École Centrale de Nantes
Laboratoire de recherche en Hydrodynamique, Énergétique et Environnement Atmosphérique

CSI (~~CST?~~) Report, 1st year.
Development of a Finite Volume scheme on an adaptively refined Cartesian grid in the presence of air/water interfaces.

03/06/2020

Tommaso Zanelli

Supervisor : Guillaume Oger
Co-supervisors : Zhe Li (David? Louis? Mathieu?)
Zhe Li/David Le Touzé
Co-supervisors Nextflow: Louis VITTOZ

Contents

Contents ii

List of Figures iii

1 Introduction 1
- 1.1 Motivations . . . 1
- 1.2 History . . . 2

2 The Solver 3
- 2.1 Governing equations . . . 3
- 2.2 Projection Scheme . . . 4
- 2.3 Linear System Solution . . . 6
- 2.4 Time Integration . . . 6
- 2.5 Finite Volumes Formulation . . . 6
- 2.6 Grid Management . . . 7
- 2.7 Parallelization . . . 8
- 2.8 Numerical Fluxes . . . 10
  - 2.8.1 First Order Scheme . . . 11
  - 2.8.2 MUSCL . . . 11
  - 2.8.3 WENO . . . 11
- 2.9 Immersed Boundary . . . 13

3 Multiphase flows 15
- 3.1 Numerical description of the boundary . . . 15
  - 3.1.1 Lagrangian Methods . . . 15
  - 3.1.2 Volume of Fluid Method . . . 16
  - 3.1.3 Level Set Method . . . 16
- 3.2 Evolution algorithm for the boundary . . . 17
- 3.3 Scheme for imposing boundary conditions at the interface . . . 18

4 Implementation 19
- 4.1 Description of the Method . . . 19
  - 4.1.1 The Level Set function . . . 19
  - 4.1.2 Discontinuity management . . . 20
  - 4.1.3 Method summary . . . 20
- 4.2 Level Set advection . . . 21
- 4.3 Variable coefficient Poisson Equation . . . 21
  - 4.3.1 PETSc Projection Scheme . . . 21
  - 4.3.2 Reinitialization of the Solver . . . 23
  - 4.3.3 Variable Density . . . 23

4.4 Multiphase Scheme 23
4.5 Other Modifications 23
4.5.1 Initial Time Step 23
4.5.2 Null Pressure 24

5 Results **25**
5.1 Level Set Advection 25
5.1.1 Zalesak Disk 25
5.2 Poisson Equation 27
5.2.1 Solver Reinitialization Tests 27
5.2.2 Poisson Equation Convergence Tests 28
5.3 Multiphase Flow Tests 29
5.3.1 Hydrostatic Test Case 29
5.3.2 Linear Sloshing 31
5.3.3 Dambreak (2019) 32
5.3.4 Dambreak (2020) 34

6 Conclusions **40**

Bibliography **41**

List of Figures

2.1 Staggered Grid 5
2.2 Multilevel Grid and Quadtree 7
2.3 Morton Keys 8
2.4 Local, Inner and Outer Blocks 9
2.5 Numerical Fluxes 10
2.6 Left and Right Fluxes 11
2.7 Weno Stencils 12
2.8 Regularized Heaviside function 14

3.1 Volume of Fluid 16
3.2 level Set curve 17

5.1 Zalesak Disk 25
5.2 Zalesak Results 26
5.3 Reinitialization CPU Time cost 27
5.4 Reinitialization CPU Time cost 28
5.5 Hydrostatic Test Case Setup 29
5.6 Hydrostatic Test Density Viscosity 29
5.7 Poisson equation convergence study 30
5.8 Linear Sloshing, interface vs. time 31
5.9 Linear Sloshing, t=4 31
5.10 Dambreak Test Setup 32
5.11 Dambreak AMR Grid 32
5.12 Dambreak AMR 2019 33
5.13 Dambreak 2019 Level Set Diffusion 33
5.14 Dambreak 2019 Denser fluid volume vs. time 34
5.15 Dambreak 2020 MUSCL - WENO Pressure field 35
5.16 Dambreak 2020 MUSCL - MUSCL Pressure field 35
5.17 Dambreak 2020 Code t=0s 36
5.18 Dambreak 2020 Code t=0.2s 36
5.19 Dambreak 2020 Code t=0.5s 37
5.20 Dambreak 2020 Code t=1s 37
5.21 Dambreak 2020 Code t=2.5s 38
5.22 Dambreak 2020 Code t=5s 38
5.23 Dambreak 2020 Code t=7.5s 39
5.24 Dambreak 2020 Code t=10s 39

Chapter 1

Introduction

Grid-flow

Gridflow (formerly WCCH) is a CFD solver being jointly developed by Nextflow Software and by the LHEEA laboratory (Laboratoire de recherche en Hydrodynamique, Énergétique et Environnement Atmosphérique) of the École Centrale de Nantes. It employs a high order finite volume discretization on an adaptively refined multi-level Cartesian grid and an immersed boundary method to account for the presence of arbitrarily-shaped bodies inside of the computational domain.
The code is written in Fortran 90 and uses MPI (Message Passing Interface) for parallelization. Optimal scalability on massively parallel architectures is one of the main objectives in its development.
Two variants of Gridflow currently exist: a fully explicit (weakly-)compressible one and a more recent, semi-implicit incompressible one based on Chorin's projection method.
Gridflow's original intended domain of application was the marine industry. In recent years Nextflow Software has widened its offer to other domains, such as the automotive, aeronautical, process, and energy industries, hence the code is now in development with a general puropose approach in mind, however, with a special attention to the problematics typical of the marine sector.
The present work focuses on the implementation of a method for simulating multiphase flows in the incompressible version of Gridflow.

1.1 Motivations

Most commercial CFD software currently on offer for industrial applications employs body-fitted unstructured meshes which, while versatile, remain difficult and time-consuming to generate.
Gridflow's mesh generation procedure is conceptually simpler, making it faster and more straightforward to automate. Dynamic adaptive refinement removes the need to know the positions of the more critical areas of the flow in advance, and the immersed boundary is more suited for changes in the topology of the domain, such as moving bodies or fluid-structure interaction problems.
A further advantage of the simplicity of a Cartesian grid is the possibility of using high order reconstruction schemes for the differential operators and, in the case of finite volumes, for the numerical fluxes.
These advantages make Gridflow suitable for a wider range of industrial problems than more traditional CFD software, while removing the need for complex and time consuming mesh generation procedures.
On the other hand, immersed boundary techniques make the capturing of boundary layers less straightforward, which is especially problematic for turbulence modelling An implemen-

tation of the Spalart-Allmaras turbulence model in Gridflow has recently been developed and is currently under testing, however, this is presently beyond the scope of this work.
Due to the code's intended domain of application, much work went into the development of methods for simulating multiphase flows in the weakly-compressible formulation of the code. However, these methods are not compatible with the more recent incompressible formulation, hence the need to develop a new method.

1.2 History

The development of the weakly-compressible version of Gridflow started in 2008, under the name WCCH (for Weakly Compressible Cartesian Hydrodynamics), at LHEEA. In 2011, The company Hydrocean, a spin-off of École Centrale that specialized in the development and marketing the school's state-of-the-art CFD solvers, started financing the PhD of Pierre Bigay [6]. This work focused on Gridflow's numerical flux reconstruction techniques and on the immersed boundary.
In the subsequent years the code was further expanded through the work of other students and researchers. Relevant to the present work was the PhD thesis of Amélie Bardin [4], which focused on multiphase simulations using the weakly-compressible version of the code and the Level-Set method.
In 2015 Hydrocean's services part was acquired by the Bureau Veritas group, while the software development part was re-incorporated as Nextflow Software.
The incompressible version of Gridflow was developed between 2016 and 2018 during the PhD work of Louis Vittoz [22], financed by Nextflow Software.

Chapter 2

The Solver

2.1 Governing equations

The weakly-compressible formulation of Gridflow is based on the compressible Navier-Stokes equations:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0\,, & \text{(2.1a)} \\ \dfrac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \rho \boldsymbol{f}\,, & \text{(2.1b)} \\ \dfrac{\partial E}{\partial t} + \nabla \cdot [(E + p)\,\boldsymbol{u}] = \nabla \cdot (\boldsymbol{\tau} \boldsymbol{u})\,, & \text{(2.1c)} \end{cases}$$

E the total energy

where the symbol $\otimes$ indicates the tensor product, $E = \rho\left(e + \frac{1}{2}|\boldsymbol{u}|^2\right)$, and $\boldsymbol{\tau}$ is the viscous stress tensor. For a Newtonian fluid, $\boldsymbol{\tau} = \mu\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right) + \boldsymbol{I}\left(\mu_v + \frac{1}{3}\mu\right)\nabla \cdot \boldsymbol{u}$, however, in the code this is approximated as $\boldsymbol{\tau} = \mu \nabla \boldsymbol{u}$.

? $\mu_v = -\frac{2}{3}\mu$ μ_v = volume viscosity

The equation of state used in Gridflow for closing the system is the barotropic Tait equation:

$$p = \frac{\rho_0\, c_0^2}{\gamma}\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right]\,, \tag{2.2}$$

which allows for a decoupling of (2.1a) and (2.1b) from the energy conservation equation (2.1c).

It should be emphasized that compressibility effects were not of interest when the code was originally conceived, and this formulation was chosen primarily because it can be integrated explicitly in time, which is advantageous for parallel scalability. In fact, the speed of sound c_0 is chosen so that the Mach number is less than 0.1 everywhere in the domain, ensuring quasi-incompressibility. The Mach number criterion is an "a priori" criterion.

The weakly-compressible formulation requires very small time steps in order to resolve acoustic perturbations without instabilities. The incompressible formulation was developed due to its less demanding stability requirements.

The incompressible Navier-Stokes equations, in conservative form, are:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) = -\dfrac{\nabla p}{\rho} + \dfrac{1}{\rho}\nabla \cdot \boldsymbol{\tau} + \boldsymbol{f}\,, & \text{(2.3a)} \\ \nabla \cdot \boldsymbol{u} = 0\,. & \text{(2.3b)} \end{cases}$$

The mass conservation condition (2.3b) is not time-dependent. This prevents a straightforward explicit time integration.

To be sure that the flow is quasi-incompressible we have to check "a posteriori" that the density remains globally constant (<1% for the amplitude of the variations)

2.2 Projection Scheme

The first projection method for integrating in time the incompressible Navier-Stokes equations (2.3) was proposed separately by Chorin [8] and Temam [21] in the late 1960s. Since then numerous variants and improvements to the method have been studied. An exhaustive overview can be found in [10]. The scheme implemented in Gridflow is close to original one. Its incremental variant is also implemented as an option.
At the base of a projection method is the Ladyzhenskaya decomposition theorem (also known as Helmholtz-Hodge decomposition theorem), which states that, for a sufficiently regular, decaying vector field $\boldsymbol{u}$ defined over a simply connected domain, there exist uniquely an irrotational vector field $\boldsymbol{u}_{irr}$ and a solenoidal vector field $\boldsymbol{u}_{sol}$ such that:

$$\boldsymbol{u} = \boldsymbol{u}_{irr} + \boldsymbol{u}_{sol} \,. \tag{2.4}$$

An irrotational vector field can be expressed as the gradient of a "potential" scalar field $\boldsymbol{u}_{irr} = \nabla\phi$. Taking the divergence of (2.4), since $\nabla \cdot \boldsymbol{u}_{sol} = 0$, one obtains the Poisson equation:

$$\nabla^2\phi = \nabla \cdot \boldsymbol{u} \,. \tag{2.5}$$

Its solution allows for the fields $\boldsymbol{u}_{irr}$ and $\boldsymbol{u}_{sol}$ to be found.
Assuming, for simplicity, that (2.3) are integrated in time using a forward Euler scheme:

$$\begin{cases} \dfrac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} = -\dfrac{\nabla p^{n+1}}{\rho} + \boldsymbol{RHS}^n \,, & \text{(2.6a)} \\ \nabla \cdot \boldsymbol{u}^{n+1} = 0 \,, & \text{(2.6b)} \end{cases}$$

with $\boldsymbol{RHS}^n = \nabla\cdot(\boldsymbol{u}^n \otimes \boldsymbol{u}^n) + \mu\nabla^2\boldsymbol{u}^n + \boldsymbol{f}^n$, it is possible to adopt a fractional step approach:

$$\begin{cases} \dfrac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = \boldsymbol{RHS}^n \,, & \text{(2.7a)} \\ \dfrac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t} = -\dfrac{\nabla p^{n+1}}{\rho} \,, & \text{(2.7b)} \\ \nabla \cdot \boldsymbol{u}^{n+1} = 0 \,. & \text{(2.7c)} \end{cases}$$

It can be noted that (2.7a) and (2.7c) can be interpreted as the Ladyzhenskaya decomposition of $\boldsymbol{u}^*$. This provides means to compute the pressure field: by taking the divergence of (2.7b), a Poisson equation in the form of (2.5) is obtained, with its result being the pressure field. This also ensures that the end-of-step velocity field $\boldsymbol{u}^{n+1}$ satisfies the mass conservation equation (2.3b).
A time-step in Chorin's projection scheme can be summed up in three passages:

1. Compute the intermediate velocity field:

$$\boldsymbol{u}^* = \boldsymbol{u}^n + \Delta t \boldsymbol{RHS}^n \,, \tag{2.8a}$$

2. Solve the Poisson equation for the pressure:

$$\nabla^2 p^{n+1} = \frac{\rho}{\Delta t}\nabla \cdot \boldsymbol{u}^* \,, \tag{2.8b}$$

3. Apply the pressure gradient to correct the velocity field:

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \frac{\Delta t}{\rho}\nabla p^{n+1} \,. \tag{2.8c}$$

* only in the case when all boundary condition of the domain are Neumann BC. But for external flows we prescribe the pressure at the outlet ($p = 0$)

A critical aspect of Chorin's projection method is boundary conditions. Dirichlet boundary conditions on the velocity translate to Neumann boundary conditions on the Pressure. This may leave equation (2.8b) underdetermined. In Gridflow, this is solved by imposing a null pressure at one point on the border of the domain.* Another problem with the boundary conditions is that they can only be applied directly on the intermediate velocity field $\boldsymbol{u}^*$ and on the pressure p^{n+1}, but affect $\boldsymbol{u}^{n+1}$ only through equation (2.8c). This means that only the component of $\boldsymbol{u}^{n+1}$ normal to the boundary satisfies the boundary conditions on the velocity. While $\boldsymbol{u}^{n+1}$ can be corrected after (2.8c), this produces a numerical boundary layer which reduces the accuracy of the scheme. / citation Goda 1979 "A multi-step ..."

The incremental version of Chorin's projection scheme differs by the inclusion of the gradient of the pressure at the previous time step in the right-hand side of (2.8a):

1. Compute the intermediate velocity field:

$$\boldsymbol{u}^* = \boldsymbol{u}^n - \frac{\Delta t}{\rho}\nabla p^n + \Delta t \boldsymbol{RHS}^n\,, \tag{2.9a}$$

2. Solve the Poisson equation for the pressure increment:

$$\nabla^2 \phi = \frac{\rho}{\Delta t}\nabla \cdot \boldsymbol{u}^*\,, \tag{2.9b}$$

3. Apply the pressure increment gradient to correct the vector field:

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \frac{\Delta t}{\rho}\nabla \phi\,, \tag{2.9c}$$

4. Add the pressure increment to the pressure field:

$$p^{n+1} = p^n + \phi\,. \tag{2.9d}$$

It can be noted that, if the pressure field at the beginning of the computation is null, (2.9) is algebraically equivalent to (2.8). However, in practice, with finite arithmetics, the incremental formulation can be advantageous, and it is the one employed in [22] when using Gridflow's immersed boundary method.

The differential operators divergence ($\nabla\cdot$) and gradient (∇) are implemented in the code through the use of centred finite differences. To avoid chequerboard instabilities, a staggered

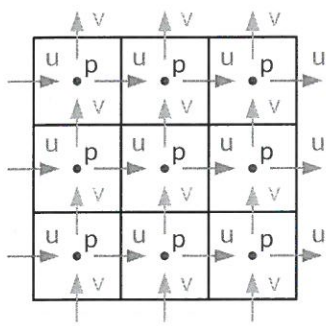


Figure 2.1 – Staggered grid: placement of the pressure and of the velocity components.

grid is used: the pressure is located at the centre of a cell, while the components of the velocity are placed at the cell faces, as shown in figure 2.1. This technique was first proposed in [11]. The centred finite difference gradient operator applied to a cell-centred variable

places the resulting vector field at the cell faces. The discrete divergence operator applied to a vector field whose components are at the cell faces, produces a cell-centred scalar field. The two operators are represented as sparse matrices in the code. The gradient matrix is obtained as the transpose of the divergence matrix, and the Laplacian matrix as their product.

2.3 Linear System Solution

The weakly-compressible version of the code is integrated in time explicitly. This is not possible in the incompressible version. While steps (2.8a) and (2.8c) of the projection scheme can be explicit, the Poisson equation for the pressure (2.8b) requires the solution of a linear system of equations.
For this purpose, the open source library PETSc [1][2], (Portable, Extensible Toolkit for Scientific Computation) developed by Argonne National Laboratory, was integrated into the code. This library, written in C, is optimized for parallel scalability, hence responding to the needs of Gridflow. PETSc offers many options for solving linear systems. for Gridflow, a GMRES (Generalized Minimal Residual) method coupled with an algebraic multigrid was chosen. restart
It was shown by Vittoz [22] that, while the need to solve the pressure Poisson equation in the incompressible formulation increases the CPU time needed for an individual time-step, compared to the weakly-compressible formulation, it does not impact excessively the parallel scalability of the code.

2.4 Time Integration

At present, two time integration options exist in Gridflow: forward Euler and a fourth-order Runge-Kutta scheme. The latter ise characterized by the Butcher tableau:

$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array} \tag{2.10}$$

may you briefly describe the meaning of the rows and the columns?

In the incompressible version, all steps of the projection scheme are executed for each stage of the Runge-Kutta algorithm.

2.5 Finite Volumes Formulation

A set of hyperbolic equations can be re-formulated as:

$$\frac{\partial \boldsymbol{W}}{\partial t} + \nabla \cdot \boldsymbol{\Psi}(\boldsymbol{W}) = \boldsymbol{S}\,, \tag{2.11}$$

where the vector $\boldsymbol{W}$ collects all the conserved quantities, tensor $\boldsymbol{\Psi}(\boldsymbol{W})$ represents the fluxes, and $\boldsymbol{S}$ is a source term. The systems of equations (2.1) and (2.3), while not strictly hyperbolic in nature, can still be re-formulated in the same way by defining the viscous fluxes $\boldsymbol{\Psi}_v(\boldsymbol{W}) = \boldsymbol{\tau}$ in addition to the advective ones.
The *finite volume* method for solving a set of PDEs considers a partitioning of the computational domain into a set of geometrical elements, or control volumes:

$$\Omega = \bigcup_{i=1}^{N} \Omega_i\,. \tag{2.12}$$

Integrating (2.11) over a control volume Ω_i, then applying Reynolds' transport theorem to the time derivative and the divergence theorem to the fluxes tensor produces:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_i} \boldsymbol{W}\mathrm{d}\Omega - \int_{\partial\Omega_i} \boldsymbol{W}\left(\boldsymbol{u}_{\Omega_i}\cdot\boldsymbol{n}\right)\mathrm{d}S + \int_{\partial\Omega_i} \boldsymbol{\Psi}\left(\boldsymbol{W}\right)\cdot\boldsymbol{n}\mathrm{d}S = \int_{\partial\Omega_i} \boldsymbol{S}\mathrm{d}\Omega\,, \tag{2.13}$$

where $\boldsymbol{u}_{\Omega_i}$ is the speed of the frontier of the control volume, which is usually null.
The frontier of each control volume can be expressed as: $\partial\Omega_i = \bigcup_{j=1}^{M} A_{ij}$, therefore:

$$\int_{\partial\Omega_i} \boldsymbol{\Psi}\left(\boldsymbol{W}\right)\cdot\boldsymbol{n}\mathrm{d}S = \sum_{j=1}^{M}\int_{A_{ij}} \boldsymbol{\Psi}\left(\boldsymbol{W}\right)\cdot\boldsymbol{n}\mathrm{d}S = \sum_{j=1}^{M}\boldsymbol{F}_{ij}\,A_{ij}\,,$$

where $\boldsymbol{F}_{ij}$ is the *numerical flux* across face A_{ij}. Defining the volume averages:

$$\overline{\boldsymbol{W}}_i = \frac{1}{\Omega_i}\int_{\Omega_i}\boldsymbol{W}\mathrm{d}\Omega\,; \qquad \overline{\boldsymbol{S}}_i = \frac{1}{\Omega_i}\int_{\Omega_i}\boldsymbol{S}\mathrm{d}\Omega\,;$$

(2.11) can be re-written as:

$$\frac{\mathrm{d}\overline{\boldsymbol{W}}_i}{\mathrm{d}t} = -\sum_{j=1}^{M}\frac{A_{ij}}{\Omega_i}\boldsymbol{F}_{ij} + \overline{\boldsymbol{S}}_i\,, \tag{2.14}$$

therefore transforming a continuous PDE into a set of discrete relations between the volume-averaged values $\overline{\boldsymbol{W}}_i$.
Two aspects remain to be defined: how the domain is partitioned into control volumes, and how the numerical fluxes $\boldsymbol{F}_{ij}$ are reconstructed from the volume-averaged values $\overline{\boldsymbol{W}}_i$.

2.6 Grid Management

The finite volume formalism can be applied to control volumes of any shape. In fact, it is commonly used by commercial CFD software based on unstructured grids.
Gridflow, on the other hand uses rectangular control volumes (in two dimensions) or rectangular prisms (in three dimensions). The computational domain is partitioned recursively into

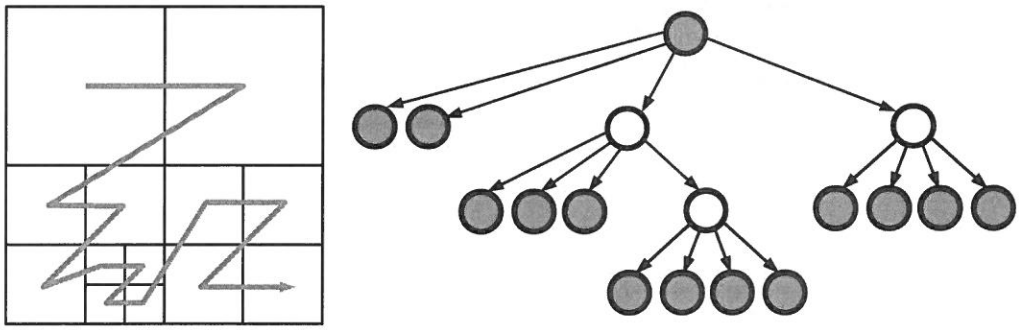


Figure 2.2 – An example of two-dimensional multi-level Cartesian grid, showing the Z-order curve and the quadtree structure.

blocks, with the first block corresponding to the whole domain. In subsequent refinement passages, each block is subdivided into four (or eight, in three dimensions) smaller *children*

blocks until the desired level of refinement is reached. Blocks generated in this way are naturally organized in a quadtree (or octree, in three dimensions) graph, originating from the initial (root) block. Blocks that have no children are marked as leaf blocks. These blocks are not necessarily at the same level of refinement, as refinement criteria can vary across the domain, however, neighbouring blocks cannot differ in their level of refinement of more than one.
The leaf blocks are ordered in memory so that they follow a Z-order curve in the domain. This allows for an efficient way to refer to their position: interleaving the binary representations of their spatial coordinates returns their position (Morton key) along the Z-order curve.
Leaf blocks are further divided into a fixed number of cells, with a minimum of two per

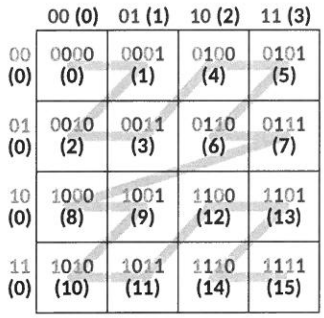


Figure 2.3 – How the interleaving of binary coordinates produce the Morton keys.

direction. These cells constitute the control volumes over which the finite volume formalism is applied.
This method of adaptive mesh refinement (or AMR) is very versatile in semi-automatically producing a grid which is refined locally where needed, while still preserving the convenience of the Cartesian structure.

2.7 Parallelization

The Z-order curve preserves locality. This means that, partitioning the domain along it, subdomains constituted by (mostly) adjacent blocks are obtained, which is of great aid for parallelization
The MPI library uses a *distributed memory* model: the execution of the program is split into semi-independent processes, each acting on a portion of the data and exchanging information at specific points in the algorithm. Parallel performance is negatively affected by the amount of information exchanged between processes, which should be kept to a minimum. This is due to data transfer between processors being slower than data transfer between a processor and its allotted memory.
Another aspect of parallelization is whether communication between processes is synchronous or asynchronous. In synchronous communication the execution of a process stops at every communication event, until its completion. Gridflow employs asynchronous communication: execution continues normally after data is sent to a different process, stopping only if required data has not been received yet.
Each process is allotted an equal number of blocks. The blocks in a process can be classified as local, inner and outer ones. Local and inner blocks are the ones in which each process performs operations, with inner blocks being at the boundaries of the subdomain allotted to the process. Outer blocks correspond to the inner blocks of the adjacent processes, and

contain information needed to operate on the inner blocks. In short, each process sends the data of its inner blocks to the others, and fills its outer blocks with the data received from them. Each communication event in Gridflow follows roughly the same steps:

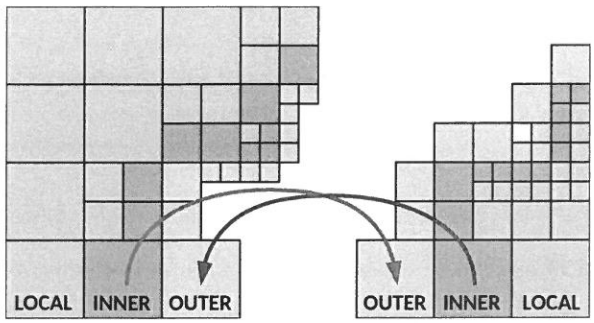


Figure 2.4 – The subdivision into local, inner and outer blocks in a process: inner and outer values are exchanged between processes.

1. Any operation is performed first on the inner blocks. Gridflow's explicit nature means that all required data is already present.
2. Updated information from the inner blocks is sent to the other processes.
3. The operation is performed on the local blocks. This part does not require any inter-process communication.
4. Updated information for the outer blocks is received by the other processes.

Asynchronous execution means that different processes can be at different stages of an operation. Inner blocks are operated on, and their data sent as soon as possible so that, if any process is ahead of the sending one, it can receive the data it needs and continue. Operations on the local blocks (which take most of the computational time) are performed before the receiving step to give time to any process that may be lagging behind to send its inner block data.
It should be noted that if processes run at significantly different speeds, the code will only run as fast as the slowest one. However, for the unavoidable variations in speed that are to be expected during a computation, this strategy reduces idle times.
One operation that does not employ the scheme above is the solution of the pressure equation. The PETSc library is designed to operate as a "black box", although it employs distributed memory and asynchronous communications as well.
With PETSc, parallel vectors are defined, whose elements are distributed among all processes. It is possible to define the overall size of the vector and let PETSc decide how to spread it, or to define the size of its portions for each process. The second option is employed in Gridflow, which allows each process to interact with the portion of a PETSc vector corresponding to its subdomain. Central to PETSc is also the definition of parallel sparse matrices, whose rows are spread among processes. These are stored in memory efficiently by keeping track of non-zero values only, and of their positions in the matrices. However, if the non-zero pattern of a sparse matrix changes, it will have to be reallocated in memory, at a performance cost.

2.8 Numerical Fluxes

Control volumes in Gridflow, as mentioned in 2.6, are either rectangles or rectangular prisms. This means that equation (2.14) can be re-written as:

$$\begin{aligned}\frac{\mathrm{d}\overline{\boldsymbol{W}}_{i,j,k}}{\mathrm{d}t} = &-\frac{\cancel{\Delta y}\cancel{\Delta z}}{\Delta x\cancel{\Delta y}\cancel{\Delta z}}\left(\boldsymbol{F}_{i+\frac{1}{2},j,k}-\boldsymbol{F}_{i-\frac{1}{2},j,k}\right)-\\ &-\frac{\cancel{\Delta z}\cancel{\Delta x}}{\Delta y\cancel{\Delta z}\cancel{\Delta x}}\left(\boldsymbol{G}_{i,j+\frac{1}{2},k}-\boldsymbol{G}_{i,j-\frac{1}{2},k}\right)-\\ &-\frac{\cancel{\Delta x}\cancel{\Delta y}}{\Delta z\cancel{\Delta x}\cancel{\Delta y}}\left(\boldsymbol{H}_{i,j,k+\frac{1}{2}}-\boldsymbol{H}_{i,j,k-\frac{1}{2}}\right)+\overline{\boldsymbol{S}}_{i,j,k-\frac{1}{2}}\,,\end{aligned} \tag{2.15}$$

where $\boldsymbol{F}$, $\boldsymbol{G}$ and $\boldsymbol{H}$ are the fluxes along the cartesian directions, as shown in figure 2.5. If one

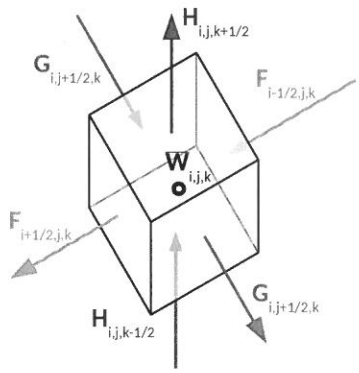


Figure 2.5 – The numerical fluxes entering and exiting of a three dimensional control volume.

were to substitute the fluxes with the values of $\boldsymbol{\Psi}\left(\overline{\boldsymbol{W}}_{i,j,k}\right)$ at the adjacent cells, the terms on the right-hand-side of equation (2.15) would reduce to centred finite differences. This is problematic as it causes instability. Considering the one-dimensional transport equation:

$$\frac{\partial y}{\partial t}+u\frac{\partial y}{\partial x}=0\,, \tag{2.16}$$

it is easy to show that a centred forward Euler scheme in the form:

$$y_i^{n+1}-y_i^n=-u\frac{\Delta t}{2\Delta x}\left(y_{i+1}^n-y_{i-1}^n\right) \tag{2.17}$$

is unconditionally unstable. This brings the necessity of breaking the symmetry in the stencil for the right-hand side. The usual strategy is *upwinding*: given that the velocity field identifies a propagation direction for information, the sign of the velocity can be used to chose which direction to skew the stencil. For (2.16), this can be done as:

$$y_i^{n+1}-y_i^n=-u\frac{\Delta t}{\Delta x}\left(y_{i+1}^n-y_i^n\right) \qquad \text{if } u<0\,, \tag{2.18a}$$

$$y_i^{n+1}-y_i^n=-u\frac{\Delta t}{\Delta x}\left(y_i^n-y_{i-1}^n\right) \qquad \text{if } u>0\,, \tag{2.18b}$$

which is first order accurate in space, as opposed to the second order accuracy of a centred difference, but stable for sufficiently low $\Delta t/\Delta x$ ratios.
One aspect of upwinding is the introduction of numerical diffusion, which contributes to stability by damping spurious oscillations. However, excessive damping may penalize the

accuracy of the solution.
In Gridflow, at each interface between cells two fluxes are reconstructed: a left value, computed on a stencil centred on the cell on the left of the interface, and a right value, computed on a stencil centred on the cell on the right (figure 2.6).
The flux that goes into (2.15) is, in general, a combination of the left and right values, that

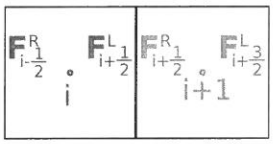


Figure 2.6 – Left and right reconstructed fluxes at the interface between two cells.

keeps into account the local signs of $\boldsymbol{W}$. Many options are implemented in Gridflow for this combination, referred to as *hyperbolic solvers*, however, for the incompressible formulation, the flux is simply chosen as:

$$\begin{cases} \boldsymbol{F}_{i+\frac{1}{2},j,k} = \boldsymbol{F}^L_{i+\frac{1}{2},j,k} & \text{if } u_{i+\frac{1}{2},j,k} > 0\,, \quad (2.19a) \\ \boldsymbol{F}_{i+\frac{1}{2},j,k} = \frac{1}{2}\left(\boldsymbol{F}^L_{i+\frac{1}{2},j,k} + \boldsymbol{F}^R_{i+\frac{1}{2},j,k}\right) & \text{if } u_{i+\frac{1}{2},j,k} = 0\,, \quad (2.19b) \\ \boldsymbol{F}_{i+\frac{1}{2},j,k} = \boldsymbol{F}^R_{i+\frac{1}{2},j,k} & \text{if } u_{i+\frac{1}{2},j,k} < 0\,, \quad (2.19c) \end{cases}$$

with analogous expressions for the other two directions. What is still left undefined is how the left and right reconstructions of the fluxes are computed. In Gridflow, the flux reconstruction methods available are:

2.8.1 First Order Scheme

The simplest possible strategy is using the values at the centre of the left cell for the left flux and the one in the right cell for the right flux. This would produce a first order accurate approximation analogous to (2.18).

2.8.2 MUSCL

MUSCL stands for Monotonic Upstream-centred Scheme for Conservation Laws. Given the variable u_i, representing a generic quantity whose flux needs to be reconstructed at cell faces, the MUSCL scheme defines the left and right fluxes as:

$$\begin{cases} u^R_{i+\frac{1}{2}} = u_{i+1} + \frac{1}{2}\Theta\left(\kappa_{i+1}\right)\left(u_{i+2} - u_{i+1}\right)\,, & (2.20a) \\ u^L_{i+\frac{1}{2}} = \quad u_i + \frac{1}{2}\Theta\left(\kappa_i\right)\left(u_{i+1} - u_i\right)\,, & (2.20b) \end{cases}$$

where $\kappa_i = \frac{u_i - u_{i-1}}{u_{i+1} - u_i}$ and the function $\Theta(\kappa)$ is referred to as *limiter*. A vast array of limiters have been proposed over the years. A simple one implemented in Gridflow is the *Minmod* limiter: $\Theta(\kappa) = \max[0, \min(1, \kappa)]$. If the limiter satisfies certain conditions, the MUSCL scheme is second order accurate in space and total variation diminishing.

2.8.3 WENO

WENO stands for Weighted essentially non-oscillatory. The idea at the base of the method is to consider three different stencils around a cell, as in figure 2.7.

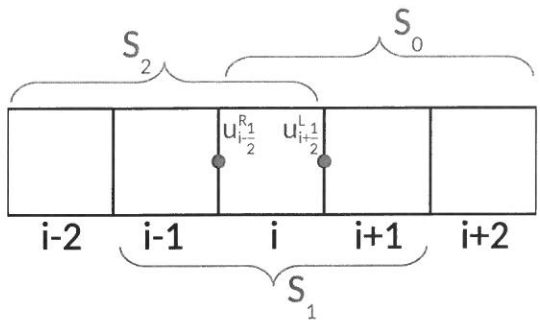


Figure 2.7 – The three stencils used for the WENO reconstruction.

A third order accurate approximation of the fluxes u at the cell faces is possible on each one of these stencils.
The reconstructed fluxes are obtained as a weighted combination of these three approximations:

$$\begin{cases} u^L_{i+\frac{1}{2}} = \omega^L_0 \left(\frac{1}{3}u_i + \frac{5}{6}u_{i+1} - \frac{1}{6}u_{i+2}\right) + \omega^L_1 \left(-\frac{1}{6}u_{i-1} + \frac{5}{6}u_i + \frac{1}{3}u_{i+1}\right) + \\ \qquad + \omega^L_2 \left(\frac{1}{3}u_{i-2} - \frac{7}{6}u_{i-1} + \frac{11}{6}u_i\right) , \\ u^R_{i-\frac{1}{2}} = \omega^R_0 \left(\frac{11}{6}u_i - \frac{7}{6}u_{i+1} + \frac{1}{3}u_{i+2}\right) + \omega^R_1 \left(\frac{1}{3}u_{i-1} + \frac{5}{6}u_i - \frac{1}{6}u_{i+1}\right) + \\ \qquad + \omega^R_2 \left(-\frac{1}{6}u_{i-2} + \frac{5}{6}u_{i-1} + \frac{1}{3}u_i\right) . \end{cases} \tag{2.21}$$

The base values of the weights are: $d^L_0 = \frac{3}{10}$, $d^L_1 = \frac{3}{5}$, $d^L_2 = \frac{1}{10}$, $d^R_0 = \frac{1}{10}$, $d^R_1 = \frac{3}{5}$, and $d^R_2 = \frac{3}{10}$.
With $\omega^L_i = d^L_i$ and $\omega^R_i = d^R_i$ for $i = 0, 1, 2$, reconstructions (2.21) would be fifth order accurate. The weights, however, are chosen in order to penalize or prioritize certain stencils, based on how regular u is on them. The three regularity indicators

$$\begin{cases} \mathrm{IS}_0 = \frac{13}{12}\left(u_{i+2} - 2u_{i+1} + u_i\right)^2 + \frac{1}{4}\left(u_{i+2} - 4u_{i+1} + 3u_i\right)^2 , \\ \mathrm{IS}_1 = \frac{13}{12}\left(u_{i+1} - 2u_i + u_{i-1}\right)^2 + \frac{1}{4}\left(u_{i+1} - u_{i-1}\right)^2 , \\ \mathrm{IS}_2 = \frac{13}{12}\left(u_i - 2u_{i-1} + u_{i-2}\right)^2 + \frac{1}{4}\left(3u_i - 4u_{i-1} + u_{i-2}\right)^2 \end{cases} \tag{2.22}$$

are defined, and the weights are computed as:

$$\alpha_i = \frac{d^L_i}{\mathrm{IS}_i + \varepsilon} , \qquad \omega^L_i = \frac{\alpha_i}{\sum_{j=1}^3 \alpha_j} , \qquad i = 0, 1, 2 , \tag{2.23}$$

whth $\varepsilon = 10^{-6}$ to avoid divisions by zero, and the same procedure applied to ω^R_i.
This weight definition is meant to prevent oscillations due to shocks in the solution. A discontinuity in u would cause the local regularity indicator to spike in value, in turn reducing the weight associated to the corresponding stencil. (2.21) would still produce a third order accurate approximation while avoiding the discontinuity altogether. On more regular portions of u, the indicators would be lower and the weights closer to their default values, allowing the reconstruction (2.21) to reach (ideally) fifth-order accuracy.

In Gridflow, the implementation allows the user to chose whether to use the smoothness indicators or not. If they are active, the reconstruction is performed as described above. If not, the weights are automatically set to their default values for fifth order accuracy.

The previous discussion presupposes a uniform grid, however, this is not always the case in Gridflow. Given that adjacent blocks differ by at most one level of refinement, and that they contain at least two cells per direction, the number of possible "configurations" is relatively limited.
For the weakly-compressible version of Gridflow, coefficients for a variable-step WENO reconstruction are defined.
For the incompressible version of the code, things are further complicated by the adoption of the *staggered* grid.
As shown in figure 2.1, the components of the velocity are no longer placed at the centre of a cell but at its faces. On a uniform staggered grid, the control volumes for each component of the velocity are shifted along the corresponding direction, but maintain the same shape. While the fluxes along directions normal to the component of the velocity considered require some averaging, the WENO scheme is mostly unaltered. The situation is different for non-uniform grids, on which the staggering complicates the flux reconstruction procedure. The strategy adopted in Gridflow consists in adapting the shape of the control volumes at the interfaces between different levels of refinement, and excluding some stencils.
The staggered grid causes an incompatibility between the WENO scheme used in the weakly-compressible formulation and the WENO scheme that had to be developed for the incompressible one.

2.9 Immersed Boundary

While not relevant at the present stage of this work, the immersed boundary technique implemented in the code deserves a mention.
At the base of the method is the definition, for each cell i, of a quantity H_i representing the fraction of the cell volume occupied by the solid body. Ideally, its value is one inside the solid body, zero in the fluid and in between at the interface between them. The total momentum of a cell will be the sum of the momentum of the fluid and of that of the solid body, and the same will be true of its variation in time:

$$\left.\frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{d}t}\right|^n = (1-H_i)\left.\frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{d}t}\right|^n_{fluid} + H_i\left.\frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{d}t}\right|^n_{solid}, \tag{2.24}$$

assuming uniform density. The motion of the fluid is given by the time integration of the Navier-Stokes equations, while that of the solid is usually given:

$$\left.\frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{d}t}\right|^n_{fluid} \approx \boldsymbol{RHS}_i^n, \qquad \left.\frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{d}t}\right|^n_{solid} \approx \boldsymbol{a}_i^n.$$

Combining the previous expressions gives:

$$\left.\frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{d}t}\right|^n \approx (1-H_i)\,\boldsymbol{RHS}_i^n + H_i\boldsymbol{a}_i^n = \boldsymbol{RHS}_i^n + \boldsymbol{S}_i^n, \tag{2.25}$$

with $\boldsymbol{S}_i^n = H_i\left(\boldsymbol{a}_i^n - \boldsymbol{RHS}_i^n\right)$ the force that the solid applies on the fluid. By substituting $\boldsymbol{RHS}_i^n$ and $\boldsymbol{a}_i^n$ with the finite difference expressions of the velocity time derivatives gives the alternative expression for $\boldsymbol{S}_i^n$:

$$\boldsymbol{S}_i^n = H_i\left(\frac{\boldsymbol{u}_i^{n+1}\big|_{solid} - \boldsymbol{u}_i^n}{\Delta t} - \frac{\boldsymbol{u}_i^{n+1}\big|_{fluid} - \boldsymbol{u}_i^n}{\Delta t}\right) = H_i\frac{\boldsymbol{u}_i^{n+1}\big|_{solid} - \boldsymbol{u}_i^{n+1}\big|_{fluid}}{\Delta t}, \tag{2.26}$$

which shows that the force represents a "correction" of the velocity field.
In practice, H_i is smoothed around the interface. In Gridflow, this is done using the regularized Heaviside function:

$$H(\phi) = \frac{1}{2}\left[1 + \tanh\left(\alpha \frac{\phi}{\Delta x}\right)\right], \tag{2.27}$$

where ϕ is the distance from the interface, Δx the local cell size and α a parameter defining the span of the transition zone.
It should be noted that in the incompressible version of the code, two conflicting corrections are applied to the velocity field, one at the end of the projection scheme, to impose mass conservation, and one to account for the presence of the body. In [22] the problem of which correction to apply first is considered.

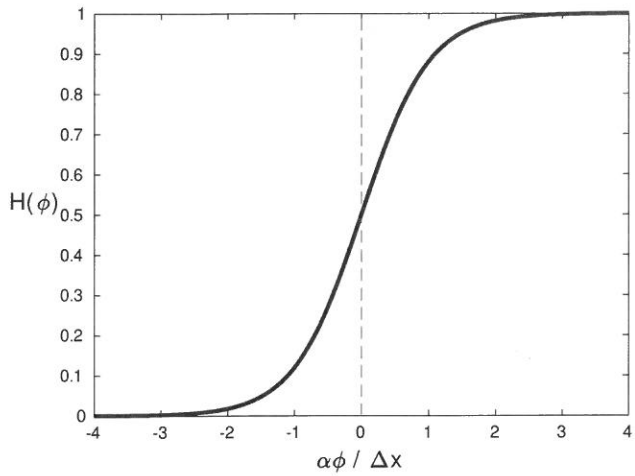


Figure 2.8 – The regularized Heaviside function (2.27).

Chapter 3

Multiphase flows

In this context, a multiphase flow is defined as a flow involving two (or more) immiscible fluids, having different properties (i.e. density, viscosity).
Hirt and Nichols, in their original article presenting the Volume of Fluid method [13], proposed a breakdown of the problem of simulating multiphase flows into three main aspects:

- Numerical description of the location and shape of the boundary between the fluids.
- Evolution algorithm for the boundary.
- Scheme for imposing boundary conditions at the interface.

3.1 Numerical description of the boundary

Numerous strategies exist for describing the evolution of the interface between the fluids, with many variations and hybrid approaches between them. A first distinction can be made between Lagrangian methods, in which the interface is followed directly, and Eulerian methods, in which an additional quantity is introduced to keep track of the interface. The two principal Eulerian methods are the Volume-of-Fluid method and the Level Set method.

3.1.1 Lagrangian Methods

For codes that employ an Eulerian description of the flow, massless particles can be added and advected in order to follow the interface's movement. One of the earliest examples is the Marker and Cell method developed by Harlow and Welch [11]. While initially developed to tackle free-surface problems, this method is easy to adapt to multiphase flows: one of the fluids is initially filled with a large number of massless particles, which are then advected, keeping track of where the tagged fluid is. This method is expensive, as it requires a great number of advected particles in order to be precise.
An alternative is the use of massless particles to tag the interface itself (surface tracking), rather than one of the fluids (volume tracking). This reduces the number of particles needed. However, in this case the interface is known through a set of discrete points, with no information on their connectivity. This is particularly problematic for breaks in the interface.
In general, Lagrangian methods need to be coupled with techniques to reconstruct the shape of the interface from the discrete set of particles, which can add considerable algorithmic complexity, especially in three dimensions.
Lagrangian methods can, on the other hand, be used in tandem with an Eulerian approach, with the two methods interacting through reciprocal corrections. → ALE ?

For mesh-free codes that employ a Lagrangian description of the fluid, such as the SPH (Smoothed-particle hydrodynamics) method, the particles on which the computation is based can be constituted of different fluids. The movement of the interface is then implicit in that of the fluid particles.

3.1.2 Volume of Fluid Method

The Volume-of-Fluid method, first proposed in [13], introduces a quantity α called *volume fraction*. This quantity represents, for each cell in the discretized computational domain, how much of it is filled by one particular fluid.
This method has the advantage that, as long as the quantity α is conserved, so is the mass

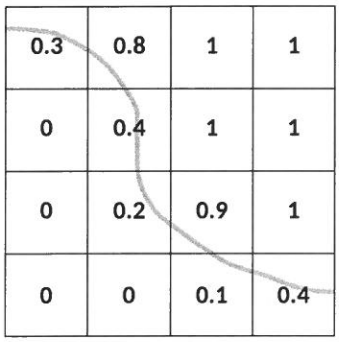


Figure 3.1 – Values of α around an interface.

of both fluids. The main disadvantage is that the actual position and shape of the interface is only known hazily.
In general keeping track of α is not sufficient, and a method for reconstructing the interface based on α's values is needed.
Numerous reconstruction methods have been proposed. An overview can be found in [20]

3.1.3 Level Set Method

The Level Set method, first proposed by Osher and Sethian [19] in 1988, defines a scalar field ϕ, such that its zero level curve (or level surface, in three dimensions) represents the interface between the fluids:

$$\partial\Omega = \{\boldsymbol{x} |\, \phi(\boldsymbol{x}, t) = 0\} \tag{3.1}$$

Advecting the Level Set function as a passive scalar moves the zero level curve, and hence the interface, with the fluid. Unlike the volume fraction, the Level Set function does contain information on the position and shape of the interface. In fact, the normal $\boldsymbol{n}$ to the interface and its curvature κ can be computed as:

$$\boldsymbol{n} = \frac{\nabla\phi}{|\nabla\phi|}, \qquad \kappa = -\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right). \tag{3.2}$$

Usually, but not exclusively, the Level Set function ϕ is defined as the *signed distance* from the interface. This means that the of ϕ conveys which fluid is present, while its value the

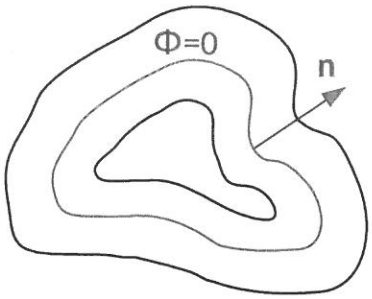


Figure 3.2 – Level curves around an interface.

distance from the border between the two fluids.
One characteristic of a distance field is that:

$$|\nabla\phi| = 1\,. \tag{3.3}$$

One downside of the Level Set is that, unlike the volume fraction, it is a geometrical quantity, not directly linked to the physical "amounts" of the two fluids, which can result in mass conservation problems.

It can be noted that the volume fraction α for a control volume Ω_i can be computed from the Level-Set function ϕ:

$$\alpha = \frac{1}{\Omega_i}\int_{\Omega_i} \mathrm{H}(\phi)\,\mathrm{d}\Omega\,, \qquad \text{where} \qquad \mathrm{H}(\phi) = \begin{cases} 1\,, & \text{if} \quad \phi > 0\,, \\ 0\,, & \text{if} \quad \phi \leq 0\,. \end{cases} \tag{3.4}$$

The inverse relation ($\alpha \to \phi$) is more complicated, but feasible to implement.
One possible strategy to overcome the mass conservation problems of the Level Set while preserving its advantages is to couple it with a volume of fluid method. The two methods can be used to correct each other through (3.4) and its inverse. This strategy, named CLSVOF (for Conservative Level Set Volume of Fluid), is the one explored in the PhD work of Thibaut Menard [17].

3.2 Evolution algorithm for the boundary

The movement of a fluid particle is described by the equation:

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{u}(\boldsymbol{x}, t)\,. \tag{3.5}$$

For Lagrangian methods, this expression can be integrated in time to keep track of massles particles. For Eulerian methods, quantities such as the volume fraction and the Level Set remain constant along the paths of individual particles, hence their material derivative is null:

$$\frac{\mathrm{D}\phi}{\mathrm{D}t} = \frac{\partial\phi}{\partial t} + \boldsymbol{u}\cdot\nabla\phi = 0\,. \tag{3.6}$$

However, for interface capturing methods, simply integrating (3.6) in time may be problematic. Some amount of numerical diffusion is bound to be introduced by the finite arithmetic, causing smearing or distortion of the interface.
In volume-of-fluid methods, in general, specialized advection schemes are developed to ensure that α remains bounded between 0 and 1 and to avoid excessive diffusion.

In Level-Set methods, integration of (3.6) will cause level curves to no longer be parallel, prompting ϕ to lose its meaning of signed distance. To restore it an additional passage is usually applied, in order to make the ϕ field satisfy (3.3) again.
A common strategy is to consider the equation:

$$\frac{\partial d}{\partial \tau} + \operatorname{sign}(\phi)\left(|\nabla d| - 1\right) = 0\,, \tag{3.7}$$

whose steady state is $|\nabla d| = 1$. d is initialized as equal to ϕ, and ϕ is set equal to d's final value. Few (pseudo-)time iterations are required for the level curves to become parallel again, at least close to the interface, where it is needed.
As discussed in [12], the use of (3.7) comes with some caveats, such as the necessity for upwinding ∇d.
The sign function is generally regularized as $\phi/\sqrt{\phi^2 + \Delta x^2}$.

3.3 Scheme for imposing boundary conditions at the interface

Across the interface between two fluids there is a discontinuity in fluid properties, such as density and viscosity. Velocity is generally considered continuous across the interface, at least for viscous flows. Stresses acting on the fluid are not continuous across the interface due to surface tension:

$$[\boldsymbol{\tau n} - p\boldsymbol{n}] = -\sigma\kappa\boldsymbol{n}\,. \tag{3.8}$$

In numerical schemes for incompressible flows, there are two main strategies to apply surface tension: as an impulsive force in the momentum balance equation (2.3a) or as a discontinuity in pressure. A detailed presentation of the jump conditions between two incompressible fluids can be found in [14].
The easiest way to handle discontinuities is to smear them through the use of regularized Heaviside functions, or *delta functions*. Given the signed distance ϕ from the interface:

$$\begin{cases} \rho(\phi) = \rho_1 + (\rho_2 - \rho_1)\,H(\phi)\,, & \text{(3.9a)} \\ \mu(\phi) = \mu_1 + (\mu_2 - \mu_1)\,H(\phi)\,, & \text{(3.9b)} \\ \boldsymbol{f}_{sf}(\phi) = \sigma\kappa(\phi)\,\nabla H\,. & \text{(3.9c)} \end{cases}$$

The function $H(\phi)$ can be, for example, (2.27), however, a variant more commonly used in the multiphase flows literature [7][14] is:

$$H(\phi) = \begin{cases} 0 & \phi < -\varepsilon\,, \\ \dfrac{1}{2} + \dfrac{\phi}{2\varepsilon} + \dfrac{1}{2\pi}\sin\left(\dfrac{\pi\phi}{\varepsilon}\right) & -\varepsilon \le \phi \le \varepsilon\,, \\ 1 & \phi > \varepsilon\,. \end{cases} \tag{3.10}$$

This method while simple, has the downside of creating an artificial interface thickness which impacts the accuracy of the solution. An alternative to the use of delta functions is to modify the discrete operations to account for the jump conditions. One method that employs this strategy is the Ghost Fluid method, first proposed by Fedkiw et al. [9] in 1999. This method consists in extending each fluid across the interface of one (or more) cell(s). The values at these ghost cells are determined by imposing the satisfaction of the jump conditions. This method was adapted to the solution of the Poisson equation on a discontinuous domain in [16], and to a multiphasic incompressible fluid flow in [14].

Chapter 4

Implementation

In the current implementation of the weakly-compressible version of Gridflow, a volume-of-fluid method for simulating multiphase flows exists. This method employs two different mass conservation equations for the two fluids, and derives the value of the volume fraction α by imposing that their partial pressures are at equilibrium [15]. Due to its reference to mass conservation equations involving density, and its use of an equation of state for finding the partial pressures, this method cannot be adapted to the incompressible formulation. algebraic VOF

It was decided instead to implement a scheme based on the Level Set. The reasons are the method's relative simplicity and robustness, the vast literature dedicated to it and its many possible routes for improvement. Another reason is the opportunity to build on the previous implementation of the method by Amélie Bardin [4] in an older version of Gridflow. The main interventions on the code were:

1. Adaptation of the pre-existing Level-Set advection subroutines for the weakly-compressible version to the incompressible version.
2. Addition of the option to reinitialize the Laplacian matrix at every time step, and implementation of variable density inside the Poisson equation for the pressure.
3. Coupling between the variable density and the Level-Set by implementing (4.2) in the code.

4.1 Description of the Method

4.1.1 The Level Set function

In the context of a staggered grid, the Level Set function ϕ is placed at the center of the cell. Advection takes place integrating in time equation

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\boldsymbol{u}\phi) = 0\,, \tag{4.1}$$

which is the conservative form of (3.6). The two formulations are equivalent for incompressible flows, since $\nabla \cdot \boldsymbol{u} = 0$, but (4.1) is more straightforward to implement in a finite volume formulation and is the one used in this work.

It should be noted that in the literature on the Level Set the non-conservative form (3.6) is more commonly seen. In [24], (4.1) has the source term $\phi \nabla \cdot \boldsymbol{u}$ added on the right, in order to be compatible with (3.6) if $\nabla \cdot \boldsymbol{u} = 0$ is not satisfied exactly.

In Amélie Bardin's work [4] both the conservative form (4.1) and the non-conservative form (3.6) were tested, the latter in a finite difference formulation. The conservative form was

kept as it performed better.
The flux reconstruction scheme used for the Level-Set advection equation changed during development, a more detailed discussion is presented further.
Time integration is performed with forward Euler or with the fourth order Runge-Kutta scheme presented in 2.4.

4.1.2 Discontinuity management

In this work viscous effects are ignored, and no surface tension model was implemented, so the only discontinuity is in the density of the two fluids. The density jump is regularized through expression (3.9a), coupled with the Heaviside function (2.27):

$$\rho(\phi) = \rho_1 + \frac{\rho_2 - \rho_1}{2}\left[1 + \tanh\left(\alpha\frac{\phi}{\Delta x}\right)\right]. \tag{4.2}$$

The non-uniform density affects the Poisson equation for pressure (2.8b):

$$\nabla \cdot \left(\frac{1}{\rho}\nabla p\right) = \frac{1}{\Delta t}\nabla \cdot \boldsymbol{u}^*. \tag{4.3}$$

It is easy to show in one dimension that, with centered finite differences, the values of the pressure are needed at the cell faces:

$$\left.\frac{\partial^2 p}{\partial x^2}\right|_i \approx \frac{1}{\Delta x}\left[\frac{1}{\rho_{i+\frac{1}{2}}}\frac{p_{i+1} - p_i}{\Delta x} - \frac{1}{\rho_{i-\frac{1}{2}}}\frac{p_i - p_{i-1}}{\Delta x}\right],$$

however, the Level Set is stored at the cell centre. It is possible to apply (4.2) to the Level-Set at the cell centres, then interpolate the density at the cell faces, or interpolate the Level Set first, and then compute the density. In [7] both strategies are tested, with the former resulting in additional spurious (albeit stable) oscillations. In this work, the latter strategy is employed. In fact, the values of the Level Set are reconstructed at the cell faces when computing the fluxes for equation (4.1). Their values are stored and then fed to (4.2) to compute the density.

4.1.3 Method summary

Assuming integration in time is performed with forward Euler, and the non-incremental form of Chorin's projection method is used, the scheme can be summarized as:

1. Update the Level Set:

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = -\nabla \cdot (\boldsymbol{u}^n \phi^n), \tag{4.4a}$$

2. Compute the intermediate velocity field:

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -\nabla \cdot (\boldsymbol{u}^n \otimes \boldsymbol{u}^n) + \boldsymbol{g}, \tag{4.4b}$$

3. Compute the density at the cell faces:

$$\rho(\phi^{n+1}) = \rho_1 + \frac{\rho_2 - \rho_1}{2}\left[1 + \tanh\left(\alpha\frac{\phi^{n+1}}{\Delta x}\right)\right], \tag{4.4c}$$

4. Solve the variable coefficient Poisson equation for the pressure:

$$\nabla \cdot \left(\frac{1}{\rho\left(\phi^{n+1}\right)} \nabla p^{n+1} \right) = \frac{1}{\Delta t} \nabla \cdot \boldsymbol{u}^{*} , \tag{4.4d}$$

5. Apply the pressure gradient to correct the vector field:

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^{*} - \frac{\Delta t}{\rho\left(\phi^{n+1}\right)} \nabla p^{n+1} . \tag{4.4e}$$

The rationale behind using the density field at t^{n+1} in (4.4d) is given in [14]: the solution of this equation is used to correct the updated velocity field $\boldsymbol{u}^{n+1}$.

4.2 Level Set advection

Gridflow is a code under active development, subject to frequent changes and updates. The developments presented here started in 2019 on a version of the code having some limitations to the flux reconstruction subroutines, that were lifted in a later (2020) release. For this reason, the advection scheme for the Level Set in the incompressible formulation had to be implemented twice: initially in the 2019 version of the code, and later in the 2020 version, to take full advantage of its improvements.
As mentioned in section 2.8, the use of a staggered grid in the incompressible formulation meant that the WENO scheme had to be substantially modified. In fact, the weakly-compressible and the incompressible versions of the code use different subroutines for the WENO reconstructions, with little inter-compatibility.
In the 2019 version of the code, the WENO scheme for cell-centred variables in use in the weakly-compressible formulation was not available to the incompressible formulation, where only face-centred variables needed flux reconstruction. The MUSCL scheme, at this point, was not compatible either with the incompressible formulation. The only option for the advection of cell-centred quantities in the incompressible version, already in use for the eddy viscosity of the Spalart-Allmaras turbulence model, was the first order scheme described in 2.8.1. This scheme was relatively easy to adapt to the advection of ϕ. The first results were obtained on the 2019 version of the code, with this scheme used for advecting ϕ.
In the more recent version of the code, the MUSCL scheme and WENO subroutines for cell-centred variables were made available to the incompressible formulation. This allowed their use for advecting the Level Set.
For pure transport tests the first order and the MUSCL schemes proved too dissipative, consistently with what was observed in [4], while the WENO scheme preserved satisfactorily the shape of the interface.
For multiphasic tests, while still too dissipative, the first order and MUSCL schemes demonstrated an advantage over WENO in terms of stability.

4.3 Variable coefficient Poisson Equation

4.3.1 PETSc Projection Scheme

In the single-fluid incompressible version of Gridflow, all differential operators used to compute the pressure and its gradient are rebuilt every time there is a change in the grid. All sparse matrices are re-initialized, since their non-zero pattern depends on how the cells are distributed on the domain. The first operators to be built are the divergence matrices, one per direction.:

$$\boldsymbol{D}_x ; \qquad \boldsymbol{D}_y ; \qquad \boldsymbol{D}_z .$$

Then, the gradient matrices are obtained as their transpose, with each row divided by the cell volume:

$$\boldsymbol{G}_x = \text{diag}\left\{\frac{1}{V_i}\right\}_x \boldsymbol{D}_x^T\,; \qquad \boldsymbol{G}_y = \text{diag}\left\{\frac{1}{V_i}\right\}_y \boldsymbol{D}_y^T\,; \qquad \boldsymbol{G}_z = \text{diag}\left\{\frac{1}{V_i}\right\}_z \boldsymbol{D}_z^T\,.$$

After that, the divergence matrices are modified, in the case of a non-uniform grid, to include interpolations.
Considering the set Ω_c of quantities defined at the cell centres, and the sets Ω_x, Ω_y, Ω_z of quantities defined at the cell faces along the three Cartesian directions, these operators map one into the others:

$$\begin{aligned} \boldsymbol{D}_x &: \Omega_x \longrightarrow \Omega_c & \qquad \boldsymbol{G}_x &: \Omega_c \longrightarrow \Omega_x\,, \\ \boldsymbol{D}_y &: \Omega_y \longrightarrow \Omega_c & \qquad \boldsymbol{G}_y &: \Omega_c \longrightarrow \Omega_y\,, \\ \boldsymbol{D}_z &: \Omega_z \longrightarrow \Omega_c & \qquad \boldsymbol{G}_z &: \Omega_c \longrightarrow \Omega_z\,, \end{aligned}$$

The Laplacian matrix is obtained as the sum of their products:

$$\boldsymbol{L} = \boldsymbol{D}_x\boldsymbol{G}_x + \boldsymbol{D}_y\boldsymbol{G}_y + \boldsymbol{D}_z\boldsymbol{G}_z\,, \tag{4.5}$$

which naturally places Neumann boundary conditions in $\boldsymbol{L}$. In order to avoid an underdetermined linear problem, the Laplacian matrix is modified in order to impose a null pressure at one point in the domain.
It can be noted that $\boldsymbol{L} : \Omega_c \longrightarrow \Omega_c$.
The last passage is the initialization of the GMRES solver:

```
call KSPSetOperators(operator, solutionMatrix, preconditionerIfAny, errorCode)
```

KSP standing for Krylov Subspace, the family of methods that includes the GMRES algorithm. This passage must be performed every time the solution matrix $\boldsymbol{L}$ is modified in any way, even if the non-zero structure is unaltered. A simple change in value of one element will require the reinitialization step. However, if the matrix is unchanged, the same PETSc `operator` can be used with different right-hand sides.
The projection scheme is as follows: first, the components of the intermediate velocity $\boldsymbol{u}^*$ are converted from Gridflow's native data structures to the PETSc vectors $\boldsymbol{u} \in \Omega_x$, $\boldsymbol{v} \in \Omega_y$ and $\boldsymbol{w} \in \Omega_z$. Then, the right-hand side of the pressure equation is computed as:

$$\boldsymbol{b} = \frac{\rho}{\Delta t}\left(\boldsymbol{D}_x\boldsymbol{u} + \boldsymbol{D}_y\boldsymbol{v} + \boldsymbol{D}_z\boldsymbol{w}\right)\,, \tag{4.6}$$

where $\boldsymbol{b} \in \Omega_c$. The pressure field is obtained solving the linear system:

$$\boldsymbol{p} = \boldsymbol{L}^{-1}\boldsymbol{b}\,. \tag{4.7}$$

The correction step (2.8c) is applied while data is still in PETSc vector format:

$$\boldsymbol{u} = \boldsymbol{u} - \frac{\Delta t}{\rho}\boldsymbol{G}_x\boldsymbol{p}\,; \qquad \boldsymbol{v} = \boldsymbol{v} - \frac{\Delta t}{\rho}\boldsymbol{G}_y\boldsymbol{p}\,; \qquad \boldsymbol{w} = \boldsymbol{w} - \frac{\Delta t}{\rho}\boldsymbol{G}_z\boldsymbol{p}\,.$$

After this passage, the components of the velocity, the pressure and, if requested, the components of the pressure gradient are converted back into Gridflow's data structures.

4.3.2 Reinitialization of the Solver

If the density changes at every time step, so do the elements of $\boldsymbol{L}$. However, if the grid is not touched, the non-zero structure of all matrices remains the same.
In order to handle these changes more efficiently, the initialization procedure was divided in two subroutines: one, `createIncompressibleObjects`, handling the creation "from scratch" of all the operators; the other, `updateIncompressibleObjects` performing multiplication (4.5) to obtain $\boldsymbol{L}$, correcting it for boundary conditions and calling `KSPSetOperators`.
`createIncompressibleObjects` is used at the beginning and at every change in the grid, `updateIncompressibleObjects` is used, if necessary, at every time iteration

4.3.3 Variable Density

Once the density at the cell faces is known, it can be used to fill the diagonal matrices:

$$\mathrm{diag}\left\{\frac{1}{\rho}\right\}_x : \Omega_x \longrightarrow \Omega_x\,; \qquad \mathrm{diag}\left\{\frac{1}{\rho}\right\}_y : \Omega_y \longrightarrow \Omega_y\,; \qquad \mathrm{diag}\left\{\frac{1}{\rho}\right\}_z : \Omega_z \longrightarrow \Omega_z\,.$$

The initialization of these matrices is added to both `createIncompressibleObjects` and `updateIncompressibleObjects`. (4.5) is modified as:

$$\boldsymbol{L} = \boldsymbol{D}_x \mathrm{diag}\left\{\frac{1}{\rho}\right\}_x \boldsymbol{G}_x + \boldsymbol{D}_y \mathrm{diag}\left\{\frac{1}{\rho}\right\}_y \boldsymbol{G}_y + \boldsymbol{D}_z \mathrm{diag}\left\{\frac{1}{\rho}\right\}_z \boldsymbol{G}_z\,, \tag{4.8}$$

ρ is removed from the right-hand side (4.6), and the correction step is reformulated as:

$$\boldsymbol{u} = \boldsymbol{u} - \Delta t \mathrm{diag}\left\{\frac{1}{\rho}\right\}_x \boldsymbol{G}_x \boldsymbol{p}\,; \quad \boldsymbol{v} = \boldsymbol{v} - \Delta t \mathrm{diag}\left\{\frac{1}{\rho}\right\}_y \boldsymbol{G}_y \boldsymbol{p}\,; \quad \boldsymbol{w} = \boldsymbol{w} - \Delta t \mathrm{diag}\left\{\frac{1}{\rho}\right\}_z \boldsymbol{G}_z \boldsymbol{p}\,.$$

4.4 Multiphase Scheme

The Level-Set advection determines how the flow acts on the distribution of the fluids, while the variable density Poisson equation determines how the distribution of the two fluids acts on the flow. The coupling between the two is obtained through equation (4.2).
In Gridflow, step (4.4a), the advection of cell-centred quantities, is performed first. By this point the values of ϕ have been stored at the cell faces. Right after, the advection of face-centred variables, (4.4b), is performed.
(4.2), or (4.4c) is applied to ϕ at the cell faces, and ρ obtained, right before calling `updateIncompressibleObjects`. Immediately after, the Pressure equation (4.4d) is solved and the correction step (4.4e) is applied.

4.5 Other Modifications

4.5.1 Initial Time Step

Grifdlow employs an adaptive time step. The value of Δt is updated at every time iteration based on the conditions:

$$\begin{cases} \Delta t_h = \mathrm{CFL}_h \dfrac{\Delta x}{\max_\Omega (|u_i|)}\,, & (4.9a) \\ \Delta t_v = \mathrm{CFL}_v \dfrac{\Delta x^2}{\nu}\,, & (4.9b) \\ \Delta t = \min(\Delta t_h, \Delta t_v)\,, & (4.9c) \end{cases}$$

where CFL_h and CFL_v are user specified values which should be in the [0, 1] interval. By default, they are set to 0.7 and 0.15 respectively. If the flow is inviscid, the "viscous" time step Δt_v is not considered and the "hyperbolic" one Δt_h is chosen by default. This can be problematic if the velocity is null everywhere in the domain. If this is the case, Gridflow corrects the maximum velocity to $2.22 \cdot 10^{-16}$ but this, while preventing an error, does produce a time step excessively large. actually this number is the smallest that can be represented in the memory of the computer (epsilon(1._rp))

It should be noted that this problem does not appear in the weakly compressible version of the code, since in it the reference velocity is defined as $\max_\Omega (|u_i + c_i|)$ and the speed of sound c_i is always non-null.

In order to be able to test situations in which the velocity is initially null and viscosity is absent, an upper cap to the time step was added. This simply modifies (4.9c) as:

$$\Delta t = \min\left(\Delta t_h,\, \Delta t_v,\, \Delta t_{max}\right) . \tag{4.9d}$$

4.5.2 Null Pressure

In section 4.3.1 it was mentioned that, to close the linear system resulting from the discretization of (2.8b), it is necessary to impose the value of the pressure at some point in the domain. By default, Gridflow does so at the bottom-left.

In a multiphase flow subject to gravity, the denser fluid will tend to settle at the bottom. If the interface between the two fluids, above the point where the reference pressure is set to zero, changes in height, this will cause the whole pressure field to oscillate wildly. It should be noted that this effect is purely non-physical, an artefact of the arbitrariness of the reference pressure, and does not impact the fluid flow, which is influenced by the pressure only through its gradient. However, this effect does make the data less readable. For this reason, the option to specify where to impose the reference pressure was added. In all multiphase simulations subject to gravity presented in the next chapter, the pressure is imposed null at the top of the domain.

Chapter 5

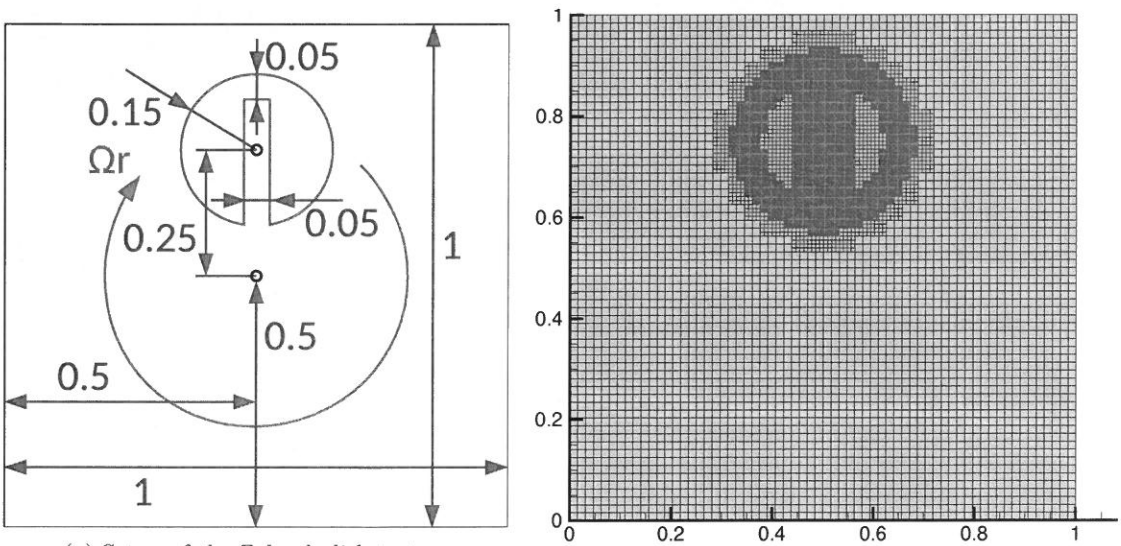
Results

All tests presented here are two-dimensional. Unless it's not explicitly stated otherwise, they were run on the most recent (2020) version of the code.

5.1 Level Set Advection

5.1.1 Zalesak Disk

Proposed in [23], the Zalesak disk is often used in the literature to test the accuracy of an advection scheme for fluid interfaces, such as the Level Set method.
A two dimensional square domain is filled by two fluids, one occupying a closed region shaped like a slit disk. The velocity field is imposed, consisting of a linear vortex advecting the disk, initially at the top, around the domain.
The purpose of the simulation is to see how distorted is the shape of the disk after a full



(a) Setup of the Zalesak disk test case.

(b) Initial mesh used in Gridflow.

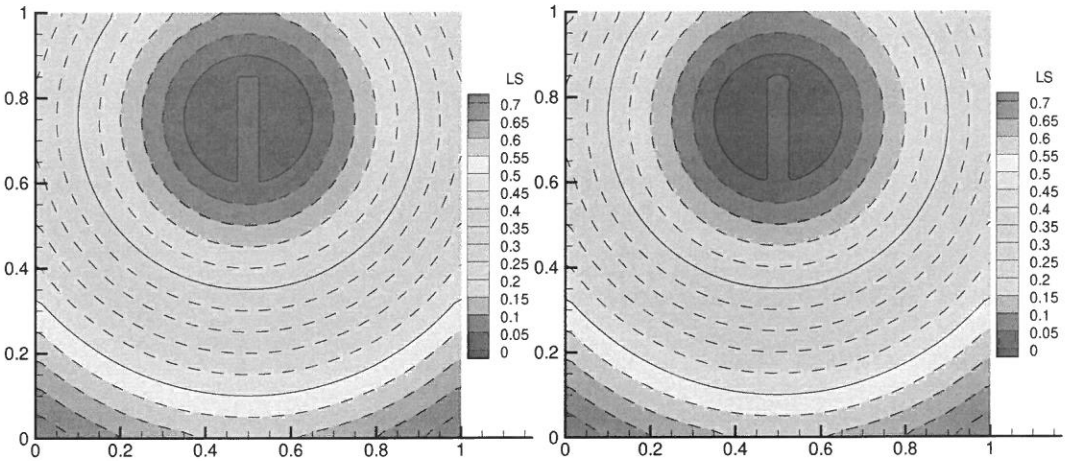
Figure 5.1

rotation. The amount of distortion is indicative of the quality of the advection scheme.

The velocity field is defined as:

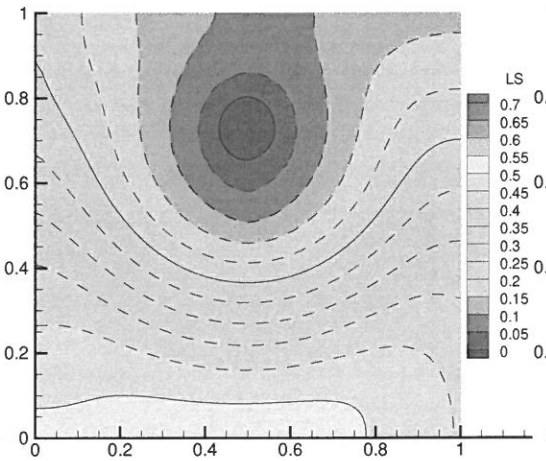
$$\begin{cases} u = -\Omega\,(y - 0.5)\ , \\ v = \quad \Omega\,(x - 0.5)\ , \end{cases} \tag{5.1}$$

with angular velocity $\Omega = 2\pi$. A complete rotation should take one time unit of simulation. In Gridflow, the Level Set function is initialized as the signed distance, negative inside the

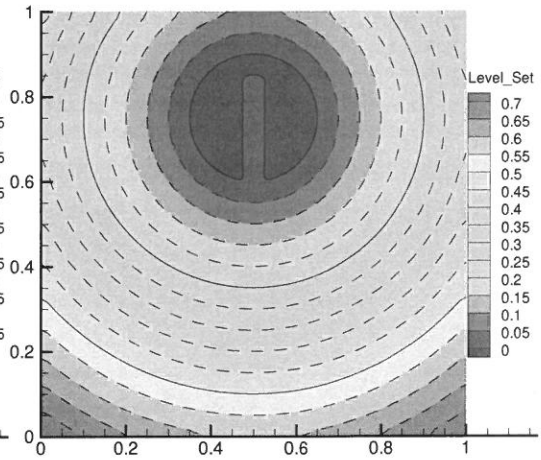


(a) Level Set function at the beginning of the simulation.

(b) Level Set function at $t = 1$ using the weakly-compressible version of the code (WENO).



(c) Level Set function at $t = 1$ using the 2019 incompressible version of the code (1st order).



(d) Level Set function at $t = 1$ using the 2020 incompressible version of the code (WENO).

Figure 5.2

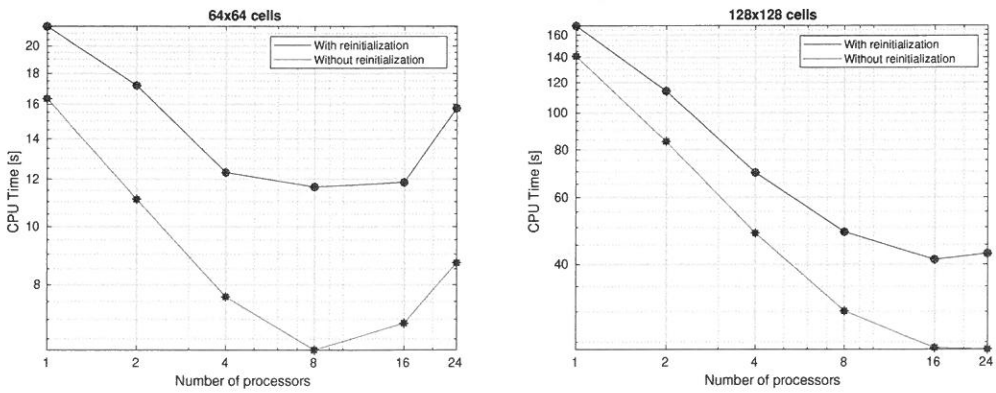
disk. The time integration of the momentum equation is bypassed so that the velocity field remains equal to (5.1) throughout the computation.
A dynamically adaptive grid is used, with the highest level of refinement imposed where the absolute value of the level set is close to the minimum grid size, as shown in figure 5.1b. The base grid is 64×64 cells ($\Delta x = 1.5625 \cdot 10^{-2}$) with the finer cells two levels of refinement deeper ($\Delta x = 3.90625 \cdot 10^{-3}$). During the simulation, the "patch" of finer cells follows the disk around the domain.

Figure 5.2a shows the Level Set field at the beginning of the simulation. The level curves appear very regular despite the coarseness of the grid far from the disk. This is a product of the post-processing software. However, it can be seen that the sharp angles of the slit are well resolved by the finer cells around the disk.
Figures 5.2b 5.2c ans 5.2d show ϕ at $t = 1$, after a complete rotation. They are obtained using, respectively, WENO reconstruction in the weakly-compressible version, a first order accurate reconstruction in the 2019 incompressible version and WENO again, in the 2020 incompressible version. It can be seen that WENO, both in the weakly-compressible and in incompressible formulation preserves well the shape of the disk, albeit blunting the corners of the slit. The first order scheme is instead too diffusive, as the disk almost disappears.

5.2 Poisson Equation

5.2.1 Solver Reinitialization Tests

The multiphase method implemented requires a reinitialization of the PETSc solver at each time iteration (as described in section 4.3.2).
This operation is bound to slow down the execution of the code to a certain extent. In order



(a) (b)

Figure 5.3 – CPU Time for the Taylor-Green vortex test case, with and without reinitialization of the PETSc solver at every time step.

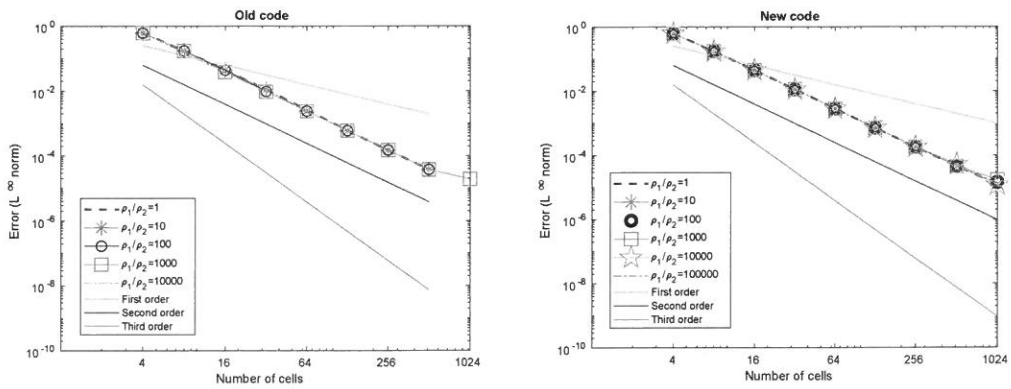
to quantify this additional cost, the possibility of forcing the reinitialization of the PETSc solver at every time iteration was added to the base incompressible scheme. The idea is to run the same simulation with and without this forced realization, then compare the CPU times. The setup chosen for this test is the two-dimensional Taylor-Green vortex, which is initialized as:

$$\begin{cases} u(x,y) = U_0 \sin\left(2\pi\frac{x}{L}\right)\cos\left(2\pi\frac{y}{L}\right), \\ v(x,y) = U_0 \cos\left(2\pi\frac{x}{L}\right)\sin\left(2\pi\frac{y}{L}\right), \\ p(x,y) = \frac{\rho U_0^2}{4}\left[\cos\left(4\pi\frac{x}{L}\right) + \cos\left(4\pi\frac{y}{L}\right)\right], \end{cases} \tag{5.2}$$

and advected with no explicit viscosity. The test was run on École Centrale's computer cluster *Liger*. A single node was used, constituted by two 12-core Intel Xeon E5-2680v3.
Figure 5.3 shows the results. Each point in the graph represents the average over ten simulations. Two grids were used, one made up of 64×64 cells, the other of 128×128 cells.

Different numbers of processors were used: 1, 2, 4, 8, 16 or 24.
It can be noted that, not only the reinitialization slows down execution by a tangible amount, but that the divide seems to increase with the number of processors. On the other hand, it should be considered that a test up to 24 processors is not particularly indicative of the scalability of the procedure.
Also, both the reinitialized and the non-reinitialized tests start to slow down for an increasing number of processors, hinting that perhaps the grids considered are too small for scalability studies.
In the end this test proved inconclusive, and the reinitialization of the solver is in any case unavoidable for a varying density field.

5.2.2 Poisson Equation Convergence Tests



(a) 2019 version of the code.

(b) 2020 version of the code.

Figure 5.4 – L^∞ error norm in the solution to the Poisson equation.

In order to test the implementation of (4.8), a convergence study was devised. For this test, the unit square domain $[0,\,1] \times [0,\,1]$ is considered. The upper and lower halves of the domain are filled with different fluids, with the lower one being denser. The velocity field is initialized as

$$\begin{cases} u = \dfrac{-2\Delta t \pi \sin\left(2\pi x\right)\cos\left(2\pi y\right)}{\rho\left(x,\,y\right)}\,, \\ v = \dfrac{-2\Delta t \pi \cos\left(2\pi x\right)\sin\left(2\pi y\right)}{\rho\left(x,\,y\right)}\,. \end{cases}$$

This is analogous to the Taylor-Green vortex, except for the variable density. For this test, the advection and source terms of the momentum equation are bypassed and only one time step executed. This means that, at the end of this single time step, the solution to the pressure Poisson equation (4.4d) is:

$$p = \cos\left(2\pi x\right)\cos\left(2\pi y\right)\,.$$

This expression can be used to quantify the error in the solution. Figure 5.4 shows the error in the L^∞ norm for various mesh sizes and different density ratios between the lower and upper portions of the domain.
It can be seen that the operators used for solving equation (4.4d) are second order accurate, even with a variable density.

5.3 Multiphase Flow Tests

5.3.1 Hydrostatic Test Case

The setup of this test case is similar to the one descibed previously (5.2.2): a square domain is considered, the top half is filled by one fluid, the bottom by a second fluid 1000 times denser. The two fluids are subject to the effects of gravity, in the form of a constant source term added to the momentum equation, pulling downward. The boundary conditions are free-slip, due to the absence of viscosity). The velocity is initially null.
Ideally no fluid movement should arise during the computation. This is found to be the case

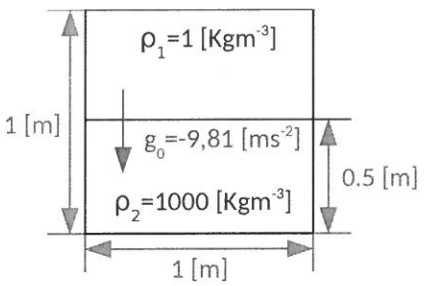


Figure 5.5 – Initial setup of the hydrostatic test case.

if the difference in density between the two fluids is removed, or if the density update (4.4c) is turned off. Applying the full multiphase scheme (4.4), oscillations start to originate at the interface between the two fluids.
Figure 5.6 was obtained on a 128×128 uniform static grid, using WENO for flux recon-

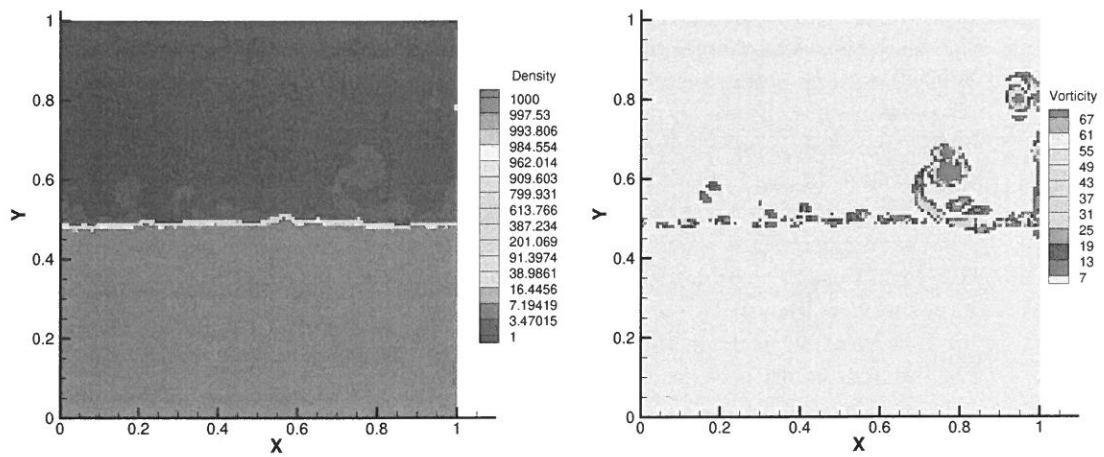
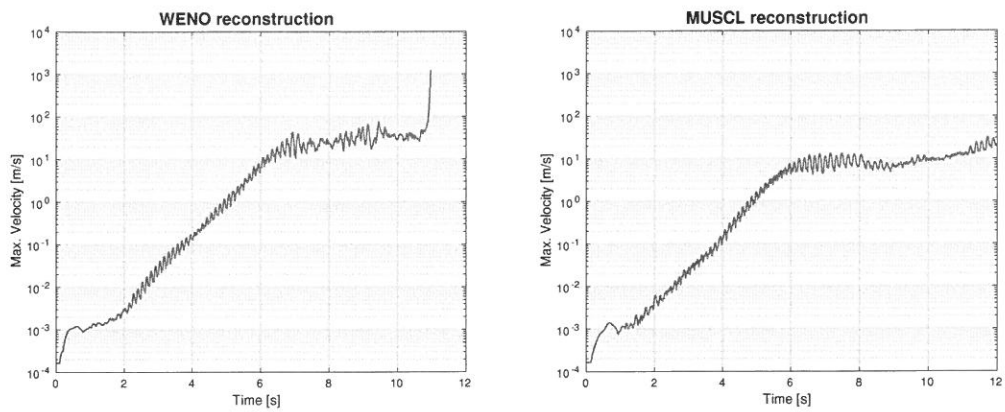


Figure 5.6 – The density and the vorticity field at $t = 6.83[s]$.

struction and forward Euler for time integration. It shows, after 6.83 seconds of simulation time, how the oscillations ripple the interface between the two fluids, shedding vortices in the lighter one.
Figure 5.7 shows how the oscillations evolve in time by plotting the magnitude of the maximum velocity in semi-logarithmic scale. This data was obtained on a 32×32 grid. Time

integration is, again, forward Euler. The graph on the left is obtained using a WENO reconstruction for both the Level Set and the velocity, while graph on the right using WENO for the velocity and MUSCL for the Level-Set. In both cases, at the beginnig the oscillations grow exponentially, then, their amplitude "stabilizes". The WENO -WENO simulation, at $t \approx 11\,[s]$, starts growing again and crashes, while the MUSCL - WENO one is still running at $t = 12\,[s]$.
Many factors influence the growth rate of the oscillations and their final amplitude. Integra-



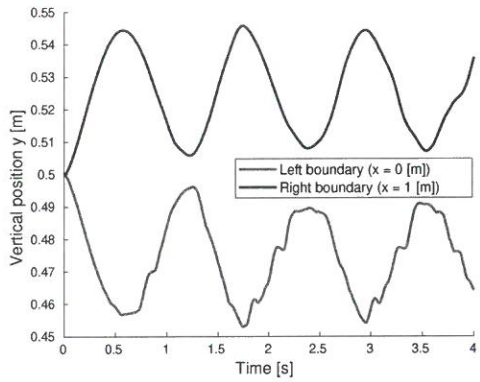
(a) 2019 version of the code.

(b) 2020 version of the code.

Figure 5.7 – L^∞ error norm in the solution to the Poisson equation.

tion with the fourth order Runge-Kutta scheme, as opposed to forward Euler, reduces their amplitude considerably, and so does decreasing the tolerance of the GMRES solver, which is set by default to 10^{-5} by PETSc. This hints that the main source of these oscillations may be the imperfect solution to the pressure Poisson equation. In general, higher order spatial discretization schemes affect stability negatively, with the worst results obtained using WENO with no smoothness indicators.
The maximum time step presented in section 4.5.1 is, in general, valid only at the beginning of the simulation, since at some point the oscillations reach enough magnitude for Δt_h to be higher than Δt_{max}. This initial phase, however, affects the whole simulation: lower values of Δt_{max} slow down the onset of instability.
Another aspect whose effect on stability was investigated is the span of the Heaviside function (2.27) used to regularize the density field, determined through the parameter α. While imposing a sudden jump (very high α) in density proved, as expected, detrimental to stability, a correlation between the span and the onset of oscillations could not be found.
For the hydrostatic case described above, there is one way to avoid oscillations entirely, which is to use the incremental version of the projection scheme. Unfortunately, this is only due to the fact that, if the initial pressure field is chosen accurately enough to balance the effects of gravity, the right-hand side of the pressure equation is always null. This is useless as it does not address the source of the oscillations, but simply bypasses the numerical scheme altogether.
The incremental formulation did not provide advantages in terms of stability in more dynamic computations, such as the dambreak.
By choosing carefully the factors listed above, one can set up a simulation in which oscillations grow slowly enough and are sufficiently low in amplitude, at least for a static or slowly moving interface.

5.3.2 Linear Sloshing



may you add the analytical solution that is expected for this test case?

Figure 5.8 – The temporal evolution of the vertical position of the interface at the left and right extremes of the domain.

The linear sloshing test case has a setup almost identical to the hydrostatic one, except for the addition of a small horizontal acceleration. This means that the equilibrium position for the interface between the two fluids is slightly inclined, compared to the initial horizontal one. During the simulation, the extremes of the interface start to oscillate up and down. Ideally, in the absence of viscosity, this oscillation would continue indefinitely.
The simulation is run on a 256×256 cells static grid, with $\Delta t_{max} = 2^{-7}\,[s]$, using WENO

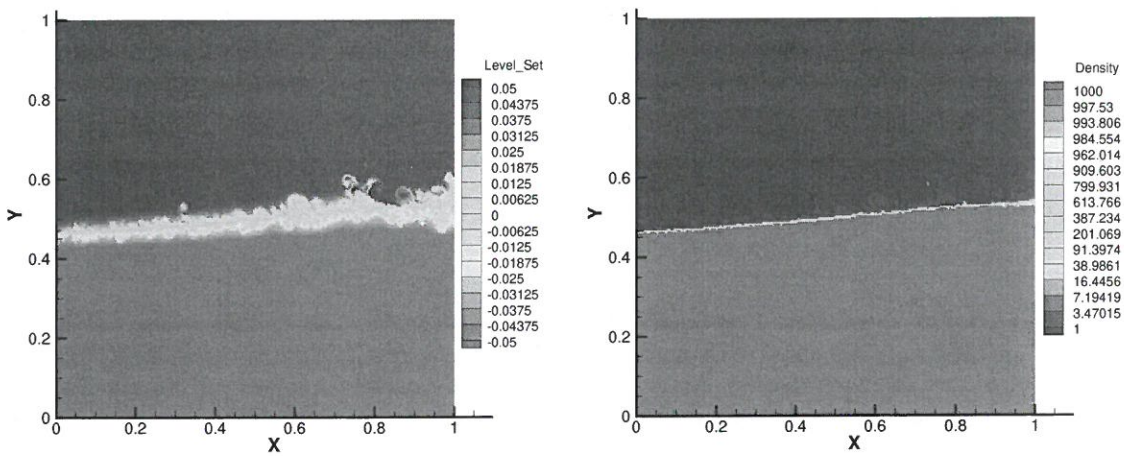


Figure 5.9 – Level Set and density fields at $t = 4\,[s]$.

for advecting both the velocity and the Level Set, and fourth order Runge-Kutta for time integration. Four seconds of simulation are computed.
While no crash occurs during this time, and the behaviour of the interface is close to the expected one, it is clear from figure 5.9 that the oscillations seen in the hydrostatic test have started developing.

5.3.3 Dambreak (2019)

The dambreak test consists in having a pocket of denser fluid on one side of the domain which, when the simulation starts, floods the other side of the domain.
The domain is now rectangular ([0, 1] × [0, 0.5]).

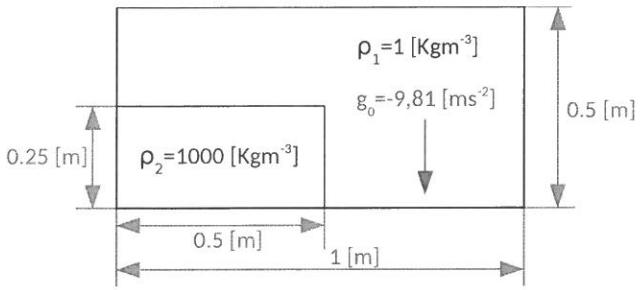


Figure 5.10 – Initial setup of the Dambreak test.

This test was first run on the earlier (2019) implementation of the scheme, with the velocity

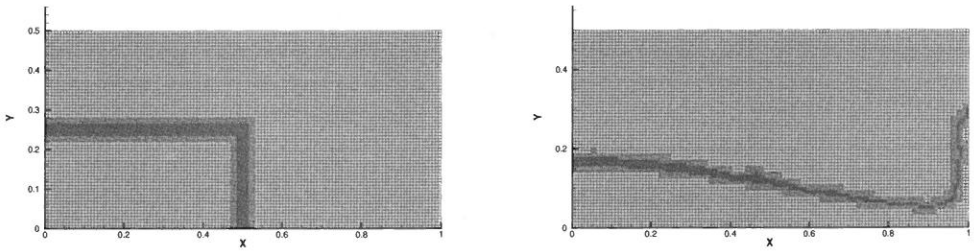


Figure 5.11 – The dynamically adaptive grid at $t = 0\,[s]$ (left) and at $t = 0.5\,[s]$ (right) for the 2019 test.

fluxes reconstruted using WENO and the level Set fluxes using the first order scheme. For this first simulation, dynamic adaptive refinement is used, with the same Level-Set-based refinement criterion used for the Zalesak disk. The base grid is made of 128×64 cells ($\Delta x = 7.8125 \cdot 10^{-3}\,[m]$), the finer cells are two levels of refinement deeper ($\Delta x = 1.953125 \cdot 10^{-3}\,[m]$). $\Delta t_{max} = 0.01[s]$ and the simulation is run for 25 seconds.
No crash is encountered, and the oscillations observed in the hydrostatic and linear sloshing tests are not evident. On the downside, the denser fluid loses mass to the lighter one and disappears completely after $t \approx 18\,[s]$. This is due to the excessive diffusion of the advection scheme used for the Level Set. It can be observed in figure 5.13 that the Level Set field is almost entirely uniform by the end of the simulation.
Further tests revealed that this setup may suffer from stability problems as well. While for carefully chosen parameters the computation runs smoothly until the chosen 25 seconds limit, in other situations the velocity starts to grow exponentially, and the simulation is run to a halt due to (4.9a).
It was, unfortunately, not possible to identify a precise trend in the choice of parameters more conductive to stability. As observed in section 5.3.1 for the hydrostatic test, a lower Δt_{max} seems to aid stability, however, lowering it below a certain point is actually detrimental.
The effects of the span of the Heaviside function were investigated, with spans as wide as 9 cells considered. Again, a correlation with the stability of the simulation was not found. All

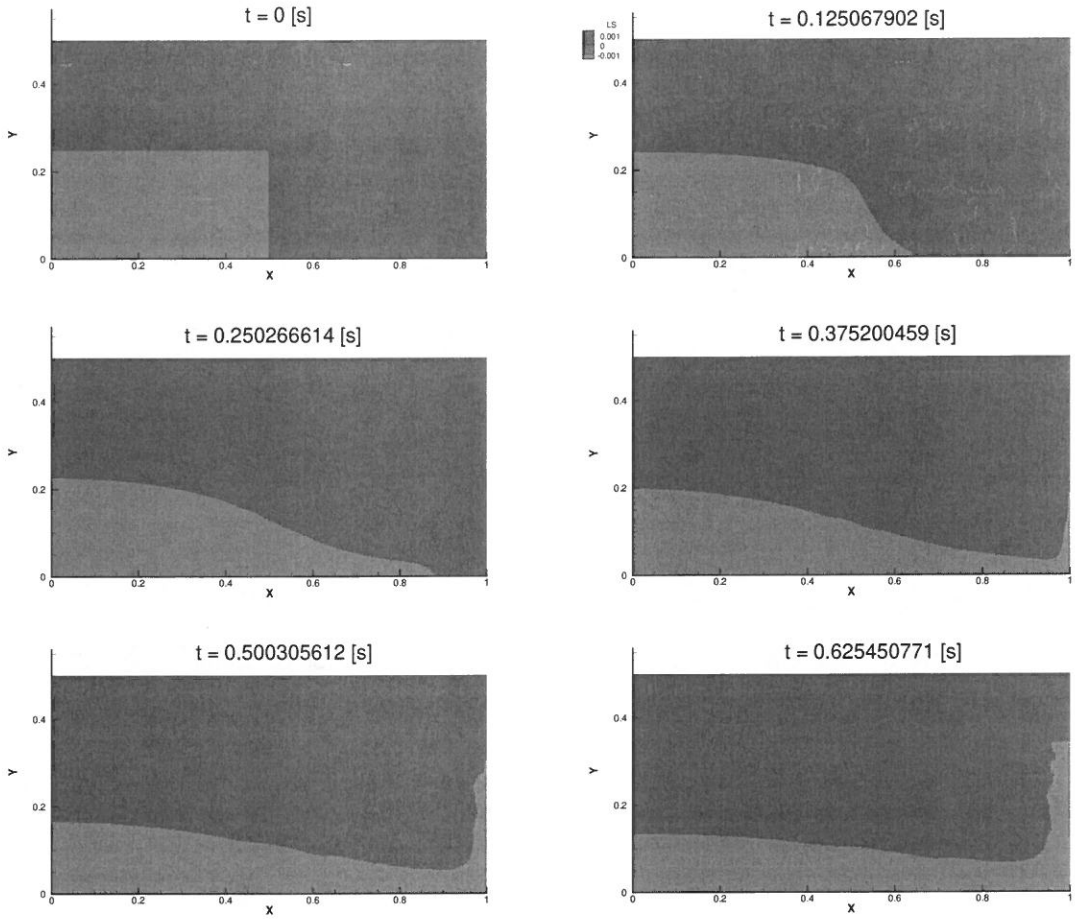


Figure 5.12 – The two fluids at various times in the dambreak simulation run on the 2019 code.

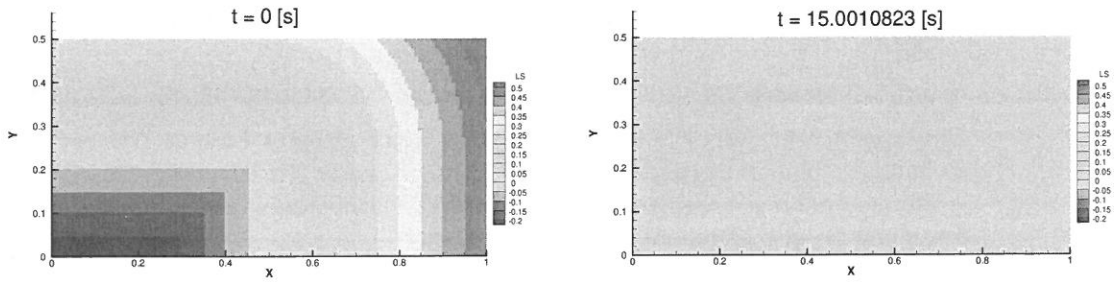


Figure 5.13 – The Level Set field at two different times in the dambreak simulation run on the 2019 code. In the second one the denser fluid has almost entirely disappeared due to diffusion.

results reported here were obtained setting $\alpha = 3$, which is the default value for (2.27) when applied to the immersed boundary. This corresponds to a span of about 2-3 cells.

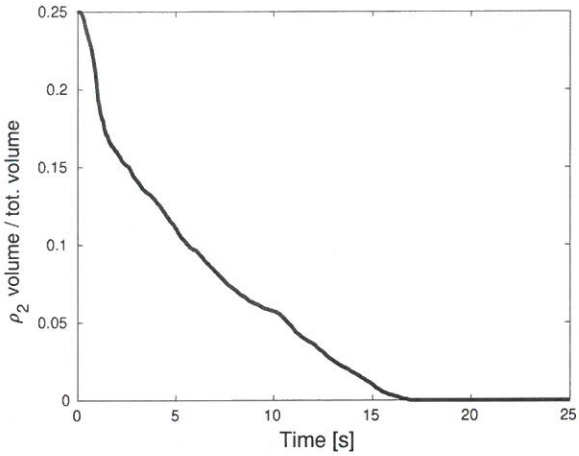


Figure 5.14 – Fraction of the domain volume occupied by the denser fluid as a function of time. Simulation run on the 2019 version of the code.

5.3.4 Dambreak (2020)

After the upgrade to the 2020 version of the code, both WENO and MUSCL reconstruction schemes were available for advection. It was decided to test four possible combinations of reconstruction schemes:

- Velocity reconstructed with WENO, Level-Set reconstructed with WENO.
- Velocity reconstructed with WENO, Level-Set reconstructed with MUSCL.
- Velocity reconstructed with MUSCL, Level-Set reconstructed with WENO.
- Velocity reconstructed with MUSCL, Level-Set reconstructed with MUSCL.

For these tests, the dynamically refined grid was abandoned in favour of a static 256×128 ($\Delta x = 3.90625 \cdot 10^{-3}\,[m]$) grid, and the time limit was set to 10 rather than 25 seconds. The setup of the test shown in figure 5.10 is otherwise unaltered. The evolution in time of the desity field is shown in figures 5.17, 5.18, 5.19, 5.20, 5.21, 5.22, 5.23 and 5.24.

density

Velocity reconstructed with WENO, Level-Set reconstructed with WENO

This combination was the least successful. Every attempt crashed between 0.3 and 0.4 seconds of simulation time, before the front of denser fluid reaches the right boundary. Attempts were made with and without the WENO smoothness indicators. When active, the crash is delayed by no more than 0.1 seconds.

Velocity reconstructed with WENO, Level-Set reconstructed with MUSCL

This setup behaved similarly to the 2019 code, with the simulation not crashing and the denser fluid progressively disappearing due to the diffusion of the advection scheme used for the Level Set. It can be noted that no denser fluid remains at the 10-second mark, hence diffusion is actually faster. This could be explained by the fact that, while the base grid used for the 2019 test was coarser than this one, the cells around the interface were finer.

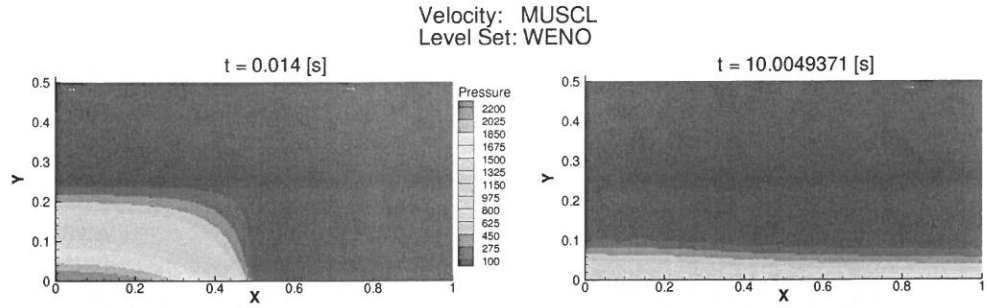


Figure 5.15 – Pressure field at the beginning and at the end of the simulation using MUSCL to advect the velocity and WENO to advect the Level Set.

Velocity reconstructed with MUSCL, Level-Set reconstructed with WENO

This combination ran without problems up to the 10-second mark, with seemingly no loss of denser fluid. This result was obtained with the WENO smoothness indicators turned on. An earlier attempt with them deactivated diverged after a few seconds of simulation time. While the previous two results show how the higher diffusivity of the MUSCL scheme helps keep in check irregularities in the Level Set field, this test shows that this is not the only factor contributing to stability. The velocities involved when using MUSCL for advecting them are lower, and this is sufficient to keep the instabilities in check, at least for up to 10 seconds.

Velocity reconstructed with MUSCL, Level-Set reconstructed with MUSCL

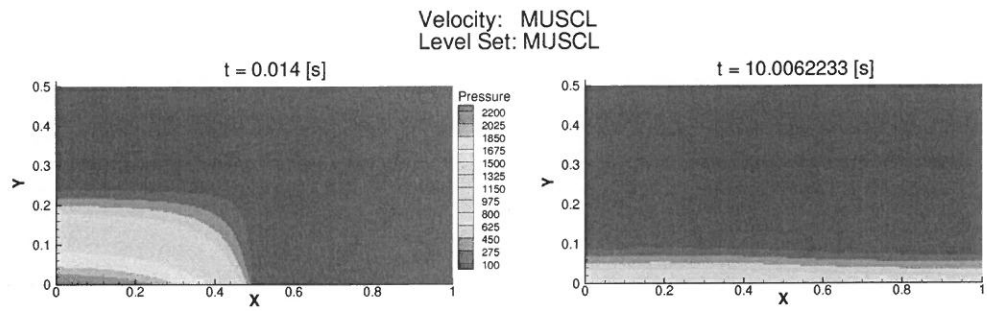


Figure 5.16 – Pressure field at the beginning and at the end of the simulation using MUSCL to advect both the velocity and the Level Set.

This setup, again, ran smoothly for the entire 10 seconds of simulation. Interestingly, despite the higher diffusivity of the MUSCL scheme for advecting the Level-Set, no loss of denser fluid is evident. This is likely due to the lower velocities involved. The behaviour is surprisingly similar to the test using MUSCL for the velocity and WENO for the Level Set, however, the shape of the interface between the two fluids appears more regular.

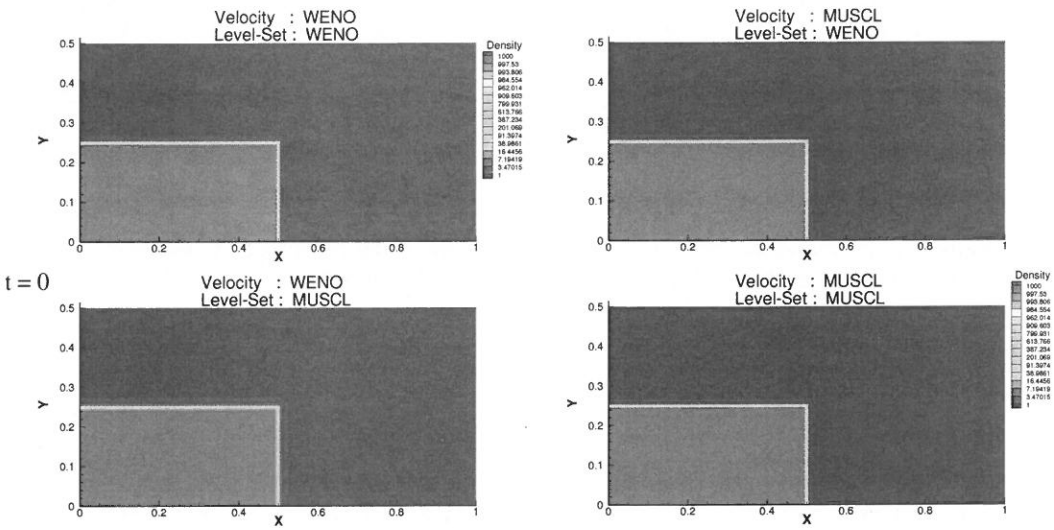


Figure 5.17 – Dambreak tests in the 2020 version fo the code: $t = 0\,[s]$.

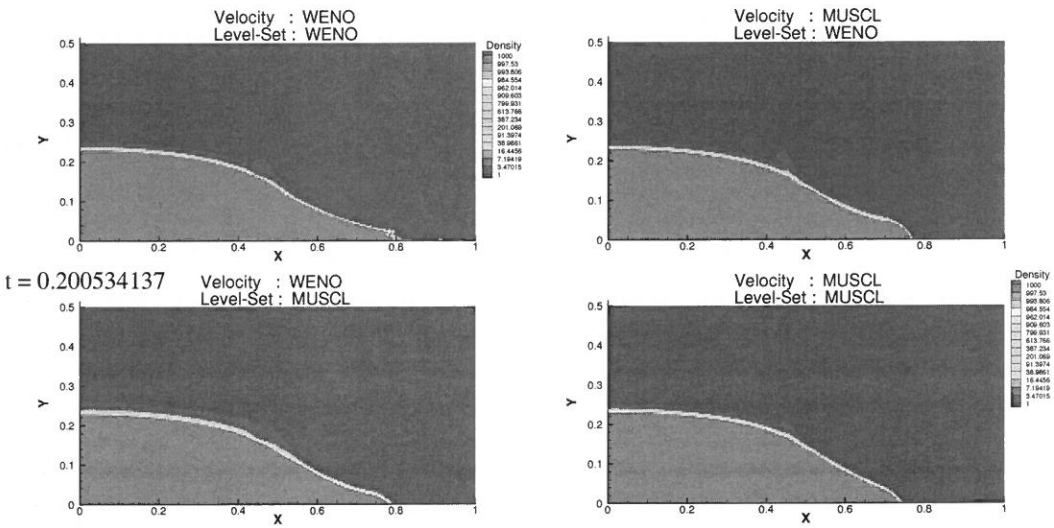


Figure 5.18 – Dambreak tests in the 2020 version fo the code: $t = 0.2\,[s]$.
After, the WENO - WENO simulation diverges.

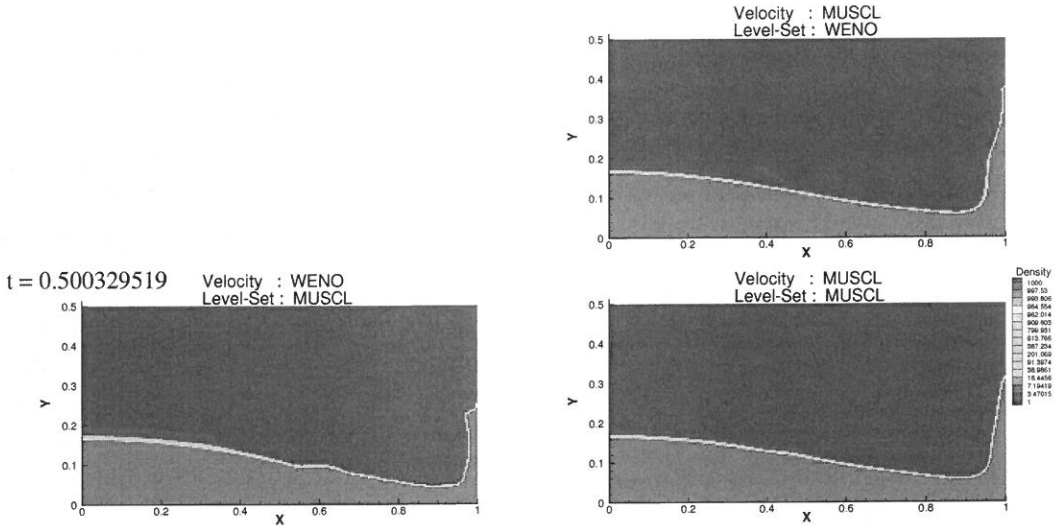


Figure 5.19 – Dambreak tests in the 2020 version fo the code: $t = 0.5\,[s]$.

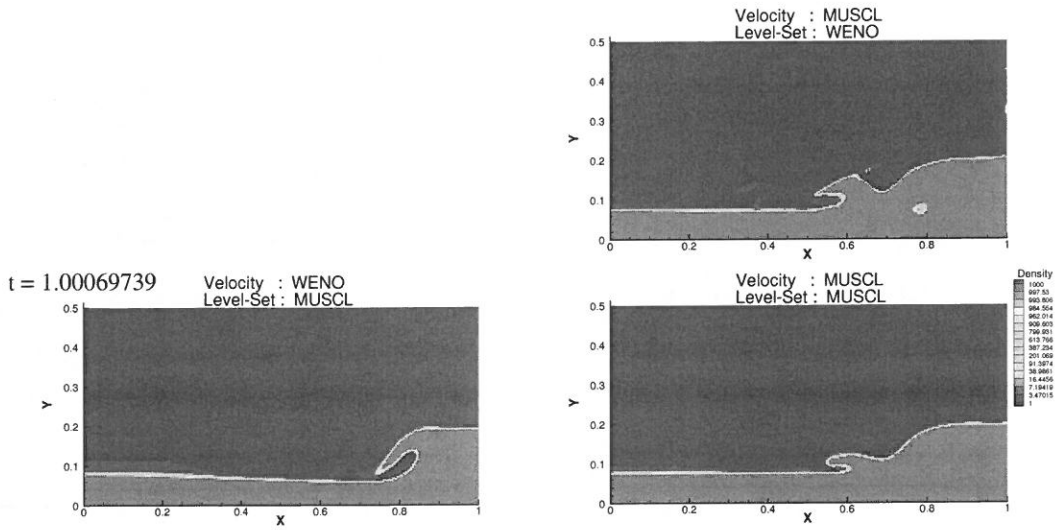


Figure 5.20 – Dambreak tests in the 2020 version fo the code:$t = 1\,[s]$.

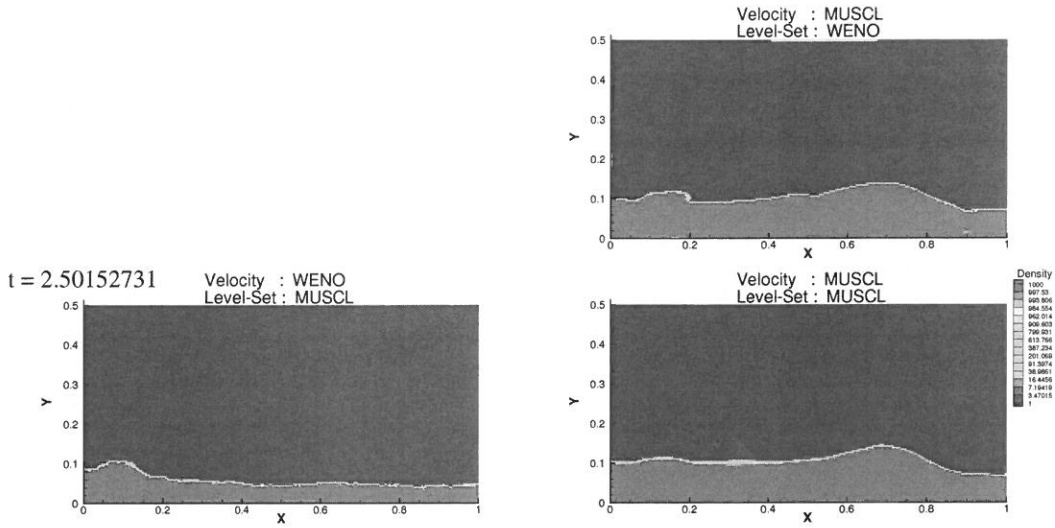


Figure 5.21 – Dambreak tests in the 2020 version fo the code: $t = 2.5, [s]$.

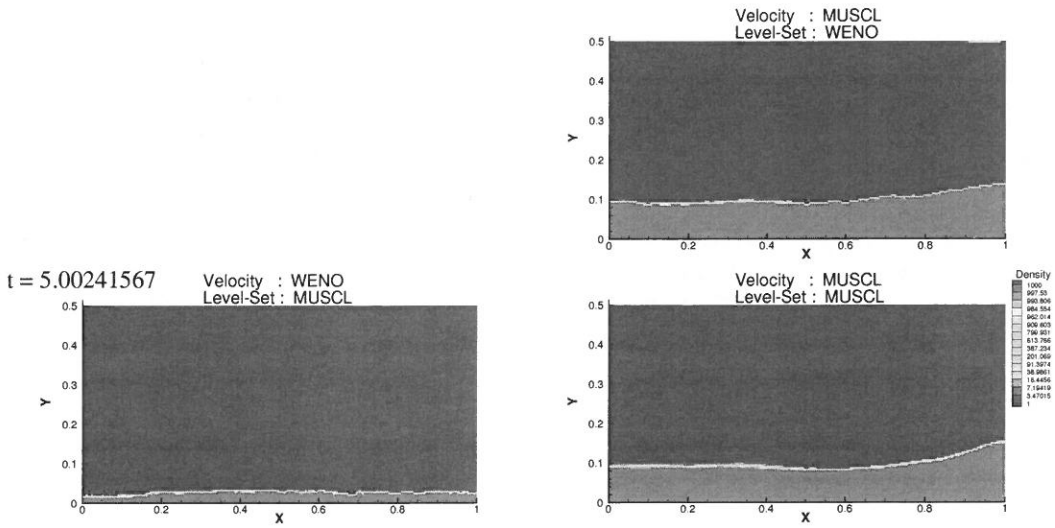


Figure 5.22 – Dambreak tests in the 2020 version fo the code: $t = 5\,[s]$.

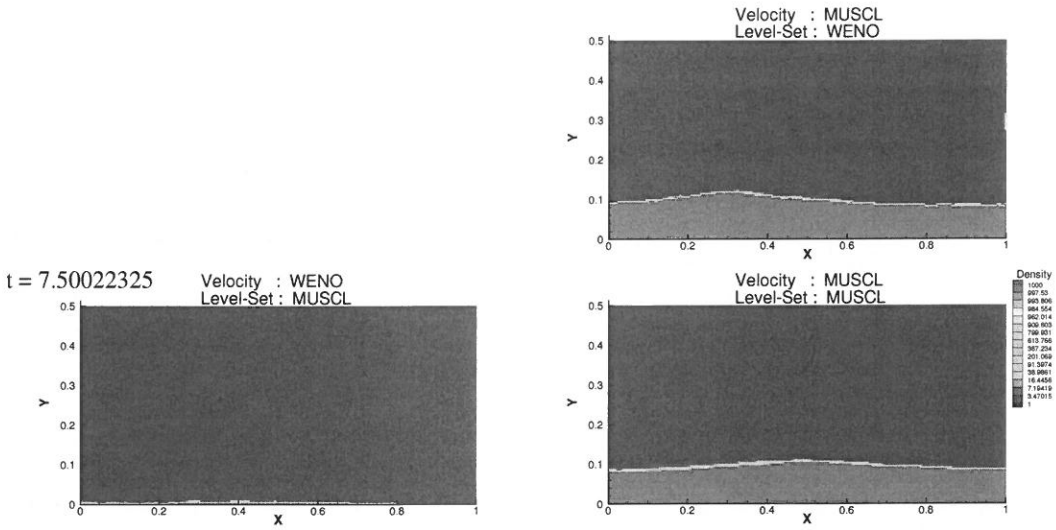


Figure 5.23 – Dambreak tests in the 2020 version fo the code: $t = 7.5\,[s]$.

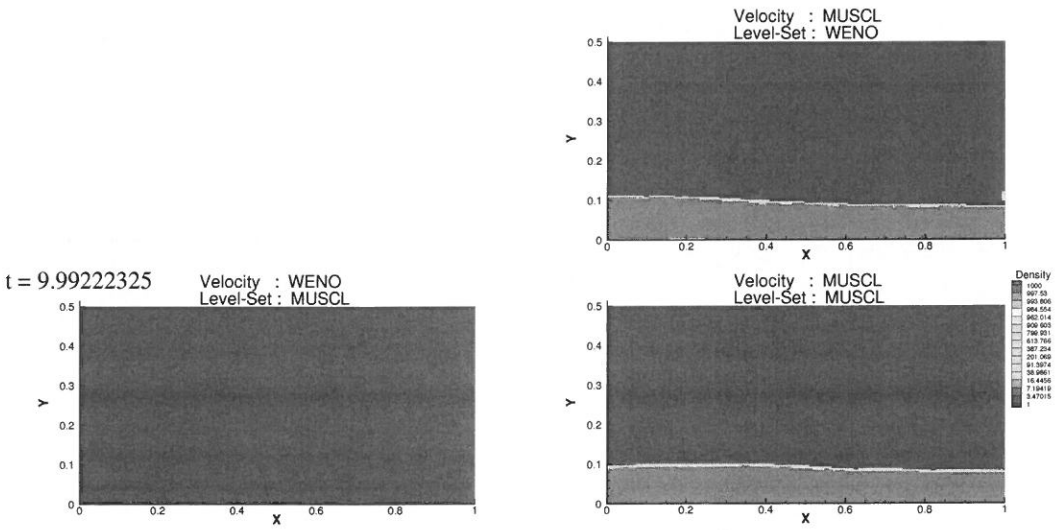


Figure 5.24 – Dambreak tests in the 2020 version fo the code: $t = 10\,[s]$.

Chapter 6

Conclusions

The development of a multiphase method in Gridflow is, presently, at an early stage. Some of the results obtained with the lower order reconstruction schemes are promising, but many obstacles remain.
At the moment, the principal difficulties are the instability of the WENO - WENO simulations and the spurious oscillations in the hydrostatic case.
It should be noted that, on one side, multiphase flows with high density ratios are notoriously difficult to treat and, on the other, a vast literature on multi-fluid schemes has accumulated over the last few decades.
Many avenues are open to further investigate and to try to mitigate the instabilities encountered. Checks on the regularity of the density field can be implemented, modifications can be applied to the advection equation, or to the computation of the density field from the Level-Set.
An aspect that so far has not been explored is the redistantiation function (3.7). While its use for correcting the Level Set field has been proven, in [4], to be a source of instabilities for Gridflow's WENO scheme, it could be used to regularize it when computing the density.
The ghost fluid method, as an alternative to the delta functions used so far, can be explored.
Once the stability problems are tackled, the mass conservation properties of the scheme should be investigated, as they are known to be a critical point for the Level Set method. If these are deemed insufficient, many improvements to the base method can be explored, such as coupling with a volume of fluid method, coupling with a Lagrangian method and so on.
Other aspects that were ignored and that warrant further exploration are the inclusion of variable viscosity and surface tension.
Once these basic aspects are taken care of, more advanced ones, such as the coupling of the method with the immersed boundary and with the turbulence model can be dealt with.
These other aspects of Gridflow are themselves currently under development, and their exploration and expansion can be, in the future, another aspect of this work.

Bibliography

[1] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dmitry Karpeyev, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.13, Argonne National Laboratory, 2020.

[2] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dmitry Karpeyev, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc Web page. `https://www.mcs.anl.gov/petsc`, 2019.

[3] A. Bardin, G. Oger, and D. Le Touze. Simulation of interfacial flows using a cartesian explicit finite volume solver. *11th World Congress on Computational Mechanics, WCCM 2014, 5th European Conference on Computational Mechanics, ECCM 2014 and 6th European Conference on Computational Fluid Dynamics, ECFD 2014*, (Wccm Xi):5402–5414, 2014.

[4] Amélie Bardin. *Développement de méthodes d'interface dans un solveur hydrodynamique basé sur une résolution volumes finis sur maillage cartésien en hypothèse faiblement compressible.* PhD thesis, 2015.

[5] P Bigay, A Bardin, G Oger, and D Le Touzé. A cartesian explicit solver for complex hydrodynamic applications. *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE*, 2:1–10, 2014.

[6] Pierre Bigay. *Développement d'un solveur faiblement compressible sur maillage cartésien pour les écoulements hydrodynamiques autour de corps.* PhD thesis, 2015.

[7] Hans Bihs, Arun Kamath, Mayilvahanan [Alagan Chella], Ankit Aggarwal, and Øivind A. Arntsen. A new level set numerical wave tank with improved density interpolation for complex wave hydrodynamics. *Computers & Fluids*, 140:191 – 208, 2016.

[8] Alexandre Joel Chorin. Numerical solution of the navier-stokes equations. *Mathematics of computation*, 22(104):745–762, 1968.

[9] Ronald P Fedkiw, Tariq Aslam, Barry Merriman, Stanley Osher, et al. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of computational physics*, 152(2):457–492, 1999.

[10] Jean-Luc Guermond, Peter Minev, and Jie Shen. An overview of projection methods for incompressible flows. *Computer methods in applied mechanics and engineering*, 195(44-47):6011–6045, 2006.

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

[12] Daniel Hartmann, Matthias Meinke, and Wolfgang Schröder. Differential equation based constrained reinitialization for level set methods. *Journal of Computational Physics*, 227(14):6821–6845, 2008.

[13] Cyril W Hirt and Billy D Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of computational physics*, 39(1):201–225, 1981.

[14] Myungjoo Kang, Ronald P Fedkiw, and Xu-Dong Liu. A boundary condition capturing method for multiphase incompressible flow. *Journal of Scientific Computing*, 15(3):323–360, 2000.

[15] Zhe Li, Guillaume Oger, and David Le Touzé. A finite volume weno scheme for immiscible inviscid two-phase flows. *Journal of Computational Physics*, page 109601, 05 2020.

[16] Xu-Dong Liu, Ronald P Fedkiw, and Myungjoo Kang. A boundary condition capturing method for poisson's equation on irregular domains. *Journal of computational Physics*, 160(1):151–178, 2000.

[17] Thibaut Ménard. Développement d'une méthode Level Set pour le suivi d'interface. Application de la rupture de jet liquide. 2007.

[18] Stanley Osher and Ronald P Fedkiw. Level set methods: an overview and some recent results. *Journal of Computational physics*, 169(2):463–502, 2001.

[19] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[20] James Edward Pilliod Jr and Elbridge Gerry Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *Journal of Computational Physics*, 199(2):465–502, 2004.

[21] Roger Temam. Sur l'approximation de la solution des équations de navier-stokes par la méthode des pas fractionnaires (ii). *Archive for rational mechanics and analysis*, 33(5):377–385, 1969.

[22] Louis Vittoz. *Contributions au développement d'un solveur volumes finis sur grille cartésienne localement raffinée en vue d'application à l'hydrodynamique navale.* PhD thesis, 2018.

[23] Steven T. Zalesak. Fully Multidimensional Flux-Corrected Transport Algorithms for Fluids. *Journal of computational physics*, 31(3):335–362, 1979.

[24] Davide Zuzio, Annagrazia Orazzo, Jean Luc Estivalèzes, and Isabelle Lagrange. A new efficient momentum preserving Level-Set/VOF method for high density and momentum ratio incompressible two-phase flows. *Journal of Computational Physics*, 410:109342, 2020.