

Université catholique de Louvain Ecole polytechnique de Louvain Institut de Mécanique, Matériaux et Génie Civil

DEVELOPMENT AND VALIDATION OF A 2D VORTEX PARTICLE-MESH METHOD FOR INCOMPRESSIBLE MULTIPHASE FLOWS

Gaël Lorieul

Jury:

- Pr. R. Ronsse (UCL, chairman)
- Pr. Y. Bartosiewicz (UCL, supervisor)
- Pr. P. Chatelain (UCL, supervisor)
- Dr. V. Deledicque (Bel V)
- Pr. P. Brancher (IMFT, Toulouse)
- Pr. G. Winckelmans (UCL)
- Pr. J-F. Remacle (UCL)
- Pr. L. Bricteux (UMons)

Louvain-la-Neuve, August 2018

©2018 Gaël Lorieul All Rights Reserved

Abstract

The motivation of the present thesis is to assess the potential gains that Vortex Particle-Mesh methods could bring to numerical simulations for the nuclear industry, and in particular to pool-type reactors using heavy liquid metal coolants. One such reactor is the Belgian MYRRHA Gen IV prototype reactor. Indeed, the flow taking place within the vessel (i) tends to be highly-advective and (ii) has local sources and sinks of momentum that generate global motion of the coolant in much greater volumes. Those conditions makes the use of Vortex Particle-Mesh method potentially attractive.

The present thesis develops a Vortex Particle-Mesh method for multiphase flows with heat transfer. Starting from an existing VPM solver for single phase flows, features are added (such as level set or smeared interface methods), numerical difficulties are studied and solutions to the latter are proposed. The correctness of the resulting VPM solver is assessed on popular CFD benchmarks, and its performance both in terms of accuracy and computational efficiency is compared to the one of a traditional velocity-pressure solver. Eventually a case of study similar to those that could be encountered in nuclear reactors is run to demonstrate the relevance of the solver for industrial flows. vi

One difficulty is that it is far easier to produce numbers from a computer than understand their relevance to real flows.

> Saffman and Baker, Vortex interaction in 1979 Annual Review of Fluid mechanics



(a) CIC interpolation (b) A simulation result (c) Slinging of particles scheme due to circular motion

Figure 1: Figures from Christiansen's original 1970 lab report [1] where the Vortex Particle-Mesh (aka Vortex-in-Cell) method was introduced for the first time in the History of CFD.

viii

Acknowledgements

A PhD is never the work of a single person. Here, I would like to express my gratitude to all people who made a significant contribution to either my work or more generally to my experience as a PhD student. Also, as a forreigner who came to Belgium especially to pursue that PhD, and who will move to another country after its completion, the "PhD experience" does not limit itself to what happens at the workplace but to everything that I have experienced during my stay here in Belgium.

Firstly, I would like to thank my supervisors Prof Yann Bartosiewicz and Prof Philippe Chatelain for their help all along the thesis. Their support was valuable both regarding the scientific aspects and the writing or preparation of scientific communications, which includes the present dissertation. Also for their patience and openness, especially when faced my doubts or when discussing against my contradictory opinions. Furthermore, I would like to mention the support of Laurent Bricteux and Matthieu Duponcheel, members of my "comité d'accompagnement".

BelV represented by Vincent Deledicque without the funding of which this PhD could never have existed. Thank you for putting your trust in me, especially at the "PhD confirmation" step where much was to be achieved yet. I hope that the content of the present document satisfies your expectations.

I would also like to thank Pierre Brancher, Grégoire Winckelmans, Jean-François Remacle, Laurent Bricteux and Renaud Ronsse, the members of my "jury de thèse" for their careful reviewing of the present document.

Prof Dr.-Ing. Ernst von Lavante for convincing me into doing a PhD. Pursuing a PhD programme is often associated with learning bleeding-edge technical skills. This is certainly true, however in my case I think the biggest benefit was all the non-technical skills that I developped. This PhD was an excellent exercise to learn how to set, follow & achieve objectives, manage my personnal motivation through ups and downs as well as keeping a constant level of hard work. In other words, it was a way to get closer from Freedom. I am really happy to have made the decision of doing a PhD, and if it wasn't for you, professor Lavante, I might have chosen otherwise...

My parents for your understanding when I decided to go for a PhD eventhough that was not what you had in mind, for supporting me throughout the PhD and for your involvment in my education from my (very) early years. Also thank you Mom for taking the decision to stay at home during a large part of your life so that you could spend the entirety of your time breeding my brother and I, despite the underlying risks and sacrifices.

The people at the TFL for their friendliness, with special mentions to (in alphabetical order): Subir Bhaduri, Frank Hesbois, Haddy Mbuyi Katshiatshia, Ehsan Tavakoli and Jie Zheng. Many thanks to Isabelle Hennau, the kind fairy who takes care of all the logistics at the TFL so that the life of everyone else can run smoothly (and who delivers "Christmas presents" all year long...).

Subir, you showed me that the world had plenty of opportunities to offer. You have been able to move out from India and to live for more than three years here in Belgium, all that out of a mere online PhD offer and a Skype interview. You got out of your PhD with no clear idea in mind and eventually you went on to work at an NGO then started your own business, undertaking projects that seem truly meaningful in respect to your ideals. I used to fear the future and take conservative decisions that narrowed my perspectives. Thanks to you I am more confident and open to life and whatever it might reserve for myself.

Ehsan, I am very impressed by your strong-will, your sense of sacrifice and your hard work. For keeping your friendliness and taking some of your precious time to explain things to me, even in the harshest and most stressful moments for you. We haven't met for very long but your sense of struggle is very inspiring to me and also reminds me of the privilege that I may have thanks to my roots or schooling path. I heartfully wish you the best for yourself and the laboratory you are establishing. May you keep strong and follow your path, despite the interference of geopolitical events that are far beyond your control.

I would also like to thank mathematics professors Luc Haine, Augusto Ponce and Paolo Roselli for welcoming me into their classes eventhough I sometimes lacked some prerequisites. Thanks to teaching assistant Justin Dekeyser for answering my (sometimes naïve) questions. Following these classes allowed me to better appreciate the difference in problem-solving approach between engineers and mathematicians. Understanding the mathematicians' way-ofthinking and sense of aesthetics now allows me to read any mathematical text, not just those written by engineers for engineers: Mathematics with little to no calculations, where emphasis is given to concepts, curiosity and exploration. Thanks to you, a wide field of knowledge and science is now accessible to me, let alone a way of seeing the World.

Similarly, I would like to thank professor Kim Mens for welcoming me in his software development class. Following this lecture gave me a broader sight on the existing software development techniques used in the industry, helped me become a better programmer and made me understand what research in computer sciences is about. But beyond this it was also an excellent opportunity for me to evaluate the software development skills I have gained during my PhD as I was developping a C++ CFD code against thee of soon-to-gratuate master students, and hence know better where I stand. All of this will definitively be very useful to me since I will soon be looking for a job, and that it is very likely for it to have a software development component.

A special thought to professor Paolo Rosseli. Eventhough we don't know each other so well, I must say that I am very impressed by your ability to explain things simply, your humility, your acceptation of contradiction and doubt, your friendliness, your open-mindness to seek collaboration with people outside your field and your positive problem-solving attitude, all that on top of your "ordinary" research activities. This is a lot for a single person and your example is very inspiring to me. I wish you best luck with your work on unearthing the Clifford algebra theory and using it to solve practical engineering problems. I was delighted to learn the excellent results you achieved in 2D gestual recognition and wish you the same success in the much more difficult 3D case...

I would also like to thank the anonymous crowd behind the free (as in freedom) software tools that I have been using since high school and without which this thesis would have been a lot more difficult and expensive to pursue. Also a thought to people like Aaron Schwartz or Alexandra Elbakyan who struggle to make the research world a better place, sometimes at the sacrifice of their own lives.

To all the friends that I met during my stay in Belgium for their company and shared experiences: Séverine, Louisa, Yafei, Anthony, Donovann, Ana, Priscilla, and so many others.

The Tori Kago Ju-jutsu dojo for showing me a different side of Belgium (not every Walloonian works at the university !) and for helping me gain confidence in myself and in my relation to others. Also the partnership with Segawa dojo and our membership at the VJJF was a good opportunity to have a glimpse into Flanders. Of course, I must mention Stephane Piret's passion and dedication¹, as well as the good-spirit of all members, without which the dojo would not exist.

The Brussels Manneke Piss Hash House Harriers for the cheers and its Beer Meister Snakes & B. for making me discover the wide variety of Belgian brews².

Higgins, it is a pleasure to have crossed paths with you. You showed me that one could always be cheerful, open to discussion and convincing at the same time. Rest assured that your teachings of Disasterism will not be forgotten...

On on to whatever awaits in the future ! As long as there are two blobs or more, I'll know where to go!

Que sera, sera Whatever will be, will be The future's not ours, to see Que Sera, Sera What will be, will be

¹Eventhough I am quite sure he will not like to see his name being pushed forward... :P ²The other Beer Meisters have done an amazing job too but Snakes was first! (and she over-achieved as usual!)

xii

Contents

1	Introduction			1			
	1.1	Contex	xt	. 1			
	1.2	Motiva	ation for the present study	. 3			
	1.3	3 Lagrangian Vortex methods					
		1.3.1	Eulerian, Lagrangian and Eulerian-Lagr. methods	. 5			
		1.3.2	Vortex methods and velocity-pressure formulation	. 8			
		1.3.3	A first general choice: Lagrangian vortex methods	. 9			
	1.4	4 Vortex Particle-Mesh methods					
		1.4.1	Particle-particle and particle-mesh methods	. 9			
		1.4.2	Classification by discretization	. 11			
		1.4.3	Extensibility to 3D	. 15			
		1.4.4	Arguments in favor of the VPM method	. 16			
	1.5	Moder	n and/or relevant VPM methods	. 16			
	1.6	Resear	ch questions	. 18			
2	Sing	gle-fluio	d VPM method	21			
	2.1	Govern	ning equations	. 21			
	2.2	Some i	important conventions and definitions	. 22			
		2.2.1	Systems of coordinates	. 22			
		2.2.2	Frames of reference	. 23			
		2.2.3	Discretization of space	. 23			
	2.3	3 Numerical method					
		2.3.1	Discretization of space	. 24			
		2.3.2	General algorithm of the VPM method	. 26			
		2.3.3	Interpolating between particles and grid	. 26			
	2.4	Enforc	ing boundary conditions	. 30			
		2.4.1	Free-slip boundary conditions	. 30			
		2.4.2	No-slip boundary conditions	. 30			
	2.5	Summa	ary	. 33			
3	Inte	erface o	capturing VPM method	35			
	3.1	Selecti	ng an interface capturing method	. 35			
		3.1.1	Front tracking	. 36			
		3.1.2	Particle methods	. 37			
		3.1.3	Level set	. 38			
		3.1.4	Volume of fluid	. 38			

		3.1.5	Hybrid methods	39
		3.1.6	${\rm Recent\ evolutions\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\$	39
		3.1.7	Chosen method \ldots	40
	3.2	Level	set method	40
		3.2.1	Updating the level set field to follow the interface	41
		3.2.2	Boundary conditions for level set	42
		3.2.3	Non-smoothness of level set field $\ldots \ldots \ldots \ldots \ldots$	43
		3.2.4	Level set reinitialization	43
		3.2.5	Local level set method $\ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	48
		3.2.6	Strategies to trigger level set reinitialization	52
	3.3	Partic	ele-based level set	54
4	Cor	nputir	ng the phase- and interface-dependent terms	57
	4.1	Surfac	ce tension term	57
		4.1.1	Level set "Flip-flop" mode	59
		4.1.2	Computation of curvature	66
	4.2	Visco	us term	69
		4.2.1	Single-fluid model	69
5	Hig	h mas	s-density ratio VPM method	73
	5.1^{-}	Introd	luction	73
		5.1.1	Interface modelling methods	73
		5.1.2	Smeared interface vortex methods	74
		5.1.3	Buoyancy numerical instability	75
	5.2	One-d	limensional model of the instability	75
		5.2.1	General view on the problem	75
		5.2.2	Simplified governing equations	76
		5.2.3	Simplified geometry	77
		5.2.4	Non-dimensionalization of problem	78
		5.2.5	Discretization of the material derivative	79
		5.2.6	Governing equation on errors	80
		5.2.7	General sequence describing the instability	81
		5.2.8	Results with 1D model	82
	5.3	Comp	parison of 1D model against 2D simulations	86
		5.3.1	Comparison of material acceleration $D_t v$ profiles	86
		5.3.2	Comparison of convergence rates of error norm E	87
		5.3.3	Comparison of critical mass density ratio ρ_+/ρ values	90
		5.3.4	Summary	90
	5.4	Achie	ving stability through under-relaxation	91
		5.4.1	Ideal under-relaxation	91
		5.4.2	Alternative to ideal under-relaxation	93
		5.4.3	Results of under-relaxation	94
		5.4.4	Comparison of 1D model against 2D VPM solver	95
	5.5	Harm	onic analysis of the instability	95
	5.6	Practi	ical view on the buoyancy numerical instability	99
		5.6.1	Influence of the under-relaxation factor α	100
		5.6.2	Influence of the time integrator	100
		5.6.3	Influence of the mass density profile	102

	5.7	5.6.4 Measuring the "unstableness" of a method	$104 \\ 105$
6	Nur	nerics and solver validation	107
Ū	61	Overview of the benchmarks	107
	0.1	6.1.1 Benchmarks assessing the interface capturing method	107
		6.1.2 Benchmarks assessing the computation of surface tension	108
		6.1.3 Benchmarks on complex flows	108
	6.2	Definition of criteria	113
	-	6.2.1 Criteria used in benchmarks	113
		6.2.2 Numerical calculation of criteria	114
	6.3	Dimensionless numbers	118
	6.4	An Eulerian velocity-pressure solver to compare to	119
	6.5	Numerical parameters of the solver	121
	6.6	Numerical investigations	121
		6.6.1 Choice of level set reinitialization strategy	121
		6.6.2 Choice of the surface tension term	122
		6.6.3 Appearance of trenches	123
		6.6.4 Choice of tangential filter for the level set field	123
		6.6.5 Influence of missing term in Thirifay's formulation	124
		$6.6.6 {\rm Influence \ of \ under-relaxation \ on \ simulations' \ output} . .$	125
		6.6.7 Comparison of elliptic and parabolic vorticity solvers	128
	6.7	Validation of the VPM level set method	133
	6.8	Validation of surface tension	139
	6.9	Validation of full solver	145
		6.9.1 Computational efficiency	152
	6.10	Simulations and extension to heat transfer	154
		6.10.1 VPM method with heat transfer	155
		6.10.2 Influence of Prandtl number	157
	0.44	6.10.3 Simulation of nuclear-like cases	157
	6.11	Summary	159
7	Con	clusions and perspectives	161
	7.1	General conclusion	161
	7.2	Achievements and research results	162
		7.2.1 Answers to research questions	162
		7.2.2 Summary of the work accomplished	163
	7.3	Perspectives	165
A	Timeline of some methods18		
в	Weno5 schemes 18		
\mathbf{C}	Algo	orithms	191
	C.1	Level set	192
		C.1.1 Hamilton-Jacobi reinitialization	192
		C.1.2 Fast Marching Method for reinitialization	193
	C.2	VeloGrid: Eulerian velocity-pressure solver	195

Contents

	C.3	Vortex Particle-Mesh	196
D	\mathbf{Tim}	ne-integrators	199
	D.1	Euler-Explicit	199
	D.2	RK2 Midpoint	200
	D.3	Another RK2 time-integrator	200
	D.4	A low storage RK3 time-integrator	201
\mathbf{E}	Flui	id properties	203
\mathbf{F}	Proofs and justification		
	F.1	Single-fluid VPM method	209
		F.1.1 Procedure to obtain velocity from vorticity (Eq. 2.3)	209
	F.2	Interface capturing VPM method	210
		F.2.1 Equivalence between Rouy scheme and upwinding	210
		F.2.2 Level set flip-flop	212
	F.3	Viscous term	212
		F.3.1 Correct expansion of viscous term (Eq. 4.37)	212
	F.4	High mass-density ratio VPM method	213
		F.4.1 Material acceleration sequence for any time integrator	
		(Eq. 5.32)	213
	F.5	Numerics and solver validation	214
		F.5.1 Volume calculation using mollifier	214
\mathbf{G}	Vise	cous term in vorticity equation	215

xvi

Chapter 1

Introduction

1.1 Context

The last decade has seen a dramatic reshuffling of the energy geopolitics: depletion of traditional oil and gas reserves, emergence of new fossil reserves (access to arctic oil and gas [4], oil sands [5], shale gas [6]) mostly in regions other than the traditional oil and gas extracting countries, steady development and improvement of renewable technologies, nuclear reactors reaching the end of their original lifespan in Western countries, etc.

In the same time, the needs of customers and nations have been evolving. Indeed, while the energy needs of emerging and developing countries continues to follow exponential growth, developed countries see their energy consumption stagnate (Fig. 1.1a). At the same time, the need for clean energy is stronger than ever, be it in terms of emission of greenhouse gases, NOx or microparticles. Eventually many countries pursue more self-reliance in their energy supply¹.

Given its geopolitical situation, each state decides on an energy mix that aims at combining the strength of each energy source: kWh price, steadiness of energy production, volumes of energy that can be produced, availability of the fuel, response time to demand fluctuations, compatibility with the electric grid², environmental cost and consequences on public health (Fig. 1.1b), etc. In an effort to always keep improving the competitivity of the nuclear energy, and for it to remain an energy of choice in those mixes in the decades to come, the nuclear industry addresses several challenges. The first concerns its safety, which has attracted a renewed public attention after the 2011 Fukushima-Daiichi accident, as well as the public defiance regarding the plans for lifetime extension plans of existing reactors in Western countries. The second is the

¹Post-Fukushima Japan is a good example of the economic pressure related to energy imports. Indeed, after being forced to shut down their nuclear facilities due to public hostility, its government had to open them again to compensate for the economic imbalance caused by the energy imports [7] [8].

²The energy supplied by renewable technologies fluctuates greatly in time [12], which can lead either to a blackout of the grid if the supply is insufficient [13], or a need to evacuate the excess energy if it is overpowered, sometimes through exports at negative prices [14]. To protect its grid from an overflow of German electricity, Poland and Czech Republic installed phase-shifting transformers at their German border [15].



(a) Primary Energy Consumption (Quad BTU) (Source : [9]).



(b) Heavy smog on 24 Oct 2014 in Beijing (Photo: [10]).^{*a*}

^aCoal facilities are large contributors to the Beijing smog, hence the authorities plan to substitute them with nuclear power plants [11].

Figure 1.1: Energy consumption and environmental considerations

cost of nuclear energy, not only the cost of construction and operation but also that of decommissioning and nuclear waste management & storage. Last but not least, the long lifespans of nuclear reactors make the preservation of the accumulated know-how in nuclear reactor construction difficult over the decades, and generates substantial additional costs when a new reactor must be built after a long pause in construction [16].

The nuclear industry proposes two solutions to this problem. The first is to radically change the size of the plants and sell them as energy production units that are low-maintenance, resilient, reliable and assembled through continuous production : those are Small Modular Reactors $(SMR)^3$ (Fig. 1.2a). They are targeted towards isolated regions as a substitute to diesel generators currently used, in particular in the Canadian Arctic. The second is to have a technological change in the reactor design, but keep the unit-production and large-scale plant paradigms : those are GenIV reactors and aim to address the bulk of the energy demand.

Of all the possible GenIV reactor technologies, Lead and Sodium cooled Fast Reactors (resp. LFR and SFR) have very interesting features including long term refueling, nuclear waste produced both in smaller amounts and with much shorter half-times, as well as passive cooling on shutdown through natural convection. This allows to produce electricity at greater efficiency, with even higher safety standards, using more commonly available fuel and with a reduced environmental footprint.

For those reasons, Belgian research center in nuclear energy SCK•CEN decided to focus on the Heavy Liquid Metal (HLM) technology and is currently designing the MYRRHA demonstrator (Fig. 1.2b) [17]. It uses lead as a coolant and has a pool-type configuration. Those two technical decisions have a large impact on the reactor's design at all scales and give rise to new technical challenges, including safety challenges. As one of Belgium's Technical

³https://energy.gov/ne/benefits-small-modular-reactors-smrs



(a) The SEALER design, an example of liquid metal Small Modular Reactor.

(b) A former design of the MYRRHA reactor, a large-scale liquid metal reactor.

Figure 1.2: Two liquid metal reactor designs.

Safety Organizations (TSO), our partner Bel V (who is financing this thesis) is responsible of MYRRHA's commissioning and must assess the safety risks it might pose.

1.2 Motivation for the present study

As for most devices involving fluid flows, nuclear reactors are designed with the help of both experiments and CFD simulations. Those are helpful to assist the general design of the nuclear reactor and also to assess the safety risks. Although the design of nuclear facilities encourages caution and hence the use of well established CFD simulation methods, the nuclear industry has historically played a very important role in the development of novel CFD methods, in particular multiphase methods for bubbly and boiling flows. This was motivated by the inability of the commercial codes of those days to model accurately such physical phenomena that are critical in the design process of a nuclear reactor.

On the other hand, the Vortex Particle-Mesh method, also known as Vortexin-Cell method has proven itself to be very useful for simulating the wakes of aircraft and wind turbines. The VPM method has long been used both in academia and by industries to help address practical engineering problems. As an example, several partnerships and contracts have been performed between the Université catholique de Louvain (UCL) and industries, including energy company Engie. Also the UCL spin-off WaPT is currently working in partnership with the European air traffic manager Eurocontrol. More generally,



Figure 1.3: Simplified schematics of loop- and pool-type reactors.

several research groups are interested in those methods such as the Technical University of Denmark, the Université Grenoble-Alpes, ETH Zürich or the California Institute of Technology. The VPM method is therefore well established in those fields. However to our best knowledge it has never been used in the nuclear industry. Therefore, this thesis aims at assessing the potential usefulness of the VPM method for nuclear applications.

It has been shown in a 2005 article by Hieber and Koumousakos [18] that the VPM method can be very effective at capturing fluid interfaces in the spirit of the level set method. Unlike the more common loop-type reactors (Fig. 1.3a), pool-type nuclear reactors such as MYRRHA have a fluid interface (Fig. 1.3b) whose dynamics is commonly studied with CFD for the reactor design and safety assessment. In particular sloshing simulations have become increasingly popular [19]. Moreover, the reasons for the attractiveness of VPM methods in the simulation of wakes are that (i) the flows take place in large or unbounded domains and (ii) the Reynolds number is high. Pool type reactors having large vessels, the simulation of advection-dominated interfacial fluid flows in those domains appears as a good application case for VPM.

An example of such situation is the conjugated heat transfer at the contact line on the vessel's walls. In pool type reactors, there is a fluid interface separating the liquid metal coolant from the cover gas (e.g. argon). That fluid interface reaches the vessel's and internals' walls where it forms contact lines. Globally the fluid interface is flat, but it does not remain perfectly still. Indeed, the jet exiting the fuel core and the fluid channelled into the heat exchangers, amongst other things, generate small interface oscillations. Those oscillations propagate to the contact line at the walls and makes it move up and down in time with small amplitude. This is described by Figure 1.4, which shows the coolant phase (\square) and cover gas phase (\square) separated by a fluid interface (---). With time the contact line (\bullet) at the wall (\square) moves up and down. Therefore, there is a thin region of the wall (highlighted in the figure) that is at times immersed in the (liquid metal) coolant (Fig. 1.4c), where heat transfers are very strong, and in contact with the cover gas (Fig. 1.4b) the



Figure 1.4: Unsteady heat transfers at a wall with oscillating free surface. (\square) is the coolant, (\square) the cover gas, (\square) a solid wall, (\bullet) the contact line, and (\rightarrow) the heat transfers.

rest of the time, where heat transfers are much weaker. Hence, that region of the wall is subject to heat fluxes (\rightarrow) whose intensity is strongly fluctuating in time. This can lead to thermal stripping of the walls' material. The fact that the flow takes place in large volumes and that it is driven by advection phenomena (interface motion) makes VPM methods desirable for such practical problems.

Since its inception in the 30s, the field of CFD has given birth to an impressive number of methods⁴. Each has strengths of its own and depending on the problem considered, a method will be more appropriate than another. Therefore, why consider VPM and not another CFD method for the simulation of liquid metal surface oscillations? To answer this question the following sections review existing numerical methods, briefly explains their general principles and details their strengths and weaknesses.

CFD methods can be classified in a number of ways: Eulerian versus Lagrangian methods, vorticity versus velocity-pressure formulations, etc. In the next section, a brief overview of such general categories is given. Then, once it will have appeared that a vorticity formulation in a Lagrangian frame-ofreference is appropriate, a more detailed choice will be made in the following section leading to VPM. Eventually, works introducing methods that share similarities with ours will be stated as a reference, and that will allow to appreciate the gap that the present work fills.

1.3 Lagrangian Vortex methods

1.3.1 Eulerian, Lagrangian and Eulerian-Lagr. methods

The most widespread classification of CFD methods is probably to sort them as being part of the Eulerian, Lagrangian and Eulerian-Lagrangian families of methods. Table 1.1 gives a summary of such classification applied to some CFD methods.

⁴For curiosity, see appendix A.

	Eulerian	Eulerian-Lagrangian	Lagrangian
	discretization	discretization	discretization
Eulerian	Classic finite	N / A	NI / A
frame of ref	differences/volumes/etc.	IN/A	\mathbf{N}/\mathbf{A}
		Particle-laden flows,	
Eulerian-		Particle-based interface	
Lagrangian	N/A	capturing methods	N/A
frame of ref		(MAC, front tracking),	
		Arbitrary Lagrangian-Eul.	
Lagrangian	N / A	Particle-mesh methods	SDH
frame of ref	1N/PL	(PIC, VPM)	51.11

Table 1.1: A classification of some CFD methods through the Eulerian/Lagrangian prism.

Purely Eulerian or Lagrangian methods

Most times, governing equations of flows are written in the Eulerian frame-ofreference

$$\frac{\partial q}{\partial t} + (\boldsymbol{u} \cdot \nabla)(q) = Rhs(q). \tag{1.1}$$

In that case flow fields are represented by functions q of Eulerian coordinates $q(\boldsymbol{x},t)$ where \boldsymbol{x} corresponds to a position in space and t is the time. It is nevertheless possible to express those same fields by functions \tilde{q} of Lagrangian coordinates $\tilde{q}(X,t)$ where X denotes a particle of fluid and t is the time

$$\frac{d\tilde{q}}{dt} = Rhs(\tilde{q}). \tag{1.2}$$

Note that the advection term vanishes, while the Right-Hand Side (RHS) operator remains unaltered.

Since Eulerian governing equations are written in terms of Eulerian coordinates, they are more naturally discretized in space by a static grid. On the other hands, Lagrangian equations are more naturally discretized by a collection of particles that move along the flow. For some simple flows the particles might preserve a fairly structured formation and behave as a "moving grid". However in most cases the particles are stirred by the flow and no regular pattern emerges.

The main benefit of using a Lagrangian formulation of the Navier-Stokes equations is that the advection term vanishes. Beyond removing the only nonlinear term of the equation, this implies that (i) the CFL constraint on the time step is alleviated, and (ii) the dissipation and dispersion errors caused by its numerical discretization vanish. Note however that using particles introduces numerical errors as well, although it is weaker, mostly dissipation error but also dispersion error (the latter being caused by the remeshing procedure that will be introduced in chapter 2 as was shown by Marichal [20]). The main drawback is that the irregularity of the particle arrangement can make the right-hand side more difficult to compute. Additionally, the particles may empty some areas of the domain and/or cluster in other regions which can cause some numerical issues (Lagrangian distortion, see Section 2.3.1). In other words, Lagrangian methods tend to be more complex than Eulerian methods because they handle a cloud of particles instead of a grid but can be advantageous by alleviating the advective term.

Because of the simplicity of its discretizations, the Eulerian approach is employed by most CFD methods. On the other hand, the Lagrangian approach is very appropriate for the simulation of particles and hence has long been very popular amongst plasma physicists [21] [22] or for the study of particles carried away by a flow [23]. Nevertheless, since the early days⁵ of CFD, Lagrangian methods also have proved themselves superior to Eulerian methods for the simulation of certain continuums, in particular supersonic flows, where they were very effective at capturing shock waves [24] [2]. Indeed Eulerian methods tended to smear out shocks, whereas Lagrangian methods preserved them much better.

CFD methods used in most commercial software (ANSYS Fluent, Star-CCM+, etc.) rely mostly on finite volume methods for their simulations, which is to say grid-based Eulerian solvers. On the other hand, a very well known purely Lagrangian method is Smooth Particle Hydrodynamics⁶ (SPH). It handles the unstructured behavior of the particles' cloud by using quadrature operators on the particles themselves.

Hybrid Eulerian-Lagrangian methods

Whereas the families of purely Eulerian and purely Lagrangian methods are very specific, the class of Eulerian-Lagrangian methods is fairly broad and covers a variety of methods using very different paradigms. As a general fact, their motivation is to find a compromise between both Eulerian and Lagrangian approaches and balance their respective strengths and weaknesses.

In particular one of the early motivations was to develop a method that would capture shocks with little dissipation while preserving from Lagrangian distortion. The most revealing example is probably the Arbitrary Lagrangian-Eulerian (ALE) [2] method that allows to solve the governing equations anywhere on the Eulerian-to-Lagrangian spectrum for any points of time and space. This allows to benefit from Lagrangian or near-Lagrangian equations where useful, and at the same time prevent a too important grid distortion.

Also, the Lagrangian approach describes very naturally and efficiently the motion of fluid interfaces. Examples include particle-based methods, such as the MAC method [29] [30], or front tracking methods [31]. Both MAC and Tryggavson's methods use an Eulerian approach for the bulk of the solver, therefore those methods can be thought as Eulerian methods with a Lagrangian discretization of the interface. Note that for some flows, surface tension may need to be computed from those particles [30].

 $^{^5\}mathrm{Von}$ Neumann noticed the usefulness of particle methods for the simulation of shock flows as early as 1944 [24]

 $^{^{6}}$ Nice review of SPH in [25], explanation in [26], user experience in [27], see also a brief summary of its historical development in [28].

Some methods express the flow's physics using a mix of Eulerian and Lagrangian governing equations. A typical example are the methods used for particle-laden flows, which advect the particles/bubbles/droplets using a Lagrangian approach and resolve the Eulerian form of the Navier-Stokes equation for the immersion fluid [23].

Eventually, some methods express the flow's governing equations in the Lagrangian frame-of-reference but perform parts of the computation on an Eulerian grid: those are the particle-mesh methods and will be described in more detail in a following section (see section 1.4.1). This allows to work with a collection of particles in an arbitrary formation, while still being able to use methods requiring more structured meshes. Examples include the Particle-in-Cell method (PIC) [2] introduced by Harlow and Evans in 1957 [32] and the Vortex Particle-Mesh method (VPM), also known as Vortex-in-Cell method (VIC), proposed by Christiansen in 1973 [33].

1.3.2 Vortex methods and velocity-pressure formulation

Vortex methods can be advantageous on unbounded or "half-bounded" domains where vorticity is compact. An example of that is the trailing vortices of aircraft [34] (Fig. 1.5a). There, the wing acts as a source of vorticity, initially forming a vortex tube. Outside of this tube, the vorticity decays quickly to zero whereas velocity fields remain strong. Where velocity-pressure methods would need special requirements for boundary conditions and/or a larger and hence more expensive computational domain, vortex methods can tolerate an homogeneous Dirichlet boundary condition enforced on the boundaries of a fairly compact computational domain (see for example the jet simulation on Figure 1.5b). For those reasons, vorticity has played a very important role in the study of aircraft aerodynamics for a long time [35]. More generally vortex methods are relevant to flows in unbounded domains [36], whereas they don't have an edge on velocity-pressure formulations for flow structures occupying a whole (bounded) domain or for boundary layers⁷.

On the other hand, the major difficulty with vortex methods is the handling of no-slip wall boundary conditions that cannot be expressed simply in terms of vorticity. Solutions to this issue exist and will be described in the next chapter (see section 2.4). However, they enforce the boundary condition in an indirect way. Additionally, boundary layers present very thin flow structures with high gradients in the normal direction to the wall. Simulation of such flow is therefore best handled with anisotropic Eulerian meshes whose cells are much larger in the tangential direction and finer in the normal direction. Hence several research works [37] [38] have considered the coupling of velocitypressure solvers for the simulation of the hydrodynamics in the surrounding of a bluff body, and thee of the trailing wake.

⁷Note that viscous diffusivity is the same for velocity or vorticity.



(a) Vorticity field in the wake of an aircraft [35]. Vorticity decays quickly to zero which allows a smaller computational domain.

(b) Discretized particles of fluid in a jet and its Kelvin-Helmholtz shear layer [36]. No particles have been seeded outside the jet.

Figure 1.5: Examples of flows where vortex methods exploit the compact support of the working variable more efficiently than velocity-pressure formulations.

1.3.3 A first general choice: Lagrangian vortex methods

For the simulation of unbounded or "half-bounded" flows, vortex methods appear superior to other flow variable choices, while particle-based methods are appropriate for highly advective flows. Hence it is relevant to use vortex methods expressing their governing equations in a Lagrangian frame-of-reference for the study of the fluid interface dynamics of nuclear reactors, especially if the computational domain does not cover the whole reactor vessel but only the near-interface region.

1.4 Vortex Particle-Mesh methods

Now that the choice of a vortex methods expressed in the Lagrangian frame-ofreference has been made, the exact method still needs to be selected. Indeed, this class of methods is still broad.

1.4.1 Particle-particle and particle-mesh methods

Obtention of particles' advection velocity from the vorticity field

For simplicity this subsection considers inviscid methods, which is to say homogeneous transport equations. Historically, the earliest Lagrangian methods were so, such as Rosenhead's method of point vortices [39] or Christiansen's Vortex-in-Cell [33]. The problem here is to convert the vorticity information carried by the particles into a velocity information that can be used to advect them. Several solutions exist, some of them rely exclusively on particles (particle-particle methods), others use a grid (particle-mesh methods).

In Rosenhead's 1931 method of point vortices [39], each particle corresponds to a pointwise vortex that induces advection velocities onto all the other particles following a Biot-Savart law of interaction⁸. The main drawback is that the algorithm must consider for each particle the influence of all the other particles, which yields a quadratic complexity $O(N^2)$ for the algorithm. The original method was computed by hand⁹ and hence Rosenhead considered only a few particles. However, since it was implemented on computer the number of particles simulated increased dramatically and because of the polynomial complexity of the algorithm, the cost of the computation quickly became limiting [42].

The Fast Multipole Method introduced by Greengard and Rokhlin in 1987 [43] as an improvement to Barnes and Hut's 1986 Multipole Method [44], lowers the algorithm complexity to linear complexity O(N) (the original multipole method had an O(Nlog(N)) complexity). It achieves so by grouping far-away particles as one "meta-particle". Closeby particles still need to be considered one by one. Example of such work is Ploumhans Winckelmans et al.'s 2002 article [45]. For an overview of the method refer to Oxley's PhD thesis [38], or the Encyclopedy of Computational Mechanics [46].

Another solution suggested by Christiansen's 1973 Vortex-in-Cell [33] is to compute the velocity from vorticity on a grid by solving an elliptic problem (Eq. 2.3). This requires to interpolate the quantities carried by the particles onto the grid and back. Originally, the Cloud-in-Cell interpolant [47] was used, although it has been replaced by higher order interpolants since then (see section 2.3.3).

Evaluation of Navier-Stokes right-hand side terms

Whereas computing Right-Hand Side terms is straightforward on an Eulerian domain, it is not so much the case when working with particles because of their complex and time-changing cloud arrangement. As in the previous section, two families of methods solve this problem: the particle-particle methods which compute the effects of the RHS directly on the particles, and the particle-mesh methods that perform the computation of the RHS on an Eulerian grid.

The particle-mesh method for the evaluation of Right-Hand Sides is the continuity of the particle-mesh method to transform vorticity into velocity and uses the same interpolants. With this approach, the RHS is computed through traditional methods such as finite differences on the grid before being interpolated to the particles, where it is time-integrated. This can be done by viscous

 $^{^{8}\}mathrm{In}$ the case of a pointwise vortex, the induced velocity scales as one over distance to vortex center.

 $^{^{9}}$ The method of point vortices was originally devised to simulate the Kelvin-Helmholtz instability at a time when computers where not available. The earliest computer implementations of the method are from Birkhoff in 1959 [40] and Hama's 1962 work [41].

splitting which consists in isolating the RHS into a separate equation, as was the original spirit of the Particle-in-Cell method [2]. In that case the advection of the particles is time-integrated separately from the viscous effects. As a result, viscous effects cannot have an influence on the particles' advection within a time step. Therefore splitting comes with a lower convergence rate [48] [49]. Fortunately better alternatives have since then been developed and modern methods handle the viscous term in a coupled way [34] (Algo. 1), which is to be preferred.

Particle Strength Exchange methods including Degond's 1989 original scheme [50] are a family of particle-particle method which implements viscous diffusion through an integral operator that redistributes vorticity between particles in such a way that vorticity is preserved. A generalization of PSE for any operator is given by Eldredge et al. in 2002 [51]. For further information on the matter see the section dedicated to it in Cottet's 2000 book [52], the Encyclopedy of Computational Mechanics [46], or Winckelmans' and Leonard's 1993 paper [53].

A more radical approach is the weight differentiating approach embedded in Smooth Particle Hydrodynamics (SPH) methods. In SPH each particle is seen as a member of a statistical sampling of a flow variable. The field can be interpolated anywhere by performing a Monte Carlo integration which takes the form of a weighted sum. The laplacian (or other differential operators) embedded in e.g. the viscous term of the momentum equation are directly calculated on each particle as weighted sums as well [54] [55] [56].

Eventually some older methods have been tried but are no longer relevant nowadays. One such example specific to vortex blob methods is the vortex blob spreading [57] which consists in diffusing vortex blobs individually. It has been proved to be inconsistent by Greengard in 1985 [58]. Another example is the random vortex walk introduced by Chorin in 1973 [59], which consists in advecting particles as random walkers to emulate diffusion. It is a converging but not very efficient algorithm.

Particle-particle and particle-mesh methods

Usually if a particle-particle (resp. particle-mesh) method is used to obtain particle advection velocity from their vorticity, then a particle-particle (resp. particle-mesh) method will be used for evaluating the Right-Hand Side terms. Nevertheless, in all generality one can classify methods according to whether the computations are performed on a mesh (mesh-based methods), particles (particle-particle methods) or if it involves an interpolation between both (particlemesh methods). Table 1.2 classifies some methods in those three categories. In the case of particle-mesh and particle-particle methods, Table 1.3 summarizes different methods for obtaining the particles' advection velocity and to evaluate the RHS term.

1.4.2 Classification by discretization

Vortex methods can be classified in the way vorticity is been discretized. Figures 1.6 and 1.7 illustrates each discretization type.

Table 1.2: Some CFD methods classified according to particle and/or mesh discretization and the flow variables used.

	Velocity-pressure methods	Vortex methods
Mach	E.g. level set finite differences,	Vorticity finite differences,
Mesn	VOF finite volumes, etc.	Vorticity Transport Model, etc.
Particle-Mesh	Particle-in-Cell	Vortex Particle-Mesh
Particle-particle	Smooth Particle Hydrodynamics	Method of point vortices

Table 1.3: Methods for evaluating particle advection velocity and the equation's source terms.

Family of methods	Obtaining particle advection velocity	Evaluating RHS terms
Particle-Mesh	Particle-mesh interpolation	Viscous splitting, Viscous coupling
Particle-particle	Biot-Savart law of interaction, Fast Multipole Methods	Particle Strength Exchange, Weight differentiation, Random vortex walk, Vortex core spreading,

Finite differences methods

Vortex methods based on finite differences [60] [61] discretized the local form of the governing equations over a grid of nodes. They are very simple to implement and can exhibit high order of accuracy using the appropriate high order schemes. Nevertheless they require either a regular mesh or a conformal mapping to a regular mesh.

Finite volume methods

Finite volume methods discretize the domain in a collection of cells that may have any size or shape (although that has an influence on quality of the results). The integral form of the conservation laws are written in each of these cells. Hence the working variable is an (average) integral of the quantity (e.g. momentum) held in the cell and fluxes are performed with neighbouring cells. Such methods are very robust and conservative by construction, that is why they are the method of choice for most commercial simulation software. However, it is difficult to achieve orders of convergence higher than two with finite volume methods.

The vorticity variant of finite volumes is the Vorticity Transport Model (VTM) and is surprisingly recent. Indeed it was introduced by Brown in 2000 [62] and extended to adaptive meshes by Brown and Line in 2005 [63]. It was devised as a response to other Eulerian numerical methods used for the simulation of helicopter wakes that tended to dissipate vorticity too quickly. By using a conservative finite volume scheme, the VTM is able to simulate such wakes without those drawbacks. It is a competitor of the VPM method as it

has been and is used for the simulation of wakes, such as wind turbine wakes [64] [65] or helicopter wakes [63].

Spectral methods

Spectral methods discretize the fields as a sum of basis functions. This method is known to converge the fastest of all CFD methods when the solution is smooth. However, it does not handle complex geometries well, and moreover it has issues when some areas of the flow domain concentrate most of its features, unlike finite difference or finite volume methods where the grid can be very fin in some areas and much coarser elsewhere.

Those problems can be alleviated using a finite element formulation, such as the Discontinuous Galerkin formulation [66] [67] or spectral element methods. However, the drawback of those methods is the relatively high computational cost and storage requirement.

Boundary-integral methods

Boundary-integral methods express the problems as an integral equation on the domain boundary. They make use of Green's functions to generate a solution in the volume from boundary information. As such they can be extremely computationally efficient for problems whose geometry has a small boundary surface for a large domain volume. Also, they naturally handle unbounded domains. However, they can only solve problem for which the Green's function is available.

Boundary-integral methods typically discretize the boundary as a succession of panels, in which case they are known as boundary-element methods [68]. The latter were introduced by Hess and Smith in 1964 [69]. An examples of vorticity-stream function solver is the 1999 work by Skerget et al. [70].

Method of point vortices

One of the earliest CFD vortex method was Rosenhead's method of point vortices proposed in 1931 [39]. It discretized vorticity as a collection of particles carrying a single vorticity value. Christiansen's original Vortex-in-Cell used the same paradigm, although later VPM methods moved to a Lagrangian volume approach to benefit from smoother fields. Indeed point vortex methods can generate non-regular solutions such as singular velocity fields. Some later works also use this approach, as Schochet's 1996 work for instance [71].

Vortex blob method

In the vortex blob method, each particle is associated to a vorticity blob which has a Gaussian-like distribution with a given amplitude and spread. Note that two blobs may superpose. The method was first introduced by Chorin in 1973 [59]. An accuracy and convergence assessment of the method is described in Hald 1979 article [72] and the 1982 work by Nakamura et al. [73]. For a brief overview of this method, refer to the Encyclopedy of Computational Mechanics [46], or to Winckelmans' and Leonard's 1993 article [53].



Figure 1.6: Eulerian space-discretizations for vortex methods



Figure 1.7: Lagrangian space-discretizations for vortex methods

Lagrangian volumes

In some Vortex Particle-Mesh methods as in a 1983 article by Cottet [49], the domain is cut in smaller fluid volumes [49] [74] [34]. The volumes are advected by the flow, which generates both a displacement of their center and a distortion of their shape. Contrary to the vortex blob methods, volumes never overlap. Using volumes allows for conservative methods to be used (see section 2.3.3).

Contour dynamics

The Contour Dynamics Model [75] (introduced by Deem and Zabusky in 1978 [76] [77]) discretizes the spectrum of vorticity values into a finite set of values. The vorticity field is then approximated by a piecewise constant function with each stage having one of the legal vorticity values. The domain is them decomposed in a different regions, one for each vorticity value. Marker particles are placed at the interface between two neighbouring regions and advected following a front tracking approach, where the advection velocity is determined form the jump in vorticity at the interface. However, as for all front tracking approaches, topological changes in the vorticity "layers" are not handled naturally by this method [78].

Vortex sheet, tube, ring methods

Some methods [79] [80] have been specifically designed to study one or few vortex sheet(s), tube(s) or ring(s) and the related phenomena such as Crow instability. Those works usually consider the theoretical study of those very

specific flow structures. In that sense they follow the spirit of earlier works [81] [82] on vortex structures. As a result, the numerical methods employed in those works are not applicable to most flows. For instance, topological changes of fluid interfaces, as it occurs during bubble entrainment phenomena, are not handled naturally by vortex sheet methods.

In that family of method, the most popular appears to be the vortex-sheet methods. For instance Rosenhead's 1931 work considered a Kelvin-Helmholtz instability [39] mixing two fluids of same mass density. Later, Meng studied a rising bubble problem using a vortex sheet method in 1978 [83] and compared a particle-particle method with a particle-mesh method. Also Baker studied a Rayleigh-Taylor instability in 1980 [84] and free-surface flow in 1982 [85]. Finally, Tryggvason studied a Hele-Shaw fingering instability in 1983 [86] and Rayleigh-Taylor instability in 1989 [87] between two fluids of different mass density. Vortex sheets allow to study flows with arbitrary high Atwood numbers with a quite sharp interface discretization. However, topology changes in the bubble's shape e.g. bubble splitting and/or merging is far from trivial.

1.4.3 Extensibility to 3D

The flows within nuclear reactors have 3D flow structures. Although this thesis will only cover 2D flows, the methods investigated need to be easily extendable to 3D flows, which is not the case of all vortex methods. This is due to the fact that vorticity can be stretched in 3D, which does not occur in 2D. While some numerical methods handle that effect with little modification, others require more fundamental changes.

The original vortex blob method does not account for vortex stretching and hence cannot be directly used in 3D. The first solution is to account for the stretching by working with vortex filament segments as Chorin proposed in his 1980 Vortex Segment Method [80]. The height of the filament segments can change in time to account for stretching [80]. Of course the vorticity equation remains deprived of a vortex stretching term [80]. In that article the Vortex Segment Method is used to simulate a boundary layer, which is a much more complex flow than previously simulated with vortex filaments methods. It's convergence is proven by Greengard in 1986 [88].

However Beale and Majda proved in 1982 [89] that the original 2D vortex blob methods with an additional stretching term in the vorticity equation [89] is consistent and can be of arbitrarily-high order in 3D simulations. Compared with Chorin's Vortex Segment Method, the numerical implementation of this method is much simpler. In Beale's 1982 article [89] the stretching term was computed on a Lagrangian grid, but it can also be computed by particle-particle methods as did the same author in 1986 [90]. Particle-mesh methods can easily support the vortex-stretching phenomenon by adding a stretching term in the Right-Hand Side that is computed on the grid. The first 3D Vortex-in-Cell was introduced by Couët in 1981 [42] and proceeded this way.

Those principles layed the basis for modern VPM simulations of threedimensional flows, which now has reached maturity. Cottet and Poncet presented in 2003 [91] a VPM method for the DNS of wall-bounded flows. The previous year, Cottet et al. [92] compared the performance of a VPM method against spectral methods for isotropic turbulence in a periodic box and achieved satisfactory results. In particular it was established that although VPM methods were slightly inferior to spectral methods for very high wavenumbers and twice expensive, especially close to the wall, the VPM method presented a very good resolution of large and intermediate scales as well as a very low dissipation of vorticity and it prevented the accumulation of energy in the tail of the energy spectrum (contrary to the spectral method). A similar study has been conducted by van Rees et al. in 2010 [93] but this time on the interaction between two vortex tubes. It concluded that the VPM method achieved results of similar accuracy than spectral method but at a lower cost.

Regarding fast implementations of the VPM method, Cocle et al. [94] presented a Fast Multipole three-dimensional VPM method in 2007, Chatelain et al. [34] proposed a heavily parallelized implementation of VPM in 2008, and Kosior and Kudela [95] presented in 2012 an implementation of a three-dimensional VPM method on GPUs. Eventually note the three-dimensional simulation of the wake behind a bluff body by Ploumhans and Winckelmans in 2002 [45] which is an extension to 3D of their 2000 article [96].

1.4.4 Arguments in favor of the VPM method

The previous section has already determined that the method shall be a vortex method with a Lagrangian frame-of-reference formulation of the governing equations. Now that an overview of such CFD methods has been given, the choice of VPM as a method of choice is justified in the present section.

Firstly the method must be able to handle generic flows, hence vortex sheet/tube/ring methods, contour-dynamics or boundary-integral methods are not applicable. The point vortices, vortex blob and Lagrangian volume methods are closely related to each other, yet the Lagrangian volume approach is the most effective and modern approach thanks to the conservation properties it offers. Eventually the spectral methods could be a candidate but will not be considered in this thesis.

Also, due to the multiphase nature of the flow, additional terms will be added in the governing equations. The selected method must hence be able to handle those terms naturally. Only the Particle-mesh, Particle Strength Exchange and weight function differentiation (SPH) methods allow this. Moreover, the method must have solid convergence properties, which is not the case of the Smooth Particle Hydrodynamics SPH method [97] [98] [99].

For those reasons the Vortex Particle-Mesh is a good candidate for the target application.

1.5 Modern and/or relevant VPM methods

The VPM method has proved its effectiveness at simulating wakes [100] [101] [34] [102]. Other uses exist such as simulation of animal locomotion [103], geophysical flows [104] or computer rendering [105].

However, little literature could be found for VPM and multiphase flows. The only VPM work directly connected to our study that can be found is from Shakouchi in 2014 [106] and concerns the VPM simulation of a fluid interface separating water and brine that is put in motion and mixed by a jet coming from downwards. The flow studied has a fluid interface and a jump in the fluid properties across it, similarly to the flow problems that are to be instigated in this thesis. However, the mass density of both fluids is fairly similar although not explicitly indicated, and there is no surface tension which makes the problem considered different from ours. Eventually, the Boussinesq approximation is used, leading to a viscous term with constant kinematic and dynamic viscosity, whereas the fluid properties in our case are changing.

It is in fact more usual to see the velocity-pressure formulation used for interfaces. Hence additionally to all the finite differences [107] or finite volumes [108] simulations of multiphase flows (amongst other methods), several SPH simulations of such flows have been performed as well, at least since 2003 [109]. Nevertheless, equivalent works with VPM are extremely scarce.

Most vortex methods for interfaces seem to consider the interface as a vortex sheet, sometimes with Vortex-in-Cell elements in them [110] [87] [83], instead of considering the vorticity in the volume as we aim at doing. We also note the 1997 free-surface one-fluid method by Lundgren and Koumoutsakos [111], yet although there are some similarities, our two-phase approach remains quite different.

More unusual is Fromm's 1984 work [112] which is a variant of the Markerin-Cell method to simulate an inkjet printer ejecting a drop. The problem is expressed in terms of both velocity and vorticity, each of those two variables having a parabolic equation of their own that is time-integrated. Surface tension is implemented through the momentum equation on velocity. On the other hand, and since the viscosity and mass density of the fluid are considered constant, the vorticity equation is oblivious of the presence of the interface, and remains an advection-diffusion equation despite the presence of two phases.

Some work has been performed on Vortex Particle-Mesh methods for weakly variable mass density single flows, based on Anderson's 1985 method [113]. In his 2002 article with Winckelmans [114] (see also his thesis [115]) Thirifay presents a combustion solver with mass density ratio of two and in [103] Gazzola presents a fluid-structure interaction solver with mass density differences of 1.01.

Let us remind also the 2005 article by Hieber and Koumoutsakos [18] that demonstrated the usefulness of particle-mesh methods for capturing ("tracking") fluid interfaces. Also, immersed interface methods for Vortex Particle-Mesh solvers have been published since 2002 [116] [117] [118] [20] [119]. Those immersed interface methods seem to have only been used for the flow around solid objects, not two-phase flows. Note also a VPM method for particle-laden flows by Uchiyama in 2015 [120].

More generally, in terms of fairly recent evolutions or works on VPM methods, one can note a 3D DNS VPM simulation in a wall-bounded domain conducted by Poncet in 2003 [91], VPM simulations for immersed bodies [96], staggered VPM proposed by Uchiyama and al. in 2013 [121] (Fig. 1.8) and VPM for compressible flows from 2002 [122] [123] [38].



(a) Collocated mesh for vortex methods.

All fields are stored in the same place.

(b) Staggered mesh for vortex methods.

Cell corners are used to store vorticity, edges for velocity and centers for velocity potential.

Figure 1.8: Staggered method proposed by Uchiyama [121]

1.6 Research questions

The present work attempts to assess the potential of the VPM method as compared to traditional Eulerian velocity-pressure solvers for the simulation of multiphase flows with large fluid properties differences, both in terms of accuracy and computational performance. Given the existing litterature presented in the previous sections, several research questions arise:

- It has been shown in 2005 by Hieber and Koumoutsakos [18] that a Lagrangian implementation of the level set method is superior to the original Eulerian implementation. Can similar advantages be achieved by using a Lagrangian implementation of a full-fledged flow solver?
- More generally, how do multiphase VPM methods perform in respect to traditional Eulerian velocity-pressure methods, both in terms of accuracy and computational efficiency?
- Only one work has been found that performs a multiphase VPM simulation, namely the 2014 paper by Shakouchi [106]. As noted previously, this work employed the Boussinesq approximation and did not consider surface tension. How and how well can VPM methods handle multiphase flow features, in particular surface tension and variations of fluid properties?
- Numerical simulations of flows with high mass density ratio is notoriously difficult, including with the most widespread methods. Also, no previous work has been found that simulated such flows using the VPM method. How easily can VPM method handle large mass density ratios?

The next chapter introduces the underlying principles of a single-phase Vortex Particle-Mesh method which will serve as a starting point. Each following chapters will then add upon this base until the full-featured solver that can perform simulations of the target study case is obtained: firstly a method to capture the interface location will be developed, then the methods used to discretize the terms of the equation dependent on the multiphase nature of the flow are presented, thirdly a method for handling large mass density differences between phases is described. Eventually, the solver will be validated on benchmarks and some simulations will be performed in cases similar to what could be encountered in the nuclear industry.
Chapter 2

Single-fluid VPM method

In this chapter a simple "advection-diffusion" Vortex Particle-Mesh solver is presented. It will serve as a basis for the later work of this thesis as more functionalities are being added in the following chapters, until the full solver capable of simulating our problem is obtained. The solver considered in this chapter is applicable for constant fluid properties 2D flows in closed (bounded) domains and can take into account merely advection, viscous diffusion and weak buoyancy phenomena. First free-slip boundary conditions are considered, then no-slip boundary conditions will be enforced.

2.1 Governing equations

The classical Navier-Stokes equation is written as

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla) \boldsymbol{u} = - \frac{\nabla p}{\rho} + \frac{1}{\rho} \nabla \cdot \boldsymbol{\tau} + \boldsymbol{g}$$
(2.1)

in an Eulerian frame of reference, where \boldsymbol{u} is the velocity, p the static pressure, μ the dynamic viscosity and ρ the mass density. For the time being, potential simplifications based on the incompressible nature of the flow or constant fluid properties are not considered for the sake of preserving generality. Taking the curl of the previous equation yields the equation for vorticity $\boldsymbol{\omega}$

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\boldsymbol{u} \cdot \nabla)(\boldsymbol{\omega}) = - \frac{\nabla p}{\rho} \times \frac{\nabla \rho}{\rho} + \nabla \times \left(\frac{1}{\rho} \nabla \cdot \boldsymbol{\tau}\right)$$
(2.2)

Note that since we work on 2D problems, vorticity has only one non-zero component $\boldsymbol{\omega} = (0, 0, \omega)$ and simulation can be performed with this scalar quantity directly. In order to time integrate the vorticity, velocity \boldsymbol{u} and pressure p must be known.

In a closed domain, velocity is obtained from vorticity by solving (see ap-

pendix F.1.1)

Find stream function
$$\psi$$
 such that:
$$\begin{cases} \nabla^2 \psi = -\omega & \text{on } \Omega\\ \psi = 0 & \text{on } \partial\Omega \end{cases},$$
(2.3a)

Get velocity \boldsymbol{u} from stream function $\psi: \boldsymbol{u} = \nabla \times \psi \, \boldsymbol{\hat{e}}_z.$ (2.3b)

On the other hand pressure is obtained through the momentum equation (Eq. 2.1), which yields

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\boldsymbol{u} \cdot \nabla) \boldsymbol{\omega} = \left(\frac{D\boldsymbol{u}}{Dt} - \boldsymbol{g}\right) \times \frac{\nabla \rho}{\rho} + \frac{1}{\rho} \nabla \times (\nabla \cdot \boldsymbol{\tau})$$
(2.4)

Notice that the viscous terms of both the velocity (Eq. 2.1) and original vorticity (Eq. 2.2) equations are merged into a single term.

In this chapter, the continuity and energy equations will not be considered.

2.2 Some important conventions and definitions

Because of the VPM method mixing Lagrangian and Eulerian space-discretizations, clear definitions must be given regarding frames of references and systems of coordinates.

2.2.1 Systems of coordinates

Lagrangian particles are marker points that are advected by the flow. They are characterized by their original location X at the initial condition and the lapse of time $t - t_{ini}$ during which they have been carried around by the flow. At any instant of time t the location of any particle X is given by the particle position function $\chi(X, t)$. In particular we have

$$X = \chi(X, 0). \tag{2.5}$$

The Vortex Particle-Mesh method solves the governing equations in their Lagrangian form. However those equations are discretized both on an Eulerian grid and on Lagrangian particles. Eulerian coordinates (\boldsymbol{x}, t) are used when working on the grid, while Lagrangian coordinates (\boldsymbol{X}, t) are used when working with particles.

$$(\boldsymbol{x},t) = (\boldsymbol{\chi}(\boldsymbol{X},t),t) \tag{2.6a}$$

$$(X,t) = (\chi^{-1}(\boldsymbol{x},t),t)$$
 (2.6b)

Moreover, since there will often be transformations between Eulerian and Lagrangian coordinates, it is critical to differentiate the functions q taking Eulerian coordinates as input $q(\boldsymbol{x},t)$, to the functions \tilde{q} taking Lagrangian coordinates as input $\tilde{q}(X,t)$. Both functions are connected by the particle position function $\chi(\cdot)$

$$q(\chi(X,t),t) = \tilde{q}(X,t) \tag{2.7}$$

Eventually note that in the same way that Eulerian equations are typically written analytically on a continuous space before being discretized using the Method of Lines, the same can be done of the Lagrangian form of the equation. Hence before discretization there is an infinite number of "particles", a given particle X_p being merely one of them, in the same way as there are an infinite number of points in the flow domain Ω before grid discretization reduces it to a finite set of nodes $\boldsymbol{x}_{i,j}$.

2.2.2 Frames of reference

An important property that will be used in the rest of the thesis is that a transport equation expressed in the Eulerian frame of reference

$$\frac{\partial q}{\partial t} + (\boldsymbol{u} \cdot \nabla)(q) = Rhs(q)$$
(2.8)

sees its advective term vanish when expressed in the Lagrangian frame of reference

$$\frac{d\tilde{q}}{dt} = Rhs(\tilde{q}) \tag{2.9}$$

while the Right-Hand-Side operator $Rhs(\cdot)$ remains unaltered. In particular, consider the level set advection equation (Eq. 3.6) : once expressed in the Lagrangian frame of reference, it becomes

$$\frac{\partial \tilde{\phi}}{\partial t} = 0 \tag{2.10}$$

In other words : the Lagrangian level set function $\tilde{\phi}$ is constant. This is the cornerstone of any Lagrangian method, including the VPM: by removing the advective term, the numerical errors it generated as well as the CFL constraint on time step it imposed both vanish. In the case of highly-advective flows, the most stringent stability constraint is typically the CFL constraint. Hence by simulating such flows with a Lagrangian method, one can choose larger time steps values than with a purely Eulerian method.

2.2.3 Discretization of space

Eulerian space-discretized methods use a grid $\Omega_{\mathbf{h}}$ that is a collection of nodes $\boldsymbol{x}_{i,j}$. On the other hand a Lagrangian space-discretized method works with a finite number of individual particles indexed $p \in [0, |\Omega_{\mathbf{h}}|[$ whose position \boldsymbol{X}_p is updated in time by time-integrating the Lagrangian velocity $\tilde{\boldsymbol{u}}$ since

$$\frac{d\boldsymbol{X}_p}{dt} = \tilde{\boldsymbol{u}}_p \tag{2.11}$$

2.3 Numerical method

2.3.1 Discretization of space

Lagrangian volumes

The VPM method cuts the domain into "Lagrangian volumes" of fluid V_p . Initially, they correspond to square-shaped volumes of equal size, centered in each grid node $\boldsymbol{x}_{i,j}$, and of side length equal to a grid spacing h_x . As time goes, the V_p volumes are advected by the flow and see their shape being distorted, although their volume remain constant in solenoïdal flow fields (Fig. 2.1). At all times the union of all V_p volumes hence correspond to the domain Ω (to within a null set) and they never overlap each other.

More formally, a Lagrangian volume centered around particle X_c is defined as the collection of particles initially located within a square that is centered in X_c , and has a radius half a grid spacing $h_x/2$.

$$V_p(X_c,t) := \left\{ \chi(X,t) : \|X - X_c\|_{\infty} < \frac{h_x}{2} \right\}$$
 (2.12)



Figure 2.1: Advection of 16 Lagrangian volumes in the analytical case.

Numerically however, the volumes' shape is not stored, only their center is. In order to achieve a simple graphical view of the Lagrangian distortion this causes, it will be assumed that the volumes remain square-shaped (Fig. 2.2). After a certain number of time steps however, this assumption is no longer reasonable. In particular, volumes overlap in parts of the domain while other regions remain uncovered (Fig. 2.2c). Additionally, particles might have scattered from certain regions of the domain, while other might be clustering elsewhere (Fig. 2.3b). It then becomes necessary to "remesh" the particles, that is to put them back into their original upright square lattice formation. This is done by (i) interpolating the particles' carriage to the grid, (ii) re-positioning the particles into an upright square lattice and (iii) interpolating the quantities from the grid back onto the particles.

Eventually, let us stress out that although the rest of this document will hold more occurrences of the word "particle" than "Lagrangian volume", the atomic discretization element of the VPM method is indeed a volume, not a



Figure 2.2: Advection of 16 Lagrangian volumes in the discretized case.

particle. However, since the only information of a Lagrangian volume that is stored and managed by the computer is its center, it is numerically handled as a particle.

Lagrangian distortion

The longer in time the particles are advected, the more distorted the particle lattice will be. Therefore, there is a maximum time step value beyond which a single time step would lead to a particle lattice that is too distorted. Note however that this critical time step value is typically much greater than the time step limit imposed by a CFL constraint. That distortion is caused by two effects : (i) the particles are "slinged away" by the rotative motion of the flow which tends to pull them away from centers of rotation (Fig. 2.3a), and (ii) the shear and compressive¹ motions of the flow alters the shape of the V_p volumes (Fig. 2.1c) (Fig. 2.2c).



Figure 2.3: Particle slinging and its main consequence: particle clustering. On the left subfigure, due to a circular motion of the flow around rotation center (\bullet), particle (\bullet) should follow trajectory (---) and hence move to (\bigcirc). However, over an (Euler Explicit) time step the local velocity (--) brings it to (O) instead.

 $^{^1\}mathrm{Since}$ we work on an incompressible flow, only numerical errors may compress the flow.

The magnitude of the first phenomenon is measured by the Lagrangian CFL (LCFL) on rotation

$$LCFL_{rot} := \max_{\boldsymbol{x}_{i,j} \in \Omega_{\boldsymbol{h}}} \left(|\omega_{i,j} h_t| \right)$$
 (2.13)

while the second type of distortion is measured by the LCFL on shear

$$LCFL_{shear} := \max_{\boldsymbol{x}_{i,j} \in \Omega_{\boldsymbol{h}}} \left(\| \boldsymbol{S}(\boldsymbol{x}_{i,j}) \| h_t \right)$$
(2.14)

where \boldsymbol{S} is the strain tensor and h_t is the time step value. The $LCFL_{shear}$ is calculated using the ℓ^1 norm². Eventually, for ease of use we define the "global" Lagrangian CFL as

$$LCFL := \max(LCFL_{rot}, LCFL_{shear})$$
 (2.15)

Note that both $LCFL_{rot}$ and $LCFL_{shear}$ are measures for the rate at which distortions are made, and do not indicate directly how distorted the flow is. Note also that the amplitude of the error on particle motion ("particle-slinging") depends a lot on the time integrator used. It is particularly strong (Fig. 2.3a) for the Euler Explicit scheme, and hence other time integration schemes are prefered instead such as the Runge-Kutta 2 midpoint scheme (see appendix D.2) or the low storage Runge-Kutta 3 scheme described in Appendix D.4.

2.3.2 General algorithm of the VPM method

Unlike a grid, particles are not necessarily disposed in an upright square lattice. Hence in particle methods, finite differences cannot be used directly to evaluate the Right-Hand-Side (RHS). Several methods exist to alleviate this issue, the VPM method being merely one of them. The spirit behind it is to interpolate particle data on the grid so that the right hand side can be evaluated.

The corresponding algorithm for an Euler Explicit time integrator is given in Algorithm 1. As can be seen, the vorticity information is firstly interpolated from particles to grid (Algo. 1-1) using the P2M ("Particle-to-Mesh") subroutine. Secondly, calculations are performed on the grid so as to obtain velocity \boldsymbol{u} (Algo. 1-2) and the time derivative of vorticity $\frac{\partial \omega}{\partial t}$ (Algo. 1-3) on the grid. Then, both fields are interpolated back to the particles (Algo. 1-4) (using the "Meshto-Particles" M2P sub-routine) where the integration is performed (Algo. 1-5).

The following section explains in greater details how the interpolation works.

2.3.3 Interpolating between particles and grid

Interpolation between particles and mesh is one of the core elements of a VPM method. Firstly, the interpolation from mesh to particle is presented, then the reverse interpolation is described. Eventually, more details are given on the interpolating kernels.

²Ideally the ℓ^2 norm should be used, however it is difficult to compute. On the other hand, the ℓ^1 norm is much easier to evaluate and is always greater than it: in other words it safely over-estimates the distortion.

Algorithm 1: General VPM algorithm (Euler Explicit time integration)

- 1 Interpolate from particles to grid $\omega \leftarrow P2M(\mathbf{X}_{p}^{n}, \omega_{p}^{n})$
- 2 Compute velocity from vorticity Find ψ such that : $\nabla^2 \psi = -\omega$ $\mathbf{u} \leftarrow \nabla \times (\psi \, \hat{\boldsymbol{e}}_z)$
- $\begin{array}{c} \textbf{s} \ \text{Compute RHS on grid} \\ \frac{\texttt{D}\boldsymbol{\omega}}{\texttt{D}\texttt{t}} \leftarrow \texttt{Rhs}\left(\boldsymbol{\omega}, \textbf{u}, \texttt{t}\right) \end{array}$
- $\begin{array}{l} \textbf{4} \ \text{Interpolate from grid to particles} \\ \left\{ \begin{array}{c} \frac{D\boldsymbol{\omega}}{Dt} \Big|_p \\ \textbf{u}_p \end{array} \right\} \leftarrow \texttt{M2P} \left(\textbf{X}_{p}, \left\{ \begin{array}{c} \frac{D\boldsymbol{\omega}}{Dt} \\ \textbf{u} \end{array} \right\} \right) \end{array}$
- 5 Integrate particles' position and vorticity on the particles
 - $\begin{cases} \boldsymbol{\omega}_p^{n+1} \\ \boldsymbol{X}_p^{n+1} \end{cases} \leftarrow \begin{cases} \boldsymbol{\omega}_p^n \\ \boldsymbol{X}_p^n \end{cases} + \mathbf{h}_t \begin{cases} \frac{D\boldsymbol{\omega}}{Dt} \big|_p \\ \mathbf{u}_p \end{cases}$

Interpolation from mesh to particles

In order to interpolate quantity q from the grid $\Omega_{\pmb{h}}$ to a particle p, an interpolation scheme is used

$$\alpha[\tilde{q}](\boldsymbol{X}_p) = \sum_{\boldsymbol{x}_{i,j} \in \Omega_{\boldsymbol{h}}} \alpha[q](\boldsymbol{x}_{i,j}) \ W_h(\boldsymbol{x}_{i,j} - \boldsymbol{X}_p)$$
(2.16)

where W_h is an interpolation kernel of moment 1 at least, and the intensity $\alpha[q]$ of field q is defined as the integral over the Lagrangian volume

$$\alpha[q](\boldsymbol{X}_p, t) := \int_{V_p(\boldsymbol{X}_p, t)} q \, dx \tag{2.17}$$

The reason why this scheme interpolates an integral of vorticity instead of pointwise vorticity, is that we want the vorticity to be conserved through interpolation. Indeed, since the interpolation kernel has a moment greater or equal to one, the integral of vorticity remains unchanged through interpolation and hence vorticity is conserved.

Interpolation from particles to mesh

The procedure is a little more delicate when interpolating from particles to grid since interpolation schemes such as the M'4 scheme require their inputs to be expressed at regularly spaced points, which is not the case of the particles in the general case. Instead, each particle is considered one by one, and for each of them a square lattice of particles carrying value zero is created. Since the nodes in that lattice are regularly spaced, the interpolation scheme can now be used. Practically speaking, for each individual particle p, the value it carries is

interpolated individually to each node $\boldsymbol{x}_{i,j}$ of the grid $\Omega_{\boldsymbol{h}}$

$$(\alpha_p)_{i,j} = \tilde{\alpha}_p W_h(\boldsymbol{X}_p - \boldsymbol{x}_{i,j}).$$
(2.18)

where $\alpha_p(\cdot)$ is the interpolated field generated by a single particle p (and its cloud of null particles). Hence, the (total) interpolated field $\alpha(\cdot)$ is the sum of the individual interpolations of each particle \boldsymbol{X}_p within the cloud of particles $\widetilde{\Omega}_h$

$$\alpha_{i,j} = \sum_{\boldsymbol{X}_p \in \widetilde{\Omega}_h} (\alpha_p)_{i,j} = \sum_{\boldsymbol{X}_p \in \widetilde{\Omega}_h} \widetilde{\alpha}_p \ W_h(\boldsymbol{X}_p - \boldsymbol{x}_{i,j})$$
(2.19)

Interpolation kernels

The interpolation kernel is chosen to be the product of two one-dimensional kernels

$$W_h(\boldsymbol{x}) := w\left(\frac{|\boldsymbol{x} \cdot \hat{\boldsymbol{e}}_x|}{h_x}\right) w\left(\frac{|\boldsymbol{x} \cdot \hat{\boldsymbol{e}}_y|}{h_y}\right)$$
(2.20)

where $h_x = h_y$ are the grid's spacing, and $w(\cdot)$ is the weighting function.

Several criteria guide the choice of a kernel, mainly: moment order, continuity of the n^{th} first derivatives, order of accuracy of the interpolation error and strength of numerical diffusion. In one dimension, a kernel W_h of moment k is a kernel such that

$$\sum_{k=-\infty}^{+\infty} W_h(x+i\,h) = 1 \ , \ \forall x \in [0,h[$$
(2.21)

and

$$\sum_{i=-\infty}^{+\infty} (x+ih)^n W_h(x+ih) = 0 , \ \forall x \in [0,h[$$
(2.22)

for all $n \in [1, k - 1]$.

One of the simplest choices for a kernel would be a witch-hat function. Historically, that was what the Cloud-in-Cell (CIC) method consisted in [47]. The hat (piecewise constant) interpolant has also been used under the name of Nearest Grid Point (NGP) interpolant [124]. However such kernel yields interpolated fields that are not smooth enough, even when interpolating from a smooth field. This causes issues when finite differences are computed on the outputted interpolated field. Ideally we pursue a kernel of class C^1 at least.

On the other hand, amongst all possible functions, polynomials are very attractive thanks to their lower computational cost. Hence B-Splines are a natural choice [125]. In particular the cubic spline kernel has an order of accuracy of two, a moment of two and is C^2 . It is defined as

$$w(\zeta) = \begin{cases} \frac{1}{6} \left(4 - 6\,\zeta^2 + 3\,\zeta^3\right) &, \text{ if } 0 \le \zeta < 1\\ \frac{1}{6} \left(2 - \zeta\right)^3 &, \text{ if } 1 \le \zeta < 2\\ 0 &, \text{ if } 2 < \zeta \end{cases}$$
(2.23)

Moreover, in his 1985 work, Monaghan [126] has performed a Richardson extrapolation on those B-Splines in order to improve the order of accuracy of

the interpolation by one at the cost of the order of the interpolation function class being lowered by one. This leads in particular to the widely used M'4 kernel that is a good compromise between previously indicated criteria: it is of class C^1 , third order of accuracy and moment of order three. It is defined as

$$w(\zeta) = \begin{cases} 1 - \frac{5}{2}\zeta^2 + \frac{3}{2}\zeta^3 & , \text{ if } 0 \le \zeta < 1\\ \frac{1}{2}(2-\zeta)^2(1-\zeta) & , \text{ if } 1 \le \zeta < 2\\ 0 & , \text{ if } 2 < \zeta \end{cases}$$
(2.24)

In the solver of the current chapter and in the rest of the thesis, the M'4 kernel will be used unless indicated otherwise. The profiles of each of the four weighting function discussed in the present section are shown in Figure 2.4, and some of their general properties are shown in Table 2.1.



Figure 2.4: Some interpolation kernels

	Order of Differentiability		Moment	Dogitivo
	$convergence^3$	class	moment	1 OSITIVE
Witchhat (M_2)	0	0	2	Yes
Quadratic Spline (M_3)	1	1	3	Yes
Cubic Spline (M_4)	2	2	4	Yes
Richardson-extrapolated M_4 (M'_4)	3	3	3	No

Table 2.1: Some examples of interpolation kernels

 $^{^3\}mathrm{Note}$ that second-order central differences will be used, hence the global accuracy of the method is limited to second-order.



Figure 2.5: Correction of the free-slip vorticity field to obtain a no-slip vorticity field. The wall is visible on the left of the picture (\square), as well as the velocity profile (\rightarrow) and the vorticity field (\square).

2.4 Enforcing boundary conditions

2.4.1 Free-slip boundary conditions

Free-slip boundary conditions are characterized by a velocity field of normal component u_n and tangential component u_t observing no through flow

$$u_n = 0 \text{ on } \partial\Omega \tag{2.25}$$

and no shear at the boundary

$$\nabla u_t \cdot \hat{\boldsymbol{n}} = 0 \text{ on } \partial \Omega \tag{2.26}$$

Such boundary condition is naturally imposed in vortex methods by enforcing (e.g. homogeneous) Dirichlet boundary conditions on the stream-function ψ (Eq. 2.3a)

$$\psi = 0 \text{ on } \partial\Omega, \tag{2.27}$$

which yields no through flow, and homogeneous Dirichlet boundary conditions on the vorticity

$$\omega = 0 \text{ on } \partial\Omega, \tag{2.28}$$

which yields no shear flow.

2.4.2 No-slip boundary conditions

On the other hand, no-slip boundary conditions are characterized by a velocity field observing no through flow

$$u_n = 0 \text{ on } \partial\Omega \tag{2.29}$$

and zero tangential velocity at the wall

$$u_t = 0 \text{ on } \partial\Omega. \tag{2.30}$$

However, there is no simple way to express this constraint in terms of vorticity. A solution presented in Koumoutsakos et al's 1994 article [127] or Pépin's 1990

PhD thesis [128], amongst others, consists in first integrating a time step using a free-slip formulation, then correcting the resulting vorticity field knowing the wall slip velocity (Algo. 2). In other words, the no-slip vorticity field ω is decomposed into a free-slip vorticity field ω_{slip} and a correcting vorticity field ω_{corr}

$$\omega = \omega_{slip} + \omega_{corr}.\tag{2.31}$$

This is illustrated by Figure 2.5 and Algorithm 2: firstly the free-slip problem is solved over a time-step (Algo. 2-1), which yields the slip vorticity field ω_{slip} . The velocity field \boldsymbol{u}_{slip} corresponding to this vorticity field (Algo. 2-2) is nonzero at the wall (Fig. 2.5a). In order to cancel out the latter, the correcting vorticity field ω_{corr} must correspond to a vortex sheet $d\Omega$ of intensity $d\gamma$ such that it generates a wall-slip velocity of equal amplitude but opposite direction (Fig. 2.5b) (Algo. 2-3). By summing both fields one obtains a no-slip vorticity field ω , whose corresponding velocity field in indeed equal to zero at the wall (Fig. 2.5c) (Algo. 2-4).

Algorithm 2: Enforcing no-slip boundary conditions in vorticity

1 Time-integrate the free-slip problem over one time-step $\omega_{\texttt{slip}}^n \gets \omega^n$

$$\omega_{\texttt{slip}}^{\texttt{n+1}} \leftarrow \omega_{\texttt{slip}}^{\texttt{n}} + \int_{\texttt{t}^{\texttt{n}}}^{\texttt{t}^{\texttt{n+1}}} \frac{\partial \omega_{\texttt{slip}}}{\partial \texttt{t}} \, \texttt{dt}$$

- 2 Obtain wall slip velocity \mathbf{u}_{slip} from free-slip vorticity $\boldsymbol{\omega}_{\text{slip}}$ $\mathbf{u}_{\text{slip}}^{n+1} \leftarrow \boldsymbol{\omega} 2\boldsymbol{u}(\boldsymbol{\omega}_{\text{slip}}^{n+1})$
- s Compute correcting vorticity $\omega_{\tt corr}$ from slip velocity $\boldsymbol{u}_{\tt slip}$ by solving problem 2.32

Find ω_{corr}^{n+1} such that $\omega 2u(\omega_{corr}^{n+1}) = -\mathbf{u}_{slip}^{n+1}$ on $\partial \Omega$

Mathematical problem for correction vorticity

Unlike the problem for free-slip vorticity, the problem for correction vorticity requires shear at the wall. More specifically, since the flow is motionless at the wall, the velocity on both sides of the vortex sheet must be of same amplitude. Knowing the intensity of the vortex sheet, the problem for correction vorticity describes its viscous diffusion on the domain Ω over the timespan of the time step.

Assuming that the diffusive flux of vorticity across the boundary caused by wall friction is constant in time within each time step, the correcting vorticity



Figure 2.6: On each boundary element, the free-slip velocity field (\longrightarrow) must be cancelled by a vortex sheet ($\bullet \bullet \bullet$). The velocity ($\rightarrow \bullet$) induced by the vortex sheet must be of same amplitude but opposite direction compared to the wall-slip velocity ($\rightarrow \bullet$).

 ω_{corr}^n can be taken as the solution of the problem

$$\frac{\partial \omega_{corr}}{\partial t} = \nu \, \nabla^2 \omega_{corr} \qquad \text{on } \Omega \qquad (2.32a)$$

$$\omega_{corr} = 0 \qquad \text{at } t = 0 \qquad (2.32b)$$

$$\nabla \omega_{corr} \cdot \hat{\boldsymbol{n}} = -\frac{u_{slip,t}^{n}}{\nu h_{t}} \qquad \text{on } \partial \Omega \qquad (2.32c)$$

at time $t = h_t$.

Numerically, the problem 2.32 is solved using the method suggested by Ploumhans and Winckelmans in [96] which is based on a boundary-element method. The domain boundary is discretized as a collection of panels (boundaryelements) of length h_x such that each edge of a grid cell overlapping the domain boundary is a panel. Each panel is then considered to be a vortex sheet of intensity $-2\boldsymbol{u}_{slip}\cdot\boldsymbol{\hat{t}}h_x$, where $\boldsymbol{\hat{t}}$ is the tangent vector to the wall (Fig. 2.6). The slip-velocity (and hence the vorticity diffusive flux $\nabla \omega \cdot \boldsymbol{\hat{n}}$) is assumed constant in space along it. According to Ploumhans and Winckelmans [96], the correcting vorticity field generated by a single panel of the bottom wall centered in zero is

$$\omega_{corr} = -\frac{u_{slip,t}\sqrt{\nu}}{2h_x^2h_t} \int_0^{h_t} \left(\sqrt{t} \left[erfc(s)\right]_{s=g\left(y-\frac{h_x}{2},t\right)}^{s=g\left(y+\frac{h_x}{2},t\right)} \left(\left[ierfc(s)\right]_0^{s=g\left(h_x,t\right)} -\left[ierfc(s)\right]_{s=g\left(-h_x,t\right)}^0\right)\right) dt$$
(2.33)

where function $g(\cdot, \cdot)$ is defined as

$$g(x,t) = \frac{x}{\sqrt{4\nu t}} \tag{2.34}$$

Eventually the contributions of every panel is summed up to obtain the correcting vorticity field ω_{corr} .

2.5 Summary

A traditional VPM numerical method has been presented which can be used for the simulation of monophasic flows with free-slip or no-slip boundary conditions. The next chapters will add features to that solver, starting in the following chapter with the ability to capture the fluid interface's position.

Chapter 3

Interface capturing VPM method

We now consider a two-phase flows where the flow domain Ω is split into an interior region Ω^- holding one fluid phase and an exterior region Ω^+ containing the other. Both regions are separated by a fluid interface Γ (Fig. 3.1a).

The Navier-Stokes equations embed fluid properties such as mass density or dynamic viscosity. For instance, a typical momentum equation is written as

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla) \boldsymbol{u} = \frac{1}{\rho} \nabla \cdot (2 \,\mu \,\boldsymbol{S}) \tag{3.1}$$

In a two-phase flow, the properties of both fluids are often different. Hence, one must know in which fluid one is located so as to select the proper fluid properties. Additionally, if the flow is subject to capillary action, the surface tension force will act only on the interface and its strength is proportional to the local curvature of the interface. Therefore computing the surface tension term requires to know where the interface is located and its shape. Several numerical methods allow to update and retrieve such information about a fluid interface.

Before tackling jumps in fluid properties across the fluid interface, the solver must be able to handle two phases of identical properties. This is the topic covered by the present chapter. First an overview of existing interface capturing/tracking methods is given, and the choice of the level set method is justified in regards to the requirements of the flows to be simulated. Then the algorithm of the VPM method is modified to accommodate this method.

3.1 Selecting an interface capturing method

Multiphase CFD is anterior to electronic computers [39]. Yet with the advent of electronic computers and the subsequent strong interest in CFD, several radically different interface tracking/capturing paradigms have been devised up until the nineties. From there on no new approach has emerged, yet former methods were consolidated and hybrids of existing methods were developed.



Figure 3.1: The interface as perceived by the computer

In this chapter we describe the different families of interface capturing/tracking methods as well as the more recent developments on the matter. Figure 3.1 gives a graphical overview of the families of methods considered. Then the pros and cons of each method is given. Eventually one of those methods is chosen for our solver and that choice is motivated by the features of the flows that are to be simulated.

3.1.1 Front tracking

The most naive approach consists in seeding marker particles at the interface location (Fig. 3.1b). Those marker particles are advected with the flow to keep the interface location information up-to-date through time. This is referred to as *front tracking*. The earliest use of such a method is Rosenhead's 1931 work [39]. Unverdi and Tryggvason also use that approach in his 1992 article [31].

However, it has a major disadvantage : handling topological changes such as splitting or merging of bubbles is far from easy. As an example, Figure 3.2 shows two bubbles that come closer to each other and merge. Originally, marker particles are placed around both bubbles (Fig. 3.2a). Then the bubbles happen to be pushed towards one another by the flow until they overlap (Fig. 3.2b). This of course is not physical: both bubbles should merge to become a single bubble (Fig. 3.2d). To achieve this result, the computer must be capable of detecting the overlap, delete the unnecessary marker particles and add particles at the junction between bubbles (Fig. 3.2c). Hence the merging of two bubbles even-though it appears to be a very natural phenomenon, is in fact difficult to program on a computer, although feasible. More generally the difficulty for front tracking methods to handle topological changes has motivated the



(a) Initial condition: two distinct bubbles



(c) The program detects the overlap and deletes the nodes (\bigcirc) that are no longer required...



(b) The flow happens to push both bubbles towards each other until they overlap.



(d) . . . then adds nodes (${\ensuremath{\, \bullet \,}}$) at the intersection points.

Figure 3.2: The main challenge of the front tracking method is to handle topological changes. Note that steps (c) and (d) involve connectivity information (i.e. knowing which are the neighbours of any node), which is not always straightforward in 2D, and much more difficult in 3D.

development of alternative methods that can handle topological changes more naturally. All those other methods handle the interface in a more indirect way and are therefore described as *interface capturing methods* as opposed to the *interface tracking method*. On the other hand, front tracking methods are known to advect the fluid interface very accurately and are still being used as of today for that reason.

3.1.2 Particle methods

Particle methods (Fig. 3.1c) attempt at addressing the shortcomings of the front tracking method, while reusing the concept of particles advected by the flow. Unlike the front tracking method, particles are seeded within one or both fluid phases instead of being placed on the interface itself. In the same way as front tracking, those particles are advected with the flow using the local velocity field.

The advantage of particle methods in comparison to the front tracking approach is that connectivity information (which particles are the neighbours of each particle) is not required, and hence topological changes are handled more naturally.

Historically, particle methods are one of the earliest methods used to capture a fluid interface with Harlow's 1957 Particle-in-Cell (PIC) method [32] (first unpublished work in 1955 [129]). However, it appears to have lost most of its popularity and is no longer very much used except for Smooth Particle Hydrodynamics (SPH) simulations [130].

3.1.3 Level set

The *level set* method (Fig. 3.1d) introduced by Osher and Sethian in 1988 [131] uses a radically different paradigm. Indeed it defines a so-called level set field that indicates for any given point \boldsymbol{x} both its distance to the interface Γ and within which phase it is located.

The advantage of the level set method is that it allows to compute very easily geometric information of the interface. For instance, the normal is equal to the gradient of the level set field and the curvature is its laplacian. Nevertheless, the level set method tends to alter the volume of each phase in the course of the simulation. This occurs because the level set field itself does not hold an information on the volume of either phases. Therefore, when the level set field is updated at each time step to account for the motion of the interface, it is not possible to numerically enforce volume preservation, and the interface might be moved a little bit too far in some places or not far enough in other places, resulting in a net gain or loss of phase volume.

For comprehensive and detailed explanations of the subject see Sethian's book [132] or Osher's book [133], for an overview of traditional level set methods see the reviews by Osher and Sethian [134] [135], for an historical point of view see Sethian's PhD thesis [136] this article by the same author [137] or the vulgarization article by Sethian [138] in the American Scientist.

3.1.4 Volume of fluid

Perhaps the most widespread interface capturing method is the Volume of Fluid (VOF) method (Fig. 3.1e). Indeed many commercial CFD software propose VOF as the default or even the only method for simulating multiphase flows. In VOF, each cell in the domain holds a volume fraction value : if f = 0, then the cell is full of fluid #1, if f = 1 it is filled with fluid #2, lastly if 0 < f < 1, the cell is filled with a certain proportion of both fluids. Hence the interface crosses all cells whose volume fraction is between 0 and 1, but its shape is not specified directly.

The volume fraction field is kept updated by solving an advection equation using a finite volume scheme, which guarantees exact conservation of volume fraction: no amount of fluid #1 or #2 can vanish or appear during the simulation.

Whereas the level set field holds geometrical information but no volume information, the VOF does exactly the opposite: it holds a volume information but no geometrical information. As a consequence, it is very good at conserving volume, however computing normals or curvature is more difficult. If the evaluation of such quantities is required by the governing equations (for instance because they embed a surface tension term), then the interface must be *reconstructed* from the volume fraction information. Several reconstruction methods exist including PLIC¹ [139], PROST² [140] or THINC³ [141] [142] etc. Moreover, for each reconstruction method, several numerical implementations

¹Piecewise Linear Interface Calculation.

²Parabolic Reconstruction Of Surface Tension.

³Tangent of Hyperbola for INterface Capturing.

of that method exist: for instance PLIC can be implemented using Young's method or $LVIRA^4$ [143] amongst others.

3.1.5 Hybrid methods

Each method has advantages and weaknesses of its own. Therefore, selection of the method strongly depends on the type of problem that is being solved. A problem which emphasizes mass conservation will prefer using the VOF method, while a problem requiring accurate computation of the surface tension force will prefer a level set method, etc. In an effort to combine the strength of each individual method, methods combining several "pure" methods have been developed. Such methods are referred to as *hybrid methods*.

For instance the Coupled Level Set/Volume-Of-Fluid (CLSVOF) method introduced by Sussman and Puckett in 2000 [144] combines a VOF method that naturally conserves mass with a level set method that allows to compute curvature more naturally.

Another example is the particle level set method introduced by Enright, Fedkiw et al. in 2002 [145]. It was designed as an alternative to Sussman's CLSVOF that aims at a more accurate computation of curvature than what can be achieved with VOF or CLSVOF methods.

The main drawback of hybrid methods is their complexity as compared to "pure" methods. This implies greater implementation efforts of course, as well as those methods being more expensive in terms of memory and/or CPU usage, and them being more complex.

3.1.6 Recent evolutions

In the last decade, the two most popular methods were the VOF method, used for problem where volume preservation is important, and the level set method preferred for cases where computing geometric information is critical. Authors have tried to address those weaknesses in a quest for the ultimate method : the one which would both conserve volume and at the same time allow easy computation of geometric information.

On one hand VOF users have investigated other ways to reconstruct the interface or new implementations of existing reconstruction paradigms such as WLIC⁵ [146] or 3D-THINC [147], leading to some improvements while not changing the core of the VOF method itself.

On the other hand some authors have attempted to devise a conservative level set method. One of the early works in that direction was made by Olsson and Kreiss in 2005 [148] (see also Marchandise et al's 2007 work [149] for a spectral conservative level set method). This work showed that contrary to a widespread belief level set methods could possibly be conservative. From that point on several authors have proposed improvements to that method eventually leading to Chiodi and Desjardins' 2017 solution [150].

 $^{^4\}mathrm{Least}$ square Volume-of-fluid Interface Reconstruction Algorithm.

 $^{^5 \}rm W eighted$ Line Interface Calculation.

It is important to note that conservative level set methods are very new and not yet very mature, although results are very encouraging. Therefore special caution must be taken. In particular, before Chiodi and Desjardins' 2017 work [150] several authors, such as Olsson Kreiss & Zahedi in 2007 [151], Waclawczyk in 2015 [152] or Desjardins himself with Pitsch in 2008 [153], have been proposing conservative level set methods that were flawed. In particular, several methods tend to artificially generate oscillations at the fluid interface which can in some case result in the creation of trailling bubbles that do not exist in the physical flow.

Additionally, even-though particle methods seem to have lost in attractiveness, a particle-mesh discretization of the "pure" level set method by Hieber and Koumoutsakos in 2005 [18] borrows some advantages of the particle method in terms of advection accuracy, while keeping the power and simplicity of the classic level set method. This approach will be used in the present work.

3.1.7 Chosen method

The ideal interface tracking/capturing method (i) advects the interface accurately, (ii) handles topological changes robustly and easily, (iii) allows the computation of curvature accurately and with little effort, (iv) conserves the volume of each phase. Unfortunately, such an ideal method does not exist... Instead each has strengths and weaknesses of its own as is summarized in Table 3.1. Therefore one needs to consider which method is most appropriate to our case.

In our target application (see section 1.1), the accurate computation of surface tension effects is critical to us, while volume conservation not so much. Indeed it is extremely important to accurately model the geometry of the meniscus formed at the line of contact between liquid metal, argon atmosphere and the walls of the reactor's pool and internals. Additionally we want our method to be able to simulate most flows occurring in a nuclear reactor, including bubble entrainment which involves topological changes. Eventually we want our method to be simple and elegant and hence robust. Therefore hybrid methods will not be used. Hence the level set method is the most attractive method in our case.

3.2 Level set method

In the level set method (Fig. 3.1d), the interface's position is captured by using a signed distance function $\phi(\cdot)$. At any point \boldsymbol{x} in space, the absolute value of $\phi(\boldsymbol{x})$ indicates the shortest distance separating \boldsymbol{x} from the interface

$$|\phi(\boldsymbol{x})| = \min_{\boldsymbol{x}_{\Gamma} \in \Gamma} \|\boldsymbol{x} - \boldsymbol{x}_{\Gamma}\|$$
(3.2)

		Interface tracking	Interface capturing		Hybrid methods		Emerging methods	
	Relevance to study case	Front tracking	Level set	VOF	Particles	CLSVOF	Particle level set	Conservative level set
Handling topological changes	+++	+	+++	+++	+++	+++	+++	+++
Computing curvature	+++	+	+++	+	+	++	+++	+++
Conserving volume	+	+	+	+++	++	+++	+++	+++
Advecting interface	+++	+++	++	++	+++	++	++	++
Simplicity	N/A	+	+++	+++	+++	+	+	++
Maturity	N/A	+++	+++	+++	+++	++	++	+

Table 3.1: Capabilities of interface tracking/capturing methods

and its sign indicates whether \boldsymbol{x} is within one fluid or the other

$$\phi(\boldsymbol{x}) < 0 \quad \Rightarrow \quad \boldsymbol{x} \in \Omega^{-} \tag{3.3a}$$

$$\phi(\boldsymbol{x}) > 0 \quad \Rightarrow \quad \boldsymbol{x} \in \Omega^+ \tag{3.3b}$$

$$\phi(\boldsymbol{x}) = 0 \quad \Rightarrow \quad \boldsymbol{x} \in \Gamma \tag{3.3c}$$

3.2.1 Updating the level set field to follow the interface

Since the interface moves through time, the level set information must be kept updated accordingly. To do so, the level set equation is used, which is merely an advection equation

$$\frac{\partial \phi}{\partial t} + (\boldsymbol{u}_{\Gamma} \cdot \boldsymbol{\nabla})\phi = 0 \tag{3.4}$$

where \boldsymbol{u}_{Γ} at a given point \boldsymbol{x} is equal to the velocity $\boldsymbol{u}(\boldsymbol{x}_{\Gamma})$ at the closest interface point \boldsymbol{x}_{Γ} . Said differently, \boldsymbol{u}_{Γ} is the extension of the velocity \boldsymbol{u} at the interface Γ to the rest of the domain Ω such that the velocity \boldsymbol{u}_{Γ} remains constant along each normal line N

$$\boldsymbol{u}_{\Gamma} = \boldsymbol{u} \quad \text{on } \Gamma \tag{3.5a}$$

$$\boldsymbol{u}_{\Gamma}(\boldsymbol{x}) = cte \quad \forall \boldsymbol{x} \in N_{\boldsymbol{x}_{\Gamma}} \tag{3.5b}$$

where $N_{\boldsymbol{x}_{\Gamma}}$ is the normal line crossing the interface Γ at point \boldsymbol{x}_{Γ} .

There are ways to numerically extend interface velocities: the idea was first suggested by Adalsteinsson and Sethian in 1999 [154] and has evolved since, in particular see the more recent work of McCaslin Courtine and Desjardins in



2014 [155]). However they are quite $costly^6$. An alternative is to assume that the velocity at points in the neighbourhood of the interface is sufficiently close to the interface velocity, such that it can be used directly

$$\frac{\partial \phi}{\partial t} + (\boldsymbol{u} \cdot \nabla)\phi = 0. \tag{3.6}$$

However, this assumption is not exactly true. The small fluctuations of velocity values in the normal direction will either "compress" or "stretch" the level set field along normals, thereby displacing level set iso-contours. This is shown by Figure 3.3 which shows three level set set iso-contours that are advected either at equal velocity (Fig. 3.3a), increasing velocities in the vertical direction (Fig. 3.3b) or decreasing velocities (Fig. 3.3c). As a consequence, the level set field no longer corresponds to a distance function.

3.2.2 Boundary conditions for level set

The level set advection and reinitialisation problems require a boundary condition on the level set field. The choice of this boundary condition has a direct influence on the contact angle of the level set at the wall. For instance, enforcing an even boundary condition on level set will naturally yield a fluid interface that is always normal to the wall. Apart from periodic boundary conditions, two distinct physical cases can be encountered : either (i) the flow is bounded by a wall or (ii) the flow is unbounded.

In the latter case, it is most natural to pursue the level set field beyond the boundary of the computational domain by enforcing $\frac{D^2\phi}{Dn^2} = 0$. However, without additional care this boundary condition can lead to spurious apparition of bubbles in the ghost node region.

In the first case, an angle of contact must be chosen as a boundary condition for the level set field. To do so, three options are possible. Those are listed from the simplest to implement to the most realistic : either (i) enforce a contact angle of 90 degrees, which merely consist in considering the level set field as an

 $^{^{6}}$ The procedure is very similar to the reinitialization procedure that will be presented in Section 3.2.4 but it must be performed at each time step.

even function around the boundary. Or (ii) use the fluid's static contact angle [156], which is a constant. Finally (iii) the third and most accurate option is to devise a boundary condition that accounts for the dynamic contact angle.

However, the physical law that governs it is very complex. This is due in great part to the influence of very small scale phenomena on macroscopic behavior of the fluid. For instance, the 2010 article by Ren et al. [157] discussed the influence of molecular dynamics on the motion of the contact line. Additionally a 2014 work by Kirkinis and Davis showed the existence of Moffatt vortices on the sides of a moving contact line [158], phenomenon further characterized in 2017 by Febres and Legendre [159]. However, some methods exist that are valid in certain situations. In particular a 2015 article by Legendre and Maglio presents a numerical implementation for moving contact lines that was validated against experiments [160]. It is based in part on the Cox-Voinov model instead of the Navier slip length model used by most authors (for instance see the 2009 work by Afkhami et al. [161]). The Cox-Voinov law is an asymptotic solution valid for low Capillary numbers and which was first derived by USSR researcher Voinov in 1976 [162] and later reintroduced in the Western world by Cox in 1986 [163].

In this thesis, the simple even boundary condition has been used on the level set field. More advanced methods are relevant for future works but will not be used here.

3.2.3 Non-smoothness of level set field

The level set field is always continuous, but not its derivatives. Practically, maximas of level set appear as kinks in the field which sometimes are referred to as *level set shocks* in the literature [133]. Figure 3.4 shows three bubble shapes for which such kinks can be observed. The ellipse (Fig. 3.4a) shows that even in the case of a rounded, smooth fluid interface such kinks can occur. In that case it appears in the center of the ellipse where the level set characteristics from the upper side of the bubble meet the characteristics from the bottom side. The square-shaped bubble (Fig. 3.4b) is of course generally not obtained in physical situations, but is presented here as it is a good example of the level set field around corners. The presence of corners on a fluid interface yields "expansion fans" on one side of the fluid interface and a shock on the other side. Eventually, the drop-shapped bubble (Fig. 3.4c) displays a combination of the effects observed in the two previous cases.

It is therefore important to use schemes that are resilient against discontinuities. Therefore WENO schemes [164] are used to discretize the partial derivative operator in the level set equations. A high-order spatial discretization scheme (WENO5) is used because accuracy in the capturing of the interface motion is critical in our application (see appendix B).

3.2.4 Level set reinitialization

Unfortunately, as the simulation advances, the small distortions in the level set field caused by the constant-velocity assumption of equation 3.6 will accumulate, leading to an error too large to be negligible. To regularize $\phi(\cdot)$ back into



Figure 3.4: Possible shapes for bubbles or drops are shown with their corresponding level set fields (\blacksquare) and interfaces (---). Shocks (---) and expansion fans (---) can be observed.

a signed distance function, the simulation is frozen in time and the level set reinitialization procedure is performed.

There are two main ways to reinitialize the level set field : the Hamilton-Jacobi and the Fast Marching approaches. The spirit of both methods are briefly described, then details of their numerical implementation is given. The study of both methods is required because of a discussion on the curvature calculation that will follow. For a review of reinitialization methods see the 2006 article by Jones et al. [165].

General approach of the Hamilton-Jacobi reinitialization

A first approach is the *Hamilton-Jacobi reinitialization* and consists in propagating correct level set values from the interface to the rest of the domain.

Practically, just before calling the reinitialization procedure, the flow simulation has been frozen at time t. The level set field at that point of time $\phi(\mathbf{x}, t)$ happens to be distorted. The reinitialization procedure consists in solving an initial value problem on a new level set field variable $\phi_{reinit}(\mathbf{x}, \tau)$ that is function of space and of a "reinitialization time" τ . For all values of the reinitialization time τ , the actual time remains frozen at a value t. Firstly, the reinitialization level set ϕ_{reinit} is initialized as the distorted level set $\phi(t)$ from the flow simulation, so that the reinitialization problems know about the interface's location. Then the reinitialization problem in the exterior region Ω^+ is solved

$$\begin{cases} \frac{\partial \phi_{reinit}}{\partial \tau} + (\hat{\boldsymbol{n}} \cdot \nabla) \phi_{reinit} = 1 & \text{on } \Omega^+ \\ \phi_{reinit} = \phi(t) & \text{at } \tau = 0 \\ \phi_{reinit} = 0 & \text{on } \Gamma \end{cases}$$
(3.7)

as well as the reinitialization problem in the interior region Ω^-

$$\begin{cases} \frac{\partial \phi_{reinit}}{\partial \tau} + ((-\hat{\boldsymbol{n}}) \cdot \nabla) \phi_{reinit} = -1 & \text{on } \Omega^+ \\ \phi_{reinit} = \phi(t) & \text{at } \tau = 0 \\ \phi_{reinit} = 0 & \text{on } \Gamma \end{cases}$$
(3.8)

Eventually, the simulation's distorted level set field $\phi(t)$ is overwritten with the fully reinitialized level set field $\phi_{reinit}(\tau \to \infty)$

$$\phi(t) \leftarrow \phi_{reinit}(\tau \to \infty) \tag{3.9}$$

To better understand the meaning of those two problems' equations, note that they are of hyperbolic form and hence can be expressed as a set ODEs

$$\begin{cases} \frac{D\phi_{reinit}}{D\tau} = +1 & \text{on } N(\boldsymbol{x}_{\Gamma}) \cap \Omega^{+} \\ \phi_{reinit} = 0 & \text{on } \Gamma \end{cases}$$
(3.10)

and

$$\begin{cases} \frac{D\phi_{reinit}}{D\tau} = -1 & \text{on } N(\boldsymbol{x}_{\Gamma}) \cap \Omega^{-} \\ \phi_{reinit} = 0 & \text{on } \Gamma \end{cases}$$
(3.11)

along characteristics. The latter are straight lines following normals $N(\boldsymbol{x}_{\Gamma})$ in space and a slope of 1 in time. Integrating those ODEs therefore corresponds to moving along a normal line starting from the interface, and increasing (or decreasing) the level set values at a rate of one per unit distance travelled (Fig. 3.5). In other words, this propagates correct level set values from the interface to the rest of the domain.



(a) Reinitialising in the exterior region (b) Reinitialising in the interior region

Figure 3.5: Hamilton-Jacobi approach to level set reinitialization : : (i) start from the interface, (ii) move away from the interface at a speed of one following the normals and (iii) the level set value at current point of space is equal to the pseudo-time elapsed since the departure from the interface.

General approach of the Fast Marching reinitialization

The other popular reinitialization method is the *Fast Marching Method* (FMM) approach which was introduced by Sethian in 1996 first briefly in [137] then with more details in another article of the same year [166]. It consists in replacing the current distorted level set field $\phi(t)$ by a reinitialized level set field ϕ_{reinit}

$$\phi \leftarrow \phi_{reinit} \tag{3.12}$$

where $\phi_{reinit}(\boldsymbol{x})$ is the solution to the eikonal problem

find
$$\phi_{reinit}$$
 such that
$$\begin{cases} \|\nabla \phi_{reinit}\| = 1 & \text{on } \Omega\\ \phi_{reinit} = 0 & \text{on } \Gamma \end{cases}$$
(3.13)

Contrary to the Hamilton-Jacobi method, the reinitialization level set $\phi_{reinit}(\boldsymbol{x})$ is merely a function of space \boldsymbol{x} (and not a function of a "reinitialization time" τ). The eikonal equation is solved using the Fast Marching algorithm, hence the name of this method. From the location of the interface Γ indicated by the distorted level set field $\phi(t)$, the Fast Marching algorithm finds the closest cell and computes the level value it must hold such that the eikonal equation holds true. This process is repeated until all cells have been processed.

Hence the FMM requires a sorting algorithm to find out which is the closest cell. Depending on the sorting strategy different algorithmic complexities can be achieved. Most FMM methods use heap-sort and have $O(N \log(N))$ complexity, where N is the number of points where the level set function must be reinitialized. Yet Yatziv proposed in 2005 [167] a method based on an array of linked lists that reduced the algorithmic difficulty to linear complexity O(N). In comparison, Hamilton-Jacobi methods are always $O(N \log(N))$. However, because of the requirement of having to find the closest cell, Fast Marching Methods are more difficult to parallelize than Hamilton-Jacobi methods [168] [169] [170]. For that last reason, Hamilton-Jacobi methods will be preferred in this thesis.

From this point on, in order to simplify the notations, ϕ_{reinit} will simply be denoted as ϕ .

Implementation of the Hamilton-Jacobi reinitialization

The two Hamilton-Jacobi reinitialization problems that are valid in the exterior region (Eq. 3.7) and in the interior region (Eq. 3.8) are solved numerically as a single problem holding across the whole domain. The equation of the new problem is

$$\frac{\partial \phi}{\partial \tau} + \operatorname{sign}_{\phi}(\phi) \left(\| \nabla_h \phi \| - 1 \right) = 0.$$
(3.14)

where the sign function $\operatorname{sign}_{\phi}(\phi)$ allows a smooth transition between the interior Ω^{-} and exterior Ω^{+} regions' problems.

Indeed, the $\operatorname{sign}_{\phi}(\cdot)$ function cannot be a simple sign function

S

$$\operatorname{sign}_{\phi}(\phi) := \begin{cases} +1 & \text{if } \phi > 0 \\ -1 & \text{if } \phi < 0 \\ 0 & \text{if } \phi = 0 \end{cases}$$
(3.15)

because that would introduce discontinuities which is incompatible with the finite differences method. Instead a smeared variant of that function is often used [171]

$$\operatorname{sign}_{\phi}(\phi) := \frac{\phi}{\sqrt{\phi^2 + h_x^2}} \tag{3.16}$$

However, reinitializing the level set field introduces numerical errors that tend to displace slightly the interface. For that reason we prefer the sign function recommended by Russo and Smereka in their 2000 article [172] that has less influence on the interface.

$$\operatorname{sign}_{\phi}(\phi) := \frac{\phi}{\sqrt{\phi^2 + \|\nabla \phi\|^2 h_x^2}}$$
 (3.17)

The level set gradient $\nabla_h \phi$ used in (Eq. 3.14) is defined as

$$\nabla_h \phi = D_x \phi \,\, \hat{\boldsymbol{e}}_x \,\, + \,\, D_y \phi \,\, \hat{\boldsymbol{e}}_y \tag{3.18}$$

where $D_x \phi$ and $D_y \phi$ are computed as proposed by Rouy and Tourin in their 1992 work [173], which is

$$D_x\phi := \begin{cases} \max\left(0, +D_x^-\phi, -D_x^+\phi\right) & \text{if } \phi > 0 \\ 0 & (3.19a) \end{cases}$$

$$\int x \phi := \left\{ \max\left(0, -D_x^- \phi, +D_x^+ \phi\right) \quad \text{if } \phi < 0 \right.$$
(3.19b)

where $D_x^- \phi$ and $D_x^+ \phi$ are the numerical approximations of the level set xderivative obtained using respectively downwind and upwind WENO5 schemes (instead of the first order decentered schemes used by Rouy and Tourin). This equation (Eq. 3.19a) (Eq. 3.19b) is a simpler but equivalent formulation of the one used in most articles (for instance [174]). Using the Rouy and Tourin scheme is hence equivalent to enforcing upwinding

Advection velocity
$$< 0 \Rightarrow$$
 Use downwind scheme (3.20a)

Advection velocity $> 0 \implies$ Use upwind scheme (3.20b)

in all cases except in the "cup" situations (Fig. F.1e) (Fig. F.1j) where

$$\frac{\partial \phi}{\partial t} = \operatorname{sign}_{\phi}(\phi) \tag{3.21}$$

is enforced. The equivalence of Rouy and Tourin scheme with downwinding is further discussed in Appendix F.2.1. Moreover, Rouy's and Tourin's article [173] also proves the scheme's convergence, consistency and monotony in the case where the gradient is computed using a first order decentered discretization.

The full numerical implementation of the level set implementation is described in Appendix C.1.1. The space derivatives of the level set functions embedded in this algorithm are computed using the WENO schemes described in (Algo. 5).

Implementation of the Fast Marching reinitialization

The first task that a Fast Marching Method must do is find the closest node. The algorithm of a heap-sort Fast Marching Method is specified in Appendix C.1.2. More details on how nodes are sorted and prioritized is given there.

Once the closest node has been found, its level set value must be computed such that the eikonal equation (Eq. 3.13) is verified. To do so, the gradient is discretized in the same way as for the Hamilton-Jacobi reinitialization (Eq. 3.19a) (Eq. 3.19b) except that we are not computing a derivative from known level set values but instead a level set value from a known derivative value. Hence the eikonal equation

$$\sqrt{(D_x\phi)^2 + (D_y\phi)^2} = 1 \tag{3.22}$$

is transformed into a quadratic equation on $\phi_{i,j}$

$$\phi_{i,j}^2 - \phi_{i,j} \left(\phi_x + \phi_y \right) + \left(\frac{\phi_x^2 + \phi_y^2 - h^2}{2} \right) = 0$$
(3.23)

where

$$\phi_x := \begin{cases} \min(\phi_{i-1,j}, \phi_{i+1,j}, \phi_{i,j}) & \text{if } \phi > 0\\ \max(\phi_{i-1,j}, \phi_{i+1,j}, \phi_{i,j}) & \text{if } \phi < 0 \end{cases}$$
(3.24)

and

$$\phi_y := \begin{cases} \min(\phi_{i,j-1}, \phi_{i,j+1}, \phi_{i,j}) & \text{if } \phi > 0\\ \max(\phi_{i,j-1}, \phi_{i,j+1}, \phi_{i,j}) & \text{if } \phi < 0 \end{cases}$$
(3.25)

can be computed from known level set values. Equation 3.23 has two roots, one being always positive and the other always negative. Hence

$$\phi_{i,j} := \begin{cases} \frac{\phi_x + \phi_y + \sqrt{2h^2 - (\phi_x - \phi_y)^2}}{2} & \text{if } \phi > 0\\ \frac{\phi_x + \phi_y - \sqrt{2h^2 - (\phi_x - \phi_y)^2}}{2} & \text{if } \phi < 0 \end{cases}$$
(3.26)

However, equation 3.26 cannot be used in all cases. Indeed it requires to have at least one frozen neighbour on the left or right and one on the bottom or top. If only one of the neighbours is known (Fig. 3.6a) then equation 3.26 cannot be used. Instead we must start again from the eikonal equation

$$\sqrt{(D_x\phi)^2} = 1 \tag{3.27}$$

which yields

$$(\phi_{i,j} - \phi_x)^2 = h^2 \tag{3.28}$$

where ϕ_x is the same as in (Eq. 3.24) and whose solution is

$$\phi_{i,j} = \begin{cases} \phi_x + h & \text{if } \phi > 0\\ \phi_x - h & \text{if } \phi < 0 \end{cases}$$
(3.29)

Figure 3.6 shows the four situations that can be encountered. Case (a) must use equation 3.29 whereas all the other cases can use equation 3.26.

3.2.5 Local level set method

The level set function serves two purposes : (i) to decide a value for $\rho(\boldsymbol{x})$, $\mu(\boldsymbol{x})$ and $\delta_{\epsilon}(\boldsymbol{x})$ using equations 4.28 and 4.3 and (ii) to compute geometric information on the interface, namely normal vector (Eq. 4.4) and mean curvature



Figure 3.6: Possible cases where the level set value of "candidate" cell (\square) has to be computed from neighbouring "frozen" nodes (\square) whose level set value has already been fixed. (\square) are nodes whose level set value is either not yet known or not part of the stencil and hence cannot be used in the computation.

(Eq. 4.10). In both cases, level set values are only useful in the direct surroundings of the interface. To diminish the computational cost of the method, the level set function can be restricted to a band of a certain thickness surrounding the interface. However, restricting the level set function in the mathematical sense is burdensome to implement. As an alternative, the level set function is still defined on the whole domain, but its values are saturated beyond a narrow band of thickness of ϕ_{thrs} : that is the local level set method that was introduced by Adalsteinsson and Sethian in 1994 [175] (see also Peng's 1999 work [171]).

Level set saturation

After each advection or reinitialization of the level set field ϕ , the following saturation operation is applied to it

$$saturate(\phi) := \begin{cases} +\phi_{sat} & \text{if } \phi > +\phi_{thrs} \\ -\phi_{sat} & \text{if } \phi < -\phi_{thrs} \\ \phi & \text{otherwise} \end{cases}$$
(3.30)

where ϕ_{sat} is the level set saturation value and ϕ_{thrs} is the level set threshold value. Saturating level set values allows to keep level set values where they are meaningful, while leaving the choice to either ignore the nodes where they are not useful or to perform the computations there anyway (as long as their result is neutral). If the computation is expensive, such as WENO5 schemes for instance, then it might be cheaper to not perform it on saturated regions. On the other hand, for cheaper operations, the algorithmic branching ("if [...] then [...]" instructions) involved when filtering out saturated regions might be more expensive than performing the computation across all the domain.

It is critical that the level set saturation value ϕ_{sat} be greater than the level set threshold ϕ_{thrs} . Indeed, performing computations on the level set field might alter slightly the level set values in the saturated region. For the level set field in the saturated region to remain beyond the level set saturation threshold despite those numerical fluctuations, its saturation value must be slightly greater than the threshold. As a consequence, one chooses

$$\phi_{sat} = \phi_{thrs} + \frac{h}{2}.\tag{3.31}$$

Filtering out saturated regions

Whether a given node is within the saturated or non-saturated region of the level set field is indicated by the level set mask $mask(\boldsymbol{x})$. It is a field defined as

$$mask(\boldsymbol{x}) = \begin{cases} 1 & \text{, if } |\phi(\boldsymbol{x})| < \phi_{thrs} \\ 0 & \text{, otherwise} \end{cases}$$
(3.32)

Nodes with a mask value of one are said to lie within the level set mask, and correspond to nodes whose level set value can be used in the computations.

Level set values are sometimes used directly, for instance the viscous term

$$\frac{D\omega}{Dt}\Big|_{i,j} = \frac{1}{\rho_{i,j}} \nabla \times \left(\nabla \cdot \left(\mu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right)\right)\right) + [...]$$
(3.33)

embeds the mass density $\rho_{i,j}(\phi_{i,j})$ at node (i, j), whose value is computed from the level set $\phi_{i,j}$ at the same node (Eq. 4.28). Other times, finite difference schemes take as input level set values at several nodes. For instance the dynamic viscosity μ embedded inside the same viscous term is going to be used in the finite difference schemes discretizing the curl and divergence operators. Therefore, in order to compute $\frac{D\omega}{Dt}\Big|_{i,j}$ at node (i, j), the dynamic viscosities $\mu_{i-1,j}, \mu_{i,j}, \mu_{i+1,j}$, etc must be known, and they are determined respectively from $\phi_{i-1,j}, \phi_{i,j}, \phi_{i+1,j}$, etc. In such cases, the level set values that are of influence on the finite difference scheme should all belong to the level set mask.

It is therefore relevant to define a second type of mask that will be referred to as a "safe-zone", within which $\frac{D\omega}{Dt}|_{i,j}$ can safely be computed. The safe-zone is hence a subset of the mask and its thickness depends on the width of the stencils used. More precisely it is obtained by leaving out as many node layers from the mask as is necessary to make the widest stencil of the method fit within it. Note that higher-order schemes have wider stencils and hence require a larger gap between the boundary of the mask and that of the safe-zone. Also, from experience, the level set value of the outer node layer of the mask tends to be affected by the reinitialization and advection operations and would rather not be used in the computation. All in all, the safe-zone $safeZone(\mathbf{x})$ is obtained by removing the outer ring of mask-flagged nodes plus as many nodes as needed to fit all finite difference stencils, three in our case.

To formalize the present argument, the extend (\cdot) and shrink (\cdot) operators are defined that take as input a set of nodes $V_h \subset \Omega_h$ of the grid Ω_h . The extend (\cdot) operator is defined as the operator that adds all the direct neighbours of nodes already within V_h

extend
$$(V_h) := V_h \cup \{x_{i,j} : \exists x \in V_h \ s.t. \ \|x - x_{i,j}\| = h\}$$
 (3.34)

while the shrink (\cdot) operator removes all points of V_h that have at least one neighbour that is not in V_h

shrink
$$(V_h) := V_h \setminus \{x_{i,j} : \exists x \in \Omega_h \setminus V_h \ s.t. \ \|x - x_{i,j}\| = h\}$$
 (3.35)

Applying the same operator several times allows to extend or shrink the mask by more than one layer

$$\operatorname{shrink}^{2}(V_{h}) = \operatorname{shrink}(\operatorname{shrink}(V_{h}))$$
 (3.36a)

$$\operatorname{extend}^{2}(V_{h}) = \operatorname{extend}\left(\operatorname{extend}\left(V_{h}\right)\right)$$
(3.36b)

In order to define the safe-zone, note that the largest stencil used in our method happens to spread by two nodes from its center. Hence, in our case the safe zone will be obtained by shrinking the mask by three nodes : twice to allow the stencil to remain within the mask plus once to remove the outer ring of nodes whose level set values are a bit distorted by the reinitialization procedure.

$$safeZone(\boldsymbol{x}) := shrink^3(mask(\boldsymbol{x}))$$
 (3.37)

Level set reinitialization in the local level set framework

The local level set does not require any significant change in the way level set is advected, apart from filtering out nodes that are not within the mask.

On the other hand, the level set reinitialization requires special treatment. Indeed, the mask must first be extended so as to cover parts of the saturated region, otherwise the reinitialized field might be incorrect. Figure 3.8 shows a reinitialization procedure done right, whereas Figure 3.7 shows a reinitialization called without prior extension of the level set mask. Both figures start from the same level set field (Fig. 3.7a) (Fig. 3.8a), only the mask is different. The level set field is constituted by two slopes, the left one is not as steep as it should be whereas the right one is too steep. In between there are three regions of space where the level set is saturated. The problem occurs with the rightward slope (the one that is too steep). Since reinitialization only acts from within the level set mask in the first case (Fig. 3.7), the new (correct) slope does not have enough space to reach the saturated regions (Fig. 3.7b). This yields a discontinuous curve for the level set field (Fig. 3.7c). This issue is not encountered in the second case (Fig. 3.8) because the mask has been extended before the reinitialization and hence the new slope reaches and even overruns the level set saturation value (Fig. 3.8b). After saturation, a nice continuous curve is obtained for the level set field (Fig. 3.8c). A general algorithm for local reinitialization is described in Algorithm 3.

Algorithm 3: Level set reinitialization in the local level set framework
$\begin{array}{l} 1 \ \texttt{mask} \leftarrow \texttt{computeMask}(\Phi) \\ \textbf{2} \ \texttt{mask} \leftarrow \texttt{extend}^3 (\texttt{mask}) \\ \textbf{3} \ \Phi \leftarrow \texttt{reinitialize}(\Phi,\texttt{mask}) \\ \textbf{4} \ \Phi \leftarrow \texttt{saturate}(\Phi) \end{array}$

Parameters for level set reinitialization

Ideally, the reinitialized level set field is the solution of the reinitialization problem $\phi_{reinit}(\tau \to \infty)$ as the reinitialization time τ tends to infinity. However,



Figure 3.7: Local reinitialization without mask extension



Figure 3.8: Local reinitialization with mask extension.

(----) is the plot of the level set function ϕ against x and (---) are the level set saturation values. The mask is equal to 1 (---) on some part of the x-axis and to 0 (---) elsewhere. (\bullet) are the points where the level set ϕ is equal to zero which correspond to the bubble's interface.

this is not acceptable in practice. Instead, a finite integration period $\delta \tau$ is used that is sufficiently long to allow for the correct level set values to propagate from the fluid interface Γ to all nodes within the level set mask. Since the reinitialization equation uses the normal $\hat{\boldsymbol{n}}$ as advection velocity, those values propagate at a speed of one. Hence, $\delta \tau$ is chosen to be equal to the width of the level set mask times a safety coefficient : $\delta \tau := 1.2 \phi_{thrs}$.

The Hamilton-Jacobi reinitialization procedure also requires choosing a time integrator and a time step value h_{τ} . An Euler-Explicit time integration is used because (i) it is computationally cheaper than other time integrators and (ii) what is relevant here is the steady state $\phi_{reinit}(\tau \to \infty)$, not the transient state and hence temporal accuracy is not important. Additionally, the time step it uses must be chosen such that a CFL condition is satisfied. More specifically, the chosen CFL value is 0.1, which is achieved with a time step h_{τ} equal to 0.1 h_x .

3.2.6 Strategies to trigger level set reinitialization

The reinitialization process is both costly and tends to slightly displace the interface artificially. On the other hand, if not performed frequently enough, it can lead to nonphysical results. Three strategies can be considered to trigger reinitialization:

- $reinit_{full,n}$: reinitialise over the whole level set mask every n time step
- $reinit_{always,n}$: reinitialise over n grid cells every single time step
- $reinit_{distorted}$: reinitialise over the whole level set mask every time the level set becomes too distorted.

The last criterion requires a measure of the level set distortion.

Measuring level set distortion

The level set distortion can be measured by measuring the rate at which it gets distorted as proposed by McCaslin and Desjardins in 2014 [176],

$$distortion(\phi) := \max_{\boldsymbol{x}_{\Gamma} \in \Gamma} \left(| \boldsymbol{\hat{n}}^T S(\boldsymbol{x}_{\Gamma}) \, \boldsymbol{\hat{n}} | \right)$$
(3.38)

where $S(\boldsymbol{x},t) := \frac{1}{2} \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right)$ is the strain tensor. Another option is to measure it directly as suggested by Gomez et al. in 2005 [177]

$$listortion(\phi) := \max_{\boldsymbol{x}_{i,j} \in \Omega_{\boldsymbol{h}}} \left(\|\nabla \phi\| - 1 \right)$$
(3.39)

Gomez et al's measure (Eq. 3.39) can appear very attractive, however it lacks robustness because it cannot handle the level set shocks properly (Fig. 3.4). Indeed on a shock, the gradient of level set can be anywhere between zero and one. On the other hand, the first method is unaffected by this issue. Nevertheless measuring distortion rates instead of distortion itself makes the choice of a tolerance criterion difficult because two fields, one being more distorted than the other, might have identical rates of distortion. This is certainly what motivated Herrmann's criterion in his 2008 article [178], which is a variant of Gomez et al's [?] proposal that is essentially insensitive to low level set gradients.

Reinitialise whenever $\nabla \phi < 10^{-4} \text{ or } \nabla \phi > 2$ (3.40)

where the choice of those two particular values is not explained by Herrmann. In that case however, only the compression of level set iso-contours are detected, and their stretching is almost completely ignored.

Another idea would be to complete the second criterion (Eq. 3.39) by filtering out the shocks. Shocks can be detected with the second derivative of the level set field. Indeed, away from shocks the laplacian of a level set field is function the interface's curvature only, which yields rather small laplacian values. On the other hand, level set gradients vary much more across shocks which yields larger laplacian values. Therefore, one could distinguish the case away from shocks where the naive criterion (Eq. 3.39) can safely be used, and the case around shocks where only gradients that are too steep are looked after

$$distortion(\phi, \boldsymbol{x}_{i,j}) := \begin{cases} \max(\|\nabla \phi\|, 1) - 1 & \text{if } shockEval(\phi, \boldsymbol{x}_{i,j}) > shockThrs \\ \|\nabla \phi\| - 1 & \text{otherwise} \end{cases}$$
(3.41)

where the shock measure $shockEval(\boldsymbol{x},t)$ is defined as

$$shockEval(\phi, \boldsymbol{x}_{i,j}) := \|h\nabla^2 \phi_{i,j}\|$$
(3.42)

This can be seen as a variant to Herrmann's proposal (Eq. 3.40). The parameter *shockThrs* will be taken to 0.4, and the threshold value for *distortion*(ϕ) beyond which reinitialization must be performed is set to 0.1. Following the numerical experiment described in Section 6.6.1, it was decided to reinitialize whenever the level set field became too distorted in respect to criteria (Eq. 3.41).

3.3 Particle-based level set

Both the level set advection equation (Eq. 3.6) and reinitialization equations (Eq. 3.7) (Eq. 3.8) are transport equations. In other words their only term is an advection term, put aside the +1 or -1 source term of the reinitialization equation. It is hence tempting to express those equations in the Lagrangian frame of reference so that they become

$$\begin{cases} \frac{\partial \tilde{\phi}}{\partial t} = 0\\ \frac{\partial \boldsymbol{X}_p}{\partial t} = \boldsymbol{\tilde{u}} \end{cases}$$
(3.43a)

for the advection equation and

$$\begin{cases} \frac{\partial \tilde{\phi}_{reinit}}{\partial \tau} = +1 \\ \frac{\partial \boldsymbol{X}_p}{\partial \tau} = +\boldsymbol{\hat{n}} \end{cases}$$
(3.44a)
$$\begin{cases} \frac{\partial \tilde{\phi}_{reinit}}{\partial \tau} = -1 \\ \frac{\partial \boldsymbol{X}_p}{\partial \boldsymbol{X}_p} \end{pmatrix}$$
(3.44b)

for the reinitialization equations, where $\tilde{\phi}(\boldsymbol{X}_p, t)$ and $\tilde{\phi}_{reinit}(\boldsymbol{X}_p, t)$ are functions of Lagrangian coordinates (\boldsymbol{X}_p, t) , that is particles' initial position \boldsymbol{X}_p and time t.

Unfortunately, such Lagrangian level set reinitialization is not that simple to implement, and it must instead be solved on the grid using either the Hamilton-Jacobi procedure (Algo. 10) or the Fast Marching algorithm (Algo. 11). This is due to the fact that the reinitialization equations describe the propagation of correct level set values from the interface to the rest of the domain. The Lagrangian-form of the reinitialization equations hence corresponds to moving particles from the fluid interface to the rest of the domain. The problem is that (i) it requires to seed particles at the interface explicitly whereas the traditional grid-based reinitialization handles any point of the grid regardless of how far from the interface they are located and (ii) on one side of a curved interface, the particles will tend to move away from each other leaving a gap in between them which yields Lagrangian distortion (see section 2.3.1).

Nevertheless, the advection equation can still be solved in the Lagrangian frame of reference. This idea was proposed by Hieber and Koumoutsakos in their 2005 article [18]. A summary of the implementation of a particle-based level set method in given in Algorithm 4. Eventually, note that the local level set framework can also be used for particle-based level set, and so will be the case of the VPM method developed in the present thesis.

Algorithm 4: Particle-based level set implementation

Advect level set
 for each particle in particleCloud do
 X_pⁿ⁺¹ = **X**_pⁿ + ∫<sub>t<sup>n</sub></sub>^{tⁿ⁺¹} **u**_p(t) dt
 Φ_pⁿ⁺¹ = Φ_pⁿ
 2 Reinitialise level set
 if isTimeToReinitialise then
 Φⁿ⁺¹ = P2M(Φ_pⁿ⁺¹)
 reinitialise(Φⁿ⁺¹)
 Φ_pⁿ⁺¹ = M2P(Φⁿ⁺¹)
 Example 1
 Φ_pⁿ⁺¹ = M2P(Φⁿ⁺¹)
 For the matrix of the matrix o</sub></sup>
Chapter 4

Computing the phase- and interface-dependent terms

In the previous chapter, a level set method has been presented to keep track of the interface location. Using that information, source terms of the momentum equation can be computed. In particular, the surface tension term, which requires to know the interface's location and curvature, and the viscous term which needs to know in which phase and how far from the interface the current point is located.

This chapter presents the numerical methods employed to compute both of those terms, starting with the surface tension term and followed by the viscous term.

4.1 Surface tension term

Now that a method has been established to capture the interface's motion, terms that require to know its location can be computed. This is the case of the surface tension term which generates momentum only at or near the interface and whose intensity is a function of its curvature.

Continuum Surface Force

Following the Continuum Surface Force (CSF) method that Brackbill Kothe & Zemach introduced in their 1991 article [179], the surface tension force is evaluated, for a velocity-pressure solver, as a volume force in the neighbourhood of the interface

$$\rho \boldsymbol{F}_{\Gamma} := -\sigma \,\delta_{\epsilon} \,\kappa \,\boldsymbol{\hat{n}} \tag{4.1}$$

where the signed curvature κ is defined as

$$\kappa := \nabla \cdot \hat{\boldsymbol{n}} \tag{4.2}$$

and the smeared mollifier δ_{ϵ} is defined as

$$\delta_{\epsilon}(\phi) := \frac{1}{2\epsilon} \left(1 + \cos\left(\frac{\pi}{\epsilon}\phi\right) \right)$$
(4.3)



Figure 4.1: Computing direction of the surface tension force. An interface Γ (—) separates the interior region Ω^- (—) from an exterior region Ω^+ (—) . Normal vectors (\rightarrow) point from the interface towards the exterior region. In both cases however, the surface tension force (\rightarrow) must point towards the center of curvature (\bullet).

and the normal to the interface $\hat{\boldsymbol{n}}$ is computed as

$$\hat{\boldsymbol{n}} = \frac{\nabla\phi}{\|\nabla\phi\|} \tag{4.4}$$

Several things can be noted: (i) that the force vector is along the normal direction $+\hat{\boldsymbol{n}}$ or $-\hat{\boldsymbol{n}}$, (ii) that it is proportional to the surface tension coefficient σ , (iii) that it is proportional to the curvature of the interface $\kappa := |\nabla \cdot \hat{\boldsymbol{n}}|$, (iv) that it embeds a product with field $\frac{\delta_{\epsilon}(\phi)}{\rho(\phi)}$ which is invariant tangentially to the interface and has a bell shape normally to it, and (v) that the direction of the force is given by $-\operatorname{sign}(\nabla \cdot \hat{\boldsymbol{n}}) \hat{\boldsymbol{n}}$. To explain the latter point, note that the surface tension force tries to flatten the interface and hence must always point towards the center of curvature. Figure 4.1 shows the two possible scenarios: a curved interface Γ separates the interior region Ω^- to the exterior region Ω^+ , and both subfigures show opposite locations of both phases. Note that normal vectors $\hat{\boldsymbol{n}}$ always point towards the exterior region Ω^+ by convention. In the first case (Fig. 4.1a), the normal vectors $\hat{\boldsymbol{n}}$ point away from the center of curvature. Moreover, the divergence of the normals $\nabla \cdot \hat{\boldsymbol{n}}$ is positive and hence $-\operatorname{sign}(\nabla \cdot \hat{\boldsymbol{n}}) \hat{\boldsymbol{n}}$ equals $-\hat{\boldsymbol{n}}$ which indeed points towards the center of curvature. In the second case (Fig. 4.1b), normals point towards the center of curvature. Additionally the normals' divergence $\nabla \cdot \hat{\boldsymbol{n}}$ is negative and hence $-\operatorname{sign}(\nabla \cdot \hat{\boldsymbol{n}}) \hat{\boldsymbol{n}}$ equals $+\hat{n}$, which once again points towards the center of curvature.

For a vorticity equation the surface tension term is equal to the curl of the one of the velocity equation (Eq. 4.1)

$$\boldsymbol{F}_{\Gamma} := -\nabla \times \left(\frac{\sigma}{\rho} \,\delta_{\epsilon} \,\nabla \cdot \,\hat{\boldsymbol{n}} \,\hat{\boldsymbol{n}}\right) \tag{4.5}$$

Note that σ is a constant. Moreover, $\frac{\delta_{\epsilon}}{\rho}$ is a function of the normal coordinates only, hence its gradient $\nabla \frac{\delta_{\epsilon}}{\rho}$ is along $\hat{\boldsymbol{n}}$, and thus $\nabla \frac{\delta_{\epsilon}}{\rho} \times \hat{\boldsymbol{n}} = \boldsymbol{0}$. Therefore, the surface tension term becomes

$$\boldsymbol{F}_{\Gamma} := -\frac{\sigma}{\rho} \,\delta_{\epsilon} \,\nabla \times \left(\boldsymbol{\hat{n}} \,\nabla \cdot \, \boldsymbol{\hat{n}} \right) \tag{4.6}$$

or even

$$\boldsymbol{F}_{\Gamma} = -\frac{\sigma}{\rho} \,\delta_{\epsilon} \,\nabla(\nabla \cdot \boldsymbol{\hat{n}}) \times \,\boldsymbol{\hat{n}}$$
(4.7)

since $\nabla \times \hat{\boldsymbol{n}} = \boldsymbol{0}$.

However, the latter and most expanded form (Eq. 4.7) tends to artificially align the interface on the grid, whereas the other form doesn't (Eq. 4.6). This is shown in Section 6.6.2.

Computing curvature

Analytically, the curvature κ is defined as the absolute value of the divergence of the normal to the interface $\hat{\pmb{n}}$

$$\kappa := |\nabla \cdot \hat{\boldsymbol{n}}| \tag{4.8}$$

and since the normal is equal to the gradient of the level set field we have

$$\kappa = |\nabla^2 \phi| \tag{4.9}$$

However, that is only true for actual level set field and in the case of distorted level set field this calculation generates substantial numerical error. Instead, the level set gradient is normalized by its amplitude and hence calculated as

$$\nabla \cdot \hat{\boldsymbol{n}} = \nabla \cdot \left(\frac{\nabla \phi}{\|\nabla \phi\|}\right) = \frac{\partial_{xx} \phi (\partial_y \phi)^2 + \partial_{yy} \phi (\partial_x \phi)^2 - 2 \partial_x \phi \partial_y \phi \partial_{xy} \phi}{\|\nabla \phi\|^{3/2}}$$
(4.10)

However several authors have pointed out that computing curvature this way (Eq. 4.10) can be problematic. Tornberg [180] uses a low-pass filter on level set in her level set finite element code, Marchandise [149] uses a least mean square level set reconstruction method in her Volume-of-Fluid Discontinuous Galerkin solver.

Also some authors interpolate the curvature on the interface, for instance using a Hamilton-Jacobi equation [181], or read the curvature at the interface as done by Coquerelle and Glockner in 2015 [182]. Also Macklin and Lowengrub [183] proposed in 2005 a method to compute curvature when the level set field is distorted because of topological changes, method that has been generalized by Lervåg in 2013 to be used on all types of solvers [184].

Also [185] uses a cut-off filter on curvature to eliminate values that would be meaningless given the mesh's grid spacing.

Those more advanced techniques won't be used here, and instead the original method will be used. This choice is motivated by the fact that multiphase VPM solvers are very uncommon, and hence it is preferable to use simpler and hence safer methods at least in a first stage.

4.1.1 Level set "Flip-flop" mode

A problem however occurs in vorticity codes when the shape of the fluid interface displays or embeds a wavelength equal to two grid spacings i.e. a "flip-flop mode". Indeed with a collocated mesh it is impossible to "sense" this wavelength.

This is shown by Figure 4.2 which presents plots of the flip-flop mode of a level set field corresponding to a horizontal interface

$$\phi(\boldsymbol{x}_{i,j}) = (\boldsymbol{x}_{i,j} \cdot \boldsymbol{\hat{e}}_y) + \cos\left(\frac{\pi}{h} \left(\boldsymbol{x}_{i,j} \cdot \boldsymbol{\hat{e}}_x\right)\right)$$
(4.11)

Note that the numerical gradient of level set $\nabla_h \phi$ is equal to $\hat{\boldsymbol{e}}_y$ at all grid nodes, in other words this level set field appears numerically as non-distorted. On the plots of Figure 4.2, each kink of the zig-zag corresponds to a level set value stored at a $\boldsymbol{x}_{i,j}$ grid node. The two subfigures differ in which grid nodes are used to discretize vorticity and velocity: in Subfigure 4.2a velocity (black markers) is collocated in respect to the level set field, while vorticity (gray markers) is stored on a staggered grid. On the other hand, the other subfigure (Fig. 4.2b) does the opposite: it expresses velocity in a staggered manner and vorticity in a collocated way. As will be shown, the first subfigure corresponds to a correct discretization while the second is subject to odd-even decoupling. In both cases, and at all nodes, the normal vector is computed according to equation (Eq. 4.4) which yields $\hat{\boldsymbol{n}} = \hat{\boldsymbol{e}}_y$.

In the case of a velocity solver, the surface tension term in the momentum equation is proportional to the divergence of normals $-\nabla \cdot \hat{\boldsymbol{n}} \hat{\boldsymbol{n}}$. For a non-distorted level set field this is equal to $-\nabla^2 \phi \hat{\boldsymbol{n}}$, which can be further simplified to a second order partial derivative in the x-direction $-\partial_{xx}\phi \hat{\boldsymbol{n}}$. At the collocated nodes (\bullet), this second derivative is alternatively strictly positive and negative, which yields a certain surface tension force (\rightarrow) in both directions that tends at flattening the interface. On the other hand the second derivative $\partial_{xx}\phi$ is exactly zero at the staggered nodes (\blacksquare).

In the case of a vortex method, the surface tension term is proportional to the gradient of the divergence of normals $-\nabla (\nabla \cdot \hat{\boldsymbol{n}}) \hat{\boldsymbol{n}}$. This reduces in our case to a third order derivative $-\partial_{xxx}\phi \hat{\boldsymbol{n}}$, which can also be expressed as $\partial_x(\partial_{xx}\phi)$. It has been shown in the previous paragraph that $\partial_{xx}\phi$ is exactly zero on the staggered grid (\blacksquare and \blacksquare), and of alternatively positive and negative on the collocated grid (\blacksquare and \blacksquare). Therefore, it is the opposite for the third derivative $\partial_{xxx}\phi$: it is zero on the collocated nodes (\blacksquare and \blacksquare) and alternatively positive and negative on the staggered nodes (\blacksquare and \blacksquare). Hence the vorticity field is best discretized on the staggered grid.

To summarize, the velocity solver requires a collocated grid, whereas the vorticity solver needs a staggered grid. A more formal demonstration can be found in Appendix F.2.2. One could point out that solvers for incompressible flows that are based on the velocity-pressure formulation typically store the velocity in a staggered fashion (so-called "MAC grid"). This staggering is such that the x- (resp. y-) component of the velocity is stored on the West (resp. South) edge of the grid cell. Note however, that the collocation requirement that we are recommending here for the velocity solver consists in storing the x- (resp. y-) velocity on the same line (resp. column) than the level set values. Therefore the MAC staggering does observe this collocation requirement.

In practice, the consequence of using a collocated mesh in the vortex solver is an odd-even decoupling where surface tension applies to odd and even nodes in a



(a) Good : collocated mesh for velocity and staggered mesh for vorticity. The surface tension term is non-zero in both velocity and vorticity equations.



(b) Bad : staggered mesh for velocity and collocated mesh for vorticity. The surface tension term is exactly zero in both velocity and vorticity equations.

Figure 4.2: Level set flip-flop mode and influence of grid staggering. (—) is a plot of the level set field. The round markers (\bullet) or (\bullet) resp. square markers (\bullet) or (\bullet) represent nodes of the collocated resp. staggered grid. Level set is always discretized on the collocated grid (\bullet) or (\bullet). Black resp. gray markers indicates nodes where velocity resp. vorticity is discretized. Arrows (\rightarrow) resp. (\circ) indicate the sign of the surface tension term for the velocity resp. vorticity equations.



Figure 4.3: Level set field (\blacksquare) and fluid interface ($-\!\!-\!\!-\!\!-$) on a collocated grid, no filters applied.



Figure 4.4: Same as Figure 4.3 but using a first order 2D filter.

segregated way. This can lead to very nonphysical simulation results. Figure 4.3 shows a simulation of an oscillating surface that has been ran using a collocated mesh. The subfigures show the level set field and the interface location at different times, from t = 2.6 to t = 6.6. As can be seen at t = 2.6 (Fig. 4.3a) the interface appears to be smooth up until that time. Then small-scale oscillations of small-amplitude become visible (Fig. 4.3b). Those grow stronger (Fig. 4.3c) until they get so large that they alter significantly the physics of the flow and even break the symmetry of the problem (Fig. 4.3d).

There are two solutions to this problem: either (i) a staggered mesh can be used for vorticity or (ii) the collocated mesh can be kept but a filter must be applied so as to remove the flip-flop mode in the level set field. The first solution is far from trivial for a Lagrangian method such as the Vortex Particle-Mesh methods. Indeed, having a staggered mesh implies that there are two sets of particles and it is not clear how they must be advected (which requires the velocity to be expressed on both sets of particles), nor if special care must be taken to prevent particles of both sets to cross paths.

On the other hand, the filtering solution is much simpler to implement, however it means that if the level set field is stored on a grid of spacing hthen the actual resolution of the level set computations will be of 2h because of the filtering. Hence we achieve the same accuracy as on a grid twice coarser but with a $2^2 = 4$ times larger computational cost (in 2D). Nevertheless the filtering solution will be used in the solver because multiphase Vortex Particle-Mesh solvers are not common, and hence simplicity is to be preferred over performance in such an early stage.

Two-dimensional discrete filters

In one dimension, the filtering can be applied using the discrete n^{th} order filter (see Jeanmart and Winckelmans 2007 $L_{x,n}(\cdot)$ [186]).

$$q_{i,j,filtered} \leftarrow \mathscr{L}_{x,n}(q)$$
 (4.12)

defined as

$$\mathscr{L}_{x,n}(q) := q_{i,j} - \mathscr{L}_x^n(q) \tag{4.13}$$

where \mathscr{L}_x is defined as

$$\mathscr{L}_x(q) := -\frac{1}{4} \left(q_{i-1,j} - 2q_{i,j} + q_{i+1,j} \right). \tag{4.14}$$

Such filtering achieves a gain of

$$\frac{|h_{\lambda,filtered}|}{|h_{\lambda}|} = 1 - \sin^{2n}\left(\frac{k\,h}{2}\right) \tag{4.15}$$

where h_{λ} is the harmonic of q of wavelength λ .

Two-dimensional filters can be obtained by composition of two one-dimensional

filters

$$q_{i,j,filtered} \leftarrow \mathscr{L}_{y,n}(\mathscr{L}_{x,n}(q))$$
 (4.16a)

$$\mathscr{L}_{x,n}(q) := q_{i,j} - \mathscr{L}_x^n(q) \tag{4.16b}$$

$$\mathscr{L}_{y,n}(q) := q_{i,j} - \mathscr{L}_y^n(q) \tag{4.16c}$$

$$\mathscr{L}_x(q) := -\frac{1}{4} \left(q_{i-1,j} - 2q_{i,j} + q_{i+1,j} \right).$$
(4.16d)

$$\mathscr{L}_{y}(q) := -\frac{1}{4} (q_{i,j-1} - 2q_{i,j} + q_{i,j+1}).$$
 (4.16e)

where $L_{y,n}$ is defined similarly to $L_{x,n}$ but in the y-direction. The gain of a composition filter corresponds to the product of the gains of both onedimensional filters (Eq. 4.15). On the other hand if one tries to combine both filters by summing $\mathscr{L}_x(\cdot)$ and $\mathscr{L}_y(\cdot)$

$$q_{i,j,filtered} \leftarrow \mathscr{L}'_n(q) \tag{4.17a}$$
$$\mathscr{L}'_n(q) := q_{i,j} - \mathscr{L}'^n(q) \tag{4.17b}$$

$$\mathscr{L}'(q) := \mathscr{L}_x(q) + \mathscr{L}_y(q) = -\frac{1}{4} \left(q_{i-1,j} + q_{i+1,j} + q_{i,j-1} + q_{i,j+1} - 4q_{i,j} \right).$$

then the resulting filter yields a combined gain that is the sum of the individual filters' gains (Eq. 4.15). As a consequence it is non-zero when only both λ_x and λ_y wavelengths are equal to 2h, which is not the behavior expected (see Jeanmart and Winckelmans 2007 [186]).

The effectiveness of two-dimensional discrete filters is visible on Figure 4.4. This figure shows the same simulation as in Figure 4.3, but uses a first-order two-dimensional discrete filter. The small perturbation on the interface that were visible in Figure 4.3b are no longer present in Figure 4.4b. More spectacular is that the very nonphysical and asymmetric behavior observable in Figure 4.3d has completely vanished in Figure 4.4d.

Selection of the filter's order

The performance of the filter improves with increasing orders n. This affects two parameters: (i) the selectivity of the filter i.e. its capability of altering the flip-flop node without altering other modes, and (ii) its conservativity i.e. its capability at preserving the volume of bubbles.

Firstly, Figure 4.5 shows the gain of one-dimensional filters against each individual mode's wavenumber, as given by the equation 4.15. This is shown for different one-dimensional discrete filters of orders n ranging from 1 to 5. It can be seen that all five schemes filter-out completely the flip flop mode (gain of zero at $k_c h = \pi$), but higher order filters are more selective in the sense that they damp less the lower frequency modes. Note however, that the width of the stencil increases linearly with the order n of the filter. Indeed the stencil covers 2n + 1 nodes.

Eventually, the volume-preserving properties of the filters can be appreciated on Figure 4.6. The later shows three bubbles (subfigures (b) to (d)) subjected to two-dimensional filters of various orders, and whose shape and volume



Figure 4.5: Gain profiles of discrete two-dimensional filters depending on their order n. The solid line (—) corresponds to the first order (n = 1) filter, the finely dotted line (—) is the fifth order (n = 5) filter and the lines in between correspond to the second, third and fourth order filters.



Figure 4.6: Volume preservation of two-dimensional filtering (Eq. 4.16) shown on a circular bubble originally of diameter 0.5^1 .

can be compared to a simulation performed without filtering (Fig. 4.6a). Those bubbles correspond to a snapshot of the second rising bubble benchmark (see section 6.9) at a time of t = 0.1, where a filtering operation has been performed once at each time step. It appears clearly that the first order filter does not perform well enough (Fig. 4.6b) whereas the second (Fig. 4.6c) and third (Fig. 4.6d) order filters are satisfactory.

In our case a good compromise between the filter's order, which impacts the selectiveness and conservativity of the scheme, and the stencil's width is to use a third-order discrete filter.

Tangential discrete filters

Unfortunately two-dimensional filters affects the level set field in the normal direction, which generates nonphysical bubble shapes. This is visible on Figure 6.7b which shows the same simulation with different filtering schemes. The simulation considered is the rising bubble B benchmark (see section 6.9) where the surface tension coefficient has been set to zero. The first subfigure shows a simulation without filtering, the second uses a two-dimensional third order filter, and the third employs a tangential variant of the same filter that will

¹To be more accurate: it is a rising bubble B (see section 6.9) simulation at t = 0.1.

be developed in the present section. As can be seen in the second subfigure (Fig. 6.7b) the simulation filtered with the incorrect scheme yields very non-physical phenomena that corrupts the results.

A close-up of the formation of those "trenches" is shown in Section 6.6.3. This phenomenon is caused by the filtering in the normal direction. Instead the filtering can be applied exclusively in the tangential direction. By analogy with the one-dimensional case (Eq. 4.12) (Eq. 4.13) (Eq. 4.14), the tangential filtering $\mathscr{L}_{t,n}(\cdot)$ is performed as

$$q_{i,j,filtered} \leftarrow \mathscr{L}_{t,n}(q)$$
 (4.18a)

$$\mathscr{L}_{t,n}(q) := q_{i,j} - \mathscr{L}_t^n(q) \tag{4.18b}$$

$$\mathscr{L}_t(q) := -\frac{1}{4} \frac{\partial^2 \phi}{\partial t^2}$$
(4.18c)

where the second order derivative on level set can either be computed in the "direct" form

$$\frac{\partial^2 \phi}{\partial t^2} = \hat{\boldsymbol{t}}^T \cdot \nabla \left(\nabla \phi \right) \cdot \hat{\boldsymbol{t}}$$
(4.19)

or in the divergence form

$$\frac{\partial^2 \phi}{\partial t^2} = \nabla \cdot \left(\left(\boldsymbol{I} - \hat{\boldsymbol{n}} \otimes \hat{\boldsymbol{n}} \right) \nabla \phi \right)$$
(4.20)

Numerically, we hence have three possible implementations of the tangential filtering. The first implementation is the direct form (Eq. 4.21) where both $\nabla(\nabla \phi)$ and $\hat{\boldsymbol{n}}$ are expressed at cell centers.

$$\frac{\partial^2 \phi}{\partial t^2} = \hat{\boldsymbol{n}}_{i,j}^T \cdot (\nabla (\nabla \phi))_{i,j} \cdot \hat{\boldsymbol{n}}_{i,j}$$
(4.21)

The second and third implementations use the divergence form (Eq. 4.20) where both $\nabla \phi$ and $\hat{\boldsymbol{n}}$ are expressed at cell faces

$$\frac{\partial^2 \phi}{\partial t^2} = \frac{1}{h} \left(\left((\boldsymbol{I} - \boldsymbol{\hat{n}} \otimes \boldsymbol{\hat{n}})_{i+\frac{1}{2},j} \nabla \phi_{i+\frac{1}{2},j} \right) - \left((\boldsymbol{I} - \boldsymbol{\hat{n}} \otimes \boldsymbol{\hat{n}})_{i-\frac{1}{2},j} \nabla \phi_{i-\frac{1}{2},j} \right) + [...]$$
(4.22a)

$$\hat{\boldsymbol{n}}_{i-\frac{1}{2},j} \leftarrow \frac{\nabla \phi_{i-\frac{1}{2},j}}{\|\nabla \phi_{i-\frac{1}{2},j}\|}$$
(4.22b)

and where $\nabla \phi_{i-\frac{1}{2},j}$ is either discretized using a scheme on the cell faces (second implementation)

$$\nabla \phi_{i-\frac{1}{2},j} \leftarrow \left\{ \frac{\frac{1}{h_x} (\phi_{i,j} - \phi_{i-1,j})}{\frac{1}{2h_y} (\frac{\phi_{i,j+1} + \phi_{i-1,j+1}}{2} - \frac{\phi_{i,j-1} + \phi_{i-1,j-1}}{2})} \right\}$$
(4.23)

or by averaging the normals computed in the center both neighbouring cells (third implementation)

$$\nabla \phi_{i-\frac{1}{2},j} \leftarrow \left\{ \frac{\frac{1}{2} \left(\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2h_x} + \frac{\phi_{i,j} - \phi_{i-2,j}}{2h_x} \right)}{\frac{1}{2} \left(\frac{\phi_{i,j+1} - \phi_{i,j-1}}{2h_y} + \frac{\phi_{i-1,j+1} - \phi_{i-1,j-1}}{2h_y} \right)} \right\}$$
(4.24)



Figure 4.7: Surface tension term of the vorticity equation on the first rising bubble case (see section 6.9) computed on the first sub-step of the first time step. A 128x256 grid was used. The original benchmark's circular bubble has been replaced by an ellipse of main radii 0.3 and 0.15.

The correctness of those three implementations is assessed in section 6.6.4. Eventually, the third-order direct form of the tangential discrete filter is chosen. It is computed according to Equation 4.21.

4.1.2 Computation of curvature

Empirically, it can be observed that the surface tension term generates very complex and highly fluctuating vorticity fields, even for simple geometries.

Vorticity fluctuations generated by surface tension term

As an example, Figure 4.7 shows the case of an ellipse-shaped bubble. What is shown are the surface tension terms computed using both implementations described in Section 4.1 (i.e. equations 4.6 and 4.7). Those terms were computed from a reinitialized level set field, since there exist no simple analytical definition of the level set field for an ellipse-shaped bubble. In the case of an horizontally flattened bubble, the surface tension tends at bringing it back towards a circular shape. This implies a positive vorticity in the top-left and bottom-right corners, and a negative vorticity in the top-right and bottom-left corners. At least, that is what is expected.

However, Figure 4.7 shows a succession of dents of intense positive and negative vorticity that are oriented normally to the interface. Both implementations (subfigures 4.7a and 4.7b) are about as bad as one another regarding that point. That being said, on average one can see lighter shades in the top-left and bottom-right corners, and darker shades in the top-right and bottom-left corners, as is to be expected. Moreover, running the simulations with a term that generates such fluctuating fields still yields good results.

Nevertheless one can wonder where those fluctuations come from and whether they can be prevented.

Root cause

Plotting the different terms constituting the surface tension term ((Eq. 4.6) or (Eq. 4.7)) has shown that it is the "signed curvature" component $\nabla \cdot \hat{\boldsymbol{n}}$ that is responsible for those fluctuations. In order to further study this phenomenon the case of a disk is considered.

As seen previously the signed curvature is defined as the divergence of the normal vector

$$\kappa := \nabla \cdot \hat{\boldsymbol{n}} \tag{4.25}$$

which corresponds to the laplacian of the level set

$$\kappa = \nabla^2 \phi \tag{4.26}$$

but in actual simulations it must be computed as

$$\kappa \leftarrow \nabla \cdot \left(\frac{\nabla \phi}{\|\nabla \phi\|}\right)$$
(4.27)

to prevent distortions of the level set field to affect the curvature values. In this section however, only analytical level set fields or level set right after reinitialization will be considered. Therefore, the curvature will be computed as the laplacian of the level set function (Eq. 4.26), since it is a simpler scheme and displays the same "vorticity dents" than the more complex normalized scheme (Eq. 4.27) that is used in actual simulations.

Results are shown in figures 4.8 to 4.11, which are organized as a 4x3 array. Each of the three columns corresponds to a different case: either (left column) the analytical level set field is used directly in the computations, or (central column) a field equal to twice the analytical level set is first reinitialized using the Hamilton-Jacobi approach (see section 3.2.4), or (rightmost column) the Fast Marching Method is employed instead. Each line of the array shows a different type of plot.

Firstly, consider the level set fields in Figure 4.8. They all look very similar, such that it is impossible to tell the difference between them by the naked eye. Next, see the curvature fields (evaluated as $\nabla^2 \phi$) computed from those same fields: the difference is obvious. In the case of the analytical level set field (Fig. 4.9a), the curvature varies normally to the interface and remains constant in the tangential direction, as it should be. This yields a very smooth field. In the case of the Hamilton-Jacobi and of the Fast Marching Method however, dents have appeared (Fig. 4.9b) (Fig. 4.9c). Those dents are strongest in the vicinity of the interface. Hence, it is the reinitialization procedure that produces the dents.

Eventually, whereas the surface tension term of velocity-pressure formulations of Navier-Stokes is proportional the curvature $\nabla \cdot \hat{\boldsymbol{n}}$ directly, the one of the vorticity equation is proportional to its gradient $\nabla(\nabla \cdot \hat{\boldsymbol{n}})$. The additional differentiation operation that it requires amplifies the fluctuations. This is visible in Figure 4.10 where the scale of the fluctuations has one or two orders of magnitude difference between the analytical field (Fig. 4.10a) and the Hamilton-Jacobi (Fig. 4.10b) and FMM (Fig. 4.10c) reinitialized fields respectively.



Figure 4.11: Tangential gradient of curvature, computed from a level set field filtered with the third order tangential filter of direct form (Eq. 4.21)

Recall that a level set reinitialization procedure propagates correct level set field values from the interface to the rest of the domain. This implies that across the interface the advection velocity of the reinitialization equation changes direction from $-\hat{n}$ in the interior region Ω^- to $+\hat{n}$ in the exterior region Ω^+ . Numerically, this requires a switching of downwinding direction between both sides of the interface, as is enforced by the Rouy scheme (Eq. 3.19a) (Eq. 3.19b) in the case of a Hamilton-Jacobi reinitialization. This is what causes numerical fluctuations in the second derivative of the level set function. And since the downwinding scheme is present in both the Hamilton-Jacobi approach and the Fast Marching Method, both are affected.

It appears that in order to prevent those oscillations from appearing, a completely new approach to level set reinitialization must be devised that is not based on an upwinding technique. Such a method has not been investigated in this thesis.

Additionally, one could wonder whether the tangential filtering that is required by the surface tension term on collocated grids could help to damp those fluctuations. Figure 4.11 shows the same gradient of curvature fields $\nabla(\nabla \cdot \hat{\boldsymbol{n}})$ than Figure 4.10 except that the level set field, after being reinitialized, have been filtered using a third-order tangential discrete filter. Unfortunately no visible improvement is to be observed in comparison to the non filtered case (Fig. 4.10).

Nevertheless, those fluctuations do not seem to affect the flow significantly as the surface tension benchmarks still match the reference data well (see section 6.8).

4.2 Viscous term

Up to now, the fluid properties of both fluid phases were considered identical. In this chapter, the handling of two-phase flows with variable fluid properties is introduced. Several technical solutions exists for that purpose, including the single-fluid model, the ghost fluid method or the immersed interface method. In this PhD, a single-fluid model will be used. This choice is motivated mainly by the simplicity of the method.

4.2.1 Single-fluid model

Within each phase, the fluid properties of the corresponding fluid while near the interface, one transitions smoothly from the properties of one fluid to thee of the other. This transition allows to prevent discontinuities that finite difference methods cannot handle. In this approach the fluid properties (for instance the mass density ρ or the dynamic viscosity μ) are defined as a convex sum of the values within both phases

$$\rho(x) := \rho_{-} (1 - H_{\epsilon}(x)) + \rho_{+} H_{\epsilon}(x)$$
(4.28a)

$$\mu(x) := \mu_{-} (1 - H_{\epsilon}(x)) + \mu_{+} H_{\epsilon}(x)$$
(4.28b)

Hence the same definition of mass density or viscosity are used in all terms of the Navier-Stokes equation. As the interface thickness tends to zero, this model converges to a two-fluid case.

Note that the Brackbill Continuum Surface Force model (see section 4.1) introduced a surface tension force that embedded a smeared mollifier function δ_{ϵ} that had the shape of a cosine

$$\delta_{\epsilon}(\phi) := \frac{1}{2\epsilon} \left(1 + \cos\left(\frac{\pi}{\epsilon}\phi\right) \right) \tag{4.29}$$

and all fluid properties q such as mass density ρ or dynamic viscosity μ are smeared across the interface using the same Heaviside function $H_{\epsilon}(\cdot)$

$$q(\phi) := q_+ H_{\epsilon}(\phi) + q_- (1 - H_{\epsilon}(\phi))$$
 (4.30)

the latter being defined as the integral of the mollifier δ_{ϵ}

$$H_{\epsilon}(\phi) := \int_{-\epsilon}^{\phi} \delta_{\epsilon}(\phi) \, d\phi \tag{4.31}$$

which is

$$H_{\epsilon}(\phi) = \frac{\phi}{2\epsilon} + \frac{1}{2\pi} \sin\left(\frac{\pi}{\epsilon}\phi\right) + \frac{1}{2}$$
(4.32)

within the transition region Γ_{ϵ} .

Additionally, the expression of the vorticity equation (Eq. 2.2) is to be simplified. Starting from a momentum equation

$$\frac{D\boldsymbol{u}}{Dt} = -\frac{\nabla p}{\rho} + \boldsymbol{g} + \frac{1}{\rho} \nabla \cdot \boldsymbol{\tau}$$
(4.33)

and the corresponding vorticity equation

$$\frac{D\omega}{Dt} = -\frac{\nabla p}{\rho} \times \frac{\nabla \rho}{\rho} + \nabla \times \left(\frac{1}{\rho} \nabla \cdot \boldsymbol{\tau}\right)$$
(4.34)

the momentum equation is injected into the baroclinic term of the vorticity equation

$$\frac{D\boldsymbol{\omega}}{Dt} = \left(\frac{D\boldsymbol{u}}{Dt} - \boldsymbol{g}\right) \times \frac{\nabla\rho}{\rho} + \frac{1}{\rho} \nabla \times \left(\nabla \cdot \boldsymbol{\tau}\right)$$
(4.35)

Unfortunately in its present form, the viscous term is unstable. In order to achieve stability, it is decomposed into a laplacian term plus a sum of other terms. Of course a simple expansion can be used to achieve that result,

$$\nabla \times \nabla \cdot \boldsymbol{\tau} = \mu \nabla^2 \omega + \nabla \mu \times \nabla^2 \boldsymbol{u} + \nabla \times \left((\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) \cdot \nabla \mu \right)$$
(4.36)

However a better expression can be obtained by a more sophisticated manipulation. Such a procedure is described by Thirifay's 2006 PhD thesis [115] but unfortunately, the demonstration contains a mistake (see Appendix G for a corrected version of the derivation, and for a list of possible formulations). The correct expansion for a 2D incompressible flow can be expressed as

$$\nabla \times \nabla \cdot \boldsymbol{\tau} = \mu \nabla^2 \omega + 2 \nabla \omega \cdot \nabla \mu - \omega \nabla^2 \mu - 2 \sum_{i=1}^3 \epsilon_{ipq} \frac{\partial u_p}{\partial x_i} \frac{\partial (\nabla \mu)_q}{\partial x_i}$$
(4.37)

or yet still, as an alternative,

$$\nabla \times \nabla \cdot \boldsymbol{\tau} = \nabla^2 (\mu \, \omega) - 2 \, \omega \, \nabla^2 \mu - 2 \sum_{i=1}^3 \epsilon_{ipq} \, \frac{\partial u_p}{\partial x_i} \, \frac{\partial (\nabla \mu)_q}{\partial x_i}$$
(4.38)

The influence of the missing term in Thirifay's original formulation is discussed in Section 6.6.5.

Chapter 5

High mass-density ratio VPM method

5.1 Introduction

Simulation of incompressible multiphase flows with large differences in fluid properties between phases is notoriously difficult because of the steep gradients they introduce [187]. This is especially true with methods such as finite differences that require a certain degree of smoothness of the fields. However several industrial applications require CFD simulations of such flows. Examples of such applications are argon-stirred ladles in metallurgy [188], pool-type heavy liquid metal reactors in the nuclear industry [19] or pipe flow in the petroleum industry [189], amongst others.

5.1.1 Interface modelling methods

One of the most popular methods for multiphase flows is the smeared interface method [190] in which both phases are considered to belong to one single fluid whose properties such as dynamic viscosity μ or mass density ρ are functions of space. More specifically, a mollified Heaviside function $H_{\epsilon}(\cdot)$ is chosen such that the fluid property values transition smoothly across the interface through the convex sum

$$\rho(\boldsymbol{x}) = \rho_{-} \left(1 - H_{\epsilon}(\phi)\right) + \rho_{+} H_{\epsilon}(\phi)$$
(5.1a)

$$\mu(\mathbf{x}) = \mu_{-} (1 - H_{\epsilon}(\phi)) + \mu_{+} H_{\epsilon}(\phi)$$
(5.1b)

where ϕ is the distance to the interface.

Several other multiphase methods exist including the popular immersed interface [191] [192] and ghost fluid methods [193] [194]. In comparison to these methods, the smeared interface method is simpler to implement but not the fastest to converge. Indeed, in a smeared interface method convergence is achieved by shrinking the thickness of the smeared interface to zero, but at the same time the number of nodes across the interface must be increased (for finite difference methods) so that the discretisation errors also converge. This "double convergence" result in a slower global convergence. Additionally, compared to a sharp interface method such as the immersed interface method, the region of the flow affected by the smeared interface in the vincinity of the interface is much greater.

Nevertheless, immersed interface and ghost fluid methods introduce additional complexity because they alter the scheme at the interface. Additionally, in the framework of vortex methods, both also require significant changes in the Poisson solver used to convert vorticity into velocity [118].

All in all the smeared interface method is still relevant thanks to its simplicity, and smeared interface methods that can handle relatively large fluid properties differences remain an important tool for numerical scientists.

5.1.2 Smeared interface vortex methods

Smeared interface vortex methods, which combine both the velocity-vorticity formulation and smeared interface methods, solve the vorticity equation

$$\frac{D\omega}{Dt} = \underbrace{-\frac{\nabla p}{\rho} \times \frac{\nabla \rho}{\rho}}_{\text{baroclinic term}} + \nabla \times \left(\frac{1}{\rho} \nabla \cdot \boldsymbol{\tau}\right) + \nabla \times \boldsymbol{F}_{s}$$
(5.2)

over the whole domain instead of the momentum equation

$$\frac{D\boldsymbol{u}}{Dt} = \boldsymbol{g} - \frac{\nabla p}{\rho} + \frac{1}{\rho} \nabla \cdot \boldsymbol{\tau} + \boldsymbol{F}_s$$
(5.3)

used in velocity-pressure formulations. Here F_s represents the source terms of the momentum equation, whatever they might be, for instance surface tension. In the vorticity equation (Eq. 5.2), the buoyancy effects are not represented directly by a distinct term in the equation, but are handled by the baroclinic term.

The pressure gradient embedded in the baroclinic term of the vorticity equation (Eq. 5.2) can be evaluated in two ways. The most straightforward solution is to inject the momentum equation (Eq. 5.3) on velocity in place of the pressure term

$$\frac{D\omega}{Dt} = \left(\frac{D\boldsymbol{u}}{Dt} - \boldsymbol{g}\right) \times \frac{\nabla\rho}{\rho} + \frac{1}{\rho} \nabla \times (\nabla \cdot \boldsymbol{\tau}) + \nabla \times \boldsymbol{F}_s.$$
(5.4)

The second approach consists in taking the divergence of the momentum equation (Eq. 5.3) in order to transform it into an elliptic equation on pressure

Find
$$p$$
 such that
$$\begin{cases} \nabla \cdot \frac{\nabla p}{\rho} = \underbrace{D}{Dt} (\nabla \cdot \boldsymbol{u}) \\ = 0 \end{cases} + \nabla \cdot \left(\frac{1}{\rho} \nabla \cdot \boldsymbol{\tau}\right) + \nabla \cdot \boldsymbol{F}_s & \text{on } \Omega \\ \nabla p \cdot \boldsymbol{\hat{n}} = 0 & \text{on } \partial\Omega \\ (5.5) \end{cases}$$

then inject that definition of pressure into the vorticity equation (Eq. 5.2). This approach is used in particular in Thirifay's 2002 article [114], or in the

Encyclopedy of Computational Mechanics [46]. However, solving an elliptic problem with non-constant coefficients is very costly since it requires the use of a geometric or algebraic solver. Moreover, in the elliptic approach, terms (for instance the viscous term) are present once in the pressure equation (Eq. 5.5) and once in the vorticity equation (Eq. 5.2). Beyond the clumsiness of the formulation, one can wonder how both terms interact with each other. For that reason, the parabolic approach (Eq. 5.4) appears as a much more acceptable solution.

5.1.3 Buoyancy numerical instability

However, our numerical investigations have discovered evidence of an instability on buoyancy for the parabolic approach (Eq. 5.4). This instability is triggered when the ratio of mass densities between both phases $\frac{\rho_+}{\rho_-}$ exceeds a critical value. That critical value depends on the time integrator that is used but is invariant on the time step value, unlike most numerical instabilities encountered in CFD. It will sometimes be referred to as the "buoyancy instability" for convenience, although it has no connection with physical buoyancy instabilities encountered in e.g. geophysical [195] or astrophysical flows [196]. The instability considered in the present work is purely numerical, not physical.

In this chapter, a simplified 1D model is devised whose outputs will be shown to be consistent with those of a 2D Vortex Particle-Mesh method (also referred to in the literature as a Vortex-in-Cell method). Then an underrelaxation solution to the stability problem is proposed and its behavior is characterized. Following that, it will be shown that a classic harmonic analysis of the instability is not capable of capturing it accurately enough. Eventually, the results of this study will be presented in a practical perspective and an example of simulation will be shown. Additional properties of the instability, such as its sensitivity to simulation parameters, are presented.

5.2 One-dimensional model of the instability

To better capture and understand the dynamics of the instability, it is critical to conceive a model as simple as possible that exhibits the same numerical behavior. The current section simplifies the buoyancy problem in a number of ways until such a simple model is obtained.

5.2.1 General view on the problem

The stability issue arises from the material derivative embedded in the baroclinic term (Eq. 5.4). Indeed, the parabolic form of the vorticity equation (Eq. 5.4) entails a strong coupling: the purpose of this equation is to update the vorticity field in time, vorticity being a way to express the velocity information. In the same time however, the time derivative of velocity is used as an argument to the equation's Right Hand Side. In other words, the same velocity information that is to be obtained from the vorticity equation is also the one



Figure 5.1: $\frac{Dv}{Dt}$ profiles along a line cutting an elongated drop (Fig. 5.2b) horizontally in its center. Shown at time step 0 (—), 1 (— —), 2 (— —), 3 (– – –), 4 (– – –) and 5 (– – –) for an Euler-Explicit time integrator. Gravity acceleration \boldsymbol{g} is set to one $\hat{\boldsymbol{e}}_{\boldsymbol{y}}$.

that has been inputted to it. In practice, the numerical method guesses the definition of the material derivative $\frac{D\boldsymbol{u}}{Dt}$ from previous velocity values.

Consider now the side of a drop of heavier fluid surrounded by a lighter fluid (Fig. 5.2b). Because of the presence of the other phase, the drop's fall is slowed down by buoyancy effects. Hence it should fall at an acceleration lower or equal to free fall acceleration g. On the other hand, the surrounding fluid can rise at arbitrarily high accelerations, within added mass effects. Numerically however, the parabolic form of the equation (Eq. 5.4) performs the first time step by guessing a material acceleration $\frac{D\mathbf{u}}{Dt}$ equal to **0**. Time step after time step, the guess for $\frac{Du}{Dt}$ improves until it converges to the physical material acceleration field. This works well with low mass density ratios for which convergence is reached within a few time steps (Fig. 5.1a). Note that this work considers gravity acceleration vectors \boldsymbol{g} pointed upwards for simplicity of the equations, which means that the "falling" phase is the one moving upwards. With increasing mass density ratios, the convergence becomes slower. Also, beyond a certain $\frac{\rho_+}{\rho_-}$ ratio it achieves convergence after over-shooting an acceleration of g (Fig. 5.1b). In other words, locally and during a few time steps, the falling drop falls faster than free fall, which is nonphysical. Eventually, for even higher mass density ratios, the overshoots are great enough for the simulation to become unstable (Fig. 5.1c). The plots shown in Figure 5.1 have been obtained with the one-dimensional model that will be derived in the present section and use the non-dimensionalizing presented in Section 5.2.4. Before that model can be derived however, the problem must first be simplified.

5.2.2 Simplified governing equations

From the full vorticity equation (Eq. 5.2) and momentum equation (Eq. 5.3), only the terms related to Archimedes principle are responsible for the buoyancy instability: the baroclinic term of the vorticity equation, and the pressure and gravity terms of the momentum equation (Eq. 5.3). The other terms of the equation are either of no influence on this phenomenon or oppose it: surface tension works in the normal direction instead of the tangential direction and viscosity tends to reduce acceleration. To study this numerical phenomenon, we work on the simplest model possible. Therefore, the vorticity equation is reduced to

$$\frac{D\omega}{Dt} = \left(\frac{D\boldsymbol{u}}{Dt} - \boldsymbol{g}\right) \times \frac{\nabla\rho}{\rho} \tag{5.6a}$$

$$\omega(t=0) = 0 \tag{5.6b}$$

$$\boldsymbol{u}(t=0) = \boldsymbol{0} \tag{5.6c}$$

5.2.3 Simplified geometry

Any shape of bubble or drop is potentially subject to the buoyancy instability. For studying the instability however, the interface's shape should be as simple as possible. This study considers a very elongated drop surrounded by a heavier fluid (Fig. 5.2b). More specifically it studies the velocity profile along a line cutting the drop horizontally in its center. Such a drop shape has several advantages. Firstly, note that the baroclinic term generates most vorticity on the sides of the drop, not on its bottom or top. Hence the development of the instability should be observed on the drop's sides. Additionally, a column of fluid (Fig. 5.2a) cannot be used because Archimedes principle only applies to a fully immersed body. For instance, if a solid cylinder is immersed into water such that its bottom touches the bottom of the water vessel in a sealed way, then no buoyancy force will take place. For that reason the infinite column of fluid is not subject to buoyancy and hence cannot be used. Eventually at middle height of a very tall drop (Fig. 5.2b), the geometry becomes very similar to that of a column of fluid (Fig. 5.2a) and some assumptions can be made : that the horizontal component of velocity is zero u = 0 and, as a consequence to the incompressibility condition $\nabla \cdot \boldsymbol{u} = 0$, that the vertical component of velocity is constant in the vertical direction i.e. tangentially to the drop's interface. This yields

$$\omega = \frac{\partial v}{\partial x} \tag{5.7}$$

and

$$\frac{Dv}{Dt} = \frac{\partial v}{\partial t} \,. \tag{5.8}$$

The problem's governing equation (Eq. 5.6a) hence reduces to the one-dimensional ODE $\,$

$$\frac{\partial}{\partial t} \left(\frac{\partial v}{\partial x} \right) = \left((\boldsymbol{g} \cdot \hat{\boldsymbol{e}}_y) - \frac{Dv}{Dt} \right) \frac{\partial \ln(\rho)}{\partial x}$$
(5.9)

where v is a function of merely the horizontal coordinate x and time t, and g is the gravity acceleration.

Furthermore, the previous equation which embeds vorticity $\omega = \frac{\partial v}{\partial x}$ (Eq. 5.9) can be transformed into an equation on velocity only by applying the $S(\cdot)$



Figure 5.2: Case of study for buoyancy

integral operator

$$\frac{\partial v}{\partial t} = \mathcal{S}\left(\left((\boldsymbol{g} \cdot \hat{\boldsymbol{e}}_y) - \frac{Dv}{Dt}\right) \frac{\partial \ln(\rho)}{\partial x}\right)$$
(5.10)

which corresponds to the operator transforming vorticity ω into velocity v through a Biot-Savart law. In the case of a closed box spanning from x_{left} to x_{right} the operator $\mathcal{S}(\cdot)$ is defined as

$$v = \mathcal{S}\left(\frac{\partial v}{\partial x}\right) := \int_{x=x_{left}}^{x} \frac{\partial v}{\partial x} \, dx - \frac{1}{x_{right} - x_{left}} \int_{x_{left}}^{x_{right}} \int_{x_{left}}^{x} \frac{\partial v}{\partial x}(x) \, dx \, dx$$
(5.11)

In other words, $S(\cdot)$ performs the Biot-Savart integral offsetted such that the average vertical velocity across the domain is zero. This offset derives from the fact that the flow takes place within a closed box. Eventually, this closed box is chosen to have free-slip walls.

5.2.4 Non-dimensionalization of problem

This problem is non-dimensionalized as

$$\frac{\partial v^*}{\partial t^*} = \mathcal{S}\left(\left(1 - \frac{Dv^*}{Dt^*}\right) \frac{\partial \ln(\rho)}{\partial x^*}\right) \tag{5.12}$$

where space is non-dimensionalized relative to the interface mollification length ϵ and time is non-dimensionalized relative to the gravity acceleration

$$x^* = \frac{x}{\epsilon}, \qquad t^* = \frac{t}{\sqrt{\frac{\epsilon}{\|g\|}}}$$

$$(5.13)$$

Hence the new system of coordinates x^* is such that the transition region Γ_{ϵ} of the smeared interface method spans from $x^* = -1/2$ to $x^* = +1/2$ (Fig. 5.2d). The transition region Γ_{ϵ} separates the deep interior region Ω_{ϵ}^- from the deep exterior region Ω_{ϵ}^+

$$x^* < -1/2 \Leftrightarrow \boldsymbol{x} \in \Omega_{\epsilon}^-,$$
 (5.14a)

$$-1/2 < x^* < +1/2 \iff \boldsymbol{x} \in \Gamma_{\epsilon}.$$
(5.14b)

$$+1/2 < x^* \qquad \Leftrightarrow \boldsymbol{x} \in \Omega_{\epsilon}^+,$$
 (5.14c)

As a result of this non-dimensionalization, the non-dimensionalized gravity acceleration is $+\hat{\boldsymbol{e}}_y$ and hence the "falling" phase actually moves "upwards". In the following, the non-dimensionalized quantities such as x^* and t^* will be written without the star (e.g. x, t, etc) for the sake of simplicity.

5.2.5 Discretization of the material derivative

The Right Hand Side of both the original vorticity equation (Eq. 5.4) and the 1D model's governing equation (Eq. 5.12) embed a material derivative of velocity $\frac{Dv}{Dt}$ that solvers have to evaluate. To do so it is approximated in both cases to the first order using the operator $D_t[v(\boldsymbol{x},t)](t_2,t_1)$ defined as

$$D_{t}v(t_{2},t_{1}) := \frac{v(t_{2}) - v(t_{1})}{t_{2} - t_{1}}$$
(5.15)

In practice, within time step n (the time step that yields v^{n+1} from v^n) one would use

$$D_t v(t, t^{n-1}) = \frac{v(t) - v^{n-1}}{t - t^{n-1}}.$$
(5.16)

For convenience in later use, we also define $\mathbf{D}_{\mathbf{t}}v^n$ as

$$D_t v^n := D_t v(t^n, t^{n-1})$$
 (5.17)

Note that, for time integrators with substeps, we always use the same v^{n-1} value for all substeps.

For multi-step time integrators, the gap in time between the time $t^{n,m}$ at current substep m and t^{n-1} might be large (up to almost two time steps h_t). Therefore, one could be tempted to discretize the material derivative using the time $t^{n,(m-1)}$ of previous substep m-1 instead. This would yield

$$D_{t}v = \frac{v(t^{n,m}) - v(t^{n,(m-1)})}{t^{n,m} - t^{n,(m-1)}}$$
(5.18)

where $v^{n,m}$ denotes the m^{th} substep of time step n. However, using that form would not be wise. Firstly because the successive substeps of Runge-Kutta time integrators (for instance) are placed at arbitrary locations of the (v, t) space, and hence two consecutive (v, t) points might have a $D_t v$ slope very different from the actual material acceleration $\frac{Dv}{Dt}$. This might generate instability. More formally, that discretization of the material acceleration (Eq. 5.18) does not allow the governing equation (Eq. 5.12) to be written in the form of an equation whose right-hand side operator $Rhs(\cdot)$ takes only two arguments

$$\frac{\partial v}{\partial t} = Rhs(v, t) \tag{5.19}$$

Yet this is the canonical form over which time integrators' stability and convergence properties are typically asserted. By using such a definition of the material derivative (Eq. 5.18) those properties can hence no longer be guaranteed.

Note also that the time argument t of the right-hand side operator $Rhs(\cdot)$ is indeed used in our case because of the D_t operator (Eq. 5.16), contrary to most situations encountered in CFD. This has important repercussions since most studies on Total Variation Diminishing (TVD) [197] or Strong Stability Preserving (SSP) [198] [199] [200] time integration schemes consider a unary right-hand side operator

$$\frac{\partial v}{\partial t} = Rhs(v) \tag{5.20}$$

Hence the results of those studies cannot be used here. The TVD, SSP or Total Variation Bounded (TVB) property of our scheme is beyond the scope of this study.

To summarize, the material acceleration $\frac{Dv}{Dt}$ will be approximated using the first order scheme presented in equation 5.16 which yields

$$\frac{\partial v}{\partial t} = \mathcal{S}\left(\left(1 - \mathcal{D}_{t}v\left(t, t^{n-1}\right)\right)\frac{\partial \ln(\rho)}{\partial x}\right)$$
(5.21)

Note that there is one different ODE per time step, since the time t^{n-1} changes from a time step to the other. Eventually, the initial material acceleration $D_t v^0$ cannot be evaluated because we lack the velocity v^{n-1} and therefore it will be set to zero for the first time step $D_t v^0 = 0$. Solving the problem hence consists in time integrating the previous equation, which yields the sequence of material derivatives $(D_t v^n)_{n \in \mathbb{N}}$

$$\mathbf{D}_{\mathbf{t}}v^0 = 0 \tag{5.22a}$$

$$D_{t}v^{n+1} = \frac{1}{h_{t}} \int_{t^{n}}^{t^{n+1}} \frac{\partial v}{\partial t} dt$$
 (5.22b)

where $\frac{\partial v}{\partial t}$ is determined for each time step by the ODE (Eq. 5.21). Studying the buoyancy instability hence consists in analyzing the convergence or divergence of that sequence of functions.

5.2.6 Governing equation on errors

Since it is an instability that is being studied, it would be more natural to work with an error on material acceleration $D_t v'$ than with material acceleration $D_t v$ itself.

First note that for a given mass density profile $\rho(x)$, there exists a function $\frac{Dv}{Dt}^{\infty}(x)$ function such that

$$\frac{Dv}{Dt}^{\infty} = S\left(\left(1 - \frac{Dv}{Dt}^{\infty}\right) \frac{\partial \ln(\rho)}{\partial x}\right)$$
(5.23)

This function will be called the convergence limit in acceleration since if $D_t v(t, t^{n-1})$ converges, then it converges to $\frac{Dv}{Dt}^{\infty}$. Following this, the error in material acceleration $\frac{Dv'}{Dt}$ is defined analytically as

$$\frac{Dv'}{Dt} = \frac{Dv}{Dt} - \frac{Dv}{Dt}^{\infty}$$
(5.24)

which implies the splitting of the approximations to material acceleration

$$D_t v'(t, t^{n-1}) = D_t v(t, t^{n-1}) - \frac{Dv}{Dt}^{\infty}$$
 (5.25)

Recall also that due to the geometrical assumptions done in Section 5.2.3, the material derivative $\frac{Dv'}{Dt}$ is equal to the partial time derivative $\frac{\partial v'}{\partial t}$. This splitting of the material derivative (Eq. 5.25) yields the governing equation on material acceleration error

$$\frac{\partial v'}{\partial t} = -\mathcal{S}\left(\left(\mathcal{D}_{t}v'\left(t,t^{n-1}\right)\right)\frac{\partial\ln(\rho)}{\partial x}\right)$$
(5.26)

which holds true on $]t^n, t^{n+1}[$ for every time step n. The sequence of errors in material acceleration $(D_t v'^n)_{n \in \mathbb{N}}$ is hence computed as

$$D_t v'^{\ 0} = -\frac{Dv}{Dt}^{\infty} \tag{5.27a}$$

$$D_t v'^{n+1} = \frac{1}{h_t} \int_{t^n}^{t^{n+1}} \frac{\partial v'}{\partial t} dt.$$
 (5.27b)

where the integral is evaluated from the ODE (Eq. 5.26). Therefore studying the stability or instability of the scheme can be done by analyzing the convergence or divergence of the sequence of functions $(D_t v'^n)_{n \in \mathbb{N}}$.

5.2.7 General sequence describing the instability

The generic form of the $(D_t v'^n)_{n \in \mathbb{N}}$ sequence shown previously (Eq. 5.27) is not easy to work with. Instead, we would like to find a definition of it in the form

$$\mathbf{D}_{\mathbf{t}} v'^{\,0} = -\frac{Dv}{Dt}^{\infty} \tag{5.28a}$$

$$\mathbf{D}_{\mathbf{t}} v'^{n+1} = \mathscr{L}(\mathbf{D}_{\mathbf{t}} v'^{n}) \tag{5.28b}$$

where $\mathscr{L}(\cdot)$ is the operator whose definition must be found. This can be achieved by evaluating numerically the integral. However, each time integrator has an $\mathscr{L}(\cdot)$ operator of its own.

As an example, the sequence for the Euler-Explicit time integrator is

$$D_t v'^0 = -\frac{Dv'}{Dt}^\infty$$
(5.29a)

$$D_{t}v'^{n+1} = \mathcal{S}\left(-D_{t}v'^{n}\frac{\partial\ln(\rho)}{\partial x}\right).$$
(5.29b)

whereas the midpoint Runge-Kutta 2 time integrator has

$$\mathbf{D}_{\mathbf{t}} v'^{\,0} = -\frac{Dv'^{\,\infty}}{Dt} \tag{5.30a}$$

$$D_{t}v'^{n+1} = -\frac{2}{3}\mathcal{S}\left(D_{t}v'^{n}\frac{\partial\ln(\rho)}{\partial x}\right) + \frac{1}{3}\mathcal{S}\left(D_{t}v'^{n}\frac{\partial\ln(\rho)}{\partial x}\right)$$
(5.30b)

In the general case, for any m-stage Runge-Kutta time integrator, or more generally any time integrator that can be written in the form

$$\frac{\partial v}{\partial t} = Rhs(v,t) \tag{5.31a}$$

$$v^{n+1} = v^n + h_t \sum_{k_1=1}^m b_{k_1} Rhs_{k_1}$$
 (5.31b)

$$Rhs_{k} = Rhs\left(t^{n} + c_{k}h_{t}, v^{n} + h_{t}\sum_{k'=1}^{k-1} a_{kk'}Rhs_{k'}\right) , \forall k \qquad (5.31c)$$

which corresponds to a Butcher table of the form shown in Table 5.1, the recursion law of its sequence is defined as

$$D_{t}v'^{n+1} = \sum_{l=1}^{m} d_{l} \mathscr{I}^{l} (D_{t}v'^{n})$$
(5.32a)

$$d_{l} := \sum_{k_{1}=1}^{m} \sum_{k_{2}=1}^{k_{1}-1} \cdots \sum_{k_{l}=1}^{k_{l-1}-1} \frac{b_{k_{1}} a_{k_{1}k_{2}} \cdots a_{k_{l-1}k_{l}}}{(1+c_{k_{1}})(1+c_{k_{2}}) \cdots (1+c_{k_{l}})}$$
(5.32b)

$$\mathscr{I}(\mathbf{D}_{\mathsf{t}}v'^{n}) := -\mathscr{S}\left(\mathbf{D}_{\mathsf{t}}v'^{n}\frac{\partial\ln(x)}{\partial x}\right)$$
(5.32c)

To summarize, the sequence of error in material acceleration $(D_t v'^n)_{n \in \mathbb{N}}$ for any given time integrator consists of the initial sequence element $D_t v'^0 = \frac{Dv}{Dt}^{\infty}$ followed by the recursion rule $D_t v'^{n+1} = \mathscr{L}(D_t v'^n)$ where the $\mathscr{L}(\cdot)$ operator of the time integrator is obtained from (Eq. 5.32). Practical examples of such sequences are given for some time integrators in Appendix D.

5.2.8 Results with 1D model

The previous section showed how to obtain a simple sequence that could capture the numerics of the buoyancy instability. From this knowledge, the current section computes the terms of the sequences corresponding to the Euler-Explicit

Table 5.1: Numbering convention for Butcher table's coefficients

and Runge-Kutta 3 time integrators defined in Appendix D. The numerical behavior of those numerical instabilities is studied.

In this section and more generally in the rest of the chapter (unless noted otherwise), the mass density of the left phase ρ_{-} will be fixed at 1 and the mass density will be computed as the integral of a cosine, which in the non-dimensionalized space is

$$\delta_{\epsilon}(x) := 1 + \cos(2\pi x) \tag{5.33a}$$

$$H_{\epsilon}(x) := \int_{x=-1/2}^{x} \delta_{\epsilon}(x) dx = \frac{1}{2} + x + \frac{1}{2\pi} \sin(2\pi x)$$
(5.33b)

$$\rho(x) := \rho_{-} (1 - H_{\epsilon}(x)) + \rho_{+} H_{\epsilon}(x)$$
(5.33c)

Figures 5.3 and 5.4 show respectively an example of a stable and unstable case. Both have been obtained for an Euler-Explicit time integrator, only the mass density of the drop (right-most phase) ρ_+ changes between the two figures. Indeed, the stable case uses $\rho_+ = 30$ whereas the unstable case uses $\rho_+ = 50$. In the stable case, the material acceleration converges to $\frac{Dv}{Dt}^{\infty}$ (Fig. 5.3a) and the error in material acceleration to zero (Fig. 5.3b). On the other hand, for the unstable case, both the material acceleration $D_t v$ and the error in acceleration $D_t v$ diverge (Fig. 5.4a) (Fig. 5.4b). In particular, one can note that the material acceleration $D_t v$ at time step 20 exceeds 1 which means that the drop (located on the right side of the domain) "falls" faster than free fall (Fig. 5.4a) (recall that the gravity acceleration \boldsymbol{g} has been non-dimensionalized to $+\hat{\boldsymbol{e}}_y$).

To appreciate the rate of convergence or divergence of a case of study, one needs to choose an error norm E and plot its evolution as the time step number n increases. The error norm E used in this paper is defined as the Euclidian norm of the error in material acceleration $L^2(D_tv')$ normalized relative to the first term of the sequence

$$E^{n} := \frac{L^{2} (\mathbf{D}_{t} v'^{n})}{L^{2} (\mathbf{D}_{t} v'^{0})}$$
(5.34)

Figure 5.5 shows the influence of the mass density ρ_+ on the instability growth rate for five different ρ_+ values. It shows two plots, each corresponding to a distinct time integrator (Euler-Explicit (see section D.1) or Runge-Kutta 3 (see section D.4)). Each plot displays five curves, one for each ρ_+ value.



(a) Material acceleration $(D_t v^n)_{n \in \mathbb{N}}$ (b) Error on material acceleration $(D_t {v'}^n)_{n \in \mathbb{N}}$

Figure 5.3: Example of a stable problem: Euler-Explicit with $\rho_+/\rho_- = 30$. Apart from $\frac{Dv}{Dt}^{\infty}$ (), each curve corresponds to a single time step: $D_t v^0$ (), $D_t v^4$ (), $D_t v^4$ (), $D_t v^8$ (), $D_t v^{12}$ (), $D_t v^{12}$ (), $D_t v^{12}$ (), $D_t v^{16}$ (), $D_t v^{20}$ ().



(a) Material acceleration $(D_t v^n)_{n \in \mathbb{N}}$ (b) Error on material acceleration $(D_t {v'}^n)_{n \in \mathbb{N}}$

Figure 5.4: Example of an unstable problem: Euler-Explicit with $\rho_+/\rho_- = 50$. Same legend as Figure 5.3.



(a) Euler-Explicit. From the fastest growing to the fastest descending curve: $\frac{\rho_+}{\rho_-} = 80, \frac{\rho_+}{\rho_-} = 50, \frac{\rho_+}{\rho_-} = 41.568718, \frac{\rho_+}{\rho_-} = 35 \text{ and } \frac{\rho_+}{\rho_-} = 30.$

(b) Runge-Kutta 3. From the fastest growing to the fastest descending curve: $\frac{\rho_+}{\rho_-} = 230$, $\frac{\rho_+}{\rho_-} = 180$, $\frac{\rho_+}{\rho_-} = 154.899888$, $\frac{\rho_+}{\rho_-} = 120$ and $\frac{\rho_+}{\rho_-} = 90$.

Figure 5.5: Influence of the mass density ratio.

Increasing curves correspond to unstable cases where the material acceleration diverges to infinity (in absolute value), whereas decreasing curves are stable cases where material acceleration $D_t v$ converges to $\frac{Dv}{Dt}^{\infty}$. The generally flat curve is the marginal stability limit of the scheme. The steepest the curve, the fastest the divergence or convergence rate of the scheme.

Empirically, it appears that the plots of the error norm (Fig. 5.5) consist of a periodic signal trapped into a linearly-decaying envelope. As noted in the previous paragraph, the general slope of the envelope indicates the convergence or divergence rate of the algorithm. It is therefore important to recover the slope information from the plot, which is done using a linear regression $E_{reg}(n)$ in the sense of least squares in the lin-log space

$$\ln(E_{reg}) = a_{reg} n + b_{reg} \tag{5.35}$$

where n is the time step number. This corresponds to a geometric sequence $(E_{reg}^n)_{n\in\mathbb{N}}$ of common ratio $e^{a_{reg}}$

$$E_{reg}^0 = e^{b_{reg}} \tag{5.36a}$$

$$E_{reg}^{n+1} = e^{a_{reg}} E_{reg}^n \tag{5.36b}$$

This common ratio hence corresponds to the average gain of the buoyancy instability.

The critical mass density ratio is the ρ_+/ρ_- value below which the problem is stable and beyond which it becomes numerically unstable. More specifically it is defined as the ρ_+/ρ_- value such that common ratio of the average sequence E_{reg} is equal to 1. The critical mass density ratios for four particular time integrators are specified in Table (Table 5.3) in the column "1D model".

	Geo	meti	ŗy		Pł	nysics			Nume	rics	
	Shur	Risi	ng b.		Shur	Risir	ıg b.		Shur	Rising	bubble
	Siug	Α	В		Siug	Α	В		blug	А	В
L	2	1	1	ρ_{-}	1	100	1	ϵ	1/2	14 h	14 h
H	16	2	2	ρ_+	—	1000	1000	h_t	$1e^{-8}$	adapt.	adapt.
R	1	1/2	1/2	$\boldsymbol{g}\cdot\boldsymbol{\hat{e}}_y$	+1	-0.98	-0.98	BC left	Free-slip	Free-slip	Free-slip
h	4	_	—	σ	0	24.5	1.96	BC right	Free-slip	Free-slip	Free-slip
				μ_{-}	1	1	0.1	BC bottom	Free-slip	No-slip	No-slip
				μ_+	1	10	10	BC top	Free-slip	No-slip	No-slip
								Grid	512×4096	128×256	256×512

Table 5.2: Parameters for 2D VPM simulations

Additionally to the critical ratio, it is also relevant to study the particular $\frac{\rho_+}{\rho_-}$ values for which the instability decays as a rate of (i) halved every 10 time steps (ii) divided by ten every 10 time steps and (iii) divided by a hundred every 100 time steps.

Several parameters can influence the growth rate of the buoyancy instability, which are the time-integrator used (see section 5.6.2), the shape of the mass density profile (see section 5.6.3) and the mass density values within each phase. A sensitivity analysis of the simulation parameters is conducted in Section 5.6. In particular, it will be shown in Section 5.6.3 that $\frac{\rho_+}{\rho_-}$ is a good measure of how far from the critical stability regime one is located.

5.3 Comparison of 1D model against 2D simulations

The analysis of the 1D model's stability or instability done in the previous section is interesting. However it remains useless unless its behavior is shown to be consistent with the one of a 2D numerical solver. Hence this section compares the outputs of a 1D model against those of a 2D Vortex Particle-Mesh solver (also referred to in the literature as a Vortex-in-Cell solver). Note however, that the 1D model is meant to be useful for any vortex method embedding a smeared interface Method. The VPM solver is hence merely one example of such solvers.

The 2D VPM solver uses the parameters presented in Table 5.2 and the initial location of the fluid interface is shown in Figure 5.2c. The comparison is performed both for a one stage integrator (Euler-Explicit time integrator (see section D.1)) and for a multiple stage integrator (Runge-Kutta 3 time integrator (see section D.4)).

5.3.1 Comparison of material acceleration $D_t v$ profiles

Firstly, both 1D model and 2D simulation must predict similar material acceleration profiles $D_t v$. However, the 1D model works with the error in material

acceleration $D_t v'$ instead. Hence it must be converted into a material acceleration $D_t v$ through

$$D_{t}v^{n} = D_{t}v^{\prime n} - D_{t}v^{\prime 0}$$
(5.37)

Note that the initial element $D_t v'^0$ is defined (Eq. 5.28a) as $-\frac{Dv}{Dt}^{\infty}$.

Figures 5.6 and 5.7 compare the $D_t v$ computed by the 2D VPM solver (Subfigure a) to the one calculated by the 1D model (Subfigure b) in the case of an Euler-Explicit and Runge-Kutta 3 time integrator respectively. Qualitatively, the agreement is good between both solvers. Moreover, for a better quantitative measurement, Figure 5.8 plots the difference of material acceleration between 1D model and VPM solver $D_t v^n|_{1D} - D_t v^n|_{2D}$ in those two cases. As can be seen, outside from the regions with high gradients, the difference is an order of magnitude below the one of the raw plots (Fig. 5.6) (Fig. 5.7) which indicates a good agreement.

5.3.2 Comparison of convergence rates of error norm E

Then, the convergence behavior of the error on material acceleration $(D_t v'^n)_{n \in \mathbb{N}}$ sequences is compared between the 1D model and the 2D VPM solver. For unstable or non-converging problems, the error in material acceleration is computed by subtracting the analytical material acceleration $\frac{Dv}{Dt}^{\infty}$

$$D_{t}v'^{n} = D_{t}v^{n} - \frac{Dv}{Dt}^{\infty}$$
(5.38)

where $\frac{Dv}{Dt}^{\infty}$ is the analytical solution of equation 5.23. For converging solutions however, the VPM 2D solver will not have its material acceleration converge to $\frac{Dv}{Dt}^{\infty}$ exactly but to a slightly different function because of numerical errors. Therefore, in stable cases the error in acceleration will be computed as

$$D_t v'^{\ n} = D_t v^n - D_t v^{500} \tag{5.39}$$

In other words, it is assumed that the acceleration at time step 500 corresponds to the numerical convergence limit of the material acceleration $D_t v$. Since the largest time step number that will be considered is 200, this approximation is reasonable. Now that a procedure has been established to obtain the error in acceleration $D_t v'$ in both the 2D simulation and 1D model, the error norm can be computed according to its definition shown in equation 5.34.

Figure 5.9 shows plots of the error norm E^n against the number of the time step number n. Hence if the error norm decreases with n, then the problem is stable and the $(D_t v^n)_{n \in \mathbb{N}}$ converges to $\frac{Dv}{Dt}^{\infty}$. On the other hand, if the error norm E^n increases, then the problem is unstable and the sequence $(D_t v^n)_{n \in \mathbb{N}}$ diverges. Figures 5.9a to 5.9f show a very good agreement in the instability decay rates between 1D model and 2D VPM solver for converging or marginally stable cases. This is even more striking considering that the Euler-Explicit and Runge-Kutta 3 time integrators have very different definitions but are both captured consistently by both 1D and 2D calculations. Note that the 1D curves decay linearly whereas the 2D simulations first descend at a similar rate but then flatten. This is because the 2D simulation works with material



Figure 5.6: Comparison of material accelerations between 2D VPM solver and 1D model: Euler-Explicit with $\frac{\rho_+}{\rho_-} = 50$. Each curve corresponds to a single time step: $D_t v^0$ (—), $D_t v^1$ (— —), $D_t v^2$ (—), $D_t v^3$ (– –), $D_t v^4$ (— —), $D_t v^5$ (— —).





(b) Predicted by 1D model

Figure 5.7: Comparison of material accelerations between 2D VPM solver and 1D model: Runge-Kutta 3 with $\frac{\rho_+}{\rho_-} = 180$. Same legend as (Fig. 5.6)



Figure 5.8: Error between 1D model and 2D VPM solver. Same legend as (Fig. 5.6)



Figure 5.9: Convergence rates without relaxation ($\alpha = 1$)

accelerations $D_t v$ whereas the 1D model works with errors on acceleration $D_t v'$. Indeed, numerically, the calculations are performed using a double precision floating type which provides a finite number of decimals for the mantissa. When performing calculations on material acceleration $D_t v$, the error component of each $D_t v$ has less and less available decimals to work with as it gets smaller. On the other hand by working with the error $D_t v'$ directly, the 1D model is not affected by this phenomenon: no matter how small the error, the same number of significant decimals can always be stored in memory, until the exponent approaches the floating type's exponent limit (10^{-307} for IEEE 754 double precision).

For diverging cases, the behavior of the 1D model seems somewhat different from the one of the 2D solver. Firstly it appears as if the VPM methods prevents the instability growth rate to exceed a maximum limit (Fig. 5.9g) (Fig. 5.9i) (Fig. 5.9j). Secondly, when using the Runge-Kutta 3 integrator, the VPM solver has a range of mass density ratios between 155 and 180 for which the solver is converging extremely slowly. Nevertheless, parameters used to configure 2D simulations should be such that the calculations are strongly stable. Hence the fact that there are differences in the instability growth rates between 1D model and 2D simulation in the unstable or marginally stable cases is not truly problematic. On a different subject, note that the offset at the origin of curves may sometimes be different (Fig. 5.9h), but that translates a difference in the first few time steps and hence does not play a meaningful role in the instability. Indeed what is of interest in the plots of Figure 5.9 is the general slope of the curves, not the offset at the origin.

5.3.3 Comparison of critical mass density ratio ρ_+/ρ_- values

Eventually, Table 5.3 shows the critical mass density ratios ρ_+/ρ_- for both the 1D model and the 2D solver. Since it is difficult to find the exact critical ratio in the case of the 2D solver, the table shows the highest stable ratios below which simulations are stable and the lowest ratio beyond which simulations become unstable. The critical ratio in the case of the 2D solver hence lies in between those two values. On the other hand, the critical ratios are obtained in the 1D model with a relative tolerance of $5e^{-9}$.

One can observe a good agreement between the critical ratios of the 1D model and the 2D VPM solver, with the latter being consistently slightly more stable than the 1D model. The difference is greatest in the case of the RK3 time integrator, but the simulations between 154 and 175 are actually marginally stable, as was noted in the previous paragraph and is visible on figures 5.9f and 5.9h. Hence in practice, 2D simulations start being significantly converging below the 154 value predicted by the 1D model.

5.3.4 Summary

The 1D model behaves consistently with the 2D simulations it is designed to emulate. Hence the 1D model can be used to predict the behavior of 2D simulations and in particular determine their stability regime. It is worth Table 5.3: Comparison of critical ρ_+/ρ_- ratios between the 1D model and a 2D VPM solver for four time integrators at $\rho_- = 1$ and without relaxation $\alpha = 1$

Time integrator	2D VPM	1D model		
Time integrator	Highest stable	Lowest unstable	TD model	
Euler-Explicit	40	42	41.57	
RK2 Other	77	82	77.82	
RK2 Midpoint	85	90	84.47	
Runge-Kutta 3	175	185	154.9	

noting that a 2D simulation performed with a different solver (for instance a spectral solver) might have yielded slightly different results than the ones the VPM solver gave. Hence what has been done here is not truly a validation of the 1D model but rather an assertion of the consistency of the 1D model's outputs with those of a 2D solver.

5.4 Achieving stability through under-relaxation

In the previous sections, both the 1D model and the 2D simulations it emulates have been shown to diverge for large mass density ratios. The present section assesses an under-relaxation solution to allow for unstable simulations to become stable.

5.4.1 Ideal under-relaxation

Any given sequence $(q)_{n \in \mathbb{N}}$ defined as

$$q^{n+1} = \mathscr{L}(q^n) \tag{5.40}$$

where ${\mathscr L}$ can be any operator, can be under-relaxed by a relaxation factor α

$$q^{n+1} = \alpha \mathscr{L}(q^n) + (1-\alpha)q^n.$$
(5.41)

Under-relaxation tampers the intensity of the oscillatory behavior of a sequence, and by doing so it can make a divergent sequence become convergent. On the other hand, it damps all value changes in an undiscriminating way, including those due to physical phenomena. In our case however, the acceleration Dv/Dtvaries fairly slowly in time and hence most fluctuations at high mass density ratios are caused by the buoyancy instability. Therefore, under-relaxation appears as an acceptable solution. This will be assessed in Section 6.6.6.

In particular, let us consider a geometric sequence $(q^n)_{n\in\mathbb{N}}$ of common ratio -g

$$q^{n+1} = -g \, q^n \tag{5.42}$$

where g is a positive number. It has a gain of g which must be less than one for the sequence to converge. On the other hand, its under-relaxed variant has a gain g_{α} of

$$g_{\alpha} = 1 - \alpha \, (1+g).$$
 (5.43)

which can be brought below one no matter the original gain value g. Indeed, for any given gain g, the under-relaxed gain g_{α} can be brought down to a critical value of one by using an under-relaxation coefficient of

$$\alpha = \frac{2}{1+g} \tag{5.44}$$

Note that $\alpha \in]0, 2[$ for all g values.

As noted before (see section 5.2.8), the average sequence of acceleration error E_{reg} displays a convergence or divergence rate that tends to be globally constant albeit with some fluctuations (Fig. 5.5). Hence the buoyancy instability behaves on average like a geometric sequence (Eq. 5.36). As a consequence it appears meaningful to perform under-relaxation so as to alter the instability growth rate and bring the sequence to convergence. The generic definition of the $(D_t v'^n)_{n \in \mathbb{N}}$ sequence previously introduced in equation 5.28 should hence substitute its recursion rule by

$$\mathbf{D}_{\mathbf{t}}v^{n+1} = \alpha \mathscr{L}(\mathbf{D}_{\mathbf{t}}v^n) + (1-\alpha)\mathbf{D}_{\mathbf{t}}v^n.$$
(5.45)

However, this simple change in the framework of the 1D model cannot be implemented simply in 2D solvers without significant consequences. Indeed consider a vorticity equation of the form

$$\frac{\partial \omega}{\partial t} = Rhs(\omega) \tag{5.46}$$

where $Rhs(\cdot)$ corresponds to the Right-Hand Side operator of the vorticity equation. The under-relaxed recursion rule (Eq. 5.45) translates in the framework of 2D solvers as

$$\omega^{n+1} = \omega^n + \alpha \int_{t^n}^{t^{n+1}} Rhs(\omega) dt + \underbrace{(1-\alpha)h_t \nabla \times \frac{D\boldsymbol{u}^n}{Dt}}_{\text{"under-relaxation term"}}$$
(5.47)

where the integral is evaluated from the ordinary differential equation 5.46. However, ideally the right-hand side of the vorticity equation should be modified

$$\frac{\partial\omega}{\partial t} = Rhs'_{\alpha}(\omega) \tag{5.48}$$

such that the under-relaxation is performed from within the right-hand side directly

$$\omega^{n+1} = \omega^n + \int_{t^n}^{t^{n+1}} Rhs'_{\alpha}(\omega) dt \qquad (5.49)$$

Unfortunately the under-relaxation term cannot be simply brought under the integral without altering the outputs of the solver. Indeed, a vorticity equation defined as

$$\frac{\partial \omega}{\partial t} = Rhs(\omega) + \frac{1-\alpha}{\alpha} \nabla \times \frac{D\boldsymbol{u}}{Dt}^{n-1}.$$
(5.50)


(a) 1D model using ideal under- (b) 1D model using "unsplit" relaxrelaxation (Eq. 5.47) ation (Eq. 5.50)

Figure 5.10: Influence of under-relaxation type on material accelerations. Shown using a RK2 Midpoint time integrator, $\alpha = 0.5$, $\rho_+/\rho_- = 1000$. Each curve corresponds to a single time step: $D_t v^0$ (—), $D_t v^1$ (—), $D_t v^1$ (—), $D_t v^2$ (—), $D_t v^3$ (– –), $D_t v^4$ (—), $D_t v^5$ (—).

and integrated as

-

$$\omega^{n+1} = \omega^n + \alpha \int_{t^n}^{t^{n+1}} Rhs'_{\alpha}(\omega) dt$$
(5.51)

is equivalent to (Eq. 5.47) only to first order, as for any splitting scheme. Figure 5.10 compares the output of a 1D model performing the ideal under-relaxation (Eq. 5.47) and that of a 1D model modified to perform the "unsplit" underrelaxation (Eq. 5.50). It is very clear that the output of both cases are very different from each other. Moreover, the "ideal under-relaxation" case (Fig. 5.10a) converges at an average common ratio of 0.9617 whereas the "unsplit nuderrelaxation" form (Fig. 5.10b) diverges with an average common ratio of 1.0458, which is to say a relative error of 8.75%. Eventually, note that those differences are obtained just by moving the under-relaxation term from outside the integral (Eq. 5.47) to within the ODE (Eq. 5.50). The under-relaxation factor α is still outside the integral (Eq. 5.51). Moving it to within the integral will also bring changes to the solver's output. It is hence shown in this section that the ideal under-relaxation cannot be implemented by a change in the vorticity equation without altering greatly the solver's outputs.

5.4.2 Alternative to ideal under-relaxation

Since an "ideal" under-relaxation cannot be implemented through a simple change of the vorticity equation, an alternative solution is chosen. The vorticity equation (Eq. 5.4) is modified by substituting the material acceleration in the baroclinic term by an "under-relaxed" material derivative approximation $D_t \boldsymbol{u}^*$

$$\frac{D\omega}{Dt} = (\mathbf{D}_{t}\boldsymbol{u}^{*} - \boldsymbol{g}) \times \frac{\nabla\rho}{\rho} + \frac{1}{\rho}\nabla\times(\nabla\cdot\boldsymbol{\tau}) + \nabla\times\boldsymbol{F}_{s}$$
(5.52)



(a) Euler-Explicit under-relaxed with $\alpha = 0,75$. From the slowest to the fastest descending curve: $\rho_+ = 80$, $\rho_+ = 50$, $\rho_+ = 41.568718$, $\rho_+ = 35$ and $\rho_+ = 30$.

(b) Runge-Kutta 3 under-relaxed with $\alpha = 0.90$. From the slowest to the fastest descending curve: $\rho_+ = 230$, $\rho_+ = 180$, $\rho_+ = 154.899888$, $\rho_+ = 120$ and $\rho_+ = 90$.

Figure 5.11: Same cases as in (Fig. 5.5) but under-relaxed such that they all converge. Each curve corresponds to a given ρ_+ while ρ_- remains fixed at 1

where the under-relaxed material acceleration approximation $D_t u^*$ is calculated within each time step n as

$$D_{t}\boldsymbol{u}^{*,n} = \alpha D_{t}\boldsymbol{u}(t,t^{n-1}) + (1-\alpha) D_{t}\boldsymbol{u}^{*,n-1}$$
(5.53)

and the initial under-relaxed material acceleration $\frac{D\boldsymbol{u}}{Dt}^{*,-1}$ is taken equal to zero. This form of under-relaxation is chosen because it brings very small changes to the vorticity equation and only affects the term responsible for the instability i.e. the baroclinic term.

In the 1D model, the under-relaxation on material acceleration $D_t \boldsymbol{u}$ translates itself as an under-relaxation on the acceleration error $D_t v'$

$$D_{t}v'^{*,n} = \alpha D_{t}v'(t,t^{n-1}) + (1-\alpha) D_{t}v'^{*,n-1}$$
(5.54)

Similarly to the non-relaxed case (see section 5.2.7), and using the same formalism (see equation 5.31), the under-relaxed sequence is defined as

$$D_{t}v^{*,n} = \sum_{l=1}^{m} \alpha^{l-1} d_{l} \mathscr{L}^{l} \left(\alpha D_{t}v^{n} + (1-\alpha) (1+c_{k_{l}}) D_{t}v^{*,n-1} \right)$$
(5.55a)

$$d_{l} = \sum_{k_{1}=1}^{m} \sum_{k_{2}=1}^{k_{1}-1} \cdots \sum_{k_{l}=1}^{k_{l-1}-1} \frac{b_{k_{1}} a_{k_{1}k_{2}} \cdots a_{k_{l-1}k_{l}}}{(1+c_{k_{1}})(1+c_{k_{2}}) \cdots (1+c_{k_{l}})}$$
(5.55b)

The under-relaxed sequence of four time-integrators is given in Appendix D.

5.4.3 Results of under-relaxation

Under-relaxation aims at stabilizing previously unstable simulations. Figure 5.11 shows the same five cases as the ones previously studied in the non-relaxed

cases (Fig. 5.5) considered in Section 5.2.8. The Euler-Explicit and Runge-Kutta 3 time integrators defined in (D.1) and (D.4) use an under-relaxation factor α of respectively 0.75 and 0.90. Those values have been chosen arbitrarily such that all the unstable or non-converging cases became stable, as can be seen on Figure 5.11.

5.4.4 Comparison of 1D model against 2D VPM solver

Similarly as in the earlier section (see section 5.3), the output of the 1D model is compared to that of a 2D VPM solver, but this time with under-relaxation.

Firstly, the material acceleration profiles outputted by the 1D model is compared to the one calculated by the 2D VPM solver. Figures 5.13 and 5.12 show those profiles in the case of an Euler-Explicit and of a Runge-Kutta 3 time integrators. One can observe a very good match between the output of both solvers. This is confirmed with the plot on absolute error (Fig. 5.14) which shows that the absolute error is still an order of magnitude smaller outside the regions of high-gradients.

Moreover, the convergence plots shown in Figure 5.15 confirm a good consistency between prediction between the under-relaxed 1D model and the corresponding 2D simulations. The agreement is excellent for the Euler-Explicit time integrator. On the other hand, the predictions of the 1D model are not as accurate in the Runge-Kutta 3 case, yet they remain very good.

5.5 Harmonic analysis of the instability

The motivation behind the development of the 1D model presented in Section 5.2 is to express the numerical instability problem in the simplest form possible. In the case of the 1D model, it is represented by a sequence of functions $(D_t v'^n)_{n \in \mathbb{N}}$ having a linear recursion rule. In practice, the user would need to first find the recursion rule corresponding to the time integrator used with the help of equation 5.55, then run a numerical solver to compute the $D_t v'^n$ elements of the sequence one by one to see whether and how fast it converges. However, one could wish for an even easier-to-use solution.

Traditionally, instabilities are characterized through harmonic analysis. This method is very popular thanks to its extreme simplicity and its accuracy at capturing the behavior of linear instabilities. The present section attempts at performing a harmonic analysis of the sequence of acceleration errors $(D_t v'^n)_{n \in \mathbb{N}}$ in the hope of obtaining accurate instability growth or decay rates without having to resort to a numerical solver.

Unfortunately, in the case of the buoyancy numerical instability, the harmonic analysis cannot be conducted exactly. Indeed, the $S(\cdot)$ operator introduces a shift in the values because of its very definition (Eq. 5.11). For instance when inputting a cosine, $S(\cdot)$ outputs a sine plus an offset. This shift translates in the equations as an additional term, which is not easily handled by a harmonic analysis. However, one can assume the shift to be equal to zero and perform an harmonic analysis based on this hypothesis. The impact of this



Figure 5.12: Comparison of material accelerations between 2D VPM solver and 1D model: Euler-Explicit with $\frac{\rho_+}{\rho_-} = 80$ and $\alpha = 0.75$. Each curve corresponds to a single time step: $D_t v^0$ (---), $D_t v^1$ (----), $D_t v^2$ (---), $D_t v^2$ (---), $D_t v^5$ (----).





(b) Predicted by 1D model

Figure 5.13: Comparison of material accelerations between 2D VPM solver and 1D model: Runge-Kutta 3 with $\frac{\rho_+}{\rho_-} = 230$ and $\alpha = 0.90$. Same legend as (Fig. 5.6)



Figure 5.14: Error between 1D model and 2D VPM solver. Same legend as (Fig. 5.12)



Figure 5.15: Convergence rates with under-relaxation

assumption on the accuracy of the prediction will be assessed in the case of an Euler-Explicit time-integrator.

The error in material acceleration $D_t v'$ is Fourier-decomposed in a sum of harmonics $D_t v'_{\lambda}$ of wavelength $\lambda = 2\pi/k$. Each are a product of an amplitude time-dependent function A(t) and a shape function $e^{-i k x}$ that is space-dependent

$$\mathbf{D}_{\mathbf{t}}v_{\lambda}' = A(t)\,e^{-i\,k\,x} \tag{5.56}$$

In the absence of offset, the local equation on acceleration error (Eq. 5.26) is equivalent to

$$\frac{\partial}{\partial x} \left(\mathbf{D}_{\mathbf{t}} v'^{n+1} \right) = - \mathbf{D}_{\mathbf{t}} v'^{n} \frac{\partial \ln(x)}{\partial x}$$
(5.57)

Each harmonic is inputted individually in the previous equation which yields

$$A^{n+1} i k^{n+1} e^{-i k^{n+1} x + \varphi} = A^n e^{-i k^{n+1} x} \frac{\partial \ln(\rho)}{\partial x}$$
(5.58)

where φ is the offset in phase. Taking the argument yields

$$k^{n+1}x + \frac{\pi}{2} + \varphi = k^n x$$
 (5.59)

Hence the harmonic at the next time step is phase-shifted by $-\frac{\pi}{2}$ and keeps the same frequency. Taking the modulus yields

$$\frac{A^{n+1}}{A^n} = \frac{1}{k^n} \frac{\partial \ln(\rho)}{\partial x}$$
(5.60)

Hence the stability or instability of the scheme is dictated by the amplitude of the $\frac{\partial \ln(\rho)}{\partial x}$ function. The highest gain is obtained at the longest admissible wavelength which is equal to the thickness 2ϵ of the interfacial transition region Γ_{ϵ} . Indeed, since next time step's acceleration will be phase-shifted by $\frac{\pi}{2}$, a wavelength of twice the interfacial thickness cannot survive more than a time step. Hence

$$g_{max} = \frac{1}{2\pi} \max_{x \in \Gamma_{\epsilon}} \left(\frac{\partial \ln(\rho)}{\partial x} \right)$$
(5.61)

For the scheme to be stable one requires $g_{max} < 1$, which is to say

$$\max_{x \in \Gamma_{\epsilon}} \left(\frac{\partial \ln(\rho)}{\partial x} \right) < 2\pi$$
(5.62)

Recall that the problem has been non-dimensionalized in Section 5.2.4. With the left-phase mass density fixed as $\rho_{-} = 1$ and the mass density profile being computed as specified in equation 5.33, the marginal stability limit is reached at a mass density ratio ρ_{+}/ρ_{-} of 15.64.

To summarize, the harmonic instability predicts preservation of the wavenumber

$$k^{n+1} = k^n \tag{5.63}$$

phase-shifting of the modes by

$$\varphi = -\frac{\pi}{2} \tag{5.64}$$

and a maximum gain of

$$g_{max} = \frac{1}{2\pi} \max_{x \in \Gamma_{\epsilon}} \left(\frac{\partial \ln(\rho)}{\partial x} \right)$$
(5.65)

Previously, the 1D model predicted a critical mass density ratio $\rho_+/\rho_$ of 41.5 (Table 5.3). In comparison, the critical value of 15.64 obtained by the harmonic analysis captures the correct order of magnitude but is off by a 62% relative error. Hence the approximation performed at the beginning of the harmonic study which consisted in neglecting the effect of the offset introduced by operator $S(\cdot)$ is not acceptable, although it can be used to have an evaluation of the order of magnitude of the mass density ratio ρ_+/ρ_- .

On the other hand, one can note that the acceleration error profiles $D_t v'$ outputted by the 1D model are very different from a time step to the next (Fig. 5.6b), yet bear similarities when taken over a 4 time steps period (Fig. 5.3b). This seems consistent with the $-\frac{\pi}{2}$ phase-offset predicted by the harmonic study.

Likewise, the exponential growth of the buoyancy instability (i.e. linear growth on a log scale) (Fig. 5.5) is consistent with the preservation of the wavenumber $k^{n+1} = k^n$ predicted by the harmonic instability. Indeed, if each mode had a different gain and that the wavenumber changed from a time step to the next, then this exponential growth or decay of the instability would not be observed.

To put it back into a practical perspective, the buoyancy instability's growth or decay rate cannot be reliably obtained through harmonic analysis. Instead it must be obtained through the use of a numerical solver that computes each term of the sequence of acceleration errors $(D_t v'^n)_{n \in \mathbb{N}}$ one after the other.

5.6 Practical view on the buoyancy numerical instability

At the present stage, a method to predict the growth or decay rate of the buoyancy instability has been proposed. Practically, a user whose simulation is subject to the buoyancy instability would need to (i) find the recursion rule of the material acceleration sequence $(D_t v'^n)_{n \in \mathbb{N}}$ corresponding to the time integrator used (Eq. 5.55), (ii) run a script to calculate the sequence elements $D_t v'^n$, (iii) observe whether their method is convergent or not, (iv) if not, change the simulation parameters and start back from step (i) until convergence is achieved.

From the user perspective, two simulation parameters are to be chosen carefully when considering a simulation subject to the buoyancy numerical instability. Firstly, the under-relaxation factor α obviously plays an important role. Secondly, the choice of the time integrator has a strong influence on the stability of the numerical method as well. This was hinted in Table 5.3 which showed that different time integrators have different values for the critical mass density ratio ρ_+/ρ_- . Eventually, the user sets the mass density values and profile according to the requirement of the simulation to be performed. Hence those parameters cannot be freely chosen, yet they are instrumental to the development of the instability.

The purpose of this section is to give a more complete detailed description of the influence of these parameters, and help the user in the choice of their simulations' parameters.

5.6.1 Influence of the under-relaxation factor α

For a given time integrator, each under-relaxation factor α allows for a certain critical mass density ratio ρ_+/ρ_- . Figure 5.16 plots the marginal stability curve (—) of four time integrators. Each (—) curve connects under-relaxation values α to the respective critical mass density ratios ρ_+/ρ_- they allow. This plot can also be read in the other direction and help a user who would like to know, given the mass density ratio ρ_+/ρ_- of their simulation, what under-relaxation factor they must choose.

To each curve corresponds a given convergence rate, and for that reason they will be referred to as "iso-convergence" curves in the rest of this document. For instance (----) represents all cases that converge at a rate of 1/2 every 10 time steps, whereas the marginally stable curve (----) is the collection of situations where the convergence rate is zero.

Notice that for relaxation factors α below 0.4, a lower relaxation factor does not necessarily imply more stability. This is due by the fact that the "under-relaxation" implemented is not exactly an under-relaxation (see section 5.4.1).

5.6.2 Influence of the time integrator

Previously Table 5.3 showed in the non-relaxed case that the critical mass density ratio value ρ_+/ρ_- depended on the time integrator used. This information can be generalized in the under-relaxed case. This is the purpose of Figure 5.17 which shows the marginal stability iso-convergence curves of four different time integrators on the same plot. It appears that higher-order time integrators tend to be more stable, although the data is too scarce to support this statement in the general case.

It also shows that two time integrators of same order (RK2 Midpoint (see section D.2) and RK2 "Other" (see section D.3) in the present case) might not have the same behavior regarding the buoyancy numerical instability, with one (RK2 Midpoint) being more stable than the other (RK2 "Other"). This is even more remarkable considering that both have the same stability properties according to a von Neumann stability analysis. This property is further discussed in the next sub-section.

Time integrators of identical von Neumann stability properties

This section shows that two time integrators of identical stability properties in the sense of the von Neumann stability analysis may have different behaviors



Figure 5.16: Iso-convergence curves: marginal stability (—), error halved every 10 time steps (—) , error divided by 10 every 10 time steps (—), error divided by 100 every 100 time steps (– –) and, as a reference, critical stability of Euler-Explicit (—)



Figure 5.17: Influence of time integrator on marginal stability. Euler explicit (_____), RK2 Midpoint (_____) , RK2 (---), RK3 (---).

regarding the buoyancy numerical instability. In particular, the RK2 midpoint and RK2 time integrators defined in appendices D.2 and D.3 are considered here.

Following von Neumann stability analysis, the generic equation

$$\frac{\partial q}{\partial t} = Rhs(q) \tag{5.66}$$

is being integrated for each individual harmonic q_{λ} of q, where λ is its wavelength. For each wavelength λ , the spatial von Neumann stability analysis provides a complex coefficient z_{λ} such that

$$Rhs(q_{\lambda}) = z_{\lambda} \ q_{\lambda} \tag{5.67}$$

Injecting this definition of the right-hand side in the algorithm of both RK2 time integrators (Algo. 16) (Algo. 17) yields the same definition for the gain over a time step

$$\left|\frac{q_{\lambda}^{n+1}}{q_{\lambda}^{n}}\right| = \left|1 + h_t z_{\lambda} + \frac{h_t^2}{2} z_{\lambda}\right|$$
(5.68)

for both RK2 time integrators, as was to be expected. Hence although they have different definitions and Butcher tables, both still share the same stability properties in the sense of the von Neumann stability analysis.

However, in the case of the buoyancy instability, the RK2 midpoint scheme behaves as

$$D_{t}v'^{n+1} = -\frac{2}{3}\mathcal{S}\left(D_{t}v'^{n}\frac{\partial\ln(\rho)}{\partial x}\right) + \frac{1}{3}\mathcal{S}\left(\mathcal{S}\left(D_{t}v'^{n}\frac{\partial\ln(\rho)}{\partial x}\right)\frac{\partial\ln(\rho)}{\partial x}\right)$$
(5.69)

whereas the other RK2 scheme behaves as

$$D_{t}v'^{n+1} = -\frac{3}{4}\mathcal{S}\left(D_{t}v'^{n}\frac{\partial\ln(\rho)}{\partial x}\right) + \frac{1}{4}\mathcal{S}\left(\mathcal{S}\left(D_{t}v'^{n}\frac{\partial\ln(\rho)}{\partial x}\right)\frac{\partial\ln(\rho)}{\partial x}\right)$$
(5.70)

Hence they have different stability properties in regards to the buoyancy instability, despite having identical stability properties in the sense of von Neumann. An interesting research topic would then be to know for a given order of Runge-Kutta time integrator, what is the highest mass density ratio that can be achieved, and what is the definition of the corresponding time integration scheme.

5.6.3 Influence of the mass density profile

The mass density profile is characterized by two things: firstly, by the mass density values ρ_{-} and ρ_{+} within both phases Ω^{-} and Ω^{+} which are fixed by the problem to be solved. Secondly by its shape, which is chosen by the user.

In particular, the profile defined in equation (Eq. 5.33) and used so far in this chapter has been chosen arbitrarily. Other profiles are possible, and perhaps may encourage or weaken the buoyancy instability.

Additionally, iso-convergence curves obtained so far consider mass density ratios ρ_+/ρ_- . However, those curves are only meaningful if they are valid for any value of the left mass density ρ_- .

The current section hence studies the influence of the shape of the mass density profile and of the left mass density value on the instability.

Influence of the shape of the mass density profile

In this sub-section, three shapes of the $\frac{\partial \ln(\rho)}{\partial x}$ function are tested. The motivation is to see if, for a given mass density ratio ρ_+/ρ_- imposed by the study case to simulate, there can be better profiles than others. Firstly, the $\frac{\partial \ln(\rho)}{\partial x}$ profile corresponding to a mass density profile taken as the integral of a cosine mollifier

$$\delta_{\epsilon}(x) := \frac{1}{2} (1 + \cos(2\pi x))$$
(5.71a)

$$H_{\epsilon}(x) := \int_{x=-1/2}^{x} \delta_{\epsilon}(x) dx = \frac{1}{2} + x + \frac{1}{2\pi} \sin(2\pi x)$$
(5.71b)

$$\rho(x) := \rho_{-} \left(1 - H_{\epsilon}(x) \right) + \rho_{+} H_{\epsilon}(x)$$
(5.71c)

$$\frac{\partial \ln(\rho)}{\partial x} = \frac{\frac{\partial H_{\epsilon}(x)}{\partial x}}{\frac{1}{2}\frac{\rho + + \rho_{-}}{\rho_{+} - \rho_{-}}} + H_{\epsilon}(x) - \frac{1}{2}$$
(5.71d)

Secondly, a $\frac{\partial \ln(\rho)}{\partial x}$ profile in the shape of a cosine

$$\rho(x) := \sqrt{\rho_{-} \rho_{+}} \exp\left(\frac{1}{2} \ln(\rho_{-}/\rho_{+}) \sin(-\pi x)\right)$$
(5.72a)

$$\frac{\partial \ln(\rho)}{\partial x} = \frac{1}{2} \pi \cos(\pi x) \ln\left(\frac{\rho_+}{\rho_-}\right)$$
(5.72b)

Eventually a constant $\frac{\partial \ln(\rho)}{\partial x}$ profile

Ì

$$\rho(x) := \rho_{-} \exp\left(\left(x + \frac{1}{2}\right) \ln\left(\frac{\rho_{+}}{\rho_{-}}\right)\right)$$
(5.73a)

$$\frac{\partial \ln(\rho)}{\partial x} = \ln\left(\frac{\rho_{+}}{\rho_{-}}\right) \tag{5.73b}$$

Figure 5.18 plots the marginal stability curve of each of those three profiles in the case of an Euler-Explicit time integrator. One can see that both the mass density profiles $\rho(x)$ shaped as the integral of a cosine and such that $\frac{\partial \ln(\rho)}{\partial x}$ is a cosine perform best. Indeed for any value of the under-relaxation factor α they allow for the highest mass density ratios ρ_+/ρ_- . On the other hand, the mass density profile $\rho(x)$ shaped such that $\frac{\partial \ln(\rho)}{\partial x}$ is flat performs the worst. The difference is small for unexisting or weak under-relaxation (α close to 1), but grows up to almost three orders of magnitude difference for an under-relaxation factor α of 0.1. Note however, that using a flat profile for $\frac{\partial \ln(\rho)}{\partial x}$ would not be acceptable in practical simulations because of the discontinuities it introduces.



(c) Atwood number

Figure 5.18: Marginal stability curves of the Euler-Explicit time integrator for different measures of "unstableness". Each measure is used over three shapes of color function: mass density as integral of cosine (Eq. 5.71) (---), $\frac{\partial \ln(\rho)}{\partial x}$ is constant (Eq. 5.73) (----).

On the other hand, the two other mass density profiles $\rho(x)$ would both be acceptable, and despite having different definitions they perform about as well.

To summarize, the choice of the mass density profile do affect the maximum admissible mass density ratio ρ_+/ρ_- but in a negligible way. In order to stabilize a method it would be much more effective to resort to other means, such as changing the time integrator or the under-relaxation factor α used. Besides, the baroclinic term is not the only term affected by the choice of the mass density profile. For instance the viscous term of the vorticity equation (Eq. 5.4) is also a function of mass density. Hence if a mass density profile is to be chosen then its influence on the viscous term is to be considered very seriously.

5.6.4 Measuring the "unstableness" of a method

Iso-convergence plots (Fig. 5.16) (Fig. 5.17) aim to show the under-relaxation factor α required to run a simulation of a certain "unstableness". So far this unstableness has been measured by the ratio of mass density ρ_+/ρ_- . This section aims at assessing whether it is indeed a good measure.

Amongst the parameters that have an influence on the instability, some have a minor influence (such as the shape of the mass density profile), whereas others strongly affect it (such as the time integrator or the under-relaxation factor α). Ideally, the measure of unstableness should be insensitive to the low-influence parameters while displaying a difference for high-influence parameters. A good measure of unstableness must hence have a weak coupling with the shape of the mass density profile $\rho(x)$. That way, an iso-convergence curve will be valid for any profile of mass density $\rho(x)$.

The approximate harmonic study of the Euler-Explicit buoyancy instability (see section 5.5) suggests that the instability growth rate is strongly dependent on the maxima of the $\frac{\partial \ln(\rho)}{\partial x}$ function (Eq. 5.65). This maxima could hence be an interesting measure of the unstableness of the method. Additionally two other options can be considered: the mass density ratio $\frac{\rho_+}{\rho_-}$ or the Atwood number $A = \frac{\rho_+ + \rho_-}{\rho_+ - \rho_-}$. Figure 5.18 plots the iso-convergence curves in the case of an Euler-Explicit

Figure 5.18 plots the iso-convergence curves in the case of an Euler-Explicit time integrator using as abscissa each of those three measures. None of the three measures appears to be completely uncoupled with the shape of the mass density profile $\rho(x)$. Nevertheless, the ratio of mass densities (Fig. 5.18b) appears to be less sensitive than the maxima of $\frac{\partial \ln(\rho)}{\partial x}$ (Fig. 5.18a). Also, the Atwood number (Fig. 5.18c) is not an appropriate measure because it compresses the curves towards A = 1 for high mass density ratios. This does not allow an easy reading of the curves. Moreover, at high mass density ratios, the quantity $\frac{1}{1-A}$ is almost equal to the mass density ratio $\frac{\rho_+}{\rho_-}$. In other words, plots (Fig. 5.18c) and (Fig. 5.18b) are two different ways to display almost the same information.

It hence appears that the ratio of mass densities ρ_+/ρ_- is indeed a good measure of "unstableness", although it is not perfect.

On a side note, remark that the mass density ratio $\frac{\rho_+}{\rho_-}$ is directly connected to the integral of $\frac{\partial \ln(\rho)}{\partial x}$ over the interfacial transition region Γ_{ϵ}

$$\frac{\rho_{+}}{\rho_{-}} = \exp\left(\int_{-\epsilon}^{+\epsilon} \frac{\partial \ln(\rho)}{\partial x} \, dx\right) \tag{5.74}$$

Hence the preference of using the mass density ratio $\frac{\rho_+}{\rho_-}$ over $\max\left(\frac{\partial \ln(\rho)}{\partial x}\right)$ to measure instability corresponds to assessing that the integral of the $\frac{\partial \ln(\rho)}{\partial x}$ function is more strongly related to the instability's growth than its maxima.

5.7 Summary

A numerical instability exists in the parabolic form of the vorticity equation (Eq. 5.4). It is caused by a strong coupling between the vorticity that this equation integrates in time and the material acceleration $\frac{D\boldsymbol{u}}{Dt}$ embedded in the baroclinic term of the same equation. In practice, the instability occurs as the guessed material acceleration $\frac{D\boldsymbol{u}}{Dt}$ greatly exceed the gravity acceleration \boldsymbol{g} in the falling phase.

Any problem with sufficiently large mass density ratios ρ_+/ρ_- is potentially subject to this issue. The instability depends strongly on the mass density ratio $\frac{\rho_+}{\rho_-}$ and on the time integrator used. It is not influenced by the time step value (assuming it is constant from a time step to the next) unlike most numerical instabilities encountered in CFD. Eventually, two time integrators of identical stability properties according to a von Neumann analysis might have a different behaviour on the buoyancy instability, with one being more stable than the other. It would be interesting to study what is the most stable Runge-Kutta time integrator of a given order and how far in the mass density ratios $\rho_+/\rho_$ it can go.

A harmonic analysis of the instability can be useful to capture the general behavior and to estimate the order of magnitude of the critical mass density ratio ρ_+/ρ_- . However it only describes very coarsely the phenomenon. Instead, a 1D model has been devised that allows a very good correlation with 2D simulations. This model describes the instability through a sequence of functions. The recursion rule of the sequence is changes depending on the time integrator used.

An under-relaxation of the 2D numerical method is proposed that allows higher mass density ratios ρ_+/ρ_- . The 1D model is extended to predict the stability of such simulations. "Iso-convergence" curves have been plotted for four popular time integrators. They allow to select the under-relaxation factor α required to achieve a given stability decay rate at the mass density ratio of the problem at hand. Eventually, a general method has been given to obtain the sequence describing the behavior of any Runge-Kutta time integrator.

Chapter 6

Numerics and solver validation

6.1 Overview of the benchmarks

Before discussing numerical results, the present section gives an overview of the benchmarks that will be used, what they allow to assess and the motivations for their selection. Also it is explained why the dam break benchmark will not be used, despite its popularity in the CFD literature. Eventually, note that Figure 6.1 shows a graphical description of the benchmarks, which is designed to help the reader in their reading. Moreover, Table 6.3 indicates the non-dimensional numbers for each of the benchmarks.

6.1.1 Benchmarks assessing the interface capturing method

The Zalesak benchmark introduced by Zalesak in 1979 [201] performs a solid rotation on a bubble shaped as a slotted disk. After one revolution, the shape of the bubble is compared to what it used to be initially. This allows to see how well the level set advection equation performs under rotating flows. The slot in the bubble is quite challenging to model because of the small space separating each sides of the slot and of the sharp angles it introduces in the geometry. On the other hand, the outer part of the bubble is quite wide and regular which allows to see how well smooth shapes are advected. Hence this benchmark indicates well the performance of the level set advection for most bubble shapes.

The vortex-stretched bubble benchmark introduced by Rider et al. in 1995 [202] advects a circular bubble by a vortex flow. The bubble is being stretched and rolled around the domain center for a time lapse δt . After that, the flow is reverted and the bubble is brought back to its original form. Comparing the final and initial bubbles allows to evaluate the loss of mass caused when the bubble is being stretched by the flow. The most stretched shape reached by the bubble just before reverting the flow also indicates how well the level set method models the trailing edge of the bubble. Eventually, the influence of the

reinitialization procedure on the loss of mass and artificial smoothing of the bubble can be investigated.

6.1.2 Benchmarks assessing the computation of surface tension

The static surface tension benchmark [203] [147] considers a spherical bubble in a gravity-free space. Analytically, the initial condition corresponds to the permanent regime and one expects the bubble to stay still. However, because of discretization errors the solvers will generate a spurious velocity field. Its magnitude gives an indication on the accuracy of the method. Moreover, the pressure jump across the interface δp does not match exactly the one given by Laplace law

$$\delta p = \frac{4\sigma}{d} \tag{6.1}$$

with d the diameter of the bubble. The deviation between the numerical prediction of the pressure jump and its value given by the Laplace law is often used to assess the accuracy of the method additionally to the intensity of spurious currents.

The standing wave benchmark [178] [204] [205] starts with an interface disturbed in a sinusoidal shape. At simulation start the surface tension puts the interface in motion thereby generating a standing wave, while viscous effects damp those oscillations which flattens the interface. This benchmark studies the interaction of the surface tension term with viscous effects. It is very similar to the oscillating bubble benchmark [206] which is also very popular in the literature. However, the standing wave benchmark has an analytical solution given by Prosperetti [207], which the oscillating bubble does not have. For that reason, the standing wave benchmark was preferred over the oscillating bubble benchmark.

6.1.3 Benchmarks on complex flows

The Rising Bubble benchmark [208] [108] studies an initially still and circular bubble which is being pushed upwards by buoyancy effects. This benchmark evaluates the Navier-Stokes implementation in high viscosity flows with an emphasis on the modelling of the surface tension term. The velocity of the bubble's ascent is checked against reference data. Also, surface tension effects are important in that benchmark and the shape taken by the bubble at a given time is studied and compared to a reference. Experimental data exists in 3D but the experiment is impossible to conduct in 2D. Therefore, our results are compared to the self-convergence limit of simulations run by various codes from different laboratories. To be specific, the reference we use is the work by Hysing and al. in 2009 [208].

The Rayleigh-Taylor benchmark considers, as its name suggests, a Rayleigh-Taylor instability [209] [210]. This benchmark has been used for a very long time, as Baker et al. 1980's article suggests [84]. Initially, a heavier fluid is placed above a lighter fluid. In the real world the interface cannot be exactly

flat, which triggers the instability. On the other hand, in the numerical simulations, the interface separating both phases is slightly perturbated such that the problem is unconditionally unstable. The actual Rayleigh-Taylor instability starts with a linear increase of the perturbation's amplitude, with gives birth to a succession of "fingers" of heavier fluid that penetrate the lighter fluid. The fingers grow individually then interact with each other. The distance between fingers is characteristic of the spacial frequency of the instability. In the Rayleigh-Taylor benchmark however, only one finger is simulated in a free-slip box such that no interaction phenomenon can occur. Instead what is studied is the evolution of the finger's shape as it falls. There are two variants of the Rayleigh-Taylor benchmark: the classic benchmark (for instance [84]) that does not have surface tension, and a variant introduced by Gomez et al. in 2005 [177] with surface tension. The case without surface tension witnesses the apparition of fine and complex flow structures and interface shape. It is hence very sensitive to solver errors since deviations introduced in the early stages of the fall will have strong repercussions in the later shape of the interface. On the other hand, in the second case surface tension prevents very fine structure to appear by keeping the phase in one lump. Therefore it is less sensitive to solver errors. Nevertheless, the interface shows pointy corners on which it can be challenging to evaluate the surface tension term. Hence the Rayleigh-Taylor simulation with surface tension is an excellent way to assess the accuracy of the surface tension term. Note that the Reynolds number at stake are an order of magnitude greater than those reached in the Rising Bubble benchmarks (Table 6.3). Both cases are studied in Gomez et al.'s work [177], and therefore the latter will be used as a reference.

The dam break benchmark has been considered [108] [211] but will not be used for validation. This benchmark considers a closed vessel containing in a corner a column of fluid of higher mass density than the surrounding fluid (Fig. 6.1d). Upon simulation start, the column of fluid collapses and spreads in the domain until the heavier fluid eventually occupies the lower part of the domain. This benchmark evaluates highly advective flows with negligible surface tension. Strong topology changes also occur in that benchmark with the breakup of droplets and a plunging wave. However, this benchmark has major drawbacks. Firstly, several experimental bias are present in the original 1952 article by Martin and Moyce [211], in particular a diaphragm is being burned using strong electric currents, however it is not clear whether the whole diaphragm vanishes or if an unburned part of it is being carried away with the flow. Additionally, the authors are not certain of the exact time when the diaphragm ruptures ¹ therefore their plots may be offsetted in time. In more recent experimental works [212] [213] [214] the diaphragm being burned is replaced by a partitioning wall that is being lifted very quickly. However doing so lifts the line of contact between water, air and dam thereby distorting the shape of the water region. All in all, the dam break experiments performed

¹From [211]: "No doubt further improvements in technique could be made, but in the results tabulated here the times have been normalized so as to give, for any given record in a series, the same time reading at a finite extent of spread. This time is of the order of magnitude of that which elapsed between application of the heating current and attainment of the degree of spread in question."

so far and that are known to us can be useful to give coarse information on the solution but they cannot be used to accurately validate code.



Figure 6.1: Initial conditions for each benchmark (\longrightarrow) velocity field, (\square) interior region Ω^- , (\square) exterior region Ω^+ and (•) is the domain's origin

	Zalocak	VSB	Static bubble Standing wave Rising Bubble Rayleigh Taylor								
	Latesak	VSD	Α	В	A	В	Α	В	С	Α	В
Ref	[201]	[153]	[147]	[203]	[178]	[178]	[208]	[208]	N/A	[177]	[177]
L	1.00	1.00	1.00	0.05	2π	2π	1.00	1.00	1.00	0.50	0.50
H	1.00	1.00	1.00	0.05	2π	2π	2.00	2.00	2.00	4.00	4.00
d	0.30	0.30	0.50	0.02	-	-	0.50	0.50	0.50	-	-
a	0.25	0.25	-	-	-	-	-	-	-	-	-
h	0.25	-	-	-	-	-	0.50	0.50	0.50	-	-
w	0.05	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	$\frac{2\pi}{100}$	$\frac{2\pi}{100}$	-	-	-	0.05	0.05
λ	-	-	-	-	2π	2π	-	-	-	1.00	1.00
t_{end}	-	8.0	-	-	-	-	-	-	-	-	-
\pmb{u}_{ini}	(Eq. 6.37)	(Eq. 6.39)	0	0	0	0	0	0	0	0	0

Table 6.1: Benchmarks' general parameters

 Table 6.2: Benchmarks' physics parameters.

Benchmarks noted with (*) were convergence analyses and hence several meshes of various refinment were used.

	Static	bubble	Stand	ing wave	Rising Bubble			Rayleigh Taylor	
	Α	В	A	В	A	В	С	A	В
g	0	0	0	0	0.98	0.98	0.98	9.81	9.81
σ	0.357	0.01	2	2	24.5	1.96	2.5	0	0.1337
ρ_{-}	4	1	1	1000	100	1	50	0.1694	0.1694
$ ho_+$	4	1000	1	1	1000	1000	1000	1.225	1.225
μ_{-}	1	$1e^{-5}$	$0.0647 \\ 20863$	0.0647 20863	1	0.1	0.05	$3.13e^{-3}$	$3.13e^{-3}$
μ_+	1	$1e^{-3}$	$0.0647 \\ 20863$	$0.0647 \\ 20863$	10	10	0.50	$3.13e^{-3}$	$3.13e^{-3}$
BC left	Slip	Slip	Slip	Slip	Slip	Slip	Slip	Slip	Slip
BC right	Slip	Slip	Slip	Slip	Slip	Slip	Slip	Slip	Slip
BC bot.	Slip	Slip	Slip	Slip	No-slip	No-slip	No-slip	Slip	Slip
BC top	Slip	Slip	Slip	Slip	No-slip	No-slip	No-slip	Slip	Slip
Mesh	(*)	(*)	(*)	(*)	128x256	256×512	512x1024	128 x 1024	128×1024

 Table 6.3: Benchmarks' dimensionless numbers

	Static bubble		Standing wave		Risi	ing Bu	Rayleigh Taylor		
	А	В	А	В	A	В	С	A	В
L_{ref}	d	d	$\lambda/2$	$\lambda/2$	d	d	d	λ	λ
Re	0	0	1.1	0.47	9.8	11	230	140	100
Eo	0	0	0	0	18.0	225	186	-	-
Mo	0	0	-	-	$6.00e^{-4}$	1.30	$3.72e^{-6}$	-	-
Α	0	0.998	0	0.998	0.818	0.998	0.905	0.757	0.757

6.2 Definition of criteria

Each benchmark consists in comparing the output of the VPM solver with either analytical solutions or reference results from the literature. That comparison can be done qualitatively, for example by comparing graphically the shapes of bubbles. However, for some benchmarks criteria are defined that measure features of the flow, for instance mass conservation or average bubble velocity, so as to allow quantitative comparison as well.

6.2.1 Criteria used in benchmarks

The general definitions of the criteria used by all benchmarks are presented here. Their numerical calculation will be considered in the next section.

Volume conservation

All simulations considered within this thesis are multiphase incompressible flows without phase change. Hence, the volume of each phase should ideally remain constant throughout simulations. The volume conservation criterion, also often referred to as "mass conservation", allows to measure the increase or decrease of the volume of each phase. This measure is performed by computing for a given phase (either the interior Ω^- or exterior Ω^+ region) the ratio between the phase's current and original volumes

$$\epsilon_{mass}^{-} := \frac{|\Omega^{-}(t)|}{|\Omega^{-}(t=0)|}$$
(6.2a)

$$\epsilon_{mass}^{+} := \frac{|\Omega^{+}(t)|}{|\Omega^{+}(t=0)|}$$
 (6.2b)

where $|\cdot|$ is the Lebesgue measure : if V is a solid, then |V| is its volume

$$|V| = \int_{V} 1 \, dx \tag{6.3}$$

and $|\partial V|$ is its surface area

$$|\partial V| = \int_{\partial V} 1 \, dx \tag{6.4}$$

Bubble center

The position of the bubble's center of gravity \boldsymbol{x}_c is obtained by calculating its barycenter

$$\boldsymbol{x}_c := \frac{1}{|\Omega^-|} \int_{\Omega^-} \boldsymbol{x} \, dx \tag{6.5}$$

Average bubble velocity

Similarly, its average velocity \boldsymbol{u}_c is calculated as the average of the velocity field other the bubble's volume

1

$$\boldsymbol{u}_c := \frac{1}{|\Omega^-|} \int_{\Omega^-} \boldsymbol{u} \, dx \tag{6.6}$$

Bubble sphericity

Additionally, its sphericity Sph is defined as the ratio between the surface area of bubble Ω^- and that of a ball $B_{eq}(V)$ of equal volume $|\Omega^-| = |B_{eq}(\Omega^-)|$

$$Sph(V) := \frac{|\partial B_{eq}(V)|}{|\partial \Omega^{-}|} \tag{6.7}$$

According to the isoperimetric theorem, the shape of largest surface area for a given volume is the ball, and hence the isoperimetric inequality $B_{eq}(\Omega^{-}) \leq \Omega^{-}$ is always true at all times. Therefore the sphericity measure cannot exceed one.

Also note that since the simulation may lose mass, the volume of the interior region Ω^- may change through time and hence the radius of the equivalent ball B_{eq} must be reevaluated anytime sphericity is to be evaluated.

Height of fluid column

The dam break benchmark measures the height h_{col} and surge distance d_{col} of the column of fluid Ω^- at time t. h_{col} is defined as the y-coordinate of the first point on the y-axis to be within the other phase Ω^+

$$h_{col} := \min_{y} \left(\left\{ y : y \in \Omega^+ \right\} \cup \{H\} \right) \tag{6.8}$$

while d_{col} is set to be the x-coordinate of the first point on the x-axis to be within the other phase Ω^+

$$d_{col} := \min\left(\left\{x : x \in \Omega^+\right\} \cup \{L\}\right) \tag{6.9}$$

6.2.2 Numerical calculation of criteria

Most criteria introduced in the previous section require the computation of integrals over volumes \int_V or surface areas $\int_{\partial V}$, where V denotes any subset of the domain $V \subset \Omega$. So far, analytical expressions have been given. However, they can be evaluated numerically in several ways. The present section makes a list of numerical methods to do so, and compares their respective accuracy.

Ways to compute volume, surface area and volume integrals

The surface area $|\partial V|$ can be computed: (i) using an interface mollifier

$$|\partial V| \simeq \int_{\Omega} \delta_{\epsilon} dx,$$
 (6.10)

or (ii) using a PLIC-like² reconstruction algorithm which consists in representing the interface as a straight line on each cell of the grid.

The volume |V| can be computed: (i) using a smeared color function

$$|V| \simeq \int_{\Omega} \chi_{V,\epsilon} \, dx, \qquad (6.11)$$

²Piecewise Linear Interface Calculation



Figure 6.2: Methods to compute volume and/or surface area

(ii) using an interface mollifier³

$$|V| \simeq \frac{1}{\dim(\Omega)} \int_{\Omega} \delta_{\epsilon} \boldsymbol{x} \cdot \hat{\boldsymbol{n}} dx,$$
 (6.12)

(iii) using a PLIC-like reconstruction algorithm or (iv) using a threshold i.e. each cell is assumed to be either completely in the interior region or completely in the exterior region.

$$|V| \simeq \operatorname{cardinal}\left(\left\{(i,j) \in \Omega_{\boldsymbol{h}} : \phi_{i,j} < 0\right\}\right) h_x h_y \tag{6.13}$$

The different ways to compute volume are shown graphically on Figure 6.2 in the case where the volume V is an ellipse. All four cases use exactly the same ellipse. The first subfigure (Fig. 6.2a) shows a smeared characteristic field used in the characteristic function method (Eq. 6.11). The black region corresponds to the ellipse. Note the blurry edges of the ellipse due to the smearing of the characteristic field. The second subfigure (Fig. 6.2b) shows the mollifier field δ_{ϵ} used in the mollifier method (Eq. 6.12). Notice how the mollifier is maximum on the ellipse's contour and goes progressively to zero away from it. Then the third subfigure (Fig. 6.2c) shows the PLIC method. Notice how the contour of the ellipse is not smooth but in fact made of a succession of short straight lines. Eventually the fourth subfigure (Fig. 6.2d) shows the threshold method (Eq. 6.13). Notice the squarish shape of the method which is due to the fact that grid cells are either completely white or black because of the thresholding.

Eventually the integral $\int_V q \, dx$ of a quantity q over a volume V can be computed in multiple ways. Yet the fields that are being integrated are fairly smooth (e.g. velocity). Hence considering the value of the field at the center of each cell appears satisfactory

$$\int_{V} q \, dx \simeq \sum_{(i,j)\in\Omega_{\mathbf{h}}} q_{i,j} h_x h_y \tag{6.14}$$

³The proof for this definition is shown in Appendix F.5.1.

Performance comparison of measures of volume and surface area

In order to find which algorithm is best, each is tested on a few shapes that are shown in Figure 6.3. Each shape has a specificity of its own, which allows to evaluate the accuracy of the algorithm in such conditions : (i) the cube is polygonal shape with sharp angles (Fig. 6.3a), (ii) the ellipse is a smooth shape (Fig. 6.3b), (iii) the drop has a spike which implies the presence of a strong "shock" in the level set field (Fig. 6.3c), and eventually (iv) the three spheres are held close together which means the smeared characteristic functions may not have enough space to spread (Fig. 6.3d). All shapes have been slightly decentered and rotated on the grid so that no symmetry property may affect the results.

All shapes have an analytical expression for their volume and surface area, with the exception of the ellipse whose circumference cannot be determined by an exact calculation. An approximation of that value which sufficient decimals is obtained by using the convergent series

$$\pi (R_1 + R_2) \left(1 + \sum_{n=1}^{\infty} \left(\binom{0.5}{n} \left(\frac{R_1 - R_2}{R_1 + R_2} \right)^{2n} \right) \right)$$
(6.15)

where R_1 and R_2 are the ellipse's major and minor radii respectively $R_1 \ge R_2$. From that reference data, a relative error can be computed between the numerical evaluations of volume or surface tension and the analytical solution

$$\epsilon_{rel} = \frac{|V_{num}|}{|V_{ref}|} - 1 \tag{6.16a}$$

$$\epsilon_{rel} = \frac{|\partial V_{num}|}{|\partial V_{ref}|} - 1 \tag{6.16b}$$

The relative errors for each of the methods mentioned above are presented in Figure 6.4. The first subfigure (Fig. 6.4a) looks at the accuracy of the methods to compute volume, whereas the second subfigure (Fig. 6.4b) considers methods evaluating surface areas. Each gray column corresponds to a different method for computing errors. For a given method, several different accuracy results have been obtained depending on the shape used (Fig. 6.3). The result of each shape is marked using a circle, diamond, triangle or square. Then the range between the most and least accurate of those methods is emphasized by the gray bar, which span between both values. What is of interest to us is to compare the maximum relative error of all methods. The best method is the one with the smallest of those maximas. Also it is interesting to observe the spread of the gray bar. A method with a short bar will consistently output results of comparable accuracy, whereas a method with a tall gray bar will return results that can sometimes be excellent and sometimes very poor.

Looking at the results, the methods with the lowest maximal error are plic and chr1 for volume (Fig. 6.4a) and mol1 for surface area (Fig. 6.4b). Hence those methods are most accurate and should be preferred. However, in practice it appears that values obtained with chr1 and mol1 tend to vary too greatly between time steps, therefore plic and mol2 will be preferred to chr1 and mol1



Figure 6.3: Shapes whose volume and/or surface area will be computed. (---) interface Γ and (---) the level set field.



(b) Relative errors in surface area computation

Figure 6.4: Relative errors in the computation of surface areas and volumes using several algorithms. () represents the span of the relative error : the upper (resp. lower) edge of the bars is the maximum (resp. minimum) relative error. Columns chr1 to chr8 correspond to computations performed using a smeared color function over a length scale of 1 to 8 grid nodes. Columns mol1 to mol8 follow a similar idea but using a mollifier instead. Column plic uses a PLIC-like interface reconstruction and thrs applies a threshold. Eventually (squares), (circles), (triangles) and (diamonds) correspond respectively to the square (Fig. 6.3a), ellipse (Fig. 6.3b), drop (Fig. 6.3c) and three spheres (Fig. 6.3d) benchmarks

respectively. More generally, note how the maximum error tends to increase with the spread of the smeared characteristic functions and mollifiers: chr8 and mol mol8 have a higher maximum relative error than chr1 and mol1. This tendency can be observed both when computing volumes (Fig. 6.4a) and surface areas (Fig. 6.4b). Also, note that mol1 to mol8 have in general greater relative errors and spread than their chr1 to chr8 counterparts. Hence computing volumes using the mollifier method (Eq. 6.12) appears to be up to an order of magnitude less accurate than using a characteristic function (Eq. 6.11). To summarize, PLIC will be used to compute volumes and integrals over volumes, while mol2 will be used to compute surface areas.

6.3 Dimensionless numbers

A coarse characterization of the benchmark Simulations can be done by computing non-dimensional numbers. This allows in particular to know how much influence viscosity or surface tension have on the structure of the flow, amongst other things. In all the benchmarks considered here several dimensionless numbers may be considered : the Reynolds number

$$Re = \frac{U_{ref} L_{ref}}{\nu_{ref}} \tag{6.17}$$

that indicates whether a flow is dominated by viscous effects $(Re \to 0)$ or by inertial effects $(Re \to +\infty)$, the Eötvös number

$$Eo = \frac{\|\rho_{+} - \rho_{-}\| \|\boldsymbol{g}\| L_{ref}}{\sigma}$$
(6.18)

that indicates whether a steady bubble is maintained spherical because of dominant capillary forces $(Eo \rightarrow 0)$ or if it is strongly affected by gravity $(Eo \rightarrow +\infty)$, the Morton number

$$Mo = \frac{\|\boldsymbol{g}\| \ \mu_{+}^{4} \ |\rho_{+} - \rho_{-}|}{\rho_{+}^{2} \ \sigma^{3}}$$
(6.19)

that gives an indication on the shape of a bubble rising in a column of fluid [215], and the Atwood number

$$A = \frac{|\rho_{+} - \rho_{-}|}{\rho_{+} + \rho_{-}} \tag{6.20}$$

that is the dimensionless number indicating the growth rate of Rayleigh-Taylor instabilities $(A \rightarrow 0 \text{ indicates slowly growing instability}, whereas <math>A \rightarrow 1 \text{ indicates a fast growing instability}$. The dimensionless numbers relevant to each case are summarized in Table 6.3. Most of these dimensional numbers use fluid properties that can easily be read in the simulations' parameters. The Reynolds number, however, requires to choose a reference velocity U_{ref} , which is much less straightforward. The following of the present section hence presents and justifies how the Reynolds number is calculated for each benchmark.

Reynolds number for the standing wave benchmark

In the standing wave benchmark, the velocity is decaying with time as viscous effects damp the surface oscillations, and hence so does the Reynolds number. Therefore, the average interfacial velocity at the time t_{flat} when the fluid interface first becomes flat is taken as the reference velocity U_{ref} . Moreover, the half wavelength is taken as reference length instead of full wavelength because the shear occurs between both halves of the domain. Eventually, note that for Prosperetti's analytical solution (Eq. 6.45) to work, the kinematic viscosity must be constant throughout the domain.

$$Re = \frac{\max_{\boldsymbol{x}\in\Gamma} \left(\|\boldsymbol{u}(t_{flat})\|\right)\frac{\lambda}{2}}{\nu}$$
(6.21)

١

Reynolds number for the Rising Bubble benchmark

In the case of the Rising Bubble benchmark, the Reynolds number describes the balance between the inertial effects of the bubble at steady state (constant rising velocity) in relation to the friction caused by the surrounding fluid. The kinematic viscosity in the interior region influences whether the bubble behaves as a solid or not and does not describe the friction effects. Instead, it is the viscosity of the exterior region that is representative of the friction caused by the surrounding fluid. Hence, the reference velocity U_{ref} is the final rising velocity of the bubble \boldsymbol{u}_c while the reference kinematic viscosity is thee of the surrounding fluid

$$Re = \frac{\|\boldsymbol{u}_c^-\|\,d}{\nu^+} \tag{6.22}$$

Reynolds number for the Rayleigh-Taylor benchmark

In the case of the Rayleigh-Taylor benchmark, the interface is originally still then the heavier phase penetrates the lighter phase by gravity at increasing velocities. Hence the Reynolds number increases with time. Hence the number taken in (Table 6.3) is the Reynolds number at time t = 0.9, for the penetrating phase i.e. the heavier phase. Hence the velocity is measured at the lowest interface point and the kinematic viscosity is the one of the surrounding fluid i.e. the lighter phase. Additionally, the reference length is the wavelength of the initial perturbation, which also corresponds to the wavelength of the generated fluid structures.

$$Re = \frac{\left\| \boldsymbol{u} \left(\min_{\boldsymbol{x} \in \Gamma} \boldsymbol{x} \right) \right\| \lambda}{\nu^{-}}$$
(6.23)

6.4 An Eulerian velocity-pressure solver to compare to

The performance of the VPM solver developed in this thesis is compared to the performance of a more traditional Eulerian velocity-pressure solver for incompressible flows. This solver will be named "VeloGrid". It works on the momentum equation expressed in the Eulerian frame of reference

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)(\boldsymbol{u}) = -\frac{\nabla p}{\rho} + \boldsymbol{g} + \frac{1}{\rho} \nabla \cdot \left(\mu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right) \right) + \boldsymbol{F}_{\Gamma} \quad (6.24)$$

which it discretizes using finite differences on a Cartesian grid.

The incompressibility condition $\nabla \cdot \boldsymbol{u} = 0$ is satisfied by using a velocity projection method which operates in three steps. Firstly, the pressure p is assumed constant within the time step

$$p(t) = p^n \quad \forall t \in]t^n, t^{n+1}[$$
(6.25)

and based on that assumption the momentum equation is integrated over a time step

$$\boldsymbol{u}^{*n+1} \leftarrow \boldsymbol{u}^{n} + \int_{t^{n}}^{t^{n+1}} \frac{\partial \boldsymbol{u}^{*}}{\partial t} dt$$
 (6.26)

In other words, the unprojected velocity field \boldsymbol{u}^* is the solution of the initial value problem

$$\frac{\partial \boldsymbol{u}^{*}}{\partial t} + (\boldsymbol{u}^{*} \cdot \nabla)(\boldsymbol{u}^{*}) = -\frac{\nabla p^{n}}{\rho} + \boldsymbol{g} + \boldsymbol{F}_{\Gamma} + \frac{1}{\rho} \nabla \cdot \left(\mu \left(\nabla \boldsymbol{u}^{*} + \nabla \boldsymbol{u}^{*T} \right) \right)$$
(6.27a)
$$\boldsymbol{u}^{*}(t^{n}) = \boldsymbol{u}(t^{n})$$
(6.27b)

whose equation (Eq. 6.27a) differ from the momentum equation (Eq. 6.24) solely by the fact that the pressure term is constant in time. Additionally, the unprojected velocity field \boldsymbol{u}^* has the same boundary conditions than the actual (solenoïdal) velocity field \boldsymbol{u} . Secondly, the error in pressure $\delta p := p^{n+1} - p^n$ is found by solving the elliptic problem

Find
$$\delta p$$
 such that $\begin{cases} \nabla \cdot \left(\frac{\nabla(\delta p)}{\rho}\right) = \frac{\nabla \cdot \boldsymbol{u}^{* n+1}}{h_t} & \text{in } \Omega\\ \nabla(\delta p) = 0 & \text{on } \partial\Omega \end{cases}$ (6.28)

Eventually, using the pressure error δp , the velocity field \pmb{u}^* is projected in the space of solenoïdal fields \pmb{u}

$$\boldsymbol{u}^{n+1} \leftarrow \boldsymbol{u}^{*n+1} - \frac{h_t}{\rho} \nabla(\delta p)$$
(6.29)

Note that many projection methods exist. In particular, Brown et al. compared several of them in their 2000 article [216]. Also, some projection methods have been devised specifically for multiphase flows with mass density jumps across the interface, such as the second-order method proposed by Puckett et al. in 1997 [217]. Nevertheless, those advanced methods will not be considered, and instead the simpler method described above will be used.

Eventually, it captures the interface location using the level set method, which it solves on the grid. Details on the algorithm of VeloGrid can be read in Appendix C.2.

6.5 Numerical parameters of the solver

Both the VPM and VeloGrid solvers use the following parameters:

- The time step h_t is adaptive, and its value is changed from a time step to the other such that both Lagrangian CFLs (the one based on rotation and the one based on strain) as well as the mesh Fourier number are below 0.1. The increase in the time step value is capped by 10%. In the case of the VeloGrid, the Lagrangian CFL is replaced by the CFL.
- The smeared fluid properties are mollified over a half-thickness ϵ equal to 8 grid cells
- The local level set method uses a mask half-thickness of 14 grid cells. This is actually larger than the minimum value of 11, but was chosen so for safety.
- The remeshing is performed at an adaptative frequency such that one remeshing is performed every $\frac{0.1}{\max(LCFL)}$.
- The level set field is reinitialised when the level set field is detected as being too distorted using the method described in section 3.2.6.

Moreover, Velogrid and the elliptic variant of VPM both use the GMRES solver of the Hypre library for their elliptic equations on pressure. On the other hand, the parabolic variant of VPM uses a FFT solver based on the FFTW library. Recall that FFT solvers can only be used on equations with constant coefficients, hence the use of GMRES for equations with variable coefficients.

6.6 Numerical investigations

Previous chapters have discussed several numerical phenomena, some of which are to be tested on the benchmarks that were just introduced. Those results are therefore gathered in the present section.

6.6.1 Choice of level set reinitialization strategy

The influence of the choice of the criterion triggering reinitialization discussed in section 3.2.6 is shown in Figure 6.5 for the first Rayleigh-Taylor benchmark (see section 6.9). More specificly, it shows results of the Rayleigh-Taylor benchmark at t = 1.0 using several level set reinitialization criteria. As can be seen, both reinitialising every 50 time steps (Fig. 6.5a) or reinitialising if the level set is too distorted (Fig. 6.5c) yield similar results. On the other hand, reinitialising a little bit every time step (Fig. 6.5b) is clearly inferior. Moreover, one drawback of performing full reinitialisations every n time steps is that it is not obvious which value to choose for n. The optimal value might be different depending on the simulation considered, and it might even change within a single simulation. For example, in the case of the Rayleigh-Taylor benchmark, one might want larger values for n at the beginning of the simulation when the flow structure is



Figure 6.5: Influence of the reinitialization strategy, shown for the first Rayleigh-Taylor benchmark at t = 1.0 on a 128x1024 grid. (—) is the shape of the interface for the given reinitialization strategy obtained with the

VPM solver, while (—) is the reference data from Gomez [177].

still simple, and smaller values later in the same simulation when the geometry of the fluid interface has become complex. On the other hand, the "if too distorted" criterion does not require the user to set any parameter, and at the same time guarrantees that the gradient of the level set is within 0.9 and 1.1. For those reasons, the "if too distorted" criterion will be used in the rest of the present thesis.

6.6.2 Choice of the surface tension term

Section 4.1 discussed the different possible formulations of the surface tension term.

Figure 6.6 presents two simulations of the Rising Bubble B benchmark: one (Fig. 6.6a) with the form of (Eq. 4.6) whereas the other (Fig. 6.6b) employs the form of (Eq. 4.7). In both cases, the results obtained are nonphysical because of a two-dimensional filtering that has been added at each time step. This numerical behavior is further discussed in Section 4.1.1. However, despite being unphysical, those simulations are interesting because they emphasize the grid alignement of the fully expanded form (Eq. 4.7). In particular, it appears clearly that the fully expanded formulation (Eq. 4.7) artificially aligns the interface on the mesh, which is to be avoided. Hence the other, partially expanded



Figure 6.6: Influence of the expression of surface tension term on simulation output.

formulation (Eq. 4.6) will be preferred.

6.6.3 Appearance of trenches

Section 4.1.1 discusses the appearance of trenches due to isotropic 2D filtering. Figure 6.8 displays the same simulation as in Figure 6.7b but for earlier times at which the trenches are starting to develop themselves. On the first subfigure (a) one can see a slight pinching of the interface at the end of the bubble's skirt. This pinching lengthens and initiates other pinches along itself (Fig. 6.8b) develops. The same phenomena occurs for each of the new edges which leads to a more (Fig. 6.8c) and more (Fig. 6.8d) complex patterns of trenches, which eventually leads to the nonphysical results commented in the previous paragraph (Fig. 6.7b).

6.6.4 Choice of tangential filter for the level set field

Section 4.1.1 discussed three different expressions of tangential filter for the level set field. The present section assesses the performance of each.

In order to emphasize the strengths and weaknesses of each of those three tangential filters, the second Rising Bubble benchmark is run (see section 6.9) until t = 0.1 with a filtering performed at each time step. The resulting interfaces shapes are shown in Figure 6.9 for each of the three implementations. Both divergence forms (Fig. 6.9b) (Fig. 6.9c) of the filter preserve mass much better but at the cost of introducing nonphysical displacement of the interface. Indeed, in the case of the normal-averaged divergence form (Fig. 6.9c) the bubble has become squarish. On the other hand, the steps or wiggles introduced in the interface by the face-computed normals divergence-form filter (Fig. 6.9b), although much smaller are still problematic. Eventually, we note that the



Figure 6.7: Influence of filtering on benchmark's outputs, shown at t = 3.0 on a 256x512 grid with a full reinitialization every 50 time steps.



Figure 6.8: Development of "trenches" caused by normal filtering

direct form filter (Fig. 6.9a) does preserve the spherical shape of the bubble although it looses more mass than its divergence form competitors. Moreover the bubble shape that it yields corresponds to the one obtained when using a first order two-dimensional filter (Fig. 4.6b), which tends to indicate that the direct-form filter does behave as the tangential variant of the two-dimensional discrete filter. Therefore the direct-form tangential filtering is preferred.

The correctness of the direct-form tangential filter is also to be assessed on a full simulation. For that purpose, the second Rising Bubble benchmark is run, but with a surface tension value set to zero. Setting it to zero allows to compare with the non-filtered variant of the solver, which cannot handle surface tension. The results are shown in Figure 6.7. This same figure has been used before to show the nonphysical results obtained with the standard two-dimensional filter (Fig. 6.7c). Note that the solver using the tangential filter is not subject to the development of trenches (Fig. 6.7c). Moreover, the results obtained are consistent with those obtained using the non-filtered solver (Fig. 6.7a).

6.6.5 Influence of missing term in Thirifay's formulation

Section 4.2.1 presents Thirifay's formulation for the viscous term of the vorticity equation and points out that one term is missing. The present paragraph



Figure 6.9: Tangential filtering methods shown in their respective first order variants for the second Rising Bubble (see section 6.9) simulation on a 256x512 grid at t = 0.1

discusses the influence of this term on the results. Figure 6.10 compares the results of a Rising Bubble A simulation obtained with Thirifay's formulation and the corrected formulation. As can be seen, Thirifay's formulation does not yield good results, indicating that the term it misses plays an important role in the physics of the flow.

In this chapter, the code is validated over several benchmarks. First the Zalesak and vortex-stretched bubble benchmarks are performed to test the level set method. Secondly, the spurious current, oscillating wave benchmarks are performed to test the implementation of the surface tension. Thirdly, the full solver is tested over the Rising Bubble and Rayleigh-Taylor benchmarks. Eventually the now validated solver will be used on simulations relevant to user-cases of the nuclear industry.

6.6.6 Influence of under-relaxation on simulations' output

As stated previously, under-relaxing not only prevents the instability from developing but also damps physical oscillations of the velocity field. Previously (see section 5.4.1) it has been conjectured that since buoyancy effects are slowly changing in time, the under-relaxation will have little to negligible effects on the flow field. The present section assesses this statement.

To do so, the first rising bubble benchmark (see section 6.9) is run using the VPM solver with different under-relaxation coefficient values. The outputs can then be compared to observe the influence of under-relaxation. This simulation has a moderate mass density ratio of 10, for which under-relaxation is unnecessary. This is done on purpose so as to assess the influence of a wide range of under-relaxation coefficients : 1, 1/2, 1/4, 1/10. The time integrator used here is Runge-Kutta 3 D.4. Otherwise, the same parameters are used as in the first rising bubble benchmark (see section 6.9).



Figure 6.10: Influence of the viscous term definition on the Rising Bubble benchmark. Original Thirifay definition (.........), form (Eq. 4.37) (......), form (Eq. 4.38) (---), form (Eq. 4.36) (......) and reference from Hysing et al. [208].

Figure 6.11 shows the shape of the bubble Ω^- at t = 3 as well as three criteria that are quantified measures of the simulation's quality: ascension velocity defined as the average vertical velocity over the bubble's volume

$$u_c = \frac{1}{|\Omega^-|} \int_{\Omega^-} (\boldsymbol{u} \cdot \hat{\boldsymbol{e}}_y) \, dx, \qquad (6.30)$$

the sphericity defined as the ratio between the perimeter of the bubble $\partial\Omega^{-}$ and the perimeter of a circular bubble of equal volume ∂B_{eq} , and the volume conservation which is the ratio between the bubble's current volume and its initial volume

It is obvious from the plots (Fig. 6.11) that the under-relaxation coefficients have had little influence on the outputs of the simulation. The only visible change that can be seen is on the sphericity criterion (Fig. 6.11b) for very small under-relaxation values. And still, the sphericity criterion is very sensitive, in practice the plots of the bubble's shape seem to overlap (Fig. 6.11c).



Figure 6.11: Influence of under-relaxation coefficient on a rising bubble simulation output. $\alpha = 1.0$ (---), 0.5 (---), 0.25 (----) and 0.1 (----). Results are compared to Hysing's [208] (----)

6.6.7 Comparison of elliptic and parabolic vorticity solvers

The vorticity equation

$$\frac{D\omega}{Dt} = -\frac{\nabla p}{\rho} \times \frac{\nabla \rho}{\rho} + \nabla \times \left(\frac{1}{\rho} \nabla \cdot \boldsymbol{\tau}\right) + \nabla \times \boldsymbol{F}_s.$$
(6.31)

embeds a pressure-dependent term, but does not provide a way to determine this pressure directly. Instead, the momentum equation on velocity is used

$$\frac{D\boldsymbol{u}}{Dt} = -\frac{\nabla p}{\rho} + \frac{1}{\rho} \nabla \cdot \boldsymbol{\tau} + \boldsymbol{F}_s \quad \text{on } \Omega$$
(6.32)

It can be employed in two ways. So far in this thesis, it has been used to substitute the pressure gradient in the baroclinic termfrom

$$\frac{D\omega}{Dt} = \left(\frac{D\boldsymbol{u}}{Dt} - \boldsymbol{g}\right) \times \frac{\nabla\rho}{\rho} + \frac{1}{\rho} \nabla \times (\nabla \cdot \boldsymbol{\tau}) + \nabla \times \boldsymbol{F}_s.$$
(6.33)

However, it can also be used to determine pressure through the resolution of an elliptic problem, and then use this pressure field directly in the vorticity equation

Find
$$p$$
 s.t.
$$\begin{cases} \nabla \cdot \left(\frac{\nabla p}{\rho}\right) = -\underbrace{\frac{D}{Dt}(\nabla \cdot \boldsymbol{u})}_{=0} + \nabla \cdot \left(\frac{1}{\rho} \nabla \cdot \boldsymbol{\tau}\right) + \nabla \cdot \boldsymbol{F}_s & \text{on } \Omega\\ \nabla p \cdot \hat{\boldsymbol{n}} = 0 & \text{on } \partial\Omega \\ (6.34a) \end{cases}$$

Inject
$$p$$
 in vorticity eq. $\frac{D\omega}{Dt} = -\frac{\nabla p}{\rho} \times \frac{\nabla \rho}{\rho} + \nabla \times \left(\frac{1}{\rho} \nabla \cdot \boldsymbol{\tau}\right) + \nabla \times \boldsymbol{F}_s.$
(6.34b)

The first approach (Eq. 6.33) will be referred to as the parabolic problem, and the second (Eq. 6.34) as the elliptic problem. Both approaches have strengths and weaknesses. The general accuracy of both approaches is now studied and compared. Following that, some general caracteristics of both approaches will be summarized.

Figure 6.12 compares the results of the elliptic and parabolic solvers on the Rising Bubble A benchmark. As can be seen both yield almost identical results. Only a very slight shift in sphericity can be observed at times around and beyond t = 3.0 (Fig. 6.12c), and all other indicators including ascension velocity (Fig. 6.12b) and mass conservation (Fig. 6.12e) are undistinguishable. Also no observable difference can be seen on the bubble itself (Fig. 6.12a).

Figure 6.13 shows the results of the elliptic solver for the Rising Bubble B benchamrk and compares it to the results obtained with the parabolic VPM and the reference from Hysing et al. [208]. Figure 6.14 presents the time-evolution of the bubble at four different times. Unlike the Rising Bubble A benchmark, the elliptic variant of the VPM solver performs poorly whereas the parabolic version performs well. Indeed, the shape of the bubble is very


Figure 6.12: Results of the elliptic VPM method on the Rising Bubble A benchmark. Shows the results of the elliptic variant of the VPM solver (---) and of its parabolic version (---) against reference data by Hysing et al. [208] (----).



(a) Bubble at t = 3.0. The reference curve (——) is an average of the three slightly different results from Hysing et al. 2009 [208].



Figure 6.13: Results of the Rising Bubble B benchmarks. ($-\!-\!-$) is the elliptic VPM solver, (---) the parabolic VPM solver, and ($-\!-\!-$) is the reference data from Hysing et al. 2009 [208].



Figure 6.14: Rising Bubble B simulation ($\rho_+/\rho_- = 1000$) using the elliptic VPM solver.

different (Fig. 6.12a), and so is its ascention velocity (Fig. 6.12b) which is about half of the reference value. Note also that the shape of the interface of the bubble is flatter at certain angles. This behaviour has been traced back to the viscous term of the vorticity equation. Whereas it is defined as

$$\frac{1}{\rho} \nabla \times (\nabla \cdot \boldsymbol{\tau}) \tag{6.35}$$

in the case of the parabolic solver (Eq. 6.33), it is

$$\nabla \times \left(\frac{1}{\rho} \, \nabla \cdot \, \boldsymbol{\tau}\right) \tag{6.36}$$

for the elliptic solver (Eq. 6.34). In other words, the viscous term of the elliptic solver embeds the mass density within the curl operator. In the case of the Rising Bubble B benchmark, there is a ratio of a thousand between the mass density within the bubble and that of the surrounding fluid. Therefore the $1/\rho$ function behaves as a steep and offsetted step function. This appears to be the root cause of the disappointing results observed in Figures 6.30 and 6.13.

Figure 6.16 shows the results obtained for the Rising Bubble C benchmark on a fine mesh (512x1024). A full time-evolution of the simulation is visible on Figure 6.15. It is an intermediate simulation that has a mass density ratio of 20. It appears that both the results obtained using the parabolic and elliptic variants of the VPM agree well with each other. This confirms the influence of the mass density on the quality of the results.

In complement to those results, and their implications on the accuracy of both methods, some general caracteristics are now pointed out.

The main advantage of the parabolic problem is that it is much faster to compute, since it does not require an iterative solver. Additionnally, it allows



Figure 6.15: Rising Bubble C, shown with the elliptic variant of the VPM solver.



Figure 6.16: Bubble at t = 4.0 on the Rising Bubble C benchmark.

to express the source terms of momentum in a simple and elegant way. On the other hand, the elliptic method embeds several terms that describe the same physical phenomena: for instance there is one viscous term in the equation on pressure and one in the vorticity equation (note that the same argument can be made for the surface tension term). This duplication of source terms not only increases the numerical complexity of the method, but they also spark additional interogations: can both terms interact with each other? If so, how? Is there some additional care required in the numerical discretization due to the existance of those two terms?

However, the parabolic problem also has one major drawback: the purpose of the vorticity equation is to evaluate the time derivative of vorticity $\frac{\partial \omega}{\partial t}$, yet its baroclinic term embeds the time derivative of velocity $\frac{\partial \boldsymbol{u}}{\partial t}$, which is directly connected to $\frac{\partial \omega}{\partial t}$. Thus, in order to compute the time evolution of vorticity, one uses a guess of the same information albeit expressed in terms of velocity. This

causes the numerical instability on the baroclinic term that was studied earlier in Chapter 5. Besides the stability issue there also is a concern on accuracy: how strongly does this coupling influence the accuracy of the results?

To conclude, it appears that neither the parabolic nor the elliptic approaches are ideal. The parabolic approach embeds a strong coupling on the time derivative of vorticity or velocity, while the elliptic approach has a complicated formulation for its source terms of momentum. However, numerical simulations show that the parabolic solver behaves better at high mass density ratios. Hence it should be prefered over the elliptic variant of the VPM solver.

6.7 Validation of the VPM level set method

In this section and the following ones, the results of the benchmarks are presented and commented. Also additional information is given on the benchmarks when necessary. In the present section, the performance of the level set implementation of the VPM solver is assessed. Note that both benchmarks enforce each a specific velocity field, hence the part of the solver integrating the momentum equation is disabled.

Zalesak benchmark

The Zalesak benchmark performs a 360 degrees rotation of a bubble around an axis. Figure 6.17 shows an example of such simulation: the bubble starts from the top is initially shaped as a slotted disk (Fig. 6.17a), then it is moved by a velocity field

$$u = \|\boldsymbol{x}\|\,\boldsymbol{\hat{e}}_{\theta} = -R\,\sin(\theta)\,\boldsymbol{\hat{e}}_x + R\,\cos(\theta)\,\boldsymbol{\hat{e}}_y \tag{6.37}$$

corresponding to solid rotation (Fig. 6.17b) (Fig. 6.17c) with angle θ taken from $\hat{\boldsymbol{e}}_x$ to $\hat{\boldsymbol{e}}_{\rho}$. Eventually, it is brought back to its original location (Fig. 6.17d) and the shape of the bubble is compared to the original shape (Fig. 6.17a).

Note that this velocity field crosses the domain boundary, which in the case of the VPM code corresponds to advecting some particles from outside the domain to inside it. Nevertheless, the flow of interest here is the one around the Zalesak bubble. Therefore, a domain four times larger than usual Zalesak benchmarks is used and the level set field is forcibly saturated beyond a certain radius to the domain center. Eventually, the time step for the VeloGrid simulations have been chosen such that $CFL \simeq 0.55$.

Figure 6.18 shows the shape of the interface after a 360 degrees rotation around an axis, and compares it to its initial shape before the rotation (---)

. The left subfigure compares the VPM solver against the VeloGrid solver, whereas the right subfigure shows the influence of the remeshing frequency.

On the left subfigure (Fig. 6.18a), one can see that the VPM solver (---) preserved the interface much better than the VeloGrid solver (---). In particular, notice how the slot of the Zalesak disk and its circumference is better preserved.

On the right subfigure (Fig. 6.18b), notice that the slot is better preserved by the simulations with less frequent remeshing. Recall however, that the



Figure 6.17: An example of Zalesak simulation, shown with VPM and a remesh every time step.

remeshing is a necessary operation that needs to be performed once in a while in order to prevent Lagrangian distortion.

Vortex-stretched bubble (VSB)

In the vortex-stretched bubble benchmark, a circular bubble (Fig. 6.19a) is carried away and stretched by a vortex flow (Fig. 6.19b) (Fig. 6.26d). Then, the flow is reverted (Fig. 6.19d) such that the bubble takes its original form (Fig. 6.19e). The advecting velocity field is such that the corresponding stream function is

$$\psi(\boldsymbol{x},t) := \frac{UL}{\pi} \cos^2\left(\pi \frac{x}{L}\right) \cos^2\left(\pi \frac{y}{L}\right) \cos\left(\pi \frac{t}{t_{end}}\right)$$
(6.38)

with U = 1 and L = 1. In other words, the velocity field is defined as

$$\boldsymbol{u}(\boldsymbol{x},t) = U\left(-\cos^{2}\left(\pi\frac{x}{L}\right)\cos\left(\pi\frac{y}{L/2}\right)\boldsymbol{\hat{e}}_{x} + \cos^{2}\left(\pi\frac{y}{L}\right)\cos\left(\pi\frac{x}{L/2}\right)\boldsymbol{\hat{e}}_{y}\right)\cos\left(\pi\frac{t}{t_{end}}\right)$$
(6.39)





(a) Comparison of VPM (-) against VeloGrid (- -), with the analytical solution as a reference (-). Performed on a 96x96 grid (7.2 grid nodes over the disk's radius).

(b) Influence of remeshing frequency: remeshed every 1 ($\cdots \cdots \cdots$), 2 (---) and 4 (---) time steps, with the analytical solution as a reference (----).

Figure 6.18: Results of the Zalesak benchmark



Figure 6.19: An example of Vortex-stretched bubble simulation, shown with VPM with reinitialization.

Figures 6.20 and 6.21 show the results of the Vortex-stretched bubble benchmark. On each of those figures two stages of the same simulation are shown: the left subfigure shows the bubble in its most stretched out form i.e. just before reversing the flow, and the right subfigure displays the shape of the bubble after a full reversing of the flow and compares it to its original shape (_____) . The reference data in the left subfigures is the self-convergence limit of the solver and was obtained on a 512x512 mesh.

The first pair of figures (Fig. 6.20) compares the performance of the VPM solver as compared to VeloGrid and to the self-convergence limit. Notice that the VPM solver preserves the trailing tail of the bubble much better than VeloGrid (Fig. 6.20a). After reversing the flow, it appears clearly that the VPM solver preserved mass much better than VeloGrid (Fig. 6.21a). Notice also that after reversing both VPM's and VeloGrid's bubbles are quite similarly shaped (Fig. 6.21a).

The second pair of figures (Fig. 6.21) studies the influence of reinitialization

on the results of the VPM solver. In practice, a simulation of the VPM with reinitialization disabled (---) has been run. Notice on the plot before flow reversing (Fig. 6.21a) that the trailing tail of the bubble is rounded in the case of the reinitialized simulation (---), whereas it is spiked in the non-reinitialized case (---). This artificial rounding of the bubble is hence a consequence of the interface displacements caused by level set reinitialization. On the same plot both bubbles are roughly of the same volume, although the bubble in the non-reinitialized case seems of slightly smaller volume than in the reinitialized case. This difference in volume is confirmed in the right subfigure (Fig. 6.21b). In the latter, note also that the bubble of the non-reinitialized case (---) preserved its rounded shape whereas it is more strongly distorted in the reinitialized case (---) and no longer resembles an ellipsoid. Once again this is a consequence of artificial interface displacement caused by level set reinitialization.

Eventually, a convergence study is conducted to assess the convergence of the level set particle method on the vortex-stretched bubble benchmark. More specifically, it is assessed whether the shape of the interface at the final time point t = 8.0 converges to the analytical solution as the grid resolution increases.

To do so, the first half of the simulation (bubble stretching) is performed using marker particles, then the level set field is reconstructed from those marker particles, and eventually the level set particle method is used to perform the second half of the simulation (bubble unstretching). Beside reducing the simulation cost, performing the first half of the simulation using marker particles allows to better capture the distortions generated by the level set particle methods. Indeed, since this particular benchmark uses a velocity field that reverts itself, it is well possible that some of the distortions generated in the first half of the simulation are being cancelled by identical but opposite distortions appearing in the second half. Instead, by starting from the half of the simulation using a "trusted" level set field, all distortions generated in the course of the simulation will appear at the final time.

Initially, The marker particle algorithm seeds 1000 particles X_p on the interface, then those particles are advected by time-integrating the ODE

$$\frac{\partial \boldsymbol{X}_p}{\partial t} = \boldsymbol{u}(\boldsymbol{X}_p) \tag{6.40}$$

using an Euler-Explicit scheme and a time step value h_t of $1e^{-6}$. In equation 6.40, the velocity field $\boldsymbol{u}(\boldsymbol{X}_p)$ is known analytically from equation 6.39. Of course higher-order time integration schemes could have been chosen, yet the Euler-Explicit time integrator yields sufficiently accurate and fast results for that time step value. Indeed, before using the results from the marker particles half-simulation, it has been shown that a full marker particle simulation using those parameters preserves very well the shape of the bubble. This is demonstrated on Figure 6.22b. The latter shows the interface shape at the initial condition (—), and its shape after stretching and unstretching using the marker particles method (—). Both curves superpose each other very



Figure 6.20: Vortex-stretched bubble benchmark: comparison of VPM (\longrightarrow) against VeloGrid (- - -), with the self-convergence solution (\longrightarrow) as reference.



Figure 6.21: Influence of reinitialization on the Vortex-stretched bubble benchmark: comparison of VPM with reinitialization (---) to VPM without reinitialization (---), with the self-convergence solution (----) as reference.

well. Therefore, it is trusted that the interface shape obtained at time t = 4.0 (Fig. 6.22a) using the same method and identical parameters is also very close to the analytical result.

The level set field is reconstructed from the marker particles in the following way: (i) each segment connecting two consecutive marker particles generates a level set field in the normal direction equal to the distance to the segment, (ii) where no level set value has yet been set by the segments, the value chosen is the distance to the closest marker particle. The algorithm used is very inefficient $(O(N^2) \text{ complexity})$ but sufficient for our needs.

The self-convergence result for the level set particle method is shown in Figure 6.23. The same parameters as for the benchmark itself have been used, except that the time step value is chosen such that the Lagrangian CFL is around



Figure 6.22: Marker particles method used on the Vortex-stretched bubble benchmark

0.1, and the remeshing frequency is adapted such that one remeshing is performed every $\frac{0.1}{LCFL}$ time steps. Those parameters are close to the parameters used in actual flow simulations. The plot shown has been obtained for a simulation without reinitialization on a 2048x2048 grid (Fig. 6.23a). An identical result is obtained using reinitialization with a "reinitialize when level set is too distorted" criterion (Fig. 6.23b). This particular grid resolution (2048x2048) has been chosen because it showed no difference with results obtained using a 1024x1024 grid. As can be seen on Figure 6.23, the level set particle method does not converge exactly to the analytical solution, yet the deviation is small.

Summary

The particle-based level set method embedded in the VPM solver yielded better results than its grid-based counterpart embedded in the VeloGrid solver, as noted previously by Hieber and Koumoutsakos in their 2005 work [18] on particle-based level set methods. In particular, a self-convergence study on the Vortex-Stretched Bubble has shown that the VPM method converged well, even with the remeshing procedure. Those results were obtained by starting the simulations from the most stretched-out form of the bubble, which was obtained using marker particles.

More generally, the good results of the VPM method are explained by the fact that the level set advection equation is a pure transport equation, which is an ideal case for using Lagrangian methods. Indeed, while the VeloGrid solver discretizes the advection term using finite differences, the VPM solver simply keeps its Lagrangian level set function constant in time. Eventually, it was noted that frequent remeshing and reinitialization deteriorates the accuracy of



Figure 6.23: Self-convergence result using level set particle method. The final interface shape at t = 8 (——) is compared to the initial condition (——)



Figure 6.24: Initial condition of the static bubble benchmark

the method.

6.8 Validation of surface tension

The performance of the VPM solver in pure level set benchmark is very appealing. However it remains to be assessed whether the VPM can bring comparable improvements in actual full-flow solvers. Yet, before moving to this topic, the resolution of surface tension effects must first be validated. This is performed firstly on a static benchmark, then on a dynamic case.

Surface tension on a circular bubble

Starting from an excatly circular bubble (Fig. 6.24), spurious currents appear that displace the interface away from its circular shape. The magnitude of this error is measured either through the intensity of the spurious currents, or by evaluating the difference in the pressure jump accross the interface with the one given by the Young–Laplace equation

$$\delta p = \sigma \kappa \tag{6.41}$$

where δp is the pressure jump across the interface, σ is the surface tension coefficient and κ is the curvature of the interface.

In the case of a vorticity solver however, and unlike a velocity-pressure solver, the pressure does not appear directly in the equations. Although it can still be computed through post-processing, it is not a meaningful measure of the accuracy of a vortex method. Therefore, only the intensity of the spurious currents is studied here.

Since the bubble is initially spherical, all currents that may appear are spurious currents that attempt at bringing the bubble to a non-spherical shape that is closer to the "numerical" equilibrium shape of the bubble. The intensity of those currents are measured using three different norms

$$E_1(q) := \frac{1}{|\Omega_{\boldsymbol{h}}|} \sum_{\boldsymbol{x}_{i,j} \in \Omega_{\boldsymbol{h}}} ||q_{i,j}||$$
(6.42a)

$$E_2(q) := \frac{1}{\sqrt{|\Omega_{\boldsymbol{h}}|}} \sqrt{\sum_{\boldsymbol{x}_{i,j} \in \Omega_{\boldsymbol{h}}} ||q_{i,j}||^2}$$
(6.42b)

$$E_{\infty}(q) := \max_{\boldsymbol{x}_{i,j} \in \Omega_{\boldsymbol{h}}} \|q_{i,j}\|$$
(6.42c)

where $|\Omega_{\mathbf{h}}|$ is the cardinality of the finite set $\Omega_{\mathbf{h}}$, in other words it is the number of nodes $\mathbf{x}_{i,j}$ on the grid $\Omega_{\mathbf{h}}$. Each of these norms is actually a vector norm that has been normalized such that the number of points of the mesh is of no influence on the measure's outputted value. A consequence of this normalization, as can be seen on Figure 6.25 is that

$$E_{\infty}(q) < E_2(q) < E_1(q)$$
 (6.43)

whereas

$$\|q\|_1 < \|q\|_2 < \|q\|_{\infty} \tag{6.44}$$

Two cases are considered: a case with a mass density ratio ρ_+/ρ_- of one, and a case with a ratio of 1000, following the same parameters as used respectively by [147] and [203]. Error convergence results are shown for each case on figures 6.25a and 6.25b respectively. In both cases, a steady convergence towards zero is observable for all three measures, with a second-order decay rate $O(h^2)$ for E_{∞} . Hence the implementation of the surface tension term is consistent in the steady case.



Figure 6.25: Convergence of spurious currents for the static bubble benchmark. Three measures of error in $\frac{\boldsymbol{u}^1 - \boldsymbol{u}^0}{t^1 - t^0}$ are shown: E_1 (——), E_2 (– – –) and E_{∞} (——).

Standing wave

The standing wave benchmark considers the damping of oscillations of an almost horizontal fluid interface subject to surface tension. Starting from a perturbated flat interface (Fig. 6.26a) (Fig. 6.26b), the latter is put in motion by surface tension (Fig. 6.26c), thus leading to an oscillatory behavior (compare Figures 6.26b, 6.26d, and 6.26f) that is damped by viscosity (compare Figures 6.26b and 6.26f). An exact solution of the problem has been found by Prosperetti first for a one-phase free surface problem [218], then on a two-phase fluid interface problem [207]. Indeed, this article proves that the amplitude aof the wave is given in time as

$$a(t) = \frac{4(1-4\beta)(\nu k^2)^2}{8(1-4\beta)(\nu k^2)^2 + \omega_0^2} a_0 \operatorname{erfc}\left(\sqrt{\nu k^2 t}\right) + \sum_{i=1}^4 \frac{z_i}{Z_i} \left(\frac{a_0 \,\omega_0^2}{z_i^2 - \nu k^2} - u_0\right)$$
(6.45)

where a_0 is the initial amplitude, u_0 the initial velocity, z_i are the four complex roots to the quartic equation

$$z^{4} - 4\beta\sqrt{\nu k^{2}}z^{3} + 2(1-6\beta)\nu k^{2}z^{2} + 4(1-3\beta)(\nu k^{2})^{3/2}z + (1-4\beta)(\nu k^{2})^{2} + \omega_{0}^{2} = 0$$
(6.46)

 \mathbb{Z}_i are four coefficients defined as

$$Z_1 := (z_2 - z_1) (z_3 - z_1) (z_4 - z_1)$$
(6.47a)

$$Z_2 := (z_3 - z_2)(z_4 - z_2)(z_1 - z_2)$$
(6.47b)

$$Z_3 := (z_4 - z_3)(z_1 - z_3)(z_2 - z_3)$$
(6.47c)

$$Z_4 := (z_1 - z_4) (z_2 - z_4) (z_3 - z_4)$$
(6.47d)



(a) Initial condition, shown on the full domain (t = 0.0).



Figure 6.26: An example of standing wave simulation, the y-axis has been zoomed-in to emphasize the oscillations.

 β is defined as

$$\beta = \frac{\rho_+ \rho_-}{(\rho_+ + \rho_-)^2} \tag{6.48}$$

using as reference time scale the inverse of the frequency of irrotational waves in deep water ω_0 obtained through the dispersion relation

$$\omega^{2} = \frac{\rho_{-} - \rho_{+}}{\rho_{-} + \rho_{+}} k g + k^{3} \frac{\sigma}{\rho_{-} + \rho_{+}}$$
(6.49)

where σ is the surface tension, ρ the mass density of the heavier fluid and k the wave number of the standing wave.

The number obtained through the amplitude equation (Eq. 6.45) is hence a complex number but its imaginary part happens to be zero. Also the amplitude equation (Eq. 6.45) is valid for any shape of the interface. In the case of our benchmark a sinusoïd of initial amplitude a_0 is used.

This benchmark is run in two cases: using equal mass densities $\rho_+/\rho_- = 1$, and in a high mass density ratio configuration $\rho_+/\rho_- = 1000$. Figures 6.27 and 6.28 show the results for each of those cases respectively. For each figure, the left subfigure shows the time-evolution of the wave's amplitude and compares it with the analytical result (—), while the right subfigure shows the absolute error in wave amplitude. In other words, the left subfigure is useful to have a qualitative evaluation of the method's performance, whereas the right subfigures gives a more quantitative information.

Firstly, note that for both mass density ratios (Fig. 6.27a) (Fig. 6.28a), the solver's amplitude predictions converges to the analytical solution. Similarly, the absolute error tends to converge to zero (Fig. 6.27b) (Fig. 6.28b). Also note that in the low mass density ratio case (Fig. 6.27b), the amplitude error is roughly divided by four every time the grid's resolution is doubled. Hence this scheme appears to be second-order. On the other hand, the convergence seems slower in the high mass density case (Fig. 6.28b).

Summary

Both steady and dynamic benchmarks validate the consistency of the method in handling surface tension effects. The static convergence appears to be second-order (at least in the L^{∞} norm). On the other hand, the convergence in the dynamic case is also of second-order for low mass density ratios, but it is lower for higher density ratio.



Figure 6.27: Standing wave $\rho_-/\rho_+=1$ case shown for three mesh refinements: 32x32 (---), 64x64 (---) and 128x128 (---). Those three curves are compared to the analytical reference (---).



Figure 6.28: Standing wave $\rho_-/\rho_+=1000$ case. Same legend as in Figure 6.27.

6.9 Validation of full solver

Now that the interface capturing and surface tension implementations of the VPM solver has been assessed, the present section continues that study using complex flow benchmarks. First the Rising Bubble benchmark is considered, then the Rayleigh-Taylor benchmark is studied.

Rising Bubble

The Rising Bubble benchmark is run in three variants : case A which is a low mass density ratio simulation ($\rho_+/\rho_- = 10$) (Fig. 6.29), case B which has a high mass density ratio ($\rho_+/\rho_- = 1000$) (Fig. 6.30), and case C that has an intermediate mass density ratio ($\rho_+/\rho_- = 20$). The difficulty with twodimensional Rising Bubbles is that they do not occur naturally. Hence there is no experimental studies to validate against. Instead both cases are compared with numerical results from Hysing et al. 2009 [208]. This work is interesting in that it compares the self-convergence limit of three distinct finite-element solvers (TP2D, FreeLIFE and MooNMD) developed by three different laboratories (TÜ Dortmund, EPFL Lausanne and Uni Magdeburg respectively). TP2D and FreeLIFE are Eulerian solvers and capture the fluid interface using the level set method. On the other hand MooNMD is based on an Arbitrary Lagrangian-Eulerian paradigm, however in the case of their study it was used as a fully Lagrangian method and the interface motion was tracked directly by the moving mesh. For the later regular remeshing was performed when necessary. Additionally, the output of three commercial CFD software (Ansys Fluent, CFX and Comsol) are very briefly presented. The outputs of all solvers agree well with one another, although in the Rising Bubble benchmark some differences in the results can be noted for large times. In the figures presented in the present section (Fig. 6.32) (Fig. 6.33) all three solvers are always represented by the same gray line (---). Indeed, what is relevant is not to compare the results of those three solvers between themselves, but merely to have an idea of the deviation between their output so as to better appreciate how different the VPM solver behaves.

Figures 6.32 and 6.33 presents the results for both cases of the Rising Bubble benchmark. In both cases, five plots are shown. Firstly, the shape of the bubble at t = 3 is compared qualitatively (a), then plots of the quantitative criteria previously defined are shown: ascension velocity (b), sphericity (c), vertical position (d) and mass conservation (e). In each plots, the VPM solver is compared to the VeloGrid solver and reference results from Hysing et al's work [208].

In the case of the first Rising Bubble benchmark (Fig. 6.32) all solvers show a very good agreement. Only the sphericity displays a visible difference (Fig. 6.32c). However, note that it is a very sensitive criterion. In practice, the shape of the bubbles is actually very similar (Fig. 6.32a)

In the case of the second Rising Bubble benchmark (Fig. 6.33), there is a visible difference between solvers although the agreement remains good. Indeed, on one hand there is some deviation in ascension velocity beyond t = 1.7 (Fig. 6.33b), which translates in a slight difference in vertical position



Figure 6.29: Rising Bubble A simulation ($\rho_+/\rho_- = 10$) using the parabolic VPM solver.



Figure 6.30: Rising Bubble B simulation ($\rho_+/\rho_- = 1000$) using the parabolic VPM solver.



Figure 6.31: Convergence analysis on Rising Bubble B benchmark

(Fig. 6.33d). Note however, that both the VeloGrid and VPM solvers share the same tendency in terms of ascension velocity. Concerning the sphericity, the deviation between VPM and reference solvers is negligible given the difference between the outputs of the three solvers (Fig. 6.33c). Eventually, there is a good agreement in the general shape of the bubble (Fig. 6.33a): all solvers show a bubble with a skirt. The length of the skirt tends to be slightly longer for the VPM and VeloGrid solver, while its height is shorter for those solvers.

Eventually, volume preservation is more or less the same for both the VPM and VeloGrid solvers on both cases (Fig. 6.32e) (Fig. 6.33e), with a slight edge in favour of the VPM solver.

Additionally, a convergence study has been performed on the Rising Bubble B benchmark. Results are shown in Figure 6.31. As can be seen, although the general shape of the bubble is similar between the results of the VPM simulations and the results obtained by Hysing et al. [208], the VPM solver does not converge exactly to the same solution.



Figure 6.32: Results of the Rising Bubble A benchmark. Shows the results of the VPM solver (---) and of the VeloGrid solver (---) against reference data by Hysing et al. [208] (----) .



(a) Bubble at t = 3.0. The reference curve (——) is an average of the three slightly different results from Hysing et al. 2009 [208].



Figure 6.33: Results of the Rising Bubble B benchmarks. Same legend as in Figure 6.32

Rayleigh-Taylor instability

The Rayleigh-Taylor studies the time-evolution of the Rayleigh-Taylor instability, and in particular the shape of the "finger" of heavier fluid that penetrates the lighter fluid. The problem is initialized with a heavier phase in the top half of the domain, and a lighter phase in other half. In order for the system to be unconditionally unstable, and hence to allow for the instability to grow, an initial perturbation must be introduced. This takes the form of a displacement of the interface, which is positioned at

$$y(x) = -A \cos\left(\frac{2\pi}{L}\frac{\lambda}{L}x\right)$$
(6.50)

where A is the amplitude of the perturbation, and λ its wavelength.

As stated before, Gomez et al's 2005 work [177] will be used as a reference. This choice is motivated by the fact that the parameters used there are also used in other articles from different authors. Indeed, Rayleigh-Taylor simulations have been used for benchmarking purposes since the eighties at the latest. However, most of the time the sets of parameters are unique to a single work. Additionally, Gomez et al's work consisted in assessing the performance and accuracy of a numerical finite difference method with adaptive quad-tree mesh in the vicinity of the interface. Hence the simulation results that were achieved are very accurate.

Figures 6.34 and 6.35 show the results of the Rayleigh-Taylor benchmarks. Here only the shape of the interface can be qualitatively compared, as there is no quantitative criterion. Indeed, each figure shows the interface at different times. The plots shows the results of the VPM and VeloGrid solvers, as well as reference data from Gomez et al. [177].

On each case of the benchmarks, both VPM and VeloGrid solvers agree well with the reference data. Note in particular, the fact that the VPM still shows the fluid filament in Figure 6.34e whereas on the same plot it has vanished from the VeloGrid results. This is surely a consequence of the particle-based level set method embedded in the VPM solver, but not in the VeloGrid solver.

Eventually, in both cases of the benchmark, it is difficult to tell which is more accurate between the VPM and VeloGrid solvers.



Figure 6.34: Results for the Rayleigh-Taylor Case A benchmark. Shows the results of the VPM solver (---) and of the VeloGrid solver (---) against reference data by Gomez et al. [177] (----) .



Figure 6.35: Results for the Rayleigh-Taylor Case B benchmark. Same legend as in Figure 6.34.

Summary

Despite superior interface capturing capabilities (see section 6.7), the VPM solver outputs results of similar accuracy than the VeloGrid solver in complex flows. However, that can still be considered an achievement given the fact that it is a novel solver (recall that multiphase VPM solvers are very uncommon). In particular, improvements over the existing implementation might give it an edge. That being said, the VPM passes all complex flow benchmarks, which is an important realization.

Now that the VPM solver's accuracy has been studied, its computational efficiency must be considered. This is the object of the following section.

6.9.1 Computational efficiency

The computational efficiency of the VPM solver is evaluated on the first Rising Bubble benchmark, and it is compared to that of the VeloGrid solver. Two studies are performed. Firstly, the algorithmic complexity of the method is considered, in order to see if the cost of the VPM method increases at the same rate than that of the VeloGrid solver as grid refinement increases. Secondly, VPM methods are known to be particularly advantageous on highly advective flow. Hence the cost of the VPM method is measured in several variants of the same simulation, some being more viscous or more advective than others.

For this study, the surface tension term has been disabled. This is motivated by the fact that its presence impacts considerably the computational cost of the method, while its current implementation (filtering of the level set field instead of staggered discretization) is not ideal and hence not representative of the capabilities of a multiphase VPM method.

Both plots on Figure 6.36 show the computational cost of VPM and Velo-Grid methods. All curves were obtained on the first case of the Rising Bubble benchmark. More precisely, the CPU time measured corresponds to the CPU time⁴ necessary for a solver to reach the time point t = 4. A faster simulation can be achieved either by computing less time steps (which implies larger time step values), or by decreasing the computational cost of each time step (e.g. by coarsening the mesh). Measures were performed on a computer with an Intel 2.90GHz i5-2450M processor and 4GB 1333MHz DDR3 RAM⁵. All simulations were run using a single process. The left figure shows the influence of mesh refinement (Fig. 6.36a), whereas the right figure shows the influence of the viscous or advective nature of the simulation on its computational cost.

Influence of mesh refinement

Figure 6.36a shows the CPU time of the method depending on the grid spacing h for both the VPM (---) and veloGrid (---) solvers. The idea here is to evaluate the average cost of computing a single time step. Hence each

 $^{^4\}mathrm{It}$ is taken as the sum of the CPU time spent by the process in user-mode and kernel-mode i.e. the sum of "user" and "sys" measures outputted by the GNU "time" command.

 $^{^{5}}$ The operating system was patched against the Spectre and Meltdown flaws which might alter its performance regarding memory access[219].





(a) Influence of mesh refinement at CFL/Frr = 0.28.

Figure 6.36: Computational efficiency of the VPM solver (with an admissible $LCFL_{adm}$ of 0.1) (—) in comparison to VeloGrid (– –).

simulation uses the same time step value of $h_t = 1 \cdot 10^{-4}$, which guarantees that they all require 40,000 time steps in total to reach the final time of $t_{end} = 4.0$. Aside from the surface tension coefficient that had been set to zero, the parameters used are the ones of the Rising Bubble A benchmark (Table 6.2), which imply a mesh Reynolds of 0.28.

Obviously, the finer the grid, the more grid nodes there are, and the more costly the simulations become. More interestingly, both solver show the same slope which suggests identical algorithmic complexity. Also note that the veloGrid solver is on average 33% cheaper than the VPM solver.

Influence of mesh Reynolds

Figure 6.36b plots the computational costs against the CFL to mesh Fourier ratio CFL/Frr, also known as the mesh Reynolds number. Low CFL/Frr ratios indicate that the mesh Fourier is the stronger stability criterion, and will enforce a stricter constraint on the time step. On the contrary, higher CFL/Frr ratios indicate that the CFL criterion is the strictest. Around one both criteria have similar influence, although the exact tipping point depends on the values chosen for the admissible CFL or mesh Fourier. Lower (higher) CFL/Frr ratios are achieved by increasing (decreasing) the dynamic viscosity of both phases such that the μ_+/μ_- ratio remains constant. Eventually, note that the simulations of high CFL/Frr ratios are under-resolved since no subgrid-scale turbulence models are used. However, they give a good indication of the cost of corresponding LES simulations.

Both methods have the same algorithmic complexity (Fig. 6.36a), but the gains vary greatly (Fig. 6.36b). Three regimes clearly show in Figure 6.36b.

Firstly, when the mesh Fourier Frr is the most stringent stability criterion

CFL/Frr < 1, the time step of both methods is constrained by the mesh Fourier which enforces

$$h_t = \nu h_x^2 Frr_{adm} \tag{6.51}$$

where the admissible mesh Fourier Frr_{adm} and the grid spacing h_x are constants. In other words the time step is proportional to the kinematic viscosity $h_t \propto \nu$ and hence to the mesh Fourier itself $h_t \propto Frr$. This proportional law is represented in the plot by a slope of minus one for CFL/Frr < 1. In this regime, the VeloGrid method is marginally cheaper than the VPM method, by around 30%.

Secondly, for flows where the CFL is the limiting stability criterion CFL/Frr > 1, the VeloGrid simulation cannot go below a 400 seconds cpu-time limit. Indeed, for low viscosities the bubble does not have enough time to reach its steady rising velocity. In other words, the viscous effects do not have enough time to grow sufficiently strong in order to affect the bubble's ascension velocity significantly. Hence for all high CFL/Frr ratios, the ascension velocity remains of the same order of magnitude. As a consequence, the time step value enforced by the CFL constraint

$$h_t = \max_{\boldsymbol{x}_{i,j} \in \Omega_{\boldsymbol{h}}} \|\boldsymbol{u}\| h_x CFL_{adm}$$
(6.52)

is proportional to velocity, and hence also remains of the same order of magnitude for all CFL/Frr ratios. Thus the 400 seconds limit. On the other hand, the VPM solver is not limited by the CFL number and hence can use greater time step values which results in still decreasing CPU times.

Thirdly, the VPM solver reaches a lower limit at 200 and 400 seconds CPU time, for an admissible Lagrangian CFL of 0.1 and 0.5 respectively 6.36b. This occurs when the Lagrangian CFL becomes the most stringent criterion for the time step value. Indeed, when the LCFL reaches its maximum admissible value, the time step cannot be reduced any further and hence the computational cost of the method remains constant. The further increase of the CPU cost for the VPM solver with LCFL = 0.1 is not explained by this phenomenon, but by other artifacts of the simulation.

Summary

To summarize, the VPM method is shown to be about 30% slower for highlyviscous flows but can become up to four times faster for highly advective flows. Also, both VPM and VeloGrid solvers have the same algorithmic complexity.

6.10 Simulations and extension to heat transfer

Now that the VPM code has been validated over well-established benchmarks, it is used on simulations similar to those that could be encountered in the nuclear sector. In this section, firstly the solver for heat transfer is presented. Then, the previously considered Rayleigh-Taylor A simulation is run again but with added heat conduction and advection effects. Eventually, a sodium-argon reactor pool simulation is run.

6.10.1 VPM method with heat transfer

In our problems, the temperature field is governed by the conservation law

$$\frac{DT}{Dt} = \frac{1}{\rho c} \nabla \cdot \lambda \, \nabla T \tag{6.53}$$

which is obtained from the conservation law on internal energy

$$\frac{De_i}{Dt} = -\nabla \cdot \boldsymbol{\varphi} + \boldsymbol{\tau} : \nabla \boldsymbol{u}$$
(6.54)

by neglecting the heat created by friction effects. Here c is the specific heat capacity, φ the heat flux density vector and τ the deviatoric component of the stress tensor. Recall that in the case of an incompressible flow the specific heat capacity at constant pressure c_p and at constant volume c_v are equal. Additionally, as a first step, the heat flux across the domain boundary is assumed to be zero.

In Heavy Liquid Metal nuclear reactors the thermal diffusivity

$$\alpha := \frac{\lambda}{\rho c} \tag{6.55}$$

is orders of magnitude greater than viscous diffusivity ν . This yields very low Prandtl numbers

$$Pr := \frac{\nu}{\alpha} \tag{6.56}$$

Table 6.4 shows some example values for various coolants typically encountered in HLM nuclear reactors as well as for argon which will be assumed to be the reactor's cover gas. Note that the thermal diffusivity of both coolant and cover gas phases is actually very similar, and instead it is the kinematic viscosity that is different. For instance, for a reactor using sodium as a coolant and argon as a cover gas, the thermal diffusivity α is respectively of $6.3 \cdot 10^{-5}$ and $9.6 \cdot 10^{-5}$ viscous diffusivity ν of $3.1 \cdot 10^{-7}$ and $6.3 \cdot 10^{-5}$ at a pressure of 1hPa and a temperature of 700K. Additionally, note that although the thermal diffusivity is very similar, there are large jumps in conductivity λ and thermal inertia ρ across the fluid interface.

Note that in flows with Prandtl numbers around one, the Reynolds analogy is often used to model subgrid-scale phenomena occurring in boundary layers. It consists in assuming that the thermal and viscous boundary layers share similar profiles. Practically, the same profile is used for velocity and temperature fields. However, this assumption no longer holds for low Prandtl flows, as are encountered for instance in the liquid metal coolant of HLM reactors. See in particular, Cheng's and Tak's 2006 work withch compares several models [220], or 2012 article by Bricteux Duponcheel Manconi and Bartosiewicz [221]. Nevertheless in the simulations performed, no model is used for the boundary layers and hence the issue does not arise.

Note also that in order for the numerical method to be stable, the time step value h_t needs to satisfy the mesh Fourier criterion. This is true for the momentum equation but also for the heat equation. However, since the thermal diffusivity α is one to two orders magnitude greater than the viscous diffusivity

Table 6.4: Fluid properties of some coolants and cover gas employed in HLM reactors at 1hPa and 700K

	Sodium	Lead	Bismuth	LBE	Argon
Thermal inertia ρc_p [$J.m^{-3}.K^{-1}$]	1 085 000	1 540 400	1 311 500	1 458 700	357.79
Thermal conductivity λ [$W.m^{-1}.K^{-1}$]	68.00	16.90	13.55	13.38	$0.034\ 27$
Thermal diffusivity α $[m^2.s^{-1}]$	$6.270 \cdot 10^{-5}$	$1.097 \cdot 10^{-5}$	$1.033 \cdot 10^{-5}$	$9.169 \cdot 10^{-6}$	$9.577 \cdot 10^{-5}$
Mass density ρ [kg.m ⁻³]	852	10531	9 871	10 169	0.687 4
Dynamic viscosity μ [Pa.s]	$2.644 \cdot 10^{-4}$	$2.095 \cdot 10^{-3}$	$1.350 \cdot 10^{-3}$	$1.451 \cdot 10^{-3}$	$4.348 \cdot 10^{-5}$
Viscous diffusivity ν $[m^2.s^{-1}]$	$3.105 \cdot 10^{-7}$	$1.990 \cdot 10^{-7}$	$1.368 \cdot 10^{-7}$	$1.427 \cdot 10^{-7}$	$6.338 \cdot 10^{-5}$
Pr	$0.005 \ 0$	0.018 1	$0.013\ 2$	$0.015\ 6$	0.666 7

 $\nu,$ then for a given admissible mesh Fourier number Frr_{adm} the admissible time step of the heat equation

$$h_{t,adm} = \frac{h_x^2 Frr_{adm}}{\alpha} \tag{6.57}$$

is one to two orders of magnitude lower than the admissible time step for the momentum equation

$$h_{t,adm} = \frac{h_x^2 Frr_{adm}}{\nu} \tag{6.58}$$

In other words, the solving of the heat equation considerably slows down the solver as a whole. This can be alleviated by solving the heat equation implicitly, for instance using an Euler Implicit or a Crank-Nicolson time integrator. In both cases, the heat equation (Eq. 6.53) becomes a screened Poisson problem that can be solved using an elliptic solver. Indeed, in the case of an Euler Implicit time integrator it becomes

$$\frac{h_t}{\rho c} \nabla \cdot \left(\lambda \, \nabla T^{n+1} \right) - T^{n+1} = -T^n \tag{6.59}$$

By using an implicit approach instead of an explicit one, the stability of the method no longer depends on the mesh Fourier for the heat equation. Instead, only the momentum equation's mesh Fourier and the Lagrangian CFL numbers influence the choice of the time step value. Therefore much larger time step values can be used.

Nevertheless, implicitly solving the screened Poisson problem (Eq. 6.59) requires the use of an elliptic solver. Such solver is in practice more computationally expensive than parabolic solvers used for the explicit problem (Eq. 6.53). This means that the cost of computing a single time step will be higher. But since there are much less time steps in the implicit method than in the explicit one, it becomes the fastest overall.

6.10.2 Influence of Prandtl number

The implicit solver (Eq. 6.59) for the heat equation is used on the Rayleigh-Taylor benchmark, with various Prandtl number values. Initially, the temperature of the top phase is set to 100K, and that of the bottom phase to 200K. With time, the temperature field is advected by the flow and also diffuses itself because of thermal conductivity.

Three simulations are run with different values of the conductivity λ : 1, 10 and 100 $W.m^{-1}.K^{-1}$. In all three cases, the thermal capacity is set such that the product of mass density and specific heat capacity ρc is equal to $54 \, kg.m^{-1}.K^{-1}.s^{-2}$. Both phases' parameters are identical. This yields thermal diffusion coefficients α of 0.00313, 0.0313 and 0.313 $m^2.s^{-1}$. Since the benchmark's kinematic viscosity ν is of 0.00313 $m^2.s^{-1}$, the three flows' Prandtl numbers are respectively 1, 0.1 and 0.01.

Figure 6.37 shows the temperature fields for each of those simulations at time t = 1.0. One can notice that for Pr = 1 the higher gradients of the temperature field are located around the interface. On the other hand, as the Prandtl number increases, the temperature field appears more and more disconnected to the interface location.

6.10.3 Simulation of nuclear-like cases

As a proof of concept that the VPM solver developed in this thesis can be useful for nuclear applications, a simulation with parameters similar to what could be encountered in a nuclear reactor is performed. Figure 6.38 shows a simulation of a nuclear vessel containing liquid sodium in its lower half and argon above it. Hence, there is a mass density ratio of $\rho_+/\rho_- = 1000$ across the interface and Prandtl numbers of 0.01 and 1 in the lower and upper phases. At the time being, those results are preliminary and under-resolved (no subgrid-scale model has been used). It is merely meant to show that the VPM method can handle such simulation.

A heated core in the lower-middle of the pool heats and puts the fluid in motion. The upward motion is not caused by buoyancy effects (in our current model, the mass density and dynamic viscosity are not function of the temperature but merely of the level set field). Instead, numerically it is implemented as a momentum and temperature source. In order to yield more complex (and interesting) flow structures, the fluid acceleration is much stronger than the one that would be generated by buoyancy effects only.

At the interface, the high temperature region reaches beyond the interface. This is caused by the thermal diffusion of course, but also mainly by the interface thickness required by the smeared interface model.

Argon bubbles are being entrained by the recirculation of the sodium coolant, and quickly brought to temperature because of their small volume and hence heat capacity, but also because of the interface thickness that reaches halfway into the bubble.

Below the contact line a pocket of colder fluid can be seen Fig. 9a-9b. Its heating is delayed because the presence of the wall prevents it from being sucked away by the vortex, and hence cannot mix with it. It is only later that



Figure 6.37: Temperature fields on the Rayleigh-Taylor A benchmark, depending on the Prandtl number.



Figure 6.38: Temperature field (\blacksquare) and interface location (\blacksquare) during the simulation of the main vessel of a sodium-cooled pool-type reactor with an impulsively started heating core. The flow is put in motion and heated by the core in the lower-middle.

the momentum and temperature diffuse, leading to both its progressive heating and its advection downwards. As it is being pushed down, the cold fluid is then replaced with warmer fluid from the region just below the interface.

Turbulent flow structures can be seen in the argon atmosphere where the turbulent mixing of the warmer argon from the interface and the colder argon away from the interface leads to a high temperature gradient regions. In comparison, the temperature gradients are much smaller in the low-Prandtl liquid metal.

On Figure 6.38b, the two higher temperature regions in the top corners of the fluid domains correspond to two vortex tubes of argon that have been heated close to the fluid interface then advected away by recirculation currents at the contact line.

The very symmetric behaviour of the flow in the beginning of the simulation Fig. 9a leaves place to more and more asymmetry with time Fig. 9b as the flow structures of each sides start interacting with each other.

6.11 Summary

The VPM method developped in this thesis has been validated on various benchmarks, firstly regarding the interface capturing method, then the implementation of the surface tension term and eventually on complex flows. The performance of the VPM solver has been compared to that of a velocity-pressure code, both in terms of accuracy of the results, and of computational efficiency. It appears that the VPM method yields results of similar quality than the VeloGrid solver. Additionally, veloGrid is cheaper by 33% for highly viscous flows while VPM can be up to four times cheaper on highly advective flows. Eventually, a heat equation solver is implemented. The influence of the Prandtl number on results is briefly discussed, and a proof-of-concept simulation of a nuclear reactor vessel is performed.

Chapter 7

Conclusions and perspectives

7.1 General conclusion

The VPM method has proven itself superior to traditional Eulerian velocitypressure methods for certain specific applications, including the study of wakes behind aircraft or wind turbines. The initial motivation of the present work was to probe the potential of the VPM method for multiphase applications, which is a domain that has almost never been explored by other researchers so far [106].

In this thesis, a multiphase Vortex Particle-Mesh method with heat transfer has been presented. Starting from a single-phase flow VPM method (see section 2), features were added successively: interface capturing (see section 3), handling of surface tension (see section 4.1), variable fluid properties (see section 4.2), handling of high mass density-ratio flows (see section 5), and the computation of thermal conduction and convection (see section 6.10). The full VPM solver was then validated on benchmarks and its performance compared to that of a more traditional velocity-pressure finite difference Eulerian solver (see section 6.4).

The work presented in this thesis allowed for the method to support cases that were not handled initially, and whose solution was not available in the current litterature. This includes the computation of the surface tension term for finite difference vortex methods, or the buoyancy instability for multiphase vortex methods with high mass-density ratios. Work on the latter has been submitted for publication in the Computers & Fluids journal. As of today the VPM multiphase solver seems to be on par with more traditional solvers both in terms of acuracy and computational efficiency. The benchmarks show an equivalent accuracy in the results obtained. On the other hand, thanks to its Lagrangian nature, the VPM solver is up to four times faster than traditional velocity-pressure solvers for highly advective flows, whereas the computational costs of the velocity-pressure solver is slightly inferior to the cost of the VPM method by around 33% for highly viscous flows. Eventually, VPM methods are inherently a fair bit more complex to implement and hence represent an advantage only where their affordable cost outperforms that of traditional solvers.

Nevertheless, it is important to note that multiphase Vortex Particle-Mesh methods are yet fairly uncommon. Indeed, recall that only one article on a similar subject has been found [106]. Hence improvements of the method presented in this thesis could lead to a very competitive solver, at least for some applications. In particular, the implementation of a subgrid-scale turbulence model will allow the use of coarser meshes and hence weaken the mesh Fourier stability constraints. By doing so, one shifts towards the "highly-advective" class of flows for which the VPM is superior to traditional methods. Also, working on the smallest domain to encompass non-zero vorticity values will once again play in favor of the VPM method. Indeed the vorticity field is more compact than the velcotiv field. Last but not least, although great care was taken in the development of the VPM solver, the latter is still young and can probably still be optimized, both in terms of computational efficiency and accuracy. On a side note, a staggered discretisation of the level set field in respect to vorticity, or an immersed-interface method would be interesting additions. Note however, that working in vorticity will necessarily add an additional derivative on the viscous and surface tension terms as compared to a velocity-pressure solver, and that represents an additional challenge for the simulation of two-phase flows with very different fluid properties and hence very large gradients accross the fluid interface.

A more detailled summary of the contributions of the present thesis and perspectives for future works are now presented.

7.2 Achievements and research results

7.2.1 Answers to research questions

In the introduction (see section 1.6), four research questions were formulated. The present section provides a quick overview of their answers, which will be further developped in the upcoming sections:

What advantages brings the Lagrangian implementation of a multiphase flow solver?

The Lagrangian implementation has shown to provide a better computational efficiency, up to four times faster, for highly-advective flows. On the other hand it was 33% slower in highly-viscous flows.

How does a multiphase VPM compare to a more traditional Eulerian method?

The VPM solver presented in this thesis achieved similar accuracy than a velocity-pressure Eulerian solver, but can be cheaper for highly advective flows (see previous point).

How to implement surface tension and handle phases with differing properties?

A single-fluid method was used to handle the difference in fluid properties between both phases, and a the surface tension term was added. The latter caused an odd-even decoupling which was solved by using a tangential discrete filter. A better solution would be to develop a staggered method.

How to simulate flows with large mass density difference between phases?

Large mass density difference can trigger a numerical instability in the baroclinic term of the vorticity equation, when the latter is solved in a parabolic manner. This can be prevented either by using an elliptic solver for pressure or by under-relaxing the material acceleration embedded in the baroclinic term.

7.2.2 Summary of the work accomplished

Firstly, the motion of the fluid interface has been captured using a particlebased implementation of the level set method. The choice of this particular method was justified by our strong requirement of a method that could capture the geometry of the interface well, the lesser importance given to volume conservation, and the lack of maturity of the conservative level set method. Note that the level set reinitialization is performed using the Hamilton-Jacobi formulation instead of the Fast Marching Method, and that the local variant of the level set method has been chosen.

Different strategies for level set reinitialization have been investigated: the level set field may be reinitialized fully every n time steps, or only when the level set field is detected as being too distorted, or it may be partially reinitialized at every single time step. As part of that effort, several ways to measure the distortion of the level set field have been investigated. Indeed, although the main measure of distortion is how different from one the gradient of the level set field is, this measure is only meaningful away from level set shocks. Hence, the regions of the domain with shocks must be filtered out.

Secondly, the level set field has been used to compute the surface tension term of the vorticity equation. As for most terms of the vorticity equation, the surface tension term is obtained by taking the curl of its corresponding term in the momentum equation. Its expression can nevertheless be simplified using the properties of the term, such as the fact that certain components of the term are invariant tangentially to the interface. In particular, two formulations of this term have been suggested and their influence on the results of simulations has been shown. Eventually, the partially expanded variant (Eq. 4.6) has been prefered over the fully expanded one (Eq. 4.7).

Also, the existence of an odd-even decoupling for the surface tension term has been found. Indeed, it is shown that if vorticity and level set fields are expressed at collocated nodes, then the vorticity field is insensitive to some high frequency level set variations (flip-flop mode). A filtering solution was proposed, that is based on discrete filters. It was shown that two dimensional discrete filters distort the solution significantly by artificially creating "trenches" in the bubble, and leading to very unphysical results. Instead three "tangential" discrete filters were proposed. Upon comparison of their performances, the "direct" form filter was chosen because it preserved the interface's shape best.

Following that, it was noted that the process of level set reinitialization generated some artificial fluctuations in the level set field. Those fluctuations are imperceptible in the level set field itself, but become considerable in the gradient of the level set laplacian. Since the latter is directly used in the surface tension term of the vorticity equation, it is strongly impacted as well. In practice it results in the apparition of "dents" in the vorticity field, which is to say local maxima and minima of vorticity along the interface. Similar fluctuations have been observed in the surface tension term of the momentum equation, but since the latter is a function of the laplacian of the level set field (instead of the gradient of the laplacian) those fluctuations are much weaker. In practice the presence of those dents does not seem to affect the solutions of the solvers, and the gradient of curvature is hence used unchanged in the rest of the thesis.

Additionally, the viscous term of the vorticity equation has been implemented using the single-fluid model where both phases are considered as a fluid with variable fluid properties. In particular, the dynamic viscosity and mass density are mollified accross the interface. This requires a more complex expression for the viscous term than a mere Laplacian of vorticity. That term can be expanded in multiple ways as shown in the appendix (see appendix G). It was shown that a previous work on the subject by Thirifay [115] contained an error which has been corrected, and whose impact on the results has been shown.

The existence of a numerical instability on the buoyancy term was discovered and investigated. A simplified one-dimensional model was devised which reproduces satisfactorily the behavior observed in VPM simulations. Properties of this unstability were investigated, and in particular the influence of various parameters was studied. It appeared that the main criteria influencing the instability growth rate is the ratio of mass densities between both phases. It was also shown that the choice of the time integrator had an influence on the convergence and growth rate of the instability. In particular two Runge-Kutta time integrators of the same order and hence same stability regimes according to a von Neumann stability analysis, but different Butcher table coefficients appear to have different stability regimes in respect to the buoyancy instability, with one being more stable than the other.

Moreover, an under-relaxation method was devised and implemented to tackle the stability issue. The values of the under-relaxation coefficient required for achieving stability for a given mass-density ratio are presented. Also, the one-dimensional model gives consistent results with VPM simulations in the under-relaxed case.

This work has been materialized in an article submitted for publication into the Computers & Fluids journal.
Following that, the VPM method was validated on benchmarks and its performance was compared to that of a traditional Eulerian velocity-pressure solver, both in terms of accuracy and computational efficiency. It results that similar accuracy is achieved by the VPM method and that it can be up to four times faster in terms of computational times for highly advective flows. The convergence of the VPM method was also studied on a "pure level set" benchmark, and a "complex flow benchmark". On a side note, the accuracy of several methods for measuring volume and surface area from a level set field was studied.

Eventually, results obtained with the developped VPM method enriched by a heat equation solver have been presented. This includes a study of the influence of the Prandtl number on the flow. Also, in order to show the viability of the method for real-world applications, a simulation of a pool filled with Sodium and covered with Argon similar to what could be encountered in a real nuclear application is performed.

7.3 Perspectives

To pursue the work, several subjects can be investigated:

- Chapter 5 describes a numerical instability on the buoyancy term. It was shown that the choice of the time integrator sensibly influences the stability of the numerical method (see section 5.6.2). It also showed in the same section that two Runge-Kutta time integrators of same order but different Butcher tables have different stability properties in respect to the numerical buoyancy instability, despite having identical stability behavior in the sense of the von Neumann stability analysis. Hence, it should be investigated, for a given order of the Runge-Kutta time integrator, what are the values of the Butcher table that achieve the greatest stability, and what is the maximum mass density ratio that could thereby be achieved.
- It has been shown in section 4.1.1 that the surface tension term in the vorticity equation suffers from an odd-even decoupling when the vorticity and level set fields are expressed on collocated nodes. A solution based on the filtering of the level set field has been presented in this thesis. Nevertheless, it has several shortcomings. In particular, it drastically reduces the accuracy of the method: a level set field filtered on a $N \times N$ grid so as to remove the highest frequency mode ("flip-flop mode") actually has an effective resolution of $\frac{N}{2} \times \frac{N}{2}$. Thus the resolution is divided by four. In other words, one requires four times more points to achieve similar accuracy, which represents a significant increase of the cost of the method. Firstly because the computational cost of each time step increases with the number of points to handle. Secondly because a smaller grid spacing leads to a stronger mesh Fourier constraint and hence smaller admissible time steps. A better solution would be to use a staggered Vortex Particle-Mesh method but this represents significant changes in the numerical method. In particular it requires two sets of particles (one for the level set field and one for the vorticity field) and it is not obvious

how the advection of those two sets of particles should be performed. Moreover, note that the staggered VPM method proposed by Uchiyama et al. in 2013 [121] is not suitable for level set fields.

- Vortex Particle-Mesh methods are known to be especially efficient for highly advective flows. Indeed, in traditionnal Eulerian methods, the CFL number becomes the most strigent stability criteria, much before the mesh Fourier. On the other hand, VPM methods are unaffected by the CFL and can push the time step value higher until the mesh Fourier's stability limit is reached. Such CFL-dominated simulations (where VPM methods become more computationally efficient) tend to be more frequent in LES simulations than in DNS simulations. It would therefore be interesting to add a subgrid-scale model to the solver, as has been done for some single-fluid VPM methods [94].
- Several multiphase flows are relevant to study in three-dimensions. This is true of some benchmarks (for instance only three-dimensional Rising Bubble benchmarks can be compared to experiments) and for certain applications (the flow within the vessel of a nuclear reactor is very dependent on some three-dimensional geometries, in particular the presence of the heat exchangers). At first, the method presented in this thesis should be extendable to three-dimensions without apparent difficulties. In particular, the implementation of the surface tension or viscous terms should not require additional care in 3D, and the level set method does not change significantly between 2D and 3D. Hence at first glance, the biggest change seems to be the addition of a vortex stretching term on the Right Hand Side of the vorticity equation. This term has been used in previous work such as Cottet's 1981 work [42]. Nevertheless, prudence is called for when it comes to numerical methods, as the harshest difficulties are sometimes not the most obvious ones...
- Nowadays most High Performance Computing CFD simulations use parallel computing. The method presented here has been developped for single-core solvers, and it would be interesting to parallelize the code. Nevertheless, to limit possible difficulties, the methods used in the VPM solver have been chosen to be easily parallelized. For instance, the Hamilton-Jacobi implementation of the level set method has been prefered over the Fast Marching Method, in part because the later is more difficult to parallelize eventhough solutions exist [168] [169] [170].
- Eventually, one of the original motivations behind the present work was the simulation of the conjugated heat transfer between wall, coolant and cover gas in a pool-type nuclear reactor, as was presented in section 1.2 and illustrated on figure 1.4. This particular problem requires the simulation of the exchange of heat between the wall and the fluids, which would require some changes to the heat equation solver. It would indeed be necessary to set boundary conditions which allow the fluid domain and a solid wall domain to communicate with each other. Also boundary conditions on the level set field will need to be implemented in order

to simulate the physics of the dynamic contact angle (see section 3.2.2). Additionally, it would be useful to handle the fluid's properties such as dynamic viscosity and mass density as functions of temperature, instead of keeping them constant as is done in the present work¹. Eventually, the VPM does not handle walls naturally, and thence, it might be preferable to use a hybrid method that uses a velocity-pressure solver at the wall, and a vorticity solver away from the wall, in a way similar to Thirifay's work [115] for instance.

Acknowledgements

This work has been funded by Nuclear Technical Safety Organisation Bel V^2 .

 $^{^1\}mathrm{Values}$ of some fluid properties as a function of temperature are indicated in Appendix (see appendix E).

Bibliography

- [1] JP Christiansen. Vortex: a 2-dimensional hydrodynamics simulation code. Technical Report CLM-R106, Culham Laboratory, 1970. http://www.osti.gov/scitech/biblio/4080682-vortex-dimensional-hydrodynamics-simulation-code.
- [2] FH Harlow. Fluid dynamics in group t-3 los alamos national laboratory (la-ur-03-3852). Journal of Computational Physics, 2004.
- [3] KV Roberts and JP Christiansen. Topics in computational fluid mechanics. *Computer physics communication*, 1972.
- [4] A Critchlow. Arctic drilling is inevitable: if we don't find oil in the ice, then russia will. http://www. telegraph.co.uk/finance/newsbysector/energy/11080635/ Arctic-drilling-is-inevitable-if-we-dont-find-oil-in-theice-then-Russia-will.html, Sep 2014. (The Telegraph article).
- [5] National Energy Board. Canada's oil sands opportunities and challenges to 2015: An update. Technical report, National Energy Board, Jun 2006.
- [6] Navigant Research. Shale gas export boom: Don't celebrate yet. https://www.forbes.com/sites/pikeresearch/2013/10/18/ shale-gas-export-boom-dont-celebrate-yet/#7a3508a82e2f, Oct 2013. (Forbes article).
- [7] D Demetriou. Energy imports push japan trade deficit to record high. http://www.telegraph.co.uk/finance/economics/10598545/ Energy-imports-push-Japan-trade-deficit-to-record-high. html, Jan 2014. (The Telegraph article).
- [8] J McCurry. Japan restarts first nuclear reactor since fukushima disaster. https://www.theguardian.com/environment/2015/aug/11/ japan-restarts-first-nuclear-reactor-fukushima-disaster, Aug 2015. (The Guardian article).
- [9] US Energy Information Administration. Total primary energy consumption 2014. http://www.eia.gov/cfapps/ipdbproject/IEDIndex3.cfm?tid=44&pid=44&aid=2, 2014. (Eia database).

- [10] Reuters Staff. Chinese capital shuts third coal-fired plant in war on smog. https://www.reuters.com/article/us-china-pollution-beijing/ chinese-capital-shuts-third-coal-fired-plant-in-war-on-smog -idUSKBNOMG1D120150320, Mar 2015. (Reuters article).
- [11] J Duggan. China working on uranium-free nuclear plants in attempt to combat smog. https://www.theguardian.com/world/2014/mar/ 19/china-uranium-nuclear-plants-smog-thorium, Mar 2014. (The Guardian article).
- [12] R Martin. Germany runs up against the limits of renewables. https://www.technologyreview.com/s/601514/ germany-runs-up-against-the-limits-of-renewables/, Mai 2016. (MIT Technology Review).
- [13] Reuters Staff. Renewable energy increases blackout risk-report. https://www.reuters.com/article/idUSL5E7MN37M20111123, Nov 2011. (Reuters article).
- [14] C Morris. German power prices negative over weekend. https://energytransition.org/2014/05/ german-power-prices-negative-over-weekend/, May 2014. (Energy Transition article).
- [15] 50Hertz. Electricity flows between 50hertz (germany) and ceps (czech republic) are regulated by phase-shifting transformers. http://www.50hertz.com/en/News/Detail/id/3244, Jan 2017. (50Hertz press release).
- [16] A Grubler. The costs of the french nuclear scale-up a case of negative learning by doing. *Energy Policy*, 2010.
- [17] F Carré, JM Cavedon, J Knebel, P Lisowski, T Ogawa, D Pooley, A Versteegh, T Dujardin, and C Nordborg. Independent evaluation of the myrrha project, report by an international team of experts. Technical report, OECD, 2009.
- [18] SE Hieber and P Koumoutsakos. A lagrangian particle level set method. Journal of Computional Physics, Jun 2005.
- [19] K Myrillas, P Planquart, A Simonini, JM Buchlin, and M Schyns. Cfd and experimental investigation of sloshing parameters for the safety assessment of hlm reactors. *Nuclear Engineering and Design*, Jul 2016.
- [20] Y Marichal. An immersed interface vortex particle-mesh method. PhD thesis, Université catholique de Louvain, Sep 2014.
- [21] Many authors. Methods in computational physics, volume Vol 9 plasma physics. Academic Press, 1964.
- [22] RW Hockney and JW Eastwood. Computer simulation using particles. Adam Hilger, 1988.

- [23] S Subramaniam. Lagrangian-eulerian methods for multiphase flows. Progress in Energy and Combustion Science, 2012.
- [24] J von Neumann. Proposal and analysis of a new numerical method for the treatment of hydrodynamical shock problems. Technical Report OSRD-3617, Office of scientific research and development, 1944.
- [25] JJ Monaghan. Smoothed particle hydrodynamics and its diverse applications. Annual Review of Fluid Mechanics, 2012.
- [26] JJ Monaghan. Smoothed particle hydrodynamics. Reports on progress in physics, 2005.
- [27] DJ Price. Smoothed particle hydrodynamics : things i wish my mother taught me. *Unpublished*, Nov 2011.
- [28] JJ Monaghan. A brief history of sph in hydraulics. *Hydrolink*, Oct 2015.
- [29] transient fluid-flow problems involving free surfaces The MAC method, a computing technique for solving viscous incompressible. Je welch and fh harlow and jp shannon and bj daly. Technical Report LA-3425, Los Alamos National Laboratory, Mar 1966.
- [30] S McKee, MF Tomé, VG Ferreira, JA Cuminato, A Castelo, FS Sousa, and N Mangiavacchi. The mac method. *Computers and Fluids*, 2007.
- [31] SO Unverdi and G Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics*, 1992.
- [32] FH Harlow. Hydrodynamic problems involving large fluid distortions. Journal of the Association for Computing Machinery, 1957.
- [33] JP Christiansen. Numerical simulation of hydrodynamics by the method of point vortices. *Journal of Computational Physics*, 1973.
- [34] P Chatelain, A Curioni, M Bergdorf, D Rossinelli, W Andreoni, and P Koumoutsakos. Billion vortex particle direct numerical simulations of aircraft wakes. *Computational Methods Applied to Mechanical Engineering*, 2008.
- [35] Many authors. Aerodynamics of vortical type flows in three dimensions. AGARD, 1983.
- [36] T Uchiyama and A Fukase. Three-dimensional vortex method for gasparticle two-phase compound round jet. Transactions of the American Society of Mechanical Engineers, Jan 2005.
- [37] G Papadakis and SG Voutsinas. In view of accelerating cfd simulations through coupling with vortex particle approximations. *Journal of Physics: Conference Series*, 2014.
- [38] GS Oxley. A 2-D hybrid Euler-compressible vortex particle method for transonic rotorcraft flows. PhD thesis, Carleton University, Jul 2009.

- [39] L Rosenhead. The formation of vortices from a surface of discontinuity. Proceedings of the royal society, 1931.
- [40] G Birkhoff and J Fisher. Do vortex sheets roll up? In Rendiconti del Circolo matematico di Palermo, 1959.
- [41] FR Hama. Streaklines in a perturbed shear flow. *Physics of Fluids*, 1962.
- [42] B Couët, O Buneman, and A Leonard. Simulation of three-dimensional incompressible flows with a vortex-in-cell method. *Journal of Computational Physics*, 1981.
- [43] L Greengard and V Rokhlin. A fast algorithm for particle simulations. Journal of Computational Physics, 1987.
- [44] J Barnes and P Hut. A hierarchical o(n log n) force-calculation algorithm. *Nature*, Dec 1986.
- [45] P Ploumhans, GS Winckelmans, JK Salmon, A Leonard, and MS Warren. Vortex methods for direct numerical simulation of three-dimensional bluff body flows: application to the sphere at re = 300, 500, and 1000. Journal of Computational Physics, 2002.
- [46] GS Winckelmans. Encyclopedia of Computational Mechanics, chapter Vortex methods. Willey, 2004.
- [47] CK Birdsall and D Fuss. Clouds-in-clouds, clouds-in-cells physics for many-body plasma simulation. *Journal of Computational Physics*, 1969.
- [48] JT Beale and A Majda. Rates of convergence for viscous splitting of the navier-stokes equations. *Mathematics of Computation*, 1981.
- [49] GH Cottet and S Gallic. Une méthode de décomposition pour une équation de type convection-diffusion combinant résolution explicite et méthode particulaire. Compte Rendu de l'Académie des Sciences Paris, 1983.
- [50] P Degond and S Mas-Gallic. The weighted particle method for convection-diffusion equations part 1:the case of an isotropic viscosity. *Mathematics of Computation*, Okt 1989.
- [51] JD Eldredge, A Leonard, and T Colonius. A general deterministic treatment of derivatives. *Journal Computational Physics*, 2002.
- [52] G Cottet and PD Koumoutsakos. *Vortex methods : theory and practice.* Cambridge University Press, 2000.
- [53] GS Winckelmans and A Leonard. Contributions to vortex particle methods for the computation of three-dimensional incompressible unsteady flows. *Journal of Computational Physics*, 1993.
- [54] LB Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, Dec 1977.

- [55] RA Gingold and JJ Monaghan. Smoothed particle hydrodynamics : theory and application to non-spherical stars. *Journal of Mechanics and Physics of Solids*, 1977.
- [56] MJ Gourlay. Fluid simulation for video games, 2016.
- [57] A Leonard. Vortex methods for flow simulation. Journal of computational physics, 1980.
- [58] C Greengard. The core spreading vortex method approximates the wrong equation. *Journal of Computational Physics*, 1985.
- [59] AJ Chorin. Numerical study of slightly viscous flow. Journal of fluid mechanics, 1973.
- [60] JE Fromm. A method for computing nonsteady, incompressible, viscous fluid flows. Technical Report LA-2910, Los Alamos National Laboratory, May 1963.
- [61] JE Fromm and FH Harlow. Numerical solution of the problem of vortex street development. *Physics of Fluids*, Jul 1963.
- [62] RE Brown. Rotor wake modeling for flight dynamic simulation of helicopters. AIAA Journal, Jan 2000.
- [63] RE Brown and AJ Line. Efficient high-resolution wake modeling using the vorticity transport equation. AIAA Journal, Jul 2005.
- [64] F Scheurich, TM Fletcher, and RE Brown. The influence of blade curvature and helical blade twist on the performance of a vertical-axis wind turbine. In *Proceedings of the 48th AIAA Aerospace Sciences Meeting*, 2010.
- [65] TM Fletcher and RE Brown. Simulation of wind turbine wake interaction using the vorticity transport model. *Wind Energy*, Dec 2009.
- [66] WH Reed and TR Hill. Triangular mesho methods for the neutron transport equation. Technical Report LA-UR-73-479, Los Alamos National Laboratory, 1973.
- [67] JG Liu and CW Shu. A high-order discontinuous galerkin method for 2d incompressible flows. *Journal of Computational Physics*, 2000.
- [68] L Gaul, M Kögl, and M Wagner. Boundary element methods for engineers and scientists. Springer, 2003.
- [69] JL Hess and AMO Smith. Calculation of nonlifting potential flow about arbitrary three-dimensional bodies. *Journal of ship research*, Sep 1964.
- [70] L Skerget, M Hriberšek, and G Kuhn. Computational fluid dynamics by boundary-domain integral method. International Journal for numerical methods in engineering, 1999.

- [71] Steven Schochet. The point-vortex method for periodic weak solutions of the 2-d euler equations. *Communications on Pure and Applied Mathematics*, Sep 1996.
- [72] OH Hald. Convergence of vortex methods for euler's equations. ii. SIAM Journal of Numerical Analysis, 1979.
- [73] Y Nakamura, A Leonard, and P Spalart. Vortex simulation of an inviscid shear layer. In Proceedings of AIAA/ASMA 3rd Joint thermophysics, Fluids, Plasma and Heat Transfer Conference, Jun 1982.
- [74] GH Cottet and S Mas-Gallic. A particle method to solve the navier-stokes system. *Numerische Mathematik*, 1990.
- [75] DI Pullin. Contour dynamics methods. Annual Review of Fluid Mechanics, 1992.
- [76] GS Deem and NJ Zabusky. Vortex waves: Stationary "v states", interactions, recurrence, and breaking. *Physical Review Letters*, 1978.
- [77] NJ Zabusky, MH Hughes, and KV Roberts. Contour dynamics for the euler equations in two dimensions. *Journal of Computational Physics*, 1979.
- [78] DG Dritschel. Contour dynamics and contour surgery: numerical algorithms for extended, high-resolution modelling of vortex dynamics in two-dimensional, inviscid, incompressible flows. *Computer Physics Reports*, 1989.
- [79] A Leonard. Numerical simulation of interacting three-dimensional vortex filaments. In Proceedings of 4th International Conference on Numerical Methods in Fluid Dynamics, Jun 1974.
- [80] AJ Chorin. Vortex models and boundary layer instability. SIAM Journal of Scientific Computing, 1980.
- [81] H Lamb. Hydrodynamics (4th edition). Cambridge University Press, 1916.
- [82] T von Karman. Über den mechanismus des widerstandes, den ein bewegter körper in einer flüssigkeit erfährt. Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse, 1911.
- [83] JCS Meng and JAL Thomson. Numerical studies of some nonlinear hydrodynamic problems by discrete vortex element methods. *Journal of Fluid Mechanics*, 1978.
- [84] GR Baker, DI Meiron, and SA Orszag. Vortex simulations of the rayleightaylor instability. *Physics of Fluids*, Aug 1980.
- [85] GR Baker, DI Meiron, and SA Orszag. Generalized vortex methods for free-surface flow problems. *Journal of Fluid Mechanics*, 1982.

- [86] G Tryggvason and H Aref. Numerical experiments on hele shaw flow with a sharp interface. *Journal of Fluid Mechanics*, 1983.
- [87] G Tryggvason. Numerical simulations of the rayleigh-taylor instability. Journal of Computational Physics, 1989.
- [88] C Greengard. Convergence of the vortex filament method. *Mathematics of Computation*, Oct 1986.
- [89] JT Beale and A Majda. Vortex methods i : Convergence in three dimensions. *Mathematics of Computation*, Jul 1982.
- [90] JT Beale. A convergent 3-d vortex method with grid-free stretching. Mathematics of Computation, 1986.
- [91] GH Cottet and P Poncet. Advances in direct numerical simulations of 3d wall-bounded flows by vortex-in-cell methods. *Journal of Computational Physics*, 2003.
- [92] GH Cottet, B Michaux, S Ossia, and G Vanderlinden. A comparison of spectral and vortex methods. *Journal of Computational Physics*, 2002.
- [93] WM van Rees, A Leonard, DI Pullin, and P Koumoutsakos. A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high reynolds numbers. *Journal of Computational Physics*, Dec 2010.
- [94] Roger Cocle, Grégoire Winckelmans, and Goéric Daeninck. Combining the vortex-in-cell and parallel fast multipole methods for efficient domain decomposition simulations. *Journal of Computational Physics*, Oct 2007.
- [95] A Kosior and H Kudela. Parallel computations on gpu in 3d using the vortex particle method. *Computers & Fluids*, 2012.
- [96] P Ploumhans and GS Winckelmans. Vortex methods for high-resolution simulations of viscous flow past bluff bodies of general geometry. *Journal* of Computational Physics, Nov 2000.
- [97] MS Shadloo, G Oger, and D Le Touzé. Smoothed particle hydrodynamics method for fluid flows, towards industrial applications: Motivations, current state, and challenges. *Computers and Fluids*, Mai 2016.
- [98] W Dehnen and H Aly. Improving convergence in smoothed particle hydrodynamics simulations without pairing instability. *Monthly Notices of* the Royal Astronomical Society, 2012.
- [99] Q Zhu, L Hernquist, and Y Li. Numerical convergence in smoothed particle hydrodynamics. *The Astrophysical Journal*, Feb 2015.
- [100] E Branlard, G Papadakis, M Gaunaa, G Winckelmans, and TJ Larsen. Aeroelastic large eddy simulations using vortex methods : unfrozen turbulent turbulent and sheared. *Proceedings of Wake Conference 2015*, May 2015.

- [101] P Chatelain, S Backaert, G Winckelmans, and S Kern. Large eddy simulation of wind turbine wakes. *Journal of computational physics*, 2011.
- [102] A Zervos, S Huberson, and A Hemon. Three-dimensional free wake calculation of wind turbine wakes. *Journal of wind engineering and industrial* aerodynamics, 1988.
- [103] M Gazzola, P Chatelain, WM van Rees, and P. Koumoutsakos. Simulations of single and multiple swimmers with non-divergence free deforming geometries. *Journal of computational Physics*, 2011.
- [104] A Mohammadian and J Marshall. A vortex in cell model for quasigeostrophic, shallow water dynamics on the sphere. Ocean Modelling, 2010.
- [105] A Selle, N Rasmussen, and R Fedkiw. A vortex particle method for smoke, water and explosions. ACM Transactions on Graphics, 2005.
- [106] S Shakouchi, S Shimada, and T Uchiyama. Numerical study of the mixing of density-stratified fluid with a jet. *Journal of Mechanics Engineering* and Automation, 2014.
- [107] K Luo, C Shao, Y Yang, and J Fan. A mass conserving level set method for detailed numerical simulation of liquid atomization. *Journal of Computational Physics*, Jun 2015.
- [108] N Balcazar, L Jofre, O Lehmkuhl, J Castro, and J Rigola. A finitevolume/level set method for simulating two-phase flows on unstructured grids. *International Journal of Multiphase flow*, May 2014.
- [109] A Colagrossi and M Landrini. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics*, 2003.
- [110] JA Zufiria. Vortex-in-cell simulation of bubble competition in a rayleightaylor instability. *Physics of Fluids*, Jul 1988.
- [111] TS Lundgren and P Koumaoutsakos. On the generation of vorticity at a free surface. Journal of Fluid Mechanics 382, 351-366, 1999.
- [112] JE Fromm. Numerical calculation of the fluid dynamics of drop-ondemand jets. *IBM Journal of Research and Development*, 1984.
- [113] CR Anderson. A vortex method for flows with slight density variations. Journal of Computational Physics, 1985.
- [114] F Thirifay and G Winckelmans. Development of a lagrangian method for combustion and application to the planar methane-air jet diffusion flame. *Journal of Turbulence*, Dec 2002.
- [115] F Thirifay. Simulation of non-reactive and reactive shear flows using Lagrangian particle methods. PhD thesis, Université catholiques de Louvain, 2006.

- [116] JH Walther and G Morgenthal. An immersed interface method for the vortex-in-cell algorithm. *Journal of Turbulence*, 2002.
- [117] G Morgenthal and JH Walther. An immersed interface method for the vortex-in-cell algorithm. *Computers and Structures*, Mar 2007.
- [118] Y Marichal, P Chatelain, and G Winckelmans. An immersed interface solver for the 2-D unbounded poisson equation and its application to potential flow. *Computers and Fluids*, under review.
- [119] Y Marichal, P Chatelain, and G Winckelmans. Immersed interface interpolation schemes for particle-mesh methods. *Journal of Computational Physics*, Sep 2016.
- [120] T Uchiyama and Y Yoshii. Numerical simulation of bubbly flow by vortex in cell method. *Procedia IUTAM*, 2015.
- [121] T Uchiyama, Y Yoshii, and H Hamada. Direct numerical simulation of a turbulent channel flow by an improved vortex in cell method. *Interna*tional Journal of Numerical Methods for Heat & Fluid Flow, Dec 2013.
- [122] JD Eldredge, T Colonius, and A Leonard. A vortex particle method for two-dimensional compressible flow. *Journal of Computational Physics*, 2002.
- [123] IF Sbalzarini, JH Walther, M Bergdorf, SE Hieber, EM Kotsalis, and P Koumoutsakos. Ppm - a highly efficient parallel particle-mesh library for the simulation of continuum systems. *Journal of Computational Physics*, Jan 2006.
- [124] F Hohl and RW Hockney. A computer model of disks of stars. Journal of Computational Physics, 1969.
- [125] H Abe, N Sakairi, and R Itatani. High-order spline interpolations in the particle simulation. *Journal of Computational Physics*, 1986.
- [126] JJ Monaghan. Extrapolating b-splines for interpolation. Journal of computational physics, 1985.
- [127] P Koumoutsakos, A Leonard, and F Pépin. Boundary conditions for viscous vortex methods. *Journal of Computational Physics*, 1994.
- [128] F Pépin. Simulation of the flow past an impulsively started cylinder using a discrete vortex method. PhD thesis, California Institute of Technology, May 1990.
- [129] FH Harlow. A machine calculation method for hydrodynamic problems. Technical Report LAMS-1956, Los Alamos National Laboratory, Nov 1955.
- [130] AM Tartakovsky, N Trask, K Pan, B Jones, W Pan, and JR Williams. Smoothed particle hydrodynamics and its applications for multiphase flow and reactive transport in porous media. *Computational Geosciences*, Aug 2016.

- [131] S Osher and JA Sethian. Fronts propagating with curvature dependent speed: Algorithhms based on hamilton-jacobi formulations. *Journal of Computational Physics 79* (12-49), 1988.
- [132] JA Sethian. Level Set Methods and Fast Marching Methods. Cambridge Monographs on applied and computational Mathematics, 1999.
- [133] SJ Osher and RP Fedkiw. Level Set Methods and dynamic implicit surfaces. Springer, 2000.
- [134] SJ Osher and RP Fedkiw. Level set methods: an overview and some recent results. Journal of Computational Physics 169 (463-502), Sep 2000.
- [135] JA Sethian and P Smereka. Level set methods for fluid interfaces. Annual Review of Fluid Mechanics, Jan 2003.
- [136] JA Sethian. An analysis of flame propagation. PhD thesis, University of California at Berkeley, 1982.
- [137] JA Sethian. Theory, algorithms, and applications of level set methods for propagating interfaces. Acta Numerica, 1996.
- [138] JA Sethian. Level set methods : An act of violence ; evolving interfaces in geometry, fluid mechanics, computer vision and materials sciences. *American scientist*, 1997.
- [139] J Li. Calcul d'interface affine par morceaux (piecewise linear interface calculation). Compte Rendu de l'Académie des Sciences Paris, 1995.
- [140] Y Renardy and M Renardy. Prost: A parabolic reconstruction of surface tension for the volume-of-fluid method. *Journal of Computational Physics*, 2002.
- [141] F Xiao, Y Honma, and T Kono. A simple algebraic interface capturing scheme using hyperbolic tangent function. *Physics of Fluids*, 2005.
- [142] F Xiao, S Ii, and C Chen. Revisit to the thinc scheme: A simple algebraic vof algorithm. *Physics of Fluids*, 2011.
- [143] L Jofre, O Lehmkuhl, J Castro, and A Oliva. A plic-vof implementation on parallel 3d unstructured meshes. In 5th European Conference on Computational Fluid Dynamics, Jun 2010.
- [144] M Sussman and EG Puckett. A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows. *Journal of Computational Physics*, 2000.
- [145] D Enright, R Fedkiw, J Ferziger, and I Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 2002.

- [146] K Yokoi. Efficient implementation of thinc scheme: A simple and practical smoothed vof algorithm. Journal of Computational Physics, 2007.
- [147] S Ii, K Sugiyama, S Takeuchi, S Takagi, Y Matsumoto, and F Xiao. An interface capturing method with a continuous function: The thinc method with multi-dimensional reconstruction. *Journal of Computational Physics*, Dec 2011.
- [148] E Olsson and G Kreiss. A conservative level set method for two phase flow. Journal of Computational Physics 210 (225-246), Jun 2005.
- [149] E Marchandise, P Geuzaine, N Chevaugeom, and JF Remacle. A stabilised finite element method using a discontinuous level set approach for the computation of bubble dynamics. *Journal of Computational Physics*, Jan 2007.
- [150] R Chiodi and O Desjardins. A reformulation of the conservative level set reinitialization equation for accurate and robust simulation of complex multiphase flows. *Journal of Computation Physics*, Aug 2017.
- [151] E Olsson, G Kreiss, and S Zahedi. A conservative level set method for two phase flow ii. Journal of Computational Physics 210 (225-246), Jun 2007.
- [152] T Wacławczyk. A consistent solution of the re-initialization equation in the conservative level-set method. *Journal of Computational Physics*, Jul 2015.
- [153] O Desjardins, V Moureau, and H Pitsch. An accurate conservative level set / ghost fluid method for simulating turbulent atomization. *Journal* of computational physics, Jun 2008.
- [154] JA Sethian D Adalsteinsson. The fast construction of extension velocities in level set methods. *Journal of computational physics*, 1999.
- [155] JO McCaslin, E Courtine, and O Desjardins. A fast marching approach to multidimensional extrapolation. *Journal of Computational Physics*, 2014.
- [156] S Zahedi, K Gustavsson, and G Kreiss. A conservative level set method for contact line dynamics. *Journal of computational physics*, Jun 2009.
- [157] W Ren, D Hu, and W E. Continuum models for the contact line problem. *Physics of Fluids*, Oct 2010.
- [158] E Kirkinis and SH Davis. Moffatt vortices induced by the motion of a contact line. *Journal of Fluid Mechanics*, 2014.
- [159] Mijail Febres and Dominique Legendre. Existence of moffatt vortices at a moving contact line between two fluids. *Physical Review Fluids*, Nov 2017.

- [160] D Legendre and M Maglio. Comparison between numerical models for the simulation of moving contact lines. *Computers & Fluids*, 2015.
- [161] S Afkhami, S Zaleski, and M Bussmann. A mesh-dependant model for applying dynamic contact angles to vof simulations. *Journal of Computational Physics*, 2009 2009.
- [162] OV Voinov. Hydrodynamics of wetting. Izvestiya Akademii Nauk SSSR, Mekhanika Zhidkosti i Gaza, Sep 1976.
- [163] R. G. Cox. The dynamics of the spreading of liquids on a solid surface. part 1. viscous flow. *Journal of Fluid Mechanics*, 1986.
- [164] CW Shu. High order weighted essentially non-oscillatory schemes for convection dominated problems. society for industrial and applied mathematics 51-1 (82-126), Nov 2009.
- [165] MW Jones, JA Bærentzen, and M Sramek. 3d distance fields: a survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics*, Jul 2006.
- [166] JA Sethian. A fast marching level set method for monotically advancing fronts. Proceeding of the National Academy of Science, 1996.
- [167] L Yatziv, A Bartesaghi, and G Sapiro. O(n) implementation of the fast marching algorithm. *Journal of Computational Physics*, Sep 2005.
- [168] M Herrmann. A domain decomposition parallelization of the fast marching method. Annual Briefs Center for Turbulence Research, 2003.
- [169] M Breuß, E Cristiani, P Gwosdek, and O Vogel. An adaptive domaindecomposition technique for parallelization of the fast marching method. *Applied Mathematics and Computations*, 2011.
- [170] M Detrixhe, F Gibou, and C Min. A parallel fast sweeping method for the eikonal equation. *Journal of Computational Physics*, Dec 2012.
- [171] D Peng, B Merriman, S Osher, H Zhao, and M Kan. A pde-based fast local level set method. *Journal of Computational Physics 155 (410-438)*, Jul 1999.
- [172] G Russo and P Smereka. A remark on computing distance functions. Journal of computational physics 163 (51-67), May 2000.
- [173] E Rouy and A Tourin. A viscosity solutions approach to shape-fromshading. SIAM Journal on Numerical Analysis, Jun 1992.
- [174] Paul Vigneaux. Méthodes Level Set pour des problèmes d'interface en microfluidique. PhD thesis, Université de Bordeaux, Jun 2008.
- [175] D Adalsteinsson and JA Sethian. A fast level set method for propagating interfaces. Journal of Computational Physics 188 (269-277), Oct 1994.

- [176] O Desjardins JO McCaslin. A localized re-initialization equation for the conservative level-set method. Journal of Computational Physics 262 (408-426), Jan 2014.
- [177] P Gómez, J Hernández, and J López. On the reinitialization procedure in a narrow-band locally refined level set method for interfacial flows. *International journal for numerical methods in engineering*, 2005.
- [178] M Herrmann. A balanced force refined level set grid method for twophase flows on unstructured flow solver grids. *Journal of Computational Physics*, Nov 2007.
- [179] JU Brackbill, D Kothe, and C Jemach. A continuum method for modeling surface tension. *Journal of Computational Physics*, July 1991.
- [180] AK Tornberg and B Engquist. A finite element based level-set method for multiphase flow applications. *Computing and Visualization in Science*, 2000.
- [181] K Yokoi, R Onishi, X Deng, and M Sussman. Density-scaled balanced continuum surface force model with a level set based curvature interpolation technique. *International Journal of Computational Methods*, 2016.
- [182] M Coquerelle and S Glockner. A fourth-order accurate curvature computation in a level set framework for two-phase flows subjected to surface tension forces. *Journal of Computational Physics*, 2015.
- [183] P Macklin and J Lowengrub. An improved geometry-aware curvature discretisation for level set methods : Application to tumor growth. *Journal* of Computational Physics, 2005.
- [184] KY Lervåg, B Müller, and ST Munkejord. Calculation of the interface curvature and normal vector with the level-set method. *Computers and Fluids*, 2013.
- [185] M Kang, RP Fedkiw, and XD Liu. A boundary condition capturing method for multiphase incompressible flow. *Journal of Scientific Computing*, 2000.
- [186] H Jeanmart and G Winckelmans. Investigation of eddy-viscosity models modified using discrete filters: a simplified "regularized variational multiscale model" and an "enhanced field model". *Physics of Fluids*, 2007.
- [187] JP Boris and DL Book. Flux-corrected transport i. shasta, a fluid transport algorithm that works. *Journal of Computational Physics*, 1973.
- [188] H Liu, Z Qi, and M Xu. Numerical simulation of fluid flow and interfacial behavior in threephase argon stirred ladles with one plug and dual plugs. *Steel Research International*, 2011.
- [189] PA Berthelsen and T Ytrehus. Stratified smooth two-phase flow using the immersed interface method. *Computers & Fluids*, Feb 2007.

- [190] T Ye, W Shyy, CF Tai, and JN Chung. Assessment of sharp- and continuous-interface methods for drop in static equilibrium. *Computers* and Fluids, 2004.
- [191] Z Li and MC Lai. The immersed interface method for the navier-stokes equations with singular forces. *Journal of Computational Physics*, 2001.
- [192] Z Li and K Ito. The immersed interface method. SIAM, 2006.
- [193] R Fedkiw, T Aslam, B Merriman, and S Osher. A non-oscillatory eulerian approachto interfaces in multimaterial flows (the ghost fluid method). *Journal of Computational Physics*, 1999.
- [194] W Bo and JW Grove. A volume of fluid method based ghost fluid method for compressible multi-fluid flows. *Computers & Fluids*, 2013.
- [195] AJ Williams III and MG Briscoe. Observations of shear and vertical stability from a neutrally buoyant float. *Journal of Geophysical Research*, Oct 1990.
- [196] IJ Parrish and E Quataert. Nonlinear simulations of the heat flux driven buoyancy instability and its implications for galaxy clusters. *The Astro*physical Journal Letters, 2008.
- [197] S Gottlieb and CW Shu. Total variation diminishing runge-kutta schemes. Mathematics of computation 221-67 (73-85), Jan 1998.
- [198] S Gottlieb, CW Shu, and E Tadmor. Strong stability-preserving highorder time discretization methods. *SIAM Review*, 2001.
- [199] L Ferracina and M Spijker. Stepsize restrictions for the total-variationdiminishing property in general runge–kutta methods. SIAM Journal of Numerical Analysis, 2004.
- [200] S Gottlieb. On high order strong stability preserving runge-kutta and multi step time discretizations. *Journal of Scientific Computing*, Nov 2005.
- [201] ST Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of computational physics*, 1979.
- [202] WJ Rider and DB Kothe. Stretching and tearing interface tracking methods. In Proceedings of the American Institute of Aeronautics and Astronautics 12th Computational Fluid Dynamics Conference, Jun 1995.
- [203] A Albadawi, DB Donoghue, AJ Robinson, DB Murray, and YMC Delauré. Influence of surface tension implementation in volume of fluid and coupled volume of fluid with level set methods for bubble growth and detachment. *International Journal of multiphase flow*, Feb 2013.
- [204] M Owkes and O Desjardins. A discontinuous galerkin conservative level set scheme for interface capturing in multiphase flows. *Journal of Computational Physics*, May 2013.

- [205] P Trontin, S Vincent, JL Estivalezes, and JP Caltagirone. A subgrid computation of the curvature by a particle/level-set method. application to a front-tracking/ghost-fluid method for incompressible flows. *Journal of Computational Physics*, Jul 2012.
- [206] M Sussman, P Smereka, and S Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of computational* physics 114 (146-159), Mar 1994.
- [207] A Prosperetti. Motion of two superposed viscous fluids. *Physics of fluids*, 1981.
- [208] S Hysing, S Turet, D Kuzmin, N Parolini, E Burman, S Ganesan, and L Tobiska. Quantitative benchmark computations of two-dimensional bubble dynamics. *International journal of numerical methods in fluids*, 2009.
- [209] Lord Rayleigh. Investigation of the character of the equilibrium of an incompressible heavy fluid of variable density. In *Proceedings of the London Mathematical Society*, Apr 1883.
- [210] Geoffrey Taylor. The instability of liquid surfaces when accelerated in a direction perpendicular to their planes i. *Proceedings of the royal society of london*, 1950.
- [211] JC Martin and WJ Moyce. Part iv an experimental study of the collapse of liquid columns on a rigid horizontal plane. *Philosophical Transactions* of the Royal Society of London. Series A, Mathematical and Physical Sciences, Mar 1952.
- [212] PK Stansby, A Chegini, and TCD Barnes. The initial stages of dam-break flow. Journal of Fluid Mechanics, 1998.
- [213] C Hu and M Sueyoshi. Numerical simulation and experiment on dam break problem. *Journal of Marine Science and Application*, 2010.
- [214] RJF Aleixo. Experimental study of the early stages of a dam-break flow over fixed and mobile beds. PhD thesis, Université catholique de Louvain, Jan 2013.
- [215] M Pfister and WH Hager. History and significance of the morton number in hydraulic engineering. *Journal of hydraulic engineering*, 2014.
- [216] DL Brown, R Cortez, and ML Minion. Accurate projection methods for the incompressible navier–stokes equations. *Journal of Computational Physics*, (168), 2000.
- [217] EG Puckett, AS Almgren, JB Bell, DL Marcus, and WJ Ridier. A highorder projection method for tracking fluid interfaces in variable density incompressible flows. *Journal of Computational Physics*, 1997.
- [218] A Prosperetti. Viscous effects on small-amplitude surface waves. *Physics of fluids*, 1976.

- [219] D Legrand. Failles meltdown/spectre : Intel prépare des mises à jour avec ses partenaires pour le 9 janvier. https://www.nextinpact.com/news/ 105909-failles-securite-et-kpti-intel-publie-courte-reponse -fin-embargo-9-janvier.htm, Jan 2018. (NextInpact article).
- [220] X Cheng and NI Tak. Investigation on turbulent heat transfer to leadbismuth eutectic flows in circular tubes for nuclear applications. *Nuclear Engineering and Design*, 2006.
- [221] L Bricteux, M Duponcheel, M Manconi, and Y Bartosiewicz. Numerical prediction of turbulent heat transfer at low prandtl number. *Journal of Physics*, 2012.
- [222] P Chatelain. Contributions to the three-dimensional vortex element method and spinning bluff body flows. PhD thesis, California Institute of Technology, Dec 2004.
- [223] R Courant, K Friedrichs, and H Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische annalen*, 1928.
- [224] J von Neumann and RD Richtmyer. A method for the numerical calculation of hydrodynamic shocks. *Journal of Applied Physics*, 1950.
- [225] AN Tikhonov and AA Samarskii. Homogeneous differences schemes on non-uniform nets. USSR Computational Mathematics and Mathematical Physics, 1962.
- [226] JW Cooley and JW Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, Apr 1965.
- [227] AJ Chorin. The numerical solution of the navier-stokes equations for an incompressible fluid. Bulletin of the American Mathematical Society, 1967.
- [228] HL Berk, CE Nielsen, and KV Roberts. Phase space hydrodynamics of equivalent nonlinear systems: experimental and computational observations. *Physics of Fluids*, Apr 1970.
- [229] JE Fromm and FH Harlow. Numerical solution of the problem of vortex street development. *Physics of Fluids*, Jul 1963.
- [230] RH Levy and RW Hockney. Computer experiments on low-density crossed-field electron beams. *Physics of Fluids*, Apr 1968.
- [231] FH Harlow and JE Welch. Numerical calculation of timedependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 1965.
- [232] CW Hirt, AA Amsden, and JL Cook. An arbitrary lagrangian-eulerian computing method for all flow speeds. *Journal of Computational Physics*, 1974.

- [233] JK Fink and L Leibowitz. Thermodynamic and transport properties of sodium liquid and vapor. Technical Report ANL-RE-95-2, Argonne National Laboratory, Jan 1995.
- [234] Many authors. Handbook on Lead-bismuth Eutectic Alloy and Lead properties, materials compatibility, thermal-hydraulics and technologies. Number NEA-6195 in Nuclear Energy Agency Report. Nuclear Energy Agency OECD, Jan 1995.
- [235] RB Stewart and RT Jacobsen. Thermodynamic properties of argon from triple point to 1200k with pressures to 1000mpa. *Journal of Physical and Chemical Reference Data*, 1989.
- [236] GC Maitland and EB Smith. Critical reassessment of viscosities of 11 common gases. Journal of Chemical and Engineering Data, 1972.
- [237] H Ziebland. Thermal conductivity of fluid substances. Pure & Applied Chemistry, 1977.
- [238] EW Lemmon and RT Jacobsen. Viscosity and thermal conductivity equations for nitrogen, oxygen, argon, and air. *International Journal of ther*mophysics, Jan 2004.

Appendix A Timeline of some methods

LAGRANGIAN METHODS LINEAGE	VORTEX LIN	K METH NEAGE	ODS CFD MILESTONES
	Analytical e.g. von Ka	point vo arman 19	ortices 11 [82]
Method of point	at vortices 931 [39]		CFL stability constraint Courant Friedrich Lewy 1928 [223]
Rosenhead 19 Early velocity-pressure lagrangian			Von Neumann stability analysis von Neumann 1950 [224]
method (shock propagation) von Neumann 1944 [24] Particle.in.Cell		L	Krylov subspace iteration anczos Hestenes Stiefel early 1950s
Harlow 1957 [32] (1955 [129])			
First method of po implemented on Birkhoff 1959 [40], H	oint vortices computer ama 1962 [41]	I	Finite volume method Tikhonov 1962 [225]
	Vorticity fi Fromm 19	nite diffe 63 [229]	rences ([60])
	Boundary-e (aka "par Hess	element r nel meth 1964 [69]	nethod od") I
Marker-in-Cell Harlow 1965 [231] (1966 [29])		L.	Fast Fourier Transform Cooley 1965 [226]
			Velocity projection Chorin 1967 [227]
Cloud- interpolatic Birdsall	in-Cells on (plasmas) 1969 [47]	Particle-i (pla Levy 1	mesh solver smas) 968 [230] ⁄
Arbitrary-Lagrangian-Eulerian	Vortex-ir Christiansen	n-Cell 1973 [33	Analogy between hydrodynamics, plasmas and astrophysics Berk 1970 [228]
11110 1374 [232]	(1970 [1])	[1])	Discontinuous Galerkin Reed 1973 [66]
	Vorticit Br	y Transp rown 200	oort Model 0 [62]

-

Appendix B

Weno5 schemes

This appendix specifies the Weighted Essentially Non-Oscillatory (WENO) schemes used for level set advection and reinitialisation. For more general information on WENO schemes, refer to Shu's 2009 review article [164].

Algorithm 5: Computes upwind/downwind space derivatives using WENO5 schemes

 $\begin{array}{l} 1 \ \ D_x^- q \leftarrow \texttt{weno5Diff}_{dwn}(q_{i-3,j} \ , \ q_{i-2,j} \ , \ q_{i-1,j} \ , \ q_{i,j} \ , \ q_{i+1,j} \ , \ q_{i+2,j}) \\ 2 \ \ D_x^+ q \leftarrow \texttt{weno5Diff}_{upw}(q_{i-2,j} \ , \ q_{i-1,j} \ , \ q_{i,j} \ , \ q_{i+1,j} \ , \ q_{i+2,j} \ , \ q_{i+3,j}) \\ 3 \ \ D_y^- q \leftarrow \texttt{weno5Diff}_{dwn}(q_{i,j-3} \ , \ q_{i,j-2} \ , \ q_{i,j-1} \ , \ q_{i,j} \ , \ q_{i,j+1} \ , \ q_{i,j+2}) \\ 4 \ \ D_y^+ q \leftarrow \texttt{weno5Diff}_{upw}(q_{i,j-2} \ , \ q_{i,j-1} \ , \ q_{i,j} \ , \ q_{i,j+1} \ , \ q_{i,j+2} \) \\ 5 \ return \ \left\{ D_x^- q \ , \ D_x^+ q \ , \ D_y^- q \ , \ D_y^+ q \right\}$

Algorithm 6: weno5Diff_dwn($q_{i-3}, q_{i-2}, q_{i-1}, q_i, q_{i+1}, q_{i+2}$) Downwind WENO5 differentiation

$$\begin{split} & 1 \ \mathrm{Compute \ derivative \ of \ cell-averaged \ level \ set} \\ & \left\{ D_x \bar{q}_{i+\frac{1}{2}+\delta i,j} \right\}_{\delta i \in [-3,+1]} \leftarrow \left\{ \frac{q_{i+1+\delta i,j}-q_{i+\delta i,j}}{h_x} \right\}_{\delta i \in [-3,+1]} \\ & \mathbf{2} \ \mathrm{Reconstruct \ level \ set \ derivative} \end{split}$$

 $\mathtt{D}_{\mathtt{x}}\mathtt{q} \gets \mathtt{weno5Reconstruct}_{\mathtt{dwn}} \left(\left\{ \mathtt{D}_{\mathtt{x}} \overline{\mathtt{q}}_{\mathtt{i}+\frac{1}{2}+\delta\mathtt{i},\mathtt{j}} \right\}_{\mathtt{\delta}\mathtt{i} \in [-3,+1]} \right)$

 $\boldsymbol{\mathfrak{s}}$ return $\boldsymbol{D}_{\boldsymbol{x}}\boldsymbol{q}$

Algorithm 7: weno5Diff_{upw} $(q_{i-2}, q_{i-1}, q_i, q_{i+1}, q_{i+2}, q_{i+3})$ Upwind WENO5 differentiation

- $$\begin{split} 1 \ \mathrm{Compute \ derivative \ of \ cell-averaged \ level \ set} \\ \left\{ D_x \bar{q}_{\mathtt{i}+\frac{\mathtt{i}}{2}+\delta\mathtt{i},\mathtt{j}} \right\}_{\delta\mathtt{i}\in[-2,+2]} \leftarrow \left\{ \tfrac{q_{\mathtt{i}+\mathtt{l}+\delta\mathtt{i},\mathtt{j}}-q_{\mathtt{i}+\delta\mathtt{i},\mathtt{j}}}{h_x} \right\}_{\delta\mathtt{i}\in[-2,+2]} \end{split}$$
- 2 Reconstruct level set derivative

$$\begin{split} & \mathtt{D}_{\mathtt{x}}\mathtt{q} \leftarrow \mathtt{weno5Reconstruct}_{\mathtt{upw}} \left(\left\{ \mathtt{D}_{\mathtt{x}} \bar{\mathtt{q}}_{\mathtt{i}+\frac{1}{2}+\delta\mathtt{i},\mathtt{j}} \right\}_{\delta\mathtt{i}\in[-2,+2]} \right) \\ & \mathtt{s \ return \ } \mathtt{D}_{\mathtt{x}}\mathtt{q} \end{split}$$

Algorithm 8: weno5Reconstruct_{dwn} $\left(\bar{q}_{i-\frac{5}{2}}, \bar{q}_{i-\frac{3}{2}}, \bar{q}_{i-\frac{1}{2}}, \bar{q}_{i+\frac{1}{2}}, \bar{q}_{i+\frac{3}{2}}\right)$ Downwind WENO5 reconstruction

- $\begin{array}{l} \text{1 Compute three different 3rd order approximations for } q(x_i) \\ q_{i,\text{A}} \leftarrow +\frac{1}{3} \, q_{i-\frac{5}{2}} \frac{7}{6} \, q_{i-\frac{3}{2}} + \frac{11}{6} \, q_{i-\frac{1}{2}} \\ q_{i,\text{B}} \leftarrow -\frac{1}{6} \, q_{i-\frac{3}{2}} + \frac{5}{6} \, q_{i-\frac{1}{2}} + \frac{1}{3} \, q_{i+\frac{1}{2}} \\ q_{i,\text{C}} \leftarrow +\frac{1}{3} \, q_{i-\frac{1}{2}} + \frac{5}{6} \, q_{i+\frac{1}{2}} \frac{1}{6} \, q_{i+\frac{3}{2}} \end{array}$
- 2 Compute smoothness indicators for each sub-stencil

$$\begin{split} \beta_{A} &\leftarrow +\frac{13}{12} \left(+q_{i-\frac{5}{2}} - 2\,q_{i-\frac{3}{2}} + q_{i-\frac{1}{2}} \right)^{2} + \frac{1}{4} \, \left(+q_{i-\frac{5}{2}} - 4\,q_{i-\frac{3}{2}} + 3\,q_{i-\frac{1}{2}} \right)^{2} \\ \beta_{B} &\leftarrow +\frac{13}{12} \, \left(+q_{i-\frac{3}{2}} - 2\,q_{i-\frac{1}{2}} + q_{i+\frac{1}{2}} \right)^{2} + \frac{1}{4} \, \left(+q_{i-\frac{3}{2}} - q_{i+\frac{1}{2}} \right)^{2} \\ \beta_{C} &\leftarrow +\frac{13}{12} \, \left(+q_{i-\frac{1}{2}} - 2\,q_{i+\frac{1}{2}} + q_{i+\frac{3}{2}} \right)^{2} + \frac{1}{4} \, \left(+q_{i+\frac{3}{2}} - 4\,q_{i+\frac{1}{2}} + 3\,q_{i-\frac{1}{2}} \right)^{2} \end{split}$$

3 Compute non-linear weights for each sub-stencil

$$\begin{split} \mathtt{w}_{\mathtt{A}} &\leftarrow \frac{1}{10} \frac{1}{(1 \texttt{e}^{-6} + \beta_{\mathtt{A}})^2} \\ \mathtt{w}_{\mathtt{B}} &\leftarrow \frac{6}{10} \frac{1}{(1 \texttt{e}^{-6} + \beta_{\mathtt{B}})^2} \\ \mathtt{w}_{\mathtt{C}} &\leftarrow \frac{3}{10} \frac{1}{(1 \texttt{e}^{-6} + \beta_{\mathtt{C}})^2} \end{split}$$

 $\begin{array}{l} 4 \ \, \text{Combine sub-stencils to approximate } q(x_i) \ \text{up to 5th order accuracy} \\ q_i \leftarrow \frac{1}{\texttt{w}_A + \texttt{w}_B + \texttt{w}_C} \ (\texttt{w}_A \ q_{i,A} + \texttt{w}_B \ q_{i,B} + \texttt{w}_C \ q_{i,C}) \end{array}$

 $\begin{array}{c} {\bf 5} \ {\rm Return} \ {\rm output} \\ {\bf return} \ {\bf q}_{i} \end{array}$

 $\begin{array}{l} \textbf{Algorithm 9: weno5Reconstruct}_{upw} \left(\bar{q}_{i-\frac{3}{2}}, \bar{q}_{i-\frac{1}{2}}, \bar{q}_{i+\frac{1}{2}}, \bar{q}_{i+\frac{3}{2}}, \bar{q}_{i+\frac{5}{2}} \right) \\ \textbf{Upwind WENO5 reconstruction} \\ \textbf{1} \quad \textbf{q}_{i} \leftarrow \texttt{weno5Reconstruct}_{dwn} \left(D_{x} \bar{q}_{i+\frac{5}{2}}, D_{x} \bar{q}_{i+\frac{3}{2}}, D_{x} \bar{q}_{i+\frac{1}{2}}, D_{x} \bar{q}_{i-\frac{1}{2}}, D_{x} \bar{q}_{i-\frac{3}{2}} \right) \end{array}$

 $_2$ return q_i

Appendix C Algorithms

C.1 Level set

C.1.1 Hamilton-Jacobi reinitialization

Algorithm 10: Level set reinitialization

for $\tau \leftarrow 0$ to $\delta \tau$ step h_{τ} do for each node (i, j) in grid Ω_h do Compute sign function 1 $\mathtt{S} \leftarrow \frac{\Phi_{\mathtt{i},\mathtt{j}}^{\mathtt{n}}}{\sqrt{\Phi_{\mathtt{i},\mathtt{j}}^{\mathtt{n}} + (\nabla \Phi^{\mathtt{n}}.\mathtt{h}_{\mathtt{x}})^2}}$ Compute upwind & downwind derivatives of level set $\mathbf{2}$ $\left\{ \mathtt{D}_{\mathtt{x}}^{-}\Phi, \mathtt{D}_{\mathtt{x}}^{+}\Phi, \mathtt{D}_{\mathtt{y}}^{-}\Phi, \mathtt{D}_{\mathtt{y}}^{+}\Phi \right\} \leftarrow \texttt{computeSpaceDerivatives}(\Phi^{\mathtt{n}})$ Compute norm of gradient 3 if $\Phi > 0$ then $|\mathtt{D}_{\mathtt{x}}\Phi| \leftarrow \max(\mathtt{0}, +\mathtt{D}_{\mathtt{x}}^{-}\Phi, -\mathtt{D}_{\mathtt{x}}^{+}\Phi)$ $|\mathtt{D}_{\mathtt{y}}\Phi| \leftarrow \max(\mathtt{0}, +\mathtt{D}_{\mathtt{y}}^{-}\Phi, -\mathtt{D}_{\mathtt{y}}^{+}\Phi)$ $|\nabla \Phi| \leftarrow \sqrt{|\mathtt{D}_\mathtt{x}\Phi|^2 + |\mathtt{D}_\mathtt{y}\Phi|^2}$ else if $\Phi < 0$ then $\begin{array}{l} |D_x\Phi| \leftarrow \max(0,-D_x^-\Phi,+D_x^+\Phi) \\ |D_y\Phi| \leftarrow \max(0,-D_y^-\Phi,+D_y^+\Phi) \end{array}$ $\left[\begin{array}{c} |\nabla \Phi| \leftarrow \sqrt{|\mathtt{D}_\mathtt{x} \Phi|^2 + |\mathtt{D}_\mathtt{y} \Phi|^2} \end{array} \right]$ else $| |\nabla \Phi| \leftarrow \mathbf{1}$ Compute $\partial_t \Phi$ 4 $\partial_{t}\Phi|_{i,j} \leftarrow -S(|\nabla\Phi|-1)$ Time integrate 5 $\Phi^{\mathtt{n+1}} \gets \Phi^{\mathtt{n}} + \mathtt{h}_{\mathtt{t}}.\partial_{\mathtt{t}} \Phi$



C.1.2 Fast Marching Method for reinitialization

Figure C.1: Fast Marching procedure. Shown here are cells containing frozen (\Box) , candidate (\Box) and far-away (\Box) nodes at their center; (---) is the interface. Cells with nodes that just changed status are marked with black text. Numbers correspond to the absolute value of level set with a grid spacing of h = 2.0. Steps (b) (d) and (e) use equation (Eq. 3.26), while step (c) uses equation (Eq. 3.29).

Nodes are sorted in three groups : the *frozen nodes* are the nodes that hold their definitive level set value, *candidate nodes* are the frozen nodes' neighbours and candidate at being the next frozen node, *far-away nodes* are all the other nodes that are obviously too far away from any frozen nodes to possibly be frozen at next time step.

Algorithm 11: Fast Marching algorithm : min-heap implementation		
#	Initialization	
1 f	or each node in grid do	
2	node.status \leftarrow if <i>node.isCloseToInterface()</i> then frozen else farAway	
3	$evelSet(node) \leftarrow if node.isFrozen() then abs(levelSet(node)) else$	
	$\lfloor +\infty$	
4 f	or each frozenNode to grid.frozenNodes() do	
5	for each neighbour to frozenNode.neighbours() do	
6	neighbour.status \leftarrow candidate	
7	heap.insert([computeCandidateLevelSet(neighbour), neighbour])	
#	Main loop	
8 V	vhile heap not empty do	
9	$levelSetToFreeze, nodeToFreeze \leftarrow heap.pop()$	
	# Heap may contain same node multiple times	
10	if nodeToFreeze.isFrozen() then continue	
11	nodeToFreeze.status \leftarrow frozen	
12	$evelSet(nodeToFreeze) \leftarrow evelSetToFreeze$	
13	for each neighbour to nodeToFreeze.neighbours() do	
14	$ $ neighbour.status \leftarrow candidate	
15	heap.insert([computeCandidateLevelSet(neighbour), neighbour])	

C.2 VeloGrid: Eulerian velocity-pressure solver

$$\frac{\partial \mathbf{u}^{*}}{\partial t} = -(\mathbf{u}^{*} \cdot \nabla) \mathbf{u}^{*} - \frac{\nabla P}{\rho_{\epsilon}} + \frac{1}{\rho_{\epsilon}} \nabla \cdot \left[\mu_{\epsilon} \left(\nabla \mathbf{u}^{*} + (\nabla \mathbf{u}^{*})^{T} \right) \right] \\
+ \left(1 - \frac{\rho_{ext}}{\rho_{\epsilon}} \right) \mathbf{g} - \delta_{\epsilon}(\Phi) \sigma \frac{\kappa}{\rho_{\epsilon}} \mathbf{n}$$
(C.1)

Algorithm 12: Main structure of VeloGrid solver

 ${\bf 1}$ Compute fluids' properties

 $\mathbf{2}$

$$\begin{split} \rho_{\varepsilon} &\leftarrow \frac{1}{\frac{1}{\rho_{\text{int}}} + H_{\varepsilon}(\Phi) \left(\frac{1}{\rho_{\text{ext}}} - \frac{1}{\rho_{\text{int}}}\right)} \\ \mu_{\varepsilon} &\leftarrow \mu_{\text{int}} + H_{\varepsilon}(\Phi) \left(\mu_{\text{ext}} - \mu_{\text{int}}\right) \\ \text{Compute interface's geometrical information} \\ \mathbf{n} &\leftarrow \frac{\nabla \Phi}{||\nabla \Phi||} \end{split}$$

$$\kappa \leftarrow \frac{\|\nabla \Phi\|}{\partial_{xx} \Phi \, \partial_y \Phi^2 - 2 \, \partial_x \Phi \, \partial_y \Phi \, \partial_{xy} \Phi + \partial_{yy} \Phi \, \partial_x \Phi^2}{\|\nabla \Phi\|^3}$$

3 Integrate the momentum equation $\mathbf{n}^{*n} \leftarrow \mathbf{n}^n$

$$\mathbf{u} \leftarrow \mathbf{u}$$

 $\mathbf{u}^{*n+1} \leftarrow \mathbf{u}^{*n} + \int_{h_t} \frac{\partial \mathbf{u}^*}{\partial t} dt$

4 Correct pressure and velocity fields Find δP such that : $\nabla \cdot \left(\frac{\nabla(\delta P)}{\rho}\right) = \frac{\nabla \cdot \mathbf{u}^{*n+1}}{h_t}$

$$\begin{array}{rcl} \mathbf{P}^{\mathbf{n}+1} &\leftarrow & \mathbf{P}^{\mathbf{n}}+\delta\mathbf{P} \\ \mathbf{u}^{\mathbf{n}+1} &\leftarrow & \mathbf{u}^{*\mathbf{n}+1}-\frac{\mathbf{h}_{\mathbf{t}}}{\rho}\,\boldsymbol{\nabla}(\delta\mathbf{P}) \end{array}$$

5 Update level set function

$$\Phi^{\mathtt{n+1}} \ \leftarrow \ \Phi^{\mathtt{n}} - \int_{\mathtt{h}_{\mathtt{t}}} (\mathtt{u} \cdot \boldsymbol{\nabla}) \Phi \, \mathtt{d} \mathtt{t}$$

 $\begin{aligned} \mathbf{6} \ \mbox{Reinitialize level set function on a regular basis} \\ & \mbox{If isTimeToReinitialize :} \\ & \Phi \gets \mbox{lvlSetReinit}(\Phi) \end{aligned}$

C.3 Vortex Particle-Mesh

Algorithm 13: Main structure of VPM solver

Note : the $\omega 2u()$ function transforms vorticity ω into velocity u following the procedure described in (Algo. 14-2)

1 Integrate vorticity and particles' position $\left(\omega^{n+1}\right) \quad \left(\omega^n\right) \quad \int \partial \omega_n \left(\left(\mathbf{x}\right)\right)$

$$\begin{cases} \boldsymbol{\omega}_p^{n+1} \\ \boldsymbol{X}_p^{n+1} \end{cases} \leftarrow \begin{cases} \boldsymbol{\omega}_p^n \\ \boldsymbol{X}_p^n \end{cases} + \int_{\mathbf{h}_{\mathbf{t}}} \frac{\partial \boldsymbol{\omega}_{\mathbf{p}}}{\partial \mathbf{t}} \left(\begin{cases} \boldsymbol{X}_p \\ \boldsymbol{\omega}_p \end{cases}, \mathbf{t}, \boldsymbol{\Phi}_{\mathbf{p}}^{\mathbf{n}}, \mathbf{u}_{, \mathtt{old}}^{\mathbf{n}} \right) \, \mathtt{dt}$$

2 Save velocity for next time step
$$\label{eq:unitary} \begin{split} \mathbf{u}^n \leftarrow \omega 2 u(\omega^n) \end{split}$$

$$\mathbf{u}_{,\texttt{old}}^{\texttt{n+1}} \leftarrow \mathbf{u}^{\texttt{n}}$$

- 4 Remesh particles regularly Every n_{rmsh} time steps:

$$\begin{array}{c} \mathbf{X}_{p}^{n+1}, \left\{ \begin{matrix} \boldsymbol{\omega}_{p}^{n+1} \\ \boldsymbol{\Phi}_{p}^{n+1} \end{matrix} \right\} \leftarrow \text{remesh} \left(\mathbf{X}_{p}^{n+1}, \left\{ \begin{matrix} \boldsymbol{\omega}_{p}^{n+1} \\ \boldsymbol{\Phi}_{p}^{n+1} \end{matrix} \right\} \right) \end{array}$$

5 Reinitialize level set field regularly If isTimeToReinitialize:

$$\begin{array}{c} \Phi^{n+1} \leftarrow \text{P2M}\left(\boldsymbol{X}_{p}, \Phi_{p}^{n+1}\right) \\ \Phi^{n+1} \leftarrow \text{lvlSetReinit}(\Phi^{n+1}) \\ \Phi_{p}^{n+1} \leftarrow \text{M2P}\left(\boldsymbol{X}_{p}, \Phi^{n+1}\right) \end{array}$$

Algorithm 14: Algorithm to evaluate the time derivative of vorticity $\frac{\partial \omega_p}{\partial t}$

 ${\bf 1}\,$ Interpolate vorticity and level set on grid

$$\begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{\Phi} \end{pmatrix} \leftarrow \mathtt{P2M} \left(\boldsymbol{X}_{\mathtt{p}}, \begin{pmatrix} \boldsymbol{\omega}_{p} \\ \boldsymbol{\Phi}_{p} \end{pmatrix} \right)$$

- 2 Transform vorticity into velocity Find ψ such that : $\nabla^2 \psi = -\omega$ $\mathbf{u} \leftarrow \nabla \times (\psi \, \hat{\boldsymbol{e}}_z)$
- $\begin{array}{l} \textbf{3} \ \text{Compute fluid's properties} \\ \rho_{,\varepsilon} \leftarrow (1 \mathtt{H}_{\varepsilon}(\Phi)) \, \rho_{\mathtt{int}} + \mathtt{H}_{\varepsilon}(\Phi) \, \rho_{\mathtt{ext}} \\ \mu_{,\varepsilon} \leftarrow (1 \mathtt{H}_{\varepsilon}(\Phi)) \, \mu_{\mathtt{int}} + \mathtt{H}_{\varepsilon}(\Phi) \, \mu_{\mathtt{ext}} \end{array}$
- 4 Evaluate time derivative of velocity

$$\frac{\mathtt{D} \mathbf{u}_{\mathtt{p}}}{\mathtt{D} \mathtt{t}} \leftarrow \frac{\mathbf{u}_{\mathtt{p}} - \mathbf{u}_{\mathtt{p}, \mathtt{old}}}{\mathtt{h}_{\mathtt{t}}}$$

 $5~{\rm Compute}~\partial_t\omega$ using momentum equation on vorticity

$$\begin{array}{lll} \partial_{\mathsf{t}}\boldsymbol{\omega} \ \leftarrow \ -\left(\boldsymbol{g}-\frac{\mathtt{D}\boldsymbol{u}}{\mathtt{D}\mathsf{t}}\right)\times \frac{\nabla\rho_{\varepsilon}}{\rho_{\varepsilon}} \ + \ \frac{1}{\rho_{\varepsilon}}\boldsymbol{\nabla}\times\left[\boldsymbol{\nabla}\cdot\left[\boldsymbol{\mu}_{\varepsilon}\left(\boldsymbol{\nabla}\boldsymbol{u}+\boldsymbol{\nabla}\boldsymbol{u}^{\mathrm{T}}\right)\right]\right] \\ & - \frac{\sigma}{\rho_{,\varepsilon}}\,\delta_{\varepsilon}\,\boldsymbol{\nabla}\times\left(\boldsymbol{\kappa}(\boldsymbol{\Phi})\,\boldsymbol{n}\right) \end{array}$$

 ${\bf 6}$ Interpolate $\partial_{{\sf t}}\omega$ and velocity from grid to particles

$$\begin{cases} \partial_t \omega_p \\ \mathbf{u}_p \end{cases} \leftarrow \mathsf{M2P}\left(\mathbf{X}_{\mathbf{p}}, \left\{\frac{\partial_t \omega}{\mathbf{u}}\right\}\right) \\ \mathbf{7} \text{ return } \begin{cases} \partial_t \omega_p \\ \mathbf{u}_p \end{cases}$$

Appendix D

Time-integrators

This appendix presents four time integrators. For each is specified the algorithm used in the calculations, its Butcher table as well as its sequence of error in material acceleration $(D_t v'^n)_{n \in \mathbb{N}}$.

As a reminder, the sequences $(D_t v'^n)_{n \in \mathbb{N}}$ are helpful to determine the stability properties of the time integrator relaxtive to the buoyancy instability studied in chapter 5 Those sequences are presented in the under-relaxed case where $D_t v'^{*,n-1}$ corresponds to the previously calculated under-relaxed value (see section 5.4.2). For the non-relaxed case, set the relaxation factor α to one. The recursion rules are expressed in terms of operator $\mathscr{I}(\cdot)$ which is defined as

$$\mathscr{I}(\mathbf{D}_{t}v') := -\mathcal{S}\left(\mathbf{D}_{t}v'\frac{\partial\ln(x)}{\partial x}\right)$$
(D.1)

D.1 Euler-Explicit

Algorithm 15: Euler explicit time inte-		
grator	0	
r^{n+1} (r) b f(r^{n} r^{n})		1
\mathbf{q}^{-} , $\mathbf{q}^{-} \leftarrow \mathbf{q}_{0} + \mathbf{n}_{t} \mathbf{I}(\mathbf{t}^{-}, \mathbf{q}^{-})$		

$$D_{t}v'^{n+1} = -\mathscr{I}\left(\alpha D_{t}v'^{n} + (1-\alpha) D_{t}v'^{*,n-1}\right)$$
(D.2)

D.2 RK2 Midpoint

Algorithm 16: RK2 midpoint time in- tegrator	0		
$\begin{array}{c} 1 \hspace{0.2cm} q^{(1)} \leftarrow q^n + \frac{h_t}{2} \hspace{0.2cm} \texttt{Rhs}(q^n, \texttt{t}^n) \\ 2 \hspace{0.2cm} q^{n+1} \leftarrow q^n + h_t \hspace{0.2cm} \texttt{Rhs}(q^{(1)}, \texttt{t}^n + \frac{h_t}{2}) \end{array}$	1/2	$\frac{1/2}{0}$	1

$$D_{t}v'^{n+1} = -\mathscr{I}\left(\frac{2}{3}\alpha D_{t}v'^{n} + (1-\alpha)D_{t}v^{*,n-1}\right) + \alpha \mathscr{I}^{2}\left(\frac{1}{3}\alpha D_{t}v'^{n} + \frac{1}{3}(1-\alpha)D_{t}v'^{*,n-1}\right)$$
(D.3)

D.3 Another RK2 time-integrator

Algorithm 17: Another RK2 time inte-		
grator	1	
$1 \ \mathbf{q}^{(1)} \leftarrow \mathbf{q}^{\mathbf{n}} + \mathbf{h}_{t} \operatorname{Rhs}(\mathbf{q}^{\mathbf{n}}, \mathbf{t}^{\mathbf{n}}) $	1	
$2 \mathbf{q}^{n+1} \leftarrow \mathbf{q}^n + \mathbf{h}_t \left(\frac{1}{2} \operatorname{Rhs}(\mathbf{q}^n, \mathbf{t}^n) + \frac{1}{2} \operatorname{Rhs} \left(\mathbf{q}^{(1)}, \mathbf{t}^n + \mathbf{h}_t \right) \right)$	1/2	1/2

$$D_{t}v'^{n+1} = -\mathscr{I}\left(\frac{3}{4}\alpha D_{t}v'^{n} + (1-\alpha)D_{t}v'^{*,n-1}\right) + \alpha \mathscr{I}^{2}\left(\frac{1}{4}\alpha D_{t}v'^{n} + \frac{1}{4}(1-\alpha)D_{t}v'^{*,n-1}\right)$$
(D.4)
D.4 A low storage RK3 time-integrator

Algorithm 18: A low-storage RK3 time-integrator	
$ \begin{array}{l} 1 \hspace{0.2cm} \text{Rhs}^{(1)} \leftarrow \text{Rhs}(\texttt{t}^n, \texttt{q}^n) \\ q^{(1)} \leftarrow \texttt{q}^n + \frac{1}{3} \hspace{0.2cm} \texttt{h}_{\texttt{t}} \hspace{0.2cm} \text{Rhs}^{(1)} \\ 2 \hspace{0.2cm} \text{Rhs}^{(2)} \leftarrow -\frac{5}{9} \hspace{0.2cm} \text{Rhs}^{(1)} + \hspace{0.2cm} \text{Rhs}(\texttt{t}^n + \frac{1}{4} \hspace{0.2cm} \texttt{h}_{\texttt{t}}, \texttt{q}^{(1)}) \\ q^{(2)} \leftarrow q^{(1)} + \frac{15}{16} \hspace{0.2cm} \texttt{h}_{\texttt{t}} \hspace{0.2cm} \text{Rhs}^{(2)} \\ 3 \hspace{0.2cm} \text{Rhs}^{(3)} \leftarrow -\frac{153}{128} \hspace{0.2cm} \text{Rhs}^{(2)} + \hspace{0.2cm} \text{Rhs}(\texttt{t}^n + \frac{3}{4} \hspace{0.2cm} \texttt{h}_{\texttt{t}}, \texttt{q}^{(2)}) \end{array} $	$\begin{array}{c ccccc} 0 & & & \\ 1/3 & 1/3 & & \\ 3/4 & -3/16 & 15/16 & & \\ \hline & 1/6 & 3/10 & 8/15 & \\ \end{array}$
$\textbf{q}^{\texttt{n+1}} \gets \textbf{q}^{(2)} + \tfrac{\texttt{8}}{\texttt{15}}\texttt{h}_\texttt{t}\texttt{Rhs}^{(3)}$	
$D_{t}v^{n+1} = -\mathscr{I}\left(\frac{39}{56}\alpha D_{t}v'^{n} + (1-\alpha)\right) + \alpha \mathscr{I}^{2}\left(\frac{13}{56}\alpha D_{t}v'^{n} + \frac{17}{56}(1-\alpha)^{2}\mathscr{I}^{3}\left(\frac{1}{14}\alpha D_{t}v'^{n} + \frac{1}{14}\right)\right)$	

Appendix E Fluid properties

Sodium values have been taken from [233]. Lead, Bismuth and LBE values have been taken from [234]. Mass density and specific heat capacity of Argon has been taken from [235], its dynamic viscosity from [236] and its thermal conductivity from [237]. Accuracy of Argon dynamic viscosity and thermal conductivity has been taken from [238]. Sodium specific heat capacity has been interpolated from values of [233], while mass density and specific heat capacity of argon has been interpolated from values of [235]. Surface tensions are only meaningful for liquids, not gases. From those sources, laws have either been extracted (if existing in reference article) or created as an interpolation from values given in those articles.

(Table E.1) references laws for mass density ρ , specific heat capacity c_p , thermal conductivity λ , dynamic viscosity μ and surface tension σ . On the other hand, thermal diffusivity α is computed as

$$\alpha = \frac{\lambda}{\rho \, c_p},\tag{E.1}$$

viscous diffisuvity ν (i.e. kinematic diffusivity) is computed as

$$\nu = \frac{\mu}{\rho},\tag{E.2}$$

and Prandtl number as

$$Pr = \frac{\nu}{\alpha}.$$
 (E.3)

Values for those quantities are available in tables E.2 E.3 and E.4. The heat capacities shown in those tables are taken at constant pressure.

$ \begin{array}{lll} {\rm Surface} & {\rm Sodium} \\ {\rm tension} & {\rm Lead} \\ [N.m^{-1}] & {\rm Bismuth} \\ {\rm LBE} \end{array} $	Sodium Dynamic Lead viscosity Bismuth [Pa.s] LBE Argon	$ \begin{array}{lll} & {\rm Sodium} \\ {\rm Thermal} & {\rm Lead} \\ {\rm conductivity} & {\rm Bismuth} \\ [W.m^{-1}.K^{-1}] & {\rm LBE} \\ {\rm Argon} \end{array} $	$\begin{array}{ccc} {\rm Specific} & {\rm Sodium} \\ {\rm heat} & {\rm Lead} \\ {\rm capacity} & {\rm Bismuth} \\ [J.kg^{-1}.K^{-1}] & {\rm LBE} \\ \end{array}$	$\begin{tabular}{ c c c c } \hline & Fluid \\ \hline & Sodium \\ Mass & Lead \\ density & Bismuth \\ [kg.m^{-3}] & LBE \\ & Argon \end{tabular}$
$\begin{array}{l} 371 < T < 1600 \\ 600.6 < T < 1300 \\ 544.4 < T < 1300 \\ 397.7 < T < 1400 \end{array}$	$\begin{array}{l} 371 < T < 1300 \\ 600.6 < T < 1470 \\ 544.4 < T < 1300 \\ 397.7 < T < 1100 \\ 87.178 < T < 2000 \end{array}$	$\begin{array}{l} 371 < T < 1500 \\ 600.6 < T < 1300 \\ 544.4 < T < 1000 \\ 397.7 < T < 1100 \\ 87.178 < T < 2000 \end{array}$	$\begin{array}{l} 371 < T < 1600 \\ 600.6 < T < 1300 \\ 544.4 < T < 1300 \\ 397.7 < T < 1200 \\ 87.178 < T < 1200 \end{array}$	$\begin{array}{l} \mbox{Validity range [K]} \\ \mbox{371} < T < 1400 \\ \mbox{600.6} < T < 2016 \\ \mbox{544.4} < T < 1300 \\ \mbox{397.7} < T < 1300 \\ \mbox{397.7} < T < 1300 \\ \mbox{87.178} < T < 1200 \end{array}$
11% 5% 4% 3%	5% 10% 6% 1%	$15\% \\ 3\% \\ 10\% \\ 2\%$	1% 2% 2% 10% 4%	Accuracy 0.4% 0.70% 0.40% 0.80% 1%
$\begin{array}{c} 240.5 \left(1-\frac{T}{T_c}\right)^{1.126} \\ 0.519 - 1.13 e^{-4}T \\ 0.4255 - 8.0 e^{-5}T \\ 0.437 - 6.6 e^{-5}T \end{array}$	$\begin{aligned} \exp(-6.4406 - 0.3958\ln(T) + 556.835\frac{1}{T}) \\ 4.55e - 4\exp\left(\frac{1069}{175.8}\right) \\ 4.458e^{-4}\exp\left(\frac{775.8}{175.4}\right) \\ 4.94e - 4\exp\left(\frac{775.1}{T}\right) \\ 2.228e^{-5}\exp\left(0.59077\ln(T) - \frac{92.577}{T} + \frac{2990.4}{T^2} - 3.0755\right) \end{aligned}$	$\begin{array}{c} 124.67-0.11381T+5.5226e^{-5}T^2-1.1842e^{-8}T^3\\ 9.2+0.011T\\ 6.55+1e-2T\\ 3.61+1.517e^{-2}T-1.741e^{-6}T^2\\ -3.596e^{-15}T^4+2.01e^{-11}T^3-4.463e^{-8}T^2+7.203e^{-5}T-3.162e^{-4}\end{array}$	$\begin{array}{c} 4.5049e^{-4}T^2 - 0.848284T + 1646.58 \\ +175.1 - 4.961e^{-2}T + 1.985e^{-5}T^2 - 2.099e^{-9}T^3 - 1.524e^6T^{-2} \\ 118.2 + 5.934e^{-3}T^{-3} + 71.83e^5T^{-2} \\ 159 - 2.72e^{-2}T + 7.12e^{-6}T^2 \\ 520.4 + 2.086e7T^{-2.940} \end{array}$	$ \begin{array}{c} \begin{array}{c} \text{Equation} \\ 219+275.32\left(1-\frac{T}{2503.7}\right)+511.58\left(1-\frac{T}{2503.7}\right)^{0.5} \\ ; 11367-1.1944T \\ 10726-1.2208T \\ 11096-1.3236T \\ 507.1T^{-1.008} \end{array} \right) $
[233] [234] [234]	[233] [234] [234] [236]	[233] [234] [234] [234] Interpolated from [237]	Interpolated from [233] [234] [234] [234] Interpolated from [235]	Source [233] [234] [234] [234] Interpolated from [235]

Table E.1: Fluid property equations at a pressure of 1hPa.

ಹ	s	
es at i	rmula	
value	the for	
ivity	s of t	
nduct	ange	
al co	lity r	
nerma	valic	
nd tl	e the	
city a	utsid	
capa(are o	
heat	cells	
sity,	npty	
den	년 년 년	
Mass	1hPa	
E.2:	re of	E.1)
able]	tnssə.	[able
Ë	\mathbf{pr}	L)

$[K^{-1}]$	Argon	0.022555	0.02474	0.02683	$0.028\ 81$	0.03071	0.032533	$0.034\ 27$	0.03594	0.03756	$0.039\ 13$	0.040.65	$0.042\ 14$	0.04359	$0.045\ 01$	$0.046\ 40$	0.047~78	$0.049\ 13$	$0.050\ 47$	0.05179	$0.053\ 10$	$0.054\ 39$	2%
$[W.m^{-}]$	LBE	9.40	10.08	10.76	11.43	12.09	12.73	13.38	14.01	14.63	15.25	15.85	16.45	17.04	17.62	18.19	,	,	ï	ı	ı	ı	10%
luctivity	Bismuth			ı	12.05	12.55	13.05	13.55	14.05	14.55	15.05	15.55	16.05	16.55		,	,	,		,	·	'	10%
nal cond	Lead		ı	ı	ı	ı	16.35	16.90	17.45	18.00	18.55	19.10	19.65	20.20	20.75	21.30	21.85	22.40	22.95	23.50	ı	ı	3%
Then	Sodium	87.22	83.56	80.09	76.81	73.71	70.77	68.00	65.38	62.90	60.56	58.34	56.24	54.24	52.35	50.54	48.81	47.16	45.57	44.03	42.54	41.08	15%
[]	Argon	520.87	520.73	520.64	520.58	520.54	520.51	520.49	520.47	520.46	520.45	520.44	520.44	520.43	520.43	520.42	520.42	520.42	ı	·	ı	ı	4%
$kg^{-1}.K$	LBE	149.26	148.20	147.18	146.19	145.24	144.33	143.45	142.61	141.80	141.02	140.29	139.59	138.92	138.29	137.70	137.14	136.61	,	,	,	ı	10%
apacity [J	Bismuth			,	141.95	138.15	135.20	132.86	130.97	129.42	128.14	127.07	126.16	125.38	124.72	124.14	123.63	123.19	122.80	122.45	,	'	2%
ic heat ca	Lead		,	,	,	,	147.06	146.27	145.46	144.66	143.87	143.12	142.40	141.72	141.08	140.49	139.96	139.47	139.03	138.64	,	ı	2%
Specif	Sodium	$1 \ 379.34$	$1 \ 356.08$	$1 \ 335.06$	$1 \ 316.30$	$1 \ 299.79$	$1 \ 285.53$	$1\ 273.52$	$1 \ 263.77$	$1 \ 256.27$	$1 \ 251.02$	$1 \ 248.02$	$1 \ 247.28$	$1 \ 248.79$	$1 \ 252.55$	$1 \ 258.56$	$1\ 266.83$	$1\ 277.34$	$1 \ 290.12$	$1 \ 305.14$	$1 \ 322.41$	$1 \ 341.94$	1%
	Argon	1.2084	1.073 1	0.965 0	0.876~6	0.803 0	0.740~8	0.687 4	$0.641 \ 3$	0.6009	$0.565\ 2$	0.5336	0.5053	0.479~8	0.456~8	0.4359	0.416~8	0.3993	$0.383\ 2$	$0.368 \ 3$		'	1%
m^{-3}]	LBE	10567	10500	$10 \ 434$	10 368	$10 \ 302$	$10 \ 236$	$10 \ 169$	$10 \ 103$	$10 \ 037$	9 971	9 905	9839	9772	902.6	9 640	9574	9508	9 442	9 375	,	'	0.80%
ensity [kg	Bismuth		ı	ı	$10\ 055$	9 994	$9 \ 932$	9 871	9 810	9749	9 688	9 627	9566	9505	9 444	9 383	9 322	9 261	9 200	9 139	ı	·	0.40%
Mass d	Lead			,	,		10591	10531	10 471	$10 \ 411$	$10\ 352$	$10\ 292$	$10\ 232$	$10\ 173$	$10\ 113$	$10\ 053$	9 993	$9 \ 934$	9874	9814	9755	9 695	0.70%
	Sodium	919	908	897	886	874	863	852	840	828	817	805	793	781	769	756	744	732	719	706	693	680	0.4%
Temperature $[K]$		400	450	500	550	600	650	200	750	800	850	900	950	1 000	$1 \ 050$	1 100	$1 \ 150$	$1 \ 200$	$1 \ 250$	$1 \ 300$	$1 \ 350$	1 400	Accuracy

Accuracy	1 400	$1 \ 350$	$1 \ 300$	1 250	$1\ 200$	$1 \ 150$	$1 \ 100$	$1 \ 050$	$1\ 000$	950	900	850	800	750	700	650	600	550	500	450	400		Temperature $[K]$	
5%	I	ı	$1.434 \cdot 10^{-04}$	$1.481 \cdot 10^{-04}$	$1.533 \cdot 10^{-04}$	$1.591 \cdot 10^{-04}$	$1.656 \cdot 10^{-04}$	$1.727 \cdot 10^{-04}$	$1.808 \cdot 10^{-04}$	$1.900 \cdot 10^{-04}$	$2.006 \cdot 10^{-04}$	$2.128 \cdot 10^{-04}$	$2.271 \cdot 10^{-04}$	$2.440 \cdot 10^{-04}$	$2.644 \cdot 10^{-04}$	$2.895 \cdot 10^{-04}$	$3.209 \cdot 10^{-04}$	$3.614 \cdot 10^{-04}$	$4.152 \cdot 10^{-04}$	$4.899 \cdot 10^{-04}$	$5.992 \cdot 10^{-04}$	Sodium		
10%	$9.764 \cdot 10^{-04}$	$1.004 \cdot 10^{-03}$	$1.035 \cdot 10^{-03}$	$1.070 \cdot 10^{-03}$	$1.109 \cdot 10^{-03}$	$1.153 \cdot 10^{-03}$	$1.202 \cdot 10^{-03}$	$1.259 \cdot 10^{-03}$	$1.325 \cdot 10^{-03}$	$1.402 \cdot 10^{-03}$	$1.492 \cdot 10^{-03}$	$1.600 \cdot 10^{-03}$	$1.731 \cdot 10^{-03}$	$1.892 \cdot 10^{-03}$	$2.095 \cdot 10^{-03}$	$2.356 \cdot 10^{-03}$	I	I	I	ı	T	Lead	Dyna	
10%	ı		$8.097 \cdot 10^{-04}$	$8.292 \cdot 10^{-04}$	$8.510 \cdot 10^{-04}$	$8.752 \cdot 10^{-04}$	$9.025 \cdot 10^{-04}$	$9.333 \cdot 10^{-04}$	$9.684 \cdot 10^{-04}$	$1.009 \cdot 10^{-03}$	$1.056 \cdot 10^{-03}$	$1.111 \cdot 10^{-03}$	$1.176 \cdot 10^{-03}$	$1.254 \cdot 10^{-03}$	$1.350 \cdot 10^{-03}$	$1.471 \cdot 10^{-03}$	$1.624 \cdot 10^{-03}$	$1.827 \cdot 10^{-03}$	ı			Bismuth	amic viscosity [
6%	ı			ı	ı	ı	$9.805 \cdot 10^{-04}$	$1.013 \cdot 10^{-03}$	$1.050 \cdot 10^{-03}$	$1.093 \cdot 10^{-03}$	$1.142 \cdot 10^{-03}$	$1.200 \cdot 10^{-03}$	$1.268 \cdot 10^{-03}$	$1.350 \cdot 10^{-03}$	$1.451 \cdot 10^{-03}$	$1.576 \cdot 10^{-03}$	$1.736 \cdot 10^{-03}$	$1.946 \cdot 10^{-03}$	$2.232 \cdot 10^{-03}$	$2.639 \cdot 10^{-03}$	$3.254 \cdot 10^{-03}$	LBE	Pa.s]	
1%	$6.963 \cdot 10^{-05}$	$6.800 \cdot 10^{-05}$	$6.633 \cdot 10^{-05}$	$6.464 \cdot 10^{-05}$	$6.291 \cdot 10^{-05}$	$6.115 \cdot 10^{-05}$	$5.936 \cdot 10^{-05}$	$5.754 \cdot 10^{-05}$	$5.567 \cdot 10^{-05}$	$5.377 \cdot 10^{-05}$	$5.181 \cdot 10^{-05}$	$4.981 \cdot 10^{-05}$	$4.776 \cdot 10^{-05}$	$4.565 \cdot 10^{-05}$	$4.348 \cdot 10^{-05}$	$4.123 \cdot 10^{-05}$	$3.891 \cdot 10^{-05}$	$3.651 \cdot 10^{-05}$	$3.400 \cdot 10^{-05}$	$3.139 \cdot 10^{-05}$	$2.865 \cdot 10^{-05}$	Argon		
11%	$9.562 \cdot 10^{+01}$	$1.005 \cdot 10^{+02}$	$1.054 \cdot 10^{+02}$	$1.104 \cdot 10^{+02}$	$1.153 \cdot 10^{+02}$	$1.203 \cdot 10^{+02}$	$1.254 \cdot 10^{+02}$	$1.304 \cdot 10^{+02}$	$1.355 \cdot 10^{+02}$	$1.405 \cdot 10^{+02}$	$1.456 \cdot 10^{+02}$	$1.508 \cdot 10^{+02}$	$1.559 \cdot 10^{+02}$	$1.611 \cdot 10^{+02}$	$1.662 \cdot 10^{+02}$	$1.714 \cdot 10^{+02}$	$1.767 \cdot 10^{+02}$	$1.819 \cdot 10^{+02}$	$1.871 \cdot 10^{+02}$	$1.924 \cdot 10^{+02}$	$1.977 \cdot 10^{+02}$	Sodium		
5%	ı		$3.721 \cdot 10^{-01}$	$3.778 \cdot 10^{-01}$	$3.834 \cdot 10^{-01}$	$3.891 \cdot 10^{-01}$	$3.947 \cdot 10^{-01}$	$4.004 \cdot 10^{-01}$	$4.060 \cdot 10^{-01}$	$4.117 \cdot 10^{-01}$	$4.173 \cdot 10^{-01}$	$4.230 \cdot 10^{-01}$	$4.286 \cdot 10^{-01}$	$4.343 \cdot 10^{-01}$	$4.399 \cdot 10^{-01}$	$4.456 \cdot 10^{-01}$	ı	I	ı	ı	-	Lead	Surface ter	
4%	I		$4.255 \cdot 10^{-01}$	ı		T	Bismuth	usion $[N/m]$																
3%	$3.446 \cdot 10^{-01}$	$3.479 \cdot 10^{-01}$	$3.512 \cdot 10^{-01}$	$3.545 \cdot 10^{-01}$	$3.578 \cdot 10^{-01}$	$3.611 \cdot 10^{-01}$	$3.644 \cdot 10^{-01}$	$3.677 \cdot 10^{-01}$	$3.710 \cdot 10^{-01}$	$3.743 \cdot 10^{-01}$	$3.776 \cdot 10^{-01}$	$3.809 \cdot 10^{-01}$	$3.842 \cdot 10^{-01}$	$3.875 \cdot 10^{-01}$	$3.908 \cdot 10^{-01}$	$3.941 \cdot 10^{-01}$	$3.974 \cdot 10^{-01}$	$4.007 \cdot 10^{-01}$	$4.040 \cdot 10^{-01}$	$4.073 \cdot 10^{-01}$	$4.106 \cdot 10^{-01}$	LBE		

Table E.3: Dynamic viscosity and surface tension values at a pressure of 1hPa. Empty cells are outside the validity ranges of the formulas (Table E.1)

206

Table E.4: Fluid property values at a pressure of 1hPa. Empty cells are outside the validity ranges of the formulas (Table E.1)

	Argon	0.6670	0.6669	0.6668	0.6667	0.6667	0.6667	0.6667	,			,	,	,	,	,	,		,	'	,	,
ıber	1 LBE	0.0517	0.0388	0.0305	0.0249	0.0209	0.0179	0.0156	0.0137	0.0123	0.0111	0.0101	0.0093	0.0086	0.0080	0.0074	ı	ı	ı	'	·	
ndtl num	Bismuth	1	ı	ı	0.0215	0.0179	0.0152	0.0132	0.0117	0.0105	0.0095	0.0086	0.0079	0.0073	ī	ı	ı	ī	ı	,	ı	,
Prar	Lead	1	ı	ı	ı	ı	0.0212	0.0181	0.0158	0.0139	0.0124	0.0112	0.0102	0.0093	0.0086	0.0079	0.0074	0.0069	0.0065	0.0061	·	,
	Sodium	0.0095	0.0080	0.0069	0.0062	0.0057	0.0053	0.0050	0.0047	0.0045	0.0044	0.0043	0.0042	0.0042	0.0041	0.0041	0.0041	0.0042	0.0042	0.0042	·	,
	Argon	$2.38 \cdot 10^{-5}$	$2.94 \cdot 10^{-5}$	$3.54 \cdot 10^{-5}$	$4.18\cdot10^{-5}$	$4.86\cdot10^{-5}$	$5.58 \cdot 10^{-5}$	$6.34 \cdot 10^{-5}$	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı
$[m^2.s^{-1}]$	LBE	$3.08 \cdot 10^{-7}$	$2.51 \cdot 10^{-7}$	$2.14\cdot 10^{-7}$	$1.88\cdot 10^{-7}$	$1.69 \cdot 10^{-7}$	$1.54\cdot 10^{-7}$	$1.43 \cdot 10^{-7}$	$1.34\cdot 10^{-7}$	$1.26 \cdot 10^{-7}$	$1.20 \cdot 10^{-7}$	$1.15 \cdot 10^{-7}$	$1.11 \cdot 10^{-7}$	$1.08 \cdot 10^{-7}$	$1.04\cdot 10^{-7}$	$1.02 \cdot 10^{-7}$	ı	ı	ı	'	ı	ı
diffusivity [$\operatorname{Bismuth}$		ı	ı	$1.82\cdot 10^{-7}$	$1.63\cdot 10^{-7}$	$1.48 \cdot 10^{-7}$	$1.37 \cdot 10^{-7}$	$1.28 \cdot 10^{-7}$	$1.21 \cdot 10^{-7}$	$1.15\cdot 10^{-7}$	$1.10\cdot 10^{-7}$	$1.06 \cdot 10^{-7}$	$1.02 \cdot 10^{-7}$	$9.88 \cdot 10^{-8}$	$9.62 \cdot 10^{-8}$	$9.39 \cdot 10^{-8}$	$9.19 \cdot 10^{-8}$	$9.01 \cdot 10^{-8}$	$8.86 \cdot 10^{-8}$	ı	
Viscous	Lead		1		ı	ı	$2.23 \cdot 10^{-7}$	$1.99 \cdot 10^{-7}$	$1.81 \cdot 10^{-7}$	$1.66 \cdot 10^{-7}$	$1.55\cdot 10^{-7}$	$1.45 \cdot 10^{-7}$	$1.37 \cdot 10^{-7}$	$1.30\cdot10^{-7}$	$1.25 \cdot 10^{-7}$	$1.20 \cdot 10^{-7}$	$1.15 \cdot 10^{-7}$	$1.12 \cdot 10^{-7}$	$1.08 \cdot 10^{-7}$	$1.06 \cdot 10^{-7}$	$1.03 \cdot 10^{-7}$	$1.01 \cdot 10^{-7}$
	Sodium	$6.52 \cdot 10^{-7}$	$5.40\cdot10^{-7}$	$4.63 \cdot 10^{-7}$	$4.08\cdot10^{-7}$	$3.67\cdot10^{-7}$	$3.35 \cdot 10^{-7}$	$3.11 \cdot 10^{-7}$	$2.91 \cdot 10^{-7}$	$2.74\cdot10^{-7}$	$2.61 \cdot 10^{-7}$	$2.49 \cdot 10^{-7}$	$2.40\cdot10^{-7}$	$2.32 \cdot 10^{-7}$	$2.25\cdot 10^{-7}$	$2.19 \cdot 10^{-7}$	$2.14\cdot10^{-7}$	$2.10\cdot 10^{-7}$	$2.06 \cdot 10^{-7}$	$2.03 \cdot 10^{-7}$	'	
	Argon	$3.58 \cdot 10^{-5}$	$4.43 \cdot 10^{-5}$	$5.34 \cdot 10^{-5}$	$6.31 \cdot 10^{-5}$	$7.35 \cdot 10^{-5}$	$8.44 \cdot 10^{-5}$	$9.58 \cdot 10^{-5}$	$1.08 \cdot 10^{-4}$	$1.20 \cdot 10^{-4}$	$1.33 \cdot 10^{-4}$	$1.46 \cdot 10^{-4}$	$1.60 \cdot 10^{-4}$	$1.75 \cdot 10^{-4}$	$1.89 \cdot 10^{-4}$	$2.05 \cdot 10^{-4}$	$2.20 \cdot 10^{-4}$	$2.36 \cdot 10^{-4}$	ı	ı	ı	
$[m^2.s^{-1}]$	LBE	$5.96 \cdot 10^{-6}$	$6.48\cdot 10^{-6}$	$7.00 \cdot 10^{-6}$	$7.54 \cdot 10^{-6}$	$8.08 \cdot 10^{-6}$	$8.62 \cdot 10^{-6}$	$9.17 \cdot 10^{-6}$	$9.72 \cdot 10^{-6}$	$1.03 \cdot 10^{-5}$	$1.08 \cdot 10^{-5}$	$1.14 \cdot 10^{-5}$	$1.20 \cdot 10^{-5}$	$1.26\cdot10^{-5}$	$1.31 \cdot 10^{-5}$	$1.37 \cdot 10^{-5}$	ı		ı	ı	ı	
diffusivity [$\operatorname{Bismuth}$		ı	ı	$8.44\cdot 10^{-6}$	$9.09\cdot 10^{-6}$	$9.72 \cdot 10^{-6}$	$1.03\cdot 10^{-5}$	$1.09 \cdot 10^{-5}$	$1.15\cdot 10^{-5}$	$1.21 \cdot 10^{-5}$	$1.27 \cdot 10^{-5}$	$1.33 \cdot 10^{-5}$	$1.39 \cdot 10^{-5}$			ı		ı	ı	ı	
Thermal	Lead	1	1		ı	ı	$1.05 \cdot 10^{-5}$	$1.10\cdot 10^{-5}$	$1.15 \cdot 10^{-5}$	$1.20\cdot10^{-5}$	$1.25\cdot10^{-5}$	$1.30 \cdot 10^{-5}$	$1.35 \cdot 10^{-5}$	$1.40 \cdot 10^{-5}$	$1.45 \cdot 10^{-5}$	$1.51 \cdot 10^{-5}$	$1.56 \cdot 10^{-5}$	$1.62 \cdot 10^{-5}$	$1.67 \cdot 10^{-5}$	$1.73 \cdot 10^{-5}$	'	
	Sodium	$6.88 \cdot 10^{-5}$	$6.79 \cdot 10^{-5}$	$6.69 \cdot 10^{-5}$	$6.59 \cdot 10^{-5}$	$6.49 \cdot 10^{-5}$	$6.38 \cdot 10^{-5}$	$6.27 \cdot 10^{-5}$	$6.16\cdot10^{-5}$	$6.05 \cdot 10^{-5}$	$5.93 \cdot 10^{-5}$	$5.81 \cdot 10^{-5}$	$5.69 \cdot 10^{-5}$	$5.56 \cdot 10^{-5}$	$5.44 \cdot 10^{-5}$	$5.31 \cdot 10^{-5}$	$5.18\cdot10^{-5}$	$5.05 \cdot 10^{-5}$	$4.91 \cdot 10^{-5}$	$4.78 \cdot 10^{-5}$	$4.64 \cdot 10^{-5}$	$4.50 \cdot 10^{-5}$
[K] Temp.		400	450	500	550	600	650	200	750	800	850	000	950	$1 \ 000$	$1 \ 050$	1 100	1 150	$1 \ 200$	$1 \ 250$	$1 \ 300$	$1 \ 350$	1 400

207

Appendix F

Proofs and justification

This chapter gathers proofs and arguments for several assertions encountered in different parts of the thesis.

F.1 Single-fluid VPM method

F.1.1 Procedure to obtain velocity from vorticity (Eq. 2.3)

A Helmholtz decomposition is performed on velocity so as to split it into an curl-free $\nabla \phi$ and a divergence-free $\nabla \times \mathbf{A}$ components

$$\boldsymbol{u} = \nabla \times \boldsymbol{A} + \nabla \phi \tag{F.1}$$

where ϕ is the velocity potential and \boldsymbol{A} reduces to the stream function $\psi \hat{\boldsymbol{e}}_z$ in 2D. This decomposition is not unique in \boldsymbol{A} and we use the gauge condition $\nabla \cdot \boldsymbol{A} = 0$ to enforce unicity.

Since the flow is incompressible the velocity field is solenoïdal $\nabla \cdot \boldsymbol{u} = 0$ we have $\nabla^2 \phi = 0$. Moreover since the flow domain is enclosed in a box we have a no through flow boundary condition that transaltes to $\nabla \phi = 0$ on the domain boundary $\partial \Omega$. This implies $\phi = cte$ in the whole domain Ω , hence the curl-free component of velocity is zero $\nabla \phi = 0$ which yields

$$\boldsymbol{u} = \nabla \times \boldsymbol{A} \tag{F.2}$$

Noting that the curl of velocity is vorticity by definition, one has

$$\omega = \nabla \times \nabla \times \boldsymbol{A} = -\nabla^2 \boldsymbol{A} = -\nabla^2 (\psi \, \boldsymbol{\hat{e}}_z) \tag{F.3}$$

given the gauge condition $\nabla \cdot \mathbf{A} = 0$. Moreover for flows within a closed box, the stream function is constant on the boundary and hence chosen to be zero on $\partial \Omega$. Eventually velocity can be retrieved from the stream function through

$$\boldsymbol{u} = \nabla \times \psi \, \boldsymbol{\hat{e}}_z \tag{F.4}$$

F.2 Interface capturing VPM method

F.2.1 Equivalence between Rouy scheme and upwinding

Figure F.1 shows the different cases that can be encountered when reinitialising. The purpose of Rouy's and Tourin's scheme is to select the downwinding scheme for each of those cases such as to preserve the stability of the method. Additionally it erases local minimas. The first line (figures (a) to (e)) corresponds to reinitialisation inside the exterior region Ω^+ which is the region of the domain Ω where the level set field is positive. On the other hand, the bottom line (figures (f) to (j)) corresponds to the interrior region Ω^- where the level set field is negative.

The equivalence will be shown in the exterior region case only (Eq. 3.19a) for a one-dimensional problem. A similar reasoning will lead to a corresponding result in the interior region Ω^- . Moreover, the original first order decentered schemes will be considered, as in Rouy's and Tourin's study [173]

$$D_x^- \phi = \frac{\phi_{i,j} - \phi_{i-1,j}}{h_x} \tag{F.5a}$$

$$D_x^+ \phi = \frac{\phi_{i+1,j} - \phi_{i,j}}{h_x} \tag{F.5b}$$

Three differents types of cases can be seen: (i) monotone level set fields i.e. level set fields increasing away from the interface (ii) the vincinity of local maximas where the two first order schemes $D_x^-\phi$ and $D_x^+\phi$ evaluate a slope of opposite sign, and (iii) the cup situation.

The Rouy scheme consists in using the downwinding scheme in all cases except the cup case where all values are increased at a rate $\frac{\partial \phi_{reinit}}{\partial \tau}$ of one.

Downwinding for the reinitialisation equation

Considering an one-dimensional advection equation on level set ϕ

$$\frac{\partial \phi}{\partial t} + c \frac{\partial \phi}{\partial x} = Rhs(\phi, t)$$
 (F.6)

where c is the celerity, a traditional downwinding works as

$$D_x \phi = \begin{cases} D_x^- \phi & \text{if } c > 0\\ D_x^+ \phi & \text{if } c < 0 \end{cases}$$
(F.7)

In the case of the level set reinitialisation equation, the celerity is given by the x-component of the normal vector $\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{e}}_x$ which is computed as

$$\hat{\boldsymbol{n}} = \frac{\nabla \phi}{\|\nabla \phi\|} \tag{F.8}$$

using 2nd order centered finite differences. Hence

$$c = \frac{\phi_{i+1} - \phi_{i-1}}{\|\phi_{i+1} - \phi_{i-1}\|}$$
(F.9)



Figure F.1: Upwinding/downwinding rules enforced by (Eq. 3.19a) (Eq. 3.19b) in a one-dimensional case. (•) is an interface point \boldsymbol{x}_{Γ} , (••••••) is the level set profile ϕ . Additionally, (•) is a point of the grid where the value of the level set field is known, (•••••) are two stencils of first order decentered (one downwind and one upwind) schemes for $\frac{\partial \phi}{\partial x}$. (••••••) (resp. (•••••)) indicates that the downwind (resp. upwind) stencil is used, while (••••••) denotes that the advective term is set to zero.

Moreover the 2nd order finite difference can be expressed as the average of both first order schemes

$$\frac{\phi_{i+1} - \phi_{i-1}}{2h_x} = \frac{1}{2} \left(D_x^- \phi + D_x^+ \phi \right)$$
(F.10)

This yields

$$c = \begin{cases} +1 & \text{if } D_x^- \phi + D_x^+ \phi > 0 \\ -1 & \text{if } D_x^- \phi + D_x^+ \phi < 0 \end{cases}$$
(F.11)

The generic downwinding scheme specified in equation F.7 can hence be expressed as

$$D_x \phi = \begin{cases} D_x^- \phi & \text{if } D_x^- \phi + D_x^+ \phi > 0\\ D_x^+ \phi & \text{if } D_x^- \phi + D_x^+ \phi < 0 \end{cases}$$
(F.12)

or more elegantly

$$D_x\phi| = \left|\max(D_x^-\phi, -D_x^+\phi)\right| \tag{F.13}$$

in the case of a level set reinitialisation equation.

General expression

The Rouy scheme performs downwinding in cases (a) to (d) which translates as equation F.13. Additionally in the case of a cup (Fig. F.1f), the time derivative $\frac{\partial \phi}{\partial \tau}$ is set to one such that the cup will eventually vanish. The cup case is

caracterised by $D_x^- \phi < 0$ and $D_x^+ \phi > 0$, and in such a situation the scheme must set $||D_x \phi||$ to zero. To summarise one has

$$\begin{cases} |D_x\phi| = 0 & \text{if } D_x^-\phi < 0 \text{ and } D_x^+\phi > 0\\ |D_x\phi| = |\max(D_x^-\phi, -D_x^+\phi)| & \text{otherwise} \end{cases}$$
(F.14)

This can be expressed as

$$|D_x\phi| = \left|\max(0, D_x^-\phi, -D_x^+\phi)\right|$$
 (F.15)

which is nothing less than the Rouy scheme for the exterior region Ω^+ .

F.2.2 Level set flip-flop

Consider an interface Γ_k such its points (x, y) observe

$$y = \cos(kx) \tag{F.16}$$

At each summit node x_i defined as

$$x_i = i \frac{\pi}{k} , \ \forall i \in \mathbb{N}$$
 (F.17)

we have a symmetry

$$y(x_i - x) = y(x_i + x) , \ \forall x \in \mathbb{R} \ \forall i \in \mathbb{N}$$
 (F.18)

and hence the surface tension term generates a symmetrical vertical acceleration

$$a(x_i - x) = a(x_i + x) , \forall x \in \mathbb{R} \quad \forall i \in \mathbb{N}.$$
 (F.19)

Hence vorticity at those nodes is equal to

$$\omega(\boldsymbol{x}_i) = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} = 0.$$
 (F.20)

In the case of the flip-flop mode, the wave number k is such that all grid nodes correspond to x_i points. In other words the vorticity is exactly zero at all grid nodes.

F.3 Viscous term

F.3.1 Correct expansion of viscous term (Eq. 4.37)

Thirifay's demonstration [115] assumes that for any vector \boldsymbol{u} and scalar λ one has

$$\nabla^2 \boldsymbol{u} \times \nabla \lambda = \nabla^2 \boldsymbol{u} \times \nabla \lambda + \boldsymbol{u} \times \nabla^2 \nabla \lambda \qquad (F.21)$$

which is incorrect.

Instead one has for any two vectors \boldsymbol{u} and \boldsymbol{v}

$$\nabla^2 \boldsymbol{u} \times \boldsymbol{v} = \nabla^2 \boldsymbol{u} \times \boldsymbol{v} + \boldsymbol{u} \times \nabla^2 \boldsymbol{v} + 2 \sum_{i=1}^3 \epsilon_{ipq} \frac{\partial u_p}{\partial x_i} \frac{\partial v_q}{\partial x_i}$$
(F.22)

where ϵ_{ipq} is the Levi-Civita symbol. Hence the actual general result for a compressible flow is then

$$\nabla \times \nabla \cdot \boldsymbol{\tau} = \mu \nabla^2 \boldsymbol{\omega} + 2 \nabla \mu \times (\nabla \theta - \nabla \times \boldsymbol{\omega}) + \nabla (\boldsymbol{\omega} \cdot \nabla \mu) - \boldsymbol{\omega} \nabla^2 \mu - 2 \sum_{i=1}^3 \epsilon_{ipq} \frac{\partial u_p}{\partial x_i} \frac{\partial (\nabla \mu)_q}{\partial x_i}$$
(F.23)

where θ is the dilatation and is defined as $\theta := \nabla \cdot \boldsymbol{u}$.

Omitting the trailing term alters significantly the simulations' outputs. To demonstrate this assertion, the rising bubble benchmark (see section 6.9) is run with the correct and the incorrect forms of the equation (Fig. 6.10).

F.4 High mass-density ratio VPM method

F.4.1 Material acceleration sequence for any time integrator (Eq. 5.32)

Given any equation

$$\frac{\partial v'}{\partial t} = Rhs(v', t) \quad \forall t \in]t^n, t^{n+1}[$$
(F.24)

where $Rhs(\cdot)$ is defined as

$$Rhs(v,t) := \mathscr{I}(\mathbf{D}_{t}v') \tag{F.25}$$

one has

$$Rhs_{k} = \mathscr{I}\left(\frac{\left(v'^{n} + h_{t}\sum_{k'=1}^{k-1}a_{kk'}Rhs_{k'}\right) - v'^{n-1}}{(t^{n} + c_{k}h_{t}) - t^{n-1}}\right)$$
$$= \frac{1}{1 + c_{k}}\mathscr{I}\left(D_{t}v'^{n}\right) + \frac{1}{1 + c_{k}}\sum_{k'=1}^{k-1}a_{kk'}\mathscr{I}\left(Rhs_{k'}\right)$$

Thus, by recursion

$$Rhs_{k_{1}} = \frac{1}{1+c_{k_{1}}}\mathscr{I}(\mathbf{D}_{t}v'^{n}) + \frac{1}{1+c_{k_{1}}}\sum_{k_{2}=1}^{k_{1}-1}\frac{a_{k_{1}k_{2}}}{1+c_{k_{2}}}\mathscr{I}(\mathbf{D}_{t}v'^{n}) \\ + \frac{1}{1+c_{k_{1}}}\sum_{k_{2}=1}^{k_{1}-1}\frac{a_{k_{1}k_{2}}}{1+c_{k_{2}}}\sum_{k_{3}=1}^{k-1}\frac{a_{k_{2}k_{3}}}{1+c_{k_{3}}}\mathscr{I}(\mathbf{D}_{t}v'^{n}) + [...]$$

Hence

$$D_{t}v'^{n+1} = \frac{v'^{n+1} - v'^{n}}{t^{n+1} - t^{n}} = \sum_{k_{1}=1}^{m} b_{k_{1}} Rhs_{k_{1}}$$

$$= \sum_{k_{1}=1}^{m} \frac{b_{k_{1}}}{1 + c_{k_{1}}} \mathscr{I}(D_{t}v'^{n}) + \sum_{k_{1}=1}^{m} \frac{b_{k_{1}}}{1 + c_{k_{1}}} \sum_{k_{2}=1}^{k_{1}-1} \frac{a_{k_{1}k_{2}}}{1 + c_{k_{2}}} \mathscr{I}^{2}(D_{t}v'^{n})$$

$$+ \sum_{k_{1}=1}^{m} \frac{b_{k_{1}}}{1 + c_{k_{1}}} \sum_{k_{2}=1}^{k_{1}-1} \frac{a_{k_{1}k_{2}}}{1 + c_{k_{2}}} \sum_{k_{3}=1}^{k_{2}-1} \frac{a_{k_{2}k_{3}}}{1 + c_{k_{3}}} \mathscr{I}^{3}(D_{t}v'^{n}) + [...]$$

$$= \sum_{k_{1}=1}^{m} \frac{b_{k_{1}}}{1 + c_{k_{1}}} \mathscr{I}(D_{t}v'^{n}) + \sum_{k_{1}=1}^{m} \sum_{k_{2}=1}^{k_{1}-1} \frac{b_{k_{1}}a_{k_{1}k_{2}}}{(1 + c_{k_{1}})(1 + c_{k_{2}})} \mathscr{I}^{2}(D_{t}v'^{n})$$

$$+ \sum_{k_{1}=1}^{m} \sum_{k_{2}=1}^{k_{2}-1} \sum_{k_{3}=1}^{k_{2}-1} \frac{b_{k_{1}}a_{k_{1}k_{2}}a_{k_{2}k_{3}}}{(1 + c_{k_{1}})(1 + c_{k_{3}})} \mathscr{I}^{3}(D_{t}v'^{n}) + [...]$$

F.5 Numerics and solver validation

F.5.1 Volume calculation using mollifier

$$\int_{V} = \frac{1}{\dim(\Omega)} \sum_{i=1}^{\dim(\Omega)} \left(\int_{V} 1 \right) = \frac{1}{\dim(\Omega)} \int_{V} \nabla \cdot \boldsymbol{x}$$

$$= \frac{1}{\dim(\Omega)} \int_{\partial V} \boldsymbol{x} \cdot \hat{\boldsymbol{n}} \simeq \frac{1}{\dim(\Omega)} \int_{\Omega} \delta_{\epsilon} \boldsymbol{x} \cdot \hat{\boldsymbol{n}}$$
(F.26)

Appendix G

Viscous term in vorticity equation

The present appendix presents and derivates different ways to express the viscous term of the vorticity equation. Framed equations corresponds to an expression of the viscous term that can be implemented directly in the solver.

Firstly, let us recall that the vorticity equation is

$$\frac{D\boldsymbol{\omega}}{Dt} = -\frac{\nabla p}{\rho} \times \frac{\nabla \rho}{\rho} + \nabla \times \left(\frac{1}{\rho} \nabla \cdot \underline{\boldsymbol{\tau}}\right)$$
(G.1)

It becomes

$$\frac{D\boldsymbol{\omega}}{Dt} = \left(\frac{D\boldsymbol{u}}{Dt} - \boldsymbol{g}\right) \times \frac{\nabla\rho}{\rho} + \frac{1}{\rho} \nabla \times \left(\nabla \cdot \underline{\boldsymbol{\tau}}\right)$$
(G.2)

after injection of the momentum equation in the pressure term. Note that the $\frac{1}{\rho}$ ratio is now outside of the curl operator

$$\underline{\underline{\tau}} = 2\mu \underline{\underline{S}} + \left(\mu_v - \frac{2}{3}\mu\right)\theta \underline{\underline{I}}$$
 (G.3)

with $\underline{\underline{S}}$ the strain tensor

$$\underline{\underline{S}} = \frac{1}{2} \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T \right)$$
(G.4)

and θ the compression rate

$$\boldsymbol{\theta} = \nabla \cdot \boldsymbol{u} \tag{G.5}$$

Taking the divergence of the stress tensor yields

$$\nabla \cdot \underline{\underline{\tau}} = -\mu \nabla \times \boldsymbol{\omega} + \left(\mu_v + \frac{4}{3}\mu\right) \nabla \theta + 2\underline{\underline{S}} \cdot (\nabla \mu) + \nabla \mu_v - \frac{2}{3}\mu \quad (G.6)$$

and taking its curl results in

$$\nabla \times (\nabla \cdot \underline{\underline{\tau}}) = \mu \nabla^2 \boldsymbol{\omega} + (\nabla \mu) \times (\nabla^2 \boldsymbol{u} + \nabla \theta) + \nabla \times (2 \underline{\underline{S}} \cdot (\nabla \mu))$$
(G.7)

or, using $\nabla \times \boldsymbol{\omega} = -\nabla^2 \boldsymbol{u} + \nabla \theta$

$$\nabla \times (\nabla \cdot \underline{\underline{\tau}}) = \mu \nabla^2 \boldsymbol{\omega} + (\nabla \mu) \times (2 \nabla \theta - \nabla \times \boldsymbol{\omega}) + \nabla \times (2 \underline{\underline{S}} \cdot (\nabla \mu))$$
(G.8)

The latter equation corresponds to equation C.19 in Thirifay's thesis [115]. Additionnally, the $\nabla \times (2\underline{\underline{S}} \cdot (\nabla \mu))$ term can be expanded

$$2\underline{\underline{\boldsymbol{S}}} \cdot (\nabla \mu) = \nabla \left(\boldsymbol{u} \cdot (\nabla \mu) + \nabla \times \left(\boldsymbol{u} \times (\nabla \mu) \right) \right) - (\nabla^2 \mu) \boldsymbol{u} + \theta (\nabla \mu)$$
(G.9)

which yields¹

$$\nabla \times \left(2\underline{\underline{\mathbf{S}}} \cdot (\nabla \mu)\right) = \nabla \times \left(\nabla \times (\mathbf{u} \times (\nabla \mu))\right) - \nabla \times \left((\nabla^2 \mu) \mathbf{u}\right) + \nabla \times \left(\theta (\nabla \mu)\right)$$
(G.10)

Additionnally, since

$$\nabla \times \left(\theta \,\nabla \mu\right) = \nabla \theta \times \nabla \mu \tag{G.11}$$

one has

$$\nabla \times \left(2\underline{\underline{\mathbf{S}}} \cdot (\nabla \mu)\right) = \nabla \times \left(\nabla \times (\mathbf{u} \times (\nabla \mu))\right) - \nabla \times \left((\nabla \mu) \mathbf{u}\right) + \nabla \theta \times \nabla \mu \quad (G.12)$$

which results in

$$\nabla \times (\nabla \cdot \underline{\boldsymbol{\tau}}) = \mu \nabla^2 \boldsymbol{\omega} + (\nabla \mu) \times (\nabla^2 \boldsymbol{u}) + \nabla \times \left(\nabla \times (\boldsymbol{u} \times (\nabla \mu)) \right) - \nabla \times \left((\nabla \mu) \boldsymbol{u} \right)$$
(G.13)

This expression can be further expanded as well

$$\nabla \times \left(\nabla \times \left(\boldsymbol{u} \times (\nabla \mu) \right) \right) = -\nabla^2 \boldsymbol{u} \times (\nabla \mu) + \nabla \left(\boldsymbol{\omega} \cdot (\nabla \mu) \right)$$
(G.14)

and

$$\nabla \times \left((\nabla^2 \mu) \boldsymbol{u} \right) = \left(\nabla \left(\nabla^2 \mu \right) \right) \times \boldsymbol{u} + (\nabla^2 \mu) \boldsymbol{\omega}$$
 (G.15a)

$$= \left(\nabla^2 \left(\nabla \mu\right)\right) \times \boldsymbol{u} + (\nabla^2 \mu) \boldsymbol{\omega}$$
 (G.15b)

and yields

$$\nabla \times (\nabla \cdot \underline{\boldsymbol{\tau}}) = \mu \nabla^2 \boldsymbol{\omega} + (\nabla \mu) \times (\nabla^2 \boldsymbol{u}) + \left(-\nabla^2 \boldsymbol{u} \times (\nabla \mu) + \nabla \left(\boldsymbol{\omega} \cdot (\nabla \mu) \right) \right) + \left(\left(\nabla \left(\nabla^2 \mu \right) \right) \times \boldsymbol{u} + (\nabla^2 \mu) \boldsymbol{\omega} \right)$$
(G.16)

Also, Equation G.13 can be expanded in a different way

$$\nabla \times \left((\nabla \mu \times \boldsymbol{u}) \right) = \left(\nabla (\nabla \mu) \right) \cdot \boldsymbol{u} - (\nabla \boldsymbol{u}) \cdot (\nabla \mu) + \theta \left(\nabla \mu \right) - (\nabla^2 \mu) \boldsymbol{u}$$
(G.17)

 $^{^1\}mathrm{Same}$ as equation C.21 in Thirifay's thesis [115].

which yields

$$\nabla \times (\nabla \cdot \underline{\underline{\tau}}) = \mu \nabla^2 \boldsymbol{\omega} + (\nabla \mu) \times (\nabla^2 \boldsymbol{u}) + \nabla \times \left((\nabla \boldsymbol{u} \cdot (\nabla \mu - (\nabla (\nabla \mu)) \cdot \boldsymbol{u} - \theta (\nabla \mu)) \right)$$
(G.18)