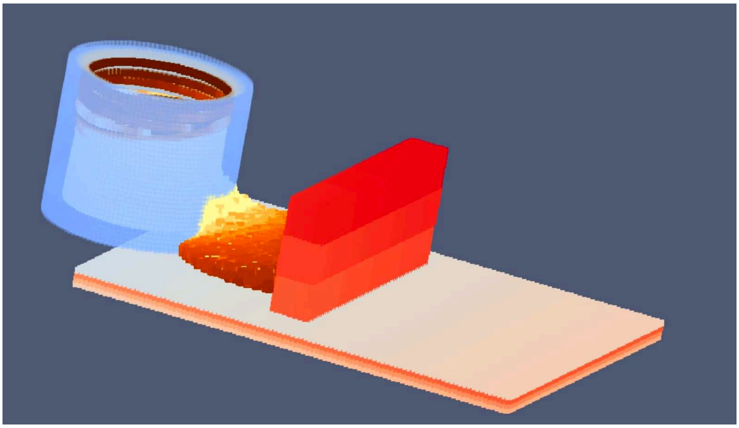


Figure 6.4: Lava flow interacting with a wall. The colors represent the force per unit mass [m^2/s] exerted by individual particles. Passing through the particle surface and density it is possible to visualize also the exerted pressure.

that would be very difficult to perform practically. By building a realistic model of the wall, it is possible to relate the entity of the volcanic event to any damage impressed to the structure. This kind of study ican be useful in design phase to reduce the vulnerability of the structure as a strategy to reduce volcanic risk, discussed in 3.1.2. Figure 6.5 shows the breaking down of the wall, subsequent the action of the lava flow. One new aspect remarked by this simulation is that simulating with SPH is not just matter of studying the kinematic of particles, but we are


(a) First stage: the lava flow reaches the wall and starts gathering behind it.

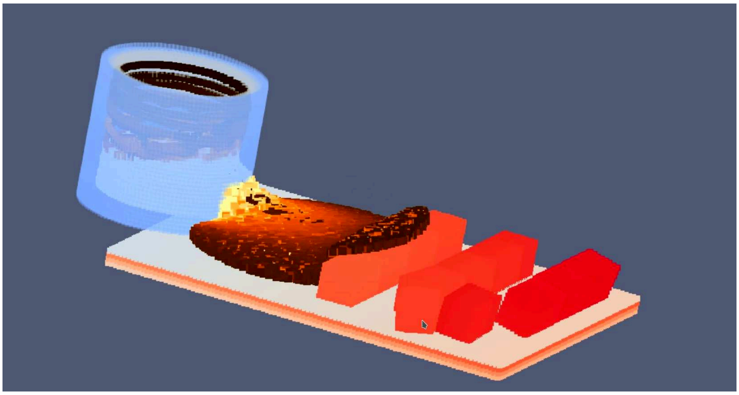

(b) Second stage: the force exerted by the lava destroys the wall. The dynamic of the individual bricks is modelled by means of the ODE library.

Figure 6.5: Simulation of a lava flow breaking a wall down. Only the colors of the lava are indicative of the temperature.

also modelling dynamical of the interactions.

6.5 Multi-fluid simulation

The classical SPH formulation introduced in 1 presents a number of issues when directly applied to multi-fluid problems. Large density ratios and large variations in mass distribution can introduce significant numerical instabilities, especially near the interface between multiple fluids. A solution to this issue is to move from a density-based formulation to one based on volumes [7]. Different approaches are possible, here we are going to use the formulation proposed by Grenier [35, 36, 37].

Let us call ω the actual particle volume, the continuity equation is written in terms of the Jacobian $J = \frac{\omega}{\omega^0} = \frac{\rho^0}{\rho}$ where ρ^0 and ω^0 indicate the initial values of the particle density and volume, respectively.

The continuity equation (2.1) is rewritten as

$$\frac{D \log J}{Dt} = \nabla \cdot \mathbf{u} \tag{6.7}$$

which is discretized with the Shepard-corrected divergence

$$\frac{D \log J}{Dt} = \frac{1}{\sigma} \sum_{\alpha} (\mathbf{u}_\alpha - \mathbf{u}_\beta) \cdot \nabla_\alpha W_{\alpha\beta} \tag{6.8}$$

where σ is defined as

$$\sigma(x) = \sum_{\alpha} W_{\alpha\beta}. \tag{6.9}$$

The density ρ_α of a particle is computed as a smoothed mass M divided by the particle volume, where M is computed with a Shepard-corrected kernel limited to the particles belonging to the same fluid:

$$M_\beta = \frac{\sum_{k \in K_\beta} m_k W_{\beta k}}{\sum_{k \in K_\beta} W_{\beta k}} \tag{6.10}$$

where K_β is the set of indices of particles having the same fluid type as particle β. If all particles belonging to the same fluid have the same mass, the smoothing can be skipped as it's algebraically equal to the particle mass m_β.

For the momentum equation, the pressure contribution is then given by

$$-\frac{1}{\rho_\beta} \sum_\alpha \left(\frac{P_\beta}{\sigma_\beta} + \frac{P_\alpha}{\sigma_\alpha} \right) \nabla_\beta W_{\alpha\beta} \tag{6.11}$$

while the viscous force is given by

$$\frac{1}{\rho_\beta} \sum_\alpha \frac{2\mu_\beta \mu_\alpha}{\mu_\beta + \mu_\alpha} \left(\frac{1}{\sigma_\beta} + \frac{1}{\sigma_\alpha} \right) \nabla_\beta W_{\alpha\beta} F_{\alpha\beta} \mathbf{u}_{\alpha\beta} \tag{6.12}$$

6.5.1 Simulation of a lava lamp

Figure 6.6 illustrates a multi-fluid problem based on a lava-lamp: two fluids with opposite conditions in terms of density, rheology and thermal expansion coefficient are contained within an adiabatic box with heated bottom and cooled top. The red fluid has density $\rho_1 = 2\,\text{kg}/m^3$, Bingham rheology with $\tau_0 = 0.001$ Pa, consistency index depending on the temperature as $k(T) = exp(-2T)$ and thermal expansion coefficient $\alpha_1 = 0.6\,\text{K}^{-1}$, while the blue one has $\rho_2 = 1\,\text{kg}/m^3$, Newtonian rheology with kinematic viscosity $\nu_2 = 1\,\text{m}^2/\text{s}$ and $\alpha_2 = 0.1\,\text{K}^{-1}$. At a resolution of 33 particles per meter, about $30,800$ particles are involved with a time-step in the order of 10^{-4} s, and using a single Maxwell Titan X GPU the obtained time-ratio is 1:10.

6.6 Phase transition

Phase transitions are modelled only from a thermal point of view [67] and [8], [42], and in particular we consider the constant temperature during phase transition. The dynamics of solid particles is currently the same as for fluid particles, and the aggregation of solid particles into larger bodies is not modelled.

Each particle carries an additional property, q, representing the fraction of latent heat gained/lost during phase transition: hence, $q = 1$ for solid particles; and $q = 0$ for liquid particles. During integration of the heat equation, if the estimated new temperature, $T^\star_\beta$ of particle β drops below the solidification temperature, T_s, while $q < 1$, then we compute the new latent heat fraction as $q^\star = q + c_{p\beta}(T_s - T^\star_\beta)/L$, modelling latent heat loss; if $q^\star \geq 1$, we set

$T_\beta = T_s - L(q^\star - 1)/c_p$ and $q = 1$, marking the end of the phase transition; otherwise we set $T_\beta = T_s$ and $q = q^\star$, and the particle represents a section of fluid that is in phase transition. In all other cases, temperature evolves normally following the

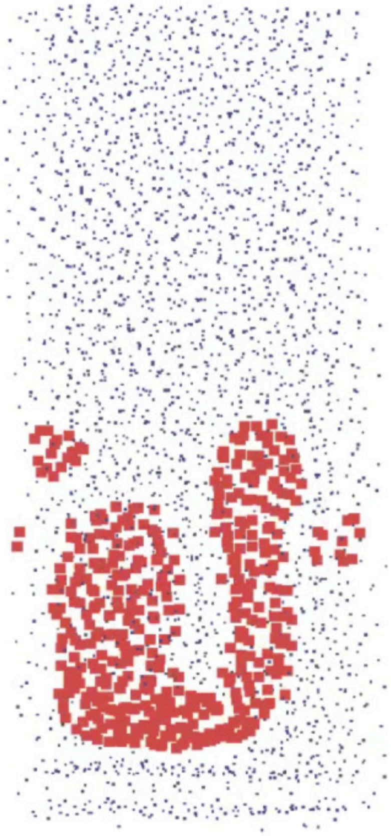
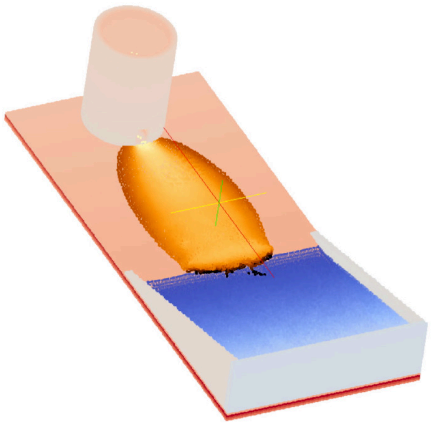


Figure 6.6: Qualitative simulation of a lava lamp. Particles are colored by fluid.

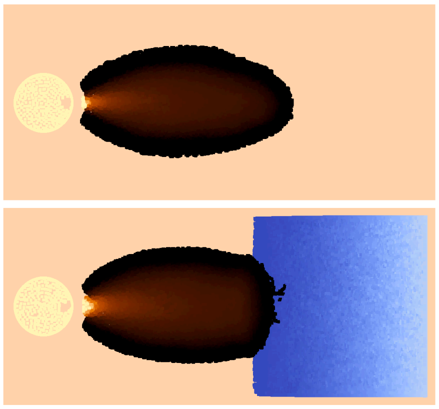
discretized heat equation. Since the phase-change temperature for lava cannot be defined explicitly, the solidus temperature is used for Ts instead. The described approach for phase transition illustrates an important benefit of Lagrangian mesh- less methods such as SPH compared to Eulerian and mesh-based methods. Indeed, in contrast to the other methods where explicit front tracking and evolution models are needed, solidification fronts form and evolve naturally in SPH, and they are implicitly defined by the particles for which $0 < q < 1$.

6.6.1 Simulation of lava-water interaction

Interaction between lava and water is a common phenomenon. Figure 6.7 illustrates a simple case of a lava flow reaching a water basin. We are modelling the thermal behavior of the lava and the water, the formed cooled down and the latter heated up during the interaction. We are linking the thermal model to the mechanical one, by considering a temperature dependent viscosity; in this preliminary tests, simulations have been conducted with a viscosity that is 10 times lower than the experimental viscosity (6.6), due to computational constraints. In this case, the sharp decrease in temperature caused by the contact with water results in a different emplacement compared to the lava flow behavior in absence of water. Thermally modelling the solidification helps to obtain realistic variations of temperature during the phase transition.



(a) A setup for the simulation of lava-water interaction.



(b) Comparison of the lava-water interaction and the dry emplacement.

Figure 6.7: Simulating lava-water interaction with GPUSPH.

Chapter 7

Results and conclusions

In this chapter we will see the results obtained by the development of the SPH method, and more specifically of the GPUSPH simulation engine, that has been described so far in this work. As first we are going to analyze the benefits introduced by the newly implemented semi-implicit integration scheme, either in terms of robustness of the simulation results and in terms of simulation times. Then we will validate the thermal and mechanical models, reproducing some known experiments, and we will compare the results with the reference ones, studying their variations in relation to the main simulation parameters.

7.1 Testing the semi-implicit integrator

We are going to see the application of the semi-implicit integration scheme to the simulation of a lava flow that involves realistic parameters.

We have preliminary validated the new formulation with the classical plane Poiseuille flow in three dimensions, comparing the results obtained with the semi-implicit formulation presented here to those obtained with the explicit integration, expecting little difference in the behavior of the fluid.

The setup is with a channel, periodic in the X and Y directions and 1m thick in the Z direction. A driving force is applied in the X direction, with acceleration magnitude $g = 0.05\,\mathrm{m/s^2}$. The fluid is Newtonian with constant viscosity and at-rest density $\rho_0 = 1000\,\mathrm{kg/m^3}$. The inter-particle distance in the initial setup is such that 32 particles fit in the Z direction.

With a kinematic viscosity of $\nu = 1\,\mathrm{m^2/s}$, the Reynolds number is $Re = 6.25 \cdot 10^{-3}$. The time step in this case in controlled by the sound speed of the fluid, so the semi-implicit and explicit formulations use the same time-step ($\Delta t = 1.407 \cdot 10^{-4}$s). The relative error in the velocity profile of the semi-implicit solution to the explicit solution has norms $L_\infty = 1.078 \cdot 10^{-5}$, $L_1 = 2.573 \cdot 10^{-4}$ and $L_2 = 2.204 \cdot 10^{-9}$. Due to both methods using the same time-step, the explicit integration is about 5 times faster than the semi-implicit one.

With a higher kinematic viscosity ($\nu = 100\,\mathrm{m^2/s}$), resulting in a Reynolds number $Re = 6.25 \cdot 10^{-7}$, the time-step is controlled by the viscous term ($\Delta t = 2.159 \cdot 10^{-6}$ s) in the explicit case. The relative error in the velocity profile in this case has norm $L_\infty = 1.088 \cdot 10^{-5}$, $L_1 = 4.686 \cdot 10^{-4}$ and $L_2 = 6.983 \cdot 10^{-9}$. Thanks to the larger time-step, the semi-implicit formulation ends up being about 10 times faster than the explicit formulation.

We observe that in the higher viscosity example the difference in the velocity profile is slightly higher, but in both cases the results are essentially the same, validating the correctness of the formulation. For a more thorough analysis of both run-times and emplacement, we then study the effectiveness in employing our method when dealing with lava flows.

7.1.1 Problem description

The geometry of the problem is depicted in figure 7.1. The simulated domain size is 80 cm $\times$ 123 cm $\times$ 55 cm, and it is discretized using $\Delta p = 0.008\,\mathrm{m}$; the number of particles involved is 100,923.

Since the maximum fluid velocity is low, the sound speed is evaluated from the hydrostatic condition, by taking the maximum free-fall speed: $c_0 = 20\sqrt{2h_{\max}g}$, that is the speed achieved at the ground by a body falling from a height $h_{\max}$, multiplied by 20 to ensure a weakly compressible state. Rounding the maximum height to $h_{\max} = 2$ m and using a gravity acceleration magnitude $g = 9.81$, we get $c_0 = 20\sqrt{4g} \approx 125.28$ m/s.

The lava is modeled following the experiments reported in [26], using a Newtonian fluid with a temperature dependent viscosity coefficient, that for temperature greater than 610 K is given by

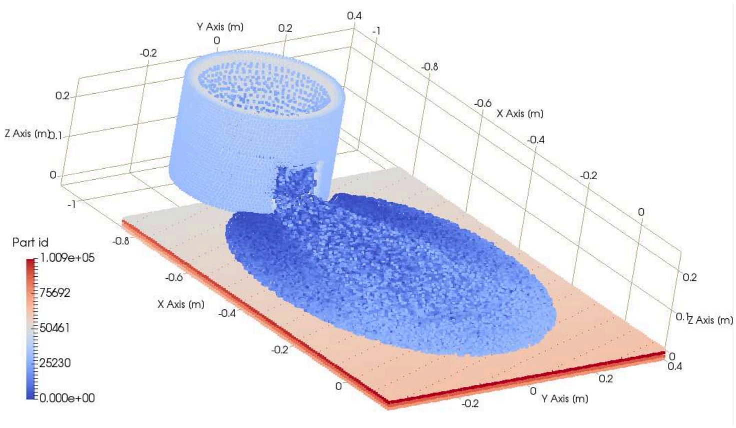


Figure 7.1: Simulation setup: a perforated crucible pouring lava onto a plane inclined of 13.25°. Particles are colored by their identification number.

the law (H. Dietterich, USGS, personal communication):

$$\mu = 10 \exp\left(-5.94 + \frac{5500}{T - 610}\right) \tag{7.1}$$

and has a density $\rho = 2,350\,\mathrm{kg/m^3}$. The initial temperature of lava and of the crucible is $T_0 = 1323\,\mathrm{K}$, leading to a kinematic viscosity $\nu_{T_0} = 0.0252\,\mathrm{m^2/s}$ while the ground temperature is $T_G = 298\,\mathrm{K}$; to use the same viscosity law of the lava, we bound the viscosity level range to $T > 1,111\,\mathrm{K}$ in order to avoid unnecessarily high viscosities. The importance of having an upper bound on the viscosity will be discussed in section 7.1.2. The law we use to describe the useful change of viscosity is then

$$\mu_u = \min\left(\mu(T), \mu(1111)\right). \tag{7.2}$$

The maximum kinematic viscosity we can obtain is thus $\nu_{max} = 46.45\ \mathrm{m}^2/\mathrm{s}$.

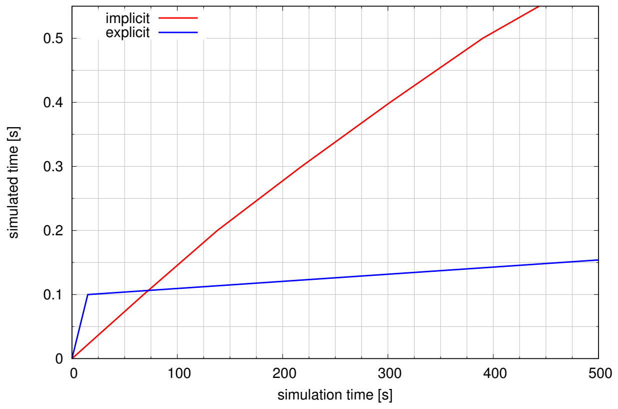
The simulated lava has thermal conductivity $\kappa = 1\,W/(\mathrm{m}\cdot\mathrm{K})$, heat capacity $c_p = 1,600\,\mathrm{J}/(\mathrm{kg}\cdot\mathrm{K})$ and thermal emissivity $\epsilon = 0.96$. Lava solidification is modeled using solidus temperature $T_s = 1,000\,\mathrm{K}$ and latent heat $L = 2.9 \cdot 10^5\,\mathrm{J/kg}$.

7.1.2 Simulation results

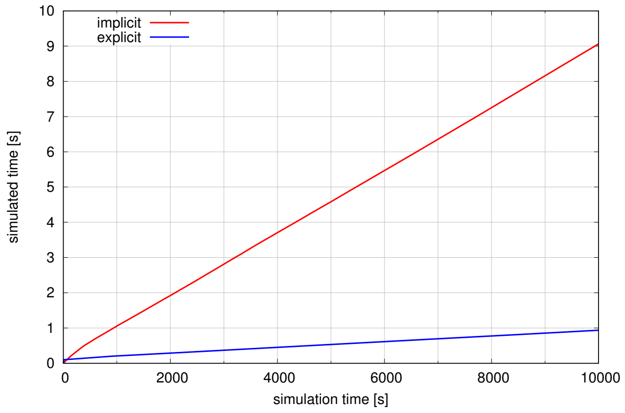
With the given choices of viscosity, density, speed of sound and average inter-particle spacing, we can compute the critical h and μ for the dominance of the viscous stability condition over the sound-speed stability condition. We have then $h^\star \simeq 0.9\,\mathrm{m}$ and $\mu^\star \simeq 1,276\,\mathrm{Pa}\cdot\mathrm{s}$, corresponding to $\nu^\star \simeq 0.54\,\mathrm{m}^2/\mathrm{s}$.

We are therefore about two orders of magnitude below the maximum smoothing length, and two orders of magnitude above the least viscosity for which the viscous stability condition becomes dominant. As we discuss the results in terms of emplacement and performance, we shall see how this justifies the use of the semi-implicit scheme over the explicit integration.

Performance comparison A comparison of the runtime for the explicit versus the semi-implicit integration schemes is shown in figure 7.2. All simulations were run on the same hardware,

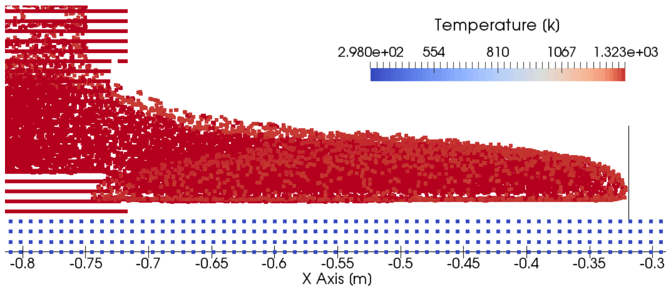


(a) First 4 minutes of runtime

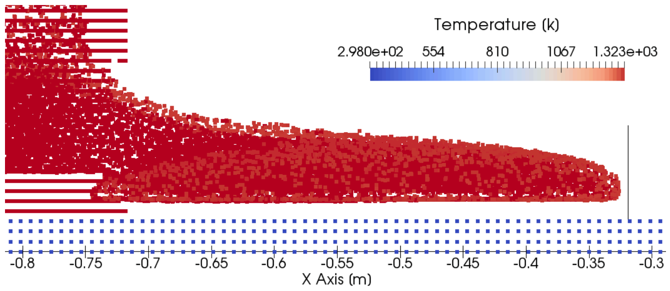


(b) Over 6 hours of runtime

Figure 7.2: Comparison of simulation speed with the explicit and semi-implicit integration scheme

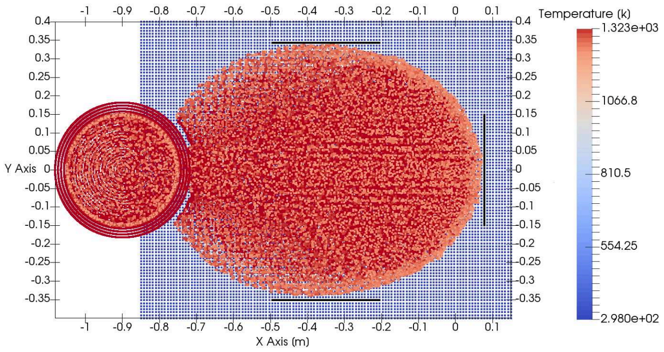


(a) Lava flow simulated with the explicit solver.

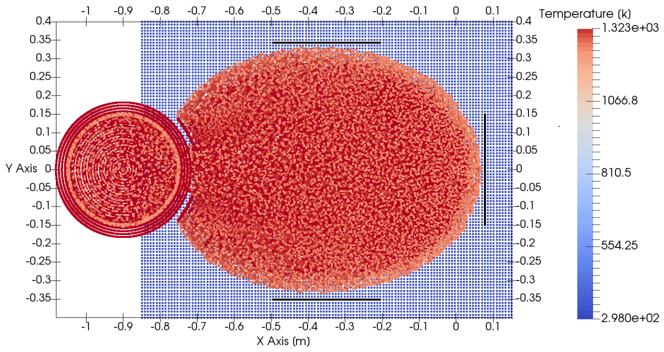


(b) Lava flow simulated with the semi-implicit solver.

Figure 7.3: Comparison of simulated lava emplacements with the explicit and semi-implicit integration schemes at $t = 2$ s. Pictures show a portion of the lateral view.



(a) Lava flow simulated with the explicit solver.



(b) Lava flow simulated with the semi-implicit solver.

Figure 7.4: Comparison of simulated lava emplacements with the explicit and semi-implicit integration schemes at $t = 13$ s. Pictures show a top view of the simulation domain.

with the computations running on an NVIDIA Titan X GPU (Maxwell architecture).

We can observe that with the explicit scheme, the ratio of runtime to simulated time is piecewise constant: in the earliest moments, before the lava touches the floor, the time-stepping is dominated by the sound-speed condition ($\Delta t = 2.5 \cdot 10^{-5}$ s); when the lava starts interacting with the floor at $t = 0.1$s, the viscous condition ($\Delta t = 3.05 \cdot 10^{-7}$ s) becomes dominant (figure 7.2a) and the runtime ratio then remains constant for the rest of the simulation, but much more unfavorable: around 75s of run-time later, this allows the semi-implicit scheme to take over.

In the semi-implicit case, the viscous condition for the time-step is absent, so that the sound-speed time-step is used throughout the simulation. Although the time-step is constant and given by the sound speed condition, we can however observe that in the very initial phase the ratio of runtime to simulated time progressively decreases due to the CG taking more iterations to converge. The reason for this is to be found in the diagonal dominance of the coefficient matrix becoming less significant.

Indeed, we recall from (5.16) and (5.17) that the diagonal dominance is due to the diagonal element being one more than the sum of the off-diagonal elements. As the viscosity grows larger, the off-diagonal elements grow larger in absolute value, and so does their sum, reducing the dominance of the diagonal. The value of particles viscosity is not the only factor affecting the matrix conditioning, but also the amount of particles with high viscosity, i.e. the number of rows in the matrix where the diagonal dominance condition is weaker. This explains the initial rapid loss of performance: as the amount of lava touching the cold ground increases, the change in the matrix conditioning is sufficient to require more iterations from the CG algorithm.

In raw numbers, once a steady state on the simulation speed is reached, the explicit formulation simulates 1s in about $12,326$ s (nearly three and a half hours) while the semi-implicit formulation simulates 1 s in $1,124$ s (about 19 minutes). The gain in simulation time is therefore a factor of about 11.

Figures 7.3a and 7.3b show a lateral view of the emplacement of the lava at $t = 2.5$ s, respectively for the explicit and semi-implicit formulations. Differences can be observed both in the slightly further distance reached by the explicit solver (less than Δp) and in the total thickness.

The discrepancy in the simulated emplacement between the explicit and semi-implicit scheme is still very limited, although it grows over time, as shown in a comparison between Figures 7.4a versus 7.4b, which further highlights the higher number of artifacts in the explicit emplacement (banding of the particle positions in the central part, thinning in the outer rims).

While the difference in total extension can be at least partially explained by the use of a different velocity in the computation of the acceleration (previous versus current step), the artifacts clearly visible in the explicit case are the result of stronger numerical errors, caused by the smaller time-step. Indeed, a lower Δt increases numerical errors through at least two independent aspects:

- when Δt is very close to machine epsilon, a larger part of the time derivative fails to be taken into account during

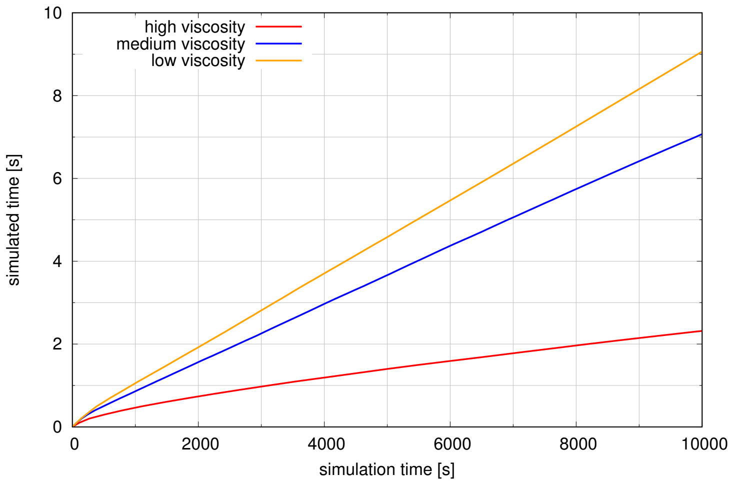


Figure 7.5: Simulation performance for three values of maximum kinematic viscosity: low $= 46.45\text{m}^2/\text{s}$, medium $= 81.93\,\text{m}^2/\text{s}$ and $high = 67,259\,\text{m}^2/\text{s}$.

integration, leading to larger rounding errors per time-step;

- a lower Δt requires a higher number of iterations per simulated time, leading to a further amplification of the numerical error.

Influence of maximum viscosity on performance The bound on the maximum viscosity over temperature, introduced in Section 7.1.1, is selected so as to allow the simulation to run with the original explicit scheme; indeed, the small time step required by higher viscosities could lead to unreasonably long simulation times or even to the impossibility to run calculations at all due to the adopted numerical precision.

With the explicit scheme, in the case of a dominant viscous stability condition, the run-time to simulated time ratio is inversely proportional to the value of the viscosity, since the computational time per time-step is constant and the time-step is inversely proportional. By contrast, in the semi-implicit scheme the time-step is independent from the viscosity, but the computational time of the individual time-step depends on how fast the linear system can be solved, and as explained in Section 7.1.2, higher viscosities lead to a higher condition number for the matrix and thus a slower convergence.

Using the semi-implicit scheme we can afford higher viscosity values and thus a better realization of the actual viscous model. We remark that, while the value reached by the viscosity does not impact the value of the time-step in the semi-implicit scheme, it does affect performance, as shown in figure 7.5, which plots the performance curves for three simulations using three levels of maximum viscosity: $\nu_{low} = 46.45\,\mathrm{m}^2/\mathrm{s}$, $\nu_{medium} = 81.93\,\mathrm{m}^2/\mathrm{s}$ and $\nu_{high} = 67,258.9\,\mathrm{m}^2/\mathrm{s}$, obtained using as threshold temperatures respectively $T_{low} = 1,111\,\mathrm{K}$, $T_{medium} = 1100\,\mathrm{K}$ and $T_{high} = 999\,\mathrm{K}$. The latter is chosen according to the value of the solidification temperature, to assign the maximum viscosity to solid lava particles. Overall, in our experiments, doubling the viscosity gives a decrease in simulation speed of about 20%, while the high viscosity case, which introduces a viscosity which is three orders of magnitude larger, shows a loss of performance of less than 75%. For the latter case it is worth noticing that using an explicit scheme, the time step dictated by the viscous stability condition would be in the order of 10^{-10} s, that, while representable, would be impossible to handle in single precision. The adoption of a higher numerical precision (e.g. double-precision) would lead to unacceptable losses in performance, due to most GPUs having a very low rate of execution of double-precision instructions (as low as 1:32 compared to single-precision ones). Alternative approaches to improve accuracy and precision without such a dramatic loss in performance would be advisable instead [43]. The relationship between computational time and viscosity, as observed in figure 7.5, is thus sublinear, but it should be noted that the ratio is not fixed, since it actually depends on the number of particles with a high viscosity, that also affects the condition number of the matrix and then the convergence speed.

Influence of residual threshold on performance and emplacement The choice of the threshold on the residual, described in 5.3.6, ensures that we reach always the best numerical approximation of the system solution independently of the simulated problem and of the magnitude of the forces acting on the particles.

A larger threshold would allow halting the CG method in fewer iterations, and thus lead to faster runtimes, at the cost of the accuracy of the result. In some applications, this may be considered an acceptable loss to achieve better performances.

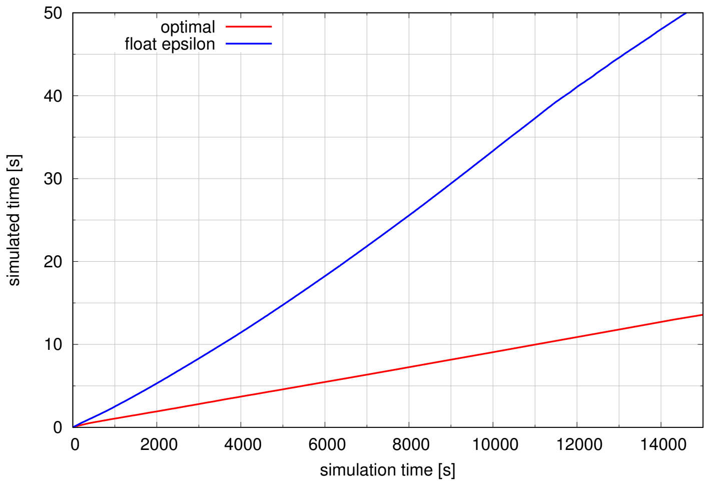


Figure 7.6: Simulation performance for two different residual thresholds.

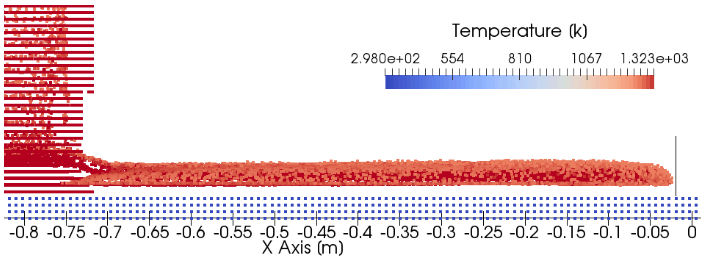
Figure 7.6 shows the simulation performance obtained using a more relaxed stopping condition for the residual vector norm, i.e. $r_\varepsilon = N \cdot \varepsilon_M$, where N is the number of particles. As we can see, we have a gain in simulation speed of about four times, therefore 1s is simulated in around 287 s. Overall, this simulation is about 43 times faster than the explicit formulation.

We can observe that the performance plot in the relaxed case presents an unusual acceleration at around 10s of simulated times. This corresponds to the flow entering a quasi-stationary phase corresponding to the nearly complete emplacement. The small change required between two consecutive steps, i.e. the small modulus of the acting forces, together with the less stringent condition, result into a smaller required number of CG iterations. The reversed trend around 40 s of simulated time is due to the lava overflowing the emplacement pad and thus changing velocity again.

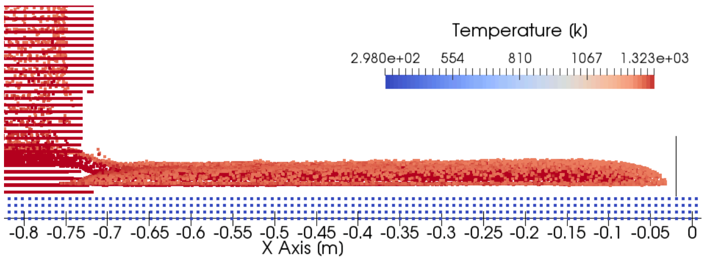
Figure 7.7 shows the difference in emplacement. The lower emplacement length reached with the less stringent condition testifies the missed forces contribution during the integration process. It should be noted that in this example the difference in emplacement is still small (within one Δp) and could be considered acceptable in many applications.

Influence of resolution on performance Varying the spatial resolution affects the simulation performances in multiple ways. Since the time-step is controlled by the ratio of the resolution to the sound speed, the number of time-steps increases linearly with the decrease in resolution. On the other hand, since our model is three-dimensional, the number of particles —and thus the order of the linear systems to solve in the implicit scheme— is proportional to the *cube* of the resolution, which affects the condition number of the coefficient matrix as well as the number of iterations needed to solve the systems.

Figure 7.8 shows the simulation performance for three resolution levels: $\Delta p_1 = 0.008$ m, $\Delta p_2 = 0.004$ m and $\Delta p_3 = 0.002$ m, where the involved number of particles is respectively $100,923$, $495,658$ and $2,781,568$, using a maximum viscosity of



(a) Lava flow extension using the optimal residual threshold.



(b) Lava flow extension using the r_ε threshold.

Figure 7.7: Comparison of simulated lava emplacements with two different residual threshold. The less stringent condition of figure 7.7b gives a shorter flow.

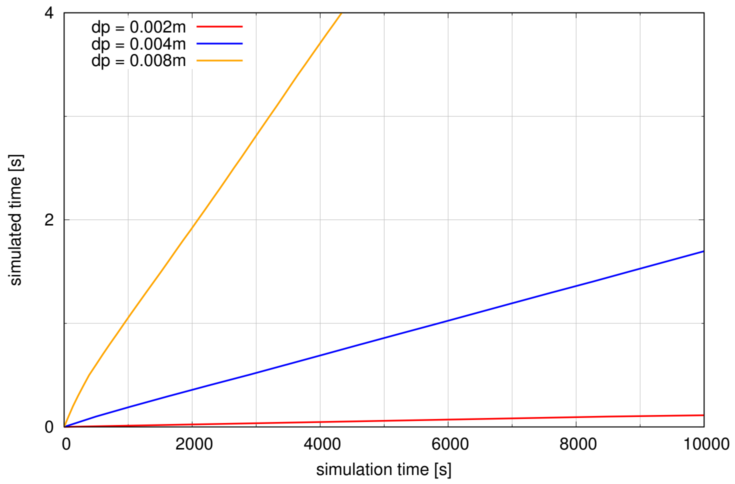


Figure 7.8: Simulation performance for three different resolutions.

46.45 m^2/s.

We remark that in this example the number of particles increases by a factor that is significantly smaller than the expected 8 for each resolution halving. This is explained by the fact that a significant number of particles are used to model the boundary of the domain, and the number of boundary particles increases with the *square* of the resolution. The overall particle scaling in our case is thus significantly lower than the expected third power, and closer to a factor of 5 for the first halving, and a factor of nearly 6 for the second one.

In terms of simulation speed, we observe a dramatic decrease, going from a simulated time to run-time ratio of $1 : 1,124$ for the semi-implicit case at the coarser resolution, to a ratio of $1 : 5,700$ after halving the resolution, and $1 : 116,000$ at the finest resolution.

While the performance decreases by a factor of 100 after reducing the resolution by 1/4th, the semi-implicit scheme still allows the simulation to continue. This is not the case for the explicit one, where the viscosity-dominated time-step drops to 10^{-8} s, making it impossible to run the simulation in single precision.

7.2 Validation of the thermal and mechanical model

The SPH discretization, as any numerical method, introduces some errors that lead to a discrepancy between the simulation results and the exact solutions. Such discrepancy can be reduced acting on the parameters of the SPH discretization, and mainly, adopting finer resolutions. In this section we will validate GPUSPH analyzing [92] the results of the simulation in terms of convergence with respect to only the discretization fineness. Some results in the literature have shown that only acting on the discretization step leads to a saturation of the error [96], but due to the small range of Δp that we are going to span in this work, we can neglect such aspects and consider a fixed ratio

$h/\Delta p$, as introduced in 1.3. We will use $h/\Delta p = 1.33$.

7.2.1 Test cases

The validation process will be conduced by simulating some benchmark test cases, from classical analytical experiments, such as the Plane Poiseuille flow, to more complex semi-analytical and experimental tests, more focused on the field of lava flows.

In some cases we will see the same experiment simulated two times adopting different strategies to model a feature, like for example a constant fluid inlet. This will set the basis to discuss the relevance of determined choices during the simulation set up.

7.2.2 Newtonian plane Poiseuille

A fundamental validation of the mechanical model of GPUSPH is performed against a classical benchmark test case, the Plane Poiseuille flow, constituted by a fluid flowing between to infinitely extended parallel planes.

The fluid adopted has a Newtonian rheology with density $\rho = 1\,\mathrm{Kg/m^3}$ and dynamic viscosity $\mu = 0.1$ Pa s. The two planes are separated by a distance $H = 1$ m and the driving force is $F = 0.05$ N.

After a transient period, the Poiseuille flow exhibits a stationary regime, and the validation can be done according to the velocity profile of this steady state flow.

The domain is developed in three dimensions, where we place the two planes limiting the flow in the xy region, and apply the driving force along the x direction. The plane Poiseuille flow is

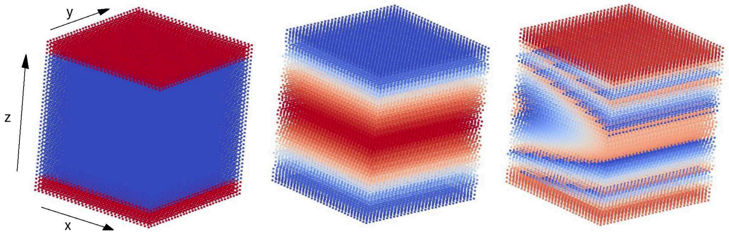


Figure 7.9: Unity cell for the Newtonian Plane Poiseuille flow. Particles colored by material (left), velocity (middle) and ID (right).

by the way a bi-dimensional problem, since the velocity field is constant over the y direction.

The steady state velocity profile over the z direction can be obtained analytically, solving the Navier-Stokes equation (2.16), under the assumption of stationary flow and no-slip boundary conditions, obtaining:

$$u_z = \frac{F}{2\nu}\left[z^2 - \left(\frac{H}{2}\right)^2\right] \tag{7.3}$$

Implementation in GPUSPH

Periodic boundary conditions are used in the x and y direction. We set up a unit cell of cubic shape with 1m long side. The boundaries are built using the Dummy boundary model in no-slip configuration. Figure 7.9 illustrates the simulation domain.

Results

We study the convergence of the method comparing the simulated steady-state velocity profile with the analytical solution, given by (7.3).

The difference between the analytical velocity profile and the simulated velocity is evaluated over the height of the channel, at intervals equal to Δp. For all the particles falling within each interval, bucketing is performed, using the arithmetic mean, in order to have a representative value for the interval. The errors, that here we call e_i, are then obtained for each interval as the difference between the analytical solution and the simulated one. To have a global measure of the error over the whole profile we can use three statistical errors: the L_1 error, defined as:

$$L_1 = \sum_{N_L} |e_i|, \tag{7.4}$$

where N_L indicates the number of bucketing layers, the L_2 error, defined as:

$$L_2 = \sqrt{\sum_{N_L} e_i^2}, \tag{7.5}$$

and the L_∞ error, defined as:

$$L_\infty = \max_{N_L} \{|e_i|\} \tag{7.6}$$

We perform simulations at three different resolution levels, a Low Resolution with inter-particle distance $\Delta p_{LR} = 1/16\text{m} = 0.0625\,\text{m}$, an Intermediate Resolution with inter-particle distance

$\Delta p_{IR} = \Delta p_{LR}/2 = 1/32\,\mathrm{m} = 0.03125\,\mathrm{m}$, and a High Resolution with $\Delta p_{HR} = \Delta p_{LR}/4 = 1/64\,\mathrm{m} = 0.015625\,\mathrm{m}$. Figure 7.10 shows a comparison between the analytical solution and the simulated ones. The convergence is apparent, since as the discretization is refined, the simulated solution moves towards the analytical one. All of the three simulations develop a perfect laminar flow, and no misalignment arises among particles of the same layer, then effect of bucketing is negligible and, in fact, the position of the dots in figure 7.10 corresponds to the position of the particles in the simulation.

The errors coming from the three simulations are illustrated in table 7.1. We can see that doubling Δt the errors are around doubled. We can then asses the convergence of the model with a slightly more than first order trend. The convergence trends are shown in 7.11.

Error type		16 parts/m	32 parts/m	64 parts/m
L_1	error [m/s]	$9.58{\cdot}10^{-3}$	$5.06{\cdot}10^{-3}$	$2.88{\cdot}10^{-3}$
	error ratio	1.89	1.76	
L_2	error [m/s]	$2.4 \cdot 10^{-3}$	$8.97{\cdot}10^{-4}$	$3.62{\cdot}10^{-4}$
	error ratio	2.68	2.48	
L_∞	error [m/s]	$9.91{\cdot}10^{-3}$	$5.42{\cdot}10^{-3}$	$3.24{\cdot}10^{-3}$
	error ratio	1.83	1.67	

Table 7.1: L_1, L_2 and L_∞ errors for the Newtonian plane Poiseuille at steady state. The ratios are computed as the ratio of the error at lower resolution over the error at higher resolution.

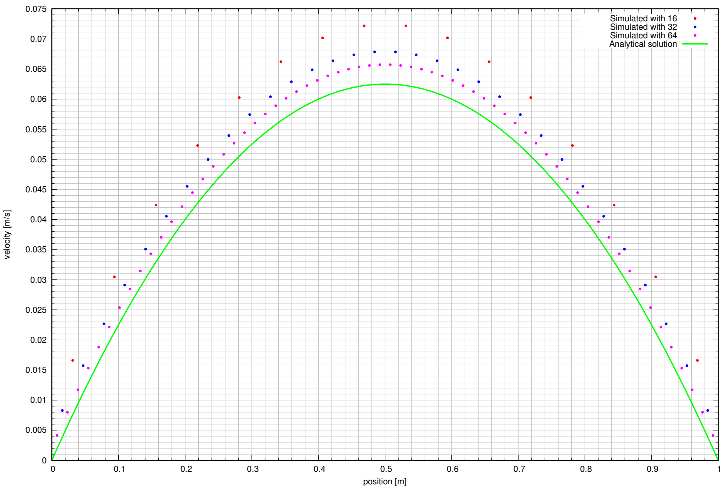


Figure 7.10: Velocity profiles of the Bingham Plane Poiseuille flow. Dotted lines are the simulated solutions, with dots in the bucketing place, the green solid one is the analytical solution.

7.2.3 Non-Newtonian plane Poiseuille

As discussed in subsection 6.1, GPUSPH is able to reproduce several rheological laws based on the Herschel-Bulkley model. We present here a validation of the Plane Poiseuille flow for a Bingham fluid. As for the Newtonian Poiseuille flow, described in 7.2.2, the distance between the top and bottom planes is 1m, and the domain is periodic in the flow and transverse directions

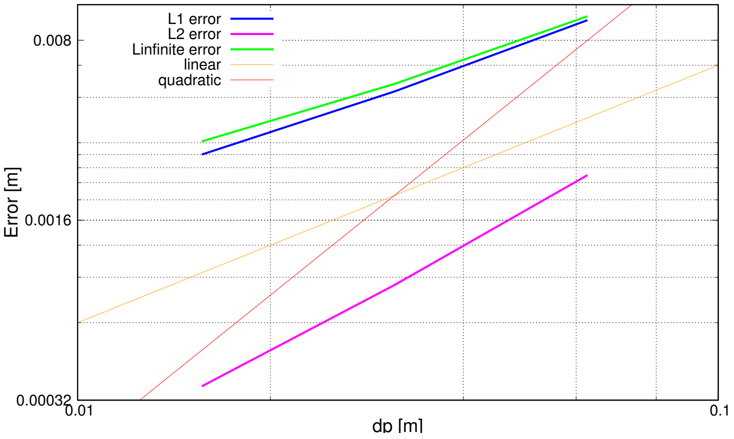


Figure 7.11: L_1, L_2 and L_∞ errors trends for increasing resolution between numerical and analytical velocity profiles of the Bingham Poiseuille flow. The two thin lines mark a reference for first order and second order convergence rate.

(figure 7.12).

The employed fluid has consistency index $k = 0.1$ Pa s, density $\rho = 1\text{kg/m}^3$ and yield strength $\tau_0 = 0.01$ Pa, and is driven by an external force conferring an acceleration $F = 0.05$ m/s^2 resulting in a Reynolds number $Re = 0.225$.

We recall from 2.1.2 that a Bingham fluid has a threshold behavior, such that the strain rate is null as long as the stress satisfies the relationship: $\tau < \tau_0$. In a Poiseuille flow with no-slip

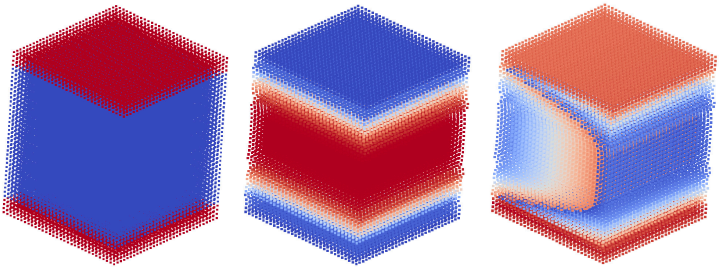


Figure 7.12: Bingham Plane Poiseuille flow. Particles colored by material (left), velocity (middle) and ID (right).

boundary conditions, the stresses are higher in proximity of the walls and decrease toward the interior of the cavity. This can generate a region of zero strain rate at the center of the flow, characterized by a flat region on velocity profile ($du_x/dz = 0$), called plug. The width of the plug is given by the expression

$$h_p = \frac{2\tau_0}{\rho F} \tag{7.7}$$

Outside the plug region the velocity profile is defined by the law:

$$u_z = \frac{F}{2\nu}\left[z^2 - \left(\frac{H}{2}\right)^2\right] - \frac{\tau_0}{\rho\nu}\left(z - \frac{H}{2}\right) \tag{7.8}$$

and the velocity at the plug is obtained by continuity from the adjacent region.

The implementation in GPUSPH follows what said for the Newtonian case, described in 7.2.2.

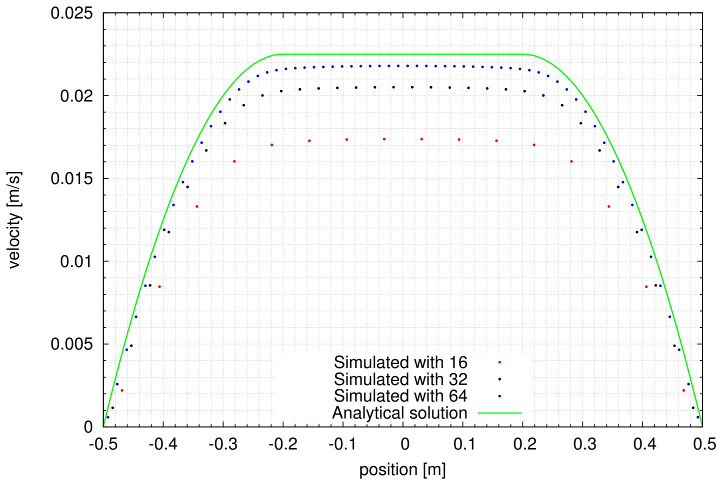


Figure 7.13: Velocity profiles of the Bingham Plane Poiseuille flow. Dotted lines are the simulated solutions, with dots in the bucketing place, the green solid one is the analytical solution.

Results

Error type		16 parts/m	32 parts/m	64 parts/m
L_1	error [m/s]	$4.42 \cdot 10^{-3}$	$1.87 \cdot 10^{-3}$	$7.71 \cdot 10^{-4}$
	error ratio	2.23	2.43	
L_2	error [m/s]	$1.14 \cdot 10^{-3}$	$3.34 \cdot 10^{-4}$	$9.68 \cdot 10^{-5}$
	error ratio	3.41	3.45	
L_∞	error [m/s]	$5.39 \cdot 10^{-3}$	$2.23 \cdot 10^{-3}$	$9.34 \cdot 10^{-4}$
	error ratio	2.42	2.39	

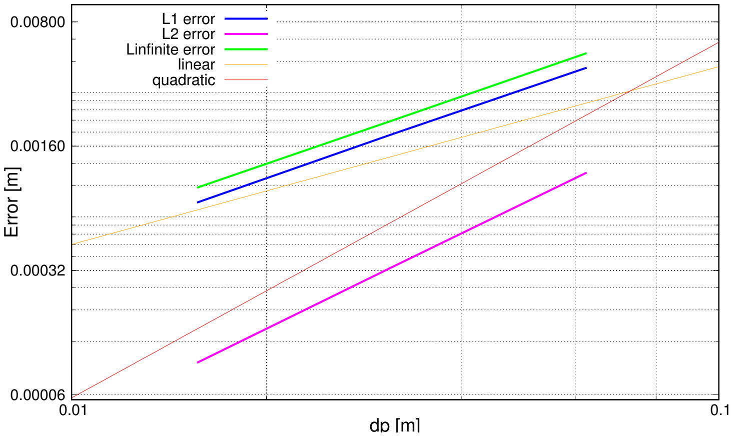


Figure 7.14: L_1, L_2 and L_∞ errors trends for increasing resolution between numerical and analytical velocity profiles of the Bingham Poiseuille flow. The two thin lines mark a reference for first order and second order convergence rate.

Table 7.2: L_1, L_2 and L_∞ errors for the Bingham plane Poiseuille at steady state. The ratios are computed as the ratio of the error at lower resolution over the error at higher resolution.

Simulation performance

Simulating this non-Newtonian experiment requires different times according to the employed resolution. Using 16 particles per meter involves $6,100$ particles and a time step $\Delta t_{16} = 5.5 \cdot$

10^{-5}; 1s of evolution is simulated in 20s. Using 32 particles per meter involves $41,000$ particles and a time step $\Delta t_{32} = 1.4 \cdot 10^{-5}$ s; 1 s of evolution is simulated in 170 s. Finally, using 64 particles per meter involves $300,000$ particles and a time step $\Delta t_{64} = 3.5 \cdot 10^{-6}$ s ; 1 s of evolution is simulated in $3,000$ s. We remark that the time-step in this case is limited by the viscosity, and thus decreases with the square of the linear resolution leading to a significant worsening of the ratio of simulated time to runtime.

7.3 The Bingham Raileigh-Taylor instability

The implementation of the non-Newtonian rheological model described in 6.1 has been tested with the plane Poiseuille experiment 7.2.3, that is a very good case for testing the accuracy of the method, since it has an analytical solution. But on the other side it is a very simple flow that could not show any hidden behavior of the fluid. A more complex experiment is constituted by the Rayleigh-Taylor (RT) instability. It is a typical example of multifluid problem, constituted by two fluids with different density, initially disposed in order to have the heavier fluid above. After an initial perturbation, the two fluids start mixing under the effect of gravity, following a well defined dynamic, up to exchange their positions. GPUSPH has already been validated with respect to the bi-dimensional Newtonian RT instability [7]. Here we are going to simulate a RT instability in a three-dimensional space, using a Bingham rheology.

The tests will follow the work presented by [25], where the RT instability is tested in terms of velocity trend of the interface between the two fluids varying the value of the yield strength and the initial perturbation. The two fluids adopted have density $\rho_1 = 1\,\mathrm{Kg}/m^3$ and $\rho_2 = 2\,\mathrm{Kg}/m^3$, and a kinematic viscosity $\nu = 5 \cdot 10^{-6}\,\mathrm{m}^2/\mathrm{s}$. The domain is periodic in the x and y direction, where the interface of the two fluids is lying at the height $z = 0$, with a width of the unity cell $L = \pi\,\mathrm{m}$. The initial disturbance is given by means of a velocity field, generated as

$$u_x = u_0 e^{-kz}\sin(kx), \quad u_z = u_0 e^{-kz}\cos(kx), \quad z \geq 0 \tag{7.9}$$

$$u_x = -u_0 e^{kz}\sin(kx), \quad u_z = u_0 e^{kz}\cos(kx), \quad z < 0 \tag{7.10}$$

where $k = 2\pi/L$ and we call u_0 amplitude of the disturbance.

There is no analytical model for the development of the Bingham RT instability; the only analytical expressions regard the velocity and curvature over time of the bubble top, for the case of a Newtonian fluid, and is given by the following second order system:

$$\frac{\partial K}{\partial t} = -(3K-1)W, \tag{7.11}$$

$$\frac{\partial W}{\partial t} = -\frac{W^2 + gK}{1-K}, \tag{7.12}$$

with K the curvature and W the velocity. To take into account the effect of the yield strength we will compare the results

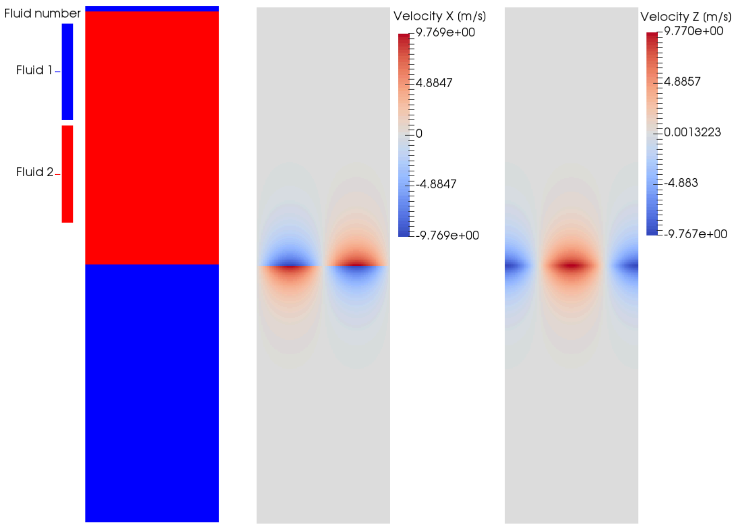


Figure 7.15: On the left the unit cell, where particles are colored by fluid. On the middle and on the right, the two components of the velocity field constituting the initial disturbance; particles are colored by velocity module.

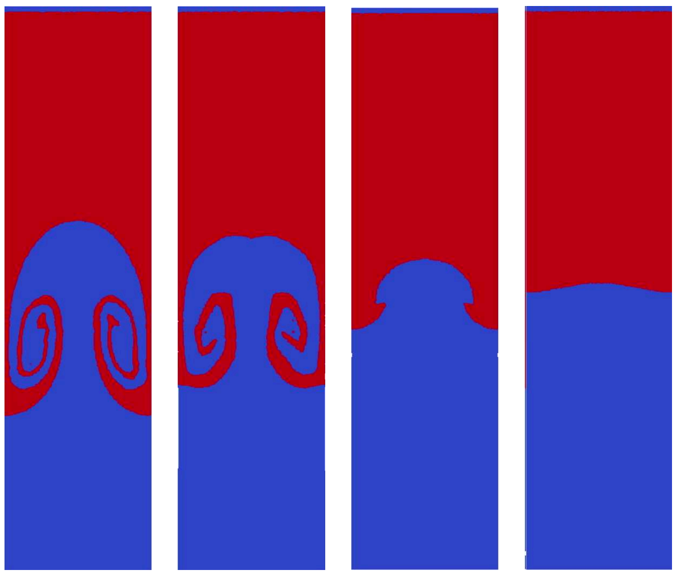
with those of an already validated model (Volume of Fluids), given in [25]. In this case we won't perform a convergence test, but we will run a single well resolved simulation and execute a qualitative comparison with the reference information.

The domain is thus implemented in GPUSPH using a discretization that considers 256 particles over the width of the unity cell; periodic boundary conditions 2.4.2 are applied in the x and y directions, while the top and bottom layer are modeled using the Dynamic boundary model 2.4.1 conditions. The Gaussian kernel is used and the polytropic constant is taken $\gamma = 1$, as indicated by the reference work [25]. The speed of sound is evaluated taking 30 times the maximum velocity between the hydrostatic velocity and the u_0. Figure 7.15 shows the unity cell and the perturbing initial velocity field.

An initial test is performed using $u_0 = 10$ m/s.

From a qualitative point of view, figure 7.16 shows the profile of the instability at $t = 0.5$ s. We can see the case relative to $u_0 = 10$ m/s for $\tau_0 = 0$ Pa, i.e. a Newtonian fluid, in figure 7.16b, and for $\tau_0 = 1$ Pa in figure 7.16d. In both cases we have the formation of a typical RT profile, that in the Bingham case is weaker.

Numerically speaking, figure 7.17 shows the evolution of the bubble velocity for the Newtonian fluid, then the simulated velocities can be directly compared with the analytical curve, obtained from (7.11) and (7.12). To plot the data from GPUSPH we consider the Shepard interpolation of the vertical velocity



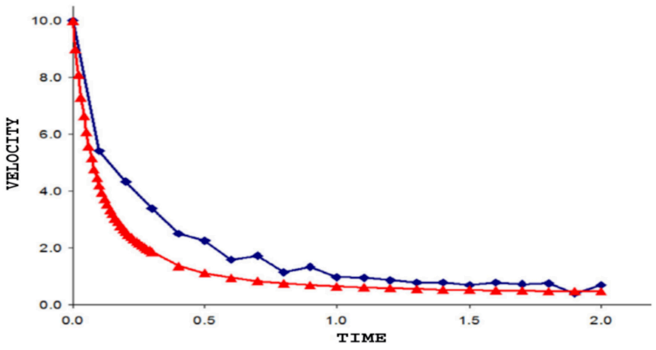
(a) $u_0 = 10\ m/s$, $\tau_0 = 0\ Pa$

(b) $u_0 = 10\ m/s$, $\tau_0 = 1\ Pa$

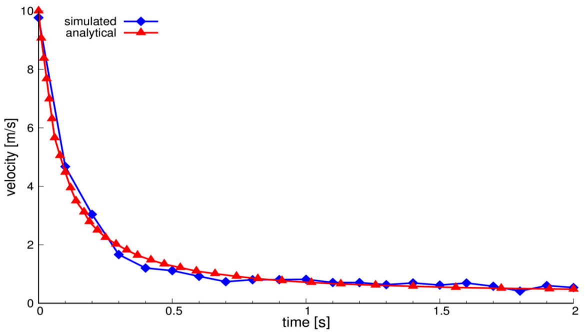
(c) $u_0 = 1\ m/s$, $\tau_0 = 0\ Pa$

(d) $u_0 = 1\ m/s$, $\tau_0 = 1\ Pa$

Figure 7.16: Rayleigh-Taylor instability at $t = 0.5$ s for different values of τ_0 and u_0.

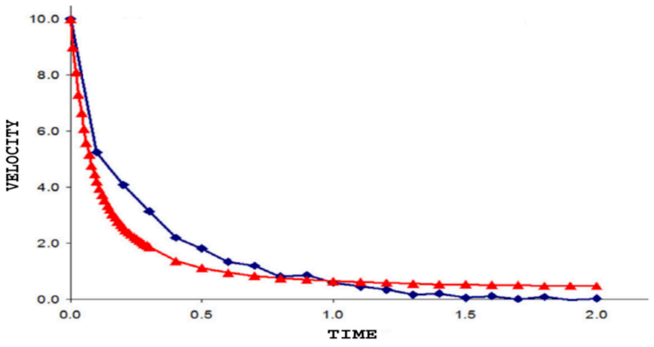


(a) Simulated with VOF. Figure taken from [25]

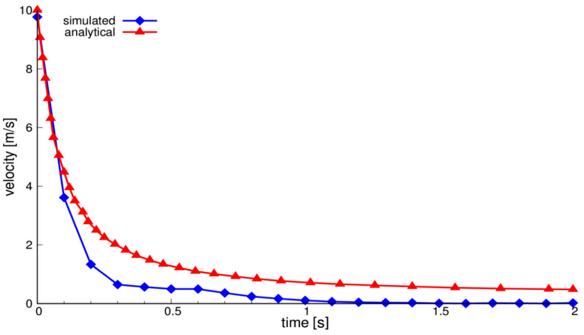


(b) Simulated with GPUSPH.

Figure 7.17: Velocity of the RT instability bubble with $u_0 = 10\,\mathrm{m/s}$ and $\tau_0 = 0$ Pa. Comparison between a simulation obtained with the VOF method (by [25]) and GPUSPH.



(a) Simulated with VOF. Figure taken from [25]



(b) Simulated with GPUSPH.

Figure 7.18: Velocity of the RT instability bubble with $u_0 = 10\,\mathrm{m/s}$ and $\tau_0 = 1$ Pa. Comparison between a simulation obtained with the VOF method (by [25]) and GPUSPH.

field, evaluated at each particle position as:

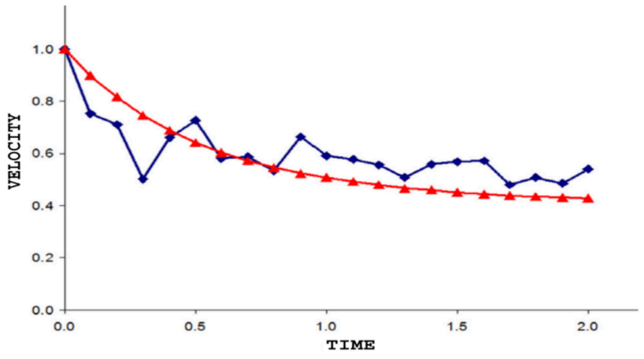
$$\hat{\mathbf{u}}_\beta = \frac{\displaystyle\sum_{\alpha \in F} \mathbf{u}_\alpha W_{\alpha\beta}}{\displaystyle\sum_{\alpha \in F} W_{\alpha\beta}}, \tag{7.13}$$

with F the set of neighbors of the central particle, and then we take the maximum value in the region surrounding the bubble top.

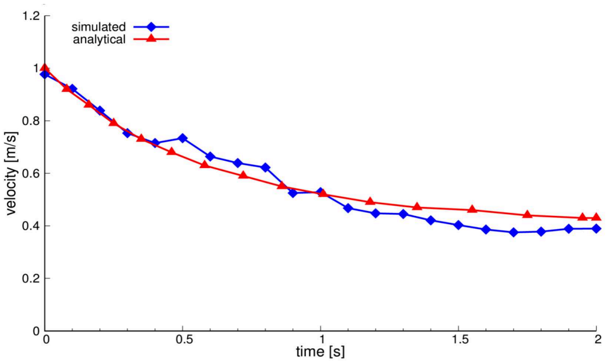
Both the VOF method and GPUSPH are able to approximate the theoretical Newtonian behavior, with GPUSPH better reproducing the trend in both the transient and steady state regions.

In figure 7.18 we have the evolution of the Bingham fluid. As expected, the simulated velocities diverge from the analytical curve because of the presence of the Bingham behavior, that nullifies the shear rate when the stress goes below the value τ_0, and at around $t = 1.5$ s eventually prevents the bubble to move at all. The GPUSPH simulation, shown in figure 7.18b, is compared with that obtained with the VOF method, shown in figure 7.18a; despite the simulated velocities differ in modulus, a good agreement is obtained in the general behavior, confirming the presence of an initial braking phase and of a static phase after about 1.5 s.

Finally, we test a behavior introduced by [25], stating that the instant when the evolution of the bubble in the Bingham flow is stopped depends on the amplitude u_0. The simulations are then repeated for $u_0 = 1$ m/s, and figures 7.16a and 7.16c



(a) Simulated with VOF. Figure taken from [25]



(b) Simulated with GPUSPH.

Figure 7.19: Velocity of the RT instability bubble with $u_0 = 1\,\mathrm{m/s}$ and $\tau_0 = 0$ Pa. Comparison between a simulation obtained with the VOF method (by [25]) and GPUSPH.

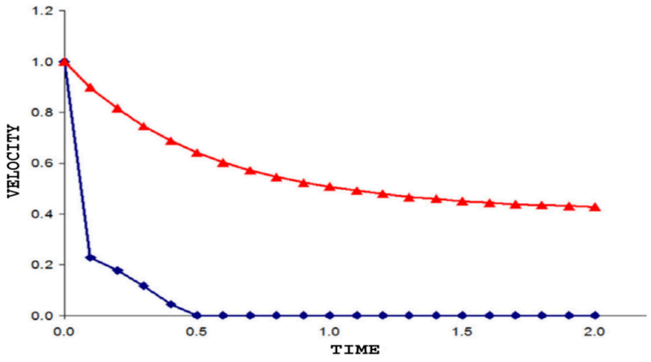
show the instability at $t = 0.5$ s; while in the Newtonian case the instability evolves, even though slowly, for the Bingham rheology the bubble suddenly stops after an initial adjustment. Looking at the evolution of the bubble velocity for the Newtonian case, shown in figure 7.19, we can see that the analytical evolution is well reproduced, with again GPUSPH better minimizing the error (figure 7.19b).

For the Bingham VOF simulation, shown in figure 7.20a, the velocity rapidly decreases, reaching zero at $t = 0.5$ s. This behavior is reproduced by GPUSPH, where the zero velocity is reached a bit earlier, at around $t = 0.4$ s. At steady state the velocity trend simulated by GPUSPH presents some oscillations

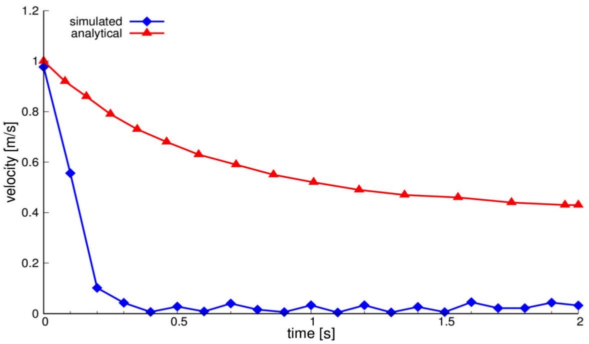
just above the zero, that can be explained as numerical noise coming from the residual movement of SPH particles, that has been observed not to generate any net displacement of the bubble surface. We remark that similar oscillations are also present in the $u_0 = 10$ m/s case (figure 7.18b), although they are more apparent in this case due to the smaller scale.

7.4 Test: Cooling of an emplaced lava flow

We applied the model for the thermal radiation, expressed by the (6.3), to simulate the cooling phase of a real lava flow. The simulation does not involve a mechanical study of the process, but is limited to the thermal aspect. The lava flow is built in GPUSPH from a map of temperature and thickness of the



(a) Simulated with VOF. Figure taken from [25]



(b) Simulated with GPUSPH.

Figure 7.20: Velocity of the RT instability bubble with $u_0 = 1$m/s and $\tau_0 = 1$ Pa. Comparison between a simulation obtained with the VOF method (by [25]) and GPUSPH.

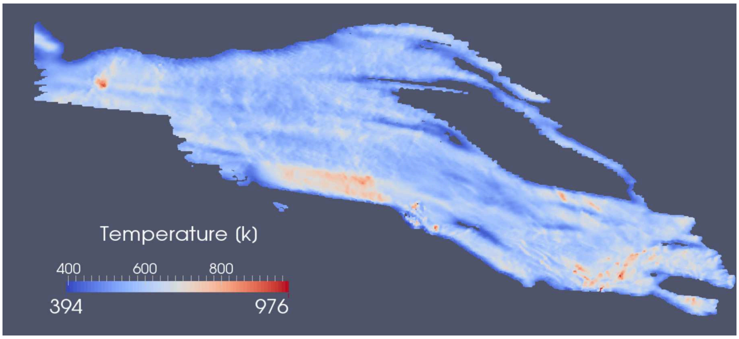


Figure 7.21: Top view of the lava flow implemented in GPUSPH. Particles are colored by temperature.

already spread flow and from the Digital Elevation Map (DEM) of the interested volcanic area. The flow under exam is the episode of the 12th August 2011 [31] at Mount Etna.

The thermal model for conduction has already been validated, as shown in [89]; the cooling phase of a cubic volume of fluid, with initial parabolic temperature profile, has been analyzed, and the comparison over time of the analytical and simulated profile has revealed first order convergence in the errors L_1, $L2$ and L_∞, expressed as in 7.2.2.

For the simulation we used a density $\rho = 2,600\,\mathrm{kg/m^3}$, thermal conductivity $\kappa = 1\,\mathrm{W\,m^{-1}\,K^{-1}}$, emissivity $\epsilon = 0.96$, heat capacity $c_p = 1,600\,\mathrm{J\,kg^{-1}\,K^{-1}}$ and ground temperature $T_g = 296\,\mathrm{K}$. The temperature of the particles forming the interior of the flow is fixed to $T_i = 1,200\,\mathrm{K}$.

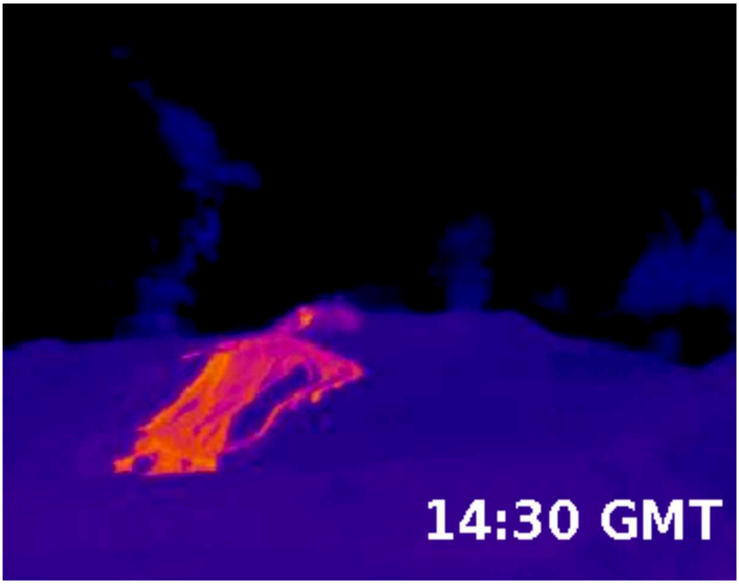


Figure 7.22: Thermal image used to reconstruct the map of the temperature used as input of the simulation and to compare the simulation results.

The thermal evolution of the flow relies on the thermal radiation from the surface and the thermal diffusion through the floor, that implements a sponge layer 2.5.3. Since the emplacement is generated as a problem initialization, all the particle have the same spacing, and we cannot have under resolved areas in the simulation. We can then model the particles surface using the geometrical approach, introduced in 6.3.1.

In this experiment we don't have a well defined reference result, but we will use the information from thermal images (7.22, the same that have been used to build the temperature map. For this reason our analysis will be based on a qualitative comparison of the simulated and measured data. Figures 7.23, 7.24 and 7.25 show the comparison between the measured and simulated data, for three points of the flow, with a respectively high, middle and low initial temperature. The initial instant for the simulation is that associated to the temperature map.

We can see that the simulations follow quite faithfully the measured data, though the simulated temperature tends to be always above the measured ones. This could be caused due to the absence in this simulation of the air convection cooling. Some larger discrepancy are present on the last phase for the low temperature point (7.25), in this case being the simulated temperature lower. This could be due to a wrong assumption made in the model, for example about the condition of the lava underneath the interested surface spot, but it could also be an artifact due to a mis-reference of the simulated point on the thermal image; another possible contribution to the discrepancy could be the uncertainty in the measured data. An overall good matching with the measures is achieved.

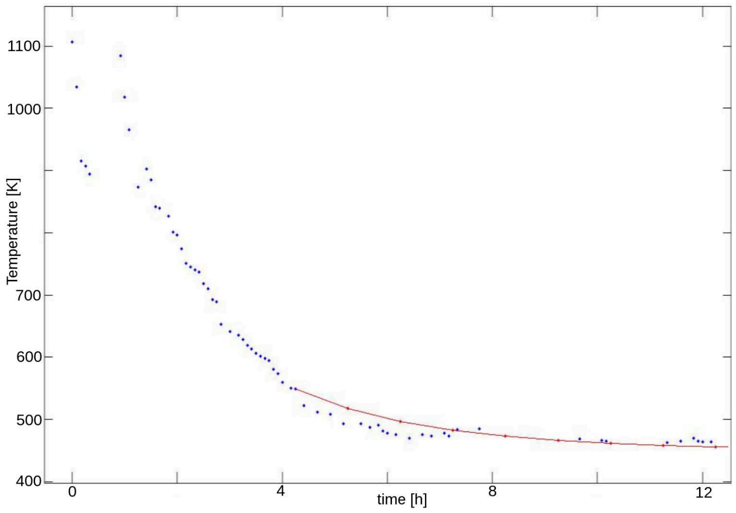


Figure 7.23: Surface temperature of an initially hot point on the lava flow. Points in blue are measurements from a thermal camera, the red line shows the temperature simulated by GPUSPH.

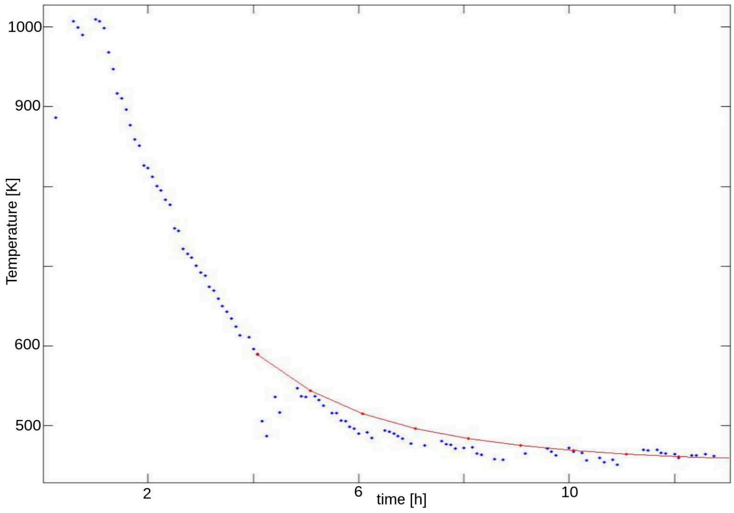


Figure 7.24: Surface temperature of a point with an initial middle temperature on the lava flow. Points in blue are measurements from a thermal camera, the red line shows the temperature simulated by GPUSPH.

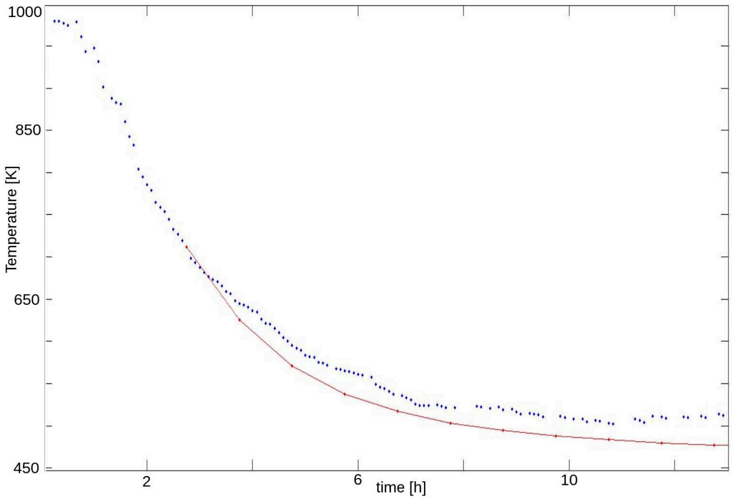


Figure 7.25: Surface temperature of a point with an initial low temperature on the lava flow. Points in blue are measurements from a thermal camera, the red line shows the temperature simulated by GPUSPH.

7.5 Viscous dam-break

Dam breaks are one of the simplest test cases in the field of CFD, and is described as a defined amount of confined fluid that is suddenly freed from one side and allowed to spread onto an horizontal plane, driven by gravity. In the simplest configurations, validation is made against the progress of the front of the flow over time. We will refer to a set up proposed in [18], where the case is referenced as Benchmark test 1. In the following we will adopt this notation, shortening it to $BM1$. the initial configuration of the fluid is a box with length $L = 6,6$ m, height $H = 1$ m and width $W = 6.6$ m. The fluid has density $\rho = 2700$ kg/m^3 and a dynamic viscosity $\mu = 10^4$ Pa $\cdot$ s. According [2] and [77], the evolution of the front over time is analytically described by

$$\frac{x_f(t)}{L} \approx \begin{cases} 0.284 \left(\frac{t}{T}\right)^{\frac{1}{2}} & \text{if } t < 2.5T \\ 1.133 \left(\frac{t}{T} + 1.221\right)^{\frac{1}{5}} - 1 & \text{if } t \geq 2.5T \end{cases} \tag{7.14}$$

where T is the characteristic time of the problem, defined as $T = (L/H)^2(\mu/\rho g h)$. In our case, we have $T = 16.45$ s.

1) Implementation in GPUSPH: The simulation domain consists of a base plane and three walls containing the fluid. The fourth side of the main fluid box is left free to allow the fluid to slump, simulating the sudden opening of a gate. A lateral view of a slice of domain is shown in figure 7.26.

The solid walls are modeled using dummy boundaries, introduced in section 2.4.1. The density diffusion approach introduced

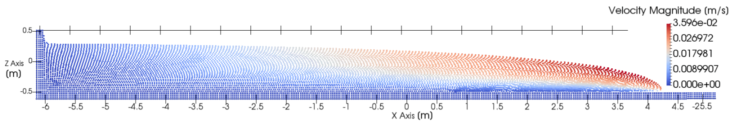


Figure 7.26: Lateral section of $BM1_{HR}$ at $t = 10$s. Particles are colored by velocity magnitude.

by Molteni and Colagrossi [2009] is used to smooth out the noise that naturally develops in the density field. For the speed of sound, the usual choice in WCSPH is to pick a value c_0 around 10 or 20 the maximum velocity value. Our experiments however show that much lower errors at a given spatial resolution can be obtained by using a higher speed of sound. There are diminishing returns in raising the value of c_0, though, due to the smaller time-step, and even a reversal when the time-step becomes too small for the available precision. The main results that we illustrate are thus obtained with a speed of sound times higher than the hydrostatic velocity, resulting in $c_0 = 443$ m/s.

7.5.1 Results for BM1

We show results for three different resolution, a Low Resolution with inter-particle distance $\Delta p_{LR} = 1/8$ m $= 0.125$ m, an Intermediate Resolution with inter-particle distance $\Delta p_{IR} = \Delta p_{LR}/2 = 1/16$ m $= 0.0625$ m, and a High Resolution with $\Delta p_{HR} = \Delta p_{LR}/4 = 1/32$ m $= 0.03125$ m. In the following we will refer to these three simulations as $BM1_{LR}$, $BM1_{IR}$ and $BM1_{HR}$. We measure the front of the fluid as the position of the

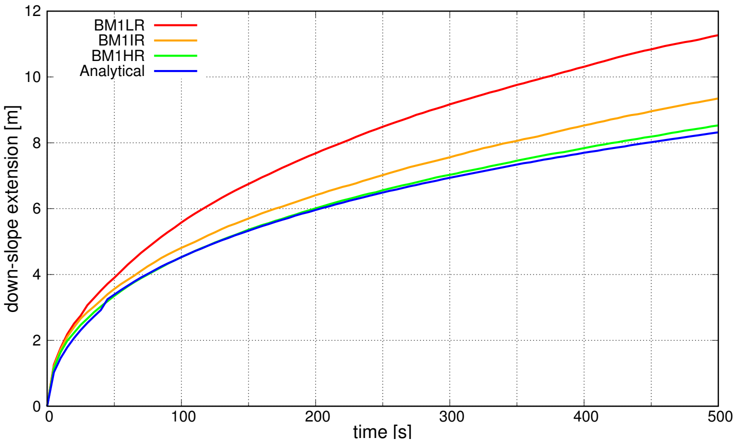


Figure 7.27: Front position over time for BM1. The comparison among the analytical solution and the solutions simulated with different resolutions reveals convergence of the method.

furthest particle in the flow, plus $\Delta p/2$ to take into account the particle volume. Results for the front position at the three resolutions, compared to the front position predicted by equations 7.14 are presented in figure 7.27.

We observe that GPUSPH slightly overestimates the theoretical solution for the front progress, and that the error becomes smaller at higher resolutions. Table 1 shows the errors obtained as the difference between the simulated and theoretical front position in respectively the low, intermediate and high resolution cases at time t, and the error ratios, obtained as the error at lower resolution over the error at higher resolution.

time [s]		$BM1_{LR}$	$BM1_{IR}$	$BM1_{HR}$
10	error [m]	0.3085	0.2628	0.1822
	error ratio	11.1739		2.8779
500	error [m]	2.9546	1.0267	0.2113
	error ratio	2.8779		4.8599

Table 7.3: Errors for $BM1$ at short and long time. The ratios are computed as the ratio of the error at lower resolution over the error at higher resolution.

From table 7.3 and figure 7.28 we can assess the convergence of the model and that the order of convergence grows over time, being in the best case around a second order trend. The latter result is due to the low-resolution simulation becoming under-resolved as the flow progresses, due to the decrease in thickness, leading to larger errors and to the formation of artifacts, as shown in figure 7.29, illustrating the situation for $BM1_{IR}$ at $t = 500$ s: we can observe that the front profile is no more well reconstructed and some artifacts are arising, like the formation of a second head and the detachment of the fluid from the ground. For $BM1_{HR}$ we can further observe that at $t = 45$ s there is a temporary inversion, with the simulation being slightly behind the theoretical result. This may be explained by the change in the expression of the analytical law, that presents a small discontinuity at $t = 2.5T = 41.125$ s, giving a bigger value from the right hand side.

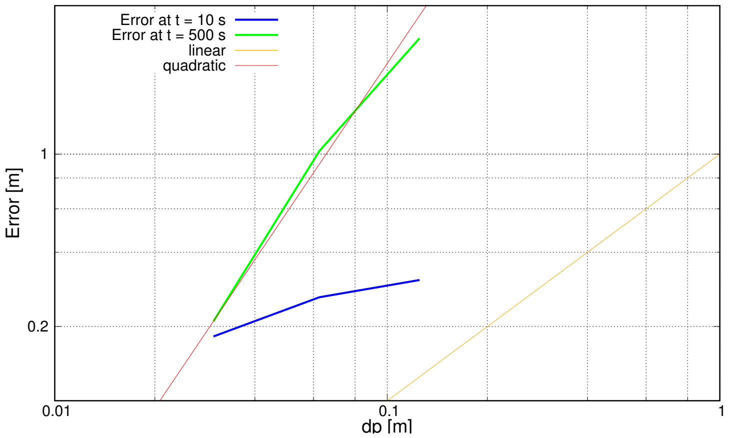


Figure 7.28: Logarithmic plot of the error for BM1 over the spatial discretization interval for different times. The convergence rate is higher at longer times due to the under-resolved condition of lower resolution simulations.

Finally, concerning the fluid height, we have from [77] that for short times the fluid height at the dam position and the end of the reservoir should remain constant (i.e. $h(t, 0)/H = 0.684$ and $h(t, -L)/H = 1$) while the surface shape rearranges, whereas for long times ($t \gg 2.5T$) the height at $x = -L$ and $x = 0$ is the same, and evolves according to the law t^{-2}. For short times (less than around 100 s) all of the three different resolution simulations match the analytical result, with an error of less

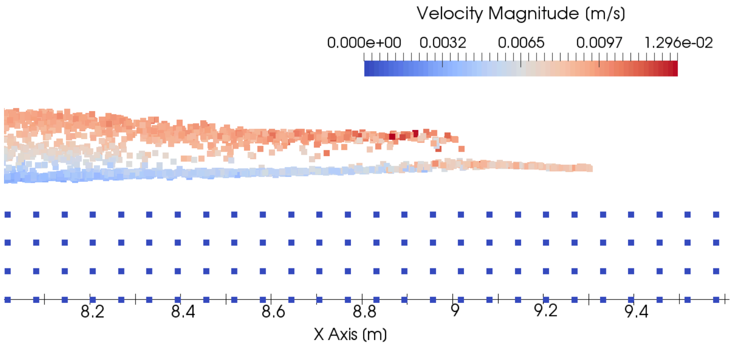


Figure 7.29: Under resolved flow front for $BM1_{IR}$ at $t = 500\,\mathrm{s}$: this is one of the resolution issues arising in $BM1$ as the flow evolves.

than Δp.

7.5.2 Simulation performance

All the simulations were performed on a NVIDIA Titan X GPU (Maxwell architecture); with the adopted spatial discretization, $BM1_{HR}$ consists of $30,890,152$ particles and the time step (controlled by the speed of sound) is $\Delta t_{BM1_{HR}} = 2.5 \cdot 10^{-5}$ s. 1 s of $BM1_{HR}$ evolution is simulated in around $1,950\,\mathrm{s}$ of real time. $BM1_{IR}$ involves $8,080,744$ particles and a time step $\Delta t_{BM1_{IR}} = 5.63 \cdot 10^{-5}$ s, and 1s of $BM1_{IR}$ evolution is simulated in around 168 s. Finally, $BM1_{LR}$ involves $187,244$ particles and a time step $\Delta t_{BM1_{LR}} = 5.6 \cdot 10^{-4}$ s and 1 s of $BM1_{LR}$ evolution is simulated in around 18 s.

7.6 Simulating eruptions using pistons

A fundamental role in the simulation of lava flows is covered by the vent. Modeling a vent implies the management of two fundamental aspects: the input of new particles in the simulation domain and the establishment of a well defined flow rate. The first aspect can be easily treated using some rotating crucibles as done for 6.4, or holed reservoirs, as done for 6.6.1, but these approaches make hard to prescribe a defined flow rate. This issue can be overcome using a piston structure as will be shown in the following.

7.6.1 Inclined viscous isothermal spreading with piston

This benchmark test regards the simulation of a fluid spreading onto an inclined plane, and follows the analytical solution derived by [57]. It is also proposed in [18], so in analogy to $BM1$ we will refer to this experiment as $BM2$. The fluid has a Newtonian rheology and is injected at a constant rate Q from a point source through the plane. We are interested in the evolution of the down-slope and cross-slope extent (L_d and y_p, respectively as shown in figure 7.30) of the flow. The plane is inclined

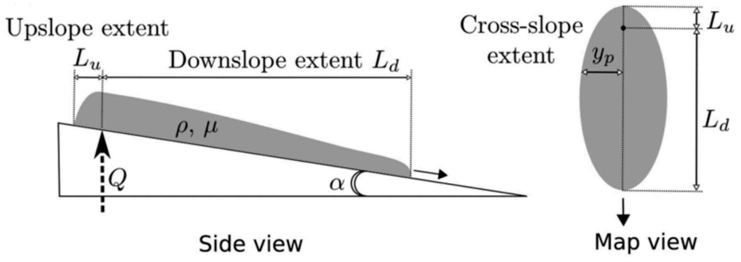


Figure 7.30: Setup for $BM2$. (image from [18].

by an angle $\alpha = 2.5°$ and the fluid, with kinematic viscosity $\nu = \mu/\rho = 11.3 \cdot 10^{-4}\,\mathrm{m}^2/\,\mathrm{s}$, flows at a rate $Q = 1.48 \cdot 10^{-6}\,\mathrm{m}^3/\,\mathrm{s}$.

The density is left free by [18], and we have opted to use the same value adopted in $BM1$, $\rho = 2700$ kg/m^3. According to [18], the analytical solution at long time for the downslope extent over time is given by:

$$L_d \approx \left[\frac{(\rho g)^3 Q^4}{(3\mu)^3} \frac{\sin^5 \alpha}{\cos^2 \alpha} \right]^{1/9} t^{7/9} = 0.0064\, t^{7/9} \qquad (7.15)$$

and for the cross slope extent, at long times, is given by

$$y_p \approx \left(\frac{Q \cos \alpha}{\sin \alpha} \right)^{1/3} t^{1/3} = 0.0324\, t^{1/3}. \qquad (7.16)$$

Implementation in GPUSPH

The simulation domain is constituted by a base plane and a piston, the latter used to obtain the constant flow rate Q, as shown in figure 7.31. Also in this case we perform a convergence test using three levels of discretization: a low resolution with inter-particle distance $\Delta p_{LR} = 1/384$ m $= 2.6 \cdot 10^{-3}$ m, an Intermediate resolution with $\Delta p_{IR} = \Delta p_{LR}/1.5 = 1/512$ m $= 1.95 \cdot 10^{-3}$ m, and a High Resolution with $\Delta p_{HR} = \Delta p_{IR}/1.5 = 1/768$ m $= 1.3 \cdot 10^{-3}$ m.

The fluid source, ideally a point source, is obtained by means of a hole in the plane, through which the fluid is extruded. Because of consistency requirements of the SPH method, the size of the source cannot be chosen arbitrarily small, but there is a lower bound dictated by the resolution. To avoid under-resolving the inlet, we impose a lower bound on the inlet diameter in order to have at least $8\Delta p$ with the coarser resolution, then we use 2 cm. For simplicity, the source has a squared shape. The width of the piston is 0.1 m. It is taken bigger than the hole section to avoid having a high fluid column that would require a very high speed of sound and consequently very small time steps. On the other side, having a too large piston implies more stresses on the fluid that would lead to higher disorder in the flow, with consequent increase in the discretization error [66]. Moreover a large piston surface, coupled with the small value of Q, can determine small particles velocities that can be affected by numerical precision during the integration process. In order to make a simpler definition of the geometry, with reduced numerical rounding, the floor is parallel to the xy coordinate

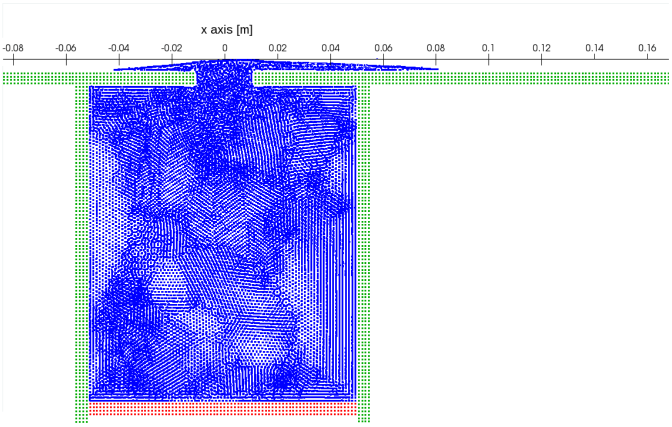


Figure 7.31: Implementation of $BM2$ in GPUSPH, lateral view of a slice. Particles are colored by type: fluid in blue, walls in green and the moving piston in red.

plane, and gravity is oriented by an angle of$-\alpha$. A side view of the simulation at $t = 15$ s is presented in figure 7.31.

As in $BM1$, the speed of sound is chosen so as to minimize compressibility while avoiding numerical instabilities due to loss of precision. For $BM2$, we use $c_0 = 125.5$ m/s.

Results for BM2

The evolution of L_d and y_p are shown in figures 7.32 and 7.33. The comparison between the simulated and analytical solutions

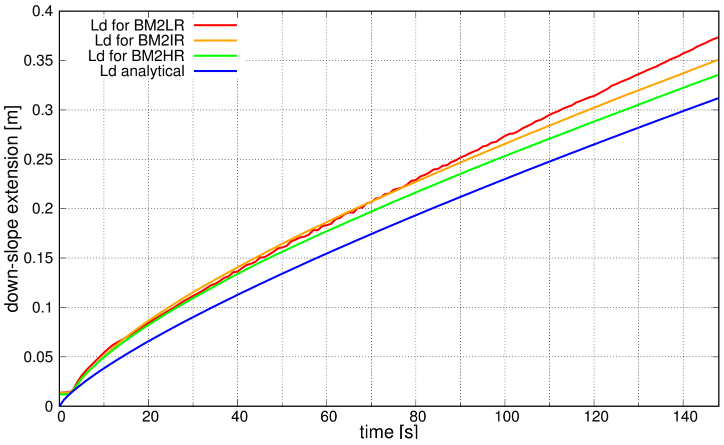


Figure 7.32: Time evolution of the down-slope extension for $BM2$.

is to be performed at long time, anyway an irregularity can be observed at the beginning of the simulation for both L_d and y_p, where, while a theoretical solution would have a growing trend, starting from zero, the simulated solutions maintain a non-zero constant value. This is due to the fact that the vent is not a point source and its size corresponds to the initial extension of the flow. We observe that for L_d (figure 7.32), the convergence is apparent, with a resolution increase leading to results closer to the theoretical solution. As in $BM1$, as the flow spreads out, it becomes under-resolved at lower resolution, leading to consider-

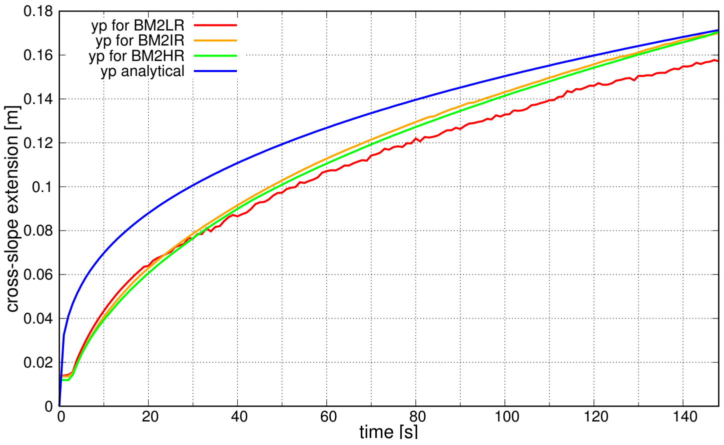


Figure 7.33: Time evolution of the cross-slope extension for $BM2$.

ably worse results that diverge from the analytical solution over time. Moreover, the under resolved flow front generates some artifacts that result in a shaky position advance.

time [s]		$BM2_{LR}$	$BM2_{IR}$	$BM2_{HR}$
100	error [m]	0.0437	0.0357	0.0233
	error ratio	1.2247		1.5290
145	error [m]	0.0606	0.0388	0.0237
	error ratio	1.5644		1.6383

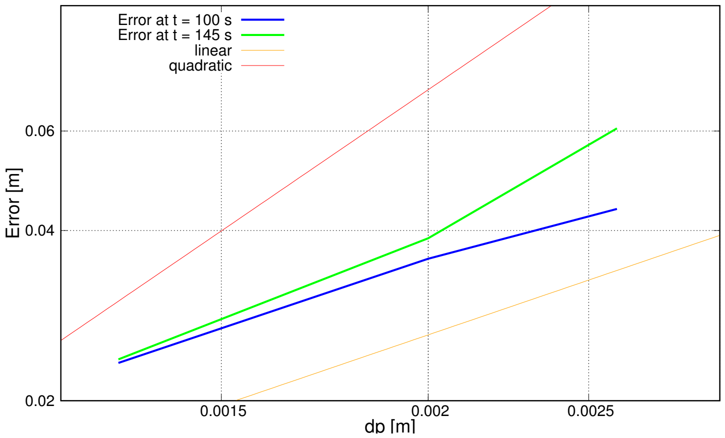


Figure 7.34: Logarithmic plot of the error for $BM2$ over the spatial discretization interval at different times. As for $BM1$, the convergence rate is higher at longer times due to the under-resolved condition of lower resolution simulations.

Table 7.4: Errors for $BM2$. The ratios are computed as the ratio of the error at lower resolution over the error at higher resolution.

The error trend for L_d is shown in 7.34 and reported in table 7.4; as for $BM1$ we have a convergence that gains orders over time. Also in here, this can be explained considering the thinning evolution of the flow, which makes the low-resolution simulation becoming under-resolved as the flow progresses.

When it comes to the cross-slope extent (figure 7.33), we have a discrepancy between the simulated and analytical extension, that leads at long time to a wider flow. The causes are not yet fully known, and could likely be due to the approximations done at numerical level such as finite size of the inlet, as well as compressibility.

Simulation performance

$BM2_{HR}$ involves $2,761,836$ particles and a time step $\Delta t_{BM2_{HR}} = 4.15 \cdot 10^{-6}\,\mathrm{s}$. Running on the same hardware as $BM1$, one second of $BM2_{HR}$ evolution is simulated in around $7,165\,\mathrm{s}$. $BM2_{IR}$ involves $1,171,807$ particles and a time step $\Delta t_{BM2_{IR}} = 6.22 \cdot 10^{-6}\,\mathrm{s}$, with one second of $BM2_{IR}$ evolution being simulated in around $1,673$ s. Finally, $BM2_{LR}$ involves $645,444$ particles and a time step $\Delta t_{BM2_{LR}} = 8.27 \cdot 10^{-6}$ s, with one second of $BM2_{IR}$ evolution being simulated in around 646 s.

7.6.2 Axisymmetric cooling and spreading with piston

This benchmark test deals with a non-isothermal flow. In analogy to $BM1$ and $BM2$, we will refer to this experiment as $BM3$. The setup exhibits many similarities with $BM2$, including a point source with fluid spreading on a plane. The floor is however horizontal in $BM3$, leading to axial symmetry in the flow emplacement. Thermal effects are also taken into account in $BM3$, with only one-way coupling, as the fluid rheology is assumed to be independent from the temperature, which therefore

Parameter	Value	Parameter	Value
Density	886kg/m^3	Convective heat transfer	2W/m^2/K
Viscosity	3.4Pa s	Thermal conductivity	0.15W/m/K
Specific heat	1500J/kg/K	Eruption temperature T_0	42°C
Bed slope	0°C	Ambient temperature T_a	20°C
Emissivity	0.96	Effusion rate	$2.2 \cdot 10^{-8}$m^3/s

Table 7.5: Parameters for $BM3$.

acts as a passive tracer. The parameters for $BM3$ are reported in table 7.5.

The radius of the expanding fluid evolves according to

$$R(t) \approx 0.715 \left(\frac{\rho g Q^3}{3\mu} \right)^{1/8} t^{1/2} = 2.23 \cdot 10^{-3}\, t^{1/2} \qquad (7.17)$$

Implementation in GPUSPH

The implementation in GPUSPH of BM3 is similar to that of $BM2$, though the difference in the magnitude of some physical parameters introduces more stringent constraints. The very small flow-rate sets an upper bound in the dimension of the source, since too large an inlet would result in too small a velocity of the piston, leading to numerical issues for its integration in single precision. The upper bound also imposes limitations on the coarsest resolution that can be used to discretize the problem and finally on the time-step. On the other hand, to be the piston capable of containing enough fluid to run the simulation, a smaller piston section implies a large piston height, requiring a higher speed of sound that would affect the time step. A solution to the fluid column could be to make a horizontal piston, parallel to the plane, but the junction with the hole would introduce disorder in the flow. The size of the piston has been chosen in order to approximate a trade- off with the issues just mentioned. The piston has a squared section, 0.005 m wide and is 0.13 m high. The plane has a side 0.12 m long and the vent is a square hole placed in the middle. In order to avoid introducing disorder in the flux, the hole shape and size matches with the piston section.

We use for the wall the same thermal parameters that we use for the fluid, and the thickness required by the dynamic boundary model is also sufficient to implement the absorbing conditions for the thermal model. The speed of sound is 40 m/s.

We perform a convergence test using three levels of discretization: a low resolution with inter-particle distance $\Delta p_{LR} = 1/512$ m $= 1.95 \cdot 10^{-3}$ m, an Intermediate resolution with $\Delta p_{IR} = \Delta p_{LR}/1.5 = 1/768$ m $= 1.3 \cdot 10^{-3}$ m, and a High Resolution with $\Delta p_{HR} = \Delta p_{IR}/1.5 = 1/1024$ m $= 9.7 \cdot 10^{-4}$ m.

Results for BM3

For the flow dynamics, the model convergence can be assessed by looking at the radius of the emplacement. We take as objective the simulated time $t = 144$ s the fluid is already spread enough to show a clear temperature profile, and artifacts due to the low resolution are yet to appear. We observe an over estimation of the emplacement radius, being $RBM3_{HR}(144) = 0.038$ m and $R(144) = 0.027$ m, probably due to discrepancies between the numerical and analytical setup.

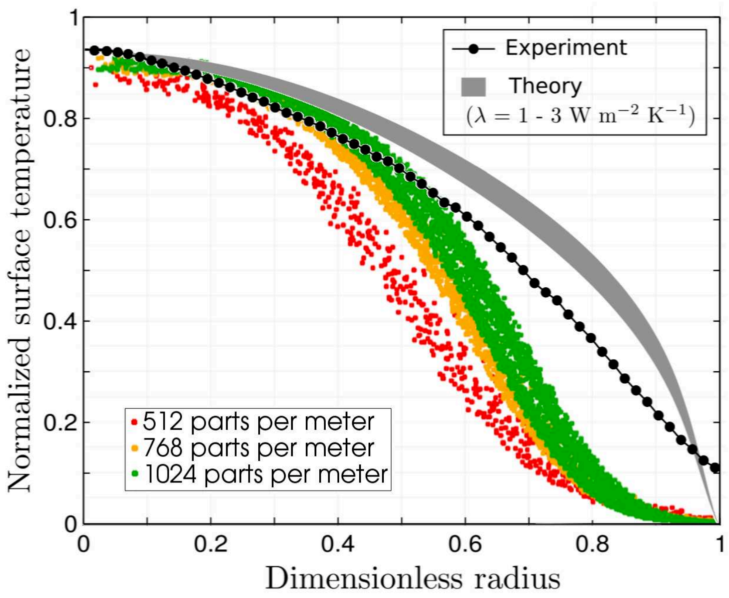


Figure 7.35: Temperature profile of $BM3$. The normalized temperature is plotted over the distance from the vent, normalized by the current flow radius. The two reference curves, i.e. analytical and experimental, are taken from [33].

Concerning the temperature profile, we perform a graphical comparison between the simulated profile, and the analytical and measured ones, as shown in figure 7.35, where we plot the normalized temperature ($T^\star = (T - T_a)/(T_0 - T_a)$) with respect to the radius normalized by the current flow extension. In the two reference curves, the temperature at the vent is lower than the temperature specified in the problem data, which has been faced by setting a lower initial temperature of the simulated fluid. From a graphical estimation, the temperature is chosen as the 93% of the normalized temperature, that is 313.6 K. We can see that although the simulated solutions do not apparently match the reference, they qualitatively tend to those as the resolution is increased. The high mismatch in the shape of the profile can be due to an incomplete implementation of the thermal model for the boundary, that is not described in [18].

Simulation performance

$BM3_{LR}$ involves $30,550$ particles and a time step $\Delta t_{BM3_{LR}} = 1.97 \cdot 10^{-5}$ s. One second of $BM3_{LR}$ evolution is simulated in around 15 s on the same hardware used for the other benchmarks. $BM3_{IR}$ involves $60'432$ particles and a time step $\Delta t_{BM3_{IR}} = 1.31 \cdot 10^{-5}$ s. One second of $BM3_{IR}$ evolution is simulated in around 134 s. Finally, $BM3_{LR}$ involves $106,724$ particles and a time step $\Delta t_{BM3_{LR}} = 9.83 \cdot 10^{-6}$ s. One second of $BM3_{LR}$ evolution is simulated in around 267 s.

7.7 Simulating eruptions using open boundaries

The two simulations $BM2$ and $BM3$ presented in 7.6, involving the generation of flows at constant rate, can be repeated with a more sophisticated way of generating the flow, that is using open boundaries 2.4.3 to create an inlet. We are going to see that this approach gives cleaner result an shorter simulation times, that will be paid in terms of implementation complexity.

7.7.1 BM2: Inclined viscous isothermal spreading with inlet

This benchmark test regards the simulation of a fluid spreading onto an inclined plane, as already described in subsection 7.6.1, where instead of relying on pistons, the generation of the flow is achieved by means of an inlet, realized using open boundaries 2.4.3, as depicted in 2.2. To facilitate comparisons with the $BM2$ implementation that uses a piston, in the following we will refer to such simulation as $BM2$-piston.

Also in this case we use three discretization levels; here, as we are going to see, with respect to $BM2$-piston we have less stringent conditions in the time speed of sound and a smaller number of particles involved, then we can afford to use finer resolutions, i.e. $\Delta p_{LR} = 1/512$, $\Delta p_{IR} = 1/768$ and $\Delta p_{HR} = 1/1024$.

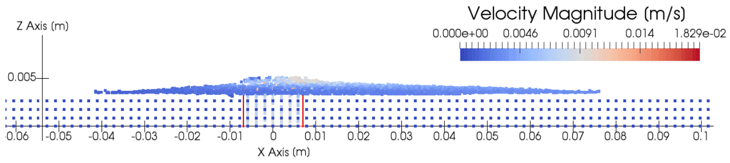


Figure 7.36: Lateral view of $BM2_{HR}$ at $t = 15$ s. The two red lines mark the inlet region.

Implementation in GPUSPH

The simulation domain consists of a base plane, and the fluid source, ideally a point source, is provided by a hole in the plane, where the inlet is placed. The floor is parallel to the xy coordinate plane, and gravity is oriented by an angle of $-\alpha$.

To avoid under-resolving the inlet, we impose a lower bound of $6\Delta p$ for the inlet diameter. Unlike in $BM2$-piston, thanks to the possibility of having inlets with random shape (see section 2.4.3) we use a vent with a round section. To ensure equal spacing among particles, the round vent is obtained by means of particles disposed on a regular grid, matching with the gird used for the plane, that are all falling inside a circular region. A side view of the simulation at $t = 15$ s is shown in Figure 7.36.

The absence of the piston chamber makes us free of annoying fluid columns, thus allowing the choice of smaller values for the speed of sound. In contrast to $BM2$-piston, where the high speed of sound implied long simulation times but also was a main contribution the stability of the simulation, here we are going to show how it is possible to push in the opposite direction, choosing small values of c_0. In any case, we stay in accordance to what said for the previous experiments, that, within certain limits, a higher speed of sound gives better results [18]. Then we evaluate the hydrostatic velocity using a height larger than the actual necessary value, so we use the domain height. The latter is obtained as the vertical projection of the inclined plane (i.e. around 5 cm, whereas the thickness of the flow is less than 1 cm.). We thus have $c_0 = 19.4$ m/s, resulting in time steps of $\Delta t_{BM2_{LR}} = 4.1 \cdot 10^{-5}$ s, $\Delta t_{BM2_{IR}} = 2.67 \cdot 10^{-5}$ s and $\Delta t_{BM2_{HR}} = 2 \cdot 10^{-5}$ s.

Results for BM2

To remark the absence of the problems that in $BM2$-piston we encountered in the design of the piston, mainly coming from the small flow rate, here we are going to study the results obtained for even smaller flow rates. We will use $Q = 1.16 \cdot 10^{-6}\mathrm{m}^3/$ s instead of $Q = 1.48 \cdot 10^{-6}$ m^3/s.

The evolution of L_d and y_p for the simulated flow is shown in Figures 7.37 and 7.38. We observe that for L_d, the convergence is apparent, with a resolution increase leading to results closer to the theoretical results. As in the BM1 case, we observe that as the flow spreads out, it becomes under-resolved at lower resolution, leading to worse results, diverging from the analytical solution over time. One apparent improvement is in the cleanness of the plots, brought by the adoption of finer resolutions and in general from the higher regularity of the flow at the vent.

The error plot is shown in figure 7.39. As for the all the

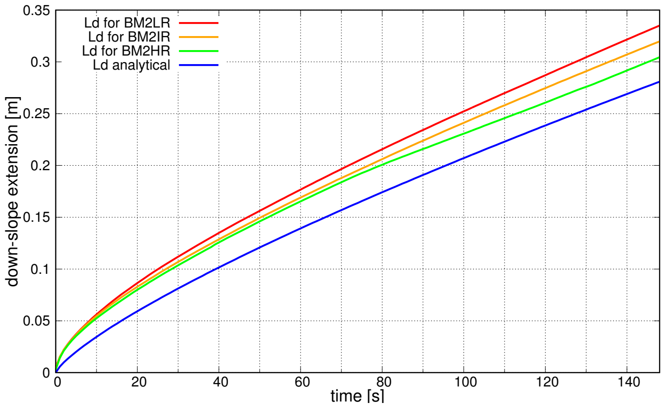


Figure 7.37: Evolution of the flow front L_d over time.

time [s]		$BM2_{LR}$	$BM2_{IR}$	$BM2_{HR}$
100	error [m]	0.0452	0.0342	0.0237
	error ratio	1.3216	1.4430	
145	error [m]	0.0536	0.0387	0.0232
	error ratio	1.3850	1.6681	

Table 7.6: Errors for $BM2$-inlet. The ratios are computed as the ratio of the error at lower resolution over the error at higher resolution.

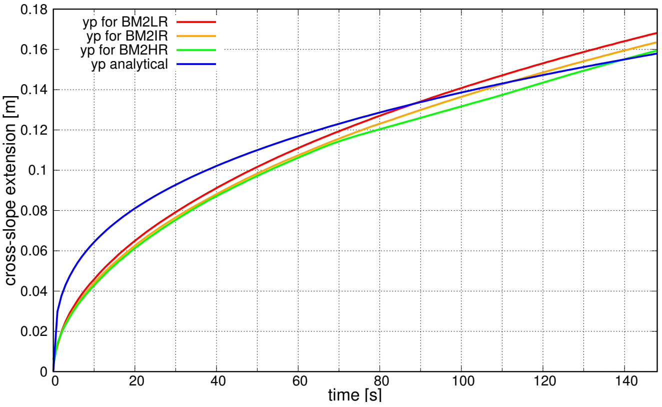


Figure 7.38: Evolution of the down-slope extent (L_d) and the cross-slope extent (y_p) for BM2.

spreading flows we have seen so far, the errors for the lower resolution simulations increase over time, leading to higher convergence rates. A major improvement can be seen here for the first time, as for the high resolution simulation the error is smaller for higher times, showing a convergence towards the analytical solution that, in fact, is defined for long times. The errors are shown in table 7.6.

When it comes to the cross-slope extent, Figure 7.38, despite the solutions are cleaner with respect to $BM2$-piston, we have we have the same kind of discrepancy between the simulated and analytical extension, that leads at long time to a wider flow. The

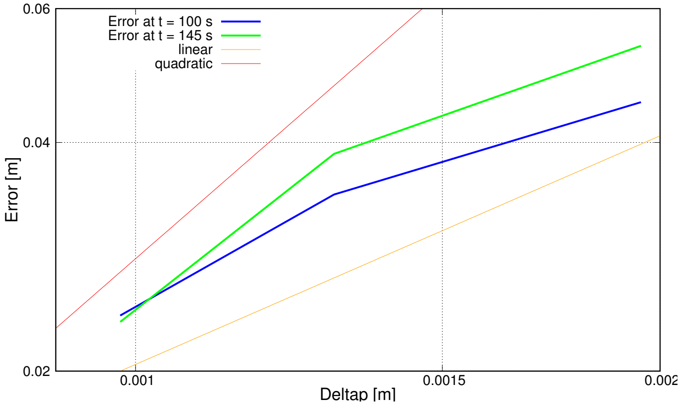


Figure 7.39: Logarithmic plot of the error for $BM2$ over the spatial discretization interval at different times. Once again, the convergence rate is higher at longer times due to the under-resolved condition of lower resolution simulations.

possible reasons are the same discussed for the case involving a piston.

Simulation performance

$BM2_{LR}$ involves initially $696,300$ particles and a time step $\Delta t_{BM2_{LR}} = 4.1 \cdot 10^{-5}$ s. One second of $BM2_{LR}$ evolution is simulated in around 156 s on the same hardware used for the other benchmarks. $BM2_{IR}$ involves initially $1,536,444$ particles and a time step $\Delta t_{BM2_{IR}} = 2.67 \cdot 10^{-5}$ s. One second of $BM2_{IR}$

evolution is simulated in around 738 s. Finally, $BM2_{LR}$ involves $2,799,236$ particles and a time step $\Delta t_{BM2_{LR}} = 2.1 \cdot 10^{-5}$ s. One second of $BM2_{LR}$ evolution is simulated in around $1,740$ s.

It should be noted that the real to simulated time ratio does not change significantly over time, despite the increasing number of particles due to the inlet generation, since the maximum amount reached is barely sufficient to saturate the compute capabilities of the GPU.

Comparison piston-inlet

We have seen two possible ways to simulate a prescribed flow rate from a vent. Both approaches give a convergence in the results.

The errors obtained here cannot be directly compared to that obtained for $BM2$-piston, since we are adopting different values of effusion rate, then we are not modeling the same problem. By the way, from a more qualitative comparison we can see that for the same resolution the errors that we get in these new simulations are larger. This is likely due to the adoption of a smaller speed of sound, that in $BM2$-piston we were forced to take large in order to compensate the fluid columns, paying in terms of simulation time. Higher valuer for the speed of sound can be anyway used in the open boundary case.

The implementation involving open boundaries have shown several advantages, like the reduction of the number of particles involved in the simulation, the reduction of the simulation time, the enlargement of the time step and improvement in terms of stability and the possibility to use a finer resolution avoiding the annoying trembling occurring at low resolutions. Moreover $BM2$-inlet has shown an improvement of the result at longer simulation times, suggesting a convergence towards the reference data, that is expressed for long times. On the other side, a big advantage of the piston is the simplicity in the implementation.

This results are also important to demonstrate that, even using the same numerical method, a proper modellization of the original problem has a strong impact on the simulation results.

7.8 Real Lava flow

This benchmark regards a real lava flow obtained experimentally with natural basalt heated above the solidus and completely degassed (the details of the experiment can be found in [26] and online at `http://lavaproject.syr.edu/`)

The flow interacts with a triangular steel obstacle 45 cm distant from the flow source. The walls of the obstacle are 14 cm long and the opening angle between them is 90°. The parameters for $BM4$ are reported in Table 7.7.

The fluid is modelled as Newtonian, with a temperature dependent viscosity, described by

$$\log_{10}\mu = -5.94 + \frac{5500}{T-610} \tag{7.18}$$

where the temperature T is expressed in Kelvin.

The reference data for this problem are given in terms of extension and front velocity as a function of time. The average front velocity before touching the obstacle is 4.06cm/s and around 2 cm/s after encountering the obstacle.

Table 7.7: Parameters for $BM4$.

Parameter	**Value**
Density	2350 kg m^{-3}
Specific heat	1500 $Jkg^{-1}K^{-1}$
Bed slope	$14°$
Effusion rate	$0.77 \cdot 10^{-3}$ m^3s^{-1}
Convective heat transfer	2 $Wm^{-2}K^{-1}$
Emissivity	0.95
Eruption temperature	1100 °C
Ambient temperature	25 °C
Thermal Conductivity of the bed	0.2 $Wm^{-1}K^{-1}$

Implementation in GPUSPH

In contrast to $BM2$ and $BM3$, the inlet in $BM4$ is not a point source, and a velocity profile would have to be assigned over its section. To avoid this issue, we place the inlet above a short inflow channel, in a subcritical configuration, thus effectively modeling only the tail end of the channel. For simplicity, in this first stage we adopt for the channel a rectangular section instead of a circular one. A lateral view of the simulation domain is shown in figure 7.40.

The speed of sound is derived as usual from the highest

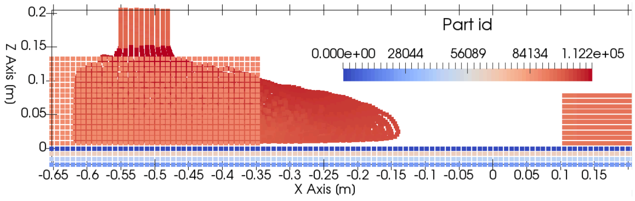


Figure 7.40: Side view of the high resolution BM4 simulation. Particles are colored by index.

velocity between the vent velocity and the hydrostatic velocity. In this configuration the hydrostatic velocity is higher, and determines a speed of sound $c_0 = 43$ m/s.

Due to the high viscosity and resolution, we use the semi-implicit formulation we presented in [93], which requires the use of the dynamic boundary model. From the thermal point of view, we impose absorbing boundaries, by means of the sponge layer technique (see 2.5.3). Applying the (2.41), we should have $H_s > 0.0024$ m, that considering a $\Delta p = 1/256 = 0.0039$ m will be already satisfied by the four layers employed for the dummy boundary model.

Additionally, following the approach seen in [93], we avoid the very high viscosity values that arise in the final cooling phases by clipping the viscosity curve at $T = 1190$ K, that corresponds to a kinematic viscosity of $\nu = 44.45$ m^2/s.

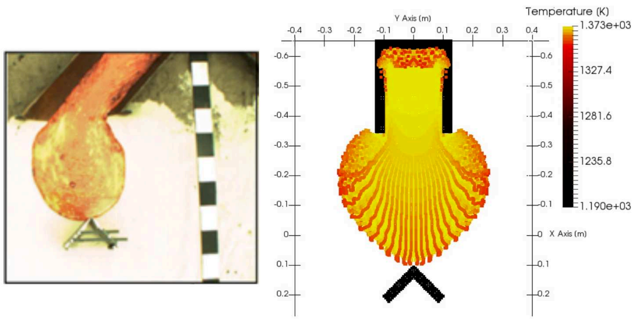


Figure 7.41: Comparison between real and simulated emplacement obtained using (7.18).

Results for BM4

For this experiment we also performed several simulations at three different resolution (64, 128 and 256 particles per meter), but we did not achieve a significant convergence toward the reference data. In fact, the simulated flows tend to be slower and thicker than expected.

Figures 7.41 and 7.42 show the experimental flow compared to the simulated flow at the same stages of the emplacement, that is, when touching the obstacle and just after overcoming it. It is clear that, since the simulated velocity differs from the real one, the comparison makes no sense in terms of temporal instant. From this visual comparison of the simulated and real emplacements we can see that the shape of the emplacement

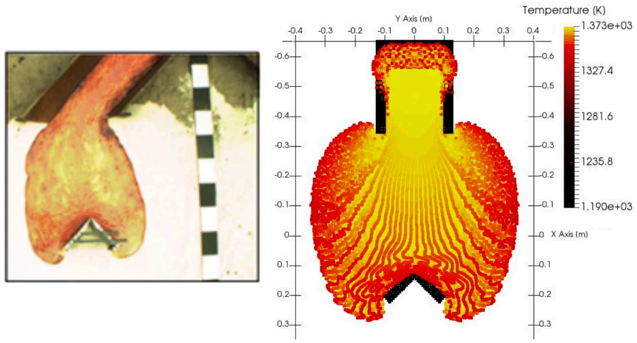


Figure 7.42: Comparison between real and simulated emplacement obtained using (6.6).

is different and that the simulated flow is wider. A number of factors potentially contribute to these two discrepancies: the boundary model in use is known to reduce adherence in the wet/ dry zone, and the simplified geometry of the channel itself, that doesn't match exactly the physical experiment (square instead of round cross-section).

Concerning the main problem, related to the flow speed, the higher thickness suggests a larger viscosity. We then repeated the simulation used the viscosity law expressed by (6.6) obtaining a much faster flow (around 6 m/s). We deduce that a possible uncertainty or error in the expression of the experimentally obtained viscosity model may be a cause of the discrepancy.

In any case, previous experiments have shown that some additional corrections to the model need to be done, both in the thermal description of the problems and in the boundary models. When any form of inaccuracy from these sources will be eliminated, the simulation will be able to reveal any form of discrepancy in the viscosity model and will be a potential tool to improve it.

7.9 Conclusions

In this thesis we have shown the development of a model based on SPH for the simulation of complex fluids, with the application to a particularly complex case, constituted by lava flows. This has been shown referring to GPUSPH, a versatile fully three-dimensional simulation engine that exploit the advantages of the SPH method and the power of modern GPUs to run very accurate and performing simulations to serve the most disparate applicative fields, ranging from engineering to geophysics.

We have seen the implementation in GPUSPH of the models required to simulate a lava flow, and tested those, showing how the final result was able to deal with features like Newtonian and non-Newtonian fluids, with constant as well as temperature-dependent rheology, with multi-phase systems and multi-fluid.

We have discussed how the explicit integration scheme can become a limiting factor in the simulation of highly viscous fluids, for which the time-step decreases with the square of the spatial resolution quickly leads to very long simulation times and how this is a problem for the application of GPUSPH to lava flows, particularly in the cooling phase, where the viscosity of lava can grow by more than two orders of magnitude. So we have implemented a semi-implicit integration scheme for WCSPH, where the viscous contribution to the momentum equation is integrated implicitly, solving the corresponding linear system with the CG method. This allows the simulation of highly viscous fluids using larger time-steps, thanks to the elimination of the viscous term from the stability condition, bringing in the best analyzed case to a reduction of the simulation time of about 43 times. We have also shown the effect that the viscosity, the resolution and the CG residual threshold have on the simulation time and on the quality of the results, and that for relatively high viscosities or relatively fine resolutions, the semi-implicit approach is the only one able to run a simulation when working in single precision.

We have seen that the increase in time-step gives advantages in term of numerical robustness and quality of results, although it affects the stability in the density evolution when using dynamic boundaries. A solution to this is to compute the density derivative from the value of the velocity used for the integration of position.

From the application to lava flows we highlighted both the runtime benefits, and the improvements in the quality of the final emplacement, due to fewer numerical errors accruing during the simulation, thanks to the smaller number of iterations necessary to complete the same simulated time.

We have seen that the largest downside to the semi-implicit scheme is the much higher computation cost of each integration step, so that performance-wise it only becomes advantageous for very large viscosities or very fine resolutions. A possible approach in this sense would be to dynamically switch between the explicit and the semi-implicit integration schemes depending on resolution and viscosity. This would be of particular interest in simulations where the particle viscosity is not constant over time, which is typically the case for lava flows.

We have validated GPUSPH for the simulation of lava flows against some benchmark tests or specific applications, presenting also the simulation of the first three benchmark tests introduced by [18]. In the latter, it was stated that the main drawback of SPH is the Weakly Compressible formulation and therefore a tuning of the parameters is needed to improve the quality of the simulations. Following this reasoning we mainly acted on the speed of sound and the boundary model in order to find a good compromise between accuracy, stability and simulation time. We have shown that for many cases we have a strong convergence; while in some tests larger discrepancies with the analytical results are present, we have proven the existence of a convergent behavior and that the errors can be mitigated choosing a proper resolution for the spatial discretization. We have also encountered some issues related to under-resolved conditions of the problem, and again we have seen that they can be eliminated using finer discretization. For other cases of discrepancies we have encountered we believe that the biggest obstacle to reproduce the real behavior has been the incomplete description of the problem setup from the reference with respect to the thermal boundary conditions.

An essential aspect for the simulation of lava flows is the modelling of the vent, to which end we have introduced open boundaries, in order to create constant prescribed flows in a robust way, showing the improvements introduced in the results.

We shown that large limitations in the present implementation are related to the choice of single precision, proposing some ways to mitigate or solve some of this problem.

Future developments will focus on the issues related to the numerical precision in order to allow running simulations with finer resolutions and thus obtain more accurate results. Further improvements are also being worked on for the semi-implicit scheme, to include support for the dummy boundary model and to better integrate with the open boundaries, the current design of which relies on the fluid velocity at the previous time-step. An extension of the semi-implicit scheme to non-Newtonian rheology is also being studied: it introduces a high degree of complexity in the resolution of the system, that becomes non-linear, but constitutes a great plus bringing the advantages of the semi-implicit scheme to the study of the rheological properties of lava.

For the performance, another significant speed-up to the semi-implicit formulation could be achieved by pre-computing the coefficient matrix, reducing the computational load of each CG iteration, at the cost of a larger memory consumption. This would however prevent large simulation from running on a single GPU, a limitation that could be avoided by introducing support for multiple GPUs in the semi-implicit scheme. Moreover we have seen a decrease in performance for higher viscosities, due to the less favorable conditioning of the coefficient matrix in the linear systems needed to solve for the viscous term, then some benefits could be achieved introducing appropriate preconditioners to improve the convergence of the CG.

Finally we have seen the simulations of some real lava flows, finding a quite large discrepancy with the reference data. We have seen how the uncertainty on the experimentally obtained viscosity law could be a possible cause of discrepancy, showing how its variations are reflected on the behavior of the fluid. This will be rediscussed after further improving the results for the simpler benchmark tests and we will assess that the results are reproduced with a good accuracy. After that we will be able to pass to a second big step, where the simulations will be used to run reverse engineering processes that will help improving the experimentally obtained models of the lava.

Acknowledgments

I would like to thank Dr. Ciro Del Negro, who has directed my research activity during these three years, making my PhD intensively rich of good and forming experiences, with affection, and always creating a warm and familiar atmosphere in our team.

Affectionate thanks to my professor Luigi Fortuna, who has instilled in me the desire to know more and more, and the inspiration to undertake this doctoral experience, contributing to the scientific and organizative aspect of these three years with wisdom and fatherly care.

A big thanks to Dr. Giuseppe Bilotta, my scientific reference and a friend, who has constantly followed my activity, teaching me a lot, both in scientific and cultural field, and has shared with me a lot of nice experiences; Thanks for the unceasing availability and for the contribution that has given to this thesis.

A big thanks to prof. Alexis Hérault, who has importantly contributed to my PhD experience and supervised my activity, sharing with me a lot of friendly and intense moments, and introducing me to many important scientific realities over the world.

A big big thanks to Dr. Annalisa Cappello and Dr. Tania Ganci, for all the nice time spent together in our team, and for the contribution that have given to my activity, helping me to enter and live the world of research with their suggestions, their scientific support and their friendly attitude.

Many thanks to prof. Robert A. Dalrymple, for his important contributions to my PhD and for allowing me to live a wonderful experience at the Northwestern University in Chicago, feeding me with his experience and his scientific grit.

Thanks to prof. Billy Edge for the times he has welcomed me at Coastal Study Institute in North Carolina, sharing with me great moments and interesting scientific activities.

Thanks to Dr. Agnés Leroy, for her suggestions and fruitful discussions, and for being a good mate in our experiences over the world.

Thanks to Dr. Eugenio Rustico, whose contribution has been important for my introduction to GPUSPH, for living with me nice experiences abroad, and gracious moments of low, low, level humor.

Thanks to Dr. Carlo Famoso, Dr. Arturo Buscarino, prof. Mattia Frasca and Dr. Valentina Gambuzza for their suggestions and for the nice moments we spent together.

Thanks to my friend Dr. Elie Saikali for the support given during the preparation of this thesis.

Thanks to my friend, colleague and flatmate Dario Guastella, for all the good moments shared in the last 7 years and the support that has given to me.

Thanks to my father Giovanni, my mother Maria Grazia and my brother Piero for the total backing thay give me everyday.

This work was developed within the framework of ATHOS Research Programme organized by INGV (Catania, Italy) and Johns Hopkins University (Baltimore, USA) and was partially supported from the DPC-INGV 2016 B2 Contract. More references and acknowledgments about GPUSPH can be found in the website http://gpusph.org.

Bibliography

[1] S. Adami, X. Y. Hu and N. A. Adams, *A generalized wall boundary condition for smoothed particle hydrodynamics*, Journal of Computational Physics, vol. 231 (21), pp. 7057–7075, 2012. doi:10.1016/j.jcp.2012.05.005.

[2] N. J. Balmforth, R. V. Craster, P. Perona, A. C. Rust, R. Sassi,*Viscoplastic dam breaks and the bostwick consistometer*, Journal of Non-Newtonian Fluid Mechanics, 142, pp. 63–78, 2007. doi:10.1016/j.jnnfm.2006.06.005.

[3] N. Bernabeu, P. Saramito, C. Smutek. *Numerical modeling of non-Newtonian viscoplastic flows: part ii. viscoplastic fluids and general tridimensional topographies.* International Journal of Numerical Analysis and Modeling, 11, 214-229. 2013.

[4] G. K. Batchelor, *An introduction to fluid mechanics.* Cambridge University Press. 1974.

[5] W. Benz, E. Asphaug, *Impact simulations with fracture*, Method and tests Icarus 1233 98-116, 1994

[6] W. Benz, E. Asphaug, *Simulations of brittle solids using smoothed particle hydrodynamics*, Comput. Phys. Commun., 87 253-65. 1995.

[7] G. Bilotta, *GPU implementation and validation of fully three-dimensional multi-fluid SPH models*, Rapporti Tecnici INGV, 292, 2014. ISSN 2039-7941

[8] G. Bilotta, A. Hérault, A. Cappello, G. Ganci and C. Del Negro, *GPUSPH:a Smoothed Particle Hydrodynamics model for the thermal and rheological evolution on lava flows*, Geological Society, London, v.426, 2015. doi:10.1144/SP426.24.

[9] G. Bilotta, G. Russo, A. Hérault, C. Del Negro, *Moving least-squares corrections for Smoothed Particle Hydrodynamics*, Annals Of Geophysics, 54, 5, 2011; doi: 10.4401/ag-5344

[10] G. Bilotta, A. Vorobyev, A. Hérault, A. Mayrhofer, and D. Violeau, *Modelling real-life flows in hydraulic water-works with GPUSPH*, Proceedings of the 9^{th} International SPHERIC Workshop, pages 335-341, Paris, March 2014.

[11] L. Brookshaw, *A method of calculating heat diffusion in particle simulations*, Astronomical Society of Australia, vol. 6, no. 2, pp. 207-210, 1985.

[12] S. Calvari, H. Pinkerton. *Formation of lava tubes and extensive flow field during the 1991- 1993 eruption of Mount*

Etna. Journal of Geophysical Research, 103, 27 291-27 301. 1998.

[13] A. Cappello A., A. Vicari, C. Del Negro. *Assessment and modeling of lava flow hazard on Mt. Etna volcano*. Bollettino di Geofisica Teorica e Applicata, 52, 2. 2011. doi:10.4430/bgta0003.

[14] A. Cappello, G. Ganci, S. Calvari, N. M. Pérez, P. A. Hernández, S. V. Silva, J. Cabral and C. Del Negro, *Lava flow hazard modeling during the 2014-2015 Fogo eruption, Cape Verde*, Journal of Geophysical Research, vol. 121 (4), 2290-2303, 2013, doi: 10.1002/2015JB012666.

[15] A. Cappello, A. Hérault, G. Bilotta, G. Ganci, C. Del Negro, *MAGFLOW: a physics-based model for the dynamics of lava-flow emplacement*, Geological Society, London, Special Publications, 0305-8719, 426, 2016, doi:10.1144/SP426.16.

[16] P. W. Cleary and J. J. Monaghan, *Conduction modelling using smoothed particle hydrodynamics*, Journal of Computational Physics, vol. 148, pp. 227–264, 1999, doi:10.1006/jcph.1998.6118.

[17] R. H. Cole, *Underwater Explosion*, (Princeton, NJ: Princeton University Press), 1948.

[18] B. Cordonnier, E. Lev, F. Garel, *Benchmarking lava-flow models*. Geological Society, London, 426, Special Publications, 2015.

[19] A. Costa and G. Macedonio, *Computational modelling of lava flows: a review.* In: M. Manga, G. Ventura (eds) *Kinematics and Dynamics of Lava Flows*, Geological Society of America, Special Papers, 396, 209–218, 2005, http://doi.org/10.1130/0-8137-2396-5.209.

[20] G. M. Crisci, S. Di Gregorio, O. Pindaro, G. A. Ranieri. *Lava flow simulation by a discrete cellular model: first implementation.* International Journal of Modelling and Simulation, 6, 137-140. 1986.

[21] A. A. J. C. Crespo, M. Gómez-Gesteira and R. A. Dalrymple, *Boundary conditions generated by dynamic particles in SPH methods*, Cmc -Tech Science Press, 3(3), 2007.

[22] C. Del Negro, A. Cappello, M. Neri, G. Bilotta, A. Hérault and G. Ganci, *Lava flow hazards at Mount Etna: constraints imposed by eruptive history and numerical simulations.* Scientific Reports, vol. 3, 3493, 2013, http://doi.pangaea.de/10.1038/srep03493.

[23] C. Del Negro, A. Cappello, G. Ganci. Quantifying lava flow hazards in response to effusive eruption, GSA Bulletin, v. 128, no. 5/6. p. 752-763. 2016. doi:10.1130/B31364.1.

[24] C. Del Negro, L. Fortuna, A. Hérault, A. Vicari, *Simulations of the 2004 lava flow at Etna volcano using the magflow cellular automata model.* Bulletin of Volcanology, 70, 805-812, 2008. http://doi.org/ 10.1007/s00445-007-0168-8

[25] A. Y. Demianov, A. N. Doludenko, N. A. Inogamov, E. E. Son, *Rayleigh-Taylor instability in a visco-plastic fluid.* Physica Scripta, 2010, T142 (2010) 014026

[26] H. R. Dietterich, K. V. Cashman, A. C. Rust and E. Lev, *Diverting lava flows in the lab*, Nature Geoscience, vol. 8, pp 494-496, 2015, doi:10.1038/ngeo2470.

[27] E. D. Fernández-Nieto, J. Garres-DÃaz, G. Narbona-Reina. A multilayer shallow model for dry granular flows with the μ(i)-rheology: application to granular collapse on erodible beds. Journal of Fluid Mechanics, 798, 643-681. 2016

[28] M. Ferrand, A. Joly,C. Kassiotis, D. Violeau, A. Leroy, F. Morel, B. D. Rogers, *Unsteady open boundaries for SPH using semi-analytical conditions and Riemann solver in 2D*, Computer Physics Communications, 210, 29-44, 2017.http://dx.doi.org/10.1016/j.cpc.2016.09.009

[29] M. Filippucci, M. Dragoni. *Simulation of lava flows with power-law rheology.* Discrete and Continuous Dynamical Systems - Series S, 6, 677-685. 2013.

[30] G. Ganci, A. Cappello, G. Bilotta, A. Hérault, V. Zago, C. Del Negro. *Mapping Volcanic Deposits of the 2011-2015 Etna Eruptive Events Using Satellite Remote Sensing*, Frontiers in Earth Science. 2018. https://doi.org/10.3389/feart.2018.00083.

[31] G. Ganci, M. R. James,S. Calvari, C. Del Negro. *Separating the thermal fingerprints of lavalava fountaining using*

ground-based thermaland SEVIRI measurements flows and simultaneous camera. GEOPHYSICAL RESEARCH LETTERS, VOL. 40, 1?6. 2013. doi:10.1002/grl.50983

[32] G. Ganci, A. Vicari, A. Cappello and C. Del Negro, *An emergent strategy for volcano hazard assessment: from termal satellite monitoring to lava flow modelling*, Remote Sensing of Environment, 119, pp. 197–207, 2013. doi:10.1016/j.rse.2011.12.021.

[33] F. Garel, E. Kaminski, S. Tait, A. Limare, *An experimental study of the surface thermal signature of hot subaerial isoviscous gravity currents: Implications for thermal monitoring of lava flows and domes*, Journal of Geophysical Research, vol. 117, B02205, 2012. doi:10.1029/2011JB008698.

[34] R. Gingold, J. J. Monaghan,*Smoothed Particle hydrodynamics:theory and application to non spherical stars*, Mon. Not. Roy. Astron. Soc., 181, 375-389, 1977.

[35] N. Grenier,*Modélisation Numérique par la méthode SPH de la séparation eau-huile dans les séparateurs gravitaires*, PhD thesis, École Centrale Nantes.

[36] N. Grenier, D. Le Touzé, M. Antuono, A. Colagrossi.*An improved SPH method for multiphase simulations*, In: Proceedings of the 8*th* International Conference on Hydrodynamics (P. Ferrant and X. B. Chen eds.), Nantes, France, 2008.

[37] N. Grenier, M. Antuono, D. Le Touzé, A. Alessandrini.*An hamiltonian interface SPH formulation for multi-fluid and*

free surface flows. Journal of Computational Physics, 228, 8380-8393, 2009.

[38] A. J. L. Harris, S. Rowland, *FLOWGO: a kinematic thermo-rheological model for lava flowing in a channel*. Bulletin of Volcanology, 63, 20-44, 2001. http://doi.org/10.1007/s004450000120

[39] A. J. L. Harris, M. Favalli, R. Wright, H. Garbeil, H*azard assessment at Mount Etna using a hybrid lava flow inundation model and satellite-based land classification*. Natural Hazards, 58, 1001-1027, 2011 http://doi.org/10.1007/s11069-010-9709-0

[40] A. Hérault, *Création d'un système d'information pour la gestion des risques volcaniques*, Informatique [cs]. Université Paris-Est, 2008. Français. tel-00470546

[41] A. Herault, G. Bilotta, R. A. Dalrymple, *SPH on GPU with CUDA*, Journal of Hydraulic Research, vol. 48 (special issue), pp. 74–79, 2010. doi:10.1080/00221686.2010.9641247

[42] A. Hérault, G. Bilotta, A. Vicari, E. Rustico and C. Del Negro, *Numerical simulation of lava flow using a GPU SPH model*, Annals of Geophysics, vol. 54, no. 5, 2011. doi:10.4401/ag-5343.

[43] A. Hérault, G. Bilotta, R. A. Dalrymple, *Achieving the best accuracy in an SPH implementation*, Proceedings of the 9^{th} International SPHERIC Workshop, pages 134-139, Paris, March 2014.

[44] A. Herault, A. Vicari, A. Ciraudo, C. Del Negro. *Forecasting lava flow hazards during the 2006 Etna eruption: using the MAGFLOW cellular automata model.* Computers and Geosciences, 35, 1050-1060, 2009. http://doi.org/10.1016/j.cageo.2007.10.008

[45] M. R. Hestenes, E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, Journal of Research of the National Bureau of Standards. Vol. 49, No 6, 1952.

[46] C. Hirsch, *Numerical Computation of Internal & external flows*, Elsevier, (2007). ISBN: 978-0-7506-6594-0

[47] X. Hu, N. Adams. *A multi-phase SPH method for macroscopic and mesoscopic flows.* Journal of Computational Physics, 213(2), 844-861, 2006.

[48] M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath,M. Teschner, *Implicit Incompressible SPH*, IEEE Transactions on Visualization and Computer Graphics, Volume: 20, Issue: 3, 2014, doi:10.1109/TVCG.2013.105.

[49] H. Jasak, A. Jemcov, Z. Tukovic. *OpenFOAM:a c++ library for complex physics simulations.* In:Terze, Z. (ed.) International Workshop on Coupled Methods in Numerical Dynamics, Dubrovnik, Croatia, 19-21 September 2007. IUC, Dubrovnik, 1-20. 2007.

[50] M. Jubelgas, V. Springel, K. Dolag, *Thermal conduction in cosmological SPH simulations*, Mon. Not. R. Astron. Soc, 2004

[51] M. Kaddiri, M. Naïmi, M. Raji, M. Hasnaoui, *Rayleigh-Bénard convection of non-Newtonian Power-law fluids with temperature-dependent viscosity*, ISRN Thermodynamics. 2012. doi:10.5402/2012/614712

[52] W. Kahan, *Further Remarks on Reducing Truncation Errors*, Communications of the ACM, 8(1):40, 1965. doi:10.1145/363707.363723.

[53] K. Kelfoun, T. Druitt. *Numerical modeling of the emplacement of Socompa Rock Avalanche, Chile.* Journal of Geophysical Research: Solid Earth, 110, B12202, 2005. http://doi.org/10.1029/2005JB003758

[54] K. Kelfoun, S. Vallejo Vargas. *VolcFlow capabilities and perspectives of development for the simulation of lava flows.* In: Harris, A. J. L., De Groeve, T., Garel, F. and Carn, S. A. (eds) Detecting, Modelling and Responding to Effusive Eruptions. Geological Society, London, Special Publications, 426. First published online June 10, 2015, http://doi.org/10.1144/SP426.8

[55] Khronos OpenCL Working Group. The OpenCL Specification [Internet]. 2018. Available from: https://www.khronos.org/registry/OpenCL/ [Accessed: 2018-07-18]

[56] M. Lastiwka, M. Basa, N.J. Quinlan, *Permeable and non-reflecting boundary conditions in SPH*, Int. J. Numer. Meth. Fluids, 61(7):709–724, 2008.

[57] J. R. Lister, *Viscous flows down an inclined plane from point and line sources*, Journal of Fluid Mechanics, 242, pp. 631–653, 1992. doi:10.1017/S0022112092002520.

[58] G. Liu, *Mesh Free Methods:moving beyond the finite elements method*, CRC press, boca raton. 2002.

[59] G. Liu and M. B. Liu, *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*, World Scientific, 2005, 9812564403.

[60] L. Lucy, *A numerical Approach to the Testing of Fission Hypothesis*, The Astronomical Journal, 82 (12), 1013-1024. 1977.

[61] G. Macedonio, A. Costa, A. Longo, *A computer model for volcanic ash fallout and assessment of subsequent hazard.* Computers and Geosciences, 31, 837- 845. 2005.

[62] O. Mahmood, D. Violeau, C. Kassiotis, B. D. Rogers, M. Ferrand, *Absorbing inlet/outlet boundary conditions for 2D SPH turbulent free-surface flows*,International SPHERIC Workshop, pp. 296–302, Prato, may 2012.

[63] D. Molteni, A. Colagrossi, *A simple procedure to improve the pressure evaluation in hydrodynamic context using the SPH*, Computer Physics Communications, 180, pp. 861–72, 2009. doi:10.1016/j.cpc.2008.12.004.

[64] J. J. Monaghan,*On the problem of penetration in particle methods*, Journal of Computational Physics, Volume 82, Issue 1, May 1989, Pages 1-15. https://doi.org/10.1016/0021-9991(89)90032-6.

[65] J. J. Monaghan, *Smoothed particle hydrodynamics* Ann. Rev. Astron. Astrophys. 30 543-74, 1992.

[66] J. J. Monaghan, *Smoothed Particle Hydrodynamics*, Reports on Progress in Physics, vol. 68, pp. 1703–1759, 2005. doi:10.1088/0034-4885/68/8/R01.

[67] J. J. Monaghan, H. E. Huppert and M. G. Worster, *Solidification using smoothed particle Hydrodynamics*, Journal of Computational Physics, vol. 206 (2), pp. 684-705, 2005, doi:10.1016/j.jcp.2004.11.039.

[68] J. J. Monaghan, A. Kos,*Solitary Waves on a Cretan Beach*, Journal of Waterway, Port, Coastal and Ocean Engineering, 125, 145-155. http://dx.doi.org/10.1061/(ASCE)0733-950X(1999)125:3(145).

[69] J. P. Morris, P. J. Fox and Y. Zhu, *Modeling low Reynolds number incompressible flows using SPH*, Journal of Computational Physics, vol. 136 (1), pp. 214–226, 1997, doi:10.1006/jcph.1997.5776.

[70] NVIDIA. CUDA C Programming Guide [Internet]. 2018. Available from: https://docs.nvidia.com/cuda/ [Accessed: 2018-07-18]

[71] ODE, Open Dynamic Engine, 2000-2006 Russell Smith. https://www.ode.org/

[72] A. Parmigiani, C. Huber, O. Bachmann, , B. Chopard. *Pore-scale mass and reactant transport in multiphase porous*

media flows. Journal of Fluid Mechanics, 686, 40-76, 2011. http://doi.org/10.1017/jfm. 2011.268

[73] A. Parmigiani,J. Latt, M. B.Begacem, B. Chopard. *A lattice Boltzmann simulation of the Rhone river*. International Journal of Modern Physics C, 24, 1340008, 2013. http://doi.org/10.1142/S01291831134 00081

[74] C. Proietti, M. Coltelli, M. Marsella, E. Fujita. *A quantitative approach for evaluating lava flow simulation reliability: LavaSIM code applied to the 2001 Etna eruption*. Geochemistry, GeophysicsGeosystems, 10, Q09003, 2009. http://doi.org/10.1029/ 2009GC002426

[75] E. Rustico, G. Bilotta, A. Hérault, C. Del Negro and G. Gallo, *Advances in Multi-GPU Smoothed Particle Hydrodynamics Simulations*, IEEE Transactions on Parallel & Distributed Systems, vol. 25 (1), pp. 43-52, Jan. 2012, doi:10.1109/TPDS.2012.340.

[76] E. Rustico, J. Jankowski, A. Hérault, G. Bilotta and C. Del Negro, *Multi-GPU, multi-node SPH implementation with arbitrary domain decomposition*, Proceedings of the 9^{th} SPHERIC international Workshop, pp 127-133, 2014.

[77] P. Saramito, C. Smutek, B. Cordonnier, *Numerical modeling of shallow non-Newtonian flows: Part I. The 1D horizontal dam break problem revisited*, International Journal of Numerical Analysis and Modeling, Series B, 4, pp. 283–298.

[78] R. Scandone and L. Giacomelli, *Vulcanologia: principi fisici e metodi di indagine*, Liguori editore, 2004, ISBN 88-207-2687-4.

[79] S. Scifoni, M. Coltelli, M. Marsella, C. Proietti, Q. Napoleoni, A. Vicari, C. Del Negro, *Mitigation of lava flow invasion hazard through optimized barrier configuration aided by numerical simulation: The case of the 2001 Etna eruption*, J. Volcanol. Geotherm. Res., 192, 16-26. 2010. doi:10.1016/j.jvolgeores.2010.02.002.

[80] A. Tallarico, M. Dragoni. *A three-dimensional Bingham model for channeled lava flows.* Journal of Geophysical Research, 105, 25 969-25 980. 2000.

[81] A. Vicari, A. Ciraudo, C. Del Negro, A.vHerault, L. Fortuna, *Lava flow simulations using effusion rates from thermal infrared satellite imagery during the 2006 Etna eruption*, Nat. Hazards, 50, 539-550. 2009. doi:10.1007/s11069-008-9306-7.

[82] A. Vicari, A. Hérault, C. Del Negro, M. Coltelli, M. Marsella, C. Proietti, *Modeling of the 2001 lava flow at Etna volcano by a cellular automata approach.* Environmental Modelling and Software, 22, 1465-1471, 2007 http://doi.org/10.1016/j.envsoft.2006.10.005

[83] A. Vicari, G. Ganci, B. Behncke, A. Cappello, M. Neri and C. Del Negro, *Near-realt-time forecasting of lava flow hazards during the 12-13 January 2011 Etna eruption*, Geophysical Research Letters, 38, L13317, 2011, http://doi.org/10.1029/2011GL047545.

[84] J. P. Vila, *On particle weighted methods and SPH*, Mathematical Models and Methods in Applied Sciences, 09(02):161-209, 1999, doi:10.1142/S0218202599000117.

[85] G. P. L. Walker, *Lengths of lava flow* Phil. Trans. Roy. Soc., 274, 107-118, 1973

[86] Z. Wei, R. A. Dalrymple, E. Rustico, A. Hérault and G. Bilotta, *Simulation of nearshore tsunami breaking by smoothed particle hydrodynamics method*, Journal of Waterway, Port, Coastal, and Ocean Engineering, vol. 142 (4), 2016.

[87] H. Wendland, *Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree*, Advances in Computational Mathematics, 4(1), 389-396.

[88] P. Young, G. Wadge. *FLOWFRONT: simulation of a lava flow.* Computers and Geosciences, 16, 1171-1191, 1990 http://doi.org/10.1016/0098-3004(90)90055-X

[89] V. Zago, G. Bilotta, A. Cappello, R. A. Dalrymple, L. Fortuna, G. Ganci, A. Hérault and C. Del Negro *Recent advances in the GPUSPH model for the therma and rheological evolution of lava flows* , Poster at European Geoscience Union General Assembly, Wien, Austria, 17-22 April 2016.

[90] V. Zago, G. Bilotta, A. Cappello, R. A. Dalrymple, L. Fortuna, G. Ganci, A. Hérault and C. Del Negro *Implicit Integration of the Viscous term and GPU implementation in*

GPUSPH for lava flows, Proceedings of the 12th SPHERIC international Workshop, Ourense, Spain, 2017.

[91] V. Zago, G. Bilotta, A. Cappello, R. A. Dalrymple, L. Fortuna, G. Ganci, A. Hérault and C. Del Negro *Benchmarking of the GPUSPH particle engine on lava flows* , Proceedings of the 13th SPHERIC international Workshop, Galway, Ireland, 2018.

[92] V. Zago, G. Bilotta, A. Cappello, R. A. Dalrymple, L. Fortuna, G. Ganci, A. Hérault and C. Del Negro *Preliminary validation of lava benchmark tests on the GPUSPH particle engine*, accepted by Annals of Geophysics, 2018.

[93] V. Zago, G. Bilotta, A. Hérault, R. A. Dalrymple, L. Fortuna, A. Cappello, G. Ganci, C. Del Negro, *Semi-implicit 3D SPH on GPU for lava flows.* Journal of Computational Physics, in press, 2018.

[94] V. Zago, G. Bilotta, A. Cappello, R. A. Dalrymple, L. Fortuna, G. Ganci, A. Hérault and C. Del Negro, *Simulating Complex Fluids with Smoothed Particle Hydrodynamics*, Annals of Geophysics, 60(6), PH669, 2017. doi:10.4401/ag-7362.

[95] H. Zhu, Y. Kim, D. Kee, *Non-Newtonian fluids with a yield stress.* Journal of Non-Newtonian Fluid Mechanics, 129, 2005.

[96] Q. Zhu, L. Hernquist, Y. Li, *Numerical convergence in Smoothed Particle Hydrodynamics.* The American Astronomical Society, 2015.