

# Towards a unified Eulerian modeling framework for two-phase flows: geometrical small-scale phenomena and associated flexible computing strategies

Thèse de doctorat de l'Institut Polytechnique de Paris  
préparée à l'École polytechnique

École doctorale n°574 École doctorale de mathématiques Hadamard (EDMH)  
Spécialité de doctorat : Mathématiques aux interfaces

Thèse présentée et soutenue à Palaiseau, le 20/09/2021, par

**RUBEN DI BATTISTA**

Composition du Jury :

Christian TENAUD Directeur de recherche, Centralsupélec (EM2C)	Président
Jean-Marc HÉRARD Directeur de Recherche, EDF (R&D)	Rapporteur
Stéphane VINCENT Professeur, Université Gustave Eiffel (MSME)	Rapporteur
Marica PELANTI Assistant Professor, ENSTA (IMSIA)	Examineur
Paolo BARBANTE Directeur de Recherche, Politecnico di Milano (Dpt. of Mathematics)	Examineur
Stéphane DE CHAISEMARTIN Ingénieur de Recherche, IFPEN (R115)	Examineur
Marc MASSOT Professeur, École polytechnique (CMAP)	Directeur de thèse
Samuel Kokh Ingénieur de recherche, CEA (Direction de l'Energie Nucléaire)	Co-directeur de thèse
Franck HERVY Directeur de recherche, DGA (AID)	Invité
Pierre KESTENER Ingénieur de recherche, CEA (DRF/IRFU/DEDIP/LILAS)	Invité







# **TOWARDS A UNIFIED EULERIAN MODELING FRAMEWORK FOR TWO-PHASE FLOWS: GEOMETRICAL SMALL-SCALE PHENOMENA AND ASSOCIATED FLEXIBLE COMPUTING STRATEGIES**

**Ruben Di Battista**

**November 3, 2021**

Thesis submitted for the degree of Doctor of Philosophy of the Institut Polytechnique de Paris, prepared at École polytechnique, Centre de Mathématiques Appliquées

*Ai sorrisi autentici di Julia*  
*To Julia and her authentic smiles*

This thesis has been funded by a French Defense grant provided by the *Direction générale de l'armement* and by a thesis grant provided by IFPEN. Their support is here gratefully acknowledged.

# Acknowledgments

*If you want to go fast, go alone. If you  
want to go far, go together*  
— African Proverb

I love to believe that the life path I walked during the period I have been working on this thesis brought me far. If we assume this as an hypothesis, then I am absolutely no one to contradict wise African proverbs. I felt I have been not alone. The first words I want to mention are undoubtedly dedicated to my family. Even if they have not had the opportunity of a high-profile education, they kept pushing me to overcome my limits, to leave my comfort zone, and to reach for the “Pillars of Hercules” of my own world, looking at the stars. And as far as this looks to me incredible, how they spontaneously embody such a wide and modern vision, I am, and always I will be, profoundly grateful to them for the education and principles they forwarded to me. After the family by blood right, I feel to mention the family by serendipity. My friend Jon, the closest thing to a soul mate I can probably imagine, despite the effective physical distance, succeeded in enduring the time and space invariant friendship that unites us together with Paola. Talking about friendship, Lorenzo and Michele have been a constant presence and support in my life since the time we met in Princeton, years ago, and never let back despite my difficult personality and my good and less good choices navigating the turbulent waters of life. Out of the borders of the “Bel Paese”, I would like to thank the friendship I have been able to build with Pierre. He has shared my crazy ideas and my need for craving novelty and I really, incredibly, loved the good time that came out of it. I hope we can keep feeding this hungry enthusiasm in the future. On the same spirit I would like to mention Arthur, Quentin, Marc-Arthur, team mates, adventure mates, bright minds. Thomas, and his absolute unbreakable genuinity. I am grateful for the opportunity of discussing about so many topics that really opened up my mind. Léa and Alejandro, that needed to cope with me opening the windows also during freezing cold winters with the urge of breathing out my frustrations. A special tribute goes to Alberto, again, serendipity. I had the chance of meeting a good person above all. A genuine fellow. An incredible friend that shared uncountable adventures in France and colored my days in Technicolor. Someone I would like to steal the ability of making friends so easily from. Thanks to Stefano, a lifesaving rock with which to talk about hard problems, with soft mood. Josie, Sonia and Alfredo met thanks to Alberto, that brought unstructured happiness to the structured life of a boring engineer. Cristina and Matteo and the simplicity of being good friends. Roxane, that had the patience to endure me and my ramblings about space, SpaceX, Elon Musk and all that nonsense, and Louis, that instead fed that indomitable fire. You have really

been a breath of fresh air, at the beginning of summer, on the Brazilian sea. I wish you the most happy life together. Speaking about love, a tribute to Julia that shared part of this journey with me. She randomly picked me on the edge of a tortuous path and diligently conducted me out of the bush with her authentic smiles. I would like to spend some words to gratefully acknowledge the administrative support I had from Nasséra and Alexandra at École Polytechnique, I know I have been a problems magnet, but you really, actively, helped me with always a smile. Sorry, instead, to Pierre and Sylvain. I know I have been a real PITA about the IT infrastructure in the lab, but I love to think that I helped pushing forward the status quo. Thanks for bearing with my countless requests. A grateful appreciation to the Centre Spatial de l'École Polytechnique, to Florian, Antoine and Nicolas notably, that allowed me to pursue my rocketry passions, launch a rocket, win a prize, and have so much fun, hopefully instating the same passion to young generations. A special tribute to my supervisor Samuel, that really kept my mood up also in the darker days. I have appreciated the genuinity of his life teachings and the talks about open source. From a supervisor to the other: Stéphane de Chaisemartin. I am grateful for the dynamic and proactive support during the thesis, reinforcing my need of doing things at my own pace, in my own way, with always a word of comfort. A similar praise I address to Stéphane Jay. My final words are then dedicated to my supervisor Marc. Marc is not a conventional supervisor, or at least he was not one for me. I felt more than a student, I felt being part of his family. I felt understood by someone that deeply shared my discomfort for status quo. I felt someone that saw the wrong sides of my soul and helped me smoothing the edges (with debatable success). I never felt judged for my different visions, but only supported. I am immensely grateful for the uncountable opportunities Marc felt no problem offering me, the sharing of small life pleasures like wine bottles, familiar dinners, jazz music and Californian sun. I know I am not an easy individual, and I am thankful for the friendship before the supervisorship. I count on pursuing our friendship also after the end of this adventure.

# Contents

Acknowledgments	iii
Contents	x
Figures	xiii
Tables	xv
Algorithms	xvii
Sommario	xix
Résumé	xxi
Abstract	xxiii
Notes on notation	xxv
<b>1 Introduction</b>	<b>1</b>
1.1 Contexte industriel . . . . .	1
1.1.1 Systèmes d’injection pour la propulsion spatiale . . . . .	1
1.1.2 Système d’injection pour la propulsion aérobie . . . . .	4
1.1.3 Système d’injection pour les moteurs injection directe . . . . .	6
1.2 Contexte scientifique . . . . .	6
1.2.1 Différents modèles pour différents régimes . . . . .	7
1.2.2 Modèles pour résoudre toutes les échelles . . . . .	8
1.2.3 Modèles d’ordre réduit . . . . .	9
1.2.4 Un cadre complet pour la dérivation et la simulation de modèles unifiés d’ordre réduit d’écoulement diphasique . . . . .	10
1.3 Contexte de calcul scientifique . . . . .	11
1.4 Contributions de cette thèse . . . . .	13
<b>2 Introduction</b>	<b>17</b>
2.1 Industrial Context . . . . .	17
2.1.1 Injection systems for space propulsion . . . . .	17
2.1.2 Injection systems for air-breathing propulsion . . . . .	24
2.1.3 Direct injection in reciprocating engines . . . . .	25

2.2	Scientific Context . . . . .	26
2.2.1	Different models for different regimes . . . . .	29
2.2.1.1	Models that aim at resolving all the scales . . . . .	29
2.2.1.2	Reduced order Models . . . . .	30
2.2.2	A comprehensive framework for the derivation and simulation of unified two-phase flow reduced order models . . . . .	32
2.3	Scientific computing context . . . . .	33
2.4	Contributions of this thesis . . . . .	34
<b>3</b>	<b>Two-phase flow models and where to find them</b>	<b>39</b>
3.1	Introduction . . . . .	41
3.2	Deriving two-phase flow models with an averaging procedure . . . . .	42
3.2.1	Interface between phases and jump conditions . . . . .	42
3.2.2	Averaged governing equations . . . . .	44
3.2.2.1	Closure for the interfacial quantities . . . . .	47
3.2.2.2	The volume fraction transport equation . . . . .	48
3.2.2.3	The interfacial area density equation . . . . .	50
3.3	Deriving two-phase flow models with a variational approach . . . . .	51
3.3.1	Variational principles for continua . . . . .	51
3.3.1.1	Variations for a family of trajectories . . . . .	54
3.4	Applying the Stationary Action Principle to a fluid problem . . . . .	55
3.4.1	Extremalizing the Hamiltonian Action . . . . .	57
3.4.1.1	Variation of the velocity . . . . .	57
3.4.1.2	Variation of the jacobian . . . . .	57
3.4.1.3	Variation of the density . . . . .	58
3.4.1.4	Variation of a field governed by a transport equation . . . . .	58
3.4.1.5	Variation of the volume fraction . . . . .	59
3.4.1.6	Generic form of the system . . . . .	59
3.4.1.7	Barotropic Euler system . . . . .	61
3.4.2	Introducing the dissipation . . . . .	61
3.5	Two-phase flow models with a rich geometrical description . . . . .	63
3.5.1	What do we mean by two-scale modeling . . . . .	64
3.5.2	Two-scale modeling of small inclusions assuming normal deformation of the interface . . . . .	67
3.5.2.1	Basic elements of the Weyl's tube formula . . . . .	67
3.5.2.2	Application of the SAP . . . . .	69
3.5.3	Two-scale modeling of small inclusions assuming incompressible constant volume oscillations . . . . .	72
3.5.3.1	Energy models for the small scale inclusions . . . . .	73
3.5.3.2	Application of the SAP . . . . .	75
3.5.3.3	Dissipation . . . . .	77
3.5.3.4	Analogy with O'Rourke and Amsden TAB model and limitations . . . . .	78

3.6	Interfacial area density models via the Stationary Action Principle . . . . .	79
3.6.1	Application of the SAP . . . . .	80
3.6.2	Including source terms through dissipation, analogy with Daniel Lhuillier (2004) . . . . .	83
3.7	Conclusions and Perspectives . . . . .	86
<b>Appendices</b>		<b>89</b>
3.A	Variation of the jacobian . . . . .	89
3.B	Energy analytical calculation of a deformed droplet . . . . .	92
3.B.1	Approximation of the droplet deformation . . . . .	92
3.B.2	Kinetic energy . . . . .	96
3.B.3	Potential energy . . . . .	97
3.B.4	Mean curvature . . . . .	98
<b>4</b>	<b>A computational framework based on the discrete estimation of geometrical properties over triangulated interfaces to perform validation of two-phase flow models</b>	<b>101</b>
4.1	Introduction . . . . .	102
4.2	Estimation of differential geometry properties on surfaces . . . . .	104
4.2.1	Geometrical properties for surfaces parametrized explicitly . . . . .	105
4.2.2	Geometrical properties for surfaces parametrized implicitly with a level set function . . . . .	105
4.2.3	The Gauss-Bonnet theorem . . . . .	107
4.2.4	Semipositivity of the Willmore Energy . . . . .	107
4.3	Curvature estimation from a discrete level-set volumetric field . . . . .	108
4.3.1	General overview of the post-processing procedure . . . . .	108
4.3.2	One traditional way of estimating curvatures . . . . .	109
4.3.3	Discrete iso-contouring algorithms . . . . .	109
4.3.4	Geometrical properties as 1-ring averaged values . . . . .	110
4.4	Verification and issues on canonical objects . . . . .	114
4.4.1	A set of verification cases on canonical objects and associated issues . . . . .	114
4.4.2	Solutions to the identified issues . . . . .	118
4.4.2.1	An averaging kernel that preserves the Gauss Bonnet Theorem . . . . .	118
4.4.2.2	A projection method . . . . .	120
4.4.2.3	Improving the quality of the mesh . . . . .	120
4.5	An application of the post-processing of DNS simulations: the collision of two droplets . . . . .	120
4.5.1	DNS configuration . . . . .	121
4.5.2	Topological footprint analysis . . . . .	121
4.5.3	Willmore Energy analysis . . . . .	127
4.5.3.1	Counting object homeomorphic to a sphere . . . . .	127
4.6	Post-processing of DNS as a modeling tool: the oscillation of an ellipsoidal droplet . . . . .	128
4.6.1	Introduction . . . . .	128

4.6.2	DNS configuration . . . . .	129
4.6.3	Analysis . . . . .	130
4.7	Conclusions . . . . .	137
4.8	Perspectives . . . . .	137
<b>Appendices</b>		<b>141</b>
4.A	Algorithm to handle multiblock simulations . . . . .	141
<b>5</b>	<b>Numerical schemes and simulation of two-phase flows</b>	<b>143</b>
5.1	Introduction . . . . .	143
5.2	Basic concepts for the discretization of (a system of) Partial Differential Equations (PDEs) with the Finite Volume Method (FVM) . . . . .	144
5.2.1	Structured mesh generation . . . . .	145
5.2.1.1	Structured boundary-fitted meshes . . . . .	146
5.2.1.2	Trans-Finite interpolation . . . . .	147
5.2.1.3	More advanced meshing techniques . . . . .	148
5.2.2	Discretization of the convective term . . . . .	149
5.2.2.1	Eigenstructure . . . . .	150
5.2.2.2	Weak solutions and Rankine-Hugoniot relations . . . . .	153
5.2.2.3	The Riemann Problem . . . . .	155
5.2.2.4	The Godunov method . . . . .	156
5.2.2.5	Approximate Riemann Solvers . . . . .	157
5.2.2.6	Higher order methods . . . . .	158
5.2.3	Discretization of the non-conservative term . . . . .	160
5.2.4	Discretization of the diffusive term . . . . .	161
5.2.4.1	Forward differencing . . . . .	161
5.2.4.2	Green-Gauss . . . . .	162
5.2.4.3	Least Squares gradient . . . . .	162
5.2.5	Discretization of the source term . . . . .	165
5.2.6	Time integration . . . . .	166
5.2.6.1	The CFL Condition . . . . .	167
5.3	The Euler system . . . . .	168
5.3.1	Eigenstructure . . . . .	169
5.3.2	An exact Riemann solver for a generic Equation of State (EoS) . . . . .	170
5.3.2.1	Rarefaction wave . . . . .	170
5.3.2.2	Shock wave . . . . .	171
5.3.2.3	Contact discontinuity . . . . .	172
5.3.2.4	Complete Solution . . . . .	173
5.3.2.5	Notes on the numerical schemes . . . . .	175
5.3.3	Description of some useful schemes . . . . .	175
5.3.3.1	The Rusanov scheme . . . . .	175
5.3.3.2	The HLL scheme . . . . .	176
5.3.3.3	The HLLC scheme . . . . .	176




5.4	The Baer-Nunziato system . . . . .	179
5.4.1	Eigenstructure . . . . .	181
5.4.2	The Rusanov scheme . . . . .	181
5.5	A two-phase two-scale model featuring geometrical fields . . . . .	181
5.5.1	Eigenstructure . . . . .	182
5.5.2	Recasting of the energy conservation for numerical implementation purposes . . . . .	184
5.5.3	Instantaneous relaxation of the phasic pressures . . . . .	184
5.5.4	The final system . . . . .	185
5.5.5	HLLC with pressure relaxation and energy correction . . . . .	187
5.6	The system featuring a governing equation for the interfacial area density . .	187
5.6.1	The Rusanov scheme . . . . .	188
5.7	A set of verification test cases . . . . .	188
5.7.1	A series of Riemann Problems (RPs) for the Euler system . . . . .	189
5.7.1.1	Euler exact Riemann Problem solver . . . . .	189
5.7.1.2	Benchmark tests from Eleuterio F. Toro . . . . .	189
5.7.1.3	Higher order . . . . .	191
5.7.2	A series of Riemann Problems for the Baer-Nunziato (B-N) system . .	198
5.7.2.1	Test 1: retrieving the Euler system results . . . . .	198
5.7.2.2	Test 2: Pure interface advection . . . . .	199
5.7.2.3	Tests 3-7: A set of Riemann Problems from S. A. Tokareva and E. F. Toro . . . . .	199
5.7.3	Hagen-Poiseuille flow for the Navier-Stokes (NS) system . . . . .	208
5.7.4	Convective subsystem and relaxation procedure from Cordesse system	209
5.7.4.1	Pure interface advection . . . . .	210
5.7.4.2	Moderate density water-air shock tube . . . . .	211
5.7.4.3	High-Density water-air shock tube . . . . .	211
5.8	Results for the convective subsystem featuring an interfacial area density equation	215
5.9	Conclusions and perspectives . . . . .	220
<b>6</b>	<b>Software architecture for scientific computing</b>	<b>221</b>
6.1	josiepy: A PDE solver written in Python without sacrificing (too much) per- formance . . . . .	221
6.1.1	Architecture of the code . . . . .	223
6.1.1.1	Physics of the Problem . . . . .	223
6.1.1.2	Mesh Generation . . . . .	227
6.1.1.3	The Numerics . . . . .	230
6.1.1.4	Wrapping things up, the Solver object . . . . .	236
6.1.1.5	I/O Control . . . . .	237
6.1.2	Future plans and perspectives . . . . .	238
6.2	Mercur(v)e: A post-processing library to perform analysis on two-phase flow simulations using differential geometry . . . . .	239
6.2.1	The architecture of the code . . . . .	239
6.2.1.1	The standalone executable . . . . .	241


## *Contents*

6.2.1.2	The Volume and Surface objects . . . . .	242
6.2.1.3	Playing with distance functions to create artificial surfaces .	243
6.2.2	Future plans and perspectives . . . . .	244
<b>Appendices</b>		<b>247</b>
6.A	A “main” file to run an Euler jet simulation with josiepy . . . . .	247
6.B	A “Mickey Mouse” shaped object obtained with the composition of basic level-set objects using <code>Mercur(v)e</code> . . . . .	251
<b>7 General conclusions and future perspectives</b>		<b>253</b>
<b>Acronyms</b>		<b>257</b>
<b>Bibliography</b>		<b>261</b>

# List of Figures

2.1	An opinionated and absolutely not in scale selection of cool “legacy” launch systems . . . . .	18
2.2	An opinionated and absolutely not in scale selection of cool “new generation” launch systems . . . . .	19
2.3	Virgin Galactic SpaceShipTwo Hybrid Rocket Engine (HRE) . . . . .	22
2.4	Liquid Rocket Engine (LRE) injector configuration, courtesy of Lefebvre and McDonell (2017) . . . . .	22
2.5	The multipoint injection feed system from David et al. (2004) . . . . .	23
2.6	LEAP aeronautical engine. Courtesy of Safran Aircraft Engines . . . . .	25
2.7	Two-phase flows are inherently multiscale . . . . .	27
2.8	A cold gas visualization of a paraffin HRE, gas flows from right to left. Courtesy of Rehakavas et al. (2015) . . . . .	28
2.9	Typical range of Weber and Reynolds number in coaxial jet stream from Lasheras and Hopfinger (2000) . . . . .	29
3.1	Evolution of the configuration of a system from a reference configuration . . . . .	52
3.2	At lower scales, the interface cannot be resolved and it needs modeling . . . . .	66
3.3	Deformation of a droplet normal to the surface . . . . .	68
3.B.1	Spherical and polar coordinate system for the upper half deformed droplet. . . . .	93
3.B.2	Numerical approximation of the ellipsoid in the $(O, \mathbf{e}_r, \mathbf{e}_z)$ plane for different aspect ratios. . . . .	96
4.1	A central slice at $\tau = 0$ of the volumetric level-set field on a uniform grid. The black lines are the 0-isoline of the level-set function . . . . .	111
4.2	The 1-ring of a point $\mathbf{x}_i$ . . . . .	111
4.3	$\mathring{A}$ for a general case where one of the triangles is obtuse. The light gray area is the 1-ring neighborhood and the dark gray area is actually $\mathring{A}$ . . . . .	112
4.4	Computing curvatures on a couple of spherical objects . . . . .	115
4.5	An example of meshes used to test the convergence on canonical objects . . . . .	116
4.6	Convergence rate of the curvature computation . . . . .	117
4.7	Error map at two different resolutions for a sphere . . . . .	117
4.8	Computing curvatures with and without applying the averaging kernel . . . . .	118
4.9	The collision sequence of two droplets. The top row shows the computation without average. The bottom row is with an averaging radius $R_{dx} = 5 dx$ . . . . .	123
4.10	The Number Density Function (NDF) computed with three different resolutions at the final snapshot . . . . .	124
4.11	The collision sequence of two droplets in terms of the NDF . . . . .	125

4.12	Statistics on the geometrical properties computed on the collision of two droplets	126
4.13	The initial configuration of the ellipsoidal droplet. . . . .	130
4.14	One period $T$ of evolution of the deformed droplet. The color map  maps values of the Gauss curvature from low to high. . . . .	130
4.15	Relative volume ratio conservation. . . . .	131
4.16	Evolution of the geometrical properties ( $p/q - 1 = 0.05$ ) . . . . .	132
4.17	Evolution of the geometrical properties ( $p/q - 1 = 0.15$ ) . . . . .	134
4.18	Evolution of the geometrical properties ( $p/q - 1 = 0.3$ ) . . . . .	135
4.19	Evolution of the geometrical properties ( $p/q - 1 = 0.5$ ) . . . . .	136
4.20	Injecting Mercury predictions at runtime during a DNS . . . . .	138
5.1	Curvilinear coordinates comparison in logical/programming and physical space	147
5.2	A mesh generated by josiépy integrated mesh generator . . . . .	149
5.3	Schematic of a mesh cell with its neighbor . . . . .	150
5.4	Example of characteristics curves for a “small hat” problem for the Burgers equation, courtesy of Massot, Series, et al. (2020) . . . . .	152
5.5	A general Riemann problem wave pattern . . . . .	156
5.6	The Godunov method applied to a mesh . . . . .	157
5.7	Slopes reconstruction in structured meshes . . . . .	158
5.8	Estimating the gradient with a least square approach . . . . .	163
5.9	The maximum CFL number allowed by the propagation of the information in hyperbolic systems . . . . .	167
5.11	The Riemann Problem for the Euler system . . . . .	173
5.12	Rusanov scheme wave structure . . . . .	176
5.13	HLL scheme wave structure . . . . .	177
5.14	HLLC scheme wave structure . . . . .	177
5.15	Density field comparison of first and second order Rusanov schemes for an Euler jet at time $t = 0.03s$ . . . . .	191
5.16	Test 1 at $t = 0.25s$ of Eleuterio F. Toro (2009) . . . . .	193
5.17	Test 2 at $t = 0.15s$ of Eleuterio F. Toro (2009) . . . . .	194
5.18	Test 3 at $t = 0.012s$ of Eleuterio F. Toro (2009) . . . . .	195
5.19	Test 4 at $t = 0.035s$ of Eleuterio F. Toro (2009) . . . . .	196
5.20	Test 5 at $t = 0.035s$ of Eleuterio F. Toro (2009) . . . . .	197
5.21	A test with constant $\alpha$ to retrieve same results as Fig. 5.16, $t = 0.25s$ . . . . .	201
5.22	An artificial test in which the convective solver is disabled and only the non-conservative term is approximated, <i>i.e.</i> an advection problem . . . . .	202
5.23	Test 1 from S. A. Tokareva and E. F. Toro (2010) . . . . .	203
5.24	Test 2 from S. A. Tokareva and E. F. Toro (2010) . . . . .	204
5.25	Test 3 from S. A. Tokareva and E. F. Toro (2010) . . . . .	205
5.26	Test 4 from S. A. Tokareva and E. F. Toro (2010) . . . . .	206
5.27	Test 5 from S. A. Tokareva and E. F. Toro (2010) . . . . .	207
5.28	Poiseuille Flow . . . . .	209
5.29	Advection test at instant $t = 229\mu s$ from Cordesse (2020) . . . . .	211
5.30	Shock tube at moderate density ratio at instant $t = 900\mu s$ from Cordesse (2020)	212

5.31	Shock tube at high density ratio at instant $t = 240 \mu\text{s}$ from Cordesse (2020) . .	214
5.32	Initial condition for the Kelvin-Helmoltz case . . . . .	217
5.33	State of the Kelvin-Helmoltz case for $t = 2.99 \text{ s}$ . Top row with the fist-order scheme on a $1500 \times 1500$ mesh. Bottom one with MUSCL-Hancock MINMOD scheme on a $500 \times 500$ mesh . . . . .	218
5.34	State of the Kelvin-Helmoltz case for $t = 4.99 \text{ s}$ . Top row with the fist-order scheme on a $1500 \times 1500$ mesh. Bottom one with MUSCL-Hancock MINMOD scheme on a $500 \times 500$ mesh . . . . .	219
6.1	The architecture of the josiepy solver . . . . .	224
6.2	The plotted boundary of the domain . . . . .	228
6.3	The mesh data structure including the ghost cells . . . . .	231
6.4	The flow of operations driven by the Solver object . . . . .	232
6.5	Each solver can have its own specific implementation of objects or share the general ones . . . . .	234
6.6	Mercur(v)e architecture . . . . .	240
6.7	A “Mickey Mouse” object made by the composition of level-set spheres. The  maps the Gauss curvature value . . . . .	244



# List of Tables

5.1	Left and right initial data for the Euler test cases . . . . .	189
5.2	Left and right initial data for the Baer-Nunziato (B-N) test 1 . . . . .	198
5.3	Left and right initial data for the B-N test 2 . . . . .	199
5.4	Left and right initial data for the B-N test from S. A. Tokareva and E. F. Toro, phase $k = 1$ . . . . .	199
5.5	Left and right initial data for the B-N test 3–8, phase $k = 2$ . . . . .	199
5.6	Pure interface advection ( $\epsilon = 1\text{e-}8, \rho_1 = 10 \text{ kg m}^{-3}, \rho_2 = 1000 \text{ kg m}^{-3}$ ) . . .	210
5.7	Water-air shock tube with moderate density ratio ( $\epsilon = 1\text{e-}8, \rho_1 = 50 \text{ kg m}^{-3}, \rho_2 =$ $1000 \text{ kg m}^{-3}$ ) . . . . .	211
5.8	Water-air shock tube with high density ratio ( $\epsilon = 1\text{e-}8, \rho_1 = 1 \text{ kg m}^{-3}, \rho_2 =$ $1000 \text{ kg m}^{-3}$ ) . . . . .	212
5.9	Initial conditions for the system with the interfacial area density equation, $P =$ $0.71$ . . . . .	215
5.10	Barotropic Equation of State (EoS) coefficients . . . . .	216





# List of Algorithms

- 4.1 Algorithm to compute the mixed area . . . . . 113
- 4.A.1 Algorithm to merge a multi-block Direct Numerical Simulation (DNS) simulation . . . . . 142
- 5.1 Algorithm to solve the Riemann Problem for the Euler System . . . . . 174



# Sommario

Negli ultimi anni stiamo assistendo a una “seconda corsa allo spazio”: aziende private come SpaceX stanno aprendo la strada a una nuova generazione di sistemi di lanciatori spaziali ottimizzati per essere efficienti nei costi e con prestazioni estreme che ragionevolmente porteranno l'umanità su Marte per la prima volta nella sua esistenza nel prossimo futuro. Un aspetto chiave di questi sistemi è quello di fornire un alto livello di riutilizzabilità associato ad un drastico calo dei costi di lancio. Questo si traduce in sistemi di propulsione che devono operare su involucri di volo più ampi, con coppie di propellenti più vantaggiose come il metano e l'ossigeno criogenici, richiedendo quindi progetti più vincolati e complessi per i sistemi di iniezione. Gli iniettori sono responsabili della corretta nebulizzazione di carburante e ossidante e hanno un impatto diretto sulle prestazioni dei motori.

L'attuale stato dell'arte delle strategie di modellazione non riesce a prevedere le corrette distribuzioni per gli spray nella camera di combustione, quindi, l'obiettivo di questa tesi è quello di offrire un quadro di modellazione unificato che affronti la derivazione del sistema di equazioni che governano i sistemi di flusso bifase caratterizzata da una solida struttura matematica attraverso un'impostazione variazionale chiamata “Principio dell'Azione Stazionaria” (SAP). Questo sforzo è sostenuto da un set di strumenti di calcolo su misura che permette la scelta razionale delle ipotesi di modellazione e la simulazione efficace dei modelli sviluppati, possibilmente su architetture di calcolo moderne.

In questo lavoro si identificano tre punti principali di miglioramento: lo sviluppo di modelli di ordine ridotto attraverso il SAP, con un insieme di equazioni che includono proprietà geometriche come la densità della superficie d'interfaccia e le curvature medie e di Gauss; l'implementazione di uno strumento di post-processing geometrica delle simulazioni che viene utilizzato per raccogliere informazioni utili da simulazioni ad alta fedeltà al fine di creare un accurato modello di ordine ridotto; lo sviluppo di una libreria Python che funge da banco di test per valutare rapidamente le nuove idee nell'ambito di schemi numerici, condizioni al contorno, configurazioni di dominio, con la potenziale capacità di sfruttare architetture di calcolo moderne come le GPU.

Keywords: flussi bifase, modellazione di sottoscala, densità di superficie, curvature, DNS, geometria computazionale, calcolo scientifico



# Résumé

Nous assistons actuellement à une “deuxième course à l'espace” : des entreprises privées comme SpaceX ouvrent la voie à une nouvelle génération de systèmes de lanceurs spatiaux optimisés pour leur rentabilité et leurs performances extrêmes, qui permettront à l'humanité d'atteindre Mars pour la première fois dans son existence. L'un des aspects essentiels de ces systèmes est d'offrir un niveau élevé de réutilisation, ce qui entraîne une baisse drastique des coûts de lancement. Cela se traduit par des systèmes de propulsion qui doivent fonctionner dans des enveloppes de vol plus larges, avec des paires d'ergols plus avantageuses comme le méthane et l'oxygène liquides, ce qui exige une conception plus rigoureuse des systèmes d'injection. Les injecteurs sont responsables de la nébulisation correcte des ergols et ils ont un impact direct sur les performances des moteurs.

Les stratégies de modélisation actuelles ne parviennent pas à prédire les distributions correctes de gouttelettes dans la chambre de combustion. L'objectif de cette thèse est donc d'offrir un cadre de modélisation unifié permettant la dérivation de systèmes d'équations pour les écoulements diphasique, caractérisé par une structure mathématique solide obtenue avec un principe variationnel appelé “Principe d'Action stationnaire” (SAP). Cet effort est soutenu par un ensemble d'outils informatiques adaptés qui permettent le choix rationnel des hypothèses de modélisation et la simulation efficace des modèles développés, éventuellement sur des architectures modernes.

Ce travail identifie trois points principaux d'amélioration : le développement de modèles d'ordre réduit avec le SAP, comportant un ensemble d'équations qui incluent des propriétés géométriques telles que la densité de la surface interfaciale et les courbures moyenne et de Gauss ; la mise en œuvre d'un outil de post-traitement géométrique pour les simulations à haute-fidélité utilisé pour recueillir des informations utiles afin d'élaborer un modèle d'ordre réduit précis, et le développement d'une bibliothèque Python qui agit comme un outil de prototypage rapide visant à tester rapidement des idées dans le contexte des schémas numériques, des conditions limites, des configurations de domaine, avec la possibilité d'exploiter des architectures de calcul modernes comme les GPU.

Mots-clés : écoulements diphasiques, modélisation de sous-échelle, densité surfacique d'interface, courbures, DNS, géométrie computationnelle, calcul scientifique



# Abstract

In current times we are witnessing a “second space race”: private companies like SpaceX are paving the way to a new generation of space launcher systems optimized for cost effectiveness and extreme performances that will bring humankind to Mars for the first time in its existence. A key aspect of those systems is to provide a high level of reusability leading to a drastic drop in launch costs. This translates into propulsion systems that need to operate on wider flight envelopes, with more advantageous propellant pairs like cryogenic methane and liquid oxygen, therefore requiring tighter designs for the injection systems. The injectors are responsible for the correct nebulization of fuel and oxidizer and they have a direct impact on the performance of the engines. This kind of problems are shared across different applications and are somehow generic.

The current state of the art modeling strategies fail at predicting the correct distributions of droplets in the combustion chamber. Therefore, the target of this thesis is to contribute to the design of a unified modeling framework addressing the derivation of system of equations governing two-phase flow systems characterized by a sound mathematical structure via a variational approach named Stationary Action Principle (SAP) coupled to the second principle of thermodynamics. This effort is backed by a tailored computational toolset that allows the rational choice of modeling assumptions and the effective simulations of the developed models, possibly on modern computing architectures.

This work identifies three main points of improvement: the development of reduced-order models via a variational procedure named the Stationary Action Principle (SAP) featuring a set of equations that include geometrical properties such as the interfacial surface density and the mean and Gauss curvatures; the implementation of a geometric DNS post-processing tool that is used to collect useful insight from high-fidelity simulations in order to craft an accurate reduced-order model, and the development of a Python library that acts as a prototyping playbook aimed at quickly testing ideas in the context of numerical schemes, boundary conditions, domain configurations, with the potential ability of leveraging modern computational architectures such as GPUs.

Keywords: two-phase, subscale modeling, interface area density, curvatures, DNS, computational geometry, scientific computing





# Notes on notation

The notation employed in this thesis for mathematics typesetting is tailored towards lightness and immediateness. In some contexts, we perform a choice of trading some unambiguousness for better clarity with the final objective of delivering a smooth and pleasant reading. For this reason, we summarize here below the mathematics typesetting definitions we decided to use in the following of this thesis.

- Generic placeholders  $\bullet$ ,  $\odot$  are used to express the action of operators in a generic fashion.  
*e.g.* the averaging operator  $\langle \bullet \rangle$
- The greek letter  $\varphi$  is used to specify a generic field.
- Functionals are expressed with a script formatting like  $\mathcal{A}$  or  $\mathcal{L}$
- Vectors are typeset with lowercase bold letters  $\mathbf{x}$ ,  $\mathbf{q}$
- Tensors are typeset with uppercase bold letters  $\mathbf{B}$ ,  $\mathbf{K}$
- Borrowing the idea from [Luigi Quartapelle and Auteri \(2013\)](#), the differential is omitted in integrals where the domain of integration is unambiguously clear from the integral subscript  $\int_{\Omega} \bullet$



# 1 | Introduction

## 1.1 Contexte industriel

---

### 1.1.1 Systèmes d'injection pour la propulsion spatiale

Le 16 juin 1969, Neil Armstrong, Buzz Aldrin et Michael Collins quittent la planète mère pour la *Mare Tranquillitatis* au sommet du plus grand, du plus lourd et du plus puissant lanceur spatial que l'espèce humaine ait jamais construit à cette date : la Saturn V. La Saturn V illustrée dans Fig. 2.1a était une fusée gargantuesque de 110.6 m de haut et de 10.1 m de diamètre, propulsée par 5 moteurs à propergol liquide F-1, chacun capable de fournir 7770 kN de poussée dans le vide grâce au cycle générateur de gaz avec oxygène et hydrogène liquides qu'ils emploient. Le 20 juin 1969, les trois astronautes se posent sur la Lune à bord du *Apollo Lunar Excursion Module (LEM)*, après s'être détachés du *Apollo command and service module (CSM)*, le module chargé de la manœuvre en orbite lunaire, propulsé par un moteur à propergol liquide hypergolique acide nitrique et méthylhydrazine. Un pas de géant qui élargit les horizons d'exploration de l'humanité, en plaçant les nouvelles "Colonnes d'Hercule" hors de l'atmosphère terrestre et au-delà. En effet, *"la Terre est le berceau de l'humanité, mais on ne peut pas y vivre éternellement"*. (K. E. Tsiolkovsky).

La "Première course à l'espace" a ouvert la voie à une innovation incroyablement rapide dans le domaine de la propulsion dont le rayonnement et l'intérêt de la recherche sont encore tangibles aujourd'hui, dans la "Seconde course à l'espace", avec l'avènement de la privatisation du secteur spatial, la révolution de la réutilisabilité de SpaceX, et les plans d'exploration et de colonisation de Mars. L'extrême motivation à l'époque des missions Apollo conduit au développement de différentes technologies de propulsion spatiale :

- Moteurs à propergol liquide : moteurs spatiaux qui présentent à la fois l'oxydant et le combustible à l'état liquide. Les propergols sont stockés dans des réservoirs à partir desquels ils sont pompés jusqu'à la tête de l'injecteur par un système pressurisé (technologie de soufflage), ou par une sorte de machinerie de pompage avec des degrés de complexité

très différents : pompes électriques (fusée Electron de Rocket Lab), turbopompes à cycle expasseur (Ariane 5, Atlas V, Delta IV, New Glenn, Longue Marche 5, KVTk, H-I, H-II), turbopompes à cycle générateur de gaz (moteurs F1, Falcon 9), turbopompes à cycle de combustion étagée (Space Shuttle, Energia, New Glenn, Starship, N1, Proton, H-II, Angara). La conception minutieuse de la plaque de l'injecteur est d'un grand intérêt pour éviter l'instabilité de la combustion et les accidents catastrophiques associés (*e.g.* Apollo 13 (IRVINE 2008)). Cette technologie est la plus efficace dans le cadre de la propulsion thermique et elle est la plus utilisée dans les systèmes classiques (Fig. 2.1) et de nouvelle génération (Fig. 2.2). Une subdivision supplémentaire peut être effectuée en fonction de la nature des propergols qui alimentent les moteurs :

- Propergols hypergoliques : Propergols qui n'ont pas besoin d'une source d'énergie initiale pour s'enflammer, le simple mélange des composants crée les conditions de l'inflammation. Les exemples notables d'oxydants hypergoliques sont : tétrazole, acide nitrique, peroxyde d'hydrogène. Comme carburant, les dérivés de l'hydrazine sont le plus souvent utilisés.
- Propergols cryogéniques : propergols à impulsion spécifique élevée dont le point d'ébullition est généralement bas et qui doivent être refroidis pour augmenter la densité et réduire le volume des réservoirs. Ces propergols alimentent les moteurs les plus puissants (l'impulsion spécifique la plus élevée a été atteinte avec la navette "Space Shuttle", Fig. 2.1b) et sont le plus souvent : l'hydrogène et plus récemment le méthane comme combustibles, l'oxygène liquide comme oxydant. Parfois, le kérosène (RP1) est utilisé comme carburant en combinaison avec de l'oxygène cryogénique. Les systèmes modernes, comme le moteur Raptor de SpaceX, le Blue Origin BE-4, le moteur Avio M10, conduisent à l'emploi de l'oxygène cryogénique et du méthane comme propergols pour les moteurs.
- Moteurs à propergol solide : moteurs spatiaux dont l'oxydant et le combustible sont à l'état d'aggrégats solide. Les propergols sont généralement coulés dans des chemises composites et ont une longue durée de stockage. Ils sont généralement utilisés comme moteurs de soutien à forte poussée/poids (*i.e.* "boosters") dans les premiers étages d'un lancement spatial. Par rapport aux moteurs à propergol liquide, les moteurs à poudre ont moins de pièces mobiles et sont en général "simples", ils peuvent être produits en masse et facilement stockés même s'ils sont moins efficaces thermochimiquement parlant. Les moteurs à propergol solide les plus célèbres sont les boosters du Space Shuttle (Fig. 2.1b), mais ils sont encore largement utilisés aujourd'hui : la fusée européenne

Ariane 5, la nouvelle génération d'Ariane 6, l'italienne Vega sont équipées de moteurs à poudre. Les propergols solides sont également utilisés dans l'industrie de la défense comme systèmes de létalité à bord des avions de chasse. Pour se limiter aux applications de propulsion spatiale, ils sont généralement constitués d'une matrice plastique liante, très souvent du HTPB, dans laquelle des métaux de taille micro ou nanométrique (souvent de l'aluminium) jouent le rôle de combustible, et du nitrate ou du perchlorate d'ammonium celui d'oxydant. Nous renvoyons le lecteur aux travaux de [DOISNEAU, DUPAYS et al. \(2011\)](#); [DOISNEAU, SIBRA et al. \(2014\)](#); [DUPIF \(2018\)](#); [FRANÇOIS et al. \(2020\)](#); [FRANÇOIS et al. \(2020\)](#) sur la modélisation et la simulation des phénomènes dans les moteurs à propergol solide.

- Moteurs hybrides : technologie moins développée pour les moteurs spatiaux pour laquelle un des deux ergols est à l'état liquide, souvent le comburant, et l'autre à l'état solide. Les moteurs hybrides présentent des avantages théoriques par rapport aux moteurs à propergol liquide et solide pour les activités spatiales commerciales car ils peuvent atteindre des performances similaires à celles du RP1-oxygène liquide avec un seul système de turbopompe au lieu de deux lorsqu'on utilise de la paraffine comme combustible, avec des avantages évidents en termes de coûts. L'une des raisons pour lesquelles ils n'ont pas trouvé d'usage industriel est que la combustion des moteurs hybrides est régulée par la diffusion (alors que pour un moteur à propergol liquide ou un moteur à propergol solide, la combustion est généralement régie par la cinétique chimique), ce qui entraîne des problèmes majeurs de flexibilité, de performance, de mise à l'échelle et de précision de la simulation. Des tentatives d'utilisation comme solution viable ont été réalisées par l'AMROC dans le passé, puis par Virgin Galactic qui a acquis la propriété intellectuelle avec son avion spatial SpaceShipTwo Fig. 2.3. Un regain d'intérêt s'est manifesté récemment à la suite de résultats intéressants obtenus en utilisant de la cire et de la paraffine comme combustibles ([KARABEYOGLU et al. 2004](#)) : des tentatives de construction de nano-lanceurs alimentés en HRE ont été lancées par Leaf Space en Italie (*Primo | A Small Lightweight Nanolauncher for Your Small Sat* 2018), Hybrid Propulsion for Space en France (*Hybrid Propulsion for Space* 2021), entre autres. Outre la paraffine, d'autres combustibles potentiels utilisés dans la nature sont le HTPB ou le polycarbonate. L'oxydant est généralement l'oxygène, mais le protoxyde d'azote est également utilisé, notamment pour les systèmes de faible niveau, comme par exemple les fusées fabriquées par des étudiants (voir Skyward Experimental Rocketry en Italie, DARE aux Pays-Bas, HyImpulse en Allemagne, entre autres).

En limitant la discussion aux configurations d'écoulement diphasique dans lesquelles les phases n'apparaissent pas à l'état solide, comme c'est le cas pour les moteurs à propergol solide, la modélisation et la simulation de l'écoulement diphasique jouent un rôle majeur pour la conception rentable et sûre d'un moteur. Comme nous l'avons déjà évoqué, la conception soignée des chicanes du réservoir, de la plaque d'injection et de la chambre de combustion est vitale pour éviter l'apparition d'oscillations et d'instabilités de pression soutenues qui peuvent facilement provoquer un "désassemblage rapide non prévu". De plus, l'optimisation d'un moteur hybride nécessite des prédictions précises du champ d'écoulement et des produits de combustion, notamment lors de l'utilisation de paraffine comme carburant, qui fond en créant un jet de gouttelettes de carburant dont l'effet est très similaire à celui d'un moteur diesel.

### 1.1.2 Système d'injection pour la propulsion aérobie

La modélisation, la simulation et l'optimisation de l'injection ne sont pas des thèmes limités au contexte de la propulsion spatiale. Ces dernières années, on a assisté à un changement substantiel de la stratégie politique concernant le changement climatique. En particulier, plus souvent qu'auparavant, des politiques plus strictes sont appliquées aux émissions produites par la production d'énergie des industries et des utilisateurs, ainsi que par les déplacements en voiture et en avion. À titre d'exemple, le gouvernement français prévoit d'interdire les vols intérieurs afin de réduire les émissions polluantes ("[France Moves to Ban Short-Haul Domestic Flights](#)" 2021). La revue faite dans [LEE et al. \(2001\)](#), même si elle n'est pas complètement à jour, fournit une analyse historique de l'évolution des émissions des avions. Elle fait état d'une multiplication possible par trois à sept des émissions de CO<sub>2</sub> d'ici 2050. Afin de se conformer aux réglementations toujours plus strictes, la technologie de propulsion doit évoluer pour offrir un meilleur rendement. Pour les applications aéronautiques, des taux de dilution plus élevés et des températures de combustion basses sont parmi les aspects clés qui permettent d'améliorer l'efficacité de la combustion et de réduire les polluants et les gaz à effet de serre, et l'injection de carburant dans la chambre de combustion a un impact majeur sur ces paramètres. Alors que pour les moteurs à propergol liquide de simples atomiseurs à orifice unique sont utilisés, parfois configurés comme des jets d'impact où le jet de carburant heurte l'oxydant à un certain angle comme illustré dans la Fig. 2.4, pour la propulsion aérospatiale à air comprimé, des configurations plus complexes sont nécessaires. À titre d'exemple, la Fig. 2.5 montre les détails techniques d'un injecteur coaxial multipoint décrit dans le brevet de [DAVID et al. \(2004\)](#), et il décrit une configuration où non seulement une alimentation primaire en carburant est présente, mais aussi plusieurs alimentations secondaires (notées **62** et **72** dans Fig. 2.5) réparties le long de l'axe central (noté axe X—X dans Fig. 2.5). Des moteurs comme le LEAP de Safran Aircraft

Engines, illustré dans le Fig. 2.6, sont le résultat de cette optimisation. Dans ces conditions extrêmes et ces enveloppes de vol serrées, la maîtrise du processus d'injection est cruciale et de nombreuses recherches sont consacrées à cette tâche (*e.g.* REMIGI (2021) tout récemment). Les différents points d'injection présents dans un moteur ont pour objectifs combinés de contrôler les flux thermiques sur les parois pour éviter les situations critiques, d'améliorer la durée de vie de ces composants coûteux en augmentant le **Mean time between failures (MTBF)** mais aussi de contrôler la température de la flamme pour éviter une production excessive de  $NO_x$  et autres polluants. La nébulisation correcte du carburant dans les chambres de combustion est donc vitale, et en raison des difficultés d'accès et d'observation inhérentes aux moteurs aéronautiques, la modélisation et la simulation sont des outils importants dans la phase initiale du cycle de conception, mais aussi plus tard lors de la mise en service, du contrôle et de la mise hors service de ces systèmes. Les simulations actuelles à l'état de l'art traitent le gaz porteur avec une approche eulérienne où les gouttelettes de carburant sont ensuite injectées individuellement ou comme des particules virtuelles composées d'un groupe d'objets physiques individuels et suivies avec une approche lagrangienne : (LEBAS et al. 2005 ; SANJOSÉ et al. 2011 ; VIGNAT et al. 2021). Cette approche peut s'avérer non prédictive lorsque la configuration devient très complexe (comme les systèmes d'injection multipoints de la Fig. 2.5) et lorsque les conditions de fonctionnement sont éloignées des conditions expérimentales, de sorte que la calibration des paramètres du modèle lagrangien ne parvient pas à représenter la physique sous-jacente de l'atomisation primaire. Cette situation est également rencontrée pour les systèmes d'injection directe automobiles. Il y a eu quelques tentatives pour développer un formalisme général pour la déstabilisation et la rupture de l'interface liquide en utilisant la représentation de la densité de surface interfaciale en commençant par le travail de VALLET et BORGHİ (1999) poursuivi avec une dérivation empirique pour les flux de phases séparées basée sur l'analyse de l'instabilité de Kelvin-Helmholtz dans JAY et al. (2006) et plus récemment une représentation avec deux densités de surface interfaciale pour les phases séparées et dispersées DEVASSY (2014) ; DEVASSY et al. (2015). Jusqu'à présent aucune de ces approches n'est vraiment capable de représenter les liens entre la physique à l'intérieur de l'injecteur et la région du spray développé à toutes les échelles et avec des dimensions réelles et des caractéristiques géométriques complexes. Par conséquent, la résolution correcte du film liquide à proximité de l'injecteur est primordiale pour améliorer les résultats de la simulation. Pour cette raison, un modèle d'ordre réduit approprié qui inclut le traitement des petites échelles significatives est nécessaire, en particulier pour la représentation de la densité de la surface interfaciale.

### 1.1.3 Système d'injection pour les moteurs injection directe

Le thème de l'environnement est aussi important dans le développement de la nouvelle génération de moteurs à pistons pour le transport terrestre, comme les voitures et les camions. Dans la plupart des pays occidentaux, les gouvernements ont prévu une élimination progressive des moteurs diesel afin de réduire la pollution. En Norvège, cette élimination progressive est prévue dès 2025 (*Phase-out of Fossil Fuel Vehicles 2021*). Par conséquent, dans un avenir prévisible, la majeure partie du financement de la R&D sera destinée à améliorer l'efficacité de la combustion des moteurs à injection directe d'essence. Dans cette technologie, le mélange d'air et de carburant doit être soigneusement régulé pour éviter l'auto-allumage et la détonation, et également pour maintenir la température à un niveau raisonnable afin d'éviter la production de  $NO_x$  (qui sont de toute façon également capturés en aval du système de moteur thermique avec un catalyseur pour se conformer aux lois anti-pollution locales strictes). Les conditions extrêmes qui sont caractéristiques de ces systèmes et aussi le fait que les chambres de combustion changent de volume dans le temps en raison du mouvement du piston, rendent l'étude du phénomène d'injection pour les moteurs alternatifs à injection directe très difficile du point de vue de la modélisation, mais aussi des aspects numériques et géométriques.

## 1.2 Contexte scientifique

L'injection est un phénomène multiéchelle par nature. La Fig. 2.7 montre le schéma d'un scénario typique où un système d'injection injecte du carburant dans une chambre de combustion dans laquelle se trouve un gaz, généralement de l'air. Dans la partie gauche de l'image, nous avons le carburant qui sort de la tête de l'injecteur. Cette partie du domaine est caractérisée par une phase liquide clairement séparée par une interface de l'atmosphère gazeuse environnante et elle est appelée zone ou régime à *phase séparée*. A l'extrémité de la chambre, l'interface s'est déformée à un niveau tel qu'un grand nombre de gouttelettes se détachent en formant un spray, c'est le régime à *phase dispersée*. Bien que les deux différents régimes soient clairement identifiés et modélisés, ils doivent encore être liés d'une manière ou d'une autre, c'est-à-dire qu'une zone supplémentaire dans laquelle les grandes et petites échelles coexistent et où une distinction claire n'est pas possible peut également être identifiée, c'est la *région mixte*. Les échelles de longueur de référence associées au diamètre de l'orifice des atomiseurs sont de l'ordre de 1e–3 m. L'interface subit ensuite une déstabilisation associée aux gradients de vitesse (instabilités de Kelvin-Helmoltz), aux effets de tension de surface (instabilité de Plateau-Rayleigh), à l'interaction avec la turbulence et à un mélange complexe de tous ces facteurs. L'interface



se déforme jusqu'à créer des ligaments. Plus en aval, ces ligaments se détachent du cœur du jet, conduisant à la **rupture (breakup) primaire**. Dans la partie la plus en aval du domaine représenté, les ligaments originaux détachés se décomposent en plus petites gouttelettes, jusqu'à la création de gouttes presque sphériques de l'ordre de  $1e-4$  à  $1e-6m$ ; c'est la **rupture (breakup) secondaire**, conduisant à la création du régime à phase dispersée. Les nombres non dimensionnels les plus importants qui régissent le phénomène sont les nombres de Reynolds ( $Re$ ), de Weber ( $We$ ) et d'Ohnesorge ( $Oh$ ) :

$$Re_k = \frac{\rho_k u_k L_k}{\mu_k} \quad (1.1)$$

$$We = \frac{\rho_g (u_g - u_l)^2 L_l}{\sigma_l} \quad (1.2)$$

$$Oh_k = \frac{\sqrt{We}}{Re_k} \quad (1.3)$$

où l'indice  $k = l, g$  indique une phase générique,  $l$  indique la phase liquide et  $g$  la phase gazeuse.  $L_k$  est une longueur de référence spécifique à la phase,  $\rho_k$  est la densité d'une phase et  $u_k$  sa vitesse.  $\sigma_k$  est le coefficient de tension de surface. La Fig. 2.9 montre la plage des quantités non dimensionnelles pour quelques cas de jets ronds coaxiaux. Une discussion détaillée des différents types d'atomiseurs pour les systèmes d'injection peut être trouvée dans LEBEVRE et McDONELL (2017).

### 1.2.1 Différents modèles pour différents régimes

Le problème de l'injection étant par nature multi-échelle, différentes approches ont été développées pour faire face à la complexité et être capable de simuler de manière prédictive même les systèmes les plus élaborés. Dans certaines situations simplifiées, des approches haute fidélité peuvent être employées pour traiter l'ensemble du spectre des échelles et des régimes, alors que dans la plupart des situations, des simplifications sont nécessaires car certaines échelles du problème ne peuvent être résolues. Des modèles d'ordre réduit sont donc nécessaires : le plus souvent, deux modèles différents sont utilisés, l'un pour le régime de phases séparées, dans lequel les champs sont généralement régis par des équations de conservation sous forme eulérienne, et l'autre où les objets dispersés individuels sont suivis de manière lagrangienne ou via leur fonction de densité en nombre, reconstruite à partir d'une sélection de ses moments. D'autres approches sont plutôt conçues pour fournir un contexte de modélisation unifié permettant de traiter les deux régimes d'écoulement en même temps.

### 1.2.2 Modèles pour résoudre toutes les échelles

Si l'on souhaite être en mesure de simuler les échelles les plus fines du problème, deux grandes approches sont possibles. La première est basée sur les travaux de [CAHN et HILLIARD \(1958\)](#) et de ses dérivés successifs (*e.g.* [C. LIU et SHEN 2003](#)), appelés “phase field” models. Le travail de CAHN et HILLIARD considère l'interface entre les deux fluides avec une épaisseur finie. Bien que cette approche soit parfaitement raisonnable dans les situations où l'épaisseur de l'interface n'est pas trop fine, dans un scénario plus général, l'épaisseur de l'interface peut être de l'ordre de quelques angströms, ce qui nécessite un maillage très fin pour être résolu avec satisfaction. En alternative, la résolution des équations [Navier-Stokes \(NS\)](#) pour chaque phase ainsi que des équations de saut à l'interface, qui est donc considérée comme une surface de discontinuité, avec un maillage très fin est également possible. Le fait que l'interface ne soit pas résolue comme dans le [CAHN et HILLIARD \(1958\)](#) est déjà une hypothèse de modélisation. Dans le scénario à phase unique, les Direct Numerical Simulations (DNSs) sont définis comme les simulations qui résolvent les équations [NS](#) jusqu'à la plus petite échelle connue qui est l'échelle de Kolmogorov, sans modélisation réduite supplémentaire d'aucune sorte. Cette approche n'est pas directement transposable aux écoulements diphasiques car ceux-ci ont une interface qui est souvent considérée comme infiniment mince, et il est difficile d'identifier une échelle de longueur “Kolmogorov”; par conséquent, les résultats de ces simulations dépendent souvent fortement de la résolution du maillage et des schémas numériques employés ([LING, FUSTER et al. 2017](#)), ainsi que des stratégies de suivi de l'interface. Cependant, pour simplifier le vocabulaire, nous utiliserons le terme [DNS](#) pour définir les modèles qui ne supposent pas de modélisation supplémentaire pour les phénomènes à petite échelle, liés à la dynamique de l'interface (en dehors des différentes techniques de suivi de l'interface employées) et à la modélisation de la turbulence, même dans le contexte d'un écoulement diphasique. L'interface peut être suivie explicitement avec un suivi lagrangien sur une grille qui se déplace avec le fluide, comme dans [TRYGGVASON et al. \(2001\)](#); [James GLIMM et al. \(2006\)](#). Une approche complètement eulérienne peut également être utilisée, où l'interface est capturée à l'aide d'une fonction de couleur advectée (méthode [Volume Of Fluid \(VOF\)](#) [HIRT et B. NICHOLS 1981](#)), ou une fonction de distance ([SETHIAN 1996](#); [RUSSO et SMEREKA 2000](#); [ZHAO 2004](#)), ou même les deux de manière couplée ([FEDKIW et al. 1999](#); [C. LIU et SHEN 2003](#); [VAUDOR et al. 2017](#)). Des techniques compressibles sont également possibles ([B. DURET et al. 2018](#)). [MIRJALILI et al. \(2019\)](#) compare les approches [VOF](#) et Level-Set. Un exemple de simulation haute-fidélité d'un écoulement diphasique compressible pour une injection supercritique est présenté dans [PETIT et al. \(2013\)](#), tandis que dans [ZOU et al. \(2019\)](#), une méthode level-set pour les écoulements compressibles Low-Mach est discutée.

### 1.2.3 Modèles d'ordre réduit

Lorsque l'on traite des modèles d'ordre réduit, on souhaite pouvoir approcher à la fois les régimes à phase séparée et à phase dispersée dans un système d'injection typique. Deux stratégies principales sont disponibles : employer différents sous-modèles pour chaque régime différent, puis les coupler souvent avec des corrélations empiriques ou semi-empiriques ou introduire un cadre unifié capable de faire face aux deux situations.

Le régime à phase séparée dans lequel les deux phases sont clairement séparées par une interface. La manière classique de dériver un ensemble d'équations gouvernantes est de faire la moyenne des équations phasiques instantanées locales à la [ISHII et MISHIMA \(1984\)](#). [ISHII, KIM et al. \(2002\)](#); [RUSCHE \(2003\)](#); [DREW et PASSMAN \(2006\)](#) exploitent cette méthode. Dans ces modèles, l'interface n'est souvent pas reconstruite, c'est un champ transporté qui diffuse autour de la position réelle qu'aurait l'interface dans la situation physique. Le degré de diffusion est contrôlé par la résolution du maillage et l'ordre du schéma numérique employé et ils n'y a généralement pas de traitement spécifique pour les petites échelles interfaciales. Le processus de moyennage génère des termes non fermés qui nécessitent une analyse plus approfondie pour être fermés. Des termes sources spécifiques ("termes de contraction") peuvent être utilisés pour forcer une résolution nette de l'interface (voir [SHUKLA et al. \(2010\)](#); [TIWARI et al. \(2013\)](#); [REMIGI \(2021\)](#)). Par ailleurs, la forme des équations de l'écoulement diphasique peut être postulée *a priori* et ensuite des termes sources sont ajoutés pour imposer le respect du second principe de la thermodynamique. [BAER et NUNZIATO \(1986\)](#) est un exemple classique dans lequel les deux phases sont prises en compte sans hypothèse d'équilibre instantané.

Le régime à phase dispersée qui se caractérise par une forte densité de petites inclusions par unité de volume, peut être pris en compte en suivant chaque particule individuelle (ou "parcel") de manière lagrangienne ([ZAMANSKY et al. 2014](#)), ou en s'appuyant sur une vision *mesoscopique* ou *cinétique*, inspirée de la théorie cinétique des gaz, dans laquelle la population d'inclusions est décrite par une fonction de densité en nombre, les équations de transport pour ses moments sont résolues sur tout le domaine, puis la fonction de densité en nombre est reconstruite à partir des valeurs de ses moments (Méthode des Moments). Le lecteur intéressé est renvoyé à [LAURENT, MASSOT et VILLEDIEU \(2004\)](#); [S. de CHAISEMARTIN et al. \(2009\)](#); [MASSOT, LAURENT et al. \(2010\)](#); [KAH et al. \(2015\)](#); [DUPIF \(2018\)](#). [ESSADKI et al. \(2019\)](#) introduisent le concept de moments fractionnaires représentant le transport de quantités géométriques.

Si nous utilisons deux stratégies de modélisation différentes pour deux régimes de l'écoulement, alors elles doivent être couplées. Une approche que nous voulons citer est l'approche

ELSA introduite pour la première fois dans VALLET et BORCHI (1999), dans laquelle une équation sur la partie fluctuante du champ de densité de la zone interfaciale est dérivée. En fonction d'une valeur seuil de cette quantité fluctuante, des particules lagrangiennes peuvent être injectées dans certaines parties du domaine. Des développements plus poussés de cette approche peuvent être trouvés dans LEBAS et al. (2005); PUGGELLI (2018); REMIGI (2021) entre autres. CORDESSE, MURRONE et al. (2018) proposent un couplage de modèles dans l'esprit de BAER et NUNZIATO (1986) pour le régime à phase séparée avec une Méthode des Moments pour la zone à phase dispersée. Le problème avec l'approche de couplage est la méthodologie employée pour appliquer réellement le couplage, très souvent basée sur des corrélations empiriques avec une enveloppe d'applicabilité limitée. Une autre approche peut être proposée, s'appuyant sur un cadre mathématique unifié basé sur une approche variationnelle peut être utilisé pour fournir des équations englobant les deux régimes dans un système à deux phases.

### 1.2.4 Un cadre complet pour la dérivation et la simulation de modèles unifiés d'ordre réduit d'écoulement diphasique

Dans les situations où les DNS ne peuvent pas être utilisées en raison de la complexité du système à étudier ou des contraintes de temps sur l'exécution d'une simulation, les modèles d'ordre réduit sont nécessaires. Afin de tenir compte simultanément des petites échelles dans les régimes de phase séparée et dans la phase dispersée, une nouvelle approche basée sur le Principe d'Action Stationnaire est possible. Le Principe d'Action Stationnaire est une approche variationnelle dans laquelle une action hamiltonienne est postulée, cette action contient une certaine forme de fonction lagrangienne constituée par un ensemble d'énergies décrivant les phénomènes sous-jacents dont le modélisateur veut s'occuper. En particulier, des énergies de petites échelles représentant un comportement spécifique dans des échelles qui ne sont pas résolues par le maillage peuvent être injectées dans le Lagrangien, ou assurer l'existence d'une structure de dissipation pour les équations obtenues en appliquant l'inégalité d'entropie. Les principes fondamentaux de la méthodologie ont été présentés dans S. GAVRILYUK et GOUIN (1999); S. GAVRILYUK et SAUREL (2002); BERDICHEVSKY (2009); DELL'ISOLA et S. L. GAVRILYUK (2012), tandis que DRUI, LARAT et al. (2019) présente les spécificités du traitement des objets sphériques pulsants à petite échelle. CORDESSE (2020) présente les efforts déployés pour surmonter la limitation consistant à supposer que seules les inclusions sphériques pulsantes ont des propriétés géométriques supplémentaires à la simple fraction de volume, mais des travaux supplémentaires sont indubitablement nécessaires pour étendre la gamme des phénomènes à petite échelle pris en compte. Lors de la modélisation des contributions à petite échelle, la

forme fonctionnelle des énergies à inclure dans la dérivation peut bénéficier des connaissances fournies par le post-traitement des simulations haute-fidélité comme la [DNS](#), grâce à des outils d'analyse spécifiques. D'autres hypothèses de modélisation peuvent également être prises en compte par l'application d'une structure dissipative cohérente, en appliquant le signe d'une fonction d'entropie comme le veut le deuxième principe de la thermodynamique. Cela permet d'introduire des termes sources qui, dans d'autres études, sont introduits par d'autres méthodes telles que le calcul de la moyenne des équations instantanées locales pour la dynamique de l'interface.

**Post-traitement des [DNS](#)** Les [DNS](#) fournissent un outil très utile pour étudier en détail des configurations d'écoulement simplifiées. Les résultats sont très utiles pour comprendre les phénomènes qui se produisent aux échelles les plus basses et qui sont ensuite utilisés pour fournir des fermetures solides pour les termes créés lors du développement de modèles d'ordre réduit, suivant une approche de moyennage (voir section 3.2) ou une approche variationnelle (voir section 3.4). Pour référence, [ESSADKI \(2018\)](#); [ESSADKI et al. \(2019\)](#) présente les principes fondamentaux des calculs géométriques sur des surfaces triangulées pour estimer les courbures de l'interface. Une autre analyse qui utilise la corrélation à deux points est décrite dans le travail de [THIESSET, DUMOUCHEL et al. \(2019\)](#); [THIESSET, MÉNARD et al. \(2019\)](#); [F. THIESSET, B. DURET et al. \(2020\)](#); [F. THIESSET, T. MÉNARD et al. \(2021\)](#). L'importance du calcul correct de la courbure de l'interface est également discutée dans [BERMEJO-MORENO et PULLIN \(2008\)](#); [BERMEJO-MORENO, PULLIN et HORIUTI \(2009\)](#) dans le contexte des champs turbulents comme l'enstrophie ou dans [EVRARD \(2017\)](#); [EVRARD et al. \(2020\)](#) pour les écoulements multiphasiques.

## 1.3 Contexte de calcul scientifique

Dans les paragraphes d'introduction précédents, nous avons déjà évoqué le coût potentiellement élevé des simulations d'écoulement diphasique. La réalisation de [DNS](#) significatives, par exemple, peut nécessiter quelques centaines de millions de cellules au moins. La simulation de beaucoup de secondes de temps physique nécessite un haut niveau de parallélisme des données et des algorithmes pour être réalisable, ce qui se traduit souvent par une conception de la programmation dictée par ces structures de données sous-jacentes, avec des [Application Programming Interfaces \(APIs\)](#) parfois difficilement accessibles, avec un langage compilé et des systèmes de compilation complexes. En outre, les codes industriels de simulation multiphasique fournissent une pléthore de fonctionnalités (par exemple le code CEDRE ([LE TOUZE](#)

2015 ; CORDESSE 2020), ou le code OpenFOAM GREENSHIELDS (2015). Ces grandes bases de code avec un historique important sont souvent difficiles à modifier et à tester. Une nouvelle génération de codes est en cours de développement pour répondre à la nécessité d’une adaptation locale du maillage, comme le code canoP DRUI, FIKL et al. (2016), basé sur la bibliothèque p4est qui offre la manipulation d’octrees, ou le code SAMURAI (BELLOTTI et al. 2021) qui offre une structure de données sans arbre optimisée pour l’adaptation du maillage multirésolution.

De l’autre côté, l’intérêt toujours croissant pour l’apprentissage automatique, l’intelligence artificielle et l’extraction de crypto-monnaies a conduit à l’avancement technologique de composants matériels spécialisés tels que les GPU et les FPGA, permettant l’utilisation de langages interprétés dynamiques comme Python, pour effectuer des tâches de calcul lourdes grâce à leur interface qui permet l’intégration de code haute performance à partir de bibliothèques compilées dans des zones très spécifiques et localisées de la base de code. Des bibliothèques comme Tensorflow, pyTorch sont des standards dans leur contexte, elles sont optimisées pour exécuter des opérations sur des graphes sur les GPU et permettent l’entraînement sur de grands ensembles de données. L’écosystème Python est riche et propose des structures de données optimisées comme NumPy (HARRIS et al. 2020) qui constitue la base sur laquelle reposent les tableaux en colonnes spécialisés. On peut citer Cupy (PREFERRED INFRASTRUCTURE, INC. 2021), une mise en œuvre de NumPy au-dessus des GPU, Dask, une mise en œuvre distribuée conforme aux tableaux NumPy ou Legate NumPy (BAUER et GARLAND 2019), une mise en œuvre distribuée de NumPy qui permet de calculer sur des configurations hybrides sur plusieurs GPU et CPU. Une autre approche pour bénéficier à la fois des bibliothèques compilées ultra-optimisées et des écosystèmes interprétés flexibles, résolvant ainsi le problème des deux langages (PERKEL 2019), est le développement d’un langage numérique de nouvelle génération comme Julia.

Les langages modernes offrent la possibilité intéressante de prototyper facilement et de refactoriser le code sans perdre les performances des bibliothèques spécifiques compilées, étroitement intégrées dans le runtime du langage. Les chaînes d’outils associées à ces langages modernes appliquent les meilleures pratiques comme Continuous Integration (CI) et Continuous Development (CD), l’auto-génération de documentation, les tests unitaires, et ces outils font souvent partie de la bibliothèque standard et sont donc facilement accessibles et enseignés (comme exemple de pratiques modernes voir KLABNIK et C. NICHOLS (2018)). Ces caractéristiques sont également souhaitables dans les logiciels High Performance Computing (HPC) de nouvelle génération.

## 1.4 Contributions de cette thèse

L'objectif principal de cette thèse est d'offrir un cadre de modélisation unifié permettant la dérivation de systèmes d'équations régissant des systèmes d'écoulement diphasique caractérisés par une structure mathématique solide via un cadre variationnel nommé Principe d'Action Stationnaire. Cet effort est soutenu par un ensemble d'outils informatiques adaptés qui permettent le choix rationnel des hypothèses de modélisation et les simulations efficaces des modèles développés, éventuellement sur des architectures informatiques modernes. Ce travail vise à traiter les trois aspects principaux concernant la modélisation de l'écoulement diphasique que nous avons identifiés dans les sections d'introduction précédentes, à savoir : le développement de modèles d'ordre réduit via le Principe d'Action Stationnaire, la mise en œuvre d'un outil de post-traitement géométrique `DNS` appelé `Mercur(v)e` qui est utilisé pour clarifier les bonnes hypothèses à faire lors de l'élaboration d'un modèle d'ordre réduit, et le développement d'une bibliothèque Python appelée `josiepy` qui agit comme un playbook pour tester rapidement les modèles de lois de conservation, les schémas numériques, les conditions limites, les configurations de domaine et une stratégie moderne du calcul scientifique. Le manuscrit est organisé comme suit :

- Le Chapitre 3 est consacré à la discussion sur les stratégies de dérivation des modèles d'ordre réduit. Nous introduisons deux méthodes différentes pour la dérivation du système d'équations concernant les écoulements diphasiques, la première basée sur la moyenne des équations Navier-Stokes (NS) locales instantanées pour chaque phase, la seconde qui exploite une méthodologie variationnelle appelée Principe d'Action Stationnaire. Ensuite, nous exploitons le Principe d'Action Stationnaire pour dériver un ensemble de modèles présentant différentes hypothèses à petite échelle dans le but final d'injecter des équations pilotant les paramètres géométriques qui pourraient améliorer la précision des modèles lorsqu'ils traitent des inclusions non sphériques.
- Le Chapitre 4 présente les efforts de développement engagés dans la création de `Mercur(v)e`, une bibliothèque dédiée au post-traitement des `DNS` d'écoulement diphasique. La bibliothèque est basée sur une triangulation de l'interface, sur laquelle sont calculées des approximations discrètes de paramètres géométriques tels que la courbure moyenne et la courbure de Gauss. L'algorithme choisi pour effectuer ces calculs préserve les invariants topologiques comme le théorème de Gauss-Bonnet, permettant de compter les objets dans un domaine qui sont homéomorphes à des sphères. De plus, la bibliothèque offre également un noyau de moyennage qui permet de lisser les calculs bruités sur des



triangulations imparfaites sans perdre la préservation des invariants topologiques. La bibliothèque est utilisée pour soutenir les hypothèses de modélisation pour l’élaboration d’un modèle d’ordre réduit supposant l’oscillation isochore de gouttelettes ellipsoïdales comme modèle à petite échelle.

- Dans le chapitre 5 nous discutons du cadre numérique sur lequel la bibliothèque `josiepy` est construite. Nous fournissons un aperçu général de la théorie de la génération de maillage et des schémas numériques dans le contexte Volumes Finis de la bibliothèque, nous reformulons la littérature classique sur le sujet sous une forme différente qui contextualise l’énonciation numérique des schémas aux spécificités de l’implémentation du code `josiepy`. Nous discutons de manière assez détaillée de tous les modèles, algorithmes et schémas qui sont déjà implémentés à ce jour dans `josiepy`. Nous montrons un ensemble de tests de vérification sur différents modèles, qui sont reproductibles avec le code actuel disponible en ligne sous une licence [Free and Open Source Software \(FOSS\)](#) et facilement disponibles sous forme de notebooks Jupyter ou de tests d’intégration dans le dépôt contenant le code source, y compris les modèles innovants dérivés et présentés dans chapitre 3.
- Le Chapitre 6 décrit tous les aspects liés au génie logiciel des bibliothèques que nous avons développées au cours de la thèse : une bibliothèque Python nommée `josiepy` visant à simuler des systèmes d’EDP potentiellement génériques et la bibliothèque `Mercur(v)` dont le but est d’exposer une procédure de post-traitement géométrique pour collecter des informations intéressantes à partir de simulations haute définition comme les [DNS](#). Dans ce chapitre, nous discutons de la philosophie qui sous-tend le choix de créer de tels logiciels, des défis actuels et des perspectives possibles dans un avenir proche.

Les sujets que nous avons explorés dans l’accomplissement de cette thèse conduisent aux contributions suivantes :

- Articles de revues :
  - Ruben DI BATTISTA, Thibault MÉNARD, Stephane DE CHAISEMARTIN et Marc MASSOT (2021). “A Computational Framework Based on the Discrete Estimation of Geometrical Properties over Triangulated Interfaces Preserving Topological Invariants to Design and Validate Two-Phase Flow Models”. In : *Fluids*. In preparation
  - Pierre CORDESSE, Ruben DI BATTISTA, Quentin CHEVALIER, Lionel MATUSZEWSKI, Thibault MÉNARD, Samuel KOKH et Marc MASSOT (2020). “A Diffuse Interface



Approach For Disperse Two-Phase Flows Involving Dual-Scale Kinematics Of Droplet Deformation Based On Geometrical Variables”. In : *ESAIM : Proceedings*, p. 22. URL : <https://hal.archives-ouvertes.fr/hal-02879950v1>

- Actes de conférence :

- Alberto REMIGI, Ruben DI BATTISTA, François-Xavier DEMOULIN, Benjamin DURET, Marc MASSOT, Thibaut MÉNARD et Hugo DENEUVILLE (19-24 mai 2019). “Exploring Different Approaches for the Simulation of Multi-Scale Atomization Process”. In : International Conference on Multiphase Flow. Rio de Janeiro. URL : <https://hal.archives-ouvertes.fr/hal-02379257>
- Pierre CORDESSE, Ruben DI BATTISTA, Samuel KOKH et Marc MASSOT (19-24 mai 2019). “Derivation of a Two-Phase Flow Model with Two-Scale Kinematics and Surface Tension by Means of Variational Calculus”. In : 10th International Conference on Multiphase Flow. Rio de Janeiro. URL : <https://hal.archives-ouvertes.fr/hal-02194951>
- Ruben DI BATTISTA, Iván BERMEJO-MORENO, Thibaut MÉNARD, Stéphane de CHAISEMARTIN et Marc MASSOT (19-24 mai 2019). “Post-Processing of Two-Phase DNS Simulations Exploiting Geometrical Features and Topological Invariants to Extract Flow Statistics and Droplets Number Density”. In : International Conference on Multiphase Flow. Rio de Janeiro. URL : <https://hal.archives-ouvertes.fr/hal-02345825v1>

- Chapitres d’ouvrage :

- Pierre CORDESSE, Ruben DI BATTISTA, Florence DRUI, Samuel KOKH et Marc MASSOT (2020). “Derivation of a Two-Phase Flow Model with Two-Scale Kinematics, Geometric Variables and Surface Tension Using Variational Calculus”. In : *Proceedings of the NASA Summer Program*. Nasa Technical Memorandum. URL : <https://hal.archives-ouvertes.fr/hal-02336996>
- Ruben DI BATTISTA, Iván BERMEJO-MORENO, Thibaut MÉNARD et Marc MASSOT (2019). “Geometrical Characterization and DNS Post-Processing of the 3D Objects in Two-Phase Flow : Collision of Two Droplets”. In : *NASA Technical Memorandum*

- Logiciels

- Ruben DI BATTISTA (2018). *Mercur(v)e — A Library to Exploit Geometrical and Topological Properties to Allow Post-Processing of DNS Simulations (and Much More)*. URL : <https://gitlab.com/rubendibattista/mercurve>
- Ruben DI BATTISTA (2019). *Josiepy — A 2D PDE Solver Written in Python without Compromising (Too Much) Performance*. URL : <https://gitlab.com/rubendibattista/josiepy>

# 2 | Introduction

## 2.1 Industrial Context

---

### 2.1.1 Injection systems for space propulsion

The 16th June 1969 Neil Armstrong, Buzz Aldrin and Michael Collins leave the motherplanet for the *Mare Tranquillitatis* on top of the biggest, largest, heaviest and most powerful space launcher human race has ever built at this date: the Saturn V. The Saturn V shown in Fig. 2.1a was a gargantuan 110.6 m tall rocket with a diameter of 10.1 m powered by 5 F-1 Liquid Rocket Engines (LREs), each one capable of delivering 7770 kN of thrust in vacuum thanks to the cryogenic Liquid Oxygen (LOX)-Liquid Hydrogen (LH2) gas-generator cycle they employ. The 20th June 1969 the three astronauts land on the Moon on board of the LEM, after detachment from the CSM, the module responsible for lunar orbit maneuvering, powered by a Nitric Acid (HNO<sub>3</sub>)-Unsymmetrical dimethylhydrazine (UDMH) hypergolic LRE. A giant leap that widens mankind horizons for exploration, placing the new “Pillars of Hercules” outside Earth atmosphere and beyond. Indeed, “*Earth is the cradle of humanity, but one cannot live in a cradle forever*” (K. E. Tsiolkovsky).

The “First Space Race” paved the way for incredibly fast-paced innovation in the propulsion field whose research interest outreach is still tangible today. We are witnessing the “Second Space Race”, with the advent of privatization of the space sector, the SpaceX reusability revolution, and the Mars exploration and colonization plans. The extreme motivation in the epoch of the Apollo missions lead to the development of different space propulsion technologies:

- Liquid Rocket Engines: space engines which feature both the oxidizer and the fuel in liquid state. The propellants are stored in tanks from which they are pumped to the injector head by a pressurized system (blow-down technology), or some sort of pumping machinery with very different degrees of complexity: electric pumps (Electron rocket by Rocket Lab), expander cycle powered turbopumps (Ariane 5, Atlas V, Delta IV, New Glenn, Long March 5, KVTk, H-I, H-II), gas generator cycle powered turbopumps (F1

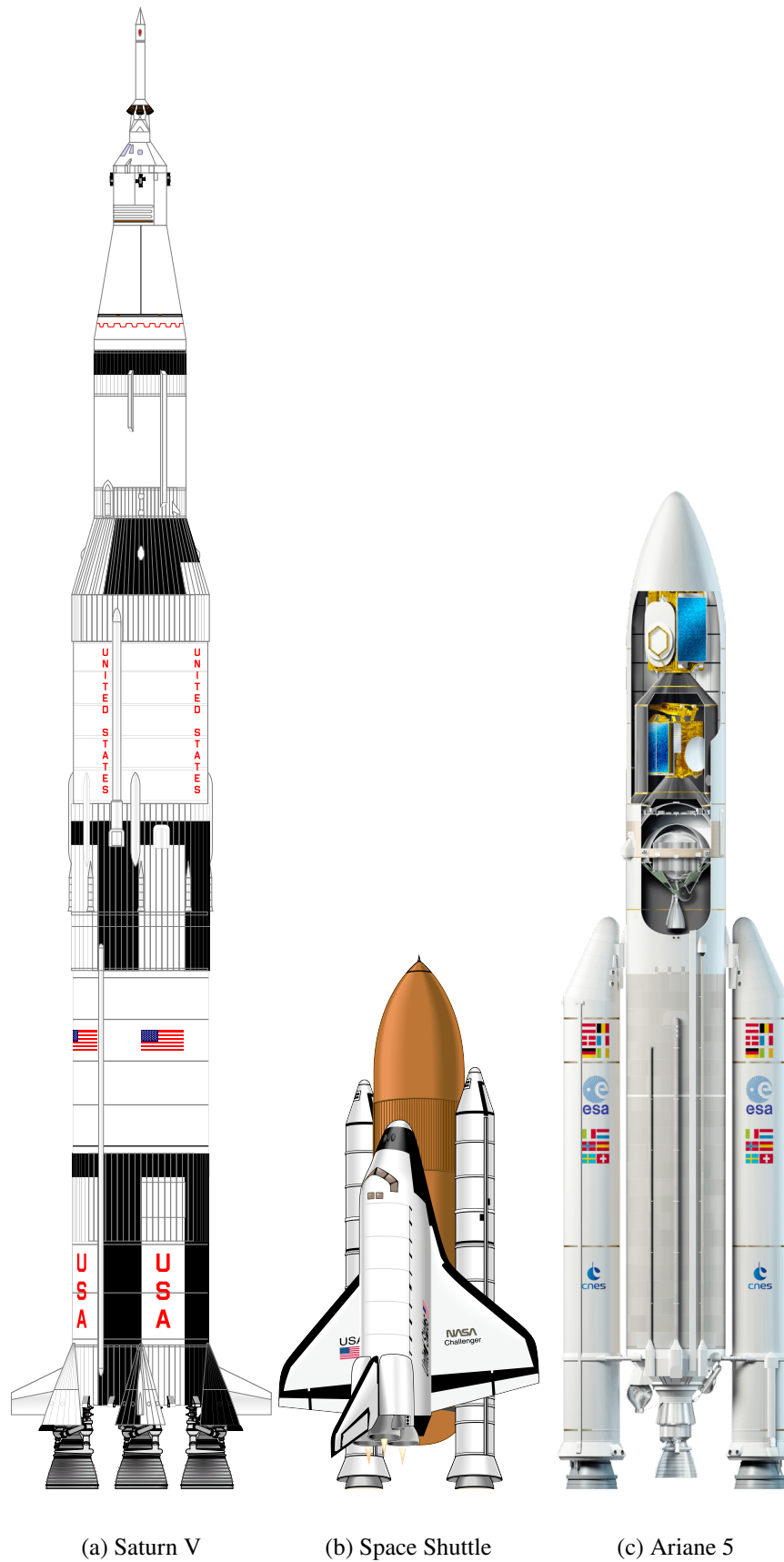


Figure 2.1: An opinionated and absolutely not in scale selection of cool “legacy” launch systems

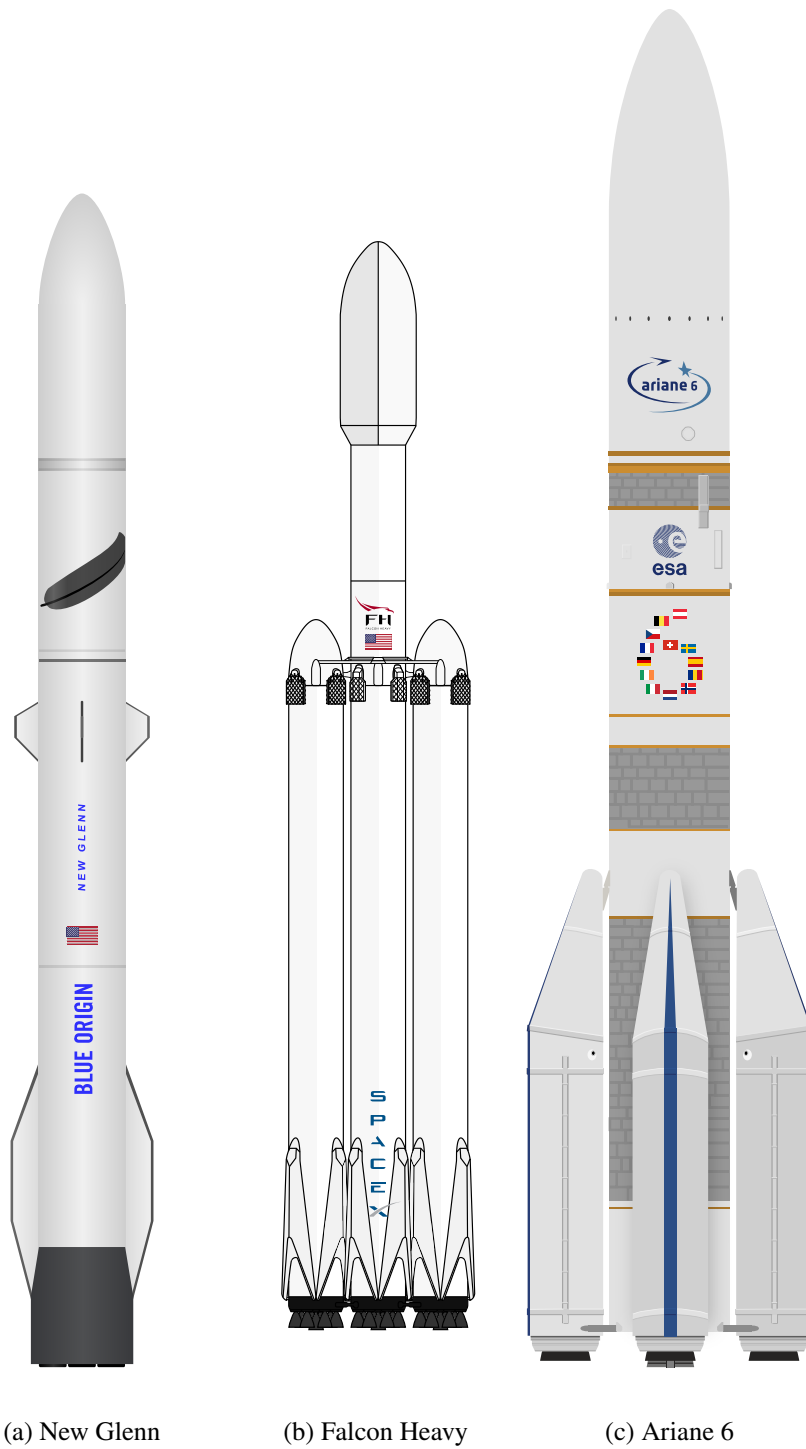


Figure 2.2: An opinionated and absolutely not in scale selection of cool “new generation” launch systems

engines, Falcon 9), staged combustion cycle powered turbopumps (Space Shuttle, Energia, New Glenn, Starship, N1, Proton, H-II, Angara). The careful design of the injector plate is very much of interest to avoid combustion instability and associated catastrophic accidents (*e.g.* Apollo 13 (Irvine 2008)). This technology is the most efficient in the framework of thermal propulsion and it is the most used in legacy (Fig. 2.1) and new generation (Fig. 2.2) systems. An additional subdivision can be done in terms of the nature of the propellants driving the engines:

- **Hypergolic Propellants:** Propellants that do not need an initial source of energy for ignition, just the mixing of the components creates the condition for ignition. Notable examples of hypergolic oxidizers are: **Dinitrogen Tetroxide**  $\text{N}_2\text{O}_4$  (NTO),  **$\text{HNO}_3$** , **Hydrogen Peroxide** ( $\text{H}_2\text{O}_2$ ). As fuel, hydrazine derivatives are mostly used.
- **Cryogenic Propellants:** high specific impulse propellants with generally low boiling point that need to be cooled down to increase density and reduce volume of tanks. These propellants power the most efficient engines (the highest specific impulse has been achieved with the space shuttle **LRE**, Fig. 2.1b for commercial systems. Higher specific impulse engines are possible with fluorine as oxidizer, but they produce emissions that are extremely toxic and corrosive, hence they are not commercially employed) and most commonly they are: hydrogen and more recently methane as fuels, **LOX** as oxidizer. Sometimes Kerosene (RP1) is used as fuel in pair with cryogenic **LOX**. Modern systems, like the Raptor engine by SpaceX, the Blue Origin BE-4, Avio M10 Engine, are leading towards the employment of cryogenic **LOX** and **LCH<sub>4</sub>** as propellants for the engines.
- **Solid Rocket Motors:** space engines that feature both the oxidizer and the fuel in solid aggregation state. The propellants are generally casted into composite liners and have long storability times (in the order of years). They are generally used as high thrust/weight support engines (*i.e.* “boosters”) in the first stages of a space launch. Compared to **LREs**, **Solid Rocket Motors** have less moving parts and are in general “simpler”, they can be mass produced and easily stored even if less efficient thermochemically speaking. The most famous solid rocket engines are the boosters of the Shuttle (Fig. 2.1b), but they are still in large use today: the European Ariane 5, the next generation Ariane 6, the Italian Vega feature **SRMs**. **SRMs** are also used in the defense industry as lethality systems to be equipped on-board of fighter jets. Limiting the discussion to space propulsion applications, they are generally made of a binder plastic matrix, very often **Hydroxyl-terminated**

polybutadiene (HTPB), in which micro or nano-sized metals (often Aluminum) act as fuel, and ammonium nitrate or perchlorate act as oxidizers. We refer the reader to the work of Doisneau, Dupays, et al. (2011); Doisneau, Sibra, et al. (2014); Dupif (2018); François et al. (2020); François et al. (2020) on the modeling and simulation of SRMs phenomena.

- Hybrid Rocket Engines: a less developed technology for space engines that feature one of the two propellants in liquid state, often the oxidizer, and the other one in solid state. HREs have theoretical benefits *w.r.t.* LREs and SRMs for commercial space business because they can achieve similar performances as RP1-LOX LREs with one turbopump system instead of two when using paraffin as fuels, with evident cost benefits. One of the reasons why they did not catch industrial usage is because the combustion of HREs is regulated by diffusion (while for a LRE or SRM the combustion is generally driven by chemical kinetics), leading to major problems in flexibility, performance, scaling and simulation accuracy. Attempts to use them as a viable solution have been performed by AMROC in the past, now Virgin Galactic that acquired the Intellectual Property (IP) with its SpaceShipTwo spaceplane Fig. 2.3. A renewed interest spanned in recent times following interesting results when using wax and paraffin as fuels (Karabeyoglu et al. 2004): attempts to build HRE-powered nano-launchers have been started by Leaf Space in Italy (*Primo | A Small Lightweight Nanolauncher for Your Small Sat* 2018), Hybrid Propulsion for Space in France (*Hybrid Propulsion for Space* 2021), among others. Apart from paraffin, other potential fuels used in the wild are HTPB or polycarbonate. Oxidizer is generally oxygen, but nitrous oxide is also used especially for low tier systems, like for example rockets made by students (see Skyward Experimental Rocketry in Italy, DARE in Netherlands, HyImpulse in Germany, among others).

Limiting the discussion to two-phase flow configurations in which phases do not appear in solid state, as is the case for SRMs, two-phase flow modeling and simulation play a major role for the cost effective and safe design of an engine. As we already touched on, the careful design of the tank baffles, injector plate and combustion chamber is vital to avoid the insurgence of sustained pressure oscillations and instabilities that can easily cause Rapid Unscheduled Disassembling (RUD). Moreover, the optimization of a HRE requires accurate predictions of the flow field and combustion products, especially when using paraffin as fuel, that melts creating a spray of fuel droplets, which bears some similarity with what is going on in Diesel engines.



Figure 2.3: Virgin Galactic SpaceShipTwo HRE

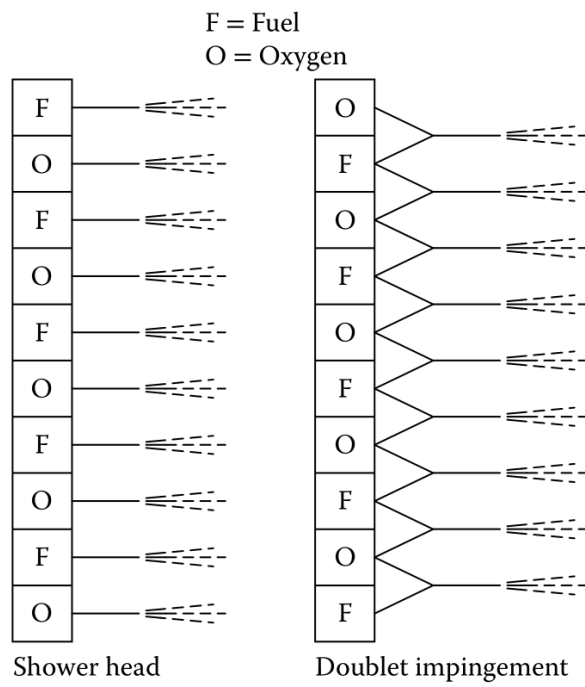


Figure 2.4: LRE injector configuration, courtesy of Lefebvre and McDonell (2017)



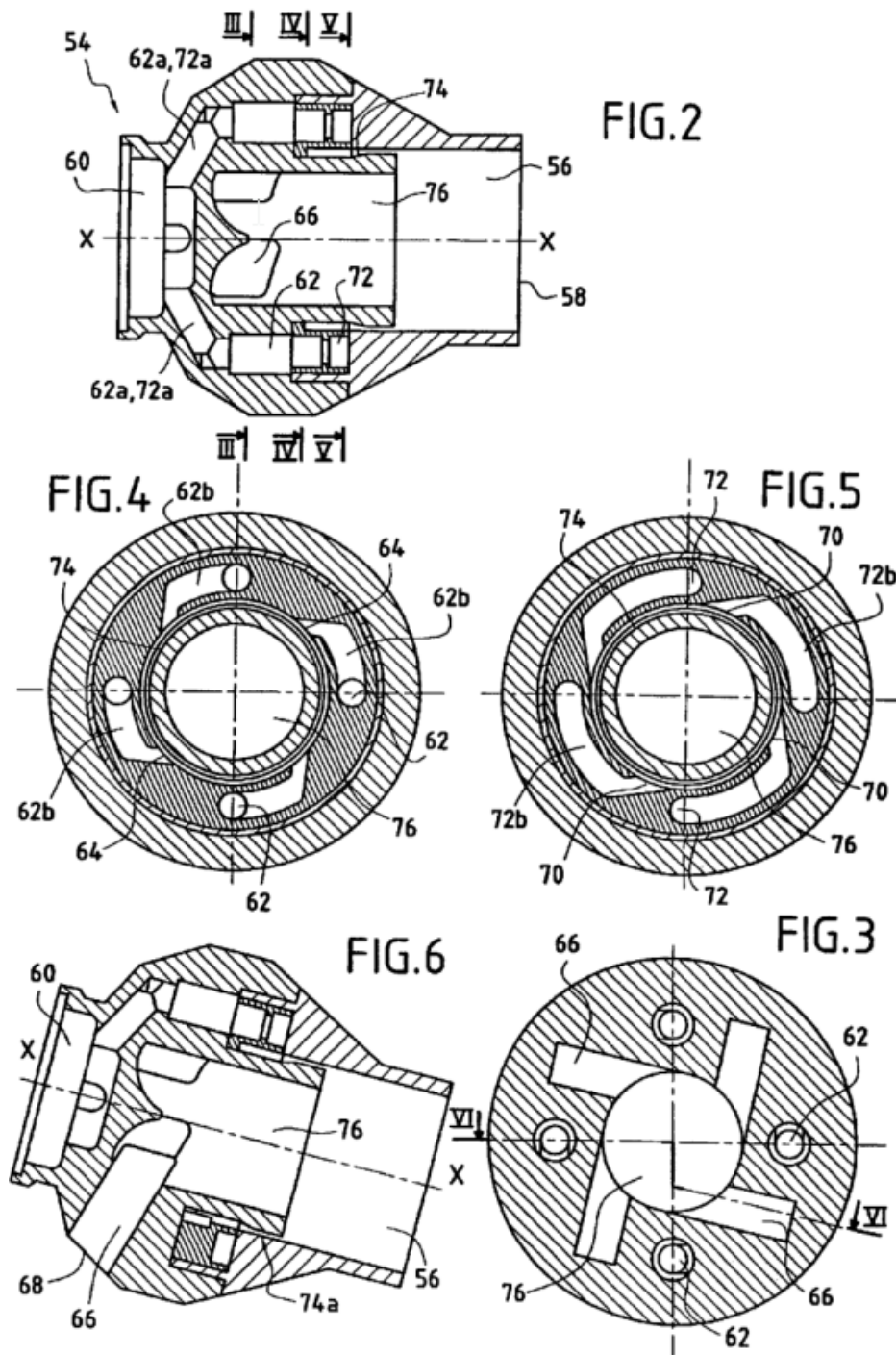


Figure 2.5: The multipoint injection feed system from David et al. (2004)

### 2.1.2 Injection systems for air-breathing propulsion

Injection modeling, simulation and optimization are themes not limited to the space propulsive context. In recent years a substantial switch in the political strategy regarding climate change has been witnessed. In particular, more often than before, stricter policies are enforced for emissions produced by industrial and user power generation, car and flight traveling. As a note, France government plans to ban domestic flights as a move to reduce pollutants (“[France Moves to Ban Short-Haul Domestic Flights](#)” 2021). [Lee et al. \(2001\)](#), even if not extremely up to date, provides an historical analysis on the evolution of emissions for aircrafts. They report a possible three-to sevenfold increase by 2050 of  $CO_2$  emissions. In order to comply to the always stricter regulations, the propulsion technology must evolve to provide better efficiency. For aeronautical applications, higher bypass ratios and low combustion temperatures are among the key aspects that allow improvement of combustion efficiency and reduction of pollutants and greenhouse gases and the fuel injection in the combustor has a major impact on those metrics. While for [LREs](#) simple single orifice atomizers are used, sometimes configured as impinging jets where the fuel jet impinges against the oxidizer at a certain angle as shown in [Fig. 2.4](#), for the aerospace air-breathing propulsion, more complex configurations are required. As an example, [Fig. 2.5](#) shows the technical details of a multi-point coaxial injector described in the patent from [David et al. \(2004\)](#), and it describes a configuration where not only one primary fuel feed is present, but also several secondary feeds (noted **62** and **72** in [Fig. 2.5](#)) distributed along the central axis (noted X—X axis in [Fig. 2.5](#)). Motors like the LEAP from Safran Aircraft Engines shown in [Fig. 2.6](#), are the result of this optimization. In those extreme conditions and tight flight envelopes, mastering the injection process is crucial and lots of research is addressed to this endeavor (*e.g.* [Remigi \(2021\)](#) most recently). The different points of injection present in an engine have the combined aim of controlling thermal fluxes on the walls to avoid [RUD](#), improving the life expectations of these costly components increasing the [MTBF](#) but also to control the flame temperature to avoid excess production of  $NO_x$  and other pollutants. The correct nebulization of fuel into the combustion chambers is therefore vital, and due to the access and observation difficulties that are inherent to aeronautical engines, modeling and simulation are important tools in the early phase of the design cycle, but also later on during the commissioning, control and decommissioning of those systems. The current state of the art simulations treat the surrounding gaseous carrier with an Eulerian approach where the fuel droplets are then injected individually or as virtual particles composed by a group of individual physical objects and tracked with a Lagrangian approach ([Lebas et al. 2005](#); [Sanjosé et al. 2011](#); [Vignat et al. 2021](#)). This approach can be non-predictive when the configuration becomes very complex (as the multipoint injection systems of [Fig. 2.5](#)) and when



Figure 2.6: LEAP aeronautical engine. Courtesy of Safran Aircraft Engines

operating conditions are far from experimental ones so that the calibration of the Lagrangian model parameters fails to represent the underlying physics of primary atomization. This situation is also encountered for automotive direct injection systems. There has been some attempts to develop a general formalism for the liquid interface destabilization and break-up using the interfacial surface density representation starting from the work of [Vallet and Borghi \(1999\)](#) continued with empirical derivation for separate phase flows based on Kelvin-Helmholtz instability analysis in [Jay et al. \(2006\)](#) and more recently a representation with two interfacial surface density for separate and disperse phase [Devassy \(2014\)](#); [Devassy et al. \(2015\)](#). To now none of these approach is really able to represent the links between the physics inside the injector and the developed spray region at all scales and with real dimensions and complex geometrical features. Therefore the correct resolution of the liquid film close to the injector is paramount to improve the simulation outcomes. For that reason, an appropriate reduced-order model that includes sound small-scales treatment is required, in particular for the interfacial surface density representation.

### 2.1.3 Direct injection in reciprocating engines

The environmental theme is as important also in the development of new generation reciprocating engines for land transport, like cars and trucks. In most Western countries, governments have planned a phase out of Diesel engines in order to arguably reduce pollution. The phase out is planned as early as by 2025 in Norway (*Phase-out of Fossil Fuel Vehicles 2021*). Therefore,

in the foreseeable future, most of the R&D funding will be destined to improve combustion efficiency of gasoline direct injection engines. In this technology, the mixture of air and fuel needs to be carefully regulated to avoid auto-ignition and detonation, and also to keep temperature at a reasonable level in order to avoid  $NO_x$  production (that are anyway also captured downstream to the thermal engine system with a catalyzer to comply with the stringent anti-pollution local laws). The extreme conditions that are characteristic of these systems and also the fact that the combustion chambers changes its volume in time due to the piston movement, make the study of the injection phenomenon for direct injection reciprocating engines very challenging on the modeling side, but also on the numerical and geometrical aspects.

### 2.2 Scientific Context

Injection is an inherently multiscale phenomenon. Fig. 2.7 shows a schematic of a typical scenario where an injector system injects some fuel into a combustion chamber in which a gas, typically air, is present. On the left part of the image, we have the fuel coming out of the injector head. This part of the domain is characterized by a liquid phase clearly separated by an interface from the surrounding gaseous atmosphere and it is named *separated phase zone* or regime. At the end of the chamber the interface deformed at such a level that a lot of droplets detach forming a spray, this is the *disperse phase regime*. While the two opposite regimes are clearly identified and modeled, they still need to be linked somehow, that is an additional zone in which big and small scales co-exist and a clear distinction is not possible named the *mixed region*. Reference length scales associated to atomizers orifice diameter are of the order of  $1e-3$  m. The interface then undergoes destabilization associated to velocity gradients (Kelvin-Helmoltz instabilities), surface tension effects (Plateau-Rayleigh instability), turbulence interaction and a complex mix of all of them. The interface deforms until it creates ligaments. Further downstream those ligaments detach from the core of the jet leading to **primary breakup**. In the most downstream part of the represented domain, the original detached ligaments additionally break up into smaller droplets, until almost spherical droplets of the order of  $1e-4$  m to  $1e-6$  m are created (Lefebvre and McDonell 2017); that is the **secondary breakup**, leading to the creation of the disperse phase regime. A practical scenario is shown in Fig. 2.8 in which the two regimes are clearly shown from right to left. The most important non-dimensional numbers that govern the phenomenon are the Reynolds (Re), Weber (We) and Ohnesorge (Oh) numbers:

$$Re_k = \frac{\rho_k u_k L_k}{\mu_k} \quad (2.1)$$

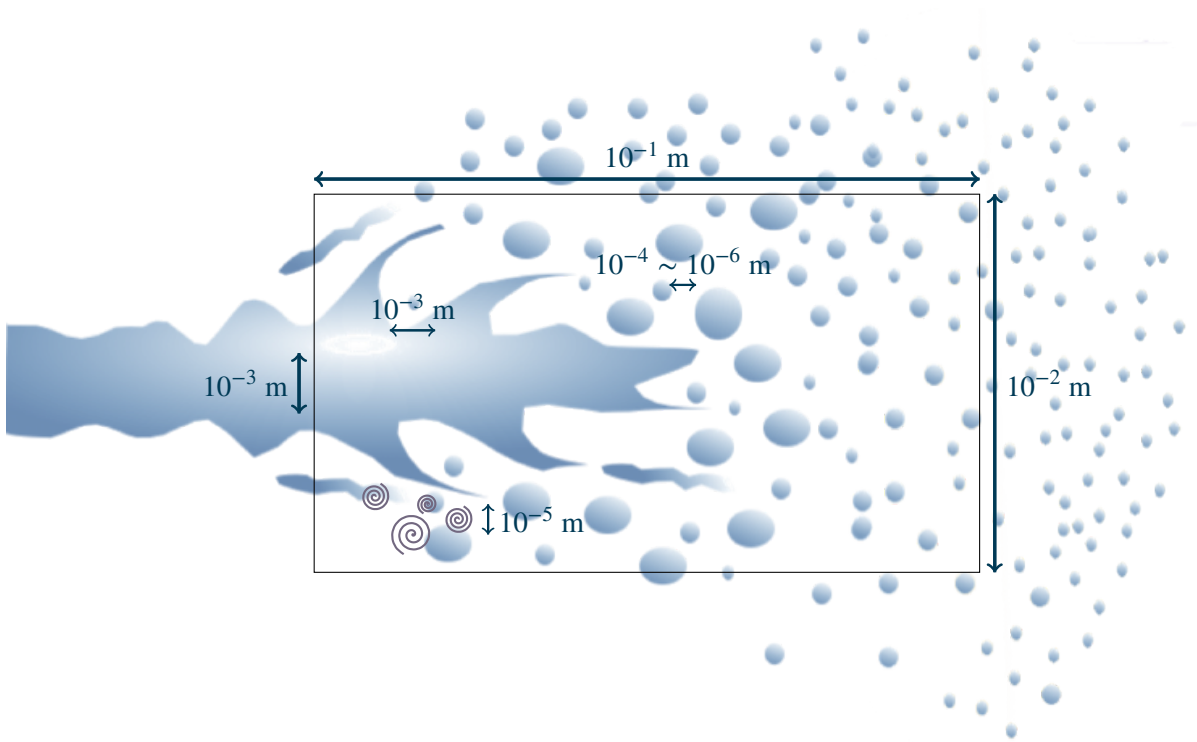
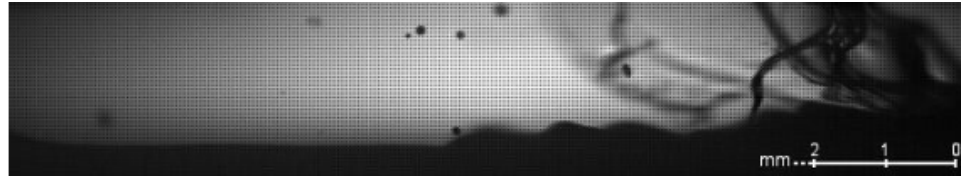
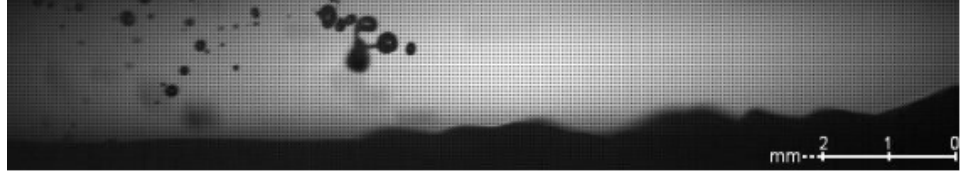


Figure 2.7: Two-phase flows are inherently multiscale



(a) Initial destabilization of the paraffin interface



(b) Secondary breakup and creation of satellite droplets downstream

Figure 2.8: A cold gas visualization of a paraffin HRE, gas flows from right to left. Courtesy of Rehakavas et al. (2015)

$$\text{We} = \frac{\rho_g (u_g - u_l)^2 L_l}{\sigma_l} \quad (2.2)$$

$$\text{Oh}_k = \frac{\sqrt{\text{We}}}{\text{Re}_k} \quad (2.3)$$

where the index  $k = l, g$  indicates a generic phase,  $l$  indicates the liquid phase and  $g$  the gaseous phase.  $L_k$  is a phase-specific reference length. Fig. 2.9 shows range for the non-dimensional quantities for few coaxial round jet cases. A detailed discussion of different type of atomizers for injection systems can be found in Lefebvre and McDonell (2017).



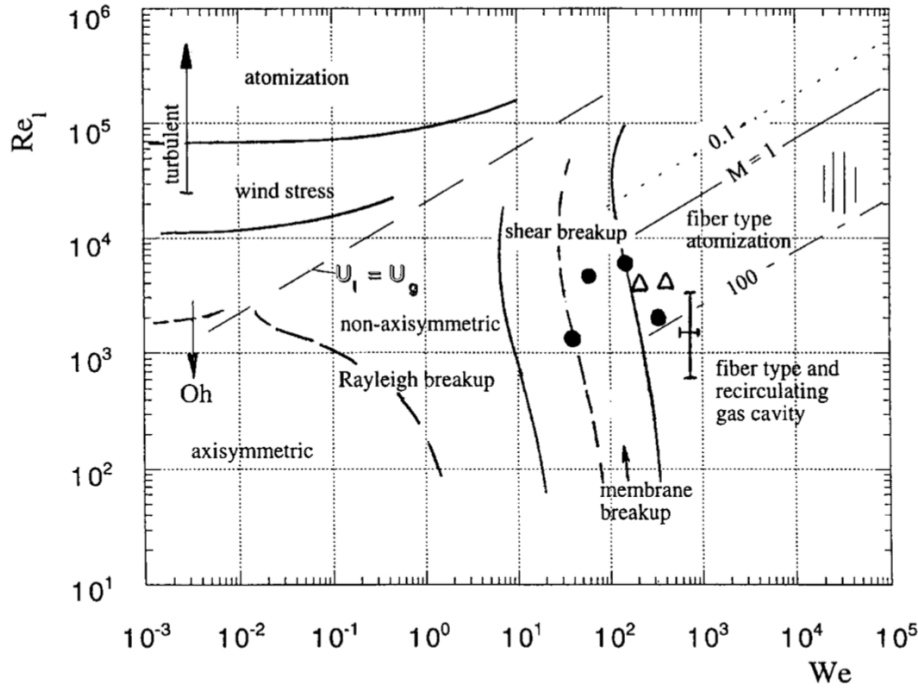


Figure 2.9: Typical range of Weber and Reynolds number in coaxial jet stream from Lasheras and Hopfinger (2000)

### 2.2.1 Different models for different regimes

Since the injection problem is inherently multiscale, different approaches have been developed to cope with the complexity and be able to simulate with predictable accuracy even the most elaborate systems. In some simplified situations, high fidelity approaches can be employed to treat the entire spectrum of scales and regimes, while in most situations, simplifications are required since some scales of the problem cannot be resolved. *Reduced order* models are therefore required: most often two different models are used, one for the separated phase regime, in which typically the fields are governed by conservation equations in Eulerian form, and the other where individual disperse objects are tracked in a Lagrangian way or their *NDF* is reconstructed from a selection of its moments. Other approaches are instead tailored to provide a unified modeling context that can tackle both flow regimes at the same time.

#### 2.2.1.1 Models that aim at resolving all the scales

If one wants to be able to simulate even the finest scales of the problem, two major approaches are possible. The first one is based on the work of Cahn and Hilliard (1958) and successive derivatives (*e.g.* C. Liu and Shen 2003), called “phase field” models. The Cahn and Hilliard

work considers the interface between the two fluids with finite thickness. While this approach is perfectly reasonable in situation where the thickness of the interface is not too thin, in a more general scenario the thickness of the interface can be of the order of few angstroms, hence requiring a very fine mesh to be resolved with satisfaction. In alternative, solving the NS equations for each phase together with jump equations at the interface, that is therefore considered as a discontinuity surface, with a very fine mesh is also possible. The fact that the interface is not resolved as it is done in the [Cahn and Hilliard \(1958\)](#) is already a modeling hypothesis. In the single phase scenario, Direct Numerical Simulations (DNSs) are defined as the simulations that solve NS equations down to the smallest known scale that is the Kolmogorov scale, without additional reduced modeling of any sort. This approach is not directly translatable to two-phase flows because those have an interface that is often thought as infinitely thin, *i.e.* it is difficult to identify a “Kolmogorov” length scale; hence the results of such simulations are often strongly dependent on mesh resolution and numerical schemes employed ([Ling, Fuster, et al. 2017](#)), together with interface tracking strategies. However, for vocabulary simplicity, we will use the term DNS to define the models that do not assume additional modeling for small-scales phenomena, both related to interface dynamics (apart from the different techniques of interface tracking employed) and turbulence modeling, even in the two-phase flow context. The interface can be tracked explicitly with Lagrangian tracking on a grid that moves with the fluid, as in [Tryggvason et al. \(2001\)](#); [James Glimm et al. \(2006\)](#). Alternatively, a fully Eulerian approach can also be used, where the interface is captured using an advected color function (VOF method ([Hirt and B. Nichols 1981](#))), or a distance function ([Sethian 1996](#); [Russo and Smereka 2000](#); [Zhao 2004](#)), or even both of them in a coupled way ([Fedkiw et al. 1999](#); [C. Liu and Shen 2003](#); [Vaudor et al. 2017](#)). Compressible VOF techniques are also possible ([B. Duret et al. 2018](#)). [Mirjalili et al. \(2019\)](#) compare the VOF and Level-Set approaches. An example of a compressible two-phase flow high-fidelity simulation for supercritical injection can be found in [Petit et al. \(2013\)](#), while in [Zou et al. \(2019\)](#) a level-set method for low-Mach compressible flows is discussed.

### 2.2.1.2 Reduced order Models

When dealing with reduced order models, we want to be able to approach both the separated phase and disperse phase regimes in a typical injection system. Two main strategies are available: employing different submodels for each different regime, and then coupling them often with empirical or semi-empirical correlations or introducing a unified framework that is capable of facing both situations.



The separated phase regime in which the two phases are clearly separated by an interface. The classical way of deriving a set of governing equations is via the averaging of local instantaneous phasic equations *à la* Ishii and Mishima (1984). Ishii, Kim, et al. (2002); Rusche (2003); Drew and Passman (2006) leverage this method. In those models, the interface is often not reconstructed, it is a transported field that diffuses around the actual position the interface would have in the physical situation. The degree of diffusion is controlled by the mesh resolution and the order of the employed numerical scheme and they generally do not include any specific treatment for the interfacial small scales. The averaging process exerts unclosed terms that need further analysis to be closed. Specific source terms (“contraction terms”) can be used to force sharp resolution of the interface (see Shukla et al. (2010); Tiwari et al. (2013); Remigi (2021)). In alternative, the form of the equations for the two-phase flow can be postulated *a priori* and then source terms are added enforcing the respect of the second principle of thermodynamics. Baer and Nunziato (1986) is a classical example in which both phases are accounted for without any instantaneous equilibrium assumption.

The disperse phase regime which features high density of small inclusions per unit volume, can be accounted for tracking each individual particle (or parcels) in a Lagrangian way (Zamansky et al. 2014), or leveraging a *mesoscopic* or *kinetic* vision, inspired by the Kinetic Theory of gases, in which the population of inclusions is described by a NDF, transport equations for its moments are solved all over the domain, and then the NDF is reconstructed from the values of its moments (Method of Moments (MoM)). The interested reader is referred to Laurent, Massot, and Villedieu (2004); S. de Chaisemartin et al. (2009); Massot, Laurent, et al. (2010); Kah et al. (2015); Dupif (2018). Essadki et al. (2019) introduce the concept of fractional moments representing the transport of geometrical quantities.

If we use two different modeling strategies for two regimes of the flow, then they need to be coupled. One approach we want to cite is the ELSA approach first introduced in Vallet and Borghi (1999), in which an equation on the fluctuating part of the interfacial area density field is derived. Based on a threshold value of this fluctuating quantity, Lagrangian particles can be injected in certain part of the domain. Further development of this approach can be found in Lebas et al. (2005); Puggelli (2018); Remigi (2021) among others. Cordesse, Murrone, et al. (2018) propose a coupling of models in the spirit of Baer and Nunziato (1986) for the separated phase regime with a MoM for the disperse phase zone. The problem with coupling approach is the methodology employed to actually enforce the coupling, very often based on empirical correlations with limited envelope of applicability. Alternatively, a unified mathematical framework would be desirable in order to provide equations encompassing both regimes in a

two-phase system with small scale modeling; one path is to use a variational approach coupled to a second principle of thermodynamics.

### 2.2.2 A comprehensive framework for the derivation and simulation of unified two-phase flow reduced order models

In the situations where **DNS** cannot be used because of the complexity of the system to study or time constraints on the execution of a simulation, reduced order models are required. In order to account for small scales in the separated phase **and** in the disperse phase regimes simultaneously, a novel approach based on the **Stationary Action Principle (SAP)** is possible. The **SAP** is a variational approach in which a Hamiltonian action is postulated, this action contains a certain form of a Lagrangian functional constituted by a set of energies describing the underlying phenomena the modeler is willing to take care of. In particular, small scales energies representing a specific behavior in scales that are not resolved by the mesh can be injected in the Lagrangian, or ensuring the existence of a dissipation structure for the equations obtained enforcing the entropy inequality. The fundamentals of the methodology have been presented in **S. Gavriluk and Gouin (1999)**; **S. Gavriluk and Saurel (2002)**; **Berdichevsky (2009)**; **Dell’Isola and S. L. Gavriluk (2012)**, while **Drui, Larat, et al. (2019)** presents the specificities of addressing small scale pulsating spherical objects. **Cordesse (2020)** introduces efforts to overcome the limitation of assuming just spherical pulsating inclusions with additional geometrical properties to the mere volume fraction, but further work is undoubtedly needed to extend the range of small scale phenomena accounted for. When modeling small scale contributions, the functional form of the energies to be included in the derivation can benefit from insights provided by the post-processing of high-fidelity simulations like the **DNS**. Additional modeling assumption can also be taken into account via the enforcement of a coherent dissipative structure, *i.e.* enforcing the sign of an entropy function as the second principle of thermodynamics mandates. This permits to introduce source terms that in other studies are introduced via other methods such as averaging local instantaneous equations for the interface dynamics.

Post-processing of **DNS** provide a very useful tool to investigate in details simplified flow configurations. The results are very useful in order to provide insight on phenomena happening at lowest scales that are then used to provide sound closures for terms that are created during the development of reduced order models, following an averaging approach (see section 3.2) or a variational one (see section 3.4). For reference, **Essadki (2018)**; **Essadki et al. (2019)** introduces the fundamentals of the geometrical computations on triangulated surfaces to estimate

curvatures of the interface. Another analysis that uses two-point correlation is described in the work of Thiesset, Dumouchel, et al. (2019); Thiesset, Ménard, et al. (2019); F. Thiesset, B. Duret, et al. (2020); F. Thiesset, T. Ménard, et al. (2021). The importance of the correct computation of the surface curvature is also discussed in Bermejo-Moreno and Pullin (2008); Bermejo-Moreno, Pullin, and Horiuti (2009) in the context of turbulent fields like the enstrophy iso-surfaces or in Evrard (2017); Evrard et al. (2020) for multiphase flows interfaces. It usually leads to some difficulties related to the triangulation of the interface given by a discretized level-set function and so far no numerical framework has been devised in order to cope with such geometrical quantities evaluation in a robust and efficient manner.

## 2.3 Scientific computing context

In the previous introductory paragraphs we already touched the potential high cost of performing two-phase flow simulations. Performing meaningful DNS, for example, might require few hundreds of millions of cells, to say the least. Simulating seconds of physical time requires a high level of data and algorithm parallelism to be feasible, it often translates into programming design driven by these underlying data structures, with APIs that are hardly accessible, with compiled language with complex build systems. In addition, industrial multiphysics simulation codes provide a large plethora of flexible functionalities (for example the CEDRE code (Le Touze 2015; Cordesse 2020), or the OpenFOAM code (Greenshields 2015)) inherited from a long legacy of different implementations. These codes are often organized in very big code bases that are difficult to modify and test. A new generation of codes is currently under heavy development addressing the necessity of local mesh adaptation such as canoP code (Druil, Fikl, et al. 2016), based on the p4est library that offers octrees manipulation, or the SAMURAI code (Bellotti et al. 2021) offering a tree-less datastructure optimized for multiresolution mesh adaptation.

On the other side of the coin, the ever-increasing interest for Machine Learning, Artificial Intelligence, and cryptocurrencies mining, drove the technological advancement of specialized hardware components like GPUs and FPGAs, allowing the usage of dynamical interpreted languages like Python, to perform heavy duty number crunching tasks thanks to their binding APIs that allow the integration of high performance code from compiled libraries in very specific and localized zones of the code base. Libraries like Tensorflow, pyTorch are *de facto* standards in their context, they are optimized to run graph operations on GPUs and allow training on big datasets. The Python ecosystem is rich and offers optimized data structures like NumPy (Harris et al. 2020) that constitutes the baseline on top of which specialized columnar arrays

are based. Examples are Cupy ([Preferred Infrastructure, Inc. 2021](#)), a NumPy implementation on top of GPUs, Dask, a distributed implementation complying with NumPy arrays or Legate NumPy ([Bauer and Garland 2019](#)), a distributed NumPy implementation that allows computing on hybrid configurations on multiple GPUs and CPUs. Another approach to benefit of both the compiled ultra-optimized libraries and the flexible interpreted ecosystems, hence solving the two Language problem ([Perkel 2019](#)), is the development of a new generation numerical language like Julia.

Modern languages offer the tempting possibility of easy prototyping and code refactoring without losing the performance of compiled, specific libraries, tightly integrated within the language runtime. The toolchains associated to those modern languages enforce best practices like [CI](#) and [CD](#), documentation auto-generation, unit testing, and these tools are often part of the standard library and hence easily accessible and taught (as an example of modern practices see [Klabnik and C. Nichols \(2018\)](#)). These features are desirable also in new generation [HPC](#) software.

### 2.4 Contributions of this thesis

---

The principal target of this thesis is to offer a unified modeling framework addressing the derivation of system of equations governing two-phase flow systems characterized by a sound mathematical structure via a variational setting named Stationary Action Principle (SAP). This effort is backed by a tailored computational toolset that allows the rational choice of modeling assumptions and the effective simulations of the developed models, possibly on modern computing architectures. This work aims at dealing with the three main aspects regarding two-phase flow modeling we identified in the previous introductory sections, *i.e.* the development of sounding reduced-order models via the Stationary Action Principle (SAP), the implementation of a geometric [DNS](#) post-processing tool called `Mercur(v)e` that is used to clarify the right assumptions to be made while crafting a reduced-order model, and the development of a Python library named `josiepy` that acts as a playbook to quickly test conservative system models, numerical schemes, boundary conditions, domain configurations and modern approach to scientific computing. The manuscript is organized as follows:

- Chapter [3](#) is dedicated to the discussion about the reduced-order models derivation strategies. We introduce two different methods for the derivation of system of equations regarding two-phase flows, the first one based on the averaging of instantaneous local Navier-Stokes (NS) equations for each phase, the second one that leverages a variational

methodology called the Stationary Action Principle (SAP). Then, we exploit the [SAP](#) to derive a set of models featuring different small scale assumptions with the final aim of injecting governing equations for geometric parameters that could improve the accuracy of the models when dealing with non-spherical inclusions.

- Chapter 4 presents the development efforts infused into the creation of `Mercur(v)e`, a library dedicated to the post-processing of two-phase flow [DNS](#). The library is based on a triangulation of the interface, on which discrete approximation of geometrical parameters like the mean and Gauss curvature are computed. The algorithm chosen to perform those computations preserves topological invariants like the Gauss-Bonnet theorem, allowing to count objects in a domain that are homeomorphic to spheres. Moreover, the library also offers an averaging kernel that allows to smooth noisy computations on dirty triangulations without losing the preservation of the topological invariants. The library is used to back the modeling assumptions for the crafting of a reduced-order model assuming isochoric oscillation of ellipsoidal droplets as small scale model.
- In chapter 5 we discuss the numerical framework on which the library `josiepy` is built. We provide a general overview on the theory for mesh generation and numerical schemes in the context of the [Finite Volume Method \(FVM\)](#), we recast the classical literature on the subject into a different form that contextualize the numerical enunciation of the schemes to the specificities of the code implementation of `josiepy`. We discuss in reasonable details all the models, algorithms and schemes that are already implemented at the current date in `josiepy`. We show a set of verification tests on different models, that are reproducible with the current code available online under a [FOSS](#) license and readily available as Jupyter notebooks or integration tests in the repository holding the source code, including the innovative models derived and presented in chapter 3.
- Chapter 6 outlines all the aspects related to the Software Engineering of the libraries we developed during the progress of the thesis: a Python library named `josiepy` aimed at simulating potentially generic [Partial Differential Equation \(PDE\)](#) systems and the `Mercur(v)e` library the aim of which is to expose a geometrical post-processing procedure to collect insightful information from high-definition simulations like [DNS](#). In the chapter we discuss the philosophy behind the choice of creating such pieces of software, the current challenges and the possible perspectives in the near future.

The subjects we explored in the accomplishment of this thesis lead to the following contributions:

- Journal Articles:
  - Ruben Di Battista, Thibault Ménard, Stephane De Chaisemartin, and Marc Massot (2021). “A Computational Framework Based on the Discrete Estimation of Geometrical Properties over Triangulated Interfaces Preserving Topological Invariants to Design and Validate Two-Phase Flow Models.” In: *Fluids*. In preparation
  - Pierre Cordesse, Ruben Di Battista, Quentin Chevalier, Lionel Matuszewski, Thibault Ménard, Samuel Kokh, and Marc Massot (2020). “A Diffuse Interface Approach For Disperse Two-Phase Flows Involving Dual-Scale Kinematics Of Droplet Deformation Based On Geometrical Variables.” In: *ESAIM: Proceedings*, p. 22. URL: <https://hal.archives-ouvertes.fr/hal-02879950v1>
- Conference Proceedings:
  - Alberto Remigi, Ruben Di Battista, François-Xavier Demoulin, Benjamin Duret, Marc Massot, Thibaut Ménard, and Hugo Deneuille (May 19–24, 2019). “Exploring Different Approaches for the Simulation of Multi-Scale Atomization Process.” In: International Conference on Multiphase Flow. Rio de Janeiro. URL: <https://hal.archives-ouvertes.fr/hal-02379257>
  - Pierre Cordesse, Ruben Di Battista, Samuel Kokh, and Marc Massot (May 19–24, 2019). “Derivation of a Two-Phase Flow Model with Two-Scale Kinematics and Surface Tension by Means of Variational Calculus.” In: 10th International Conference on Multiphase Flow. Rio de Janeiro. URL: <https://hal.archives-ouvertes.fr/hal-02194951>
  - Ruben Di Battista, Iván Bermejo-Moreno, Thibaut Ménard, Stéphane de Chaisemartin, and Marc Massot (May 19–24, 2019). “Post-Processing of Two-Phase DNS Simulations Exploiting Geometrical Features and Topological Invariants to Extract Flow Statistics and Droplets Number Density.” In: International Conference on Multiphase Flow. Rio de Janeiro. URL: <https://hal.archives-ouvertes.fr/hal-02345825v1>
- Book Chapters:
  - Pierre Cordesse, Ruben Di Battista, Florence Drui, Samuel Kokh, and Marc Massot (2020). “Derivation of a Two-Phase Flow Model with Two-Scale Kinematics, Geometric Variables and Surface Tension Using Variational Calculus.” In: *Proceedings of the NASA Summer Program*. Nasa Technical Memorandum. URL: <https://hal.archives-ouvertes.fr/hal-02336996>

- Ruben Di Battista, Iván Bermejo-Moreno, Thibaut Ménard, and Marc Massot (2019). “Geometrical Characterization and DNS Post-Processing of the 3D Objects in Two-Phase Flow: Collision of Two Droplets.” In: *NASA Technical Memorandum*
- Software
  - Ruben Di Battista (2018). *Mercur(v)e — A Library to Exploit Geometrical and Topological Properties to Allow Post-Processing of DNS Simulations (and Much More)*. URL: <https://gitlab.com/rubendibattista/mercurve>
  - Ruben Di Battista (2019). *Josiepy — A 2D PDE Solver Written in Python without Compromising (Too Much) Performance*. URL: <https://gitlab.com/rubendibattista/josiepy>





# 3

## Two-phase flow models and where to find them

The world of mathematical modeling for two-phase flow phenomena is variegated. Different models, of wide range of complexity, are available in the literature. In the context of the separated phase regime, i.e. the regime of the flow in which the two phases are clearly identified by an interface, models spanning from locally averaged Navier-Stokes (NS) equations which often, explicitly or implicitly, assume some sort of equilibrium of the flow phases (related to energy, momentum or pressure equilibrium) to models that relax all the equilibrium assumptions allowing difference in velocity, pressure and temperature between the flow phases such as the Baer-Nunziato (B-N) model are possible. In the case of the disperse phase regime, i.e. the regime of the flow in which a cloud of bubbles or droplets of one phase is immersed in a carrier, Lagrangian or Eulerian kinetic models are used to track the objects and then they are coupled with the previously presented models for the separate phase zone. The Eulerian models, moreover, can be derived exploiting different approaches: averaging, in time, space or across realizations, local instantaneous conservation equations; pre-assuming a conservative form of the averaged equations that is complemented by tailored source terms to enforce the coupling between the phases or also a variational approach that leverages the power of the Stationary Action Principle (SAP).

This chapter is devoted to the presentation of a modeling approach based on the Stationary Action Principle (SAP) for the derivation of a hierarchy of system of equations describing the behavior of two-phase flows. Initially we provide an overview on the different possible approaches for the derivation of two-phase models; we discuss the state of the art regarding the equation for the interfacial area density; then we introduce the general notions constituting the variational framework in which we operate and at the end we present a set of mathematical systems we derived in order to enrich the geometrical description of two-phase flows. In particular, geometrical terms such as curvature and surface density are obtained, widening the allowed solution space of models initially aimed at the separated phase regime to regimes also featuring droplets or bubbles. With the generalized SAP framework we also provide a different point of view on the derivation of models featuring the interfacial area density.

A list of preliminary contributions on this subject:

- Pierre Cordesse, Ruben Di Battista, Samuel Kokh, and Marc Massot (May 19–24, 2019). “Derivation of a Two-Phase Flow Model with Two-Scale Kinematics and Surface Tension by Means of Variational Calculus.” In: 10th International Conference on Multiphase Flow. Rio de Janeiro. URL: <https://hal.archives-ouvertes.fr/hal-02194951>
- Pierre Cordesse, Ruben Di Battista, Florence Drui, Samuel Kokh, and Marc Massot (2020). “Derivation of a Two-Phase Flow Model with Two-Scale Kinematics, Geometric Variables and Surface Tension Using Variational Calculus.” In: Proceedings of the NASA Summer Program. Nasa Technical Memorandum. URL: <https://hal.archives-ouvertes.fr/hal-02336996>
- Pierre Cordesse, Ruben Di Battista, Quentin Chevalier, Lionel Matuszewski, Thibault Ménard, Samuel Kokh, and Marc Massot (2020). “A Diffuse Interface Approach For

### *3 Two-phase flow models and where to find them*

Disperse Two-Phase Flows Involving Dual-Scale Kinematics Of Droplet Deformation Based On Geometrical Variables.” In: ESAIM: Proceedings, p. 22. URL: <https://hal.archives-ouvertes.fr/hal-02879950v1>

## 3.1 Introduction

---

The literature about reduced order two-phase flow modeling does not offer at the current date a satisfactory modeling framework that allows to take into account the intrinsic multiscale nature of this kind of phenomena in a unified fashion. Apart from models inspired by the work of [Cahn and Hilliard \(1958\)](#); [C. Liu and Shen \(2003\)](#), in which the interface is sharply reconstructed, and specific models that address more specifically the dispersed phase regime, *i.e.* the situations in which the domain is occupied by a population of inclusions, like the lagrangian methods ([Tryggvason et al. 2001](#); [James Glimm et al. 2006](#)) or the mesoscopic MoM ([Laurent and Massot 2001](#); [Laurent, Massot, and Villedieu 2004](#); [Kah et al. 2015](#); [Essadki 2018](#)), reduced order models can be effectively derived exploiting an averaging approach of local NS equations. Those “averaged models” can be casted in a generic way for an average operator that can be time, volume or ensemble based. The averaging approach acting on the local equations creates additional terms related to the smallest scales, the closure of which might be challenging and needs to be somehow extrapolated from external inputs like experimental data ([Ishii and Mishima 1984](#); [Ishii, Kim, et al. 2002](#); [Rusche 2003](#); [Drew and Passman 2006](#)). This averaged two-phase models can be related to the averaging strategy that is done in the turbulence framework ([Céasar Dopazo 1977](#)). On the other side, an alternative approach is provided by a variational method called Stationary Action Principle (SAP), in which the small scale hypotheses are assumed via a specific form of a Lagrangian functional, that integrated over the phase space configures a Hamiltonian action. This Hamiltonian action is made stationary, and the process generates the convective subsystem that encapsulates the modeling assumptions. The dissipative nature can be retrieved at will imposing a sign to an entropy function, as the second thermodynamics principle mandates.

In this section we provide a comparison between the two approaches, firstly introducing the fundamentals of the strategy based on the averaging of local instantaneous equations. Then we introduce the SAP tool in details, listing a series of derivation that are then used as building blocks towards a unified framework for modeling two-phase flows. It allows to take into account arbitrary small scale phenomena for two-phase flows. At the end of the chapter we present a hierarchy of systems that are obtained with the SAP, and we also discuss a way of drawing an analogy between the interface area density equations discussed in [Drew \(1990\)](#); [Morel et al. \(1999\)](#); [Daniel Lhuillier \(2004\)](#); [Morel \(2007\)](#) and the variational approach designed with the SAP.

## 3.2

# Deriving two-phase flow models with an averaging procedure

The idea of deriving a two-phase flow model as the result of averaging, independently from the chosen averaging strategy, can be referred to the work of [Ishii and Mishima \(1984\)](#). The presentation of this section aims at providing an adapted context to introduce the averaging approach and it is not by any means exhaustive. The interested reader is referred, in addition to the previously mentioned sources, also to [Candel and Poinso \(1990\)](#); [Morel et al. \(1999\)](#); [Rusche \(2003\)](#); [Delhay \(2013\)](#); [Cesar Dopazo et al. \(2018\)](#) among others.

Everything starts from the assumption that both phases occupy a domain  $\Omega$ , each phase  $k = 1, 2$ , in its own domain  $\Omega_k$ , separated by a weightless interface that has no thickness:

$$\cup_k \overline{\Omega_k} = \overline{\Omega}; \quad \Omega_1 \cap \Omega_2 = \emptyset$$

We define with  $\rho_k, \mathbf{u}_k, E_k$  respectively the density, velocity and total energy of the phase  $k$ . The instantaneous conservation equations (mass, momentum, energy) for a phase  $k$  are:

$$\frac{\partial}{\partial t}(\rho_k \varphi) + \nabla \cdot (\rho_k \varphi \mathbf{u}_k) - \nabla \cdot (\mathbf{G}_k) - \mathbf{s}_k = \mathbf{0}, \quad \mathbf{x} \in \Omega_k \quad (3.1)$$

where the generic terms  $(\varphi, \mathbf{G}_k, \mathbf{s}_k)$  are respectively for the mass, momentum and energy conservation:  $(1, 0, 0)$ ,  $(\mathbf{u}_k, \mathbf{D}_k \triangleq -p_k \mathbf{I} + \mathbf{T}_k, \mathbf{m}_k)$ ,  $(E_k, \mathbf{D}_k \cdot \mathbf{u}_k - \mathbf{v}_k, e_k)$ , in which we neglect mass transfer governed by stiff source terms or mass diffusion.  $p_k$  is the pressure for the phase  $k$ ,  $\mathbf{T}_k$  the viscous tensor,  $\mathbf{v}_k$  the conductive heat flux. Equivalently, we can write this set of equations in a compact, tensorial form, more akin to numerical implementation:

$$\frac{\partial}{\partial t}(\rho_k \mathbf{q}_k) + \nabla \cdot (\rho_k \mathbf{\Pi}_k) - \mathbf{s}_k = \mathbf{0}, \quad \mathbf{x} \in \Omega_k \quad (3.2)$$

where  $\rho_k \mathbf{\Pi}_k = \rho_k \mathbf{u}_k \varphi_k - \mathbf{G}_k$  groups the convective and diffusive tensors,  $\rho_k \mathbf{q}_k = (\rho_k, \rho_k \mathbf{u}_k, \rho_k E_k)$  is the full state of the system and  $\mathbf{s}_k = (0, \mathbf{m}_k, e_k)$  is the the group of all source terms for all the equations in which we assume no mass transfer between phases.

### 3.2.1 Interface between phases and jump conditions

As shown by [Drew \(1990\)](#), we describe the interface that separates the two phases as a weightless, zero-thickness discontinuity surface that is defined by  $\mathcal{S} = \overline{\Omega_1} \cap \overline{\Omega_2}$ . The surface  $\mathcal{S}$  is

supposed to be smooth enough so that geometric characteristics of  $\mathcal{S}$  such as mean and Gaussian curvatures are unambiguously defined. We then introduce the characteristic function:  $\mathbb{1}_{\Omega_1}$

$$\mathbb{1}_{\Omega_1}(\mathbf{x}, t) = \begin{cases} 1, & \mathbf{x} \in \Omega_1, \\ 0, & \text{otherwise.} \end{cases} \quad (3.3)$$

The function  $\mathbb{1}_{\Omega_1}$  satisfies the following eq. (3.4) in the sense of distributions:

$$\frac{\partial \mathbb{1}_{\Omega_1}}{\partial t} + \mathbf{u}^{\mathcal{S}} \cdot \nabla \mathbb{1}_{\Omega_1} = 0 \quad (3.4)$$

where  $\mathbf{u}^{\mathcal{S}}$  is the velocity at which the interface moves in the domain  $\Omega_1$ . The terms  $\nabla \mathbb{1}_{\Omega_1}$  and  $\frac{\partial \mathbb{1}_{\Omega_1}}{\partial t}$  are Dirac measures whose support is the set of  $(\mathbf{x}, t)$  such that  $\mathbf{x}$  belongs to  $\mathcal{S}$  at instant  $t$ . In order to derive jump conditions at the interface, we now write the equations eqs. (3.1) and (3.2) in integral form over the domain  $\Omega$  and we apply some manipulations of the integrals exploiting the divergence theorem, as shown in eq. (3.5):

$$\begin{aligned} \sum_k \int_{\Omega_k} \frac{\partial}{\partial t} (\rho_k \mathbf{q}_k) + \nabla \cdot (\rho_k \mathbf{\Pi}_k) - \mathfrak{J}_k &= \mathbf{0} \\ \sum_k \int_{\Omega_k} \frac{\partial}{\partial t} (\rho_k \mathbf{q}_k) - \mathfrak{J}_k + \oint_{\partial\Omega_k \setminus \mathcal{S}} \rho_k \mathbf{\Pi}_k \cdot \hat{\mathbf{n}}_k + \\ &+ \oint_{\mathcal{S}} \rho_k \mathbf{\Pi}_k \cdot \hat{\mathbf{n}}_k^{\mathcal{S}} - \oint_{\mathcal{S}} \rho_k^{\mathcal{S}} \mathbf{\Pi}_k^{\mathcal{S}} \cdot \hat{\mathbf{n}}_k^{\mathcal{S}} - \mathfrak{J}^{\mathcal{S}} &= \mathbf{0} \\ \sum_k \int_{\Omega_k} \frac{\partial}{\partial t} (\rho_k \mathbf{q}_k) + \nabla \cdot (\rho_k \mathbf{\Pi}_k) - \mathfrak{J}_k + \oint_{\mathcal{S}} (\rho_k \mathbf{\Pi}_k - \rho_k^{\mathcal{S}} \mathbf{\Pi}_k^{\mathcal{S}}) \cdot \hat{\mathbf{n}}_k^{\mathcal{S}} - \mathfrak{J}^{\mathcal{S}} &= \mathbf{0} \end{aligned} \quad (3.5)$$

The terms  $\rho_k \mathbf{\Pi}_k^{\mathcal{S}}$ ,  $\mathfrak{J}^{\mathcal{S}}$ , are respectively the flux at the interface advected by the interface velocity  $\mathbf{u}_k^{\mathcal{S}}$  and the interfacial forces like the surface tension;  $\hat{\mathbf{n}}_k$  is the outward normal to the part of the domain  $\Omega_k$  not at the interface ( $\partial\Omega_k \setminus \mathcal{S}$ ),  $\hat{\mathbf{n}}_k^{\mathcal{S}}$  is the normal at the interface pointing outward of  $\Omega_k$ . We note that the first term of eq. (3.5)  $\partial \rho_k \mathbf{q}_k / \partial t + \nabla \cdot (\rho_k \mathbf{\Pi}_k) - \mathfrak{J}_k = \mathbf{0}$  from eq. (3.2) leads to the jump conditions across the interface  $\mathcal{S}$ ,

$$[(\rho_k \mathbf{\Pi}_k - \rho_k^{\mathcal{S}} \mathbf{\Pi}_k^{\mathcal{S}})]_{\mathcal{S}} \cdot \hat{\mathbf{n}} = \mathfrak{J}^{\mathcal{S}} \quad (3.6)$$

where  $[\bullet]_{\mathcal{S}}$  defines a jump for the field  $\bullet$  across the interface  $\mathcal{S}$ . The normal is chosen  $\hat{\mathbf{n}} \triangleq \hat{\mathbf{n}}_1^{\mathcal{S}}$ . If we unpack the terms  $\mathbf{\Pi}_k$ ,  $\mathbf{\Pi}_k^{\mathcal{S}}$ ,  $\mathfrak{J}^{\mathcal{S}} = (0, \mathbf{m}^{\mathcal{S}}, e^{\mathcal{S}})$  for all the equations eq. (3.1), we obtain a

set of jump equations:

$$\begin{aligned} [\rho_k(\mathbf{u}_k - \mathbf{u}^\delta)]_\mathcal{S} \cdot \hat{\mathbf{n}} &= 0 \\ [\rho_k \mathbf{u}_k \otimes (\mathbf{u}_k - \mathbf{u}^\delta) + \mathbf{D}_k]_\mathcal{S} \cdot \hat{\mathbf{n}} &= \mathbf{m}^\delta \\ [\rho_k E_k(\mathbf{u}_k - \mathbf{u}^\delta) + \mathbf{D}_k \cdot \mathbf{u}_k - \mathbf{v}_k]_\mathcal{S} \cdot \hat{\mathbf{n}} &= e^\delta \end{aligned} \quad (3.7)$$

being  $\mathbf{m}^\delta$  the interface surface forces and  $e^\delta$  the surface energy contributions. We will not detail further those terms, the interested reader can certainly refer to [Drew \(1990\)](#); [Drew and Passman \(2006\)](#). We just highlight that if we neglect the viscous effects, and we consider the surface tension as interface surface force  $\mathbf{m}^\delta = \sigma H \hat{\mathbf{n}}$ , being  $H$  the local mean curvature of the interface, and the two phases are at velocity equilibrium  $(\mathbf{u}_k - \mathbf{u}^\delta) = \mathbf{0}$ , the jump condition for the momentum equations provides the Young-Laplace relation for the surface tension:

$$[p]_\mathcal{S} = \sigma H$$

#### 3.2.2 Averaged governing equations

Taking into account the definition of the characteristic function eq. (3.3), we can multiply the governing equations eq. (3.2) by  $\mathbb{1}_{\Omega_k}$  in order to extend their validity on the entire domain  $\Omega$ :

$$\frac{\partial}{\partial t} \left( \mathbb{1}_{\Omega_k} \rho_k \mathbf{q}_k \right) + \nabla \cdot \left( \mathbb{1}_{\Omega_k} \rho_k \mathbf{\Pi}_k \right) - \mathbb{1}_{\Omega_k} \mathcal{J}_k = (\rho_k \mathbf{\Pi}_k - \mathbf{u}^\delta \otimes \rho_k \mathbf{q}_k) \cdot \nabla \mathbb{1}_{\Omega_k} \quad (3.8)$$

We now introduce an averaging operator  $\langle \bullet \rangle$ . It can be thought as a time, space or ensemble average, and it satisfies the following properties (usual for classical choices of the average operator):

- *Linearity*

$$\langle \lambda \bullet + \odot \rangle = \lambda \langle \bullet \rangle + \langle \odot \rangle \quad (3.9)$$

- *Idempotency*

$$\langle \langle \bullet \rangle \rangle = \langle \bullet \rangle \quad (3.10)$$

- Gauss and Leibniz rules

$$\frac{\partial \langle \bullet \rangle}{\partial t} = \left\langle \frac{\partial \bullet}{\partial t} \right\rangle, \quad \frac{\partial \langle \bullet \rangle}{\partial \mathbf{x}} = \left\langle \frac{\partial \bullet}{\partial \mathbf{x}} \right\rangle \quad (3.11)$$

We introduce the fluctuation  $(\bullet)'$  defined by

$$\bullet = \langle \bullet \rangle + (\bullet)' \quad (3.12)$$

Then we define

- *Volume Fraction*

$$\alpha_k = \langle \mathbb{1}_{\Omega_k} \rangle \quad (3.13)$$

- *Phase average*

$$\bar{\bullet} = \frac{\langle \bullet \mathbb{1}_{\Omega_k} \rangle}{\alpha_k} \quad (3.14)$$

- *Favre average*

$$\tilde{\bullet} = \frac{\langle \mathbb{1}_{\Omega_k} \rho_k \bullet \rangle}{\alpha_k \bar{\rho}_k} \quad (3.15)$$

we obtain the averaged equations,

$$\frac{\partial}{\partial t}(\alpha_k \bar{\rho}_k \tilde{\mathbf{q}}_k) + \nabla \cdot (\alpha_k \bar{\rho}_k \tilde{\mathbf{\Pi}}_k) - \alpha_k \bar{\mathbf{J}}_k = \langle (\mathbf{\Pi}_k - \mathbf{u}^\mathcal{S} \otimes \mathbf{q}_k) \cdot \nabla \mathbb{1}_k \rangle \quad (3.16)$$

The compact notation of eq. (3.16) hides a lot of complexity associated to the terms  $\tilde{\mathbf{\Pi}}$ , whose average decomposition produces unclosed terms, and  $\langle (\mathbf{\Pi}_k - \mathbf{u}^\mathcal{S} \otimes \mathbf{q}_k) \cdot \nabla \mathbb{1}_k \rangle$  that also is unclosed. For clarity, unpacking eq. (3.16) in the set of equations corresponding to the mass, momentum, and energy conservation we retrieve:

$$\frac{\partial}{\partial t}(\alpha_k \bar{\rho}_k) + \nabla \cdot (\alpha_k \bar{\rho}_k \tilde{\mathbf{u}}_k) = \Gamma_k \quad (3.17a)$$

$$\frac{\partial}{\partial t}(\alpha_k \bar{\rho}_k \tilde{\mathbf{u}}_k) + \nabla \cdot \left[ \alpha_k \left( \bar{\rho}_k \tilde{\mathbf{u}}_k \otimes \tilde{\mathbf{u}}_k - \tilde{\mathbf{D}}_k \right) \right] - \alpha_k \bar{\mathbf{m}}_k = \mathbf{m}_k \quad (3.17b)$$

$$\frac{\partial}{\partial t}(\alpha_k \bar{\rho}_k \tilde{E}_k) + \nabla \cdot \left[ \alpha_k \left( \bar{\rho}_k \tilde{E}_k \mathbf{I} - \tilde{\mathbf{D}}_k \right) \cdot \tilde{\mathbf{u}}_k - \tilde{\mathbf{v}}_k \right] - \alpha_k \bar{\mathcal{E}}_k = \mathfrak{D}_k \quad (3.17c)$$

### 3 Two-phase flow models and where to find them

Let us note again  $\tilde{\mathbf{D}}_k = -\tilde{p}_k \mathbf{I} + \tilde{\mathbf{T}}_k$ . The source terms  $\Gamma$ ,  $\mathbf{m}$ ,  $\mathfrak{D}$  contain all the additional terms that are created by the averaging process,

$$\Gamma_k = \left\langle \rho_k (\mathbf{u}_k - \mathbf{u}^\mathcal{S}) \cdot \nabla (\mathbb{1}_{\Omega_k}) \right\rangle \quad (3.18a)$$

$$\mathbf{m}_k = \left\langle [\rho_k (\mathbf{u}_k - \mathbf{u}^\mathcal{S}) \otimes \mathbf{u}_k] \cdot \nabla (\mathbb{1}_{\Omega_k}) \right\rangle + \left\langle p_k \nabla (\mathbb{1}_{\Omega_k}) \right\rangle - \left\langle \mathbf{T}_k \cdot \nabla (\mathbb{1}_{\Omega_k}) \right\rangle \quad (3.18b)$$

$$\begin{aligned} \mathfrak{D}_k = & \left\langle \rho_k E_k (\mathbf{u}_k - \mathbf{u}^\mathcal{S}) \cdot \nabla (\mathbb{1}_{\Omega_k}) \right\rangle + \left\langle (p_k \mathbf{u}_k) \cdot \nabla (\mathbb{1}_{\Omega_k}) \right\rangle \\ & - \left\langle (\mathbf{T}_k \cdot \mathbf{u}_k) \cdot \nabla (\mathbb{1}_{\Omega_k}) \right\rangle - \left\langle \mathbf{v}_k \cdot \nabla (\mathbb{1}_{\Omega_k}) \right\rangle \end{aligned} \quad (3.18c)$$

Following the definitions and the decomposition presented in [Drew and Passman \(2006\)](#); [Cordesse \(2020\)](#), the averaged momentum flux due to mass transport at the interface can be written as:

$$\Gamma_k \langle \mathbf{u}^\mathcal{S} \rangle \triangleq \left\langle [\rho_k (\mathbf{u}_k - \mathbf{u}^\mathcal{S}) \otimes \mathbf{u}_k] \cdot \nabla (\mathbb{1}_{\Omega_k}) \right\rangle \quad (3.19)$$

and in turn the averaged momentum source term due to the interface pressure,

$$\left\langle p_k \nabla (\mathbb{1}_{\Omega_k}) \right\rangle = \langle p_k^\mathcal{S} \rangle \nabla \alpha_k + \left\langle (p_k)' \nabla \mathbb{1}_{\Omega_k} \right\rangle \quad (3.20)$$

The interfacial pressure  $p_k^\mathcal{S}$  is decomposed in a constant and a fluctuating contribution,

$$p_k^\mathcal{S} = \langle p_k^\mathcal{S} \rangle + (p_k^\mathcal{S})'$$

such that we can provide a definition for the averaged interface pressure,

$$\langle p_k^\mathcal{S} \rangle \nabla \alpha_k \triangleq \left\langle p_k \nabla \mathbb{1}_{\Omega_k} \right\rangle - \left\langle (p_k)' \nabla \mathbb{1}_{\Omega_k} \right\rangle \quad (3.21)$$

with the definition eq. (3.20), we can also define an average interfacial force density,

$$\mathbf{m}_k^\mathcal{S} \triangleq \left\langle ((p_k)' \mathbf{I} - \mathbf{T}) \cdot \nabla \mathbb{1}_{\Omega_k} \right\rangle \quad (3.22)$$

that is often modeled as a drag term ([Rusche 2003](#)). The average energy flux at the interface due to mass transport at the interface is defined as:

$$\Gamma_k \langle E_k^\mathcal{S} \rangle \triangleq \left\langle \rho_k E_k (\mathbf{u}_k - \mathbf{u}^\mathcal{S}) \cdot \nabla (\mathbb{1}_{\Omega_k}) \right\rangle \quad (3.23)$$



The averaged energy source term due to the interface pressure, with the additional assumption of neglecting velocity fluctuations, is:

$$\left\langle (p_k \mathbf{u}_k) \cdot \nabla (\mathbb{1}_{\Omega_k}) \right\rangle = (\langle p_k^\mathcal{S} \rangle \langle \mathbf{u}^\mathcal{S} \rangle) \cdot \nabla \alpha_k + \left\langle (p)_k' \nabla \mathbb{1}_{\Omega_k} \right\rangle \cdot \langle \mathbf{u}^\mathcal{S} \rangle \quad (3.24)$$

that allows the definition of averaged interfacial energy source,

$$\mathfrak{S}_k^\mathcal{S} \triangleq \left\langle ((p_k)' \mathbf{I} - \mathbf{T}) \cdot \nabla \mathbb{1}_{\Omega_k} \right\rangle \cdot \langle \mathbf{u}^\mathcal{S} \rangle = \mathbf{m}_k^\mathcal{S} \cdot \langle \mathbf{u}^\mathcal{S} \rangle \quad (3.25)$$

The final form of the set of averaged equations, hence is:

$$\frac{\partial}{\partial t} (\alpha_k \bar{\rho}_k) + \nabla \cdot (\alpha_k \bar{\rho}_k \tilde{\mathbf{u}}_k) = \Gamma_k \quad (3.26a)$$

$$\begin{aligned} \frac{\partial}{\partial t} (\alpha_k \bar{\rho}_k \tilde{\mathbf{u}}_k) + \nabla \cdot \left[ \alpha_k \left( \bar{\rho}_k \tilde{\mathbf{u}}_k \otimes \tilde{\mathbf{u}}_k - \tilde{\mathbf{D}}_k \right) \right] &= \Gamma_k \langle \mathbf{u}^\mathcal{S} \rangle + \langle p_k^\mathcal{S} \rangle \nabla \alpha_k \\ &+ \mathbf{m}_k^\mathcal{S} + \alpha_k \overline{\mathbf{m}}_k \end{aligned} \quad (3.26b)$$

$$\begin{aligned} \frac{\partial}{\partial t} (\alpha_k \bar{\rho}_k \tilde{E}_k) + \nabla \cdot \left[ \alpha_k \left( \bar{\rho}_k \tilde{E}_k \mathbf{I} - \tilde{\mathbf{D}}_k \right) \cdot \tilde{\mathbf{u}}_k - \tilde{\mathbf{v}}_k \right] &= \Gamma_k \langle E_k^\mathcal{S} \rangle \\ &+ (\langle p_k^\mathcal{S} \rangle \langle \mathbf{u}^\mathcal{S} \rangle) \cdot \nabla \alpha_k \\ &+ \mathfrak{S}_k^\mathcal{S} + \alpha_k \bar{e}_k \end{aligned} \quad (3.26c)$$

### 3.2.2.1 Closure for the interfacial quantities

If we carefully inspect eq. (3.26), we note that many interfacial quantities are still unclosed. Several authors have proposed different possible definitions for these terms as Coquel, Gallouët, et al. (2002); Guillemaud (2007); D. Lhuillier et al. (2013); Furfaro and Saurel (2015). Let us underline that the choice of the closure may have an important impact on the structure of the model like enabling a positive entropy production term in the (mathematical) entropy evolution equation, see for example Coquel, Gallouët, et al. (2002); Guillemaud (2007); Cordesse and Massot (2020). We will cite an example inspired by Baer and Nunziato (1986) in which:

- The interfacial velocity is taken as the velocity of the phase  $k = 1$

$$\langle \mathbf{u}^\mathcal{S} \rangle \triangleq \mathbf{u}_1 \quad (3.27)$$

- The interfacial pressures are modeled as a single interface pressure:

$$\langle p_1^\mathcal{S} \rangle = \langle p_2^\mathcal{S} \rangle \triangleq p^\mathcal{S} \quad (3.28)$$

whose value is taken as the pressure of the phase  $k = 2$

$$p^\mathcal{S} = p_2 \quad (3.29)$$

other modeling choices are possible, we refer the interested reader to the previously mentioned references. In particular Cordesse (2020); Cordesse and Massot (2020) presents the impact of the closure choice for the interfacial pressure and velocity *w.r.t.* the hyperbolicity of the system, its mathematical structure and numerical performances. They also introduce an advanced thermodynamics for the mixture with velocity non-equilibrium.

#### 3.2.2.2 The volume fraction transport equation

Let us now work on the transport equation for the characteristic function  $\mathbb{1}_{\Omega_k}$  eq. (3.4) that we recall below for the sake of readability,

$$\frac{\partial \mathbb{1}_{\Omega_1}}{\partial t} + \mathbf{u}^\mathcal{S} \cdot \nabla \mathbb{1}_{\Omega_1} = 0.$$

We can follow the same averaging approach we followed to obtain eq. (3.26),

$$\frac{\partial}{\partial t} \left( \left\langle \mathbb{1}_{\Omega_1} \right\rangle \right) + \left\langle \mathbf{u}^\mathcal{S} \cdot \nabla \mathbb{1}_{\Omega_1} \right\rangle = 0 \quad (3.30)$$

taking into consideration the definition of the volume fraction  $\alpha_k = \left\langle \mathbb{1}_{\Omega_k} \right\rangle$  at eq. (3.13), we can define a source term for the volume fraction equation that needs modeling effort:

$$A \triangleq \left\langle \mathbf{u}^\mathcal{S} \cdot \nabla \mathbb{1}_{\Omega_1} \right\rangle - \left\langle \mathbf{u}^\mathcal{S} \right\rangle \cdot \left\langle \nabla \mathbb{1}_{\Omega_1} \right\rangle \quad (3.31)$$

Since the average operator respects eq. (3.11), we can move the gradient operator out of the average and therefore we obtain a transport equation for the volume fraction  $\alpha \triangleq \alpha_1$ ,

$$\frac{\partial \alpha}{\partial t} + \mathbf{u}^\mathcal{S} \cdot \nabla \alpha = A \quad (3.32)$$

The value of the volume fraction field for the other phase can then be retrieved algebraically,

$$\alpha_2 = 1 - \alpha \quad (3.33)$$

Equation (3.32) provides a description of the interface evolution in terms of how much of the total local volume at  $(\mathbf{x}, t)$  in the domain  $\Omega$  is occupied by a phase if we interpret the averaging operator  $\langle \bullet \rangle$  as a volume average; or, in terms of how often a phase  $k$  occupies a certain zone  $(\mathbf{x}, t)$  of the domain  $\Omega$  if instead  $\langle \bullet \rangle$  is interpreted as an ensemble averaging. This volume ratio of occupation or presence probability is then the volume fraction  $\alpha(\mathbf{x}, t)$ . Another alternative strategy to recast eq. (3.30) can be found in Drew (1990), where the interfacial velocity is assumed to be normal to the interface,

$$\mathbf{u}^\mathcal{S} = u^\mathcal{S} \hat{\mathbf{n}} \quad (3.34)$$

If we now define, complementing the definitions eqs. (3.13) to (3.15), the *interface average*

$$\hat{\bullet} = \frac{\langle \bullet \nabla \mathbb{1}_\Omega \cdot \hat{\mathbf{n}} \rangle}{\Sigma} = \frac{\left\langle \bullet \frac{\partial \mathbb{1}_\Omega}{\partial \hat{\mathbf{n}}} \right\rangle}{\Sigma} \quad (3.35)$$

then the averaged advection velocity of the interface is written as,

$$\left\langle \mathbf{u}^\mathcal{S} \cdot \nabla \mathbb{1}_{\Omega_1} \right\rangle = \left\langle u^\mathcal{S} \frac{\partial \mathbb{1}_{\Omega_1}}{\partial \hat{\mathbf{n}}} \right\rangle = \hat{u}^\mathcal{S} \Sigma \quad (3.36)$$

where

$$\Sigma = \left\langle \hat{\mathbf{n}}^\mathcal{S} \cdot \nabla \mathbb{1}_{\Omega_1} \right\rangle \quad (3.37)$$

is the interfacial area density, *i.e.* the amount of area per unit volume of the interface. These assumption provide the following equation for the transport of the volume fraction,

$$\frac{\partial \alpha}{\partial t} + \hat{u}^\mathcal{S} \Sigma = 0 \quad (3.38)$$

This version of the equation for the volume fraction is not often used. It is more common to derive a direct equation for the transport of the interfacial area density that is then associated to an equation for the volume fraction of the form given by eq. (3.32). The governing equation for the interfacial area density can also be retrieved using an averaging procedure of the local instantaneous equations of the characteristic function. We will study this derivation in the next section.

### 3.2.2.3 The interfacial area density equation

Two-phase flow systems are intimately related to the evolution of the interface that separates the two phases, irrespective of the flow regime, separate or disperse. The interface is assumed, as we already did all along the entirety of this section dedicated to the derivation of the averaged equations, as a weightless surface  $\mathcal{S}$  that moves with velocity  $\mathbf{u}^\mathcal{S}$ .

More specifically, the geometric evolution of the interface and its perturbation causes the creation of satellite objects, ligaments and droplets and its correct modeling is of major importance for the correct prediction of an injection system. The spray is often described in terms of diameter distribution of the droplets composing it (Lefebvre and McDonell 2017). In eq. (3.37) we introduced a definition for the interfacial area density as given by Drew (1990). It is also possible to derive a governing equation for  $\Sigma$  based on the same averaging strategy we discussed in section 3.2.2, applied to the conservation equation of the quantity  $\hat{\mathbf{n}} \cdot \nabla \mathbb{1}_{\Omega_1}$  (eq. (3.39)). We have indeed:

$$\begin{aligned} \frac{\partial}{\partial t} (\hat{\mathbf{n}} \cdot \nabla \mathbb{1}_{\Omega_1}) + \nabla \cdot [(\mathbf{u}^\mathcal{S} \otimes \hat{\mathbf{n}}) \cdot \nabla \mathbb{1}_{\Omega_1}] &= \left[ \frac{\partial \hat{\mathbf{n}}}{\partial t} + \nabla \cdot (\mathbf{u}^\mathcal{S} \otimes \hat{\mathbf{n}}) \right] \cdot \nabla \mathbb{1}_{\Omega_1} \\ &+ \hat{\mathbf{n}} \cdot \left[ \frac{\partial}{\partial t} (\nabla \mathbb{1}_{\Omega_1}) + \mathbf{u}^\mathcal{S} \cdot \nabla (\nabla \mathbb{1}_{\Omega_1}) \right] \end{aligned} \quad (3.39)$$

After some manipulations and the averaging (explained in Drew (1990)), eq. (3.39) yields

$$\frac{\partial \Sigma}{\partial t} + \nabla \cdot (\mathbf{u}^\mathcal{S} \Sigma) = u^\mathcal{S} \hat{H} \Sigma \quad (3.40)$$

We mention that eq. (3.40) is not sufficient to model the evolution for “anisotropic” interfaces; equations for the Gauss  $G$  and Mean  $H$  curvatures are also needed (and they are provided again in Drew (1990)). A more general equation for the “anisotropy interface tensor” that takes into account the interface anisotropy can be also be obtained, see Daniel Lhuillier (2004); Morel (2015). Other valuable works that explain the derivation of the interfacial area density equation are available in literature — without the ambition of being exhaustive — in Candel and Poinso (1990); Morel et al. (1999); Vallet and Borghi (1999); Delhay (2001b); Delhay (2001a); Jay et al. (2006) or also more recently Cesar Dopazo et al. (2018), with thorough reviews available in Devassy (2014); Morel (2015). Many different evolution equations for  $\Sigma$  have been proposed in the literature. We will see in the section section 3.6 that it is possible to encompass a large range of possible governing equations for  $\Sigma$  that are compatible with the dissipative structure of a two-phase models through an entropy evolution equation.

## 3.3

## Deriving two-phase flow models with a variational approach

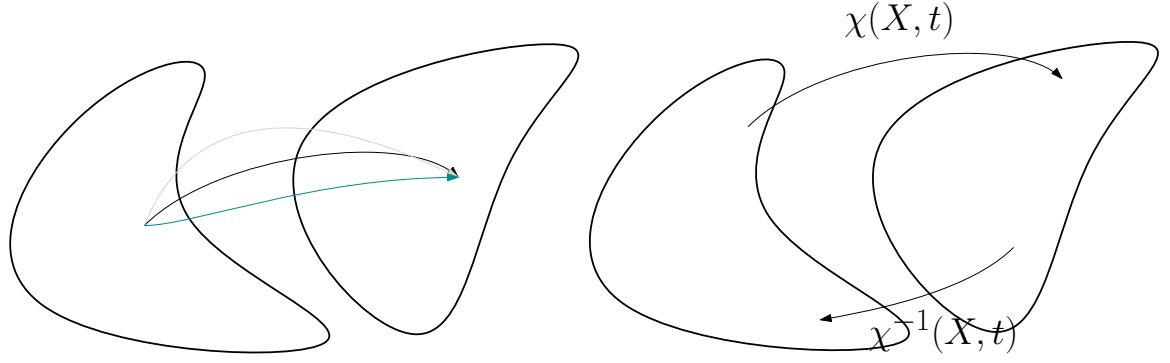
In section 3.2 we discussed a method to derive system as a sort of “bottom-up” approach: we start from the local instantaneous equations that describe the behavior of a single phase in its domain of presence, then, through averaging, a set of equations describing the behavior of the two interacting phases is obtained in an average sense. Several terms come out of the averaging process that need to be modeled, often on a case-by-case approach.

In this section we describe instead an alternative approach, conceptually opposed to the previous one, based in some sort on a “top-bottom” approach that leverages the Stationary Action Principle (SAP): the first step is to set up the assumptions that govern the phenomenon we aim at representing, in particular the type and form of the kinetic and potential energies. Those energies are casted in to the form of a functional named Lagrangian  $\mathcal{L}$  whose integral over the space-time constitutes the “Hamiltonian Action” of the system. The system, then, evolves from a reference configuration to the actual configuration at  $(\mathbf{x}, t)$  along infinite possible trajectories (see Fig. 3.1a). The SAP postulates that the physical one (Fig. 3.1b) optimizes the Hamiltonian Action. Hence, enforcing the Action to be stationary, provides a set of equations that express the conservation of some physical quantities like momentum or energy. Those conservative equations, though, still lack dissipative behavior that can be retrieved applying the second principle of thermodynamics and ensuring that (mathematical) entropy decreases globally. Important references on this kind of approach are Bedford (1985); S. Gavriluk and Gouin (1999); S. Gavriluk and Saurel (2002); Berdichevsky (2009); Gouin and Ruggeri (2009).

### 3.3.1 Variational principles for continua

Let us consider a body that occupies a portion of space  $B \subset \mathbb{R}^3$  at the instant  $t = 0$ . The system of coordinates of the points  $\mathbf{X} = (X_1, X_2, X_3)$  that belong to  $B$  at the instant  $t = 0$  can also be considered as a labeling system for the particles that compose  $B$ . At each instant  $t > 0$ , we note  $\chi(\mathbf{X}, t) \in \mathbb{R}^3$  the position of the particle  $\mathbf{X} \in B$ . This allows to define a mapping

$$\chi : (\mathbf{X}, t) \in B \times [0, \infty) \mapsto \chi(\mathbf{X}, t) \in \mathbb{R}^3 \quad (3.41)$$



(a) A family of trajectories mapping a reference configuration to the system configuration at  $(\mathbf{x}, t)$  (b) A generic mapping from the reference configuration to the system configuration at  $(\mathbf{x}, t)$

Figure 3.1: Evolution of the configuration of a system from a reference configuration

that we suppose to be a diffeomorphism so that we can consider the smooth inverse mapping

$$(\mathbf{x}, t) \in \mathbb{R}^3 \times [0, \infty) \mapsto \chi^{-1}(\mathbf{x}, t) \in B \quad (3.42)$$

A point  $\mathbf{X} \in B$ , is called a Lagrangian coordinate of a point of  $B$  and if  $\mathbf{x} \in \mathbb{R}^3$  is a position of space that is occupied by a point of the body at the instant  $t$  it will be called the Eulerian coordinate. Let us consider a fluid parameter that is associated with a field, for example a scalar field  $\varphi$ . This field can be associated with a mapping noted

$$\varphi^L : (\mathbf{X}, t) \in B \times [0, \infty) \mapsto \varphi^L(\mathbf{X}, t) \quad (3.43)$$

so that  $\varphi^L(\mathbf{X}, t)$  is the value of the parameter  $\varphi$  at point whose initial coordinates are  $\mathbf{X}$  and at instant  $t > 0$ . The mapping  $\varphi^L$  is called the Lagrangian mapping associated with  $\varphi$ . If  $\mathbf{x} \in \mathbb{R}^3$  is a position occupied by the body at an instant  $t > 0$ , we will note  $\varphi(\mathbf{x}, t)$  the value of the fluid parameter at  $\mathbf{x}$  and at the instant  $t > 0$ . This allows to define the so-called Eulerian mapping:

$$\varphi : (\mathbf{x}, t) \in \mathbb{R}^3 \times [0, \infty) \mapsto \varphi(\mathbf{x}, t) \quad (3.44)$$

Thanks to the definition of  $\chi$ , both representations are linked by the following relations:

$$\begin{aligned} \varphi^L(\mathbf{X}, t) &= \varphi(\chi(\mathbf{X}, t), t) \\ \varphi^L(\chi^{-1}(\mathbf{x}, t), t) &= \varphi(\mathbf{x}, t) \end{aligned} \quad (3.45)$$

We define the Lagrangian velocity as the time rate of the mapping  $\chi$ :

$$\mathbf{u}^L(\mathbf{X}, t) \triangleq \left( \frac{\partial \chi}{\partial t} \right)(\mathbf{X}, t) \quad (3.46)$$

This yields the definition of the Eulerian velocity

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{u}^L(\chi^{-1}(\mathbf{x}, t), t) = \left( \frac{\partial \chi}{\partial t} \right)(\chi^{-1}(\mathbf{x}, t), t) \quad (3.47)$$

We can now express the variation in time of any scalar  $\varphi$ . We have indeed that:

$$\frac{\partial \varphi^L}{\partial t}(\mathbf{X}, t) = \frac{\partial}{\partial t}(\varphi(\chi(\mathbf{X}, t), t)) \Big|_{\mathbf{X}} = \frac{\partial \varphi}{\partial t} + \frac{\partial \varphi}{\partial \mathbf{x}} \cdot \frac{\partial \chi}{\partial t} = \frac{\partial \varphi}{\partial t} + \mathbf{u} \cdot \nabla \varphi \quad (3.48)$$

This suggests the definition of the *material derivative*

$$\frac{D\varphi}{Dt}(\mathbf{x}, t) = \frac{\partial \varphi}{\partial t}(\mathbf{x}, t) + \mathbf{u}(\mathbf{x}, t) \cdot \nabla \varphi(\mathbf{x}, t) \quad (3.49)$$

so that we have

$$\frac{\partial \varphi^L}{\partial t}(\mathbf{X}, t) = \frac{D\varphi}{Dt}(\mathbf{x}, t), \quad \mathbf{x} = \chi(\mathbf{X}, t) \quad (3.50)$$

Let us now turn to the evaluation of quantities that evaluate the deformation of the body. The deformation tensor is defined as the rate of change of the configuration motion at time  $t$  w.r.t. the reference configuration  $\mathbf{X}$ :

$$\mathbf{F}(\mathbf{X}, t) = \frac{\partial \chi}{\partial \mathbf{X}}(\mathbf{X}, t) \quad (3.51)$$

We note  $J$  the determinant of  $\mathbf{F}$

$$J(\mathbf{X}, t) = \det(\mathbf{F}(\mathbf{X}, t)) \quad (3.52)$$

If we consider two points with coordinates  $\mathbf{X}$ ,  $\mathbf{X} + d\mathbf{X}$  in the configuration at time  $t = 0$ , they have a relative position at instant  $t$  that is:

$$d\mathbf{x}(\mathbf{X}, t) = \chi(\mathbf{X} + d\mathbf{X}, t) - \chi(\mathbf{X}, t) = \frac{\partial \chi}{\partial \mathbf{X}}(\mathbf{X}, t) d\mathbf{X} = \mathbf{F}(\mathbf{X}, t) d\mathbf{X} \quad (3.53)$$

Using eq. (3.53), we can compute the length of the relative position vector:

$$|d\mathbf{x}|^2(\mathbf{X}, t) = d\mathbf{x}^T d\mathbf{x} = d\mathbf{X}^T \mathbf{F}(\mathbf{X}, t)^T \mathbf{F}(\mathbf{X}, t) d\mathbf{X} \quad (3.54)$$

### 3 Two-phase flow models and where to find them

and the local deformation factor is computed as:

$$|\mathrm{d}\mathbf{x}|^2(\mathbf{X}, t) - |\mathrm{d}\mathbf{X}|^2 = \mathrm{d}\mathbf{X}^T \left( \mathbf{F}(\mathbf{X}, t)^T \mathbf{F}(\mathbf{X}, t) - \mathbf{I} \right) \mathrm{d}\mathbf{X} \quad (3.55)$$

We can also define the displacement

$$\mathbf{d}(\mathbf{X}, t) = \chi(\mathbf{X}, t) - \mathbf{X} \quad (3.56)$$

and its gradient *w.r.t.* the reference configuration:

$$\nabla_{\mathbf{X}}(\mathbf{d})(\mathbf{X}, t) = \frac{\partial \mathbf{d}}{\partial \mathbf{X}}(\mathbf{X}, t) = \frac{\partial \chi}{\partial \mathbf{X}} - \frac{\partial \mathbf{X}}{\partial \mathbf{X}} = \mathbf{F}(\mathbf{X}, t) - \mathbf{I} \quad (3.57)$$

In the next section, we will see how the above elements can be used to evaluation the volume and surface measures in both Lagrangian and Eulerian coordinates.

#### 3.3.1.1 Variations for a family of trajectories

We will now take into consideration different trajectories induced by a family of mappings  $\chi^\epsilon$  that are parameterized by  $\epsilon \in (-1, 1)$  as depicted in Fig. 3.1a

$$\chi^\epsilon(\mathbf{X}, t) = \chi(\mathbf{X}, t) + \epsilon \eta^L(\mathbf{X}, t) \quad (3.58)$$

so that we have

$$\chi^{\epsilon=0}(\mathbf{X}, t) = \chi(\mathbf{X}, t) \quad (3.59)$$

Let us now note

$$\eta^L(\mathbf{X}, t) = \eta(\mathbf{x}, t), \quad \mathbf{x} = \chi(\mathbf{X}, t) \quad (3.60)$$

If  $\varphi$  is a fluid parameter we also consider a perturbed Lagrangian field  $\varphi^{\epsilon L}$  and a perturbed Eulerian field  $\varphi^\epsilon$  such that

$$(\varphi^L)^{\epsilon=0}(\mathbf{X}, t) = \varphi^L(\mathbf{X}, t), \quad \varphi^{\epsilon=0}(\mathbf{x}, t) = \varphi(\mathbf{x}, t) \quad (3.61)$$

These mappings are related by

$$\varphi^{\epsilon L}(\mathbf{X}, t) = \varphi^\epsilon\left(\chi^\epsilon(\mathbf{X}, t), t\right) \quad (3.62)$$



Let us introduce the following variational derivatives

$$\delta\varphi^L(\mathbf{X}, t) = \lim_{\epsilon \rightarrow 0} \frac{\partial \varphi^{\epsilon L}}{\partial \epsilon}(\mathbf{X}, t) \quad \text{and} \quad \delta\varphi(\mathbf{x}, t) = \lim_{\epsilon \rightarrow 0} \frac{\partial \varphi^{\epsilon}}{\partial \epsilon}(\mathbf{x}, t) \quad (3.63)$$

By definition eq. (3.58) we immediately have

$$\delta\chi(\mathbf{X}, t) = \eta^L(\mathbf{X}, t) \quad (3.64)$$

Using eq. (3.63) taking the derivative w.r.t.  $\epsilon$ , we obtain

$$\frac{\partial \varphi^{\epsilon L}}{\partial \epsilon}(\mathbf{X}, t) = \frac{\partial \varphi^{\epsilon}}{\partial \epsilon}(\chi(\mathbf{X}, t)) + \frac{\partial \varphi^{\epsilon}}{\partial \mathbf{x}}(\chi(\mathbf{X}, t), t) \cdot \frac{\partial \chi^{\epsilon}}{\partial \epsilon}(\mathbf{X}, t) \quad (3.65)$$

so that we can connect both variational derivatives by

$$\delta\varphi^L(\mathbf{X}, t) = \delta\varphi(\mathbf{x}, t) + \nabla\varphi(\mathbf{x}, t) \cdot \eta(\mathbf{x}, t), \quad \mathbf{x} = \chi(\mathbf{X}, t) \quad (3.66)$$

Finally, we postulate that the perturbation of all quantities vanish at the boundary of the domain and at both the initial and final instants, more specifically we suppose that:

- $\eta^L(\mathbf{X}, t_1) = \eta^L(\mathbf{X}, t_2) = \mathbf{0}$ , for all  $\mathbf{X} \in B$ ,
- $\eta^L(\mathbf{X}, t) = \mathbf{0}$  for  $\mathbf{X} \in \partial B(t)$  and all  $t \in [t_1, t_2]$ ,
- $\delta\varphi^L(\mathbf{X}, t_1) = \delta\varphi^L(\mathbf{X}, t_2) = \mathbf{0}$ , for all  $\mathbf{X} \in B$ ,
- $\delta\varphi^L(\mathbf{X}, t) = \mathbf{0}$  for  $\mathbf{X} \in \partial B(t)$  and all  $t \in [t_1, t_2]$ .

In the next section, we will see how these variations come into play in the **SAP** for deriving flow models.

## 3.4

## Applying the Stationary Action Principle to a fluid problem

Let us briefly sketch here the main steps of the **SAP** in the context of our two-phase flow study. This modeling process can be divided in three steps that are inspired by Bedford (1985); S. Gavriluk and Gouin (1999); S. Gavriluk and Saurel (2002); Berdichevsky (2009); Gouin and Ruggeri (2009); Drui, Larat, et al. (2019). First, in order to the derive a system of equations

### 3 Two-phase flow models and where to find them

describing the behavior of a fluid using the [SAP](#), we have first to equip our system with a Lagrangian energy that is defined as:

$$\mathcal{L} = \mathcal{K} - \mathcal{U} \quad (3.67)$$

where  $\mathcal{K}$  is the kinetic energy of the fluid,  $\mathcal{U}$  is its potential energy that are associated with the system. We shall assume the following functional dependency of the Lagrangian  $\mathcal{L}$  *w.r.t.* the variables of a system:

$$\mathcal{L} = \mathcal{L}(\boldsymbol{\varsigma}), \quad \boldsymbol{\varsigma} = (\rho, \mathbf{u}, \alpha, D_t \alpha, \varphi_1, \dots, \varphi_{\mathcal{N}_{\text{param}}}) \quad (3.68)$$

where  $\rho, \mathbf{u}, \alpha$  are respectively the density, the velocity of the mixture, the volume fraction and where  $\varphi_1, \dots, \varphi_{\mathcal{N}_{\text{param}}}$  is a set of  $\mathcal{N}_{\text{param}}$  additional scalar fields that characterize the flow. These fields will then be specified in a case by case fashion and may feature flow parameters such as the interfacial area density  $\Sigma$ . Let us mention that such choice of variables is just meant to synthetically study all the two-phase models that we shall consider in this work but it is by no means exhaustive. The second step consists in postulating additional constraints that are fulfilled by the system like the conservation of mass or other quantities. Finally in the third step, we define the *Hamiltonian action* for the configuration of the body  $B$  at time  $t$ ,  $B(t)$ ,

$$\boldsymbol{\varsigma} \mapsto \mathcal{A}(\boldsymbol{\varsigma}) = \int_{t_1}^{t_2} \int_{B(t)} \mathcal{L}(\boldsymbol{\varsigma}). \quad (3.69)$$

One then considers a set of perturbed transformation  $\boldsymbol{\epsilon}^\epsilon$  as defined in section 3.3. The [SAP](#) boils down to seeking for the transformation  $\boldsymbol{\varsigma}$  that provide an extremum for the perturbed Action

$$\boldsymbol{\epsilon} \mapsto \mathcal{A}^\epsilon(\boldsymbol{\varsigma}).$$

Before going any further, let us underline that in this work we restrict our study to two-phase fluid systems that involve a single velocity kinematics. The derivation of fluid systems with several material velocities has been considered in the literature, see for example [S. Gavrilyuk and Saurel \(2002\)](#); [Gouin and Ruggeri \(2009\)](#); [Cordesse \(2020\)](#).

### 3.4.1 Extremalizing the Hamiltonian Action

Seeking for the extrema of  $\epsilon \mapsto \mathcal{A}(\zeta)^\epsilon$  boils down to looking for transformations that account for the constraints imposed on the flow variables which verify:

$$\delta \mathcal{A} = \delta \int_{t_1}^{t_2} \int_{B(t)} \mathcal{L} = \int_{t_1}^{t_2} \int_{B(t)} \sum_{b \in \zeta} \frac{\partial \mathcal{L}}{\partial b} \delta b = 0. \quad (3.70)$$

In the following we define  $\mathbf{K}$ ,  $\mathbf{M}$  and  $\mathbf{M}$  by

$$\mathbf{K}^T = \frac{\partial \mathcal{L}}{\partial \mathbf{u}}, \quad \mathbf{M} = \frac{\partial \mathcal{L}}{\partial \mathbf{D}_t \alpha} \quad \mathcal{L} + \mathcal{L}^* = \rho \frac{\partial \mathcal{L}}{\partial \rho}. \quad (3.71)$$

Let us underline that  $\mathcal{L}^*$  is defined as a partial Legendre transform of  $\mathcal{L}$  w.r.t. the density to simplify calculations. In order to express all the term of the internal summation in eq. (3.70), first we need to provide expressions for all the field variations  $\delta b$ ,  $b \in \zeta$ .

#### 3.4.1.1 Variation of the velocity

We consider here the evaluation of  $\delta \mathbf{u}$ . By definition of the velocity and the variational derivative we have

$$\delta \mathbf{u}^L = \lim_{\epsilon \rightarrow 0} \frac{\partial}{\partial \epsilon} \left( \frac{\partial \chi^\epsilon}{\partial t} \right) = \lim_{\epsilon \rightarrow 0} \frac{\partial}{\partial t} \left( \frac{\partial \chi^\epsilon}{\partial \epsilon} \right) = \frac{\partial \boldsymbol{\eta}^L}{\partial t} = \frac{\mathbf{D} \boldsymbol{\eta}}{\mathbf{D} t}$$

Using eq. (3.66) for each velocity component we get

$$\delta \mathbf{u}^L = \delta \mathbf{u} + (\boldsymbol{\eta} \cdot \nabla) \mathbf{u} \quad (3.72)$$

We finally obtain

$$\delta \mathbf{u} = \frac{\mathbf{D} \boldsymbol{\eta}}{\mathbf{D} t} - (\boldsymbol{\eta} \cdot \nabla) \mathbf{u} \quad (3.73)$$

#### 3.4.1.2 Variation of the jacobian

The perturbation of  $\chi^\epsilon$  yields a perturbation of the Jacobian  $\mathbf{J} = \det \left( \frac{\partial \chi^\epsilon}{\partial \mathbf{X}} \right)$ . The details of the calculations are given in section 3.A and we only report below the final result: the infinitesimal variation of the Jacobian  $\mathbf{J}$  reads

$$\delta \mathbf{J} = \mathbf{J} \nabla \cdot \boldsymbol{\eta}. \quad (3.74)$$

### 3.4.1.3 Variation of the density

In order to compute the variation of the density we choose to inject the constraint for the conservation of mass directly into the derivation of the variation for the density. Alternatively an approach exploiting Lagrange multipliers is reported in Bedford (1985). From the conservation of mass in Lagrangian form

$$\rho_0^L = J \rho^L \quad (3.75)$$

we apply the variation operator and thanks to eq. (3.74) we get:

$$\begin{aligned} \delta \rho_0^L &= \delta J \rho^L + J \delta \rho^L \\ \Rightarrow 0 &= \rho^L J \nabla \cdot \boldsymbol{\eta} + J \delta \rho^L \end{aligned}$$

where the variation  $\delta \rho_0^L$  of the density at the reference configuration vanishes because by hypothesis, we have supposed that  $\varphi^{\epsilon L}(\mathbf{X}, t = 0) = \varphi^L(\mathbf{X}, t = 0)$  for all  $\epsilon \in (-1, 1)$ . Hence,

$$\delta \rho^L = -\rho^L \nabla \cdot (\delta \mathbf{x}) \quad (3.76)$$

and using expressing  $\delta \rho^L$  thanks to eq. (3.66) yields

$$\delta \rho = -(\nabla \rho \cdot \boldsymbol{\eta} + \rho \nabla \cdot \boldsymbol{\eta}) = -\nabla \cdot (\rho \boldsymbol{\eta})$$

We finally obtain:

$$\delta \rho = -\nabla \cdot (\rho \boldsymbol{\eta}) \quad (3.77)$$

### 3.4.1.4 Variation of a field governed by a transport equation

Let us now derive the expression of the variation for a generic field that is constrained to fulfill a transport equation:

$$\frac{\partial \varphi}{\partial t} + \mathbf{u} \cdot \nabla \varphi = 0 \quad (3.78)$$

As for the case of the density in the section section 3.4.1.3, we incorporate directly this constraint into the calculation of  $\delta \varphi$ . Using the Lagrangian coordinates the transport reads equivalently:

$$\varphi^L(\mathbf{X}, t = 0) = \varphi^L(\mathbf{X}, t)$$

By hypothesis we suppose that  $\varphi^{\epsilon L}(\mathbf{X}, t = 0) = \varphi^L(\mathbf{X}, t = 0)$  for all  $\epsilon \in (-1, 1)$  so that the transport constraint eq. (3.78) reads  $\varphi^{\epsilon L}(\mathbf{X}, t) = \varphi^L(\mathbf{X}, t = 0)$ , for all  $\epsilon \in (-1, 1)$  and all

$t \in (t_1, t_2)$ . Then we immediately have that  $\delta\varphi^L = 0$  and using eq. (3.66) we obtain

$$\delta\varphi = -\boldsymbol{\eta} \cdot \nabla\varphi \quad (3.79)$$

#### 3.4.1.5 Variation of the volume fraction

The variation of the volume fraction  $\alpha$  is not subjected to any specific constraining hypothesis. Therefore the variation  $\delta\alpha$  involved with the  $\epsilon$ -parameterized family of transformations of the medium is arbitrary. We then simply have

$$\alpha^\epsilon(\mathbf{x}, t) = \alpha(\mathbf{x}, t) + \delta\alpha(\mathbf{x}, t) \quad (3.80)$$

We will see in what follows that the fact that the volume fraction variation  $\delta\alpha$  evolves independently will yield a separate equation for the volume fraction field.

#### 3.4.1.6 Generic form of the system

We have derived the expressions of all the required variations  $\delta b$  involved in eq. (3.70), that we recall below for convenience

$$\delta\mathcal{A} = \int_{t_1}^{t_2} \int_{B(t)} \sum_{b \in \zeta} \frac{\partial \mathcal{L}}{\partial b} \delta b = 0.$$

Then we can now fully express the variation of the Hamiltonian action. Let us break  $\delta\mathcal{A}$  into several contributions, as follows

$$\delta\mathcal{A} = \delta\mathcal{A}_\rho + \delta\mathcal{A}_\mathbf{u} + \delta\mathcal{A}_\alpha + \delta\mathcal{A}_{D_t(\alpha)} + \delta\mathcal{A}_b,$$

where:

$$\begin{aligned} \delta\mathcal{A}_\rho &= \int_{t_1}^{t_2} \int_{B(t)} \frac{\partial \mathcal{L}}{\partial \rho} \delta\rho = \int_{t_1}^{t_2} \int_{B(t)} \frac{\mathcal{L} + \mathcal{L}^*}{\rho} \delta\rho \\ &= \int_{t_1}^{t_2} \int_{B(t)} \left( \nabla \mathcal{L}^* + \sum_{b \in \zeta; b \neq \rho} \frac{\partial \mathcal{L}}{\partial b} \nabla b \right) \end{aligned} \quad (3.81)$$

$$\begin{aligned} \delta\mathcal{A}_\mathbf{u} &= \int_{t_1}^{t_2} \int_{B(t)} \mathbf{K} \cdot \delta\mathbf{u} = \int_{t_1}^{t_2} \int_{B(t)} \mathbf{K} \cdot [D_t(\boldsymbol{\eta}) - \nabla\mathbf{u} \cdot \boldsymbol{\eta}] \\ &= - \int_{t_1}^{t_2} \int_{B(t)} \left[ \frac{\partial \mathbf{K}}{\partial t} + \nabla \cdot (\mathbf{K} \otimes \mathbf{u}) + \mathbf{K} \cdot \nabla\mathbf{u} \right] \cdot \boldsymbol{\eta} \end{aligned} \quad (3.82)$$

$$\delta \mathcal{A}_\alpha = \int_{t_1}^{t_2} \int_{B(t)} \frac{\partial \mathcal{L}}{\partial \alpha} \delta \alpha \quad (3.83)$$

$$\begin{aligned} \delta \mathcal{A}_{D_t(\alpha)} &= \int_{t_1}^{t_2} \int_{B(t)} M \delta D_t(\alpha) = \int_{t_1}^{t_2} \int_{B(t)} M (D_t \delta \alpha + \delta \mathbf{u} \cdot \nabla \alpha + \mathbf{u} \cdot \nabla \delta \alpha) \\ &= \int_{t_1}^{t_2} \int_{B(t)} - \left[ \frac{\partial M}{\partial t} + \nabla \cdot (M \mathbf{u}) \right] \delta \alpha - \left[ \nabla \alpha \left( \frac{\partial M}{\partial t} + \nabla \cdot (M \mathbf{u}) \right) \right. \\ &\quad \left. + M \nabla (D_t \alpha) \right] \cdot \boldsymbol{\eta}. \end{aligned} \quad (3.84)$$

$$\delta \mathcal{A}_b = \int_{t_1}^{t_2} \int_{B(t)} \frac{\partial \mathcal{L}}{\partial b} \delta b = - \int_{t_1}^{t_2} \int_{B(t)} \frac{\partial \mathcal{L}}{\partial b} \nabla b \cdot \boldsymbol{\eta} \quad (3.85)$$

Summing up all the contributions from eqs. (3.81) to (3.85), we obtain the final expression for the variation of the Hamiltonian action,

$$\begin{aligned} \delta \mathcal{A} &= \left[ \nabla \mathcal{L}^* - \frac{\partial \mathbf{K}}{\partial t} + \nabla \cdot (\mathbf{K} \otimes \mathbf{u}) - \frac{\partial M}{\partial t} + \nabla \cdot (M \mathbf{u}) \nabla \alpha + \frac{\partial \mathcal{L}}{\partial \alpha} \nabla \alpha \right] \cdot \boldsymbol{\eta} \\ &\quad + \left[ -\frac{\partial M}{\partial t} + \nabla \cdot (M \mathbf{u}) + \frac{\partial \mathcal{L}}{\partial \alpha} \right] \delta \alpha = 0 \end{aligned} \quad (3.86)$$

As the variations  $\boldsymbol{\eta}$ ,  $\delta \alpha$  are arbitrary, we obtain the system,

$$\begin{cases} \frac{\partial \mathbf{K}}{\partial t} + \nabla \cdot (\mathbf{K} \otimes \mathbf{u}) - \nabla \mathcal{L}^* = \mathbf{0} \\ \frac{\partial M}{\partial t} + \nabla \cdot (M \mathbf{u}) - \frac{\partial \mathcal{L}}{\partial \alpha} = 0 \end{cases} \quad (3.87)$$

The first equation is the conservation of the momentum for the fluid under examination and the second one returns a relation that governs the evolution of the volume fraction material derivative  $D_t \alpha$ . It can be interpreted as a governing equations for the evolution of the smallest scales of the problem. Let us note that eq. (3.87) needs to be complemented by the mass conservation equation and the transport equations of the fields  $\varphi_l$ , that effectively are constraints for the optimization procedure we just performed on  $\mathcal{A}$ . They have been imposed through the injection of the variations expressions. The complete form of the system obtained via the **SAP**, hence reads:

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \\ \frac{\partial \mathbf{K}}{\partial t} + \nabla \cdot (\mathbf{K} \otimes \mathbf{u}) - \nabla \mathcal{L}^* = \mathbf{0} \\ \frac{\partial M}{\partial t} + \nabla \cdot (M \mathbf{u}) - \frac{\partial \mathcal{L}}{\partial \alpha} = 0 \\ \frac{\partial \varphi_l}{\partial t} + \mathbf{u} \cdot \nabla \varphi_l = 0, \quad l = 1, \dots, \mathcal{N}_{\text{param}}. \end{cases} \quad (3.88)$$

### 3.4.1.7 Barotropic Euler system

The Euler system is a set of equations of paramount importance and relevance in the fluid dynamics context, especially for what concerns the simulation of aeronautical flows. It is one of the most studied practical system and it is a classical test bench for numerical schemes. Specific references are LeVeque (1990); Godlewski and Raviart (1996); LeVeque (2002); Eleuterio F. Toro (2009). As an introductory first example of the application of the **SAP** we will propose a classic example (Bedford 1985; S. Gavrilyuk and Gouin 1999): we will retrieve the single component Euler system using the **SAP** framework we just presented. First of all, we will detail the form of the Lagrangian for the system,

$$\mathcal{L}(\rho, \mathbf{u}) = \frac{1}{2} \rho \mathbf{u} \cdot \mathbf{u} - \rho e(\rho) \quad (3.89)$$

where  $e$  is the barotropic potential energy. The thermodynamic pressure  $p$  is defined thanks to  $e$  by the relation

$$p = \rho^2 \frac{de}{d\rho}. \quad (3.90)$$

The terms  $\mathbf{K}$ ,  $M$  and  $\mathcal{L}^*$  take here the form

$$\begin{aligned} \mathbf{K} &= \rho \mathbf{u} \\ M &= 0 \\ \mathcal{L}^* &= \rho \frac{\partial \mathcal{L}}{\partial \rho} - \mathcal{L} = \rho \left( \frac{1}{2} \mathbf{u} \cdot \mathbf{u} - \rho \frac{de}{d\rho} - e \right) - \frac{1}{2} \rho \mathbf{u} \cdot \mathbf{u} + \rho e(\rho) = -p \end{aligned} \quad (3.91)$$

Equation (3.88) becomes then

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0 \\ \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p &= \mathbf{0}. \end{cases} \quad (3.92)$$

We thus indeed retrieve the classic form of the barotropic Euler equations.

### 3.4.2 Introducing the dissipation

The procedure described in section 3.4 that leverages the **SAP** to derive a governing set of equations for a (two-)fluid system, is only able to provide the conservative contribution to the equations. In order to put dissipation at play in the flow we propose to equip the system with an entropy inequality. We start from eq. (3.88), reported here below for convenience using

material derivatives,

$$\left\{ \begin{array}{l} \rho D_t \left( \frac{\mathbf{K}}{\rho} \right) - \nabla \mathcal{L}^* = \mathbf{0} \\ \rho D_t \left( \frac{M}{\rho} \right) - \frac{\partial \mathcal{L}}{\partial \alpha} = R \end{array} \right. \quad (3.93a)$$

$$(3.93b)$$

where we arbitrarily decided to add a source term  $R$  in the equation that governs the volume fraction time rate in order to account for dissipation effects. This choice is coherent with a dissipation that is driven by internal processes between both phases. We shall see that these can be related to the evolution of small scales phenomena in the system. If we multiply eq. (3.93a) by  $\mathbf{u}^T$  and eq. (3.93b) by  $D_t \alpha$ , and then sum the two equations, we obtain after some manipulations,

$$\rho D_t \left[ \frac{\mathbf{K} \cdot \mathbf{u} + D_t \alpha M}{\rho} \right] - \mathbf{K} \cdot D_t \mathbf{u} - \mathbf{u} \cdot \nabla \mathcal{L}^* - M D_t (D_t \alpha) - \frac{\partial \mathcal{L}}{\partial \alpha} \alpha = R D_t \alpha$$

In the context of a system equipped with a barotropic [EoS](#), we can define the Hamiltonian energy  $H$  by setting

$$H + \mathcal{L} = \mathbf{K} \cdot \mathbf{u} + D_t \alpha M \quad (3.94)$$

After some more manipulations, we see that  $H$  verifies the evolution equation

$$\rho D_t \left( \frac{H}{\rho} \right) - \nabla \cdot (\mathcal{L}^* \mathbf{u}) = R D_t \alpha - \sum_{b'} \frac{\partial \mathcal{L}}{\partial b'} D_t b' \quad (3.95)$$

where  $b' : \{b \notin (\rho, \mathbf{u}, \alpha, D_t \alpha)\}$ . Let us briefly consider the following specific case  $R = 0$  when no dissipation occurs and when the auxiliari fields  $b \notin (\rho, \mathbf{u}, \alpha, D_t \alpha)$  verifies a transport equation  $D_t b = 0$ . Then we can see that eq. (3.95) yields an additional conservation law in the form

$$\frac{\partial H}{\partial t} + \nabla \cdot (H \mathbf{u} - \mathcal{L}^* \mathbf{u}) = 0 \quad (3.96)$$

Moreover, in the simple case of the barotropic flow, one can see that this additional conservation law is indeed the conservation of the mathematical entropy of the system for smooth solutions. This suggests that in this case, the Hamiltonian  $H$  and its evolution equation are relevant candidate to equip the system with an entropy and an entropy evolution equation. Let us now turn back to the dissipative case of equation eq. (3.95). The second principle of thermodynamics requires the entropy production to be positive, so that we must ensure that

$$Q = R D_t \alpha - \sum_{b'} \frac{\partial \mathcal{L}}{\partial b'} D_t b' \leq 0 \quad (3.97)$$



The requirement eq. (3.97) must be studied on a case-by-case basis by accounting for the modeling hypotheses related to the dissipative processes in the system. We will see different examples in the next sections and we also refer the reader to Cordesse (2020) for non-barotropic flows and Guillemaud (2007); Cordesse and Massot (2020) for two-phase flows involving two-velocities.

## 3.5

## Two-phase flow models with a rich geometrical description

In section 3.4 we presented the generic SAP framework that allows the derivation of two-phase flow models using a variational approach. The generic final system takes the form of eq. (3.87), that we report here below for convenience

$$\left\{ \begin{array}{l} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \\ \frac{\partial \mathbf{K}}{\partial t} + \nabla \cdot (\mathbf{K} \otimes \mathbf{u}) - \nabla \mathcal{L}^* = \mathbf{0} \\ \frac{\partial M}{\partial t} + \nabla \cdot (M \mathbf{u}) - \frac{\partial \mathcal{L}}{\partial \alpha} = 0 \\ \frac{\partial \varphi_l}{\partial t} + \mathbf{u} \cdot \nabla \varphi_l = 0 \end{array} \right. \quad (3.98)$$

where  $l = 1 \dots N_{\text{aux.}}$  and  $N_{\text{aux.}}$  is the number of auxiliary scalar fields governed by a transport equation. In order to obtain a fully determined system for a specific problem, the actual form of the Lagrangian must be specified. This boils down to expressing the kinetic and potential energies for the problem at hand as functions of the variable set  $\boldsymbol{\varsigma} = (\rho, \mathbf{u}, \alpha, D_t \alpha, \varphi_l)$ . In the following subsections we will express different choices to introduce equations on geometric quantities that are related to the interface between both fluids in a two-phase systems. We will build a set of models that feature different aspects of the subscale modeling for interface flows, starting from the simplest case of a single-phase barotropic fluid.

Before going any further, let us introduce a few hypotheses. We shall suppose that the medium is composed of two barotropic materials  $k = 1, 2$ . Each fluid is equipped with an EOS in the form of a barotropic potential  $\rho_k \mapsto e(\rho_k)$ , so that the pressure law associated with the fluid  $k$  is given by  $\rho_k \mapsto p_k(\rho_k) = \rho_k^2 (de_k/d\rho_k)(\rho_k)$  and the sound velocity  $c_k$  of the fluid  $k$  verifies  $c_k^2 = (dp_k/d\rho_k)(\rho_k)$ . We will make the classic assumption that the density of the

medium is defined by

$$\rho = \alpha \rho_1 + (1 - \alpha) \rho_2 \quad (3.99)$$

and that the mass fraction  $Y$  of the fluid 1 verifies

$$\rho Y = \alpha \rho_1 \quad (3.100)$$

#### 3.5.1 What do we mean by two-scale modeling

Modeling the presence of an interface between both fluids in a two-phase flow problem is indeed quite challenging. The interface can be resolved as a discontinuity surface or as a field with finite thickness, depending on the modeling approach that is chosen. In the common approach that was reviewed in section 3.2.2.3, the interface is thought as a sharp, infinitely thin, surface that separates one zone of the domain occupied by one fluid  $\Omega_1$  from the other zone occupied by the other fluid  $\Omega_2$ . While this modeling choice is clear, it can lead in practice to severe limitations regarding the cases that can be simulated. If we take as an example the flow produced by an injector and we analyze the evolution of the interface between the injected liquid and the surrounding carrier gas, we encounter different regimes characterized by different time and space scales in the domain. As shown in Fig. 2.7, near the injector head a clear surface separating both phases is identifiable: the interface undergoes small perturbations and the acting scales are relatively large. Further downstream, the initial perturbation creates stronger deformations characterized by smaller space scales and faster oscillations, that lead to the creation of ligaments and droplets at the very end of the domain. In a context where computational resources are limited, we are only able to resolve the interface up to a certain level and discard fine features of the interface. This suggests to consider models that do not aim at describing the complete geometrical features of the two-phase interface but rather involve a choice of so-called large scale features that can be fully resolved and other small scale features that need to be modeled thanks to reasonable assumptions. We shall say that this approach involves a two-scale modeling where we intend to re-inject into the model informations pertaining to small scales that are deliberately discarded from the model by the scale separation. The artistic representation of Fig. 3.2 shows the effect of this scale separation on the resolution of the interface. When we are incapable of resolving the interface, the information we retrieve, for example from the volume fraction field, resembles the continuous line shown in Fig. 3.2 on the bottom. That outcome is totally non-representative of the real scenario, as displayed on the top of the figure, in the (right) zone of the domain containing the smallest scales in time and space, *i.e.* the *disperse-phase* zone. Enriching the large scale model thanks to additional

geometrical information related to the small scale interface features can be achieved by considering the evolution of the interfacial area density like in many models of the literature (Ishii and Mishima 1984; Rusche 2003). In our case, we intend to use properties like the Gauss curvature and the mean curvature, which could be used in the disperse-phase zone to discern between a situation in which the interface can be described at the large scale and the situation in which instead a spray arises (Essadki 2018; Loison 2023).

We tackle this two-scale nature of the problem within the framework of the **SAP** postulating “subscale” energies representing a particular flow structure: spherical pulsating droplets as in Drui, Larat, et al. (2019), non-spherical droplets pulsating in the normal direction (section 3.5.2), ellipsoidal droplets oscillating with constant volume (section 3.5.3). Please note that the “subscale” term we are employing in this context has no connection to turbulence modeling which, willingly, we do not consider in this work. An interesting analysis about second-order turbulence metrics in two-phase flow simulations featuring an equation for the interfacial area density is carried on in Remigi (2021). A last note on the description of the interface we just introduced: other approaches are possible that do not refer to the interface as a sharp surface; in Cordesse, Di Battista, Drui, et al. (2020) the interface is described thanks to a probabilistic approach that relies on of a **Probability Density Function (PDF)** supplemented by filtering kernel that allows to separate the large scale and the small scale of the interface in the flow.

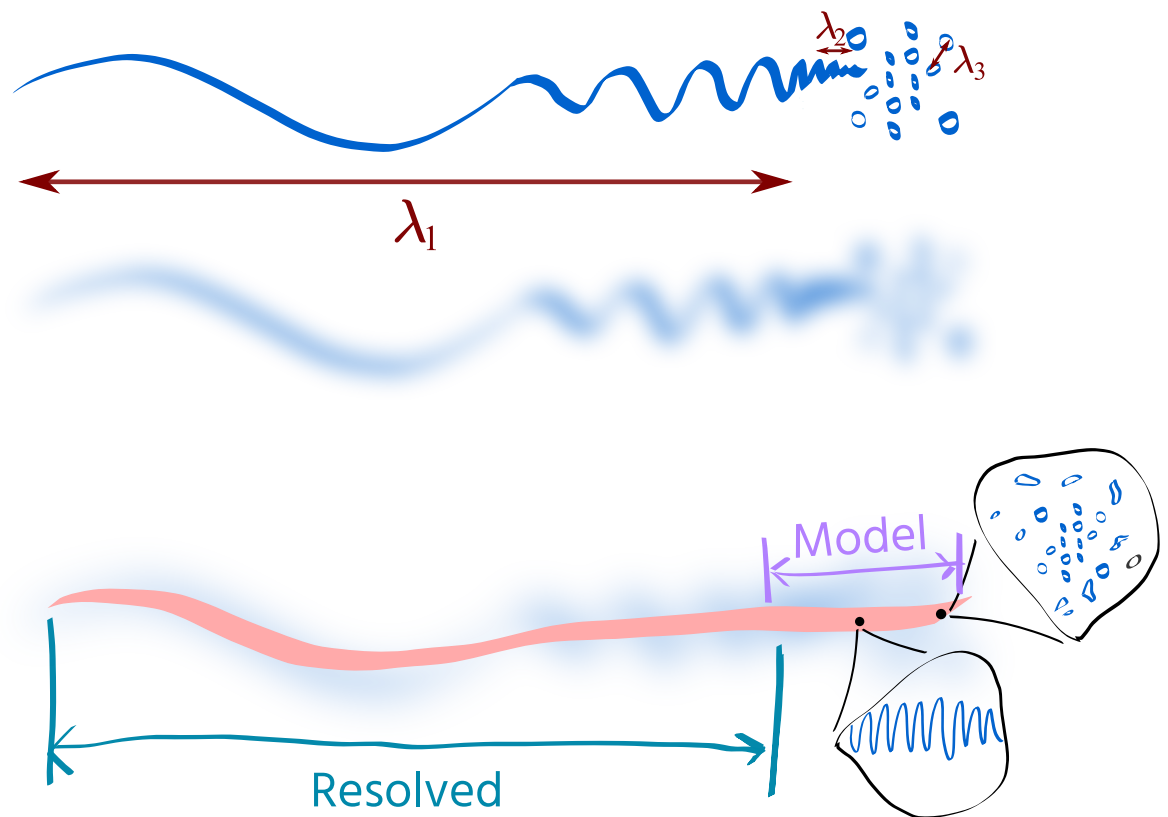


Figure 3.2: At lower scales, the interface cannot be resolved and it needs modeling

### 3.5.2 Two-scale modeling of small inclusions assuming normal deformation of the interface

In the first two-scale model we present, the large scale contribution to the Lagrangian is inspired by classical interface capturing methods (*e.g.* Chantepedrix et al. 2002). The interface position is supposed to lie in a region of fast change of the volume fraction  $\alpha$ , from the value  $\alpha = 0$ , in which only the phase 1 is present, to  $\alpha = 1$ , in which only the phase 2 is present. Therefore the normal unit vector to the interface in the large scale regime is accurately approximated by the quantity

$$\hat{n} = \frac{\nabla \alpha}{\|\nabla \alpha\|} \quad (3.101)$$

in the transition region  $0 < \alpha < 1$ . When dealing with the small scales regime instead, we assume that the interface dynamics can be approximately retrieved thanks to additional fields  $\Sigma(\mathbf{x}, t)$ ,  $H(\mathbf{x}, t)$  that respectively account for the small scales of the interfacial area density and the average mean curvature of the interface in the vicinity of the position and time  $(\mathbf{x}, t)$ . In order to use  $\Sigma$ ,  $H$  to reconstruct the small scales interfacial dynamics, a model must be specified.

In Drui, Larat, et al. (2019) the authors take as small scale model a set of pulsating spherical inclusions. In this work, we aim at generalizing that take, assuming the population of objects that are present at small scale is a set of pulsating inclusion that are **homeomorphic** to a sphere, instead of being just spherical. For this to be used in the SAP context, a set of governing equations relating  $\Sigma$  to  $H$  needs to be expressed. We leverage the Weyl's tube formula (Weyl 1939) to achieve this task.

#### 3.5.2.1 Basic elements of the Weyl's tube formula

Consider a surface  $\mathcal{S} \subset \mathbb{R}^3$  along with a choice of unit normal vectors defined over  $\mathcal{S}$ . We suppose the  $\mathcal{S}(h)$  is a family of surfaces parametrized by a small parameter  $h \in \mathbb{R}$  obtained by moving each point of  $\mathcal{S}$  away from the surface along the normal to  $\mathcal{S}$  by a distance  $h$ . We note  $S(h) = |\mathcal{S}(h)|$  the area of  $\mathcal{S}(h)$  and  $H^*(\xi)$  the **local** value of the mean curvature at point  $\xi \in \mathcal{S}$ . The tube formula (Weyl 1939) relate  $S(h)$ ,  $S(0)$  and  $H^*$  as follows:

$$S(h) = S(0) - 2h \oint_{\mathcal{S}(0)} H^*(\xi) + O(h^2). \quad (3.102)$$

We can thus define the average value of the mean curvature, *i.e.* the average mean curvature, as

$$\langle H^* \rangle_{\mathcal{S}} \triangleq \frac{\oint_{\mathcal{S}(0)} H^*(\xi)}{S} \quad (3.103)$$

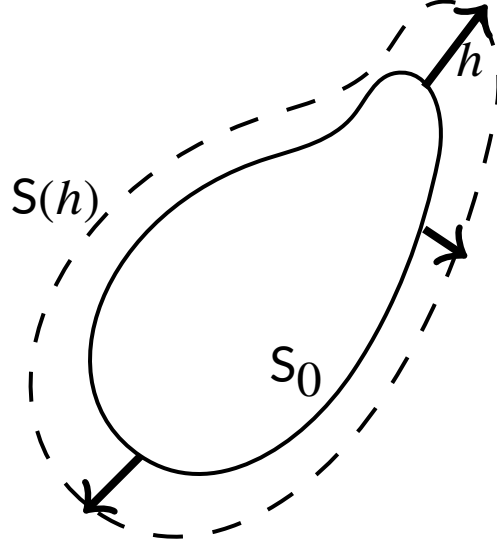


Figure 3.3: Deformation of a droplet normal to the surface

Equation (3.103) allows then to write the following expression relating the average mean curvature and the derivative of the surface area,

$$\frac{dS}{dh} = -2 \langle H^* \rangle_{\mathcal{S}} S \quad (3.104)$$

Injecting the definition of the average mean curvature (eq. (3.103)) into eq. (3.102), and integrating over the normal distance  $h$ , allows to retrieve a relation that links the volume of the inclusion and its surface area:

$$\frac{dV}{dh} = S \quad (3.105)$$

Equation (3.105) is a geometrical relation that is derived using the Weyl's tube formula and is strictly related to a single object. But if we consider a constant number of objects  $N_{\text{obj}}$  of volume  $V_k$  per unit volume, hence no coalescence or breakup is occurring, we can calculate the volume fraction in the reference unit volume  $V_{\text{ref}}$  as:

$$\frac{1}{V_{\text{ref}}} \sum_k^{N_{\text{obj}}} V_k \triangleq \alpha \quad (3.106)$$

if we now derive eq. (3.106) by the normal distance, and using eq. (3.105) per each object, we obtain the following relation for the entire population of inclusions:

$$\frac{d\alpha}{dh} = \sum_k^{N_{\text{obj}}} \frac{S_k}{V_{\text{ref}}} \triangleq \Sigma \quad (3.107)$$

Similarly, summing up all the surface contributions in eq. (3.104) per each object per unit volume, we obtain the relation for the entire population:

$$\frac{d\Sigma}{dh} = \frac{1}{V_{\text{ref}}} \sum_k^{N_{\text{obj}}} \frac{dS_k}{dh} = -\frac{2}{V_{\text{ref}}} \sum_k^{N_{\text{obj}}} \langle H_k^* \rangle_{\mathcal{S}} S_k = -2H\Sigma \quad (3.108)$$

where we have defined an average mean curvature over the ensemble of inclusions  $H$  as follows:

$$H\Sigma \triangleq \frac{1}{V_{\text{ref}}} \sum_k^{N_{\text{obj}}} \langle H_k^* \rangle_{\mathcal{S}} S_k$$

### 3.5.2.2 Application of the SAP

In order to account for the two-scale nature of the model we aim at deriving, we consider several energies that come into play in the system: a classical bulk kinetic energy  $\mathcal{K}_{\text{bulk}}$  and potential energy bulk  $\mathcal{U}_{\text{bulk}}$ , a potential energy  $\mathcal{U}_{\text{large}}$  pertaining to large scale capillary effects and two additional energies  $\mathcal{K}_{\text{small}}$  and  $\mathcal{U}_{\text{small}}$  related to small scale phenomena. By classical choice we set:

$$\mathcal{K}_{\text{bulk}} = \frac{1}{2} \rho \mathbf{u} \cdot \mathbf{u}, \quad \mathcal{U}_{\text{bulk}} = \rho e(\rho, Y, \alpha) \quad (3.109)$$

where  $e(\rho, Y, \alpha)$  is a barotropic potential energy for the two-phase mixture and  $Y$  is the mass fraction of the fluid 1. The large scale capillary potential that accounts for surface tension when the geometry of the interface can be captured by the variation of  $\alpha$  is set to

$$\mathcal{U}_{\text{large}} = \frac{1}{2} \sigma |\nabla \alpha|^2 \quad (3.110)$$

where  $\sigma > 0$  is coefficient that characterizes the surface tension. In order to account for small scale kinetic energy, we make the assumption that the small scale kinematic is governed by perturbation of the interface along its normal direction following the lines of section 3.5.2.1 and we suppose that a field  $h(\mathbf{x}, t)$  accounts for the amplitude of these deformations in the

### 3 Two-phase flow models and where to find them

vicinity of  $\mathbf{x}$ . This suggests to define the small scale kinetic energy by

$$\mathcal{K}_{\text{small}} = \frac{1}{2}m(\alpha, \Sigma)(D_t h)^2, \quad (3.111)$$

with  $m > 0$ . Finally, we suppose that the small scale surface tension is associated with a potential energy that is proportional to the interfacial area density  $\Sigma$  with

$$\mathcal{U}_{\text{small}} = \frac{1}{2}\beta\Sigma, \quad (3.112)$$

with  $\beta > 0$  a coefficient that characterize the small scale surface tension. Following [Cordesse, Di Battista, Drui, et al. \(2020\)](#) we make the very strong assumption that the field  $(\mathbf{x}, t) \mapsto \mathbf{H}$  is given *a priori* and that it is not altered by the flow. This assumption can be lifted using a slightly more complex modelling of the subscale behavior (see [Cordesse \(2020\)](#)).

Therefore we postulate that the Lagrangian of the system is composed as follows:

$$\begin{aligned} \mathcal{L}(\rho, \mathbf{u}, Y, \alpha, D_t \alpha, \nabla \alpha, \Sigma, \mathbf{H}) &= \mathcal{K}_{\text{bulk}} - \mathcal{U}_{\text{bulk}} + \mathcal{K}_{\text{small}} - \mathcal{U}_{\text{small}} - \mathcal{U}_{\text{large}} \\ &= \frac{1}{2}\rho \mathbf{u} \cdot \mathbf{u} - \rho e(\rho, Y, \alpha) \\ &\quad + \frac{1}{2}m(\alpha, \Sigma)(D_t h)^2 - \frac{1}{2}\beta\Sigma \\ &\quad - \frac{1}{2}\sigma|\nabla \alpha|^2. \end{aligned} \quad (3.113)$$

The parameters  $\sigma > 0$  and  $\beta > 0$  are assumed constant for the sake of simplicity in the present work.

Let us now turn to the constraints that shall be associated with the system. First, we postulate the classical constraints of conservation of mass and partial mass also apply:

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0 \end{cases} \quad (3.114)$$

$$\begin{cases} \frac{\partial \rho Y}{\partial t} + \nabla \cdot (\rho Y \mathbf{u}) &= 0. \end{cases} \quad (3.115)$$

Then we also consider additional constraints that pertain to the kinematics of the small scale. Using the geometrical constraints that were discussed in section 3.5.2.1 we enforce the variations of  $h$ ,  $\Sigma$  and  $\alpha$  to verify the following relations that are derived after eqs. (3.105) and (3.108):

$$\begin{cases} D_t \Sigma + 2H\Sigma D_t h &= 0 \end{cases} \quad (3.116a)$$

$$\begin{cases} D_t \alpha - \Sigma D_t h &= 0 \end{cases} \quad (3.116b)$$



In order to exhibit the equations that ensure the Hamiltonian action to reach an extremum, we need to evaluate the infinitesimal variations of the fluid parameters. As in section 3.4 we shall account for the constraints of the system in the expression of  $\delta h$ ,  $\delta \rho$ ,  $\delta Y$ ,  $\delta \Sigma$ . The relation of eqs. (3.116a) and (3.116b) imposes that

$$\delta D_t h = \frac{1}{\Sigma} \delta D_t \alpha - \frac{D_t h}{\Sigma} \delta \Sigma \quad (3.117)$$

and using eq. (3.77) for  $\delta \rho$ , eq. (3.79) for  $\delta Y$  and eq. (3.73) for  $\delta \mathbf{u}$  we obtain the following final set of variations:

$$\begin{cases} \delta \rho &= -\nabla \cdot \rho \boldsymbol{\eta} & (3.118a) \\ \delta \mathbf{u} &= D_t(\boldsymbol{\eta}) - \boldsymbol{\eta} \cdot \nabla \mathbf{u} & (3.118b) \\ \delta Y &= -\nabla Y \cdot \boldsymbol{\eta} & (3.118c) \\ \delta D_t h &= \frac{1}{\Sigma} \delta D_t \alpha - \frac{D_t h}{\Sigma} \delta \Sigma. & (3.118d) \end{cases}$$

The present Lagrangian energy and set of constraints does not exactly match the context of presented in section 3.4.1 as the set of variables features different types of constraints like eq. (3.117). Nevertheless, it is possible to carry out the calculation of  $\delta \mathcal{A}$ . These lines can be found in Cordesse, Di Battista, Drui, et al. (2020) but will not be detailed here. If we suppose that the barotropic potential  $e$  of the two-phase medium is defined by

$$e(\rho, Y, \alpha) = Y e_1(\rho_1) + (1 - Y) e_2(\rho_2), \quad (3.119)$$

then the resulting system reads:

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0 & (3.120a) \\ \frac{\partial \rho Y}{\partial t} + \nabla \cdot (\rho Y \mathbf{u}) &= 0 & (3.120b) \\ \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla P + \sigma \nabla \cdot (\nabla \alpha \otimes \nabla \alpha) &= \mathbf{0} & (3.120c) \\ D_t \alpha - \rho Y \omega \Sigma^2 &= 0 & (3.120d) \\ D_t \omega + \frac{1}{\rho Y m} (p_2 - p_1 + \nabla \cdot (\sigma \nabla \alpha)) &= 0 & (3.120e) \\ D_t(\rho \Sigma) + \frac{2 \rho Y \omega \Sigma}{\beta} (p_2 - p_1 + \nabla \cdot (\sigma \nabla \alpha)) &= 0 & (3.120f) \end{cases}$$

with:

$$\omega = \frac{D_t \alpha}{(\rho Y \Sigma^2)} \quad (3.121)$$

$$P = \alpha p_1(\rho_1) + (1 - \alpha) p_2(\rho_2) + \frac{1}{2} m (\rho Y \Sigma \omega)^2 - \frac{\sigma}{2} |\nabla \alpha|^2. \quad (3.122)$$

Equation (3.120e) is a small scale momentum equation on the variable  $\omega$  which can be interpreted as the pulsation of small scale interface structures. In the large scale momentum equation eq. (3.120c) we obtain terms involving  $\nabla \alpha$  that are also found in literature, as for example in Chantepredrix et al. (2002); Blanchard et al. (2016) that pertain to capillary effects.

**Characteristic velocities** In the absence of capillary effects, one can see that the Jacobian matrix associated with eq. (3.120a) is diagonalizable in  $\mathbb{R}^1$ . Its eigenvalues are:

$$\lambda_{1,2,3,4} = u, \lambda_{5,6} = u \pm c_h \quad (3.123)$$

where

$$c_h^2 = \frac{\partial P}{\partial \rho} + \frac{\Sigma}{\rho} \frac{\partial P}{\partial \Sigma} \quad (3.124)$$

Therefore one can conclude that the convective part of eq. (3.120a) is hyperbolic when capillarity is neglected.

### 3.5.3 Two-scale modeling of small inclusions assuming incompressible constant volume oscillations

In this section we set different hypotheses on the small scale behavior of the flow. In section 3.5.2 we described a small scale model in which inclusions that are homeomorphic to a sphere pulsate normally to their equilibrium surface. We assume here instead to have ellipsoidal inclusions that deviate from the spherical equilibrium condition with a periodic deformation that conserves the initial volume when almost no viscous dissipation is present (see Fig. 4.14 for a visualization of the DNS we performed to validate this assumption, described in details in chapter 4).

Before diving in the mathematical aspects of the development, we report that the study of oscillations of liquid inclusion in high speed gaseous flow field has been proposed in O'Rourke and Amsden (1987) base on an analogy with the original work of Taylor (Taylor 1963) published in 1949. The purpose was to propose a model (the TAB model) in order to describe secondary breakup of droplets under the condition of high slip velocity between the phases,

which causes the droplet to oscillate and potentially break into smaller droplets when sufficient deformation is reached, while viscous dissipation is damping the process. However, the study of the small amplitude oscillations of a quasi-spherical incompressible liquid droplet through surface tension can be traced back to the work of Lord Rayleigh in 1883 (John W. Strutt (3rd Baron Rayleigh) 1883; S. H. Lamb 1895; Chandrasekhar 1961) and we refer to the literature review on the subject we propose in section 4.6.1. The modeling choices reported in this section are also inspired by this original corpus of work in which the author presents a simplified modeling approach installed in the framework of small perturbations of a sphere and incompressible liquid inclusions involving potential flow.

Under this set of assumptions, it can be shown that any smooth initial surface deformation with zero flow velocity can be decomposed into spherical harmonics and each mode corresponds to an independent harmonic oscillator with prescribed frequency. The flow field inside the droplet can be evaluated analytically as well as the periodic evolution of its shape. Thus, the corresponding evolution of kinetic  $\mathcal{K}_{\text{small}}$  and potential  $\mathcal{U}_{\text{small}}$  energies per unit volume along the oscillation of the droplet can be evaluated analytically.

#### 3.5.3.1 Energy models for the small scale inclusions

The starting point of this study consists in analyzing the kinetic energy  $\mathcal{K}_{\text{single}}$  and potential energy  $\mathcal{U}_{\text{single}}$  of a single inclusion assuming its deformation can be described by an ellipsoid of revolution with axes  $p, q$  and approximated by a single spherical harmonics with the proper symmetries. We do not present these calculations that are detailed in section 3.B. The resulting expression for these energies are:

$$\mathcal{K}_{\text{single}} = \frac{8}{45} \eta^2 \sigma S_0 \sin^2 \left( t \sqrt{\frac{8\sigma}{\rho R_0^3}} \right) \quad (3.125a)$$

$$\mathcal{U}_{\text{single}} = \frac{8}{45} \eta^2 \sigma S_0 \cos^2 \left( t \sqrt{\frac{8\sigma}{\rho R_0^3}} \right) \quad (3.125b)$$

where  $\eta = p/q - 1, \eta \ll 1$  is the small stretch ratio of the ellipsoid,  $\sigma$  the surface tension,  $S_0$  the area of the surface at equilibrium,  $R_0$  the reference radius at equilibrium. In particular eq. (3.125b) is computed assuming a form of the potential energy such that

$$\mathcal{U}_{\text{single}} = \sigma(S - S_0) \quad (3.126)$$

However, in the small perturbation regime, several variables behave harmonically, for example the averaged mean curvature of the droplet does also have an harmonic form (see section 3.B). In particular, let us mention that [Herrmann \(2013\)](#) proposes an alternate relation with

$$\mathcal{U}_{\text{single}} = \sigma \frac{S_0}{H_0} (H - H_0) \quad (3.127)$$

where  $H$  the average mean curvature and  $H_0 = 1/R_0$ . Eventually, [O'Rourke and Amsden \(1987\)](#) also proposes another set of variables in order to describe this harmonic oscillator and it is not clear from the beginning what should be the proper variable and potential energy, as well as the proper set of describing variables. The reasons is that everything is equivalent as far as we are in the small perturbation regime, whereas the choice is of major importance when we reach large deviations from sphericity. The study in chapter 4 allows to conclude that the proper choice for a single droplet is given by eq. (3.126) relying on accurate DNS post-processing framework.

This analysis suggests to consider a generalization of the single inclusion relation eq. (3.126) to a population of objects in order to derive energies and constraints for the small scale phenomena.

We consider a portion of space whose volume is  $V_{\text{ref}}$  that contains a population of  $N_{\text{obj}}$  inclusions with a surface  $S_k$  and equilibrium surface  $S_{0k}$ . We set

$$\tilde{\Sigma} \triangleq \frac{1}{V_{\text{ref}}} \sum_{k=1}^{N_{\text{obj}}} (S_k - S_{0k})$$

so that  $\tilde{\Sigma}$  can be viewed as a density of potential energy associated with the population of inclusions. In order to extend eq. (3.126) to our population, we choose to define the potential energy  $\mathcal{U}_{\text{small}}$  associated with the small scale by

$$\mathcal{U}_{\text{small}} = \sigma \tilde{\Sigma}. \quad (3.128)$$

Regarding the kinetic energy  $\mathcal{K}_{\text{small}}$  of the small scale, we set

$$\mathcal{K}_{\text{small}} = \frac{1}{2} m_{\tilde{\Sigma}} \left( D_t \tilde{\Sigma} \right)^2, \quad (3.129)$$

where  $m_{\tilde{\Sigma}} > 0$  is a parameter that we assume it depends on the interfacial area defect  $\tilde{\Sigma}$ .

## 3.5.3.2 Application of the SAP

As in section 3.5.2.2 we define the bulk kinetic energy  $\mathcal{K}_{\text{bulk}}$  and the bulk potential energy  $\mathcal{U}_{\text{bulk}}$

$$\mathcal{K}_{\text{bulk}} = \frac{1}{2} \rho \mathbf{u} \cdot \mathbf{u}, \quad \mathcal{U}_{\text{bulk}} = \rho e(\rho, Y, \alpha) = Y e_1(\rho_1) + (1 - Y) e_2(\rho_2) \quad (3.130)$$

We then postulate a Lagrangian of the form:

$$\begin{aligned} \mathcal{L}(\rho, \mathbf{u}, Y, \alpha, \tilde{\Sigma}, D_t \tilde{\Sigma}) &= \mathcal{K}_{\text{bulk}} - \mathcal{U}_{\text{bulk}} + \mathcal{K}_{\text{small}} - \mathcal{U}_{\text{small}} \\ &= \frac{1}{2} \rho \mathbf{u} \cdot \mathbf{u} - \rho e(\rho, Y, \alpha) + \frac{1}{2} m_{\tilde{\Sigma}} \left( D_t \tilde{\Sigma} \right)^2 - \sigma \tilde{\Sigma} \end{aligned} \quad (3.131)$$

where the first two contributions pertain to the carrier (bulk) fluid, the last two terms showcase the subscale behavior we described in section 3.5.3.1.

Let us now turn to the constraints to be applied to parameters of the flow. Since in this configuration the deformations are not normal to the surface anymore, as we assumed instead in section 3.5.2, we need a different constraining relation *w.r.t.* eqs. (3.105) and (3.108). Since we are assuming constant volume, we impose

$$D_t \alpha = 0 \quad (3.132)$$

We also propose the relation

$$D_t \left( \tilde{\Sigma} + 3H\alpha \right) = 0 \quad (3.133)$$

that is exact for a single spherical droplet. As the volume of the inclusions is kept constant, the number density of oscillating objects does not change if no breakup or coalescence phenomena occur, thus we can extend the relation eq. (3.133) to the entire population. The relation eq. (3.133) is assumed to remain valid in the small perturbations regime that we consider here and it is currently subject of complementary studies by Loison (2023). In chapter 4 we provide numerical insights to support this hypothesis. We complement these above assumption with

the conservation of total mass and partial masses so that the constraints for our system read

$$\left\{ \begin{array}{l} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \\ \frac{\partial \rho Y}{\partial t} + \nabla \cdot (\rho Y \mathbf{u}) \end{array} \right. = 0 \quad (3.134a)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho Y}{\partial t} + \nabla \cdot (\rho Y \mathbf{u}) \\ D_t(\tilde{\Sigma} + 3\alpha H) \end{array} \right. = 0 \quad (3.134b)$$

$$\left\{ \begin{array}{l} D_t(\tilde{\Sigma} + 3\alpha H) \\ D_t \alpha \end{array} \right. = 0 \quad (3.134c)$$

$$\left\{ \begin{array}{l} D_t \alpha \end{array} \right. = 0 \quad (3.134d)$$

We can now express the infinitesimal variations that are obtained by accounting for the constraints eq. (3.134). In particular Equation (3.133) introduces an additional expression that relates the infinitesimal variation  $\delta H$ ,  $\delta \tilde{\Sigma}$ ,  $\boldsymbol{\eta}$ , as follows:

$$\delta H = -\frac{\delta \tilde{\Sigma}}{3\alpha} - \left( \frac{\nabla \tilde{\Sigma}}{3\alpha} + \nabla H \right) \cdot \boldsymbol{\eta} \quad (3.135)$$

We complement this relation by the expression of  $\delta \rho$ ,  $\delta \mathbf{u}$ ,  $\delta Y$  and  $\delta \alpha$  obtained thanks to the calculations of section 3.4 and the set of infinitesimal variations reads:

$$\left\{ \begin{array}{l} \delta \rho = -\nabla \cdot \rho \boldsymbol{\eta} \\ \delta \mathbf{u} = D_t(\boldsymbol{\eta}) - \boldsymbol{\eta} \cdot \nabla \mathbf{u} \end{array} \right. \quad (3.136a)$$

$$\left\{ \begin{array}{l} \delta \mathbf{u} = D_t(\boldsymbol{\eta}) - \boldsymbol{\eta} \cdot \nabla \mathbf{u} \\ \delta Y = -\nabla Y \cdot \boldsymbol{\eta} \end{array} \right. \quad (3.136b)$$

$$\left\{ \begin{array}{l} \delta Y = -\nabla Y \cdot \boldsymbol{\eta} \\ \delta \alpha = -\nabla \alpha \cdot \boldsymbol{\eta} \end{array} \right. \quad (3.136c)$$

$$\left\{ \begin{array}{l} \delta \alpha = -\nabla \alpha \cdot \boldsymbol{\eta} \\ \delta H = -\frac{\delta \tilde{\Sigma}}{3\alpha} - \left( \frac{\nabla \tilde{\Sigma}}{3\alpha} + \nabla H \right) \cdot \boldsymbol{\eta} \end{array} \right. \quad (3.136d)$$

$$\left\{ \begin{array}{l} \delta H = -\frac{\delta \tilde{\Sigma}}{3\alpha} - \left( \frac{\nabla \tilde{\Sigma}}{3\alpha} + \nabla H \right) \cdot \boldsymbol{\eta} \end{array} \right. \quad (3.136e)$$

All the ingredients ready: the Lagrangian expression at eq. (3.131), the density, velocity, mass and volume fraction, mean curvature variations eq. (3.136), after some calculations in the same

spirit of the ones we presented in section 3.4, we obtain the conservative part of the system:

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) & = 0 \end{cases} \quad (3.137a)$$

$$\begin{cases} \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla P & = \mathbf{0} \end{cases} \quad (3.137b)$$

$$\begin{cases} \frac{\partial \rho Y}{\partial t} + \nabla \cdot (\rho Y \mathbf{u}) & = 0 \end{cases} \quad (3.137c)$$

$$\begin{cases} D_t \tilde{\Sigma} - \rho Y \omega & = 0 \end{cases} \quad (3.137d)$$

$$\begin{cases} D_t \omega + \frac{1}{Y \rho m_{\tilde{\Sigma}}} \sigma + \frac{1}{2} \frac{\partial_{\tilde{\Sigma}} m_{\tilde{\Sigma}}}{m_{\tilde{\Sigma}}} \rho Y \omega^2 + \frac{1}{\rho Y m_{\tilde{\Sigma}}} & = 0 \end{cases} \quad (3.137e)$$

$$\begin{cases} D_t \alpha & = 0 \end{cases} \quad (3.137f)$$

$$\begin{cases} D_t H + \frac{1}{3\alpha} \rho Y \omega & = 0 \end{cases} \quad (3.137g)$$

where

$$P = \alpha p_1(\rho_1) + (1 - \alpha) p_2(\rho_2) + \frac{1}{2} m_{\tilde{\Sigma}} (\rho Y \omega)^2 - \sigma \tilde{\Sigma} \quad (3.138)$$

**Characteristic velocities** We consider the Jacobian matrix associated with the system eq. (3.137).

It is diagonalizable in  $\mathbb{R}^1$  and its eigenvalues are:

$$\lambda_{1,2,3,4,5} = u, \lambda_{6,7} = u \pm c_{\tilde{\Sigma}} \quad (3.139)$$

where the speed of sound of the system  $c_{\tilde{\Sigma}}^2$  is defined by:

$$c_{\tilde{\Sigma}}^2 = Y c_1^2 + (1 - Y) c_2^2 + m_{\tilde{\Sigma}} \rho Y^2 \omega^2 \quad (3.140)$$

We can conclude that this system is hyperbolic.

### 3.5.3.3 Dissipation

In order to introduce a dissipative contribution to eq. (3.137), we follow the lines of section 3.4.2 and supplement the evolution equation eq. (3.137e) of  $\omega$  with an unspecified source term  $R/(\rho Y m_{\tilde{\Sigma}})$ . We define the Hamiltonian energy of the system with

$$H + \mathcal{L} = \mathbf{K} \cdot \mathbf{u} + (D_t \tilde{\Sigma}) \left( \frac{\partial \mathcal{L}}{\partial D_t \tilde{\Sigma}} \right) \quad (3.141)$$

### 3 Two-phase flow models and where to find them

so that

$$H = \frac{1}{2} \rho \mathbf{u} \cdot \mathbf{u} + \rho Y \omega \left( -\sigma + \frac{1}{2} \frac{\partial m_{\tilde{\Sigma}}}{\partial \tilde{\Sigma}} (\rho Y \omega)^2 \right) \quad (3.142)$$

We choose here again to consider this  $H$  as an entropy for the system and after a manipulation of eqs. (3.137b) and (3.137e), we obtain the evolution equation

$$\rho D_t \left( \frac{H}{\rho} \right) + \nabla \cdot (P \mathbf{u}) = R D_t \tilde{\Sigma} - \rho \frac{\partial e}{\partial \alpha} D_t \alpha \quad (3.143)$$

The **Right-Hand Side (RHS)** of eq. (3.143) needs to be non-positive to ensure a dissipative solution. We choose to enforce the decrease of the mathematical entropy by setting:

$$R = -\epsilon \rho Y \omega \quad (3.144)$$

$$D_t \alpha = -\mu \rho \frac{\partial e}{\partial \alpha} = \mu (p_1 - p_2) \quad (3.145)$$

Therefore we obtain a modified system that accounts for dissipation:

$$\left\{ \begin{array}{l} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \\ \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla P \\ \frac{\partial \rho Y}{\partial t} + \nabla \cdot (\rho Y \mathbf{u}) \\ D_t \tilde{\Sigma} - \rho Y \omega \end{array} \right. \quad \begin{array}{l} = 0 \\ = \mathbf{0} \\ = 0 \\ = 0 \end{array} \quad \begin{array}{l} (3.146a) \\ (3.146b) \\ (3.146c) \\ (3.146d) \end{array}$$

$$\left\{ \begin{array}{l} D_t \omega + \frac{1}{Y \rho m_{\tilde{\Sigma}}} \sigma + \frac{1}{2} \frac{\partial_{\tilde{\Sigma}} m_{\tilde{\Sigma}}}{m_{\tilde{\Sigma}}} \rho Y \omega^2 + \frac{1}{\rho Y m_{\tilde{\Sigma}}} \end{array} \right. = -\frac{\epsilon}{m_{\tilde{\Sigma}}} \omega \quad (3.146e)$$

$$\left\{ \begin{array}{l} D_t \alpha \end{array} \right. = \mu (p_1 - p_2) \quad (3.146f)$$

$$\left\{ \begin{array}{l} D_t H + \frac{1}{3\alpha} \rho Y \omega \end{array} \right. = 0 \quad (3.146g)$$

In the next section we will see that the model of eq. (3.146) can be modified to retrieve a well-known phenomenological model of the literature, the TAB Model proposed by O'Rourke and Amsden.

#### 3.5.3.4 Analogy with O'Rourke and Amsden TAB model and limitations

In O'Rourke and Amsden (1987) a breakup criterion is presented for droplets based on a harmonic oscillator model. If one considers a linear perturbation of an equilibrium of 3.146, one can expect to retrieve a similar harmonic oscillator equation for the interfacial area density



equation replacing eq. (3.146d) into eq. (3.146e). Unfortunately, this analysis fails. A possible interpretation of this drawback is that the system 3.146 does not exhibit a harmonic behavior in the linear regime that is similar to the model of O'Rourke and Amsden (1987). Nevertheless, in order to satisfy this requirement, a modified model can be derived. Indeed, it is possible to define the small scales potential in the form of a harmonic potential. This strategy boils down to carefully choosing an alternate variable to drive the small scale potential. We propose to use the average surface defect  $\Theta = \sqrt{\tilde{\Sigma}}$ . We adopt the same definition for the potential energy, it is expressed as a quadratic function of  $\Theta$

$$\mathcal{U}_{\text{small}} = \sigma \tilde{\Sigma} = \sigma \Theta^2 \quad (3.147)$$

However, we consider an alternate definition of the small scale kinetic energy by setting

$$\mathcal{K}_{\text{small}} = \frac{1}{2} m_{\Theta} (D_t \Theta)^2 \quad (3.148)$$

with  $m_{\Theta} > 0$  a fluid parameter. This suggests then to consider the following Lagrangian energy for the system.

$$\begin{aligned} \mathcal{L}(\rho, \mathbf{u}, Y, \alpha, \tilde{\Sigma}, D_t \tilde{\Sigma}) &= \mathcal{K}_{\text{bulk}} - \mathcal{U}_{\text{bulk}} + \mathcal{K}_{\text{small}} - \mathcal{U}_{\text{small}} \\ &= \frac{1}{2} \rho \mathbf{u} \cdot \mathbf{u} - \rho e(\rho, Y, \alpha) + \frac{1}{2} m_{\Theta} (D_t \Theta)^2 - \sigma \Theta^2 \end{aligned} \quad (3.149)$$

We shall not detail the calculations here but one can see that the choice eq. (3.149) allows to retrieve an harmonic oscillator *w.r.t.* the variable  $\Theta$ .

## 3.6

## Interfacial area density models via the Stationary Action Principle

In section 3.2.2.3 we discussed the derivation of a governing equation for the interfacial area density  $\Sigma$  issued from the averaging approach. In this section we consider an alternate modeling fundamentals based on the SAP.

The starting point of this process is to postulate a form for the evolution equation of the variable  $\Sigma$ . We choose to set

$$\frac{\partial \Sigma}{\partial t} + \nabla \cdot (\Sigma \mathbf{u}) = \frac{2}{3} \Sigma \nabla \cdot \mathbf{u} + s_{\Sigma}(\mathbf{x}, t; \Sigma) \quad (3.150)$$

### 3 Two-phase flow models and where to find them

where  $s_\Sigma$  is a yet unspecified source term. This form is general enough so that it can encompass models of the literature such as [Daniel Lhuillier \(2004\)](#). We will now perform a variable change in order to exhibit a companion transport equation that is equivalent to eq. (3.150). We set

$$z \triangleq \frac{\Sigma^{\frac{3}{2}}}{\sqrt{\rho}} \quad (3.151)$$

so that eq. (3.150) now reads

$$D_t z = \frac{\partial z}{\partial t} + \mathbf{u} \cdot \nabla z = s_z(\alpha, z) \quad (3.152)$$

where  $s_z(\alpha, z)$  is an unspecified source term.

The new unknown  $z$  will be used in our modeling process to account for the evolution of the interface area density. Possible definitions for  $s_z$  will be studied in order to equip the system with a dissipative structure.

#### 3.6.1 Application of the SAP

As in the previous section, we study here the conservative structure of the model. We start by exhibiting constraints upon the variables of the system. We make the assumption that  $s_z(\alpha, z) = 0$  when no dissipation occurs. This implies that  $z$  verifies the pure transport equation

$$D_t z = \frac{\partial z}{\partial t} + \mathbf{u} \cdot \nabla z = 0 \quad (3.153)$$

The equation eq. (3.153) is supplemented by the constraints that the total mass and partial mass conservation are verified as in section 3.5.2.2 and section 3.5.3.2. Let us now turn to the definition of the energies associated with the system. As in sections 3.5.2 and 3.5.3 we define the kinetic energy  $\mathcal{K}_{\text{bulk}}$  and the bulk potential energy  $\mathcal{U}_{\text{bulk}}$  with

$$\mathcal{K}_{\text{bulk}} = \frac{1}{2} \rho \mathbf{u} \cdot \mathbf{u}, \quad \mathcal{U}_{\text{bulk}} = \rho e(\rho, Y, \alpha) = Y e_1(\rho_1) + (1 - Y) e_2(\rho_2) \quad (3.154)$$

Concerning the small scale kinetic energy of the system, as in section 3.5.2 we set

$$\mathcal{K}_{\text{small}} = \frac{1}{2} m (D_t \alpha)^2, \quad (3.155)$$

while the small scale potential energy is chosen as follows

$$\mathcal{U}_{\text{small}} = \sigma \Sigma(\rho, z), \quad (3.156)$$

Then, we postulate the Lagrangian for the system:

$$\mathcal{L}(\rho, \mathbf{u}, Y, \alpha, D_t \alpha, z) = \frac{1}{2} \rho \mathbf{u} \cdot \mathbf{u} - \rho e(\rho, Y, \alpha) + \frac{1}{2} m (D_t \alpha)^2 - \sigma \Sigma(\rho, z) \quad (3.157)$$

We now exhibit the infinitesimal variations involved with the extremalization of  $\mathcal{L}$ . The constraints enforced for the optimization of the Hamiltonian Action are: the conservation of mass and mass fraction, as usual for contribution that address the carrier fluid behavior, and the conservation of the variable  $z$  (eq. (3.153)). The corresponding infinitesimal variation expressions for the density variation (eq. (3.77)), the velocity variation (eq. (3.73)), and the variation eq. (3.79)) both for  $Y$  and  $z$  variables are

$$\delta \rho = -\nabla \cdot (\rho \boldsymbol{\eta}) \quad (3.158a)$$

$$\delta \mathbf{u} = D_t(\boldsymbol{\eta}) - \boldsymbol{\eta} \cdot \nabla \mathbf{u} \quad (3.158b)$$

$$\delta z = -\nabla z \cdot \boldsymbol{\eta} \quad (3.158c)$$

$$\delta Y = -\nabla Y \cdot \boldsymbol{\eta} \quad (3.158d)$$

With the expression of the Lagrangian in eq. (3.157) and the set of variations produced by the imposed constraints (eq. (3.158)), we can exploit the **SAP** as shown in section 3.4, to compute the variation of the Hamiltonian Action. This leads to the following system:

$$\left\{ \begin{array}{l} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \\ \frac{\partial \rho Y}{\partial t} + \nabla \cdot (\rho Y \mathbf{u}) \end{array} \right. = 0 \quad (3.159a)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho Y}{\partial t} + \nabla \cdot (\rho Y \mathbf{u}) \\ \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla P \end{array} \right. = \mathbf{0} \quad (3.159b)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla P \\ \frac{\partial \rho \alpha}{\partial t} + \nabla \cdot (\rho \alpha \mathbf{u}) - \frac{\rho Y \omega}{\sqrt{m}} \end{array} \right. = 0 \quad (3.159c)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho \alpha}{\partial t} + \nabla \cdot (\rho \alpha \mathbf{u}) - \frac{\rho Y \omega}{\sqrt{m}} \\ \frac{\partial \rho Y \omega}{\partial t} + \nabla \cdot (\rho Y \omega \mathbf{u}) + \frac{p_2 - p_1}{\sqrt{m}} \end{array} \right. = 0 \quad (3.159d)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho Y \omega}{\partial t} + \nabla \cdot (\rho Y \omega \mathbf{u}) + \frac{p_2 - p_1}{\sqrt{m}} \\ \frac{\partial \rho z}{\partial t} + \nabla \cdot (\rho z \mathbf{u}) \end{array} \right. = 0 \quad (3.159e)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho z}{\partial t} + \nabla \cdot (\rho z \mathbf{u}) \end{array} \right. = 0 \quad (3.159f)$$

### 3 Two-phase flow models and where to find them

with  $P = \sum_k \alpha_k p_k + \frac{1}{2} \rho (Y\omega)^2 - \frac{2}{3} \sigma \Sigma(\rho, z)$  and the change of variables  $D_t \alpha = \frac{\rho Y \omega}{\sqrt{m}}$ , that is represented by eq. (3.159e), following the lines of [Drui, Larat, et al. \(2019\)](#).

**Characteristic velocities** The Jacobian matrix associated with eq. (3.159) possesses the following set of eigenvalues:

$$\lambda_{1,2,3,4} = u, \lambda_{5,6} = u \pm c_z \quad (3.160)$$

where  $c_z^2 = \partial P / \partial \rho$ , more specifically

$$c_z^2 = Y c_1^2 + (1 - Y) c_2^2 + \rho Y^2 \omega^2 - \frac{2}{9} \sigma \left( \frac{z}{\rho} \right)^{\frac{2}{3}} \quad (3.161)$$

The corresponding eigenvectors are respectively, for the eigenvalues  $\lambda_{1-4}$ :

$$\mathbf{r}_{1-4} = \left( \begin{bmatrix} -\frac{\partial P}{\partial Y} \\ 0 \\ c_z^2 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -\frac{\partial P}{\partial \omega} \\ 0 \\ 0 \\ c_z^2 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -\frac{\partial P}{\partial z} \\ 0 \\ 0 \\ 0 \\ c_z^2 \\ 0 \end{bmatrix}, \begin{bmatrix} -\frac{\partial P}{\partial \alpha} \\ 0 \\ 0 \\ 0 \\ 0 \\ c_z^2 \end{bmatrix} \right), \quad (3.162)$$

It is possible to show that the nature of the characteristic fields associated to the eigenvalues  $\lambda_{1-4}$  are *linearly degenerate*,

$$\nabla \lambda_i \cdot \mathbf{r}_i = 0, \quad i = 1 - 4 \text{ (L.D)} \quad (3.163)$$

For what concerns the fields associated to the eigenvalues  $\lambda_{5-6}$ , we have:

$$\mathbf{r}_{5-6} = \left( \begin{bmatrix} \rho \\ c_z \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \rho \\ -c_z \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right) \quad (3.164)$$

In this case, there is no easy way to state the nature of the characteristic fields, because, even assuming a simple, even constant, [EoS](#) for both phases, there is the possibility, for  $\sigma \Sigma \gg 0$ ,

that the sign of the dot product of eq. (3.165) can change sign.

$$\nabla \lambda_i \cdot \mathbf{r}_i = ?, \quad i = 5 - 6 \quad (3.165)$$

Therefore, this system is diagonalizable in  $\mathbb{R}$  and therefore hyperbolic if  $c_z^2 > 0$ . Consequently, the case of potentially very large value of  $\sigma \left( \frac{z}{\rho} \right)^{\frac{2}{3}}$  may lead to an ill-posed system. The system of eq. (3.159) still misses the source terms on the  $z$  equations, as for example the one presented in Daniel Lhuillier (2004). In order to recover them and to provide them in a way that correctly satisfies the second principle of thermodynamics, we shall use the method explained in general terms in section 3.4.2 that allows to provide the correct sign to the entropy inequality.

### 3.6.2 Including source terms through dissipation, analogy with Daniel Lhuillier (2004)

We will now study how to equip the system eq. (3.159) with an inequality entropy. This will enable connections with models of the literature for the evolution equation of the interfacial area density and more specifically with Daniel Lhuillier (2004).

We now suppose that dissipation can be induced in the system by  $s_z(\alpha, z) \neq 0$  and by an additional unspecified source term  $R$  in the evolution equation of  $\omega$ . We consider the Hamiltonian energy  $H$  defined by

$$H = \mathbf{K} \cdot \mathbf{u} + D_t \alpha M, \quad \mathbf{K} = \partial \mathcal{L} / \partial \mathbf{u}, \quad M = \partial \mathcal{L} / \partial D_t \alpha, \quad (3.166)$$

As in eq. (3.95) we have the following evolution equation:

$$\rho D_t \left( \frac{H}{\rho} \right) - \nabla \cdot (\mathcal{L}^* \mathbf{u}) = R D_t \alpha - \sum_{b' \notin (\rho, \mathbf{u}, \alpha, D_t \alpha)} \frac{\partial \mathcal{L}}{\partial b'} D_t b' \quad (3.167)$$

In the present case eq. (3.167) reads

$$\rho D_t \left( \frac{H}{\rho} \right) - \nabla \cdot (\mathcal{L}^* \mathbf{u}) = R D_t \alpha - \frac{\partial \mathcal{L}}{\partial z} D_t z \quad (3.168)$$

Following section 3.4.2, choosing  $H$  as an entropy for the system requires to ensure that

$$Q = R D_t \alpha - \frac{\partial \mathcal{L}}{\partial z} D_t z = D_t \alpha \left( R - \frac{\partial \mathcal{L}}{\partial z} \frac{D_t z}{D_t \alpha} \right) \leq 0 \quad (3.169)$$

### 3 Two-phase flow models and where to find them

Ad  $D_t z = s_z$  and  $s_z$  is provided by eq. (3.174), in order to enforce the dissipation, we can for example set:

$$R - \frac{\partial \mathcal{L}}{\partial z} \frac{s_z}{D_t \alpha} \triangleq -\epsilon D_t \alpha$$

i.e.

$$R \triangleq -\epsilon D_t \alpha + \frac{\partial \mathcal{L}}{\partial z} \frac{s_z}{D_t \alpha} \quad (3.170)$$

Reminding again the change of variables  $D_t \alpha = \rho Y \omega / \sqrt{m}$ , the non-dissipative system of eq. (3.159) is modified by the introduction of dissipation and it now reads:

$$\left\{ \begin{array}{l} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \\ \frac{\partial \rho Y}{\partial t} + \nabla \cdot (\rho Y \mathbf{u}) \end{array} \right. = 0 \quad (3.171a)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla P \\ \frac{\partial \rho \alpha}{\partial t} + \nabla \cdot (\rho \alpha \mathbf{u}) - \frac{\rho Y \omega}{\sqrt{m}} \end{array} \right. = 0 \quad (3.171b)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho \alpha}{\partial t} + \nabla \cdot (\rho \alpha \mathbf{u}) - \frac{\rho Y \omega}{\sqrt{m}} \\ \frac{\partial \rho Y \omega}{\partial t} + \nabla \cdot (\rho Y \omega \mathbf{u}) + \frac{p_2 - p_1}{\sqrt{m}} \end{array} \right. = 0 \quad (3.171c)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho Y \omega}{\partial t} + \nabla \cdot (\rho Y \omega \mathbf{u}) + \frac{p_2 - p_1}{\sqrt{m}} \\ \frac{\partial \rho z}{\partial t} + \nabla \cdot (\rho z \mathbf{u}) \end{array} \right. = -\epsilon \frac{\rho Y \omega}{m} + \frac{2\sigma}{2Y\omega} \frac{s_z(\alpha, z)}{\rho^{\frac{2}{3}} z^{\frac{1}{3}}} \quad (3.171d)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho Y \omega}{\partial t} + \nabla \cdot (\rho Y \omega \mathbf{u}) + \frac{p_2 - p_1}{\sqrt{m}} \\ \frac{\partial \rho z}{\partial t} + \nabla \cdot (\rho z \mathbf{u}) \end{array} \right. = -\epsilon \frac{\rho Y \omega}{m} + \frac{2\sigma}{2Y\omega} \frac{s_z(\alpha, z)}{\rho^{\frac{2}{3}} z^{\frac{1}{3}}} \quad (3.171e)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho Y \omega}{\partial t} + \nabla \cdot (\rho Y \omega \mathbf{u}) + \frac{p_2 - p_1}{\sqrt{m}} \\ \frac{\partial \rho z}{\partial t} + \nabla \cdot (\rho z \mathbf{u}) \end{array} \right. = \rho s_z(\alpha, z) \quad (3.171f)$$

Please note that the first term of the RHS of eq. (3.171e) is the same dissipative term introduced in [Drui \(2017\)](#) and it is associated to the dissipation of pulsating inclusions. We can see that for any source term  $s_z$  acting in the evolution equation of the interfacial area density, it is possible to incorporate its effect as a dissipation into the system.

Let us now see how the system eq. (3.171) can be connected to the system proposed by in [Daniel Lhuillier \(2004\)](#) for the variable  $\Sigma$ . The original equation proposed in [Daniel Lhuillier \(2004\)](#) for  $\Sigma$  reads

$$\begin{aligned} \frac{\partial \Sigma}{\partial t} + \nabla \cdot [(\alpha \mathbf{u}_2 + (1 - \alpha) \mathbf{u}_1) \Sigma] &= \underbrace{\frac{2}{3} \Sigma \nabla \cdot \mathbf{u} + \frac{2}{3} \Sigma \left( \frac{1}{1 - \alpha} - \frac{1}{\alpha} \right) \left( A + \Gamma \frac{\rho}{\rho_1 \rho_2} \right)}_{(1)} \\ &+ \underbrace{\frac{\Sigma}{t_{BR}} - u_{COA} \Sigma^2}_{(2)} \end{aligned} \quad (3.172)$$

where the term (1) is a term related to compressibility effects and term (2) is associated to the generation and destruction of interfacial area density due to breakup and coalescence. In Daniel Lhuillier (2004), the model presented features two velocities  $\mathbf{u}_1, \mathbf{u}_2$  for the two phases, that are linked to the bulk velocity  $\mathbf{u}$  with  $\mathbf{u} = \alpha \mathbf{u}_2 + (1 - \alpha) \mathbf{u}_1$ . Since our modeling assumptions are based on a single velocity, we replace  $\mathbf{u}$  accordingly in the Left-Hand Side (LHS) of eq. (3.172). The parameter  $t_{BR}$  is a breakup time, while  $u_{COA}$  is the coalescence velocity. These two parameters need to be provided *a priori*, and Daniel Lhuillier (2004) provides an expression for  $u_{COA} = \alpha(1 - \alpha)\sigma/\mu_{eff}$  where  $\sigma$  is the surface tension and  $\mu_{eff}$  some effective viscosity of the mixture. For the sake of simplicity we will neglect the compressibility effects (1), and the source term in eq. (3.150) then reads:

$$s_\Sigma(\alpha, \Sigma) = \Sigma \left( \frac{1}{t_{BR}} - \frac{\alpha(1 - \alpha)\sigma}{\mu_{eff}} \Sigma \right) \quad (3.173)$$

if we then apply the change of variable of eq. (3.151), *i.e.*  $z = \Sigma^{2/3}/\rho^{1/2}$ , to eq. (3.173), the source term eq. (3.172) can be recasted in terms of  $z$  as a relaxation:

$$s_z(\alpha, z) = -\frac{3}{2}z \frac{\alpha(1 - \alpha)\sigma}{\mu_{eff}} (\Sigma - \Sigma_{eq}) \quad (3.174)$$

where:

$$\Sigma_{eq} \triangleq \frac{\mu_{eff}}{\alpha(1 - \alpha)\sigma t_{BR}} \quad (3.175)$$

We can now use the expressions eq. (3.174) and eq. (3.175) in the dissipative system eq. (3.171). We then get

$$\left\{ \begin{array}{l} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \\ \frac{\partial \rho Y}{\partial t} + \nabla \cdot (\rho Y \mathbf{u}) \end{array} \right. = 0 \quad (3.176a)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho Y}{\partial t} + \nabla \cdot (\rho Y \mathbf{u}) \\ \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla P \end{array} \right. = \mathbf{0} \quad (3.176b)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla P \\ \frac{\partial \rho \alpha}{\partial t} + \nabla \cdot (\rho \alpha \mathbf{u}) - \frac{\rho Y \omega}{\sqrt{m}} \end{array} \right. = 0 \quad (3.176c)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho \alpha}{\partial t} + \nabla \cdot (\rho \alpha \mathbf{u}) - \frac{\rho Y \omega}{\sqrt{m}} \\ \frac{\partial \rho Y \omega}{\partial t} + \nabla \cdot (\rho Y \omega \mathbf{u}) + \frac{p_2 - p_1}{\sqrt{m}} \end{array} \right. = 0 \quad (3.176d)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho Y \omega}{\partial t} + \nabla \cdot (\rho Y \omega \mathbf{u}) + \frac{p_2 - p_1}{\sqrt{m}} \\ \frac{\partial \rho z}{\partial t} + \nabla \cdot (\rho z \mathbf{u}) \end{array} \right. = -\epsilon \frac{\rho Y \omega}{m} + \sigma \frac{\Sigma}{\Sigma_{eq}} \frac{(\Sigma - \Sigma_{eq})}{t_{BR} \rho Y \omega} \quad (3.176e)$$

$$\left\{ \begin{array}{l} \frac{\partial \rho z}{\partial t} + \nabla \cdot (\rho z \mathbf{u}) \\ \frac{\partial \rho z}{\partial t} + \nabla \cdot (\rho z \mathbf{u}) \end{array} \right. = -\frac{3}{2} \rho z \frac{\alpha(1 - \alpha)\sigma}{\mu_{eff}} (\Sigma - \Sigma_{eq}) \quad (3.176f)$$

## 3.7 Conclusions and Perspectives

Modeling two-phase flows satisfactorily is a challenging problem. In this chapter we made an attempt to provide a fundamental picture on the different possibilities one has to build model for interfacial flows that might account for the transition from the separated phase regime to the dispersed phase regime. Notably, we present an *excursus* on the state of the art of the models that are obtained using an averaging approach. These models involve terms exerting from the averaging process that can be difficult to close in a physical way. We present as an alternative, a unified approach that employs a variational method called Stationary Action Principle (SAP) in which the modeling assumptions are taken *a priori* at the beginning of the formal modeling procedure, when the energies included in a Lagrangian are postulated. This Lagrangian is then integrated over the phase space — this integral functional is called Hamiltonian Action — and its stationarization allows to retrieve a mathematical model that represents the convective contribution of the system. In addition, it is possible to account for additional effects, introducing source terms that comply with the second principle of thermodynamics, *i.e.* they are constructed so that the entropy production of the system has the correct sign.

We have shown that the method based on the Stationary Action Principle (SAP) that was exploited in (Drui 2017; Cordesse 2020) is not restricted to theoretical models and that it can be used to also encompass more phenomenological models like (O’Rourke and Amsden 1987) and the model of Daniel Lhuillier (2004). In particular with this approach we show that we are able to create a model that includes an equation for the interfacial area density  $\Sigma$  that is compatible with a large range of models, including the model presented in Daniel Lhuillier (2004). Closure relations motivated by both post-processing of DNS studies and analytical analysis of single deformed were also proposed and used to build a complete model thanks to the Stationary Action Principle (SAP). This work contributes to the long term efforts paved by (Drui 2017; Essadki 2018; Cordesse 2020) to design a unified framework for modeling two-phase at different scales and several improvement can be envisioned. A first possible sequel to this work would be to perform a more thorough analysis of the connections between the model studied in section 3.5.3.4 including shear terms.

Another possible perspective is specific to the study of the equation of the interfacial area density: it would consists in trying to encompass additional geometrical terms that can occurs in the interfacial area density equation (Morel 2015).

In a general sense, the set of small scale models we assumed in the mathematical developments we discussed in this chapter are a small contribution towards a general treatment of



non-spherical inclusions. Relaxing hypothesis on the nature of the deformation they undergo at lower scales, but also enlarging the spectrum of sizes that are accounted for, shall help in drawing the line towards the correct handling of polydispersed populations of non-spherical objects. A general path forward on this subject is to account for polydispersion and arbitrary deformations via the use of families of spherical harmonics and a functional dependency on the surface defect as per section 3.5.3.4. This work is already in progress through the thesis of [Loison \(2023\)](#) within the research team that hosted the present work.



# Appendices

## 3.A Variation of the jacobian

$\det\left(\overset{\epsilon}{\mathbf{F}}\right)^1$  can be intended as an indirect dependency on  $\epsilon$ , i.e.

$$\det\left(\overset{\epsilon}{\mathbf{F}}\right) = \det(\mathbf{F}(\epsilon))$$

In addition to that, let's interpret the determinant as a function of the tensor components:

$$\det\left(\overset{\epsilon}{\mathbf{F}}\right) = f(F_{ij}(\epsilon))$$

So when we try to do the derivative of the determinant, we need to use the chain rule:

$$\frac{\partial}{\partial \epsilon} \left( \det\left(\overset{\epsilon}{\mathbf{F}}\right) \right) = \sum_i \sum_j \frac{\partial \det\left(\overset{\epsilon}{\mathbf{F}}\right)}{\partial F_{ij}} \frac{\partial F_{ij}}{\partial \epsilon} \quad (3.177)$$

The second term in the multiplication by definition is:

$$\frac{\partial F_{ij}}{\partial \epsilon} = \frac{\partial}{\partial \epsilon} \left( \frac{\partial \overset{\epsilon}{\chi}_i}{\partial X_j} \right) = \frac{\partial}{\partial X_j} \left( \frac{\partial \overset{\epsilon}{\chi}_i}{\partial \epsilon} \right) = \frac{\partial \eta_i^L}{\partial X_j}$$

But if we want to use the variables in the Eulerian frame  $t$ , i.e. using  $\boldsymbol{\eta}(\mathbf{x}, t)$  (and not  $\boldsymbol{\eta}^L(\mathbf{X}, t)$ ), again with the chain rule:

$$\frac{\partial \eta_i^L}{\partial X_j} = \sum_k \frac{\partial \eta_i}{\partial x_k} \frac{\partial x_k}{\partial X_j} = \sum_k \frac{\partial \eta_i}{\partial x_k} \frac{\partial \chi_k}{\partial X_j} = \sum_k \frac{\partial \eta_i}{\partial x_k} F_{kj} \quad (3.178)$$

<sup>1</sup> All the derivation is made in Lagrangian coordinates, so the  $^L$  is omitted. At the end of the derivation where we need to transform to Eulerian coordinates, we use it again for the sake of clarity

### 3 Two-phase flow models and where to find them

Let's now focus on the derivative of the determinant. Using the Laplace formula ( $\ell$  can be chosen arbitrarily from  $1 \dots n$ ,  $n$  number of rows):

$$\det(\mathbf{F}) = \sum_m F_{\ell m} \text{cof}(\mathbf{F})_{\ell m} \quad (3.179)$$

and the derivative is hence (derivative of a product):

$$\frac{\partial \det \left( \overset{\epsilon}{\mathbf{F}} \right)}{\partial F_{ij}} = \sum_m \frac{\partial F_{\ell m}}{\partial F_{ij}} \text{cof}(\mathbf{F}^*)_{\ell m} + F_{\ell m} \frac{\partial \text{cof}(\mathbf{F}^*)_{\ell m}}{\partial F_{ij}}$$

Since  $\ell$  can be chosen arbitrarily, we set  $\ell = i$ . In this case the term:

$$\frac{\partial \text{cof} \left( \overset{\epsilon}{\mathbf{F}} \right)_{im}}{\partial F_{ij}} = 0$$

because the cofactors do not depend on the elements of the same row ( $i$ ). The term

$$\frac{\partial F_{im}}{\partial F_{ij}} = \delta_{jm}$$

leading to:

$$\frac{\partial \det \left( \overset{\epsilon}{\mathbf{F}} \right)}{\partial F_{ij}} = \sum_m \frac{\partial F_{im}}{\partial F_{ij}} \text{cof} \left( \overset{\epsilon}{\mathbf{F}} \right)_{im} = \sum_m \delta_{jm} \text{cof}(\mathbf{F}^*)_{im} = \text{cof} \left( \overset{\epsilon}{\mathbf{F}} \right)_{ij}$$

And if we use the Laplace formula eq. (3.179), we can also extract that:

$$\det \left( \overset{\epsilon}{\mathbf{F}} \right) \left( \overset{\epsilon}{F_{ij}} \right)^{-1} = \text{cof} \left( \overset{\epsilon}{\mathbf{F}} \right)_{ij} \quad (3.180)$$

Finally injecting eqs. (3.178) and (3.180), in our target equation eq. (3.177), and noting that for  $\epsilon \rightarrow 0$ ;  $\overset{\epsilon}{F} \rightarrow F$ :

$$\begin{aligned} \lim_{\epsilon \rightarrow 0} \frac{d}{d\epsilon} \left( \det \left( \overset{\epsilon}{F} \right) \right) &= \sum_i \sum_j \sum_k \det(F) F_{ij}^{-1} \frac{\partial \eta_i}{\partial x_k} F_{kj} \\ &= \det(F) \sum_i \sum_j \sum_k F_{ij}^{-1} F_{kj} \frac{\partial \eta_i}{\partial x_k} \end{aligned} \quad (3.181)$$

checking:

$$\begin{aligned} \sum_k F_{ij}^{-1} F_{kj} &= \delta_{ki} \\ \lim_{\epsilon \rightarrow 0} \frac{d}{d\epsilon} \left( \det \left( \overset{\epsilon}{F} \right) \right) &= \det(F) \sum_i \sum_j \delta_{ki} \frac{\partial \eta_i}{\partial x_k} = \det(F) \sum_i \frac{\partial \eta_i}{\partial x_i} \\ &= \det(F) \nabla \cdot \boldsymbol{\eta} = J \nabla \cdot \boldsymbol{\eta} \end{aligned}$$

We obtain what we were looking for:

$$\delta J = J \nabla \cdot \boldsymbol{\eta} = J \nabla \cdot \delta \mathbf{x} \quad (3.182)$$

We can exploit the expression for the derivative of the jacobian to express the constraint of the conservation of the mass in Eulerian form. Let us take for ease of notation:

$$\frac{d\bullet^L}{dt} = \frac{\partial \bullet}{\partial t} \Big|_X$$

then,

$$\begin{aligned} \frac{DJ}{DD_t} &= \frac{dJ^L}{dt} = \sum_i \sum_j \frac{\partial J}{\partial F_{ij}} \frac{dF_{ij}}{dt} = \sum_i \sum_j J F_{ij}^{-1} \frac{dF_{ij}}{dt} = \sum_i \sum_j J F_{ij}^{-1} \frac{d}{dt} \left( \frac{\partial \chi_i^L}{\partial X_j} \right) \\ &= \sum_i \sum_j J F_{ij}^{-1} \frac{\partial}{\partial X_j} \left( \frac{d\chi_i^L}{dt} \right) = \sum_i \sum_j J F_{ij}^{-1} \frac{\partial u_i^L}{\partial X_j} = \sum_i \sum_j J F_{ij}^{-1} \frac{\partial u_i}{\partial x_i} \frac{\partial x_i}{\partial X_j} \\ &= \sum_i \sum_j J F_{ij}^{-1} \frac{\partial u_i}{\partial x_i} \frac{\partial \chi_i}{\partial X_j} = \sum_i \sum_j J F_{ij}^{-1} F_{ij} \frac{\partial u_i}{\partial x_i} = J \nabla \cdot \mathbf{u} \\ \frac{DJ}{Dt} &= J \nabla \cdot \mathbf{u} \end{aligned} \quad (3.183)$$

And we can also retrieve the Eulerian form of the conservation of mass, doing:

$$\begin{aligned}
 \frac{d\rho_0^L}{dt} &= \frac{dJ\rho^L}{dt} = 0 \\
 0 &= \frac{dJ\rho^L}{dt} = \frac{DJ}{Dt}\rho + J\frac{D\rho}{Dt} = J\rho\nabla \cdot \mathbf{u} + J\left(\frac{\partial\rho}{\partial t} + \mathbf{u} \cdot \nabla\rho\right) \\
 &\Rightarrow J\left[\frac{\partial\rho}{\partial t} + \rho\nabla \cdot \mathbf{u} + \mathbf{u} \cdot \nabla\rho\right] = 0 \\
 &\Rightarrow \frac{\partial\rho}{\partial t} + \nabla \cdot \rho\mathbf{u} = 0
 \end{aligned} \tag{3.184}$$

## 3.B Energy analytical calculation of a deformed droplet

These calculations were initiated within the context of the intership of Chevalier (Chevalier 2019). We detail here the analytical calculation of the deformed droplet dynamics. We first show that the ellipsoidal deformation of the droplet may be approximated by a single spherical harmonic. The analytical study of Subrahmanyam Chandrasekhar (1981) then allows the determination of the evolution of the kinetic energy of the droplet during its oscillations. The integration of the droplet oscillating surface and mean curvature lead to a surface energy expression displaying expected energy conservation properties.

### 3.B.1 Approximation of the droplet deformation

In the following, we consider that the initial droplet deformation is described by an ellipsoid of  $\mathbf{e}_z$  symmetry axis as shown on Figure 3.B.1a. The deformed surface is then entirely given by its semi-major axis  $p$  and its semi-minor axis  $q$  and may be described in cylindrical  $(z, r, \phi)$  (see Figure 3.B.1b) through the equations

$$\forall(\theta', \phi) \in [0, \pi] \times [0, 2\pi], \quad \begin{cases} z = p \sin \theta', \\ r = q \cos \theta', \\ \phi = \phi, \end{cases}$$

where the  $\theta'$  parameter does not necessarily reduce to the  $\theta$  spherical coordinate. In the following, colatitude will be written  $\vartheta = \pi/2 - \theta$  and the ellipsoid aspect ratio  $\epsilon = p/q$ .

We can describe the ellipsoid as a deformed sphere of equivalent volume and radius  $R$ .

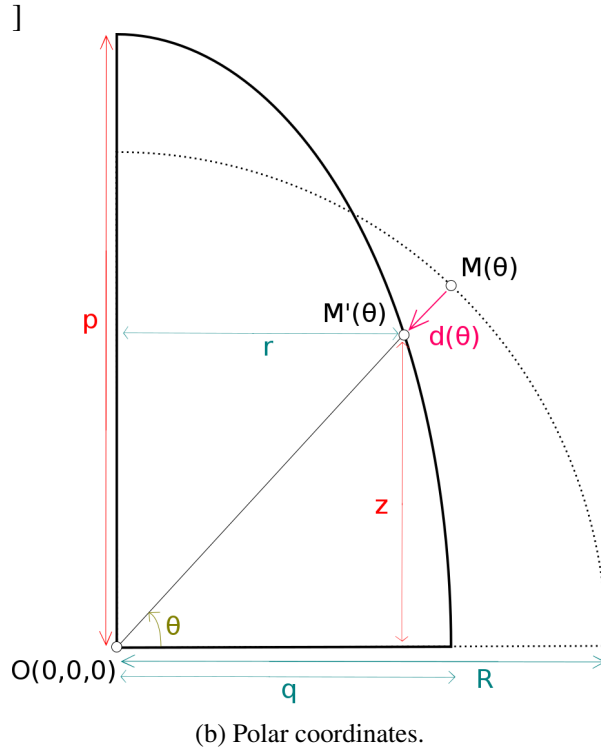
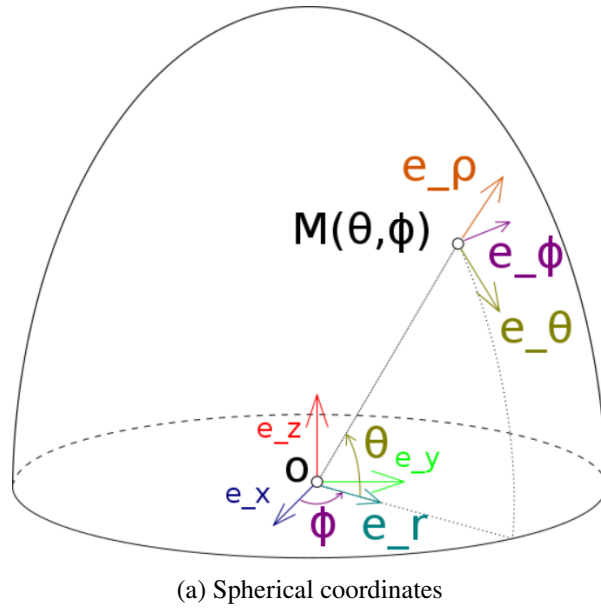


Figure 3.B.1: Spherical and polar coordinate system for the upper half deformed droplet.

Considering the following relationship between the  $\theta'$  parameter and the  $\theta$  coordinate,

$$\tan \theta = \frac{z}{r} = \epsilon \tan \theta',$$

and reducing the range of parameter  $\theta'$  to  $\theta' \in [0, \pi/2]$  thanks to system's symmetry, the

deformation  $\mathbf{d}(\theta)$  may be expressed as

$$\begin{aligned}\mathbf{d}(\theta) &= \mathbf{M} \mathbf{M}'(\theta), \\ &= (q \cos \theta' - R \cos \theta) \mathbf{e}_r + (p \sin \theta' - R \sin \theta) \mathbf{e}_z, \\ &= R \left[ \left( \frac{q}{R} \frac{\epsilon}{\sqrt{\epsilon^2 + \tan^2 \theta}} - \cos \theta \right) \mathbf{e}_r + \left( \frac{p}{R} \frac{\tan \theta}{\sqrt{\epsilon^2 + \tan^2 \theta}} - \sin \theta \right) \mathbf{e}_z \right].\end{aligned}$$

Denoting  $\bar{X} = X/R$  the scaling by droplet radius  $R$ , the deformation reduces to

$$\bar{\mathbf{d}}(\theta) = \frac{1}{\cos \theta \sqrt{\epsilon^2 + \tan^2 \theta}} (\bar{q} \epsilon - \cos \theta \sqrt{\epsilon^2 + \tan^2 \theta}) \mathbf{e}_\rho,$$

where  $\rho$  is the radial coordinate in the spherical coordinate system and  $\mathbf{e}_\rho$  is defined as  $\mathbf{e}_\rho = \mathbf{e}_r + \tan \theta \mathbf{e}_z$ . For small deformations, the ellipsoid remains close to a sphere and the aspect ratio may be expanded as

$$\epsilon = 1 + \bar{\eta}.$$

For now we suppose  $\bar{\eta}$  of small maximal amplitude  $\eta$  with no hypothesis concerning its time dependence. The Taylor expansion of the  $\theta$  dependance of  $\bar{\mathbf{d}}$  writes

$$\cos(\theta) \sqrt{\epsilon^2 + \tan^2 \theta} = \cos \theta \sqrt{2\bar{\eta} + \bar{\eta}^2 + 1/\cos^2 \theta} \underset{\eta \rightarrow 0}{=} 1 + \bar{\eta} \cos^2 \theta + \frac{\bar{\eta}^2}{2} \cos^2 \theta \sin^2 \theta + o(\eta^2).$$

Under the hypothesis of incompressible liquid, the volume of the deformed sphere must remains equal to the volume of the sphere, inducing

$$\frac{4}{3} \pi R^3 = \frac{4}{3} \pi p q^2 \Leftrightarrow \bar{q} \epsilon \underset{\eta \rightarrow 0}{=} 1 + \frac{2}{3} \bar{\eta} - \frac{\bar{\eta}^2}{9} + o(\eta^2).$$

As a consequence, the expression of deformation  $\mathbf{d}(\theta)$  may be expanded with respect to the droplet perturbation  $\eta$  as

$$\begin{aligned}\bar{\mathbf{d}}(\theta) &\underset{\eta \rightarrow 0}{=} \frac{1 + \frac{2}{3} \bar{\eta} - \frac{\bar{\eta}^2}{9} - (1 + \bar{\eta} \cos^2 \theta + \frac{\bar{\eta}^2}{2} \cos^2 \theta \sin^2 \theta) + o(\eta^2)}{1 + \bar{\eta} \cos^2 \theta + \frac{\bar{\eta}^2}{2} \cos^2 \theta \sin^2 \theta + o(\eta^2)} \mathbf{e}_\rho, \\ &\underset{\eta \rightarrow 0}{=} \bar{\eta} \left( \frac{3 \sin^2 \theta - 1}{3} + \bar{\eta} \frac{27 \sin^4 \theta - 33 \sin^2 \theta + 4}{18} \right) \mathbf{e}_\rho + o(\eta^2).\end{aligned}$$



Expressing the deformation with respect to the colatitude  $\vartheta$  yields

$$\begin{aligned}\bar{\mathbf{d}}(\vartheta) &\underset{\eta \rightarrow 0}{=} \bar{\eta} \left( \frac{3 \cos^2 \vartheta - 1}{3} + \bar{\eta} \frac{27 \cos^4 \vartheta - 33 \cos^2 \vartheta + 4}{18} \right) \mathbf{e}_\vartheta + o(\eta^2), \\ &\underset{\eta \rightarrow 0}{=} \frac{2\bar{\eta}}{3} \frac{3 \cos^2 \vartheta - 1}{2} \mathbf{e}_\vartheta + o(\eta), \\ &\underset{\eta \rightarrow 0}{=} \frac{2\bar{\eta}}{3} Y_2(\vartheta) \mathbf{e}_\vartheta + o(\eta),\end{aligned}$$

where  $Y_2$  represents the spherical harmonic  $Y_2^0$ . The initial condition of the weakly deformed ellipsoid droplet is thus similar to the analytical framework developped in [Subrahmanyam Chandrasekhar \(1981\)](#) such as similar dynamics can be expected asymptotically.

Approximating the  $\bar{\eta}$  volume-preserving ellipsoidal perturbation by the  $2\bar{\eta}Y_2/3$  spherical harmonic rises the question of volume conservation. Indeed, performing the exact integration on a  $2\bar{\eta}Y_2/3$  perturbed droplet lead to the following result

$$\begin{aligned}\int_0^{2\pi} \int_0^\pi \int_0^{R+2\bar{\eta}Y_2(\vartheta)/3} \varrho^2 \sin \vartheta d\varrho d\vartheta d\varphi &= 2\pi R^3 \int_0^\pi \frac{(1 + \bar{\eta}(3 \cos^2 \vartheta - 1)/3)^3}{3} \sin \vartheta d\vartheta, \\ &= \frac{4}{3} \pi R^3 \left( 1 + \frac{4}{15} \bar{\eta}^2 + \frac{16}{945} \bar{\eta}^3 \right).\end{aligned}$$

Volume conservation is only valid at first order in the perturbation  $\eta$  - this comes as no surprise given the linearisation - and thus must be, if necessary, enforced *a posteriori*. This will be done by imposing a spherical radius of the deformed sphere  $R_d$  slightly smaller than the reference  $R$ ,

$$R^3 = R_d^3 \left( 1 + \frac{4}{15} \bar{\eta}^2 + \frac{16}{945} \bar{\eta}^3 \right) \Rightarrow R_d \underset{\eta \rightarrow 0}{=} R \left( 1 - \frac{4}{45} \bar{\eta}^2 \right) + o(\bar{\eta}^2)$$

This leads to the volume preserving  $2\bar{\eta}Y_2/3$  expression for the deformed surface

$$r(\vartheta, t) = R \left( 1 + \frac{4}{15} \bar{\eta}^2 + \frac{16}{945} \bar{\eta}^3 \right)^{-1/3} \left( 1 + \frac{2}{3} Y_2(\vartheta) \bar{\eta} \right),$$

which remains equivalent to the ellipsoidal perturbation at first  $\eta$  order

$$r(\vartheta, t) \underset{\eta \rightarrow 0}{=} R \left( 1 + \frac{2\bar{\eta}}{3} Y_2(\vartheta) \right) + o(\eta).$$

This approximation is compared to the reference sphere and the exact ellipsoid on Figure [3.B.2](#).

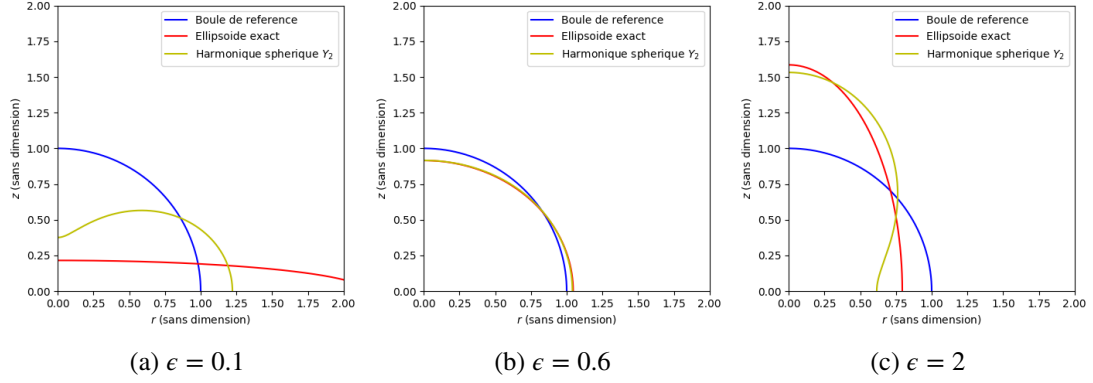


Figure 3.B.2: Numerical approximation of the ellipsoid in the  $(O, \mathbf{e}_r, \mathbf{e}_z)$  plane for different aspect ratios.

### 3.B.2 Kinetic energy

Approximating the initial ellipsoidal droplet by a sphere perturbed by an  $Y_2$  spherical harmonic allows the use of the analytical expressions developed by [Subrahmanyam Chandrasekhar \(1981\)](#) which are briefly recalled hereafter.

In the case of weakly spherical harmonic deformed bubble of non-viscous fluid of density  $\rho$  in a vacuum force-free surroundings only subject to capillarity effects with capillarity coefficient  $\sigma$ , the linearized equation of fluid dynamics can be explicitly solved and yields the following surface tension induced oscillations

$$\begin{aligned} u_{\varrho,l} &= -\sqrt{\frac{\sigma}{\rho} l(l-1)(l+2) R^{3/2-l-2} A \varrho^{l-1} Y_l} \sin(\omega_l t), \\ u_{\vartheta,l} &= -\sqrt{\frac{\sigma}{l\rho} (l-1)(l+2) R^{3/2-l-2} A \frac{\partial Y_l}{\partial \vartheta}} \sin(\omega_l t), \\ u_{\phi,l} &= 0, \end{aligned}$$

compatible with the given radial deformation  $r(\vartheta, t) = R + AY_l(\vartheta) \cos(\omega_l t)$ . Each chosen spherical harmonic  $Y_l$  then selects a precise pulsation given by the relation

$$\omega_l^2 = l(l-1)(l+2) \frac{\sigma}{\rho R^3}.$$

In the case of the ellipsoid surface approximation, the spherical harmonic  $Y_2$  is of special interest and yields

$$\omega_2 = \omega = \sqrt{\frac{8\sigma}{\rho R^3}},$$

and we will suppose  $\bar{\eta} = \eta \cos(\omega t)$  to comply with Chandrasekar's hypothesis set so that the

surface deformation will satisfy

$$r(\vartheta, t) \underset{\eta \rightarrow 0}{=} R \left( 1 + \frac{2\eta}{3} Y_2(\vartheta) \cos(\omega t) \right) + o(\eta).$$

The corresponding droplet dynamics is then given by the relation

$$u(\varrho, \vartheta, t) = \eta \sqrt{\frac{2\sigma}{\rho R^3}} \varrho \left( 2 \frac{1 - 3 \cos^2 \vartheta}{3} e_\varrho + \sin(2\vartheta) e_\vartheta \right) \sin(\omega t).$$

Writing  $\tilde{\eta} = \eta \sin(\omega t)$  and  $S_0 = 4\pi R^2$ , we can integrate the velocity field to obtain the kinetic energy

$$E_k(t) = \frac{1}{2} \int_0^{2\pi} \int_0^\pi \int_0^{r(\vartheta, t)} \rho u^2 d\varrho d\vartheta d\phi \underset{\eta \rightarrow 0}{=} \frac{8}{45} \sigma S_0 \tilde{\eta}^2 + o(\eta^2),$$

or, with the time dependency made explicit

$$E_k(t) \underset{\eta \rightarrow 0}{=} \frac{8}{45} \eta^2 \sigma S_0 \sin^2 \left( t \sqrt{\frac{8\sigma}{\rho R^3}} \right) + o(\eta^2).$$

### 3.B.3 Potential energy

The dynamics of the oscillating droplet is obtained by Chandrasekhar by considering the Laplace pressure given by the deformation as a boundary condition for the pressure field. Our purpose is here to underline the energy balance between the kinetic energy  $E_k$  and the surface energy  $E_p$  of the oscillating droplet. In order to do so, the  $2\eta Y_2/3$  perturbed droplet surface is evaluated as a function of time. Care must be taken that the expression of the perturbation has to be expanded to second  $\eta$  order to yield consistant results. Let us consider the second order corrected deformation

$$r(\vartheta, t) = R \left( 1 - \frac{4}{45} \bar{\eta}^2 \right) \left( 1 + \frac{3 \cos^2 \vartheta - 1}{3} \bar{\eta} \right),$$

with  $\bar{\eta} = \eta \cos(\omega t)$ . The corresponding mathematical surface is then given by the function

$$\mathbf{f}(\vartheta, \phi, t) = r(\vartheta, t) \mathbf{e}_\varrho = r(\vartheta, t) \begin{pmatrix} \sin \vartheta \cos \phi \\ \sin \vartheta \sin \phi \\ \cos \vartheta \end{pmatrix},$$

and leads to the following expression for a small surface element

$$\|\partial_\vartheta \mathbf{f} \times \partial_\phi \mathbf{f}\| = r \sin \vartheta \sqrt{(\partial_\vartheta r \cos \vartheta - r \sin \vartheta)^2 + (\partial_\vartheta r \sin \vartheta + r \cos \vartheta)^2} = r^2 \sin \vartheta \sqrt{1 + \left(\frac{\partial_\vartheta r}{r}\right)^2}.$$

Using the second order compatible deformation expression of  $r$ , the second order of the surface element writes

$$r^2 \sin \vartheta \sqrt{1 + \left(\frac{\partial_\vartheta r}{r}\right)^2} = R^2 \left( 1 + \frac{2}{3} \bar{\eta} (3 \cos^2 \vartheta - 1) + \bar{\eta}^2 \left( -\frac{1}{15} + \frac{4}{3} \cos^2 \vartheta - \cos^4 \vartheta \right) \right) + o(\eta^2)$$

The complete surface area of the oscillating droplet is obtained thanks to an analytic integration of surface elements which yields

$$S(t) = \int_0^{2\pi} \int_0^\pi \|\partial_\vartheta \mathbf{f} \times \partial_\phi \mathbf{f}\| d\vartheta d\phi \underset{\eta \rightarrow 0}{=} S_0 \left( 1 + \frac{8}{45} \eta^2 \cos^2(\omega t) \right) + o(\eta^2).$$

The potential energy  $E_p = \sigma(S - S_0)$  corresponding to the excess surface with respect to the minimal surface  $S_0$  then writes

$$E_p(t) \underset{\eta \rightarrow 0}{=} \frac{8}{45} \eta^2 \sigma S_0 \cos^2 \left( t \sqrt{\frac{8\sigma}{\rho R^3}} \right) + o(\eta^2),$$

which balances exactly the kinetic energy.

### 3.B.4 Mean curvature

In the case of droplet oscillation due to the surface tension effect, a natural parameter is the local mean curvature which gives birth to the Laplace pressure driving the fluid motion. In the derivation of Chandrasekhar, this Laplace pressure is imposed as a compatible boundary condition for the pressure field. In some studies, see [Herrmann \(2013\)](#), the local mean curvature is used as a parameter describing the interface non-equilibrium and giving birth to a spring-like force putting the interface into motion.

In the following, we detail the expression of local mean curvature in the case of a volume-preserving  $2\bar{\eta}Y_2/3$  perturbation. In the case of a surface of revolution, the local curvature is

given by the general formula

$$H = \frac{1}{2R \left(1 - \frac{4}{45}\bar{\eta}^2\right)} \left( \frac{r''z' - z''r'}{\left((r')^2 + (z')^2\right)^{\frac{3}{2}}} - \frac{z'}{r \left((r')^2 + (z')^2\right)^{\frac{1}{2}}} \right),$$

where the volume preserving correction is factorized to ease the computation, so that the surface coordinate in cylindrical coordinate may be directly used in the expressions. They and their derivatives write

$$\begin{aligned} z &= \left(1 + \frac{2}{3}\bar{\eta}Y_2\right) \cos \vartheta, & r &= \left(1 + \frac{2}{3}\bar{\eta}Y_2\right) \sin \vartheta, \\ z' &= -\left(1 + \frac{2}{3}\bar{\eta}Y_2\right) \sin \vartheta + \frac{2}{3}\bar{\eta}Y_2' \cos \vartheta, & r' &= \left(1 + \frac{2}{3}\bar{\eta}Y_2\right) \cos \vartheta + \frac{2}{3}\bar{\eta}Y_2' \sin \vartheta, \\ z'' &= -\left(1 + \frac{2}{3}\bar{\eta}(Y_2 - Y_2'')\right) \cos \vartheta - \frac{4}{3}\bar{\eta}Y_2' \sin \vartheta, & r'' &= -\left(1 + \frac{2}{3}\bar{\eta}(Y_2 - Y_2'')\right) \sin \vartheta + \frac{4}{3}\bar{\eta}Y_2' \cos \vartheta. \end{aligned}$$

The different terms needed in the local mean curvature expression may be detailed as follows

$$(z')^2 + (r')^2 = 1 + \frac{4}{3}\bar{\eta}Y_2 + \frac{4}{9}\bar{\eta}^2 \left( (Y_2)^2 + (Y_2')^2 \right),$$

and

$$r''z' - z''r' = 1 + \frac{2}{3}\bar{\eta}(2Y_2 - Y_2'') + \frac{4}{9}\bar{\eta}^2 \left( (Y_2)^2 + 2(Y_2')^2 - Y_2Y_2'' \right),$$

so that the local mean curvature of the volume-preserved  $2\bar{\eta}Y_2/3$  perturbation is given by

$$H = \frac{1}{2R \left(1 - \frac{4}{45}\bar{\eta}^2\right)} \left( \frac{2 + \frac{2}{3}\bar{\eta}(4Y_2 - Y_2'') + \frac{4}{9}\bar{\eta}^2 \left( 2(Y_2)^2 + 3(Y_2')^2 - Y_2Y_2'' \right)}{\left(1 + \frac{4}{3}\bar{\eta}Y_2 + \frac{4}{9}\bar{\eta}^2 \left( (Y_2)^2 + (Y_2')^2 \right)\right)^{\frac{3}{2}}} - \frac{\frac{2}{3}\bar{\eta}Y_2' \cos \vartheta}{\left(1 + \frac{2}{3}\bar{\eta}Y_2\right) \sin \vartheta \left(1 + \frac{4}{3}\bar{\eta}Y_2 + \frac{4}{9}\bar{\eta}^2 \left( (Y_2)^2 + (Y_2')^2 \right)\right)^{\frac{1}{2}}} \right).$$

The  $\eta$  second order expansion of the local curvature thus takes the form

$$\begin{aligned} H &\underset{\eta \rightarrow 0}{=} \frac{1}{R} \left( 1 - \frac{\bar{\eta}}{3} \left( 2Y_2 + Y_2' \frac{\cos \vartheta}{\sin \vartheta} + Y_2'' \right) + \frac{4\bar{\eta}^2}{9} \left( \frac{1}{5} + Y_2^2 + Y_2Y_2' \frac{\cos \vartheta}{\sin \vartheta} + Y_2Y_2'' \right) \right) + o(\eta^2), \\ &\underset{\eta \rightarrow 0}{=} \frac{1}{R} \left( 1 + \frac{2}{3}\bar{\eta}(3 \cos^2 \vartheta - 1) + \bar{\eta}^2 \left( -\frac{7}{15} + \frac{10}{3} \cos^2 \vartheta - 5 \cos^4 \vartheta \right) \right) + o(\eta^2), \end{aligned}$$

### 3 Two-phase flow models and where to find them

so that, mutliplying with the perturbed surface element, leads to the integrand

$$\|\partial_{\vartheta} \mathbf{f} \times \partial_{\phi} \mathbf{f}\| H(\vartheta) \underset{\eta \rightarrow 0}{=} R \left( 1 + \frac{4}{3} \bar{\eta} (3 \cos^2 \vartheta - 1) + \bar{\eta}^2 \left( -2 \cos^4 \vartheta + 2 \cos^2 \vartheta - \frac{4}{45} \right) \right) + o(\eta^2).$$

The integral of local mean curvature on the perturbed surface writes

$$\int_0^{2\pi} \int_0^{\pi} \|\partial_{\vartheta} \mathbf{f} \times \partial_{\phi} \mathbf{f}\| H(\vartheta) d\vartheta d\phi = 4\pi R \left( 1 + \frac{8}{45} \eta^2 \cos^2(\omega t) \right) + o(\eta^2),$$

so that the global mean curvature  $H$  may be splitted into a reference curvature  $H_0 = 1/R$  and an oscillating term

$$H = H_0 + \frac{8}{45R} \eta^2 \cos^2 \left( t \sqrt{\frac{8\sigma}{\rho R^3}} \right) + o(\eta^2).$$

Considering the potential energy derived in terms of the surface, an expression of potential energy in terms of the Mean curvature both complying to the analytical results of Chandrasekhar and the  $H$ -based approach of [Herrmann \(2013\)](#) would thus write,

$$E_p(t) = \sigma S_0 R (H - H_0) .$$

# 4

## A computational framework based on the discrete estimation of geometrical properties over triangulated interfaces to perform validation of two-phase flow models

Interfacial two-phase flows analysis, modeling and simulation usually requires the evaluation of geometrical local quantities related to the interface, such as local curvatures, but also some surface-averaged values leading to topological and geometrical invariants and the connection with disperse flow modeling based on the number density functions. In most of the existing approaches in the literature, these geometrical properties are obtained from direct evaluation from the level-set defined at the discrete level but fail to preserve the topological invariants of the existing objects. In this chapter we present a numerical strategy to evaluate geometrical properties of two-phase flows where an interface is represented through a level-set function at a discrete level, for example coming from a DNS simulation, designed in order to preserve the topological invariants. The strategy is implemented in the open-source library **Mercur(v)e** (Di Battista 2018). In the first part of the chapter, after recalling fundamentals of differential geometry of surfaces in 3D, we introduce the numerical strategy based on a triangulation of the interface and perform several verifications based on ideal test-cases. We assess the preservation at a discrete level of the Gauss-Bonnet theorem, that can be exploited to provide number density statistics on sprays of droplets homeomorphic to spheres. The drawbacks of the approach are highlighted: the discrete nature of the information on the level set yields noise in the evaluation of the geometrical properties, the semi-positivity of the local Willmore Energy is not guaranteed leading to outliers and too small / deformed triangles can produce difficulties with finite precision arithmetics. Solutions are proposed in order to fix these issues and will prove very useful for the applied part of the chapter. The proposed strategy is then applied for two purposes; first a DNS of the collision of two droplet is investigated with various levels of discretization and we show that the proposed approach is able to provide a precise evaluation of local quantities as well as to preserve topological invariants and statistics on geometrical properties of the interface. Then, the tool is used in order to design a new model for oscillating

droplets in an incompressible framework beyond the usual small perturbation context of the Taylor analogy. The proposed strategy clearly outperforms the more classical way of retrieving geometrical information of surfaces based on the differentiation of a level-set field.

The content of this chapter constitutes the material for a future publication: Ruben Di Battista, Thibault Ménard, Stephane De Chaisemartin, and Marc Massot (2021). “A Computational Framework Based on the Discrete Estimation of Geometrical Properties over Triangulated Interfaces Preserving Topological Invariants to Design and Validate Two-Phase Flow Models.” In: Fluids. In preparation.

## 4.1 Introduction

Many industrial processes and systems feature two-phase flow with a dynamic interface as main underlying phenomenon (fuel injection in aeronautical engines or automotive engines, design of spray nebulizers, nuclear safety aspersion system, irrigation sprinklers...). These flows are characterized by a wide range of time and space scales and their simulation is hence quite complex. For example, in the case of jet atomization in sub-critical conditions, the interface between the liquid and the gas experiences a dramatic change of topology from a separate-phase flow regime to a disperse flow regime, where the size spectrum of the produced droplets have a key impact on combustion regimes and pollutant formation.

Although Direct Numerical Simulations (DNS) have provided impressive results in this field (Shinjo and Umemura 2010; Desjardins et al. 2013; Ling, Zaleski, et al. 2015; Zandian et al. 2018; Vincent et al. 2019), they remain too costly in terms of computing resources to be applied in an industrial context or conduct parametric studies. Therefore, the only way to proceed is the design of predictive *reduced-order models*; it is still an important question since it has to cope with the predictive modeling of small scales of interfacial flows leading to the proper prediction of the polydisperse sprays, which are known not be easily resolved in the community and usually depend on the level of resolution.

Several authors have proposed various means to enrich these *reduced-order models* by introducing additional flow parameters that are reminiscent of small scale features that cannot be described by the bulk variables such as volume fraction or interfacial area density (Vallet and Borghi 1999; Devassy et al. 2015). Even the local mean curvature of the interface can be introduced to account for small scale dynamics connected to capillary effects (Herrmann 2013). A first step and methodology towards a unified Eulerian model describing both separated and disperse phases was introduced in Drui (2017); Drui, Larat, et al. (2019), where sub-scale modeling accounting for micro-inertia and micro-viscosity associated to bubble pulsation is proposed. The path was taken over in (Cordesse, Di Battista, Chevalier, et al. 2020) with an



attempt of introducing mean and Gauss curvatures in the sub-scale modeling. However, the development of these reduced-order models requires very often the knowledge of the evolution of quantities only available at the finest scales that are achievable only relying on DNS simulations. At this level, we have to use reliable tools in order to analyze such simulations either for model design or validation. It can even allow to improve the evaluation of interfacial forces in DNS! Based on [Essadki et al. \(2019\)](#), it can be shown that such tools have to preserve topological invariants.

In fact, the categorization of topological objects in the framework of turbulence and enstrophy iso-surfaces has been investigated in [Bermejo-Moreno and Pullin \(2008\)](#); [Bermejo-Moreno, Pullin, and Horiuti \(2009\)](#), where other geometrical parameters than mean and Gauss curvatures have been proposed as sound parameters for the correct categorization of these objects. Once a satisfying set of geometrical parameters is chosen, those parameters need to be computed on a discretized mesh (or triangulated surface). Once again, whatever the choice of relevant parameters, the accurate and computational-efficient estimation of curvatures is of high interest in the two-phase community, for example to correctly account surface tension effects ([Brackbill et al. 1992](#); [Evrard et al. 2020](#)). The Willmore energy, *i.e.* an integral functional of the local curvatures over the surface of an object (see eq. (4.18)) that describes the distance from sphericity, plays an important role in geometric modeling and processing and also in physical modeling ([Bobenko and Schröder 2005](#)). Thus, the evaluation of the geometrical properties from detailed simulations has to have some specific features in order to be reliable and the classical way of obtaining such quantities from the derivatives of a level-set function can be insufficient for modeling and analysis purposes. An in-depth review on different methodology for the estimation of geometrical interface properties is offered in [Bi et al. \(2021\)](#).

In this chapter we propose a methodology to evaluate such quantities from a discretized level-set function on a grid, thus allowing to perform post-processing of DNS simulations, and to extract geometrical properties of the interface between the two phases with specific properties, such as preservation of topological invariants. In particular, a library called `Mercur(v)e` has been developed and allows the characterization of two-phase interface geometrical features thanks to a triangulation of the interface and the calculation of geometrical properties from the discretized surface, while preserving topological and geometrical invariants of the considered objects. After introducing the key features of the numerical approach in a first part, we benchmark our approach on canonical objects. The purpose is two-fold: first, we assess the properties of the proposed approach and second, we identify the pitfalls of such an approach: noise related to the triangulation of a discretized level-set, presence of outliers

in the evaluation of the local Willmore energy and potential difficulty of convergence, when too deformed/small triangles are to be found and finite precision arithmetics errors come into play. We then introduce solutions in order to cope with each of these pitfalls. Starting from here, we tackle two representative cases in terms of applications. First we investigate a DNS simulation of the coalescence of two droplets and make use of the *Mercur(v)e* tool in order to analyze the dynamics of the interface. We, in particular, describe how the library allows to evaluate the statistics of the curvatures of the interface, while preserving the topological invariants. The proposed strategy should be able to discriminate singularities in the evaluation of the curvatures originating in the discrete level-set differentiation, from the ones coming from topological changes, each of them being clipped in most of the existing simulations tools. Eventually, we show how the library provides us with a reliable tool in designing a sub-scale model for incompressible droplet oscillations introduced in [Cordesse, Di Battista, Chevalier, et al. \(2020\)](#) and discussed also in section 3.5.3 by post-processing DNS simulations. The two DNS simulations are conducted in this chapter with the help of ARCHER code from CORIA and in collaboration with T. Ménard ([Thibault Ménard et al. 2007](#); [Benjamin Duret et al. 2012](#); [Vaudor et al. 2017](#); [Canu, Puggelli, et al. 2018](#))

## 4.2

## Estimation of differential geometry properties on surfaces

In this section we provide a general presentation of important parameters and properties of surfaces and the different strategies to represent them in a continuous description. We introduce geometrical properties that can be used to describe the topological evolution of objects and we recall some topological invariants, which will be useful in the sequel. This section does not aim at providing a complete overview of all the differential geometry results for surfaces, the interested reader is then referred to important works in the field for more details ([Drew 1990](#); [Deserno 2004](#); [Morel 2015](#); [Carmo 2016](#)).

Let us consider a 2D surface  $S$  embedded in a 3D spatial domain in  $\mathbb{R}^3$ ; generally two different approaches are employed to describe a surface:

- An **implicit** definition: the surface is described as the 0-isosurface of a scalar function, often a distance function (a level-set) or a color function.

$$S = \{\mathbf{x} \in \mathbb{R}^3 \mid \psi(\mathbf{x}) = 0\} \quad (4.1)$$

- An **explicit** definition: the surface is parameterized with two parameters  $u_1, u_2 \in \mathbb{U} \subset \mathbb{R}^2$ ,  $\mathbb{U} \in \mathbb{R}^2 \mapsto S, (u_1, u_2) \rightarrow \mathbf{X}(u_1, u_2)$

$$S = \{\mathbf{X}(u_1, u_2) \in \mathbb{R}^3, (u_1, u_2) \in \mathbb{U}\} \quad (4.2)$$

### 4.2.1 Geometrical properties for surfaces parametrized explicitly

In this section we will focus our attention on the explicit version of the description to provide the definitions we are seeking. Considering the explicit parametrization of a surface (eq. (4.2)), the tangent vectors are defined as:

$$\mathbf{e}_i = \frac{\partial \mathbf{X}}{\partial u_i}, \quad i = 1, 2 \quad (4.3)$$

the normal vector is then defined in terms of the tangent vectors:

$$\hat{\mathbf{n}} = \frac{\mathbf{e}_1 \times \mathbf{e}_2}{\|\mathbf{e}_1 \times \mathbf{e}_2\|} \quad (4.4)$$

We can then define the *second fundamental form*  $\Gamma_{ij}$  of a surface:

$$\Gamma_{ij} = \frac{\partial \mathbf{e}_i}{\partial u_j} \cdot \hat{\mathbf{n}} \quad (4.5)$$

whose eigenvalues are the *principal curvatures*  $k_1, k_2$ :

$$\text{eigs}(\Gamma_{ij}) = (k_1, k_2) \quad (4.6)$$

The principal curvatures (eq. (4.6)) allow to define intrinsic metrics of the curvature of the surface, the Gauss  $G$  and mean curvatures  $H$ ,

$$H = \frac{k_1 + k_2}{2}, \quad G = k_1 \cdot k_2 \quad (4.7)$$

### 4.2.2 Geometrical properties for surfaces parametrized implicitly with a level set function

Given a level-set  $\psi(\mathbf{x})$ , we consider that the value of the Level Set field is positive inside the object, negative outside. Hence, we define the normal unit vector:

$$\hat{\mathbf{n}} = -\frac{\nabla\psi}{\|\nabla\psi\|} \quad (4.8)$$

The curvature information is contained in the  $3 \times 3$  matrix  $\nabla\hat{\mathbf{n}}^T$ . Considering the Hessian matrix:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2\psi}{\partial x^2} & \frac{\partial^2\psi}{\partial x\partial y} & \frac{\partial^2\psi}{\partial x\partial z} \\ \frac{\partial^2\psi}{\partial x\partial y} & \frac{\partial^2\psi}{\partial y^2} & \frac{\partial^2\psi}{\partial y\partial z} \\ \frac{\partial^2\psi}{\partial x\partial z} & \frac{\partial^2\psi}{\partial y\partial z} & \frac{\partial^2\psi}{\partial z^2} \end{bmatrix} \quad (4.9)$$

the projection defined as  $\mathbf{P} = \mathbf{I} - \hat{\mathbf{n}}\hat{\mathbf{n}}^T$  projects the matrix on the tangent plane to the surface described by the function  $\psi(\mathbf{x}) = 0$  and then, as described in Kindlmann et al. (2003) and in Mitchell and Hanrahan (1992), it is possible to write the relationship:

$$\nabla\hat{\mathbf{n}}^T = -\frac{1}{\|\nabla\psi(\mathbf{x})\|}(\mathbf{P}\mathbf{H}) \quad (4.10)$$

The Hessian matrix describes how the gradient changes around the neighborhood of the points placed on an iso-surface of the function  $\psi(\mathbf{x})$ . In order to describe the curvature we are interested only in changes in the direction of the gradient. Hence we project  $\mathbf{H}$  on the tangent plane as in eq. (4.10). The restriction of the Hessian to the tangent plane is a symmetric matrix and it is possible to find an orthonormal basis  $\{\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, \hat{\mathbf{n}}\}$  able to diagonalize the matrix. In this basis we will obtain:

$$\nabla\hat{\mathbf{n}}^T = \begin{bmatrix} k_1 & 0 & \sigma_1 \\ 0 & k_2 & \sigma_2 \\ 0 & 0 & 0 \end{bmatrix} \quad (4.11)$$

$\hat{\mathbf{p}}_1$  and  $\hat{\mathbf{p}}_2$  are the eigenvectors associated to the principal curvatures, with eigenvalues  $k_1$  and  $k_2$ . The other two values  $\sigma_1$  and  $\sigma_2$  describes how the normal tilts. This aspect is called *flowline curvature*, further details can be found in Kindlmann et al. (2003); Carmo (2016). In our case the goal is to isolate  $k_1$  and  $k_2$ , and as proposed by Rumpf and Preußner (2002), we multiply  $\nabla\hat{\mathbf{n}}^T$  by  $\mathbf{P}$  in order to accomplish this task, obtaining the *Geometric Tensor*:

$$\mathbf{G} = \nabla\hat{\mathbf{n}}^T\mathbf{P} \quad (4.12)$$

The algorithm to compute the two principal curvatures can be summarized in the following steps:

1. Compute the normal  $\hat{\mathbf{n}}$  and the projection matrix  $\mathbf{P}$ ,
2. Compute the Hessian  $\mathbf{H}$  and the Geometric Tensor  $\mathbf{G}$ .
3. Analytically compute  $k_1$  and  $k_2$ .

$$k_1 = \frac{\text{tr}(\mathbf{G}) + \sqrt{2\|\mathbf{G}\|^2 - \text{tr}(\mathbf{G})^2}}{2} \quad (4.13)$$

$$k_2 = \frac{\text{tr}(\mathbf{G}) - \sqrt{2\|\mathbf{G}\|^2 - \text{tr}(\mathbf{G})^2}}{2} \quad (4.14)$$

from this it is possible to define the Mean and Gaussian curvatures as follows:

$$H = \frac{k_1 + k_2}{2} \quad (4.15)$$

$$G = k_1 \cdot k_2 \quad (4.16)$$

### 4.2.3 The Gauss-Bonnet theorem

The Gauss-Bonnet theorem yields an equality linking the integral value of the Gauss curvature of the surface of an object and the Euler characteristic, that is

$$\int_S G = 2\pi \chi \quad (4.17)$$

For example, for objects homeomorphic to spheres, the value of the characteristic is  $\chi = 2$ . Therefore, calculating the Gauss curvature over a discretized surface and dividing by  $2\pi$ , allows to estimate the number of objects that are homeomorphic to a sphere.

### 4.2.4 Semipositivity of the Willmore Energy

The Willmore energy is a surface energy functional defined as follows:

$$W = \int_S H^2 - G \quad (4.18)$$

It is an energetic description of the distance from the sphericity, where  $W = 0$ . It plays an important role in several different industrial and scientific contexts (Bobenko and Schröder 2005). Its local value on the surface at position  $\mathbf{x}$  for a continuous description is constrained by the following relation:

$$W_{\text{loc}}(\mathbf{x}) = H^2(\mathbf{x}) - G(\mathbf{x}) \geq 0 \quad (4.19)$$

## 4.3

### Curvature estimation from a discrete level-set volumetric field

---

In section 4.2 we introduced the fundamentals of the differential geometry of surfaces in the context of a continuous description. In this section, we present the framework that allows the computation of the discrete approximation of the geometrical properties introduced in section 4.2 and the associated algorithms that are then implemented into the `Mercur(v)e` library.

#### 4.3.1 General overview of the post-processing procedure

The post-processing procedure employed when using `Mercur(v)e` software can be briefly described by the following steps (and the associated sections in which they are described in more details):

1. We start from the volumetric field of a level-set discretized with a given underlying mesh (section 4.3.2). It can for exemple be generated through a possibly high-fidelity simulation. As an example, in this work, we use the ARCHER code (Thibault Ménard et al. 2007; Benjamin Duret et al. 2012; Canu, Christophe Dumouchel, et al. 2017; Vaudor et al. 2017) as the source of simulation data.
2. We triangulate the interface from the level-set field, using a triangulation algorithm such as the `Flying Edges (FE)` from William Schroeder et al. (2015) (section 4.3.3), which is conducted on the original mesh.
3. On the triangulated surface we compute the geometrical properties such as the mean and Gauss curvature (section 4.3.4) and we preserve topological invariant such as the one coming from the Gauss-Bonnet theorem (eq. (4.17))

### 4.3.2 One traditional way of estimating curvatures

One traditional way of estimating the curvatures of the interface is used in the ARCHER code and is representative of what is done in general: a level-set function is used for evaluating the normal and the curvature of the interface while the Volume of Fluid approach ensures the mass conservation of each component<sup>1</sup>. The computation of curvature is performed using the algorithm presented in Kindlmann et al. (2003) and shortly presented in section 4.2.2. Most of the time, in regions where the solution of the level-set geometry is well resolved or during the topological changes, very large values are produced and they are usually clipped since they impact the evaluation of capillarity forces in the dynamics of the interface. .

### 4.3.3 Discrete iso-contouring algorithms

In the previous section we described the data our method will rely on, that is a level-set function defined on a given mesh. The final artifact coming from a DNS simulation is indeed a volumetric scalar field of the level-set  $\psi(\mathbf{x})$  on the provided mesh. In order to compute the geometrical properties as discussed in section 4.3.4, the surface needs to be reconstructed via a triangulation routine. In this section we briefly present how Mercur(v)e performs the triangulation.

The triangulated surface is obtained applying a contouring procedure (William Schroeder et al. 2015) to the initial level-set given on the entire volume. This choice is mandated by the fact that the underlying data structure library VTK exposes an easy API for the iso-contouring and the performance of the routine is optimized for speed of execution. The volumetric level-set field is given on a regular uniform rectangular grid as the one shown in Fig. 4.1. We must note that the quality of the estimations are influenced not only by the quality of the strategy chosen to approximate the topological parameters, but also on the quality of the triangulation. Even though the Marching Cubes (MC) algorithm (with its variants as the FE algorithm used here) is a well-established method to obtain iso-surfaces of a field in a volume, there exist also more advanced implementations (Lewiner et al. 2003) of the MC algorithm that ensure additional

<sup>1</sup>In the ARCHER code, the two-phase medium considered is composed by two incompressible fluids separated by a sharp interface. The position of the interface and the description of its geometrical characteristics are computed by a Coupled Level Set/Volume of Fluid (CLSVOF) method. A key element of the numerical strategy implemented in ARCHER relies on solving a Poisson equation for the dynamic pressure that is performed using a MultiGrid preconditioned Conjugate Gradient algorithm (MGCG) (Zhang 1996) coupled with a Ghost-Fluid method (Fedkiw et al. 1999) to take into account the pressure jump due to the presence of surface tension. The time-integration is performed with a second-order Runge-Kutta scheme. For more information about the ARCHER solver, we refer the reader to Thibault Ménard et al. (2007); Benjamin Duret et al. (2012); Canu, Puggelli, et al. (2018); Vaudor et al. (2017). However, the only ingredient that is needed at this level is the data of a mesh and a level-set function on that mesh.

topological guarantees via an enriched lookup tables of possible triangle configuration within the tracking cubes that could improve the initial estimates for the needed quantities. Nonetheless, the level of improvement compared to the computational and implementation effort of these more advanced MC strategies needs to be assessed.

#### 4.3.4 Geometrical properties as 1-ring averaged values

In this section we describe the numerical strategies used to estimate the geometrical properties on a triangulated surface. The algorithm presented in this section is based on Meyer et al. (2003). The idea is to approximate the value of the mean curvature  $H$  and the Gauss curvature  $G$  on a vertex of the triangulated surface with the average value computed around the 1-ring neighborhood of said vertex. The 1-ring neighborhood of a point  $\mathbf{x}_i$  is the set of points  $\mathbf{x}_j$  that are directly connected to  $\mathbf{x}_i$ , Fig. 4.2 shows a graphical representation. The other quantities are derived from  $H$  and  $G$ . This strategy of computing geometrical properties has been implemented in a GPL-licensed library called Mercur(v)e Di Battista (2018), the details on the algorithm used to handle multi-block DNS are explained in section 4.A. In order to compute the mean curvature value, the algorithm leverages a local to the 1-ring neighborhood discretization of the Laplace-Beltrami (LB) operator whose continuous expression is reported in eq. (4.20).

$$\mathcal{LB}(\mathbf{x}) = 2H(\mathbf{x})\hat{\mathbf{n}}(\mathbf{x}) = \mathbf{K}(\mathbf{x}) \quad (4.20)$$

The LB operator can be approximated on a triangulated surface (Meyer et al. 2003) via eq. (4.21).

$$\mathring{\mathbf{K}}(\mathbf{x}_i) = \frac{1}{2\mathring{A}} \sum_j^{\text{1-ring}} (\cot \alpha_{ij} + \cot \beta_{ij}) (\mathbf{x}_i - \mathbf{x}_j) \quad (4.21)$$

With  $\mathring{\bullet}$  we denote the discrete estimation of a certain quantity on the mesh averaged around the 1-ring. Once the discrete LB has been computed, the scalar value of the mean curvature is simply the half of the norm of the LB vector.

$$\mathring{H} = \frac{1}{2} \|\mathring{\mathbf{K}}\| \quad (4.22)$$

$\mathring{A}$  is the equivalent 1-ring area: it is equal to the Voronoi region area if all the triangles that compose the 1-ring are non-obtuse, otherwise it is the region, the perimeter of which is given by all the circumcenters of the non-obtuse triangles and the midpoints of the edges opposed to the obtuse angles for the triangles that are obtuse. See Fig. 4.3 for clarity. The actual algorithm



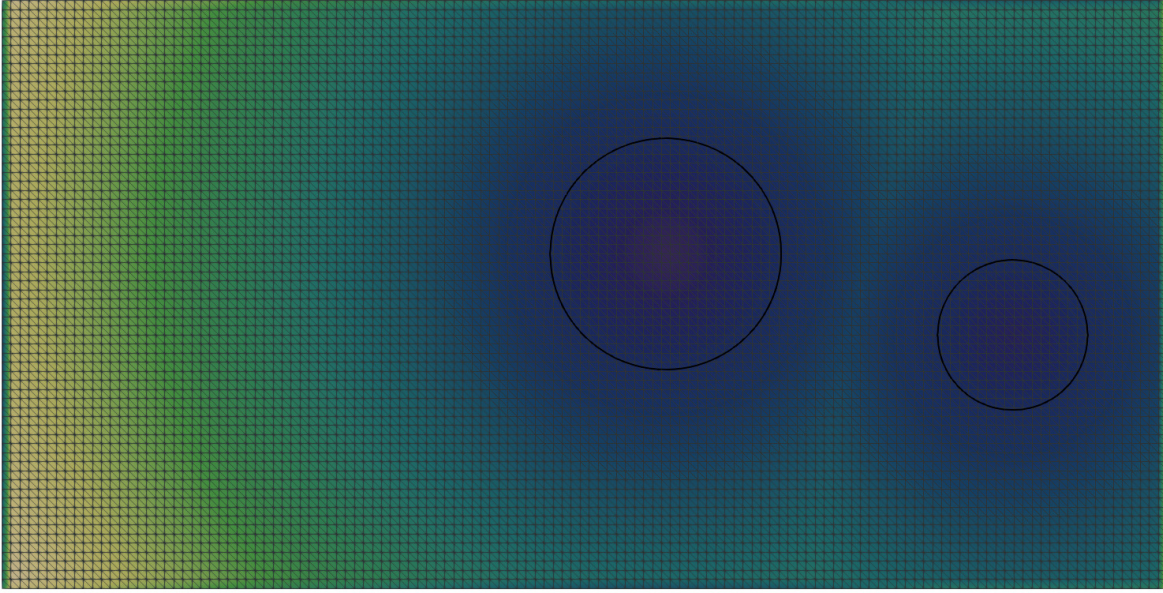


Figure 4.1: A central slice at  $\tau = 0$  of the volumetric level-set field on a uniform grid. The black lines are the 0-isoline of the level-set function

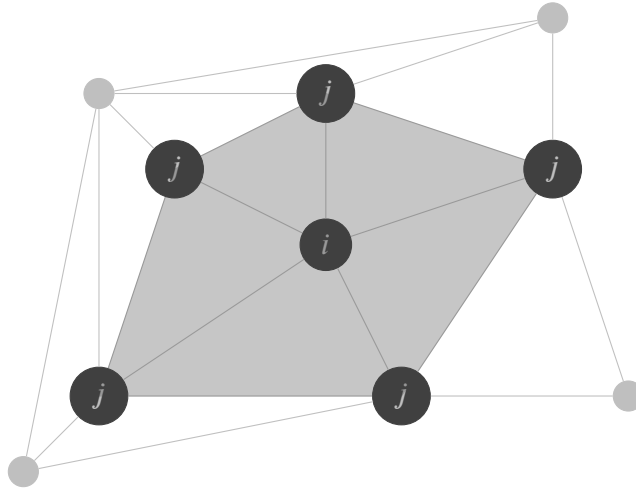


Figure 4.2: The 1-ring of a point  $\mathbf{x}_i$

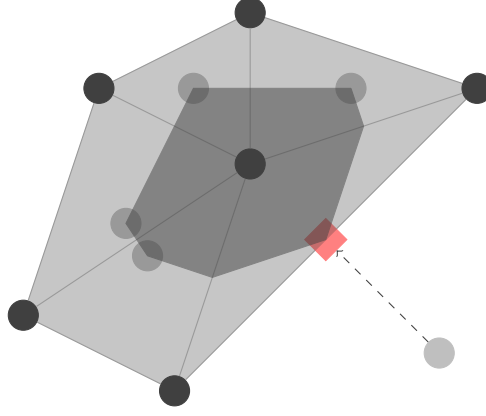


Figure 4.3:  $\hat{A}$  for a general case where one of the triangles is obtuse. The light gray area is the 1-ring neighborhood and the dark gray area is actually  $\hat{A}$

to compute  $\hat{A}$  is given in Meyer et al. (2003), and it is reported in algorithm 4.1 below for reference. What is interesting is to note that the area for a Voronoi region (*i.e.* the area of the polygon given by all the circumcenters of the triangles composing the 1-ring, all of them being non-obtuse) is given by eq. (4.23), where the  $(\cot \alpha_{ij} + \cot \beta_{ij})$  term can be reused successively in eq. (4.21) in the computation of the mean curvature, saving up on computational cost. The Voronoi area of a triangles is given by:

$$A_{\text{voronoi}} = \frac{1}{8} \sum_j^{\text{1-ring}} (\cot \alpha_{ij} + \cot \beta_{ij}) \|\mathbf{x}_i - \mathbf{x}_j\|^2 \quad (4.23)$$

Per each point  $\mathbf{x}_i$  of the surface, we need to compute an associated one-ring area  $\hat{A}$ . Let us consider a set of triangles connected to a generic point of the surface  $\mathbf{x}_i$  described by the triplet of points  $\mathbf{x}_{i\ell}, \ell = 1, 2, 3$ . This triplet can be defined arbitrarily and one of the three points will be the same point  $\mathbf{x}_i$  of the surface. For a given surface point  $\mathbf{x}_i$ , we loop over all the triplets composing the connected triangles to this point, if the triangle under examination has an obtuse internal angle, and its current vertex is not the same point as the surface point  $\mathbf{x}_i$  we are currently evaluating, then the area contribution associated to the triangle is computed as:

$$A_{i\ell} = A(\mathbf{x}_{i\ell}) = \frac{A_{\Delta}}{2}, \mathbf{x}_{i\ell} \neq \mathbf{x}_i \quad (4.24)$$

where  $A_{\Delta}$  is the area of the triangle under examination. If the vertex of the triangle  $\mathbf{x}_{i\ell}$  matches the same point as the one around which we are computing the 1-ring average area  $\hat{A}$ , then its area contribution reads

$$A_{i\ell} = \frac{A_{\Delta}}{4}, \mathbf{x}_{i\ell} = \mathbf{x}_i \quad (4.25)$$

In order to compute the value of the Gauss curvature  $G$  we can just rely on the definition of  $G$  for a geodesic polygon:

---

**Algorithm 4.1** Algorithm to compute the mixed area
 

---

```

1: procedure MIXEDAREA( $\mathbf{x}_i$ )                                ▷ We compute the  $dA$  on the 1-ring of  $\mathbf{x}_i$ 
2:    $\mathring{A} = 0$ 
3:   for  $T$  in ONERING( $\mathbf{x}_i$ ) do                                ▷ For each triangle composing the 1-ring
4:      $\hat{\mathbf{x}}_{i1}, \hat{\mathbf{x}}_{i2}, \hat{\mathbf{x}}_{i3} \leftarrow \text{ANGLES}(T)$         ▷  $\mathbf{x}_{i2}, \mathbf{x}_{i3}$  are the triangle vertices connected to  $\mathbf{x}_{i1} = \mathbf{x}_i$ 
5:      $A_\Delta \leftarrow \text{AREA}(T)$ 
6:     if  $\cot \hat{\mathbf{x}}_{i2} > 0$  or  $\cot \hat{\mathbf{x}}_{i3} > 0$  then
7:        $\mathring{A} += A_\Delta/4$ 
8:     else if  $\cot \hat{\mathbf{x}}_{i1} > 0$  then
9:        $\mathring{A} += A_\Delta/2$ 
10:    else
11:       $\mathring{A} += \frac{1}{8} |\mathbf{x}_{i1} - \mathbf{x}_{i2}|^2 \cot \hat{\mathbf{x}}_{i3} + |\mathbf{x}_{i2} - \mathbf{x}_{i3}|^2 \cot \hat{\mathbf{x}}_{i2}$ 
12:    end if
13:  end for
14: end procedure
    
```

---

**Definition 1**

Given a geodesic polygon close to a point  $\mathbf{x}_i$  of area  $A$ , the product of the Gauss curvature times the area of the polygon is equal to  $2\pi$  minus the defect of the exterior angles  $\epsilon_k$  of the polygon

$$G = \frac{1}{A} \left( 2\pi - \sum_k \epsilon_k \right) \quad (4.26)$$

Since for the triangulation of a surface the 1-ring neighborhood of a point  $\mathbf{x}_i$  is actually a special case of a geodesic polygon for which the edges are straight lines instead of geodesics, the same relation can be recasted for the internal angles  $\theta_j$ , as shown in eq. (4.27).

$$\mathring{G} = \frac{1}{\mathring{A}} \left( 2\pi - \sum_j^{\text{1-ring}} \theta_j \right) \quad (4.27)$$

Once the mean and Gauss curvature are calculated, the other typical differential geometry parameters can be retrieved from the values of  $H$  and  $G$ . For example the principal curvatures  $k_1$  and  $k_2$  are computed as follows:

$$k_1(\mathbf{x}_i) = \mathring{H}(\mathbf{x}_i) + \sqrt{\mathring{W}_{\text{loc}}(\mathbf{x})} \quad (4.28)$$

$$k_2(\mathbf{x}_i) = \mathring{H}(\mathbf{x}_i) - \sqrt{\mathring{W}_{\text{loc}}(\mathbf{x})} \quad (4.29)$$

This methodology allows to compute the Gauss curvature with a precision close to the machine resolution for floating points.

## 4.4 Verification and issues on canonical objects

In order to assess the correct implementation of the `Mercur(v)e` library, the efficacy of the algorithms as well as potential pitfall of the method, we perform three verifications on canonical objects. First, we consider two spherical objects with different radii, check that the Gauss-Bonnet formula is properly reproduced at the numerical level with our method and identify the presence of noise related to the datum of the level-set on a given mesh. Second, we perform a convergence study on canonical topological object that have analytical expressions for the mean and Gauss curvatures: a sphere and an ellipsoid. Once again the algorithm is working well, but some convergence issues are identified when too small/deformed triangles are used at the poles. We eventually apply the computation to a DNS and we compute an area-based PDF to highlight interesting footprints in the  $H - G$  phase space of the topological objects produced at each simulation time step; in particular, even if it provides interesting results, the positivity of the local Willmore energy is not always satisfied.

### 4.4.1 A set of verification cases on canonical objects and associated issues

Fig. 4.4 shows an example of computation for a pair of spherical objects and the associated curvature map. The computation of the integral of Gauss curvature on the surface of the two objects matches the Gauss-Bonnet up to machine precision. However, in a continuous case the two objects should be represented in the  $(\mathring{G}, \mathring{H})$  phase plane by two points, and a certain level of noise can be observed, which spreads the curvatures map around the analytical points. In order to assess the correct implementation of the `Mercur(v)e` library and the efficacy of the algorithms, we performed a convergence study on canonical topological object that have analytical expressions for the mean and Gauss curvatures: a sphere and an ellipsoid, the expression of mean and Gauss curvature of which are given by eqs. (4.30) and (4.31) and eqs. (4.32) and (4.33).

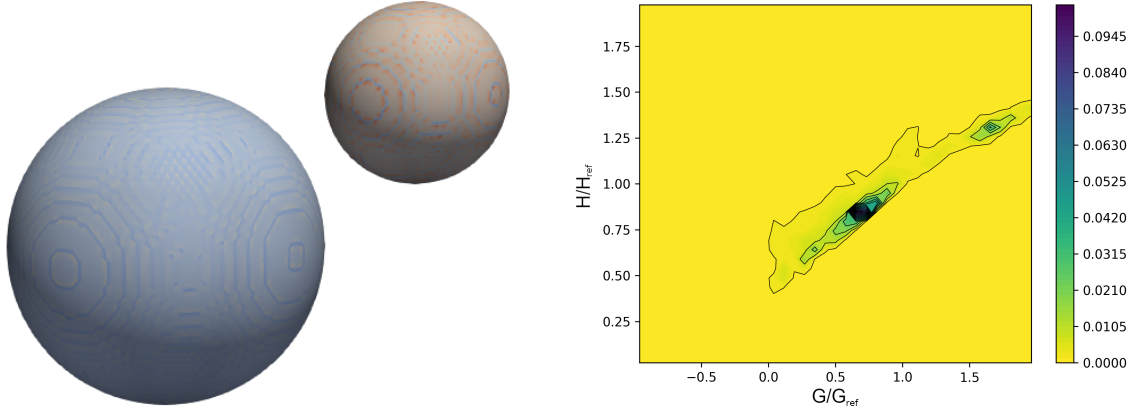


Figure 4.4: Computing curvatures on a couple of spherical objects

$$H_{\text{sphere}} = \frac{1}{R} \quad (4.30)$$

$$G_{\text{sphere}} = \frac{1}{R^2} \quad (4.31)$$

$$H_{\text{ellipsoid}} = \frac{|x^2 + y^2 + z^2 - p^2 - q^2 - r^2|}{\left[2(pqr)^2 \left(\frac{x}{p^4} \frac{y}{q^4} \frac{z}{r^4}\right)\right]^{\frac{3}{2}}} \quad (4.32)$$

$$G_{\text{ellipsoid}} = \frac{1}{\left[pqr \left(\frac{x}{p^4} \frac{y}{q^4} \frac{z}{r^4}\right)\right]^2} \quad (4.33)$$

These 3D objects are generated using superquadrics primitives from VTK library (Will Schroeder et al. 2006). The convergence study has been conducted gradually reducing the size of the triangles composing the discretized surface of these objects and calculating the cumulative relative area-weighted  $L^1$ -error  $\mathring{e}$  over all the points

$$\mathring{e} = \frac{1}{\mathring{A}} \sum_i^N \left( \mathring{C}_i - C_{\text{exact}} \right) \mathring{A}_i \quad (4.34)$$

Please note that the gradual reduction of size of the triangles composing the objects, as shown in Fig. 4.5, creates a dense accumulation of small triangles on the poles of the objects. Being  $N$  the number of points of the mesh,  $\mathring{A}$  the sum of the 1-ring cells areas

$$\mathring{A} = \sum_i^N \mathring{A}_i \quad (4.35)$$

and

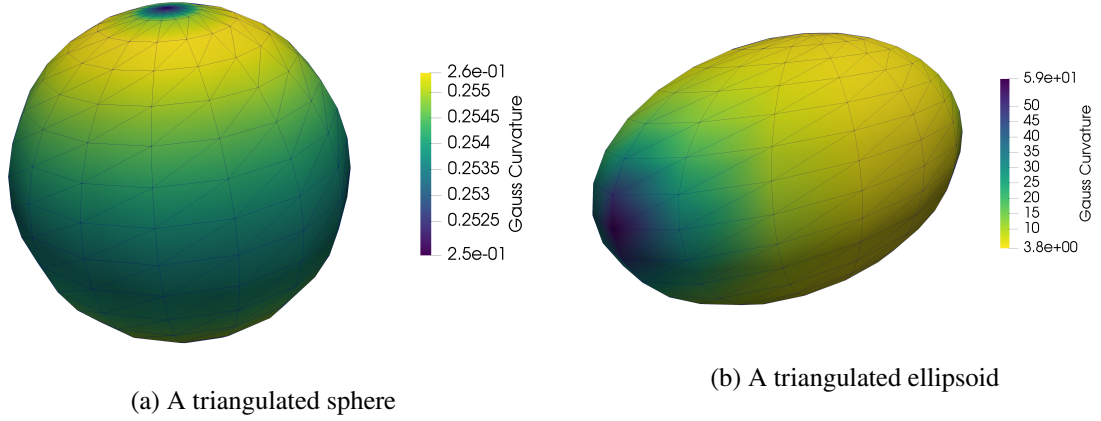


Figure 4.5: An example of meshes used to test the convergence on canonical objects

$$C \in (H, G) \quad (4.36)$$

An example of the mesh can be seen in Fig. 4.5.

As shown in Fig. 4.6, where we can see the absolute value of the relative error  $|\hat{e}|$  over all the points of the triangulated surface plotted against the number of points of the mesh, both for a sphere (section 4.4.1) and for an ellipsoid (section 4.4.1) respectively, the algorithm is capable of approximating with a reasonable accuracy the value of the curvatures up to a certain level of refinement. Beyond this threshold, the errors associated to finite floating point number representation prevail. This is highlighted in the Fig. 4.7, where we display the point number density in function of the 1-ring area  $\mathring{A}$  and a relative error for a sphere  $e_{\text{sphere}}$ :

$$\bar{n} = \bar{n}(\mathring{A}, e_{\text{sphere}}) \quad (4.37)$$

where the definition of the relative error leverages the fact that for a sphere  $H = \sqrt{G}$  exactly.

$$e_{\text{sphere}} = \frac{H - \sqrt{G}}{\sqrt{G}} \quad (4.38)$$

The error increases along a mono-dimensional manifold from big to very small cells, having optimal values for medium-sized cells. The phenomena that are leading to the generation of the error are different:

- For bigger cells, the error is associated to the fact that a 1-ring cell composed by big triangles is a poor approximation of the local surface
- For very small cells, the error is associated to floating point rounding errors produced

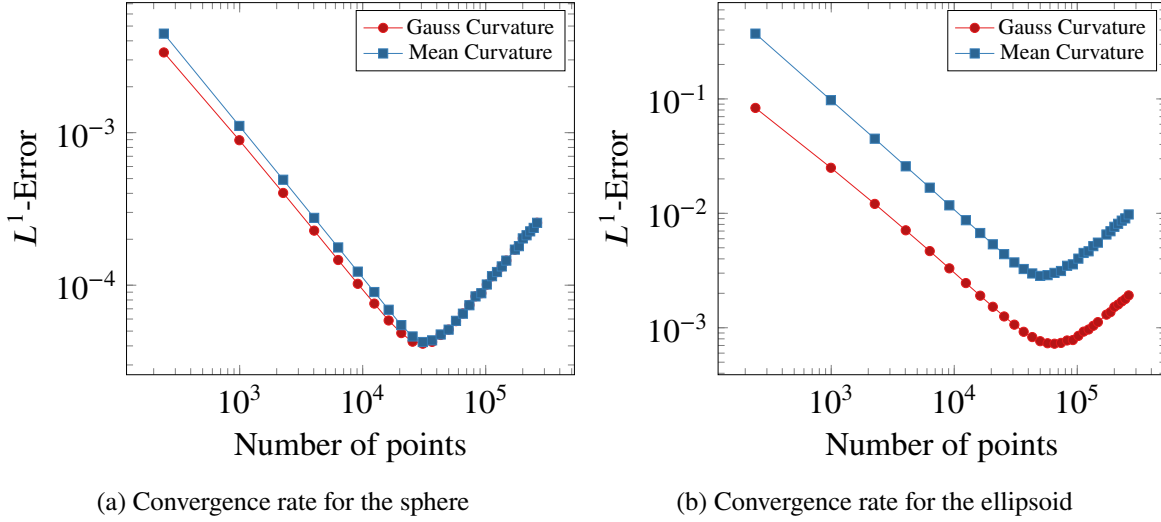


Figure 4.6: Convergence rate of the curvature computation

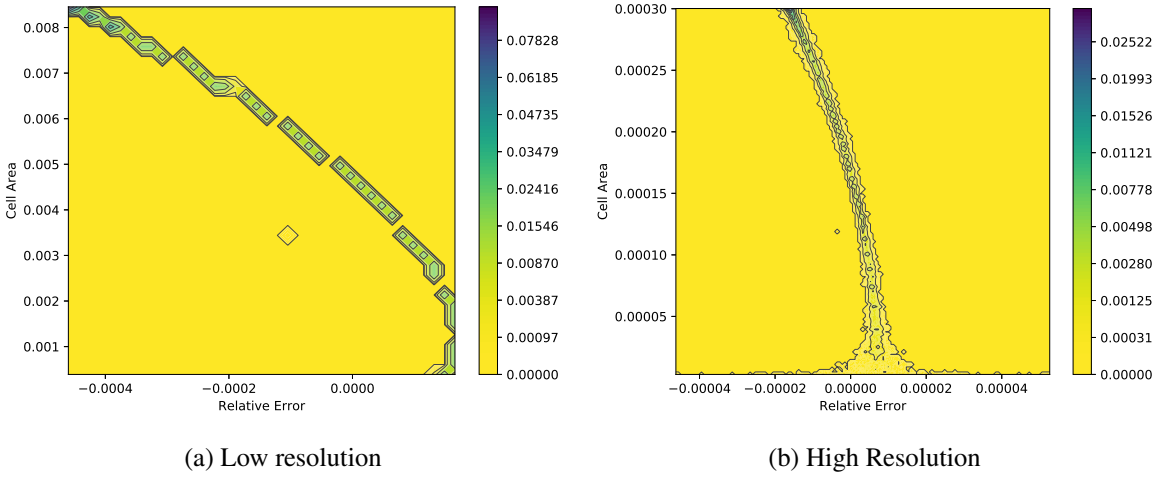


Figure 4.7: Error map at two different resolutions for a sphere

by the calculation of very small angles. In particular if we look at Fig. 4.5a, we can note that the triangles at the poles are smaller *w.r.t.* triangles on other latitudes and, indeed, looking at the error map for the finest resolution at Fig. 4.7b, it is clear that for very small cells the error smears out in both, positive and negative, directions, an indication of rounding errors.

The third and final problem is the one related to non positivity of the local Willmore energy. A problem that arises, unfortunately, because  $H$  and  $G$  are calculated independently one from the other. That means  $\dot{W}_{\text{loc}}$  can be  $< 0$  in the discrete case while in the continuous situation, it is always  $\geq 0$ . This problem is mentioned in Meyer et al. (2003) to be an “extremely rare occurrence”, while in our computations this happens quite often (in section 4.5.3 we perform a



thorough analysis on this subject *w.r.t.* the collision of two spherical objects), especially when the topological object under investigation is close to a sphere.

Thus, even if the proposed numerical strategy comes with obvious advantages, there are some issues that need to be fixed before tackling applications.

## 4.4.2 Solutions to the identified issues

### 4.4.2.1 An averaging kernel that preserves the Gauss Bonnet Theorem

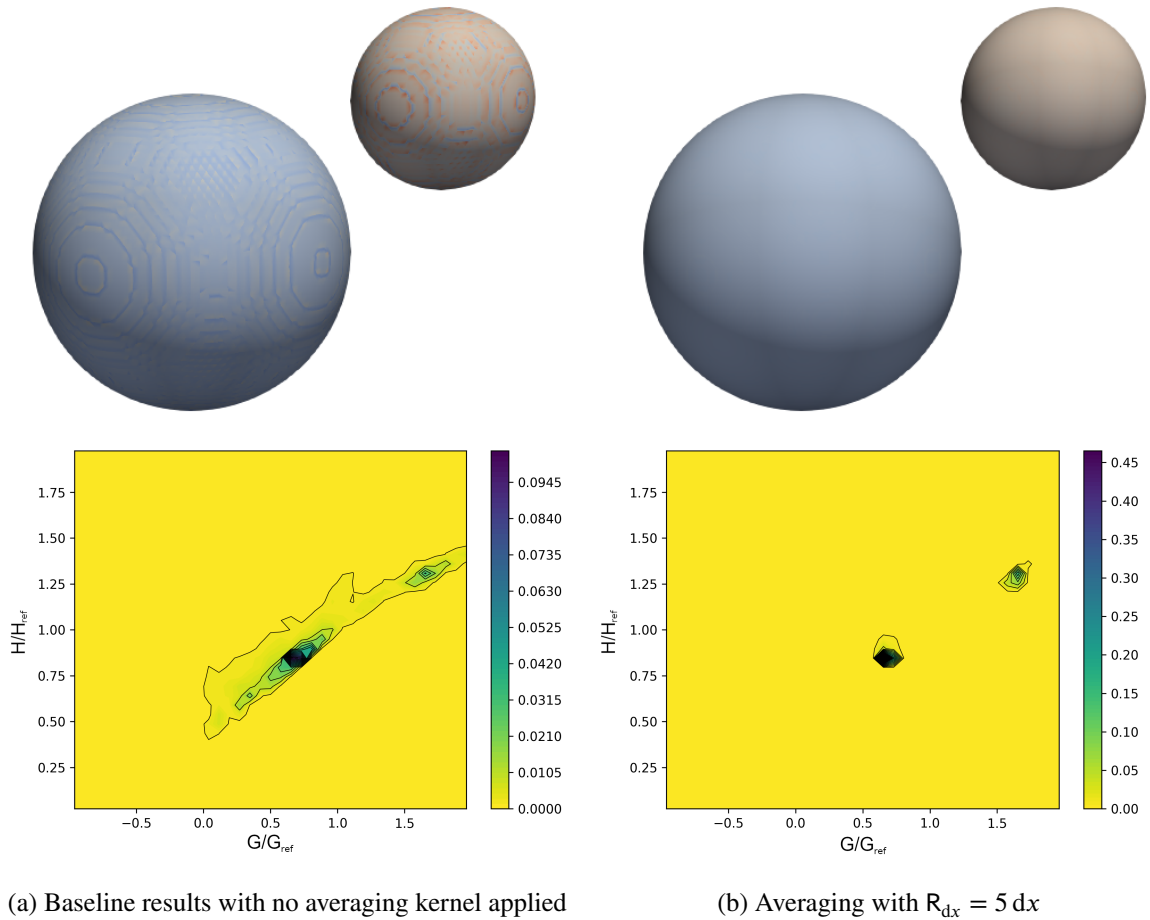


Figure 4.8: Computing curvatures with and without applying the averaging kernel

Often, when performing the **MC** procedure over the level-set volume field in order to generate the triangulated surface, the final surface can present some noise associated to the poor triangulation in certain zones. This, in turn, produces noise in the computed curvature fields when using the 1-ring algorithm presented in section 4.3.4. Another pathological situation that



can arise during a simulation is when extreme breakups or collisions occur: they can sometimes lead to topological transformation of the objects with local curvature that can be very high. What is classically done in literature to mitigate this problem and avoid numerical instabilities is to clip artificially the local value of the curvature to a somewhat arbitrary threshold, as for example in the case of curvature computation in the ARCHER code (Vaudor et al. 2017). In this work we introduce instead a regularization strategy based on applying a specific averaging kernel.

The curvature computation on triangulated interfaces we present here, as we already mentioned, has the property of conserving the Gauss-Bonnet theorem over the entire object at machine precision, hence when applying this averaging kernel, we are interested in keeping this property still valid. `Mercur(v)e`, then, implements the strategy proposed in Essadki et al. (2019) that provides a way of averaging around a surface point without losing the exact evaluation of the Gauss-Bonnet theorem at discrete level. The averaged value of a generic curvature  $\langle \mathring{C} \rangle_i = \langle \mathring{C}(\mathbf{x}_i) \rangle$ ,  $C \in (H, G)$  of a point  $i$  of the surface is computed as follows:

$$\langle \mathring{C} \rangle_i = \sum_k^{\mathcal{N}(\mathbf{x}_i, R_{dx})} \mathring{C}_k \frac{\mathring{A}_k}{\mathring{S}_k} \quad (4.39)$$

where  $\mathcal{N}(\bullet, R_{dx})$  is the set of neighbor points of the point  $\bullet$  within a radius  $R_{dx}$ .  $S_k$  is defined as:

$$\mathring{S}_k = \sum_l^{\mathcal{N}(\mathbf{x}_k, R_{dx})} \mathring{A}_l$$

Fig. 4.8 shows an example of the application of the averaging kernel. Fig. 4.8a is the condition in which no averaging is applied. As it is possible to note, the (Gauss) curvature field mapped on the surface object is grainy and noisy. This results in a area-weighted **NDF** that is more spread out *w.r.t.* the second case, at Fig. 4.8b, in which an averaging kernel within a radius of five times the grid size  $R_{dx} = 5 dx$  is applied. The map shows that the two spherical objects are more correctly identified as two individual points in the phase space  $(H, G)$  (normalized by a reference radius  $R_{ref} = 0.5(R_1 + R_2)$ ). Thus the averaging process represent one first cure to the problem of noise. However, this is also due to the presence of deformed/small triangles and we refer to the third possible cure for another and complementary way of dealing with this issue, see section 4.4.2.3.

#### 4.4.2.2 A projection method

A possible mitigation we propose in order to deal with the presence of non-positive Willmore energy is constituted of two ingredients. First, whereas the evaluation of the Gauss curvature at the vertices is fully legitimate, the evaluation of the mean curvature normally pertains more to the edges connecting the points of the triangulated surface than to the individual points and the associated 1-ring area. In our approach, the evaluations of the Gauss curvature and the mean curvature are independent, and the value  $\mathring{H}^2$  is computed squaring the value of  $\mathring{H}$ . Another way of computing of  $\mathring{H}^2$  should be used in order to evaluate the Willmore energy, possibly with the possibility of taking into account the correct value of  $\mathring{G}$  that, as we have shown, can be estimated at machine precision. However, this is not sufficient since the two curvature are not evaluated in a fully coherent manner. Thus, another possible mitigation is to re-project all the points that do not fulfill the constraint on  $\mathring{W}_{\text{loc}}$ , on the curve  $\mathring{W}_{\text{loc}} = 0$ , while preserving the Gauss quadrature evaluated at the vertex. This projection has to be conducted on both  $\mathring{H}$  and  $\mathring{H}^2$ .

#### 4.4.2.3 Improving the quality of the mesh

In order to take care of the noise associated to the discrete estimation of the curvature values, as well as the convergence difficulties we have identified previously, a complementary approach is to remesh the surface, while improving the quality of the triangulation, taking care of not accumulating small triangles locally on the surface. Classical approaches like the Laplacian smoothing can have pitfalls regarding the non conservation of the volume of the objects, leading to changes into the topological configuration. A work on the subject is in progress with the MMG team (Froehly et al. 2019) and requires quite a bit of effort in order to quantify its impact on the global geometrical parameter evaluation proposed in the coming sections. For the time being, we will only employ in the following the averaging kernel mitigation in order to deal with the potentially too-deformed/small triangles produced by our triangulation algorithm.

## 4.5 An application of the post-processing of DNS simulations: the collision of two droplets

The first application we envision is the collision of two spherical droplets with a radius respectively of  $130\mu\text{m}$  and  $200\mu\text{m}$  as a test case to our numerical strategy augmented with the proposed solutions described previously. The aim is three-fold: evaluate the statistical topolog-

ical footprint, verify that the Gauss-Bonnet topological invariant is well-preserved and allows to count the number of object homeomorphic to sphere, when they are present (beginning and end of the process) and finally evaluate the constraint given by eq. (4.19), which is not guaranteed when using the algorithm we propose (Meyer et al. 2003) on a discretized representation of the surface. In order to assess the impact on a test case of interest, we decided to analyze the percentage of points for which eq. (4.19) is not respected over the totality of the mesh points all along the course of a simulation.

### 4.5.1 DNS configuration

The two droplets are configured at the beginning of the simulation with a negative relative velocity one *w.r.t.* the other. The two spherical objects, then, start to approach until they collide causing surface deformation and topology changes. The simulation of the droplets collision is a DNS performed using the ARCHER code (Thibault Ménard et al. 2007). In order to investigate the influence of the mesh resolution on the topological configurations we used three different mesh refinement levels: a coarse configuration with  $128 \times 128 \times 256$  cells, a medium case with  $256 \times 256 \times 512$  cells and a fine case with  $512 \times 512 \times 1024$  cells. Fig. 4.9 shows three snapshots of the evolution of the system.

### 4.5.2 Topological footprint analysis

We characterize the footprint of each topological object with an area-weighted NDF in the phase space  $\mathring{G} - \mathring{H}$ :

$$\bar{n} = \bar{n}(\mathring{G}, \mathring{H}) \quad (4.40)$$

$\mathring{A}(\mathring{G}, \mathring{H})$  is the area associated to a point in the phase space having curvatures  $(\mathring{G}, \mathring{H})$  and it is used as a weighting function for the NDF. This NDF is computed on the data after filtering out outliers for which  $\mathring{W}_{\text{loc}} < 0$  with a number of bins  $n \sim \sqrt{N}$  being  $N$  the total number of points in the surface. The corresponding values of  $\mathring{H}, \mathring{G}$  are normalized with the equivalent curvatures defined as in eq. (4.41).

$$\begin{aligned} R_{\text{eq}} &= \frac{R_1 + R_2}{2} \\ H_{\text{eq}} &= \frac{1}{R_{\text{eq}}} \\ G_{\text{eq}} &= \frac{1}{R_{\text{eq}}^2} \end{aligned} \quad (4.41)$$

where  $R_1, R_2$  are the radiuses of the two spheres at rest. In Fig. 4.11 the NDF  $\bar{n}(\mathring{G}, \mathring{H})$  is shown for three different time steps  $t = (0, 0.3, 0.85 \text{ ms})$ . At the beginning of the simulation (Fig. 4.11a), two points are visible on the phase space associated to the constant values of  $\mathring{H}/\mathring{H}_{\text{ref}}$  and  $\mathring{G}/\mathring{G}_{\text{ref}}$  for the two spherical particles  $(\mathring{H}/\mathring{H}_{\text{ref}}, 1.26, 0.825)$ ,  $(\mathring{G}/\mathring{G}_{\text{ref}}, 1.61, 0.68)$ . At the moment of the collision (Fig. 4.11b) the smaller droplets coalesce into the bigger one, and this can be easily tracked in the phase space at Fig. 4.11b where the NDF at the point  $(\mathring{G}, \mathring{H}) = (1.61, 1.26)$  has a lower value and around the point  $(0.68, 0.825)$  it is more spread out *w.r.t.* the beginning of the simulation (Fig. 4.11a) with the two independent non-interacting droplets. In the following time steps the topology of the coalesced object strongly changes, and this is reflected on the phase space with a cloud of points that accumulates all along the curve  $\mathring{H} = \sqrt{\mathring{G}}$ , and in Fig. 4.11c we can note that the peak of the NDF is at a lower value of curvatures *w.r.t.* the initial case (Fig. 4.11a). The main mesh resolution effect is the reduction of the noise, see Fig. 4.10, *i.e.* points tend to stay in the physical part of the phase space ( $\mathring{H} > \sqrt{\mathring{G}}$ ) as shown also in Fig. 4.12a and close along the limit curve. This behavior can be related to the effect of the refinement of the mesh which allows a more accurate estimation of the mean curvature *w.r.t.* coarser cases. This also reflects on the fact that the topological configuration changes (we have a different number of satellite objects depending on the refinement of the mesh).

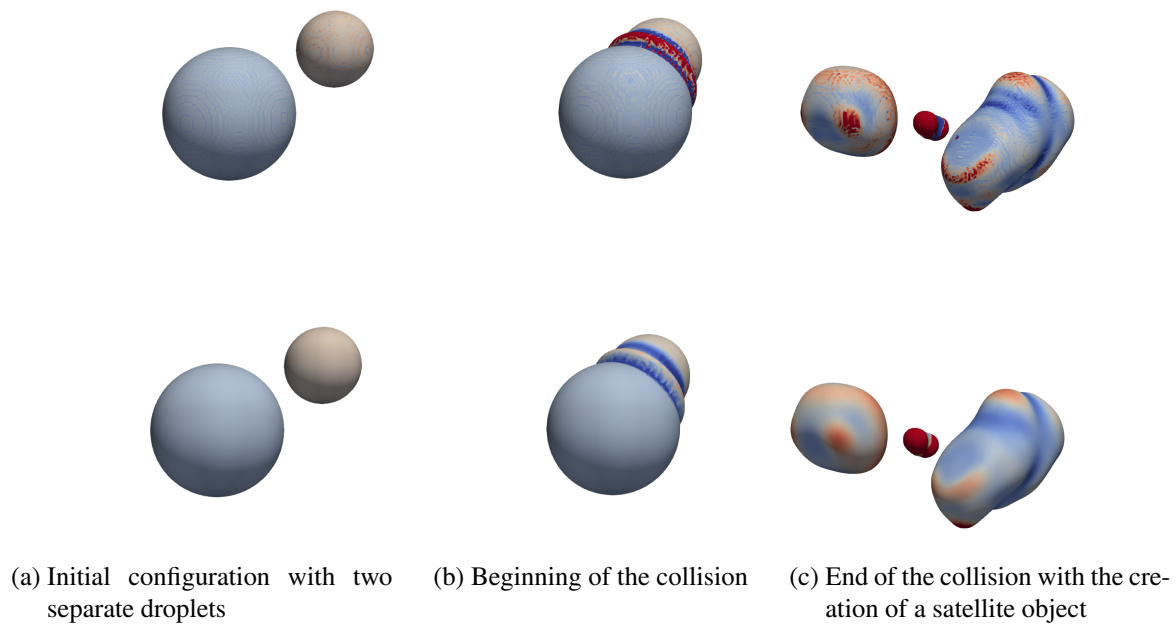


Figure 4.9: The collision sequence of two droplets. The top row shows the computation without average. The bottom row is with an averaging radius  $R_{dx} = 5 \, dx$

4 A computational framework based on the discrete estimation of geometrical properties over triangulated interfaces to perform validation of two-phase flow models

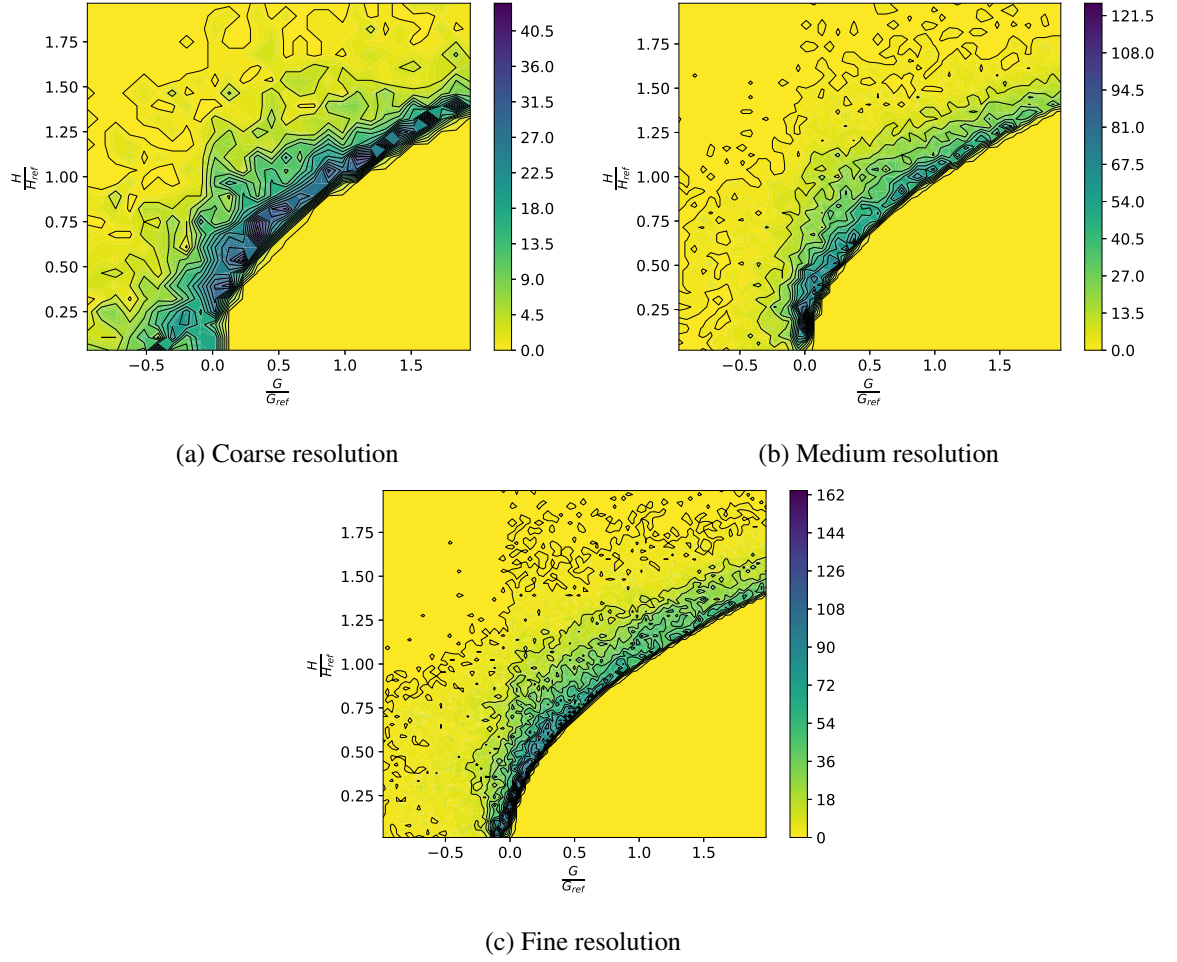


Figure 4.10: The NDF computed with three different resolutions at the final snapshot

#### 4.5 An application of the post-processing of DNS simulations: the collision of two droplets

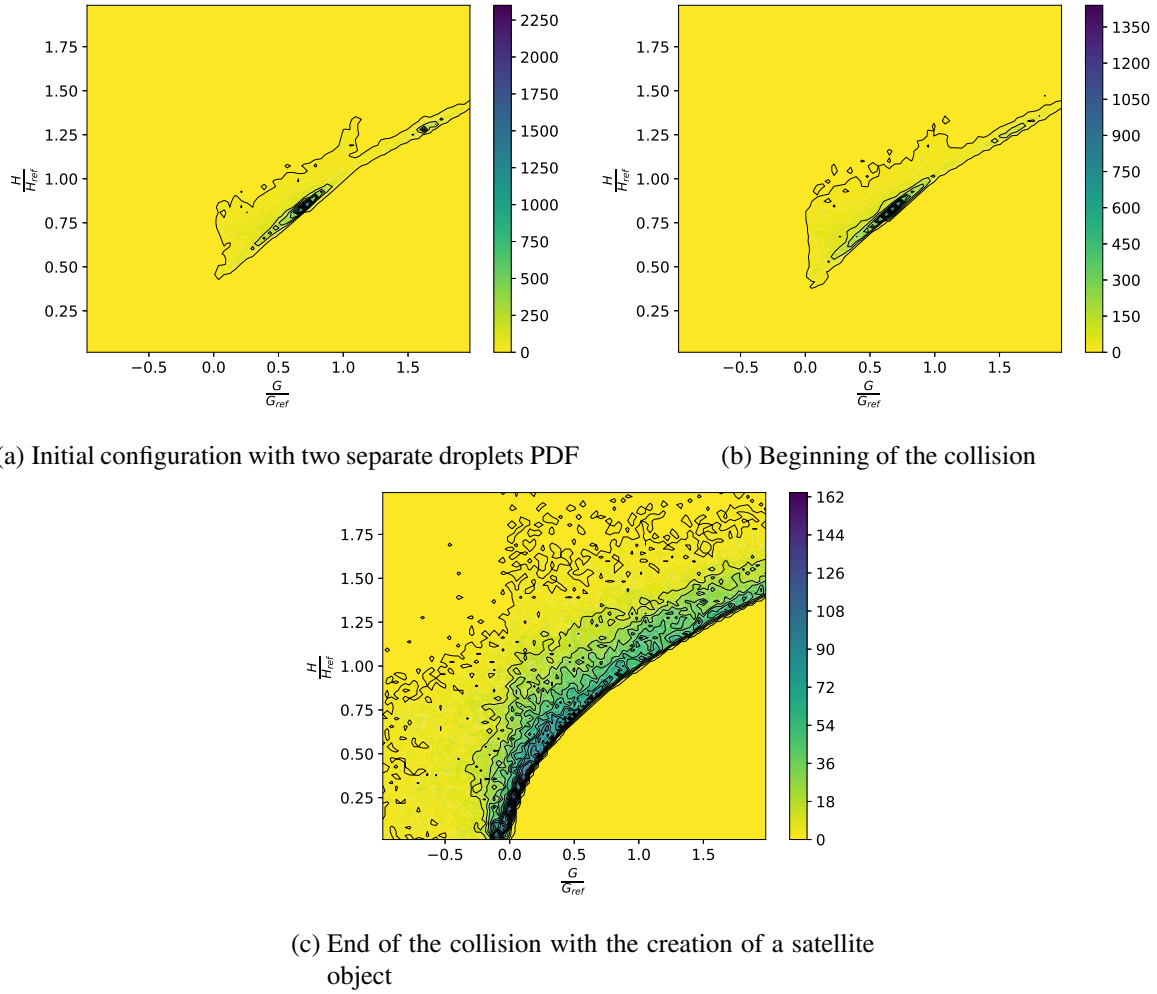
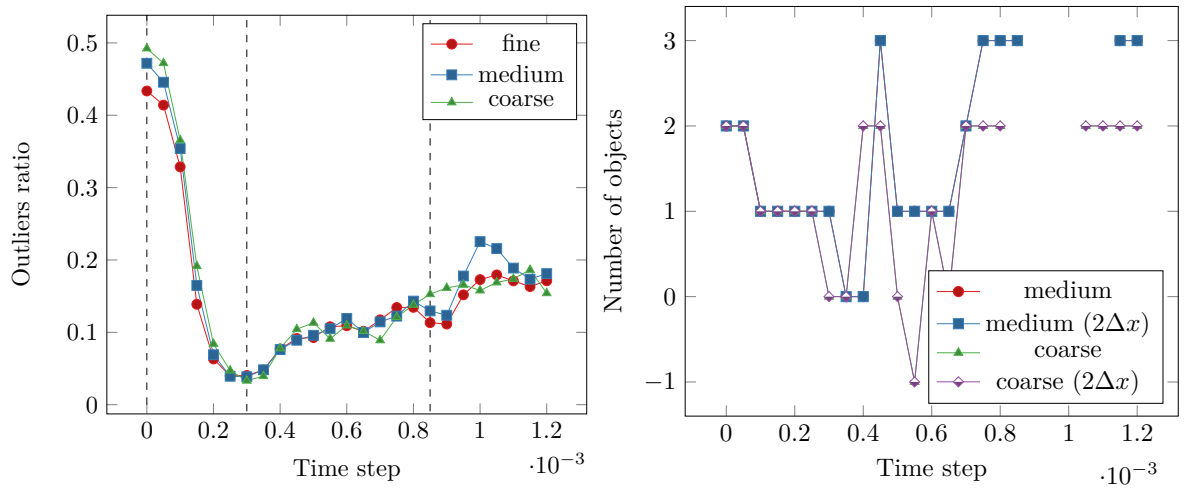


Figure 4.11: The collision sequence of two droplets in terms of the NDF



(a) The percentage of points for which eq. (4.19) is not respected. The vertical dashed lines correspond to the time instants in which the snapshots shown in Fig. 4.9 are taken

(b) The number of objects estimated via the Gauss-Bonnet theorem (eq. (4.17)), with and without averaging kernel

Figure 4.12: Statistics on the geometrical properties computed on the collision of two droplets



### 4.5.3 Willmore Energy analysis

We performed an outliers statistics analysis: we tracked all the points that do not respect eq. (4.19) along the time evolution of the simulation. In Fig. 4.12a we can appreciate how outliers are more probably produced when the topological objects resemble a sphere, in facts at the beginning of the simulation, when we have two exact spheres in the domain, *i.e.* the most critical situation since for a sphere eq. (4.19) is exactly equal to 0, the outliers are large in absolute terms, up to  $\sim 50\%$  for the coarsest configuration to  $\sim 45\%$  for the finest one. Once the objects is deformed after the collision, the outliers percentage steeply drops to  $< 10\%$  and stays under 20% for the entire simulation. It is interesting to highlight how the number of outliers is not influenced by the triangulation refinement level but it is quite exclusively associated to the intrinsic topological configuration of the object: the closer the object to a sphere, higher the number of outliers.

#### 4.5.3.1 Counting object homeomorphic to a sphere

Fig. 4.12b instead shows the evolution of the number of objects along the simulation computed using the Gauss-Bonnet theorem (eq. (4.17)) and the Gauss curvature as for eq. (4.27). The number of objects is correctly tracked for situations in which the topological configuration of the two colliding droplets stays homeomorphic to a sphere, *i.e.* at the beginning we start with the two spherical droplets, then they collide and become a single object before producing a torus around  $t = 0.3s$ , *i.e.* a topological entity that is not homeomorphic to a sphere. The toroidal objects deforms until  $t > 0.7s$ , in which again the topological configuration comes back to be homomorphic to a sphere. It is interesting to note how the averaging kernel has absolutely no effect on the correct respect of the Gauss-Bonnet Theorem and hence on the objects number estimation.

Please note that in Fig. 4.12b, for the time steps close to the interval  $t \in (0.8s, 1s)$ , the results are omitted because the objects cross the periodic boundaries and for the points at the intersection with the boundary surface, the curvature estimation is not correct since the object becomes unclosed. Fixing this implementation issue is one of the future axis of improvement for the algorithm here presented and does not impact any of the conclusion we have drawn so far.

## 4.6

# Post-processing of DNS as a modeling tool: the oscillation of an ellipsoidal droplet

## 4.6.1 Introduction

Among the various classical two-phase flow historical configurations, the oscillations of a single droplet and/or bubble plays a significant role and has been tackled really early on in the field. It naturally connects the spherical state at equilibrium to the large deformation leading to break-up in the presence of shear. Indeed, once the final break-up occurs, the dynamics of the fluid inclusion tend towards a perturbation of the spherical shape, which minimizes the surface tension energy. Therefore, the study of the motion around this steady state provides a real insight to understand this complex transition and it has already been used quite extensively, for example in the TAB model (O'Rourke and Amsden 1987).

The first significant study of this oscillatory behavior is the linear analysis for small perturbation around the spherical state provided by John W. Strutt (3rd Baron Rayleigh) (1883). Under the assumption of a potential flow, *i.e.* inviscid, incompressible and irrotational flow, and surface tension, the Navier-Stokes equations are integrated into an unsteady Bernoulli law, which can be then combined with the kinematic closure between the velocity of the flow and the one of the droplet surface. Moreover, the Laplace equation on the flow potential enforces the solution to be decomposed into a spherical harmonics basis. A second-order equation is finally obtained for each spherical mode which all behave as harmonic oscillators with their own frequency, the greater the mode is, the greater the frequency.

This problem has been successively studied by S. H. Lamb from the inviscid case in 1895 (S. H. Lamb 1895) to 1916 (H. Lamb 1916), where he started to study the viscous case. It led to the addition of a damping term to the oscillation motion where the damping rate increases with the mode under consideration. These results were also retrieved in the works of Chandrasekhar (1961) and completed by Reid (1960) and Prosperetti (1980).

Following detailed experiments, where large discrepancies were found (Trinh and Wang 1982) between the frequencies obtained from linear analysis and the ones observed during large oscillations, Tsamopoulos and Brown first investigated the non-linear expansion of the equations in 1983 thanks to a Poincaré-Lindstedt approach. This work enabled to establish the first correction for the coupling between amplitude and frequency for each mode. The frequency decreases proportionally to the square of the amplitude and this phenomenon is intensified for larger modes.

These works on non-linear behavior of the droplet oscillation always lead to tedious combinatory problems, when more than two spherical modes interact and they have been subject of numerous approaches to simplify them in the past decades. Recently, another approach was proposed by Plümacher et al. (2020) where the problem is fully reformulated in terms of surface variables. Then, a set of recursive equations describing the dynamics are obtained through a multi-scale expansion of these surface variables and a fully explicit solution is proposed for the first correction of the spherical harmonic shapes and frequencies.

One can then realize that, whereas all variables are harmonic in the limit of small perturbation and the problem remains linear, one usually rapidly encounters nonlinear effects as soon as the droplet deformation is sufficient. However, we need to define energies for an oscillating droplet and the key question is here the choice of the proper variables to do so, since the range of validity of the harmonic oscillation is not the same for all variables. Relying on a DNS configuration and on the `Mercur(v)e` library, we will show that there is a natural choice for the energies and variables to be used in the `SAP`. An accurate evaluation of the averaged geometrical properties will be essential here.

#### 4.6.2 DNS configuration

The computational domain is a box of dimension  $40 \times 40 \times 40 \mu\text{m}$  that is discretized over a regular Cartesian grid with 128 points per dimension leading to a total of 2 097 152 computational nodes. Symmetric boundary conditions are imposed on each face of the domain. The kinematic viscosity of the fluid is set to  $\nu = 1.7\text{e}-8 \text{ m s}^{-2}$  in order to configure a situation where the viscosity effects on the results can be minimized. We consider as initial condition a deformed droplet with an axisymmetric ellipsoidal shape that will evolve due to surface tension effects: the initial shape is defined thanks to the 0-level of a level-set such that the semi-major axis  $p$  over semi-minor axis  $q$  ratio is  $p/q = (1.05, 1.15)$  and such that the volume of the droplet is  $4/3 \pi R^3$  with  $R = 10 \mu\text{m}$ . The computational load for a single run is 1280 h CPU which represents 20 h of computation over 64 CPU. We have chosen two deformation amplitudes, one will still be in the small perturbation limit leading to harmonic oscillations of most of the considered quantities, while the second will allow to go beyond this regime and highlight different behaviors for the different variables. Fig. 4.13 shows the initial level-set volumetric field and the triangulated interface obtained with `Mercur(v)e`. One period  $T$  of evolution of the droplet deformation is shown in Fig. 4.14.

At  $t/T = 0$ , Fig. 4.14a, the droplet is initially at a prolate state. Due to capillarity, the droplet potential energy starts to be converted into kinetic energy such that the droplet reaches a spherical shape in Fig. 4.14b, and then continues to deform such that at  $t/T = 1.0$ , Fig. 4.14c,

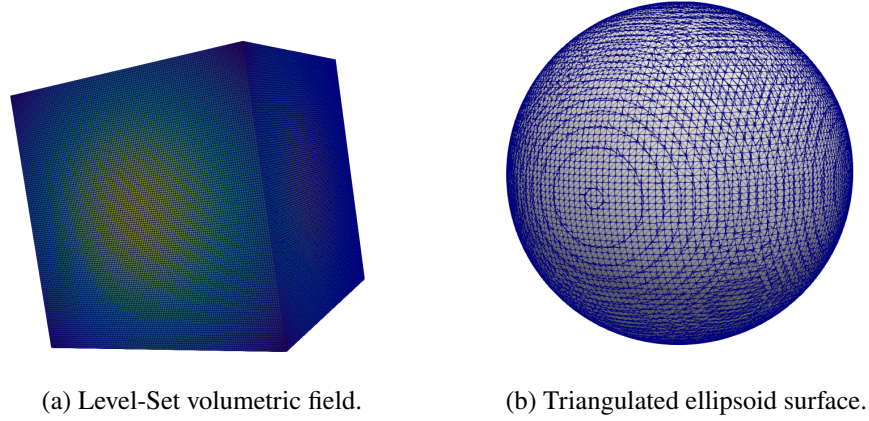


Figure 4.13: The initial configuration of the ellipsoidal droplet.

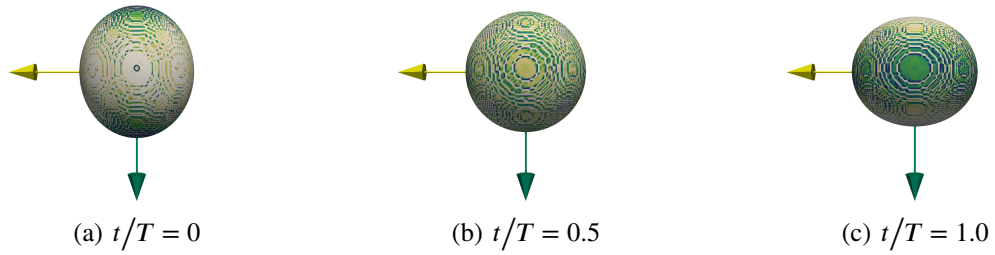


Figure 4.14: One period  $T$  of evolution of the deformed droplet. The color map maps values of the Gauss curvature from low to high.

the droplet is in an oblate configuration.

### 4.6.3 Analysis

We now investigate the evolution of geometric quantities over the whole simulation time and integrated over the surface of the droplet for two different initial configurations of the deformed droplet,  $p/q = 1.05$  and  $p/q = 1.15$ .

Figures 4.16b, 4.17b, 4.18b and 4.19b display the relative mean Gauss curvature ratio, with  $S_0 G_0$  the integral of the Gauss curvature over the spherical droplet multiplied by its surface,  $S_0 G_0 = 4\pi$ . For the computation with  $\text{Mercur}(\nu)_e$ , the ratio is constant and close to zero up to a tolerance of  $1e-12$ . This result was expected since from the Gauss-Bonnet theorem,  $\langle G \rangle$  is a topological invariant, but it attests the reliability of the post-processing library  $\text{Mercur}(\nu)_e$ .

Let us underline from the various evaluations, that it is essential to use the  $\text{Mercur}(\nu)_e$  library for two reasons : 1- we can guarantee the proper evaluation of the topological invariant through a discrete verification of the Gauss-Bonnet theorem, 2- the level of error obtained from

the library compared to the evaluation conducted in ARCHER directly from the level-set function is much lower and it is essential in this configuration of small perturbations in order to draw some firm conclusions from the investigation.

Depending on the cases, since the references quantities used for the relative quantity ratio are exact characteristics of the sphere, mainly functions of its radius, we can have a slight shift in the quantities due to the level-set formulation and to its discretization on the proposed grid: the computed volume  $V_0$  and surface  $S_0$  of the object will slightly change based on how close the 0-surface of the level-set is to the cell nodes. Eventually, Fig. 4.15 emphasizes the volume conservation over time of the deformed droplet by plotting the relative volume ratio where  $V_0$  is the volume of the droplet when reaching it spherical shape,  $V_0 = 4/3\pi R^3$ . The slight difference in the value of the volume ratio for the different cases is also associated with the initialization errors produced by the variability of the alignment of the 0-surface within the cell nodes.

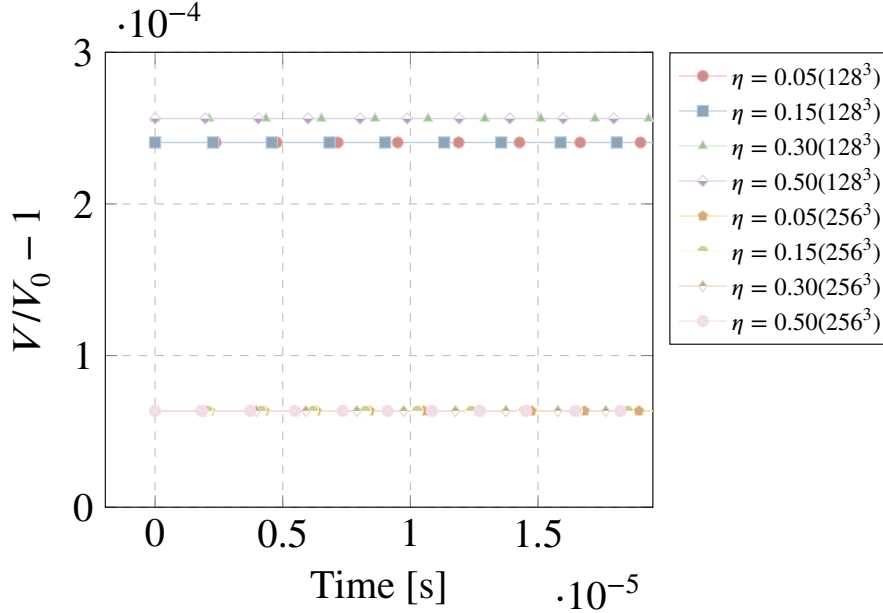
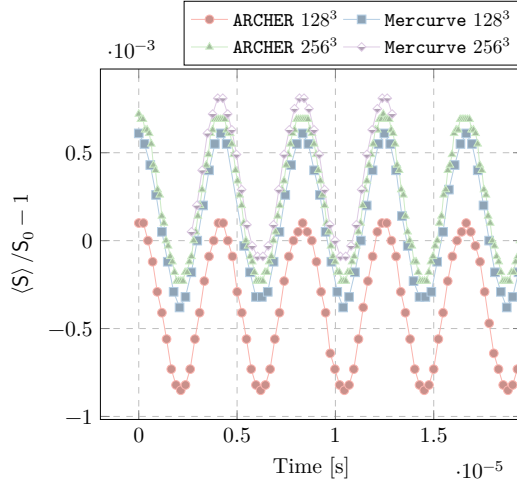


Figure 4.15: Relative volume ratio conservation.

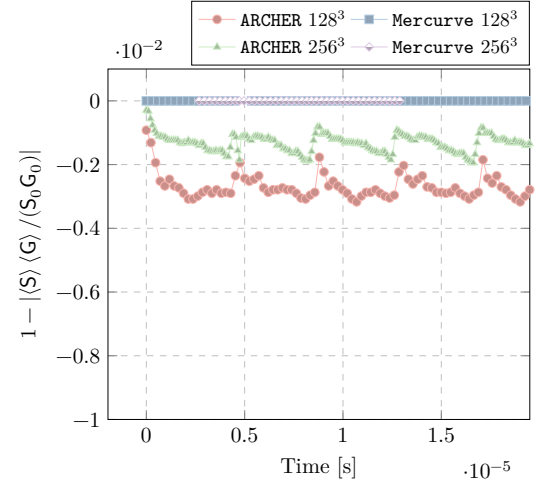
Figures 4.16a, 4.17a, 4.18a and 4.19a show the relative surface ratio evolution over the simulation time, with  $S_0 = 4\pi R^2$  the theoretical surface of the droplet when reaching a spherical shape. As we can see, once the droplet is left unconstrained in its perturbed initial configuration with zero kinetic energy, its surface evolves as an harmonic oscillator, as well as the mean curvature, even if we can detect a small departure from harmonic oscillations. This first case is coherent with the small perturbation analytical estimation of the oscillating dynamics and the observed frequency is the one identified by the theory presented in the introduction.

Then, to depart slightly from the small perturbation regime, we have provided a series of

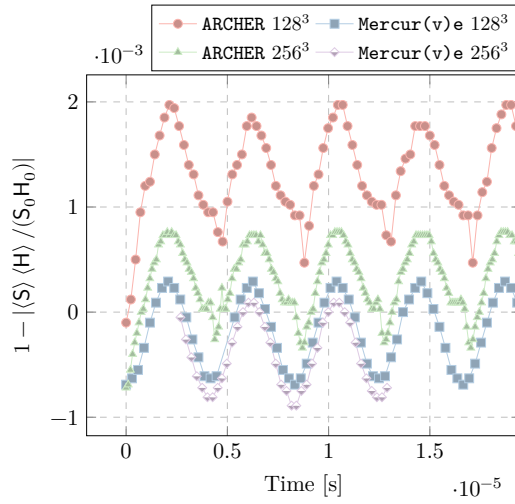
cases, with larger amplitude, where this departure is more pronounced.



(a) Relative surface ratio.



(b) Relative surface averaged Gauss curvature ratio.



(c) Relative surface averaged Mean curvature ratio.

Figure 4.16: Evolution of the geometrical properties ( $p/q - 1 = 0.05$ )

Actually, as plotted in Figures 4.16c, 4.17c, 4.18c and 4.19c the relative average mean curvature does not show the same behavior for a larger deformation. The surface integral of the mean curvature of the droplet at its spherical state reads  $H_0 = -1/R$ . Two non-identical periodic peaks are clearly visible, suggesting the surface-averaged mean curvature is not harmonic any more. In fact relying on recent references and a specific differential geometry investigation, with Loison (2023), we were able to provide some higher order perturbation analysis, which

explains this different behavior between mean curvature and surface.

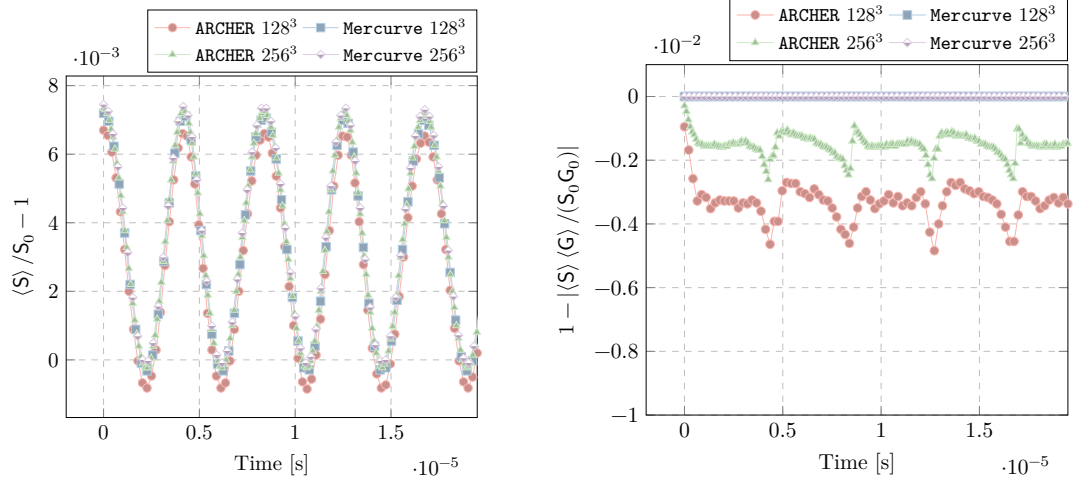
In the various plots, this phenomenon is still present at various elongation of the original ellipsoid, even if the harmonic behavior of the surface oscillation has a tendency to deteriorate when  $p/q - 1 = 0.5$ , not in amplitude but in frequency, with non-linear effects, as predicted by the theory. At this level however, we are very far from a perturbation of the original sphere!

Whereas in the literature several subscale energies and related variables describing the droplet oscillations have been proposed ranging from the elongation in O'Rourke and Amsden (1987) to the mean curvature  $\langle H \rangle$  in Herrmann (2013), our detailed study confirms that it is much more robust to consider the surface energy  $\sigma(\Sigma - \Sigma_0)$ , with related variable the square root of surface departure from sphericity, in order to capture both the case of small perturbations *w.r.t.* the equilibrium (where all choices are relevant and equivalent) as well as large deviations from the sphericity as observed in Figures 4.17a, 4.18a and 4.19a.

This is also coherent with what has been observed numerically and experimentally for bubbles (Lalanne et al. 2013), where the harmonic oscillation behavior is very robust, even within the framework of large deformations. This allows to identify a coherent energy function for subscale oscillating droplets to be used in a variational formulation to derive a reduced model (see Cordesse, Di Battista, Chevalier, et al. (2020) and section 3.5.3), and proves the impact of relying on a proper and accurate evaluation of the various surface averaged geometric quantities.

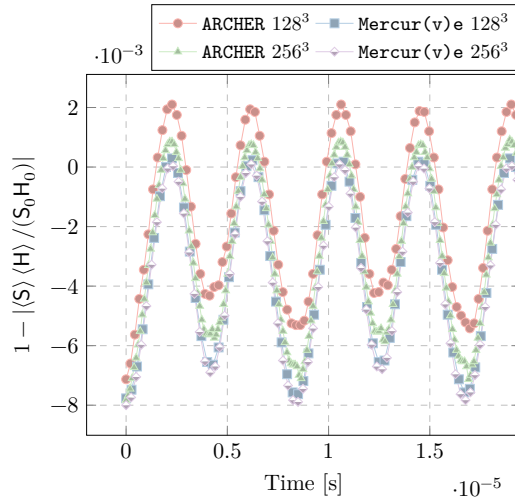


4 A computational framework based on the discrete estimation of geometrical properties over triangulated interfaces to perform validation of two-phase flow models



(a) Relative surface ratio.

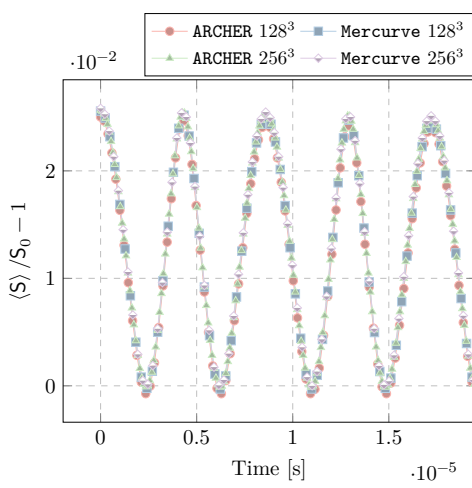
(b) Relative surface averaged Gauss curvature ratio.



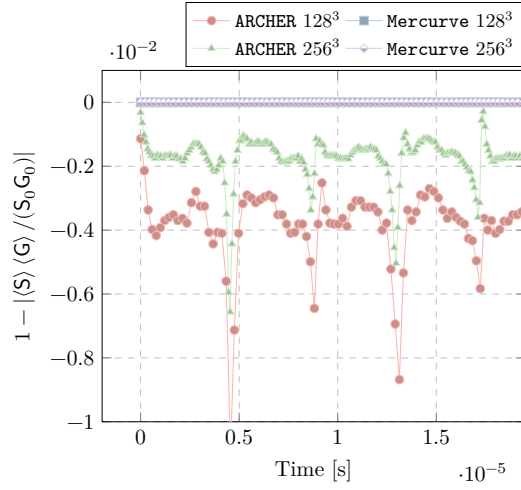
(c) Relative surface averaged Mean curvature ratio.

Figure 4.17: Evolution of the geometrical properties ( $p/q - 1 = 0.15$ )

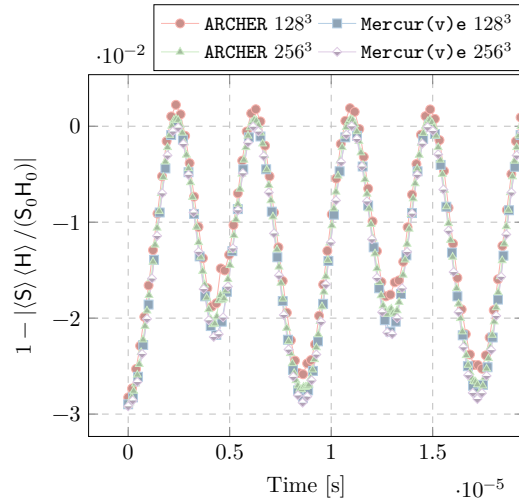




(a) Relative surface ratio.



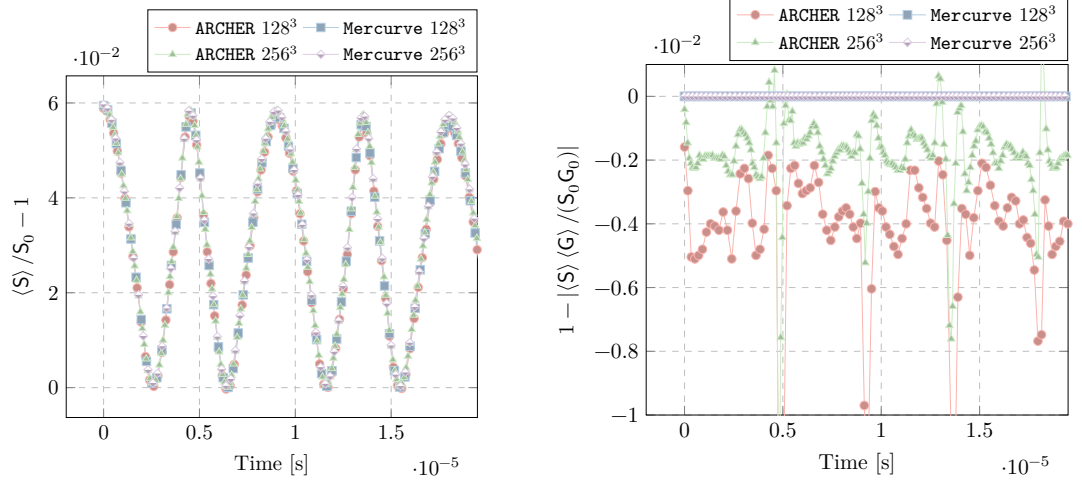
(b) Relative surface averaged Gauss curvature ratio.



(c) Relative surface averaged Mean curvature ratio.

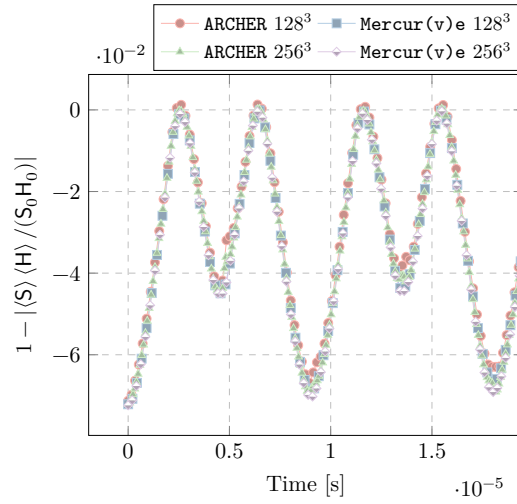
Figure 4.18: Evolution of the geometrical properties ( $p/q - 1 = 0.3$ )

4 A computational framework based on the discrete estimation of geometrical properties over triangulated interfaces to perform validation of two-phase flow models



(a) Relative surface ratio.

(b) Relative surface averaged Gauss curvature ratio.



(c) Relative surface averaged Mean curvature ratio.

Figure 4.19: Evolution of the geometrical properties ( $p/q - 1 = 0.5$ )

## 4.7 Conclusions

In this chapter we presented a computational framework based on the estimation of curvature properties over a triangulated two-phase flow interface. The algorithm used to compute the curvature fields is inspired by the work of Meyer et al. (2003) and it features important properties that are useful in the frame of perspectives this work is conducted: the algorithm allows the Gauss-Bonnet topological invariant to be preserved at machine precision and, in turn, the Gauss-Bonnet theorem preservation allows to count objects homeomorphic to spheres, introducing the possibility of extracting NDFs from DNS simulations. The post-processing routine has then be used to derive curvature maps that allow the identification of topological configurations over the duration of the DNS simulation. A practical case of ellipsoidal oscillating droplets has been presented, in which the use of the discussed post-processing strategy allowed the determination of the correct energies underlying the studied phenomena, energies that are in turn needed when constructing reduced models with the SAP, as explained in chapter 3.

Despite the very interesting properties, the described post-processing procedure highlights minor limitations, some of which have been fixed and some of which are in progress. In particular, the algorithm performs the curvatures calculations independently, failing at ensuring the local Willmore energy constraint (eq. (4.19)), especially in situations which feature topological objects with shapes close to a sphere; even if a projection has been proposed to cure this drawback, improving the mesh should also help in these matters. In addition, a detailed error analysis of the procedure is assessed highlighting the crucial role of the triangulated mesh in the definition of error rate.

All this effort translated in the development of a FOSS named `Mercur(v)e`, that is accessible at <https://gitlab.com/rubendibattista/mercurve> with a permissive license. The structure of the library in terms of code development and design choices is discussed in chapter 6.

## 4.8 Perspectives

The surface discretization strategy presented in this chapter, with the associated curvature estimation algorithm based on 1-ring averaging, is applicable only on closed objects, since the formulas exposed in section 4.3.4 are valid only on closed objects. This is not a great annoyance when performing post-processing of reasonably-sized DNS cases, but it can become one if the cases are very big, or, for example, if the curvature estimation has to be performed *run-*

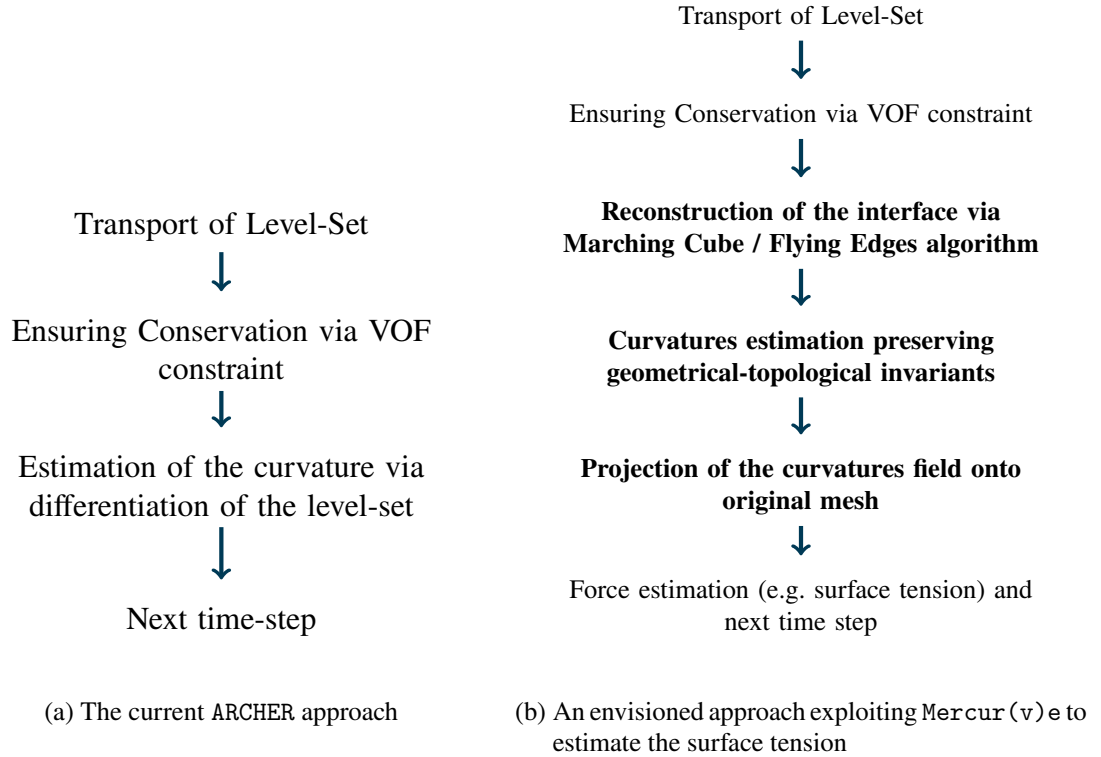


Figure 4.20: Injecting Mercure predictions at runtime during a DNS

time, at each time-step of the running DNS simulation, in which the entire level-set field is split in computational blocks, that are then sent to different computational units. Additionally, the analysis of cases with particular symmetry in the Boundary Conditions could also require the ability of computing the curvatures when the interface crosses computational boundaries. A possible idea is to retrieve the intersection of the interface with the boundaries and recompute the associated geometrical 1-ring averaged quantities taking into account the fact that the 1-ring cell is intersected by the boundaries, therefore the associated area is different from what it is generally computed on normal cells that are fully contained within the domain.

One other possible perspective of this work would be to investigate the impact of computing the mean curvature, for the purpose of correctly estimating the surface tension force, with this strategy, during a DNS simulation instead of just as post-processing procedure (see Fig. 4.20). Despite expecting the process to be way more expensive, in terms of computational cost, *w.r.t.* classical level-set differencing methods, it would still be very interesting, for reasonably sized DNS cases, to understand the differences in topological developments of the interface when the surface tension is more accurately depicted as for the results we just discussed. This topic is work in progress in collaboration with CORIA laboratory and T. Ménard in the ARCHER code.

This would be easy once the curvature estimation formulas are fixed to work for non-closed interfaces crossing process boundaries.

Another interesting perspective would be to study the impact of a different choice of geometrical properties to describe the evolution of the interface surface. In particular [Bermejo-Moreno and Pullin \(2008\)](#); [Bermejo-Moreno, Pullin, and Horiuti \(2009\)](#) show other potential possibilities that are different from the mean and Gauss curvatures we have employed in this work. These alternative geometrical representations are non-linear functions of the curvatures and might represent a more suitable choice to obtain insights from simulations. We envision a collaboration with I. Bermejo-Moreno, with whom we interacted at Stanford University during the CTR Summer Program in 2018.



# Appendices

## 4.A Algorithm to handle multiblock simulations

---

Mercur(v)e is a library (and an executable) that takes as input a DNS simulation done with CORIA's ARCHER software and produces the results presented in this work. Generally a high-fidelity DNS is computed in parallel, dividing the computational domain in blocks that are then assigned to one of the processors of the cluster. So, generally, the output of a simulation is a series of subdomains of the original domains that could not contain closed surfaces. The formulas given in section 4.3.4 could be adapted for non-closed surfaces, but in the way they are given they are strictly valid for closed surfaces. Hence in the current implementation we use a simple algorithm to merge the multiple blocks into a single block for which the MC algorithm returns a closed surface. The algorithm basically works as follows:

- Sort the blocks in ascending order *w.r.t.* the coordinates of their origin
- Allocate space for the first block
- Loop over the blocks taking care of having pointers to the current last block respectively on the  $z$ ,  $y$ ,  $x$ -axes.
- Check which coordinate changes *w.r.t.* the last block allocated starting from  $z$ , then  $y$  and lastly  $x$
- Allocate space for each block on the correct axis direction
- Loop again over the blocks actually inserting the block in the right position.

The details of the implementation are reported in algorithm 4.A.1.

---

**Algorithm 4.A.1** Algorithm to merge a multi-block DNS simulation

---

```

1: procedure MERGEBLOCKS(blocks)
2:   blocks  $\leftarrow$  SORT(blocks)
3:   b0  $\leftarrow$  blocks[0]
4:   nx, ny, nz  $\leftarrow$  GETDIMENSIONS(b0) ▷ Get first block dimensions
5:   b0  $\rightarrow$  SETSTARTIJK(0, 0, 0) ▷ The starting i, j, k indices in the merged block
6:   b0  $\rightarrow$  SETENDIJK(nx, ny, nz) ▷ The ending i, j, k indices in the merged block
7:   bx = by = bz = b0
8:   for b in blocks[1 : ] do ▷ Loop from the second block on
9:     if b  $\rightarrow$  x  $\neq$  b0  $\rightarrow$  x then ▷ Stack block over x-axis
10:      si, sj, sk  $\leftarrow$  GETSTARTIJK(bx)
11:      ei, ej, ej  $\leftarrow$  GETENDIJK(bx)
12:      Nx, Ny, Nz  $\leftarrow$  GETDIMENSIONS(b)
13:      nx += Nx
14:      b  $\rightarrow$  SETSTARTIJK(ei, sj, sk)
15:      b  $\rightarrow$  SETENDIJK(ei + Nx, sj + Ny, sk + Nz)
16:      bx = by = bz = b
17:      wrapX = true
18:      continue
19:     else if b  $\rightarrow$  y  $\neq$  b0  $\rightarrow$  y then ▷ Stack block over y-axis
20:      si, sj, sk  $\leftarrow$  GETSTARTIJK(by)
21:      ei, ej, ej  $\leftarrow$  GETENDIJK(by)
22:      Nx, Ny, Nz  $\leftarrow$  GETDIMENSIONS(b)
23:      if !wrapX then
24:        ny += Ny
25:      end if
26:      b  $\rightarrow$  SETSTARTIJK(si, ej, sk)
27:      b  $\rightarrow$  SETENDIJK(si + Nx, ej + Ny, sk + Nz)
28:      by = bz = b
29:      wrapY = true
30:      continue
31:     else ▷ Stack block over z-axis
32:      si, sj, sk  $\leftarrow$  GETSTARTIJK(bz)
33:      ei, ej, ej  $\leftarrow$  GETENDIJK(bz)
34:      Nx, Ny, Nz  $\leftarrow$  GETDIMENSIONS(b)
35:      if !wrapY then
36:        nz += Nz
37:      end if
38:      b  $\rightarrow$  SETSTARTIJK(si, sj, ek)
39:      b  $\rightarrow$  SETENDIJK(si + Nx, sj + Ny, ek + Nz)
40:      bz = b
41:    end if
42:  end for
43:  mergedBlock  $\rightarrow$  SETDIMENSIONS(nx, ny, nz) ▷ Allocate space
44:  mergedBlock  $\rightarrow$  INSERTBLOCK(b0) ▷ Insert first block
45:  for b in blocks[1 : ] do
46:    mergedBlock  $\rightarrow$  INSERTBLOCK(b) ▷ Insert all the other blocks
47:  end for
48: end procedure

```

---



# 5

## Numerical schemes and simulation of two-phase flows

In this chapter we rework the most important aspects related to the discretization of system of PDEs using the FVM. We present the fundamentals of the method recasted in a form that closely resembles the actual implementation in the `josiepy` code (the architecture of which is described in chapter 6). We present a template vector system of equations that allows the individual and independent discretization of convective, non-conservative, diffusive and source terms. We present a basic structured mesh generation functionality based on the Trans-Finite interpolation (TFI). We list in details all the alternative schemes implementations that have been investigated in the framework of this thesis supported by a wide choice of verification tests. A simulation case featuring the system of equations that was discussed in section 3.6 which propose a governing equation for the interfacial area density is also discussed as a result of the numerical framework discussed in this chapter.

### 5.1 Introduction

In chapter 3 we introduced a methodology to derive a system of equations from the initial assumption of kinetic and potential energy underlying the phenomenon we want to model. Except for specific cases, those systems of PDEs are impossible to be solved in closed analytical form, hence the necessity of employing sound numerical methods that can be executed on a computer machine. Different approaches are available for the discretization of system of PDEs, classically the Finite Difference Method (FDM), the FVM and the Finite Element Method (FEM) are featured in the literature. Those methods are generally based on the space semi-discretization of the physical domain on which the system resolution must be enforced in a set of nodes, vertices or control volumes (a *mesh*) on which the fields are defined in a discretized way and the directional derivatives are imposed following a specified *stencil*, interpolation scheme or **numerical flux** scheme. This semi-discretized form of the system is then integrated in time using a suitable Ordinary Differential Equation (ODE) solver. In addition to these classical strategies, in more recent times, with the advent of Artificial Intelligence (AI) and powerful

computational workstations, other methods that leverage [AI](#) to solve the [PDEs](#) system without the need of a *mesh* are also studied and used ([Scoggins et al. 2021](#)).

In this work, we employ the Finite Volume Method (FVM). In the following sections we introduce an opinionated framework, general enough, that presents the specificities of the [FVM](#) for our purposes with a particular interest towards hyperbolic systems and a stress on a vectorized implementation of the numerical operations that are needed to implement such approximation. This allowed an actual deployment of a solution packaged in an easy-to-use Python library called `josiepy` and presented in details in section 6.1.

## 5.2 Basic concepts for the discretization of (a system of) Partial Differential Equations (PDEs) with the Finite Volume Method (FVM)

In this section we provide a synthetic depiction of the [FVM](#). The interested reader can find further information in [LeVeque \(1990\)](#); [Godlewski and Raviart \(1991\)](#); [Godlewski and Raviart \(1996\)](#); [Versteeg and Malalasekera \(2007\)](#); [LeVeque \(2002\)](#); [Bouchut \(2004\)](#); [Eleuterio F. Toro \(2009\)](#). Please note that the notation chosen to present the [FVM](#) is more *implementation-oriented*, trading a slight loss of unambiguousness with a lighter abstraction mental effort and a closer resemblance to what it is actually coded in `josiepy` (section 6.1).

In order to introduce the computational framework of the [FVM](#) we define a generic state variable  $\mathbf{q}$  that wraps  $N_{\text{fields}}$  individual scalar fields

$$\mathbf{q} = \left\{ q_1, q_2, \dots, q_{N_{\text{eqs}}}; q_{N_{\text{eqs}}+1} \dots q_{N_{\text{fields}}} \right\} \quad (5.1)$$

where  $N_{\text{eqs}}$  is the number of fields that are present in the equations,  $\left\{ q_{N_{\text{eqs}}+1} \dots q_{N_{\text{fields}}} \right\}$  is an extended set of variables that can include auxiliary variables that can be derived from the original primitive variables  $\left\{ q_1 \dots q_{N_{\text{eqs}}} \right\}$  (e.g. in the Euler system written in conservative form, very often we need to access variables such as the pressure ( $p$ ) or the speed of sound ( $c$ ) that are normally not part of the set of states in conservative form. Hence in `josiepy`, for convenience, they are stored in an extended  $\mathbf{q}$ , trading memory pressure for computational cost. The  $\mathbf{q}$  extended vector satisfies a generic [PDE](#) vector system (eq. (5.2)) over a generic domain  $\Omega$ .

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) + \mathbf{B}(\mathbf{q}) \cdot \nabla \mathbf{q} - \nabla \cdot (\mathbf{K}(\mathbf{q}) \cdot \nabla \mathbf{q}) + s(\mathbf{q}) = \mathbf{0} \quad (5.2)$$

We define  $\nabla \cdot \mathbf{F}(\mathbf{q})$  as the *convective term*,  $\mathbf{B}(\mathbf{q}) \cdot \nabla \mathbf{q}$  as the *non-conservative term*,  $\nabla \cdot (\mathbf{K}(\mathbf{q}) \cdot \nabla \mathbf{q})$  as the *diffusive term* and  $\mathbf{s}(\mathbf{q})$  as the *source term* of the system. Moreover,  $\mathbf{F}(\mathbf{q})$  is the *convective flux* of the system,  $\mathbf{B}(\mathbf{q})$  is the *non-conservative pre-multiplier* and  $\mathbf{K}(\mathbf{q})$  is the so-called *viscosity tensor*. In the vectorial representation we abuse a bit the notation writing a state made of just the  $a, a = 1 \dots N_{\text{eqs}}$  fields that are governed by the set of equations in the same way as the full state that includes also the auxiliary fields, *i.e.*  $\mathbf{q}$ . To avoid ambiguity, the same system can be equivalently written assuming Einstein summation convention as:

$$\frac{\partial q_a}{\partial t} + \frac{\partial F_{ad}(q_b)}{\partial d} + B_{abd}(q_b) \frac{\partial q_b}{\partial x_d} - \frac{\partial}{\partial x_{d'}} \left( K_{abd'd}(q_b) \frac{\partial q_b}{\partial x_d} \right) + s_a(q_b) = 0 \quad (5.3)$$

$$a = 1 \dots N_{\text{eqs}}; b = 1 \dots N_{\text{fields}}; d, d' = 1 \dots N_{\text{dim}}$$

being  $N_{\text{dim}}$  the dimensionality of the problem (this is more an implementation defined constant, *i.e.* if the solver allows to simulate 3D domains, then  $N_{\text{dim}} = 3$ ). The indicial notation also helps to highlight how the flux operator  $\mathbf{F}(\mathbf{q})$  is a 2<sup>nd</sup> order tensor, the  $\mathbf{B}(\mathbf{q})$  is a 3<sup>rd</sup> order tensor, and the  $\mathbf{K}(\mathbf{q})$  is a 4<sup>th</sup> order tensor.

The **FVM** is based on the assumption of dividing the total domain  $\Omega$  in a large number of polyhedral subdomains  $\Omega_i$ , *i.e.* the cells of the mesh, each one of volume  $|\Omega_i|$ , surface  $|\partial\Omega_i|$ , on which eq. (5.2) is enforced in integral form. If we define the mean value of the field  $\mathbf{q}$  over the cell volume as:

$$|\Omega_i| \langle \mathbf{q} \rangle_i \triangleq \int_{\Omega_i} \mathbf{q} \quad (5.4)$$

the integral form of eq. (5.2) over a cell  $\Omega_i$ , considered of constant volume, can be written (omitting from now on the  $\langle \bullet \rangle$  operator for the sake of readability):

$$|\Omega_i| \frac{\partial \mathbf{q}_i}{\partial t} + \int_{\Omega_i} \nabla \cdot \mathbf{F}(\mathbf{q}) + \int_{\Omega_i} \mathbf{B}(\mathbf{q}) \cdot \nabla \mathbf{q} - \int_{\Omega_i} \nabla \cdot (\mathbf{K}(\mathbf{q}) \cdot \nabla \mathbf{q}) + \int_{\Omega_i} \mathbf{s}(\mathbf{q}) = \mathbf{0} \quad (5.5)$$

So the **FVM** consists in providing sound approximations of the various volume integrals in eq. (5.5), that depend only on  $\mathbf{q}_i = \langle \mathbf{q} \rangle_i$  very often expressed as a sound reconstruction over the cell centers. The simplest reconstruction being a constant value over the cell, typical for 1<sup>st</sup> order schemes as we will explain in section 5.2.2.4.

### 5.2.1 Structured mesh generation

The **FVM**, as presented, needs to be computed over a set of polyhedral cells. That means, a physical domain on which the problem must be resolved, needs to be decomposed in the set

of subdomains  $\Omega_i$  on which the PDEs are then discretized. The process of mesh generation is a complex topic, still relying on a non-negligible amount of expertise driven by hands-on activity, and we absolutely do not aim at providing a fulfilling and complete analysis on the subject; the interested reader can certainly refer to Thompson et al. (1998) for a monograph. We will limit ourselves to present the general guidelines and available macro-strategies for mesh generation, with a particular focus on Trans-Finite interpolation (TFI), that is the method the library josiepy provides with its integrated mesh generator at this date.

### 5.2.1.1 Structured boundary-fitted meshes

The easiest situation one can find when solving a PDEs system is arguably a rectangular domain. In this simplified case, the rectangle is identified by the coordinates of three points, for example  $(\mathbf{x}_{sw}, \mathbf{x}_{se}, \mathbf{x}_{ne})$  that are respectively the bottom-left (South-West (SW)), bottom-right (South-East (SE)), top-right (North-East (NE)) corners. In order to build the set of rectangular cells we can then use in the FVM, it is sufficient, in the easiest case, to create an arbitrary number of linearly spaced points in one coordinate, for example in the  $\hat{e}_x$ -direction, and per each point  $x_i$ , linearly interpolate in the orthogonal direction  $\hat{e}_y$ . In 3D, for each pair of points  $(x_i, y_i)$ , an arbitrary set of points  $z_i$  in the  $\hat{e}_z$  direction is linearly created. A rectangular Cartesian mesh is thus created. A Cartesian mesh, moreover, is a *structured mesh*.

#### Definition 2

A *structured mesh* is a grid whose entire set of points follows the same topological structure. In other words, in a structured mesh, starting from a generic point  $\mathbf{x}_i$ , it is always possible to find the set of neighbors of that point following the same stencil. *i.e.* it exists a transformation (2D for simplicity) from the computational to the physical space:

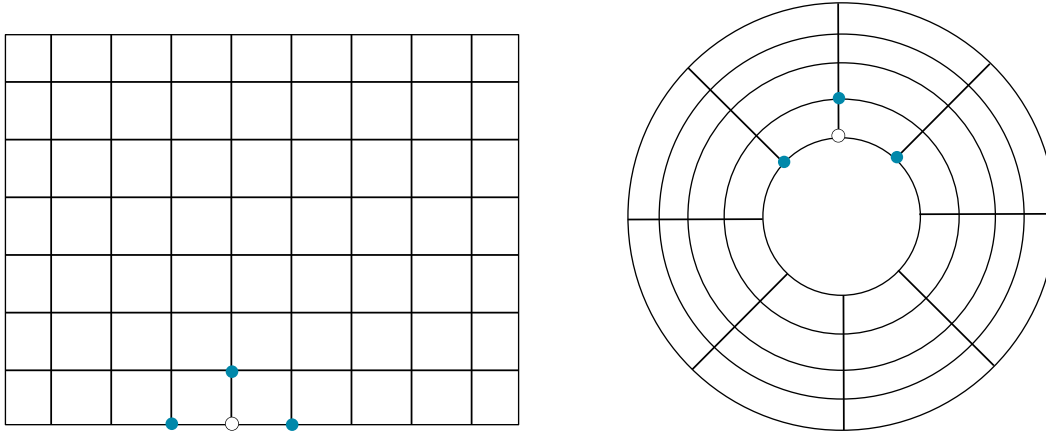
$$\mathbf{X}(\zeta_1, \zeta_2) = (x(\zeta_1, \zeta_2), y(\zeta_1, \zeta_2))^T \quad (5.6)$$

that has a discrete subset made of  $N_x \times N_y$  points,  $\{\mathbf{X}\} = (\zeta_I, \eta_J); I = 0 \dots N_x, J = 0 \dots N_y$ , indexed by  $I, J$ . The relation

$$(I, J) \leftrightarrow (\zeta_I, \eta_J) \quad (5.7)$$

implies the same direct relationship between all neighbors.

It is easy to understand that with a rectangular mesh we only get so far: a lot of problems in engineering and industrial contexts feature domains that are more complex than a bare rectangle. The easiest method to cope with the difficulty of deformed domains is to use orthogonal



(a) Curvilinear coordinates are still rectangular in the logical/programming environment

(b) Cylindrical coordinates in physical space

Figure 5.1: Curvilinear coordinates comparison in logical/programming and physical space

analytical coordinates, like for example, cylindrical or spherical coordinates,

$$\boldsymbol{\zeta} = \boldsymbol{\zeta}(\mathbf{x}) = (\zeta(x, y), \eta(x, y)) \quad (5.8)$$

These coordinates are still orthogonal. The PDEs problem then is transformed because the differential operators change on a curvilinear system, but on the implementation side, the domain is still rectangular in the new set of coordinates  $\zeta_1, \zeta_2$ . Fig. 5.1 shows a visualization of the computational domain in the logical/programming space, that is still rectangular, while in the physical space we actually have a cylindrical grid. Since the cylindrical grid “fits” exactly the boundary of the domain (*w.r.t.* a Cartesian mesh that would just approximate the boundary), these mesh can be defined boundary-fitted (Thompson et al. 1998).

### 5.2.1.2 Trans-Finite interpolation

Alternatively to this mapping with analytical orthogonal coordinates, an algebraic discrete strategy for the mesh generation of a deformed domain  $\Omega$  can be employed called Trans-Finite interpolation (TFI). Without loss of generality, we restrict the discussion to 2D mesh generation for simplicity. Given a domain  $\Omega$  delimited by the curvilinear edges  $\partial\Omega_l$ ,  $l = 0 \dots N_{\text{boundaries}}$ , we provide the discrete parametrization of the curves associated to the curvilinear edges on  $N_x \times N_y$  points:

$$\mathbf{X}_I(\zeta_I, \eta_J) = (x(\zeta_I, \eta_J), y(\zeta_I, \eta_J))^T, \quad I = 0 \dots N_x, J = 0 \dots N_y \quad (5.9)$$

and potentially also values of its  $n$  derivatives. The idea of the **TFI** is to compute an interpolation in each coordinate direction between all the provided boundary values for the parametrization (eq. (5.9)) and possibly its derivatives, *i.e.*

$$\begin{aligned} U &= \sum_l^3 \sum_I^{N_x} \sum_n \alpha_{In}(\zeta) \left[ \mathbf{X}_l(\zeta_I, \eta) + \frac{\partial^n \mathbf{X}_l}{\partial \zeta^n} \right] \\ V &= \sum_l^3 \sum_J^{N_y} \sum_n \beta_{Jn}(\eta) \left[ \mathbf{X}_l(\zeta, \eta_J) + \frac{\partial^n \mathbf{X}_l}{\partial \eta^n} \right] \end{aligned} \quad (5.10)$$

and then obtain the final interpolation as the *boolean sum of the individual coordinate interpolations*, that is the sum of the individual interpolations in  $x$  and  $y$  directions  $\mathbf{U}, \mathbf{V}$  minus their cross product,

$$\mathbf{X}(\zeta, \eta) = \mathbf{U} + \mathbf{V} - \mathbf{U}\mathbf{V} \quad (5.11)$$

$\alpha_{In}, \beta_{Jn}$  are blending functions that need to fulfill specific conditions (Thompson et al. 1998) at the boundaries  $\partial\Omega_l$ .

The easiest choice that can be made is to use linear blending functions,

$$\begin{aligned} \alpha_{00}(\zeta) &= 1 - \zeta \\ \alpha_{01}(\zeta) &= \zeta \\ \beta_{00}(\eta) &= 1 - \eta \\ \beta_{01}(\eta) &= \eta \end{aligned} \quad (5.12)$$

As an example, Fig. 5.2 shows a mesh generated by `josiepy` with a linear **TFI**.

### 5.2.1.3 More advanced meshing techniques

More advanced approaches are possible to generate the discretization of a domain. Always in the context of structured meshes, we can cite **PDEs** based methods that provide the mapping at eq. (5.6) as the solution of **PDEs**. Those **PDEs** can be elliptical or hyperbolic. In the first case a quasi-linear elliptical system of **PDEs** (“Poisson grid generation equations”) is solved to provide the *harmonic mapping* between the physical, deformed, domain and a nonuniform unit square logical domain. An additional mapping also maps the nonuniform unit square to a uniform one that is easier to handle programming-wise. In the case of hyperbolic methods, a front propagates normally to a starting surface. The equations are generally derived (Thompson et al. 1998) from grid angles and cell size considerations. In order to treat even more

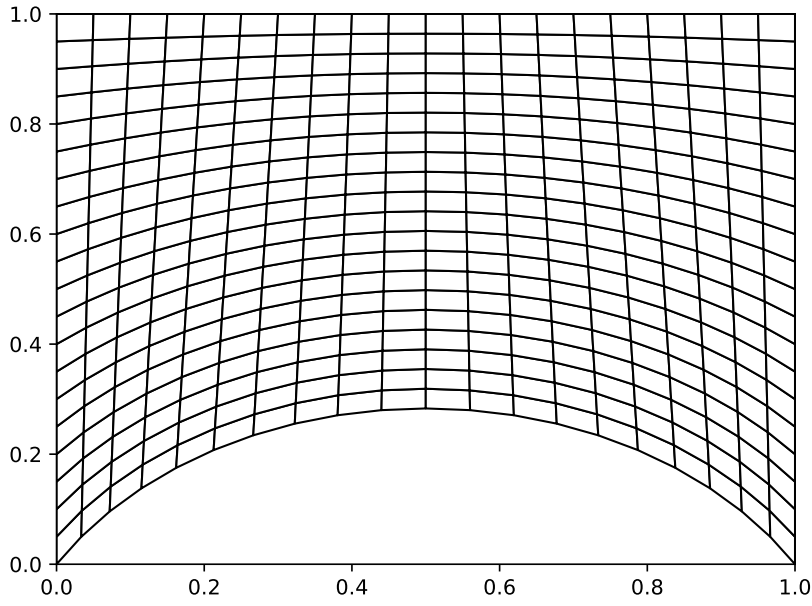


Figure 5.2: A mesh generated by josiepy integrated mesh generator

complex domains and configurations, multiblock methods have been introduced: the domain is subdivided in subdomains, each one being an independent structured mesh. Each block communicates with the next one via a set of ghost cells. As a final comment, structured mesh are not the only option: unstructured mesh can also be used. In unstructured meshes, in order to retrieve neighboring information for a given point, an indirection table is needed, because each point can have a different connectivity with its neighbors. Despite the need of more complex data structures, unstructured meshes are very common in industrial applications because they are tailored for automated mesh generation. We will not detail further these technical aspects, we refer the reader to the specialized literature such as [Thompson et al. \(1998\)](#).

### 5.2.2 Discretization of the convective term

In order to discuss the discretization strategy for the convective term we reduce the generic system described by eq. (5.2) to:

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) = \mathbf{0} \quad (5.13)$$

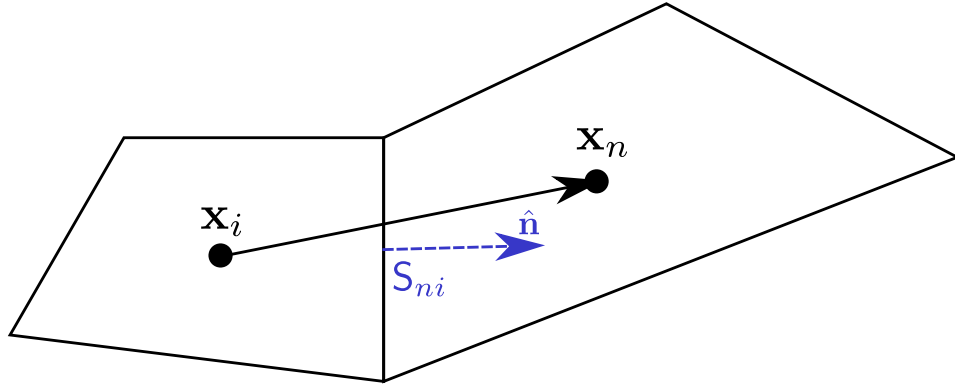


Figure 5.3: Schematic of a mesh cell with its neighbor

this will allow to briefly discuss about the properties of the solutions of such kind of systems more easily. The **FVM** integral formulation of eq. (5.13) is:

$$|\Omega_i| \frac{\partial q_i}{\partial t} + \int_{\Omega_i} \nabla \cdot F(q) = 0 \quad (5.14)$$

using the divergence theorem, the second term can be expressed as:

$$|\Omega_i| \frac{\partial q_i}{\partial t} + \oint_{\partial\Omega_i} F(q) \cdot \hat{n} = 0 \quad (5.15)$$

In the **FVM** framework, discretizing the convective term means providing a sound approximation of:

$$\oint_{\partial\Omega_i} F(q) \cdot \hat{n} \quad (5.16)$$

that is, for polyhedral mesh cells,

$$\int_{\Omega_i} \nabla \cdot F(q) = \oint_{\partial\Omega_i} F(q) \cdot \hat{n} \approx \sum_f^{N_{\text{faces}}} [F(q) \cdot \hat{n}]_f S_f \quad (5.17)$$

where  $[\bullet]_f$  is a sound approximation of  $\bullet$  on the face  $f$ , very often expressed as a linear combination of the value  $\bullet$  on the neighboring cells;  $S_f$  is the surface of the face between the cell  $\Omega_i$  and its neighbor, as shown in Fig. 5.3.

### 5.2.2.1 Eigenstructure

The discussion of the eigenstructure of a full 3D system may prove to be very complex, hence we shall restrict further the discussion here to a system where  $N_{\text{dim}} = 1$ . This assumption



is not too constraining since, we will see, when we discretize the surface integral shown in eq. (5.14), we shall deal with normal projection of the multi-dimensional convective flux

$$F_{\hat{n}}(\mathbf{q}) \triangleq \mathbf{F}(\mathbf{q}) \cdot \hat{\mathbf{n}}$$

and also because invariance by rotation also holds for the set of equations we are examining. Equation (5.13) is called *conservative form*. In case of smooth solutions, the conservative form is equivalent to the *quasi-linear* form

$$\frac{\partial \mathbf{q}}{\partial t} + \mathbf{A}(\mathbf{q}) \frac{\partial \mathbf{q}}{\partial x} = \mathbf{0} \quad (5.18)$$

$\mathbf{A}(\mathbf{q}) = \partial \mathbf{F}_{\hat{n}} / \partial \mathbf{q}$  is the jacobian of the system. The properties of the  $\mathbf{A}(\mathbf{q})$  (and hence of the derivatives of  $\mathbf{F}(\mathbf{q})$ ) are important in order to ensure the well-posedness of the problem described by eq. (5.13).

#### Definition 3

With *hyperbolic system* we define a system such that it can be written as in eq. (5.18) and its jacobian  $\mathbf{A}(\mathbf{q})$  has  $N_{\text{eqs}}$  real eigenvalues  $\lambda_1(\mathbf{q}) \leq \dots \leq \lambda_a \leq \dots \leq \lambda_{N_{\text{eqs}}}(\mathbf{q})$  with  $N_{\text{eqs}}$  corresponding distinct right eigenvectors  $\mathbf{r}_{N_{\text{eqs}}}(\mathbf{q})$ . *i.e.*  $\mathbf{A}(\mathbf{q})$  is diagonalizable with real eigenvalues.

#### Definition 4

If the eigenvalues  $\lambda_a(\mathbf{q})$  are also distinct, then the system 5.13 is *strictly hyperbolic*.

For a strictly hyperbolic system, we can associate to each  $\lambda_a$  a corresponding right eigenvector  $\mathbf{r}_a$ , such that

$$\mathbf{A}(\mathbf{q}) \mathbf{r}_a = \lambda_a \mathbf{r}_a \quad (5.19)$$

and a left eigenvector,

$$\mathbf{l}_a^T \mathbf{A}(\mathbf{q}) = \lambda_a \mathbf{l}_a^T \quad (5.20)$$

such that we can rewrite the system eq. (5.18) as

$$\mathbf{l}_a(\mathbf{q}) \left( \frac{\partial \mathbf{q}}{\partial t} + \lambda_a(\mathbf{q}) \frac{\partial \mathbf{q}}{\partial x} \right) = 0 \quad (5.21)$$

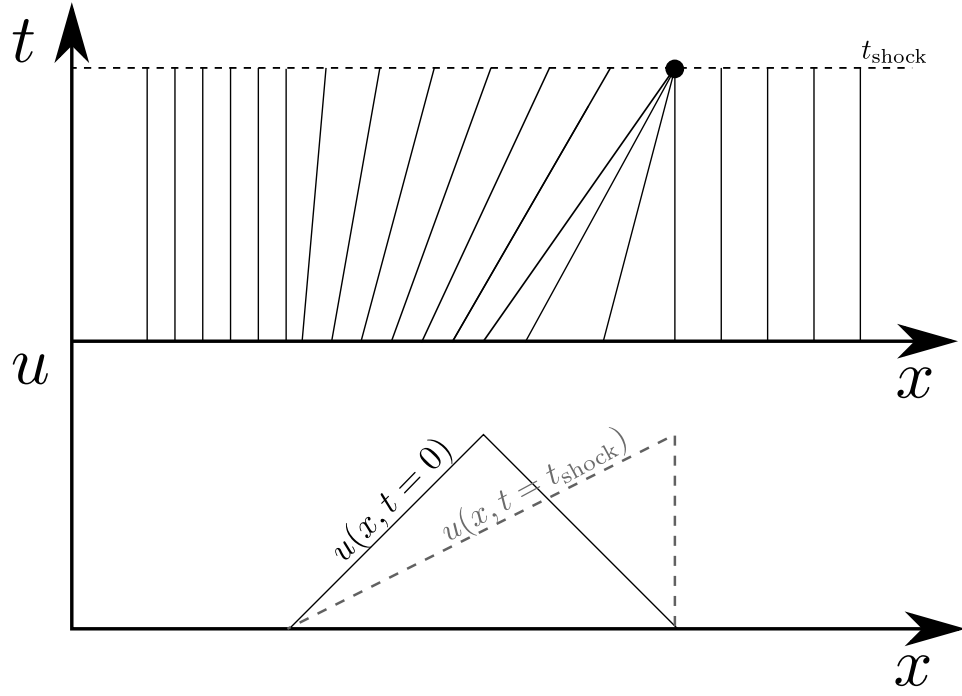


Figure 5.4: Example of characteristics curves for a “small hat” problem for the Burgers equation, courtesy of Massot, Series, et al. (2020)

We also introduce a *Riemann invariant*  $\iota(q)$  as a smooth function that satisfies eq. (5.22).

$$\nabla \iota(q) \cdot \mathbf{r}_a = 0, \forall q \quad (5.22)$$

#### Definition 5

A *a-simple-wave* associated to the eigenvalue  $\lambda_a$  is a smooth solution  $\mathbf{q}(\mathbf{x}, t)$  of eq. (5.13) for which  $\iota(q)$  is constant on the domain for any *a*-Riemann invariant  $\iota$ .

It can be shown (Godlewski and Raviart 1996) that a *a-simple wave* is constant along the curves described by:

$$\frac{dx}{dt} = \lambda_a(\mathbf{q}(x, t)) \quad (5.23)$$

called *characteristics curves*, and that these characteristic curves are straight lines. Hence eq. (5.21) are also called *characteristic equations*.

### 5.2.2.2 Weak solutions and Rankine-Hugoniot relations

For the scalar problem ( $N_{\text{eqs}} = 1$ ), when the solution is smooth, the solution of the problem can be calculated along the characteristic curves as shown in eqs. (5.21) and (5.23). Sometimes, in the plane  $(x, t)$ , as shown in Fig. 5.4, two characteristic curves will intersect at some point  $(\bar{x}, \bar{t})$ , this is typical of non-linear hyperbolic systems. From that point in time on, there is not anymore a bijectivity relation that allows to compute the solution at that point knowing the initial solution and proceeding along a characteristic curve: we have the insurgence of a discontinuity called *shock*. If we have a discontinuity we are not allowed anymore to use the quasi-linear form of the equations (eq. (5.18)); we need to find our solutions in another way: we will leverage the conservative nature of the equations (eq. (5.13)) to find the so-called **weak solutions** in order to allow piecewise  $\mathcal{C}^1$  continuous solutions (*i.e.* solutions with jump discontinuities).

Let us start from our system written in conservative form:

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) = \mathbf{0} \quad (5.24)$$

$\mathbf{q} : \mathbb{R}^{N_{\text{dim}}} \times [0, +\infty[ \rightarrow \Omega \subset \mathbb{R}^{N_{\text{eqs}}}$  is a smooth solution of the problem if it satisfies eq. (5.24) with the initial conditions

$$\mathbf{q}(\mathbf{x}, 0) = \mathbf{q}_0(\mathbf{x}) \quad (5.25)$$

if we take  $\mathbf{w} \in \mathcal{C}_0^1(\Omega \times [0, +\infty])^p$  the continuous functions with compact support over  $(\Omega \times [0, +\infty])$ , we can write the conservative equation in integral form, and we can also rearrange it exploiting the integral reduction theorems:

$$\begin{aligned} \int_{\Omega} \int_0^{+\infty} \left( \frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) \right) \cdot \mathbf{w} \, dt \, d\mathbf{x} = \\ \int_{\Omega} \int_0^{+\infty} \left[ \frac{\partial \mathbf{w}}{\partial t} \cdot \mathbf{q} + \mathbf{F}(\mathbf{q}) \cdot (\nabla \cdot \mathbf{w}) \right] \, dt \, d\mathbf{x} + \int_{\Omega} \mathbf{q}(\mathbf{x}, 0) \cdot \mathbf{w}(\mathbf{x}, 0) \end{aligned} \quad (5.26)$$

Since eqs. (5.24) and (5.25) are naturally satisfied for smooth solutions, eq. (5.26) is also equal to 0 if  $\mathbf{q}$  is a smooth solution, *i.e.* it satisfies eq. (5.27) naturally.

$$\int_{\Omega} \int_0^{+\infty} \left[ \frac{\partial \mathbf{w}}{\partial t} \cdot \mathbf{q} + \mathbf{F}(\mathbf{q}) \cdot (\nabla \cdot \mathbf{w}) \right] \, dt \, d\mathbf{x} + \int_{\Omega} \mathbf{q}(\mathbf{x}, 0) \cdot \mathbf{w}(\mathbf{x}, 0) = 0 \quad (5.27)$$

We can then define a *weak solution* as a solution  $\mathbf{q}(\mathbf{x}, t) \in \Omega$  that satisfies eq. (5.27) for arbitrary  $\mathbf{w} \in \mathcal{C}_0^1(\Omega \times [0, +\infty])^p$ . Also, a weak solution that is regular is a strong solution.

Let us retrieve now what kind of relations need to be imposed if  $\mathbf{q}$  is piecewise continuous in our solution space. Let us assume without loss of generality that  $\mathbf{q} \in \mathcal{C}^1$  everywhere except in a neighborhood of  $\mathcal{S}$ , a discontinuity surface oriented with a normal unit vector  $\hat{\mathbf{n}} = (\hat{n}_t, \hat{n}_x)$  across which  $\mathbf{q}$  undergoes a jump. Let  $\mathbf{P}(\mathcal{S})$  be a point of  $\mathcal{S}$  and  $\mathcal{B}_{\mathbf{P}}$  a ball centered in  $\mathbf{P}$ . Proceeding as suggested in [Godlewski and Raviart \(1996\)](#), considering  $\mathcal{S} \cap \mathcal{B}_{\mathbf{P}}$  the only discontinuity surface for  $\mathbf{q}$ , we can write eq. (5.27) in this situation as:

$$\begin{aligned} \int_{\mathcal{B}_{\mathbf{P}}} \int_0^{+\infty} \left[ \frac{\partial \mathbf{w}}{\partial t} \cdot \mathbf{q} + \mathbf{F}(\mathbf{q}) \cdot (\nabla \cdot \mathbf{w}) \right] dt d\mathbf{x} = \\ \int_{\mathcal{B}_{\mathbf{P}}^+} \int_0^{+\infty} \dots + \int_{\mathcal{B}_{\mathbf{P}}^-} \int_0^{+\infty} \dots = 0 \end{aligned} \quad (5.28)$$

Splitting the integral in two contributions for the two components, before  $\bullet^-$  and after  $\bullet^+$  the wave, of  $\mathcal{B}_{\mathbf{P}} = \mathcal{B}_{\mathbf{P}}^+ \cup \mathcal{B}_{\mathbf{P}}^-$  and applying the integral reduction theorems, we get:

$$\begin{aligned} 0 = & - \int_{\mathcal{B}_{\mathbf{P}}^+} \int_0^{+\infty} \left( \frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) \right) \cdot \mathbf{w} d\mathbf{x} dt \\ & - \int_{\mathcal{B}_{\mathbf{P}}^-} \int_0^{+\infty} \left( \frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) \right) \cdot \mathbf{w} d\mathbf{x} dt \\ & - \oint_{\mathcal{S} \cap \mathcal{B}_{\mathbf{P}}} (\hat{n}_t \mathbf{q}^+ + \hat{n}_x \cdot \mathbf{F}(\mathbf{q})) \cdot \mathbf{w} dA \\ & + \oint_{\mathcal{S} \cap \mathcal{B}_{\mathbf{P}}} (\hat{n}_t \mathbf{q}^- + \hat{n}_x \cdot \mathbf{F}(\mathbf{q})) \cdot \mathbf{w} dA \end{aligned} \quad (5.29)$$

The first two terms of eq. (5.29) are null because  $\mathbf{q}$  is smooth on each side of  $\mathcal{B}_{\mathbf{P}}$ , i.e. it satisfies eq. (5.24). The other two terms lead to:

$$\oint_{\mathcal{S} \cap \mathcal{B}_{\mathbf{P}}} (\hat{n}_t \mathbf{q}^+ + \hat{n}_x \cdot \mathbf{F}(\mathbf{q})) - (\hat{n}_t \mathbf{q}^- + \hat{n}_x \cdot \mathbf{F}(\mathbf{q})) \cdot \mathbf{w} dA \quad (5.30)$$

Defining the jump operator  $[\bullet]$  across the discontinuity as

$$[\bullet] = \bullet^+ - \bullet^-$$

and assuming  $\mathcal{S}$  to be a 1D curve with a parametrization  $\mathcal{S} : t \rightarrow \xi(t)$  for which  $\hat{\mathbf{n}} = (-s, 1)^T$ , we obtain from eq. (5.30), for arbitrary  $\mathbf{w}$ , the so-called **Rankine-Hugoniot** jump conditions:

$$s[\mathbf{q}] = [\mathbf{F}(\mathbf{q})] \quad (5.31)$$

Equation (5.31), for  $N_{\text{eqs}} > 1$ , is a set of relations that ensure  $\mathbf{q}$  is a weak solution of the system eq. (5.24) and  $s$  is the speed of the discontinuity.

Weak solutions are in generally not unique. In order to select the physical (*i.e.* *entropic*) solutions among the multitude of possibilities, *entropy conditions* must be imposed. There are various ways of imposing the entropy conditions, for simple cases one solution is to use the *Lax entropy conditions* that translate into specific relations between a discontinuity speed and the eigenvalues of the system computed with the states  $\mathbf{q}^+$ ,  $\mathbf{q}^-$  across the discontinuity, based on the nature of the characteristic fields associated to each eigenvalue — *i.e.* if they are *genuinely non linear* or *linearly degenerate*. Other approaches are also possible, *i.e.* *vanishing viscosity method*, such that admissible shocks are chosen as the limit of viscous profiles. We will not further detail these aspects, in practical cases of our interest these conditions are enforced with simple comparison of the pressure values (or equivalent comparisons on other fields across each discontinuity), the interested reader can refer to [Godlewski and Raviart \(1996, Chap. I, Section 5\)](#) for additional details.

### 5.2.2.3 The Riemann Problem

The *Riemann Problem (RP)* is a special case of a 1D **Initial Value Problem (IVP)** for the system of eq. (5.24) defined as finding the self-similar solution  $\mathbf{q}_{\text{RP}}(x/t; \mathbf{q}_L, \mathbf{q}_R)$  on a given domain  $x \in \mathbb{R}$ ,  $t > 0$ , where the solution has a special initial conditions in which two different discontinuous values ( $\mathbf{q}_L, \mathbf{q}_R$ ) at the boundaries of the domain undergo a discontinuous jump at  $x = x_0$  that is for convenience taken as  $x_0 = 0$ . It is a generalization of the Sod's shock tube ([Sod 1978](#)) for the Euler system with arbitrary left and right conditions and an arbitrary system of equations.

$$\begin{cases} \frac{\partial \mathbf{q}}{\partial t} + \mathbf{F}(\mathbf{q}) = \mathbf{0} \\ \mathbf{q}(x, 0) = \mathbf{q}_L, \quad x \in [x_R, 0] \\ \mathbf{q}(x, 0) = \mathbf{q}_R, \quad x \in (0, x_L] \end{cases} \quad (5.32)$$

If the system has  $N_{\text{eig}}$  eigenvalues, hence  $N_{\text{eig}}$  characteristic fields, we have *elementary waves* associated to each  $k$  field. These elementary waves, under certain conditions, can be defined as self-similar solutions that connect two states across the wave, *i.e.* **rarefaction waves**, or **discontinuities**, when weak solutions are required. Among the discontinuities, we can sort out **contact discontinuities**, in the case the  $k$  characteristic field is linearly degenerate, or **shock waves**, for genuinely non linear characteristic fields. Under the assumption of  $|\mathbf{q}_L - \mathbf{q}_R|$  “small” ([Godlewski and Raviart 1996](#)), we can retrieve the solution to eq. (5.32) as a succession of piecewise constant fields across a series of elementary waves (rarefaction waves, contact dis-

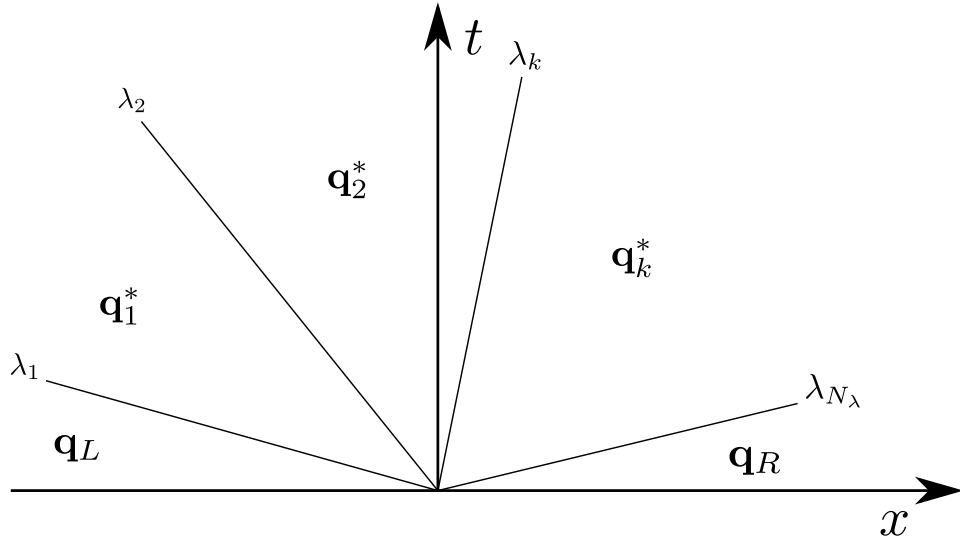


Figure 5.5: A general Riemann problem wave pattern

continuities or shocks). Fig. 5.5 shows a graphical representation of the previous explanation.

Most of the times, it does not exist a closed analytical solution for the “star-states”  $q_k^*$  across the elementary waves, and in practical situations as for example the RP for the Euler system, the problem must be solved numerically and it often involves solving multiple non-linear systems and ODEs (section 5.3.2). An in-depth discussion is provided by Godlewski and Raviart (1996, Chap. I, Section 6).

#### 5.2.2.4 The Godunov method

The Godunov method (Godunov 1959) is a clever numerical strategy to find the solution to eq. (5.24) leveraging the knowledge of the solution  $q_{RP}(x/t; q_L, q_R)$  of a 1D RP across two discontinuous states. The original paper exploits the complete solution of the RP and it also allows to provide an expression for the term  $\sum_f^{N_{\text{faces}}} [F(q) \cdot \hat{n}]_f S_f$  of the semi-discretized form of the convective term of eq. (5.2):

$$\int_{\Omega_i} \nabla \cdot F(q) = \oint_{\partial\Omega_i} F(q) \cdot \hat{n} \approx \sum_f^{N_{\text{faces}}} [F(q) \cdot \hat{n}]_f S_f$$

The key concept is understanding that the discretized domain is composed by a number of cells on which, in the case of first-order FVM schemes, the solution field  $q$  is piecewise constant, then for each pair  $(i, i + 1)$  of cell-neighbor, we basically have a RP to solve, where  $q_L = q_i$ ,  $q_R = q_{i+1}$  along the normal direction  $\hat{n}$  to the face shared by the two cells. Once we have the solution  $q_{RP}(x/t; q_i, q_{i+1})$  of the problem we can evaluate the intercell flux  $[F(q) \cdot \hat{n}]_f S_f$

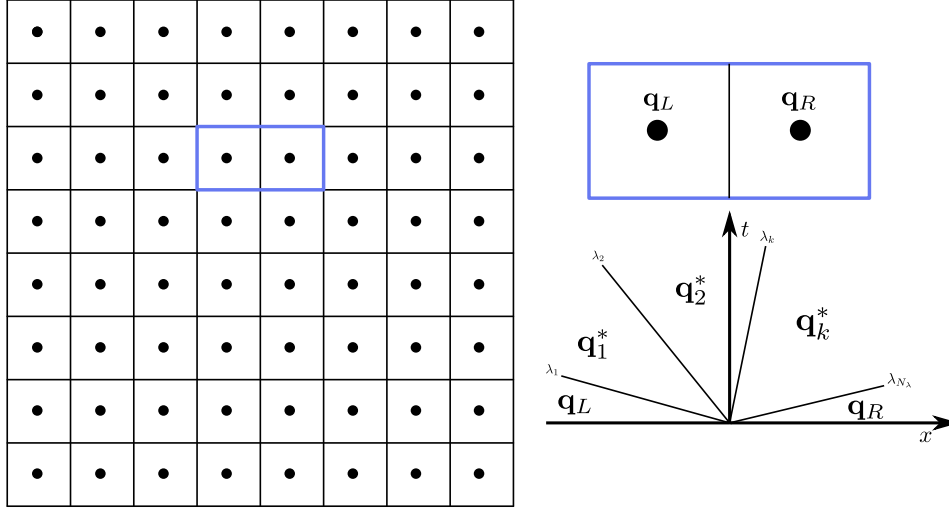


Figure 5.6: The Godunov method applied to a mesh

as:

$$[F(q) \cdot \hat{n}]_f S_f = F(q_{RP}(0, q_i, q_{i+1})) \cdot \hat{n}_f S_f \quad (5.33)$$

that is, the intercell flux is computed with the solved Riemann problem state computed at the interface  $x/t = 0$ . This is equivalent to compute the real flux across the cell face.

#### 5.2.2.5 Approximate Riemann Solvers

In section 5.2.2.3 we presented the general aspects of the solution of the Riemann Problem (RP), and in section 5.2.2.4 we also explain how the solution of the RP can be employed for the numerical simulation of a PDE system using the Godunov method. We also mentioned that in fact, for general EoS, the solution of the RP features multiple ODE solutions and non-linear root finding and it is, hence, very costly to implement on large meshes. For this reason, it is common to look for *Approximate Riemann solvers* that are faster computational-wise, still retaining the important properties that ensure the consistent convergence of the solution.

The strategies are variegated. One strategy is based on the quasi-linear form of the system eq. (5.18)

$$\frac{\partial q}{\partial t} + A(q) \cdot \nabla q = 0$$

in which the jacobian matrix  $A(q)$ , that varies with the state, is replaced by a sound approximation that is instead **constant**.

$$A(q) \sim \bar{A}(q_L, q_R)$$

In this way the non-linear hyperbolic system eq. (5.18) becomes a linear hyperbolic system

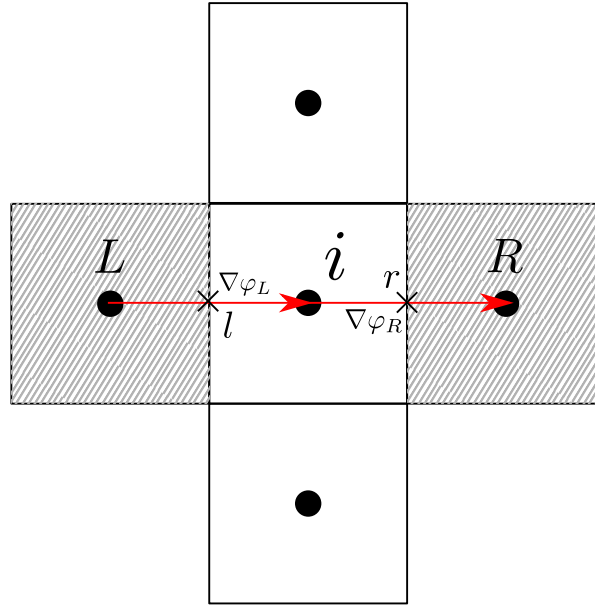


Figure 5.7: Slopes reconstruction in structured meshes

that is easier to handle and faster when used as numerical scheme in a simulation. An example of this strategy is the Roe's solver (Roe 1981). For further details we refer to LeVeque (1990); LeVeque (2002); Eleuterio F. Toro (2009). Another possible strategy leverages the interpretation of the RP as the interaction of simple waves. The idea is to postulate a wave structure, that is not required to provide the same number of waves as provided by the physical system eigenstructure, and the wave speeds are then estimated from the available state values  $(q_L, q_R)$ . Once the wave speeds are known, the Rankine-Hugoniot (eq. (5.31)) conditions can be enforced across each discontinuity wave, relating the states on the left with the ones on the right of the discontinuity, leading to an approximation of the “star-state”  $q^*$  and, hence, of the inter-cell flux  $F(q^*)$ . The Rusanov scheme (Rusanov 1962), the HLL (Amiram Harten et al. 1983) and the HLLC (Eleuterio F. Toro 2019) can be formulated under this framework. Other approaches are also possible, the interested reader is referred to LeVeque (1990); LeVeque (2002); Eleuterio F. Toro (2009) for a complete description of numerical methods for hyperbolic systems.

#### 5.2.2.6 Higher order methods

Until now, all the method we introduced present a problem of high numerical viscosity and they behave poorly in situations where high gradients of the state are present (*e.g.* in the case of contact discontinuities and shocks), the profiles that are computed by those schemes appear smeared out or they require very fine meshes to counteract the numerical diffusion. In order



to alleviate the problem, the Godunov method (section 5.2.2.4) can be extended considering piecewise linear functions locally on each cell  $\Omega_i$ ,

$$\mathbf{q}(\mathbf{x}) = \mathbf{q}_i + \nabla \mathbf{q}_i \cdot (\mathbf{x} - \mathbf{x}_i) \quad (5.34)$$

the gradient  $\nabla \mathbf{q}$  might be approximated by the methods we discuss in section 5.2.4. Unfortunately, these slopes cannot be estimated without additional constraints, because otherwise they may produce instabilities. Ami Harten (1997) in fact explains the framework of **Total variation diminishing (TVD)** schemes which provides the right constraints, or “limiters”, that need to be enforced in order to retain robustness of the scheme. In other words, the value of the slope cannot be reconstructed freely, but it needs to be **limited**. One possible solution for structured meshes, as shown in Fig. 5.7, is to consider a stencil including the cell  $i$ , and the direct neighbors  $L$  and  $R$ . Two possible slopes can be computed,

$$\nabla \mathbf{q}_R = \frac{\mathbf{q}_R - \mathbf{q}_i}{x_L - x_i} \quad (5.35)$$

$$\nabla \mathbf{q}_L = \frac{\mathbf{q}_i - \mathbf{q}_L}{x_i - x_L} \quad (5.36)$$

then the effective slope is taken as ( $f = L, R$ ):

$$\nabla \mathbf{q}'_i = \psi(r) \nabla \mathbf{q}_i \quad (5.37)$$

where  $r = \nabla \mathbf{q}_L \cdot \hat{\mathbf{n}}_L / \nabla \mathbf{q}_R \cdot \hat{\mathbf{n}}_R$  is the ratio of successive gradients and  $\psi(r)$  is the “slope limiter”. As an example, the MINMOD limiter is:

$$\psi_{\text{MINMOD}}(r) = \max(0, \min(1, r)) \quad (5.38)$$

With the limited slope computed, the state can be reconstructed on the two opposite faces  $\mathbf{q}_l, \mathbf{q}_r$ . These two new left and right states can then be used in a Riemann solver to advance in time as we discussed in section 5.2.2.5. A possible approach is given by the MUSCL-Hancock approach, that is composed by three steps:

- Limited linear reconstruction of the state on the faces at the time instant  $k$  as provided by eq. (5.34), with the gradient computed as per eq. (5.37), providing  $(\mathbf{q}(\mathbf{x}_L) = \mathbf{q}_L^k, \mathbf{q}(\mathbf{x}_R) = \mathbf{q}_R^k)$
- A prediction step  $\Delta t/2$  long in which intermediate predicted states at the interface be-

tween the cells and its neighbors are obtained

$$\mathbf{q}_L^{k+1/2} = \mathbf{q}_L^k - \frac{\Delta t}{2} \frac{S_L}{|\Omega_i|} (F(\mathbf{q}_R) - F(\mathbf{q}_L)) \quad (5.39)$$

$$\mathbf{q}_R^{k+1/2} = \mathbf{q}_R^k - \frac{\Delta t}{2} \frac{S_R}{|\Omega_i|} (F(\mathbf{q}_R) - F(\mathbf{q}_L)) \quad (5.40)$$

$$(5.41)$$

- Using the approximate Riemann solver of choice with left and right state respectively  $\mathbf{q}_L^{k+1/2}, \mathbf{q}_R^{k+1/2}$  to compute the numerical flux at the interface

$$[F(\mathbf{q}) \cdot \hat{\mathbf{n}}]_f S_f \approx F\left(\mathbf{q}_{\text{RP}}\left(0, \mathbf{q}_i, \mathbf{q}_f^{k+1/2}\right)\right) \cdot \hat{\mathbf{n}}_f S_f \quad (5.42)$$

The MUSCL-Hancock approach to build high-order schemes is not the only choice possible. More alternatives can be found, as for example the WENO schemes (Shu 2003), OSTVD high-order schemes (Daru and Tenaud 2004) or Discontinuous Galerkin (DG) methods (Cockburn and Shu 2001), possibly coupled with TVD time schemes (Gottlieb and Shu 1996; Gottlieb and Shu 1998). We refer the reader also to complete overviews on the subject of LeVeque (1990); Godlewski and Raviart (1996); LeVeque (2002); Bouchut (2004); Eleuterio F. Toro (2009) among others and a multi-slope approach for unstructured meshes described in Le Touze et al. (2014).

### 5.2.3 Discretization of the non-conservative term

The non conservative term

$$\int_{\Omega_i} \mathbf{B}(\mathbf{q}) \cdot \nabla \mathbf{q} \quad (5.43)$$

is quite tricky to approximate. If  $\mathbf{q}$  is discontinuous, for example in presence of shocks, the integral eq. (5.43) can be ill-posed. A satisfying theory on how to treat this term does not exist, different attempts are indeed present in literature (Gallice 2002; Castro and E. F. Toro 2006; Wagnier et al. 2019). In specific cases, like the system from Baer and Nunziato (1986) (section 5.4), it is possible to choose the closure for some terms of the model (notably the interfacial velocity and pressure) in order to have the correct behavior in case of shocks (Coquel, Gallouët, et al. 2002). For the sake of this work, we will limit ourselves to approximate the

term as follows:

$$\int_{\Omega_i} \mathbf{B}(\mathbf{q}) \cdot \nabla \mathbf{q} \approx \langle \mathbf{B}(\mathbf{q}) \rangle_{\Omega_i} \cdot \oint_{\partial\Omega_i} (\mathbf{q} \otimes \hat{\mathbf{n}}) \approx \langle \mathbf{B}(\mathbf{q}) \rangle_{\Omega_i} \cdot \sum_f^{N_{\text{faces}}} [\mathbf{q} \otimes \hat{\mathbf{n}}]_f S_f \quad (5.44)$$

That is, the non-conservative pre-multiplier  $\mathbf{B}(\mathbf{q})$  is computed as a sound evaluation at the cell center, and considered constant over the cell volume, allowing to use the gradient theorem for the rest of the term. For the purposes at stake in this work,  $\mathbf{B}(\mathbf{q})$  is generally made by a combination of fluid velocities and pressures (*e.g.* in the B-N model (Baer and Nunziato 1986)), for this reason most of the times the  $\langle \mathbf{B}(\mathbf{q}) \rangle_{\Omega_i}$  is estimated with an upwind scheme of some sort. We will show that for the problems we are interested in, this approximation scheme provides satisfying results.

### 5.2.4 Discretization of the diffusive term

The diffusive term

$$\int_{\Omega_i} \nabla \cdot (\mathbf{K}(\mathbf{q}) \cdot \nabla \mathbf{q}) \quad (5.45)$$

is approximated applying the divergence theorem:

$$\int_{\Omega_i} \nabla \cdot (\mathbf{K}(\mathbf{q}) \cdot \nabla \mathbf{q}) \approx \langle \mathbf{K}(\mathbf{q}) \rangle_{\Omega_i} \cdot \oint_{\partial\Omega_i} \nabla \mathbf{q} \cdot \hat{\mathbf{n}} \approx \langle \mathbf{K}(\mathbf{q}) \rangle_{\Omega_i} \cdot \sum_f^{N_{\text{faces}}} [\nabla \mathbf{q} \cdot \hat{\mathbf{n}}]_f S_f \quad (5.46)$$

$\mathbf{K}(\mathbf{q})$  is the (potentially 4<sup>th</sup>-order) viscosity tensor and it is generally provided by the constitutive equation of the fluid under study and it needs to be computed as a sound mean value over the cell volume. As it is possible to note from eq. (5.46), the diffusive scheme FVM approximation needs an estimation of the state gradient  $\nabla \mathbf{q}$  at the cell face. This estimation can be provided with different strategies, they can account for contributions from a variable amount of neighboring cells.

#### 5.2.4.1 Forward differencing

This method is the easiest one. If we check eq. (5.46), we see that what we need is the normal component to the face between a cell and the corresponding neighbor. Hence, the naive way of approximating a normal gradient is,

$$[\nabla \mathbf{q} \cdot \hat{\mathbf{n}}]_f S_f = \frac{\mathbf{q}_n - \mathbf{q}_i}{\|\Delta \mathbf{x}_{ni}\|} S_{ni} \quad (5.47)$$

where  $\mathbf{x}$  indicates the position of the cell centroid,  $S_{ni}$  the surface of the face shared by the cell and its neighbor and  $\Delta\mathbf{x}_{ni}$  the relative distance between a cell  $\Omega_i$  and the corresponding neighbor  $\Omega_n$  as shown in Fig. 5.8. Equation (5.47) is correct for pair of cells in which the two centroids are aligned with the normal direction to the face. Examining Fig. 5.3, we can see that when the centroids are misaligned, a *non-orthogonal* correction is required to improve the estimation of the gradient. Moukalled et al. (2016), among others, provide details on the subject.

#### 5.2.4.2 Green-Gauss

This strategy expects the gradient to be volume-averaged locally to the cell  $i$ ,

$$\oint_{\partial\Omega_i} \nabla \mathbf{q} \cdot \hat{\mathbf{n}} = |\Omega_i| \langle \nabla \mathbf{q} \rangle_{\Omega_i} \cdot \sum_f^{N_{\text{neigh.}}} \hat{\mathbf{n}}_f S_f \quad (5.48)$$

and then the average gradient value is approximated via the Green formula,

$$|\Omega_i| \langle \nabla \mathbf{q} \rangle_{\Omega_i} = \sum_f^{N_{\text{neigh.}}} \mathbf{q}_f \otimes \hat{\mathbf{n}}_f S_f \quad (5.49)$$

In eq. (5.49),  $\mathbf{q}_f$  is a sound approximation of the state on the cell face. This value then needs to be retrieved using interpolation. One solution that requires a small stencil is:

$$\mathbf{q}_f = w_n \mathbf{q}_i + (1 - w_n) \mathbf{q}_n \quad (5.50)$$

where  $w_n$  is a tunable weight, generally taken as the inverse of the relative distance  $\Delta\mathbf{x}_{ni}$ .

$$w_n = \frac{1}{\|\Delta\mathbf{x}_{ni}\|} \quad (5.51)$$

#### 5.2.4.3 Least Squares gradient

This numerical recipe applies a least-squares optimization over all the adjacent cell neighbors to a given cell  $\Omega_i$ . We note that a generic field  $\varphi$  can be expressed on a neighbor cell  $\Omega_n$  w.r.t. to the cell  $\Omega_i$  with:

$$\varphi_n = \varphi_i + (\nabla \varphi) \cdot (\mathbf{x}_n - \mathbf{x}_i) \quad (5.52)$$

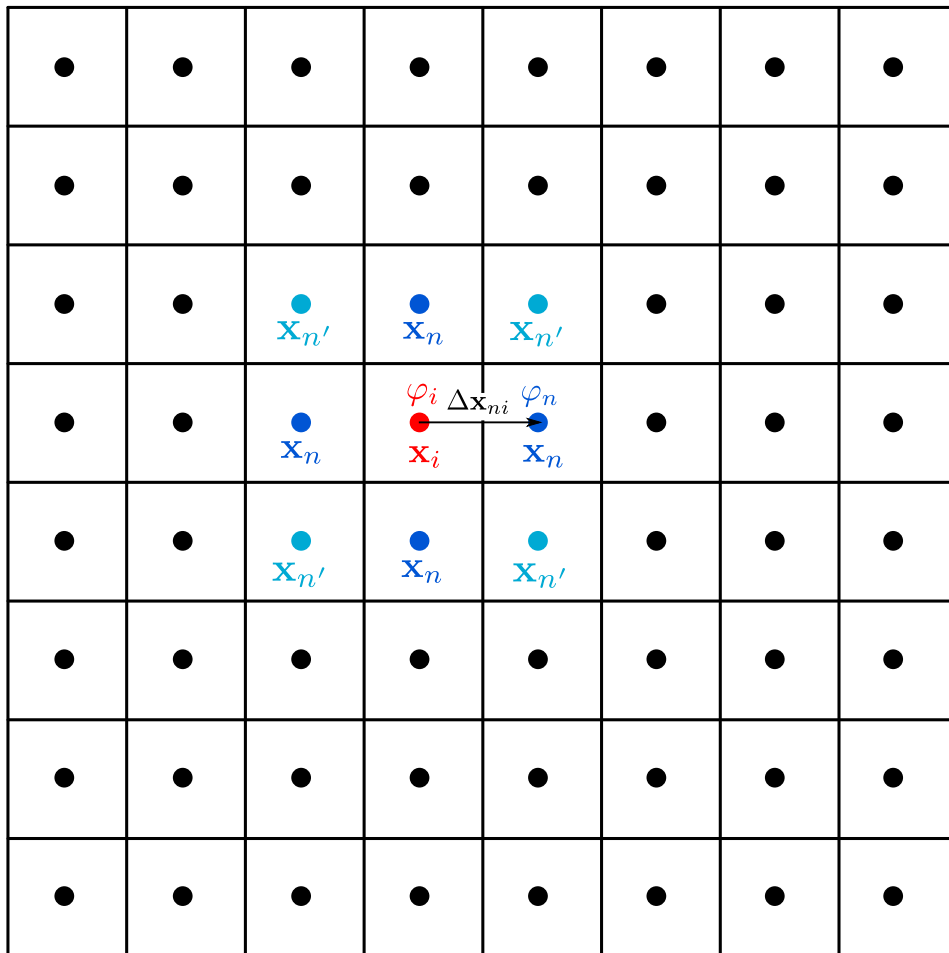


Figure 5.8: Estimating the gradient with a least square approach

We can then define a cost function  $G(\nabla\varphi; \varphi_n, \varphi_i, \mathbf{x}_i, \mathbf{x}_n)$  that takes into consideration all the neighbors of a cell

$$G(\nabla\varphi; \varphi_n, \varphi_i, \mathbf{x}_i, \mathbf{x}_n) = \sum_n^{N_{\text{neigh.}}} w_n [\varphi_n - (\varphi_i + (\nabla\varphi)_i) \cdot \Delta\mathbf{x}_{ni}]^2 \quad (5.53)$$

where  $w_n$  is a tunable weight, generally taken as the inverse of the relative distance  $\Delta\mathbf{x}_{ni}$ .  $N_{\text{neigh.}}$  is the number of neighbors of the point  $i$ . The set of neighbors of the point  $i$  in a structured mesh can also include the set of diagonal neighbors to improve the accuracy of the estimation, as shown in Fig. 5.8. The weights  $w_n$  defined as in eq. (5.51). If we optimize  $G$  to find a stationary point:

$$\frac{\partial G}{\partial \nabla\varphi} = 0$$

we obtain a linear system:

$$\mathbf{M}(w_n, \Delta\mathbf{x}_{ni}) \nabla\varphi = \mathbf{b}(w_n, \Delta\varphi_{ni}, \Delta\mathbf{x}_{ni}) \quad (5.54)$$

where  $\Delta\varphi_{ni} = \varphi_n - \varphi_i$ , and the matrix  $\mathbf{M}$  is calculated as follows:

$$\mathbf{M}(w_n, \Delta\mathbf{x}_{ni}) = \begin{bmatrix} \sum_n^{N_{\text{neigh.}}} w_n \Delta x_{ni} \Delta x_{ni} & \sum_n^{N_{\text{neigh.}}} w_n \Delta y_{ni} \Delta x_{ni} & \sum_n^{N_{\text{neigh.}}} w_n \Delta z_{ni} \Delta x_{ni} \\ \sum_n^{N_{\text{neigh.}}} w_n \Delta x_{ni} \Delta y_{ni} & \sum_n^{N_{\text{neigh.}}} w_n \Delta y_{ni} \Delta y_{ni} & \sum_n^{N_{\text{neigh.}}} w_n \Delta z_{ni} \Delta y_{ni} \\ \sum_n^{N_{\text{neigh.}}} w_n \Delta x_{ni} \Delta z_{ni} & \sum_n^{N_{\text{neigh.}}} w_n \Delta y_{ni} \Delta z_{ni} & \sum_n^{N_{\text{neigh.}}} w_n \Delta z_{ni} \Delta z_{ni} \end{bmatrix} \quad (5.55)$$

Please note that  $\mathbf{M}(w_n, \Delta\mathbf{x}_{ni})$  depends only on the geometry of the mesh, so if the mesh is not dynamic, it can be precomputed ahead of time and stored in memory, possibly in decomposed form. The RHS  $\mathbf{b}$  is computed with eq. (5.56),

$$\mathbf{b}(w_n, \Delta\varphi_{ni}, \Delta\mathbf{x}_{ni}) = \sum_n^{N_{\text{neigh.}}} w_n \Delta\varphi_{ni} \Delta\mathbf{x}_{ni} \quad (5.56)$$

and it depends on the actual field value at the cell  $\Omega_i$  and its neighbors  $\Omega_n$ , so it needs to be recomputed at each time step. Exploiting vectorized linear algebra primitives that also support broadcasting (like for example NumPy (Harris et al. 2020)), in the case of a structured mesh, the system of eq. (5.54) can be stacked for all the cells of the mesh, and for all the fields of

the state  $\mathbf{q}$  in order to provide a complete solution for the gradient  $\nabla \mathbf{q}$  all over the mesh cell centers, with optimized memory footprint

$$\widehat{\mathbf{M}}(w_n, \Delta \mathbf{x}_{ni}) \nabla \mathbf{q} = \hat{\mathbf{b}}(w_n, \Delta \varphi_{ni} \Delta \mathbf{x}_{ni}) \quad (5.57)$$

Once eq. (5.57) is solved and the gradient is available at each cell  $\Omega_i$ , the gradient at the face can be interpolated between the value at the cell and the value at its neighbor  $\Omega_n$  sharing the face  $f$ , for example with a simple mean:

$$|\nabla \mathbf{q}|_f = \frac{1}{2} (\nabla \mathbf{q}_i + \nabla \mathbf{q}_n) \quad (5.58)$$

Other interpolation schemes are possible, the interested reader is referred to [Moukalled et al. \(2016\)](#) for a detailed presentation of those possibilities.

### 5.2.5 Discretization of the source term

The source term:

$$\int_{\Omega} s(\mathbf{q}) \quad (5.59)$$

generally undergoes specific treatment depending on its nature: it can be casted as the gradient of some potential for conservative forces ( $s(\mathbf{q}) = \nabla \psi$ ) and be included in the discretization of the convective term, for example, in the same level as a pressure. As a generic presentation, if not otherwise specified, we simply discretize the source term as a volume-averaged constant value on the cell,

$$\int_{\Omega_i} s(\mathbf{q}) = \langle s(\mathbf{q}) \rangle_{\Omega_i} |\Omega_i| \quad (5.60)$$

the simplest choice for the volume-averaged constant value is to just retain the value of the source term calculated on the cell centroids,

$$\langle s(\mathbf{q}) \rangle_{\Omega_i} \approx s(\mathbf{q}_i)$$

This choice might not perform as expected for coarse cells or strongly non linear source terms ([LeVeque and Yee 1990](#)) which might required more advanced discretization techniques.

## 5.2.6 Time integration

Let us consider the full PDE system under consideration discretized with the FVM:

$$|\Omega_i| \frac{\partial \mathbf{q}}{\partial t} + \sum_f^{N_{\text{faces}}} [\mathbf{F}(\mathbf{q}) \cdot \hat{\mathbf{n}}]_f S_f + \sum_f^{N_{\text{faces}}} [\mathbf{q} \otimes \hat{\mathbf{n}}]_f S_f - \sum_f^{N_{\text{faces}}} [\nabla \mathbf{q} \cdot \hat{\mathbf{n}}]_f S_f + \langle s(\mathbf{q}) \rangle_{\Omega_i} |\Omega_i| = 0$$

and then grouping all the space terms in the term  $-\mathbf{f}(\mathbf{q}, t)$ , we achieve a semi-discretized form of the PDE system (eq. (5.61)),

$$\frac{\partial \mathbf{q}}{\partial t} = \mathbf{f}(\mathbf{q}, t) \quad (5.61)$$

which can be written in integral form:

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \int_t^{t+\Delta t} \mathbf{f}(\mathbf{q}^*, t^*) dt^* \quad (5.62)$$

The objective of the time scheme is to provide a sound discretization of the term  $\int_t^{t+\Delta t} \mathbf{f}(\mathbf{q}^*, t^*) dt^*$ . The  $*$  symbol indicates that the RHS can be treated *implicitly*, where we can have  $*$  =  $n + 1$ , or *explicitly*, where  $*$  =  $0 \dots n$ . When we employ an implicit time-scheme, being  $\mathbf{f}(\mathbf{q}^*, t^*)$  potentially non-linear, a root finding algorithm must be used at each time step to compute  $\mathbf{q}^{n+1}$ . For simplicity, in this context we will treat only explicit schemes and in particular Runge-Kutta schemes,

$$\left\{ \begin{array}{l} \mathbf{q}^{n+1} = \mathbf{q}^n + \Delta t \sum_i^s b_i K_i \\ K_1 = \mathbf{f}(\mathbf{q}^n, t^n) \\ \vdots \\ K_s = \mathbf{f}\left(\mathbf{q}^n + \Delta t \left(\sum_{l=1}^{s-1} a_{sl} K_l\right), t^n + c_s \Delta t\right) \end{array} \right. \quad (5.63)$$

where the coefficients  $a_s, b_s, c_s$  are specific to each Runge-Kutta (RK) scheme, and they are



generally graphically arranged in a *Butcher Tableau* (Butcher 1996) as the one shown in eq. (5.64).

$$\begin{array}{c|cccc}
 0 & & & & \\
 c_1 & a_{11} & & & \\
 c_2 & a_{12} & a_{22} & & \\
 \vdots & \dots & & & \\
 c_s & a_{s1} & a_{s2} & \dots & a_{ss-1} \\
 \hline
 & b_1 & b_2 & \dots & b_s
 \end{array} \tag{5.64}$$

Moreover, the coefficients can be also optimized in order to have **Strong Stability Preserving (SSP)** behavior (Gottlieb and Shu 1996; Gottlieb and Shu 1998), a property that is of special interest for the stability of high-order schemes in the simulation of conservation laws as the one we treat in this work. When they are coupled with spatial schemes that preserve this property, we talk about **TVD** schemes in the case of scalar equations.

#### 5.2.6.1 The CFL Condition

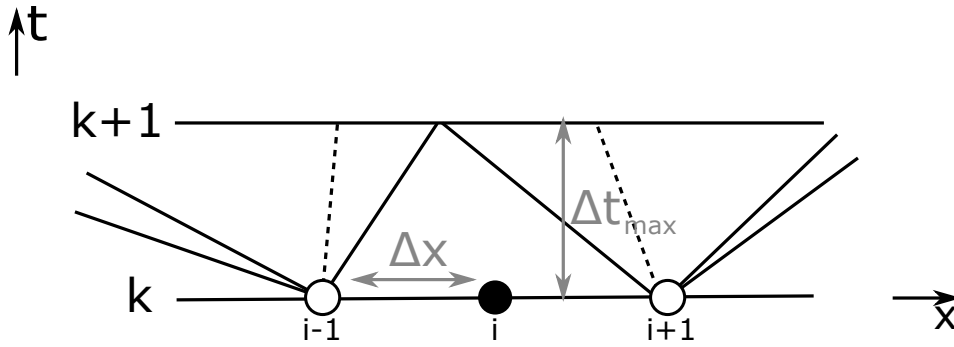


Figure 5.9: The maximum CFL number allowed by the propagation of the information in hyperbolic systems

A final note on the choice of the time step  $\Delta t$  for explicit time integration. For what concerns the convective term in hyperbolic systems, when using the Godunov method, as we do in this study, the time step cannot be chosen freely. If we look at Fig. 5.9 that shows the propagation of the information from a time instant  $n$  to the next one  $n+1$  for three generic points of the mesh  $i-1, i, i+1$  we see that for a certain  $\Delta t$ , the characteristic lines intersect leading to multivalued solutions. The characteristic waves are also shown since for each point of the domain we have a **RP** to solve. The maximum time step possible is driven by the maximum speed of the waves

along the entire mesh  $s_{\max} = \max(s_{i,\max})$ , *i.e.*

$$\Delta t_{\text{conv}} \leq \frac{\Delta x}{s_{\max}} \quad (5.65)$$

When the system also features diffusion terms, another limiting condition on the time step can be retrieved with a von Neumann stability analysis, that is:

$$\Delta t_{\text{diff}} \leq \frac{\Delta x^2}{2\mu} \quad (5.66)$$

where  $\mu$  is the viscosity coefficient. Therefore, in cases where both diffusive and convective terms are present in the system, the effective time step is takes an:

$$\Delta t = \min(\Delta t_{\text{conv}}, \Delta t_{\text{diff}}) \quad (5.67)$$

It might happen that in specific cases, the requirements on the time step from the von Neumann stability analysis might be too restrictive. In that case an implicit approach (N’Guessan 2020) or a high-order explicit scheme with extended stability (Duarte et al. 2012) might be more suitable.

## 5.3 The Euler system

The Euler system is an interesting PDE system of conservation laws of interest in high-speed external aerodynamics and internal combustion (combined with chemistry) other than mere mathematical delight. In its classical form it is defined as an homogeneous convective system that features strict hyperbolicity under common hypotheses for the EoS. In order to present it, let us recall our PDE reference model:

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) + \mathbf{B}(\mathbf{q}) \cdot \nabla \mathbf{q} - \nabla \cdot (\mathbf{K}(\mathbf{q}) \cdot \nabla \mathbf{q}) + s(\mathbf{q}) = \mathbf{0}$$

the Euler system is a homogeneous convective system,

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) = \mathbf{0}$$

where the state is given by:

$$\mathbf{q} = (\rho, \rho u, \rho v, \rho w, \rho E; \rho e, u, v, w, p, c) \quad (5.68)$$

and the convective flux is defined as

$$\mathbf{F}(\mathbf{q}) = \begin{bmatrix} \rho u & \rho v & \rho w \\ \rho u^2 + p & \rho uv & \rho uw \\ \rho vu & \rho v^2 + p & \rho vw \\ \rho wu & \rho wv & \rho w^2 + p \\ (\rho E + p)u & (\rho E + p)v & (\rho E + p)w \end{bmatrix} \quad (5.69)$$

### 5.3.1 Eigenstructure

Applying a change of variables to the system, in order to ease the computations, the eigenstructure of the system does not change and the hyperbolicity is ensured for classical **EoS**, it is then possible to find that the eigenvalues of the Euler system are:

$$\lambda_{1-5}(\mathbf{q}) = (u + c, u, u, u, u - c) \quad (5.70)$$

with corresponding right eigenvectors  $\mathbf{r}$ , that in the case the system is casted in *primitive variables*  $\tilde{\mathbf{q}} = (\rho, u, v, w, p)^T$ , read:

$$\mathbf{r}_{15} = \begin{bmatrix} \rho \\ \pm c \\ 0 \\ 0 \\ \rho c^2 \end{bmatrix}, \mathbf{r}_2 = \begin{bmatrix} 1 \\ 0 \\ v \\ w \\ 0 \end{bmatrix}, \mathbf{r}_3 = \begin{bmatrix} \rho \\ 0 \\ 1 \\ w \\ 0 \end{bmatrix}, \mathbf{r}_4 = \begin{bmatrix} \rho \\ 0 \\ v \\ 1 \\ 0 \end{bmatrix} \quad (5.71)$$

The characteristic fields associated to the eigenvalues  $\lambda_{2-4}$  are linearly degenerate and they correspond to *shear waves*, across which we have changes of the tangential velocities  $(v, w)$ . Moreover, analyzing the components of  $\mathbf{r}_{1,5}$ , we see that across the shock/rarefactions the tangential components of the velocity do not change since the components corresponding to  $(v, w)$  are null. For a detailed overview on the Euler equations we refer the reader to **Eleuterio F. Toro** (2009); **Godlewski and Raviart** (1996); **L. Quartapelle** (2015).

### 5.3.2 An exact Riemann solver for a generic Equation of State (EoS)

In section 5.2.2.3 we presented the generalities of the **RP**. Generally, the tooling around the retrieval of the exact solution of the Riemann Problem (RP) is presented and developed for **EoS** that allow closed analytical solutions to the integrals needed for the resolution of the rarefaction waves and that are easily made explicit, such that the Hugoniot locus can be expressed explicitly as a parametrization on the pressure  $p \rightarrow \rho$ :

$$\rho = \rho_{\text{hugoniot}}(p)$$

and the shock solution can be easily expressed analytically. In this section we present the numerical procedure to retrieve the exact solution of the **RP** for the Euler system with a reasonably **general EoS**  $e(p, \rho) \leftrightarrow p(\rho, e)$ , potentially tabulated or not complete. We will show how the solution requires two non-linear root-finding loops, together with coupled **ODEs** solutions in the case rarefaction waves are present. The procedure is inspired by [Colella and Glaz \(1985\)](#); [Kamm \(2015\)](#), but it features slight deviations to enhance comprehensibility. Before presenting the procedure, we briefly recap the equations that need to be solved to “connect” the left state with the right state across a wave. The type of solutions that we can have are: **rarefaction** or **shocks** for the genuinely non-linear characteristic fields associated to the eigenvalues  $\lambda_{1,3} = (u + c, u - c)$ , and a **contact discontinuity** associated to the linearly degenerate field  $\lambda_2 = u$ .

#### 5.3.2.1 Rarefaction wave

The rarefaction wave is a *simple wave* smooth solution of the **RP** that appears when the pressure after the wave is lower than the pressure before the wave, *i.e.*  $p^+ < p^-$ . Since a rarefaction wave is a simple wave, we know that (see definition 5) the Riemann invariants and the entropy  $s$  are constant across the wave. The thermodynamic entropy is defined via the Gibbs relation, *i.e.*

$$T ds = de + p d\left(\frac{1}{\rho}\right) \quad (5.72)$$

Therefore, if we decide to express our thermodynamic potentials as functions of, for example, one other potential and entropy, then across the rarefaction they are just function of the other chosen dependent variable. We choose the pressure  $p$  as the other functional dependency. For the Euler system, the two Riemann invariants are  $(u \pm \ell)$  based on if we are considering the

rarefaction wave corresponding respectively to  $\lambda_3$  or  $\lambda_1$ .  $\ell(p)$  is defined in differential form:

$$\frac{\partial \ell}{\partial p} = \frac{1}{\rho c} \quad (5.73)$$

To fully resolve the state across the rarefaction wave, we set the corresponding differential of the Riemann invariant to zero,

$$d(u \pm \ell) = 0$$

since it is constant, and we couple the definition of the speed of sound,

$$c^2 = \left. \frac{dp}{d\rho} \right| \quad (5.74)$$

Thus, we obtain the following characteristic ODEs:

$$\begin{cases} \frac{d\rho}{dp} = \frac{1}{c^2}, & \rho(p_k) = \rho_k \\ \frac{du}{dp} = \frac{\eta_k}{\rho c}, & u(p_k) = u^* \end{cases} \quad (5.75)$$

where the index  $k = (L, R)$  indicates the left or right state of the RP, and the function  $\eta_k$  is defined as follows:

$$\eta_k = \begin{cases} 1, & k = L \\ -1, & k = R \end{cases} \quad (5.76)$$

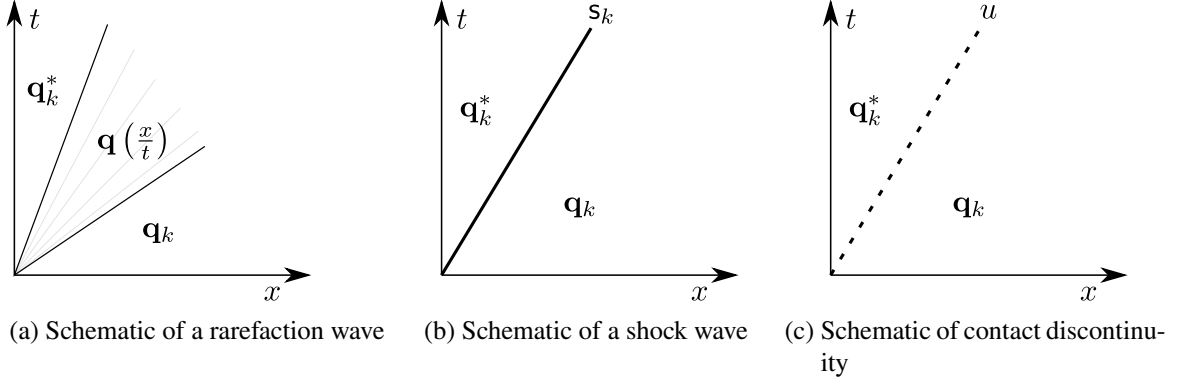
Equation (5.75) can also be written in compact form if we group the scalar fields in a state vector  $\hat{q} = (\rho, u)$  as

$$\frac{d\hat{q}}{dp} = \mathbf{f}_k(\rho, c), \quad \hat{q}(p_k) = \hat{q}_k$$

being  $\mathbf{f}_k = (1/c^2, \eta_k/(\rho c))^T$  and it needs to be integrated within the pressure interval  $p \in [p_k, p^*]$ . Please note that solving the ODE system of eq. (5.75) returns directly the value of the velocity in the star region that connects to the corresponding  $k$ -wave  $u_k^*$  (in addition to  $\rho_k^*$ ).

### 5.3.2.2 Shock wave

When the pressure after the wave  $p^+$  is higher than the one before  $p^-$ , entropy conditions tell us that the right wave typology is a shockwave. As we presented generally in section 5.2.2.2, when we have a discontinuity, we need to enforce Rankine-Hugoniot conditions. For the Euler system case, after some manipulations of the fundamental relations (eq. (5.31)), we can retrieve the Hugoniot locus, *i.e.* the set of states after the shock  $\mathbf{q}^+$  that can be linked to a given state



before the shock  $q^-$ ,

$$\mathcal{H}(p, \rho; p_k, \rho_k) : e(p, \rho) - e_k + \frac{1}{2}(p - p_k)(\tau - \tau_k) = 0 \quad (5.77)$$

taking into account that  $\tau = 1/\rho$  is the specific volume and that in this specific case of the **RP**, the state after shock are the “star-states”, and the states before the shock are the left and/or right state  $q_k$ . To be clear, eq. (5.77) represents the set of  $(p, \rho)$  that can be connected to a given left/right state  $(p_k, \tau_k = 1/\rho_k)$  by a shock. Please note that for general **EoS**, eq. (5.77) cannot be written as an explicit relation  $\rho = \rho_{\text{hugoniot}}(p)$  or viceversa  $p = p_{\text{hugoniot}}(\rho)$ , therefore it needs to be solved numerically employing a zero-finding routine, hence an initial guess for the  $\tau = 1/\rho$  is needed together with the classical initial guess for the pressure. Once we compute  $\rho_k^*$  for a given  $p^*$  from eq. (5.77), we can estimate the velocity after the shock  $u_k^*$  as:

$$u_k^* = u_k + (p^* - p_k) \sqrt{\frac{\tau_k - \tau_k^*}{p^* - p_k}} = u_k + f_k(p^*) \quad (5.78)$$

### 5.3.2.3 Contact discontinuity

For the linear degenerate characteristic fields (associated to the eigenvalue  $\lambda_2 = u$  in the 1D case), the wave typology is a **contact discontinuity**. Across this wave we can, again, enforce Rankine-Hugoniot conditions (eq. (5.31)), knowing that the speed of the discontinuity is  $s_2 = u$ . After simple manipulations, we retrieve that across a contact discontinuity pressure and velocity do not change.

$$\begin{cases} p^+ = p^- \\ u^+ = u^- \\ \rho^+ \neq \rho^- \end{cases}$$

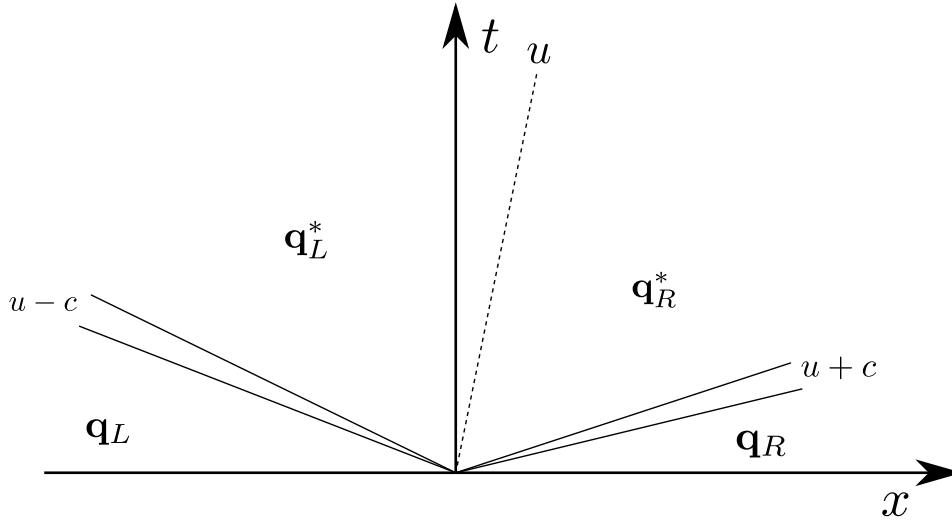


Figure 5.11: The Riemann Problem for the Euler system

and that means, for the complete **RP**, that

$$\begin{cases} p_L^* = p_R^* = p^* \\ u_L^* = u_R^* = u^* \end{cases} \quad (5.79)$$

#### 5.3.2.4 Complete Solution

Once we have all the different solutions archetypes for the different typologies of waves we can encounter in the **RP** for the Euler system, we will describe in this section the complete solution algorithm implemented in `josiepy`. The main idea is to predict the value of the velocity in the star region  $u^*$  from the left state  $u_L^*$  and from the right state  $u_R^*$ , and then find the value of  $p^*$  that minimize the error

$$u_L^* - u_R^* = 0$$

As we can understand from eqs. (5.75), (5.77) and (5.78), the expression of  $u_k^*$  are non-linear functions of  $p$ , and can be rewritten (as it is classically done) in the form:

$$u_k^* = u_k + f_k(p)$$

*i.e.*, the non-linear minimization of eq. (5.80) is rewritten as:

$$f_L(p) + f_R(p) + (u_R - u_L) = 0 \quad (5.80)$$

The complete algorithm is provided in algorithm 5.1.

---

**Algorithm 5.1** Algorithm to solve the Riemann Problem for the Euler System

---

```

1: procedure EXACTRP( $q_L, q_R$ )
2:    $p_L, u_L, \rho_L \leftarrow q_L$ 
3:    $p_R, u_R, \rho_R \leftarrow q_R$ 
4:    $\rho^* = 0.5(\rho_L + \rho_R)$   $\triangleright$  Guess a star value for the density to be used in case we have a
      shockwave

5:    $f_L(p) :$ 
6:   if  $p > p_L$  then  $\triangleright$  Assign the correct wave solution function to the left wave
7:      $\leftarrow \text{shock}(p, q_L; \rho^*)$ 
8:   else
9:      $\leftarrow \text{rarefaction}(p, q_L)$ 
10:  end if

11:   $f_R(p) :$ 
12:  if  $p > p_R$  then  $\triangleright$  Assign the correct wave solution function to the right wave
13:     $\leftarrow \text{shock}(p, q_R; \rho^*)$ 
14:  else
15:     $\leftarrow \text{rarefaction}(p, q_R)$ 
16:  end if
17:   $f(p) = f_L(p) + f_R(p) + (u_R - u_L)$ 

18:   $p^* \leftarrow \text{root}(f(p) = 0; 0 < p < 1e8)$   $\triangleright$  Non-linear root finding iterating on the pressure
      value in the star region
19:   $u^* \leftarrow 0.5(u_L + u_R) + f_R(p^*) - f_L(p^*)$ 
20: end procedure

```

---



### 5.3.2.5 Notes on the numerical schemes

The complete solution of the **RP** features potentially two root-finding loops: one inner loop in the eventuality of a shockwave (eq. (5.77)) to obtain the value  $\rho_k^*$  corresponding to the current value of the  $p^*$ , and the outer loop on  $p^*$ . While the Rankine-Hugoniot non-linear function (eq. (5.77)) is generally not problematic, and a Newton-Rhapson root-finding algorithm is stable and fast due to its convergence rate; the outer loop on  $p^*$  needs special handling, especially for left/right states that cause variation very close to the vacuum situation ( $\rho \sim 0$ ) (as for example Eleuterio F. Toro (2009, Test #2, p. 131). In that case, bracketing methods as a bisection method or the method suggested in Brent (2013, Chap. 4) are needed in order to avoid negative values for the pressure or the density.

The solution of the rarefaction (eq. (5.75)) requires the solution of a coupled **ODE**. No particular care has been necessary in the context of this work, an embedded **RK45** was used to produce the reference curves for the **RP** results in the next section 6.1.

### 5.3.3 Description of some useful schemes

While the exact **RP** solver we described in section 5.7.1.1 is a useful tool, especially for the very accurate resolution of test cases that can be casted as **RPs**, in industrial application it is preferred to use approximate Riemann solvers to approximate the flux across neighboring cells. We will now present a selection of approximated Riemann solvers schemes for the resolution of the Euler system that are based on a *a priori* hypothesis for the approximated wave structured and the associated wave speeds. The scheme are listed in order of complexity.

#### 5.3.3.1 The Rusanov scheme

The Rusanov scheme (Rusanov 1962) is an approximate Riemann solver that approximates the eigenstructure of the Euler system with two waves, with the same speed  $s$ , as shown in Fig. 5.12. The intercell flux between a cell  $i = L$  and its neighbor  $R$ , sharing a face of surface  $S_f$  is then computed as:

$$[F(q) \cdot \hat{n}]_f S_f = \frac{1}{2} [(F(q_L) + F(q_R))] \cdot \hat{n} + s(q_R - q_L) S_f \quad (5.81)$$

with the wave speed  $s$  estimated as follows ( $u_k \cdot \hat{n} = u_k$ )

$$s = \max (|u_L + c_L|, |u_R + c_R|) \quad (5.82)$$

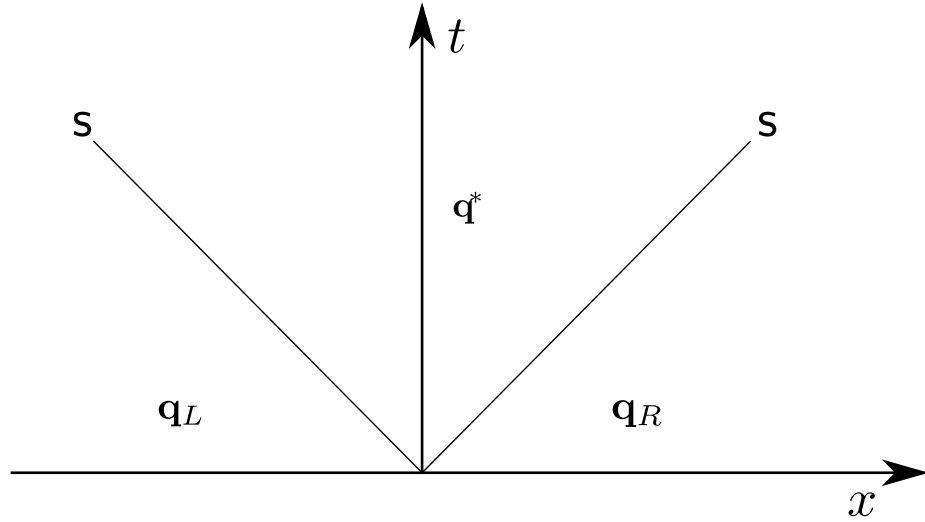


Figure 5.12: Rusanov scheme wave structure

### 5.3.3.2 The HLL scheme

The HLL scheme (Amiram Harten et al. 1983) is an approximate Riemann solver that approximates the eigenstructure of the Euler system with two waves, as for the Rusanov case, but with different speeds  $s_L, s_R$ , as shown in Fig. 5.13. The intercell flux between a cell  $i = L$  and its neighbor  $R$ , sharing a face of surface  $S_f$  is then computed as:

$$[F(q) \cdot \hat{n}]_f = \begin{cases} F(q_L) \cdot \hat{n} & \text{if } s_L \geq 0 \\ \frac{(s_R F(q_R) - s_L F(q_L)) \cdot \hat{n} + s_L s_R (q_R - q_L)}{s_R - s_L} & \text{if } s_L \leq 0 \leq s_R \\ F(q_R) \cdot \hat{n} & \text{if } s_R \leq 0 \end{cases} \quad (5.83)$$

with the wave speed  $s$  estimated as follows ( $u_k \cdot \hat{n} = u_k$ )

$$\begin{cases} s_L = \max(|u_L - c_L|, |u_R - c_R|) \\ s_R = \max(|u_L + c_L|, |u_R + c_R|) \end{cases} \quad (5.84)$$

### 5.3.3.3 The HLLC scheme

The HLLC scheme (Eleuterio F. Toro 2019) is an approximate Riemann solver that restores the contact discontinuity and shear waves (in multidimensional cases), hence featuring 3 waves: two discontinuities with speeds  $s_L, s_R$  and a contact/shear discontinuity of speed  $s^*$ , as shown

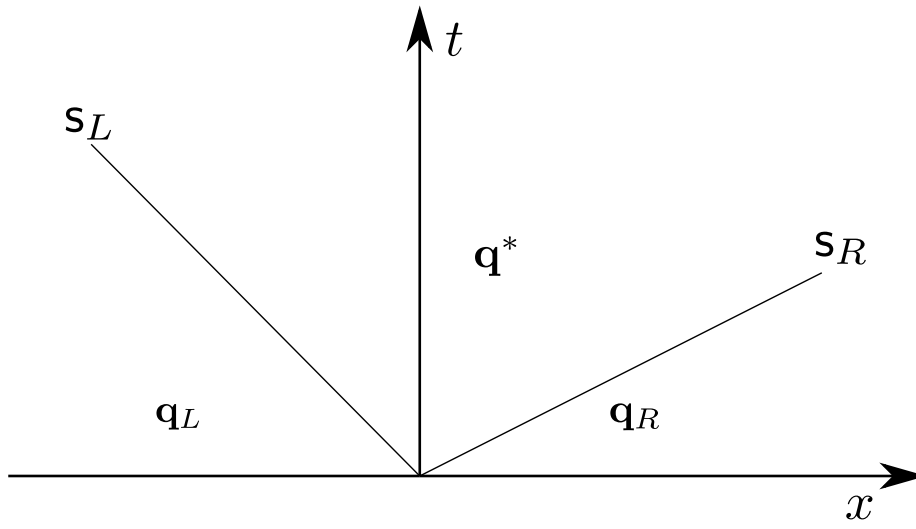


Figure 5.13: HLL scheme wave structure

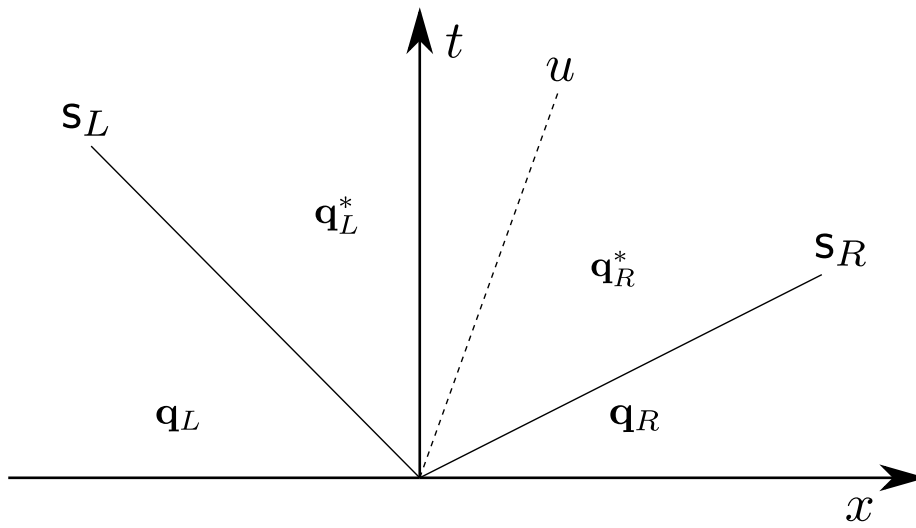


Figure 5.14: HLLC scheme wave structure

in Fig. 5.14. The intercell flux between a cell  $i = L$  and its neighbor  $R$ , sharing a face of surface  $S_f$  is then computed as:

$$[F(q) \cdot \hat{n}]_f = \begin{cases} F(q_L) \cdot \hat{n} & \text{if } s_L \geq 0 \\ F(q)_L^* \cdot \hat{n} & \text{if } s_L^* \leq 0 \leq s^* \\ F(q)_R^* \cdot \hat{n} & \text{if } s^* \leq 0 \leq s_R^* \\ F(q_R) \cdot \hat{n} & \text{if } s_R \leq 0 \end{cases} \quad (5.85)$$

with the star fluxes given by

$$F(q)_k^* \cdot \hat{n} = F(q_k) \cdot \hat{n} + s_k(q_k^* - q_k)$$

and the star states by

$$q_k^* = \rho_k \left( \frac{s_k - u_k}{s_k - s^*} \right) \begin{bmatrix} 1 \\ (Is^* - u_k) \cdot \hat{n} \\ \frac{E_k}{\rho_k} + (s^* - u_k) \left[ s^* + \frac{p_k}{\rho_k(s_k - u_k)} \right] \end{bmatrix}$$

with the wave speed  $s$  estimated as follows ( $u_k \cdot \hat{n} = u_k$ )

$$\begin{cases} s_L = \max(|u_L - c_L|, |u_R - c_R|) \\ s_R = \max(|u_L + c_L|, |u_R + c_R|) \\ s^* = \frac{p_R - p_L + \rho_L u_L (s_L - u_L) - \rho_R u_R (s_R - u_R)}{\rho_L (s_L - u_L) - \rho_R (s_R - u_R)} \end{cases} \quad (5.86)$$

The Euler system is an important building block in the context of hyperbolic conservation problems. The infrastructure provided by the library `josiepy` allowed the agile implementation of a set of different classical numerical schemes for the Euler systems and their second-order extensions with the MUSCL-Hancock method. These schemes have been a motivation also for the generation of a large test case suite that allows to benchmark the performances of the code and to catch as early as possible major implementation bugs. Moreover, a big part of the tools needed to implement schemes for the Euler system are reusable also for other systems like the one from [Baer and Nunziato \(1986\)](#). We will now describe the numerical aspects associated to the resolution of this important system for two-phase flows.

## 5.4 The Baer-Nunziato system

Accounting for velocity and pressure non-equilibrium between the two phases is a useful property a model should feature in order to account for the most complex configurations. A first attempt in that direction can be found in [Ransom and Hicks \(1982\)](#), while the work from [Baer and Nunziato \(1986\)](#) became a reference text for the subject. The Baer-Nunziato (B-N) system is a two-phase flow system that features complete disequilibrium in terms of velocity, pressures and energies (or temperatures). The model also features two quantities, the interfacial velocity  $\mathbf{u}^\delta$  and the interfacial pressure  $p^\delta$ , that needs closure. The work from [J. Glimm et al. \(1998\)](#); [Coquel, Gallouët, et al. \(2002\)](#) provide thorough analysis on that aspect. Additional work based on the Baer and Nunziato pioneering work can be found in [Seguin \(2002\)](#); [Daniel Lhuillier \(2003\)](#); [Saurel, S. Gavrilyuk, et al. \(2003\)](#); [Guillemaud \(2007\)](#); [D. Lhuillier et al. \(2013\)](#); [Y. Liu \(2013\)](#). For what concerns the numerical resolution of the model, notable references are [Saurel and Abgrall \(1999\)](#); [S. Gavrilyuk and Saurel \(2002\)](#); [Gallouët et al. \(2004\)](#); [Schwendeman et al. \(2006\)](#); [Deledicque and Papalexandris \(2007\)](#); [S. A. Tokareva and E. F. Toro \(2010\)](#); [Chalons et al. \(2011\)](#); [Coquel, Hérard, and Saleh \(2012\)](#); [Saleh \(2012\)](#); [Crouzet et al. \(2013\)](#); [Coquel, Hérard, Saleh, and Seguin \(2014\)](#); [Furfaro and Saurel \(2015\)](#); [Dallet \(2016\)](#); [Coquel, Hérard, and Saleh \(2016\)](#); [S. Tokareva and E. Toro \(2016\)](#); [Nguyen Tri Nguyen \(2018\)](#); [Zou et al. \(2019\)](#); [E. Toro et al. \(2020\)](#). In the context of this work, we implement the sourceless version of the system as it is done in [S. A. Tokareva and E. F. Toro \(2010\)](#). Recalling the reference numerical template equation,

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) + \mathbf{B}(\mathbf{q}) \cdot \nabla \mathbf{q} - \nabla \cdot (\mathbf{K}(\mathbf{q}) \cdot \nabla \mathbf{q}) + s(\mathbf{q}) = \mathbf{0}$$

with the state

$$\mathbf{q} = (\alpha, q_1, q_2) \quad (5.87)$$

The individual phasic states are defined as

$$\mathbf{q}_k = (\alpha_k \rho_k, \alpha_k \rho_k u_k, \alpha_k \rho_k v_k, \alpha_k \rho_k w_k, \alpha_k \rho_k E_k; \varphi_k, \quad k = 1, 2) \quad (5.88)$$

where the auxiliary fields are:

$$\varphi_k = (u_k, v_k, w_k, p_k, c_k) \quad (5.89)$$

We also define a modified non-conservative multiplier such that  $\mathbf{B}(\mathbf{q}) \cdot \nabla \mathbf{q} = \overline{\mathbf{B}}(\mathbf{q}) \cdot \nabla \alpha$ . The modified  $\overline{\mathbf{B}}$  pre-multiplies only the gradient of volume fraction instead of the entire state: this allows to have a smaller matrix and therefore a lower storage footprint in the software for this specific system. The convective flux is:

$$\mathbf{F}(\mathbf{q}) = \begin{bmatrix} 0 & 0 & 0 \\ \alpha_1 \rho_1 u_1 & \alpha_1 \rho_1 v_1 & \alpha_1 \rho_1 w_1 \\ \alpha_1 (\rho_1 u_1^2 + p) & \alpha_1 (\rho_1 u_1 v_1) & \alpha_1 (\rho_1 u_1 w_1) \\ \alpha_1 (\rho_1 v_1 u_1) & \alpha_1 (\rho_1 v_1^2 + p) & \alpha_1 (\rho_1 v_1 w_1) \\ \alpha_1 (\rho_1 u_1 w_1) & \alpha_1 (\rho_1 v_1 w_1) & \alpha_1 (\rho_1 w_1^2 + p) \\ \alpha_1 (\rho_1 E_1 + p) u_1 & \alpha_1 (\rho_1 E_1 + p) v_1 & \alpha_1 (\rho_1 E_1 + p) w_1 \\ \alpha_2 \rho_2 u_2 & \alpha_2 \rho_2 v_2 & \alpha_2 \rho_2 w_2 \\ \alpha_2 (\rho_2 u_2^2 + p) & \alpha_2 (\rho_2 u_2 v_2) & \alpha_2 (\rho_2 u_2 w_2) \\ \alpha_2 (\rho_2 v_2 u_2) & \alpha_2 (\rho_2 v_2^2 + p) & \alpha_2 (\rho_2 v_2 w_2) \\ \alpha_2 (\rho_2 u_2 w_2) & \alpha_2 (\rho_2 v_2 w_2) & \alpha_2 (\rho_2 w_2^2 + p) \\ \alpha_2 (\rho_2 E_2 + p) u_2 & \alpha_2 (\rho_2 E_2 + p) v_2 & \alpha_2 (\rho_2 E_2 + p) w_2 \end{bmatrix} \quad (5.90)$$

and the non-conservative multiplier,

$$\overline{\mathbf{B}}_{ad}(\mathbf{q}) = \begin{bmatrix} u^\mathcal{S} & v^\mathcal{S} & w^\mathcal{S} \\ 0 & 0 & 0 \\ -p^\mathcal{S} & 0 & 0 \\ 0 & -p^\mathcal{S} & 0 \\ 0 & 0 & -p^\mathcal{S} \\ -p^\mathcal{S} u^\mathcal{S} & -p^\mathcal{S} v^\mathcal{S} & -p^\mathcal{S} w^\mathcal{S} \\ 0 & 0 & 0 \\ p^\mathcal{S} & 0 & 0 \\ 0 & p^\mathcal{S} & 0 \\ 0 & 0 & p^\mathcal{S} \\ p^\mathcal{S} u^\mathcal{S} & p^\mathcal{S} v^\mathcal{S} & p^\mathcal{S} w^\mathcal{S} \end{bmatrix} \quad (5.91)$$

$p^\mathcal{S}$  and  $\mathbf{u}^\mathcal{S}$  are the interfacial pressure and velocity and need to be modeled. We assume the “classical” choice, as shown in [Baer and Nunziato \(1986\)](#), that yields:

$$p^\mathcal{S} = p_2, \quad \mathbf{u}^\mathcal{S} = \mathbf{u}_1 \quad (5.92)$$

### 5.4.1 Eigenstructure

As shown in S. A. Tokareva and E. F. Toro (2010), the eigenvalues of the system are:

$$\begin{aligned}\lambda_1 &= u_1 - c_1, & \lambda_{2-4} &= u_1, & \lambda_5 &= u_1 + c_1, \\ \lambda_6 &= u_2 - c_2, & \lambda_{7-9} &= u_2, & \lambda_{10} &= u_2 + c_2, \\ \lambda_{11} &= u^\delta\end{aligned}\tag{5.93}$$

### 5.4.2 The Rusanov scheme

In our tests, despite being very dissipative, we implement a Rusanov scheme that is analogous to the one discussed for the Euler scheme in section 5.3.3.1. The wave speed is computed as follows:

$$s = \max(s_1, s_2)\tag{5.94}$$

where the phasic wave speeds are computed as in the Euler case, *i.e.*

$$s_k = \max(|u_k^L + c_k^L|, |u_k^R + c_k^R|)\tag{5.95}$$

## 5.5

## A two-phase two-scale model featuring geometrical fields

---

A part of the work described in the thesis by Cordesse (2020) delves with the development of a two-phase model that includes geometrical properties like the mean and Gauss curvatures together with a two-scale description, in which small scales and larger scales are explicitly taken into account together with capillarity effects in the spirit of Brackbill et al. (1992). The process is based on a variational derivation, on the lines of what we described in section 3.4, in which a Hamiltonian action is made stationary under specific constraints in order to obtain a system of governing equations. The work of Cordesse (2020), moreover, proposes a hierarchy of models derived from the general one applying a sequence of simplifications such as the instantaneous relaxation of the phasic pressures, asymptotic limit of micro-inertia and neglecting small scales

governing equations. The homogeneous contribution of the resulting model is the following:

$$\left\{ \begin{array}{l} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \\ \frac{\partial Y_k}{\partial t} + \mathbf{u} \cdot \nabla Y_k \\ \rho \frac{\partial \mathbf{u}}{\partial t} + \rho \mathbf{u} \cdot \nabla \mathbf{u} + \nabla P + \mathbf{\Omega} \cdot \nabla \alpha \\ \frac{\partial s_k}{\partial t} + \mathbf{u} \cdot \nabla s_k \\ \frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega \\ \frac{\partial H}{\partial t} + \mathbf{u} \cdot \nabla H \\ \frac{\partial \tilde{\Sigma}}{\partial t} + \mathbf{u} \cdot \nabla \tilde{\Sigma} \\ \frac{\partial \alpha}{\partial t} + \mathbf{u} \cdot \nabla \alpha \\ \frac{\partial \nabla \alpha}{\partial t} + \mathbf{u} \cdot \nabla (\nabla \alpha) + \nabla \alpha \cdot \nabla \mathbf{u} \end{array} \right. = 0 \quad \begin{array}{l} (5.96a) \\ (5.96b) \\ (5.96c) \\ (5.96d) \\ (5.96e) \\ (5.96f) \\ (5.96g) \\ (5.96h) \\ (5.96i) \end{array}$$

where  $\mathbf{\Omega} = \sigma \|\nabla \alpha\| \mathbf{I} - \sigma \nabla \alpha \otimes \nabla \alpha / \|\nabla \alpha\|$  is a term associated to large scale capillarity effects,  $s_k$  are the phasic entropies,  $\omega$  is a subscale pulsation,  $Y_k$  are the mass fractions,  $\rho$  is the density and  $P \triangleq p - \tilde{\sigma} \tilde{\Sigma}$ ,  $p = \sum_k \alpha_k p_k$  a reduced pressure, being  $\tilde{\sigma}$  a *fluctuating surface tension coefficient* (see Cordesse (2020) for more details on how the macro scale is filtered out of the fluctuating scale). Let us note that in eq. (5.96), eq. (5.96i) is redundant with eq. (5.96h). Nevertheless, eq. (5.96i) allows to involve the capillarity terms within the wave structure of the system. The numerical schemes to solve system 5.96 have been implemented in the CEDRE code, a FVM-based proprietary code developed at ONERA. We decided to reproduce the implementation and the test cases in order to provide an open implementation of the model within an easy-to-use framework, such as `josiepy`, capable of helping with the general battle testing of the model and the associated numerical strategies. The original work from Cordesse proposes a splitting strategy in which the hydraulic flux (“the Euler subsystem”), the capillarity flux and the instantaneous relaxation of pressures are tackled separately, while in the case of this study, we solve the coupled system (neglecting capillarity effects).

### 5.5.1 Eigenstructure

In order to reproduce part of the solution proposed in Cordesse (2020), we simplify the system 5.96 assuming  $\tilde{\sigma}, \sigma = 0 \Rightarrow \mathbf{\Omega} = \mathbf{0}$ . Since capillarity effects are neglected ( $\mathbf{\Omega} = \mathbf{0}$ ),



the equation on the gradient of the volume fraction is omitted. If we consider the 1D x-split projection of eq. (5.96) and the change of variables:

$$\mathbf{w} = \left( \rho, u, v, w, Y, s_1, s_2, \omega, \mathbf{H}, \tilde{\Sigma}, \alpha \right) \quad (5.97)$$

we obtain the corresponding quasi-linear form of the system:

$$\frac{\partial \mathbf{w}}{\partial t} + \mathbf{A}(\mathbf{w}) \cdot \nabla \mathbf{w} = 0 \quad (5.98)$$

The  $11 \times 11$  matrix  $\mathbf{A}(\mathbf{w})$  is:

$$\mathbf{A}(\mathbf{w}) = \frac{1}{\rho} \begin{bmatrix} \rho u & \rho^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\partial P}{\partial \rho} & \rho u & 0 & 0 & \frac{\partial P}{\partial Y} & \frac{\partial P}{\partial s_1} & \frac{\partial P}{\partial s_2} & 0 & 0 & 0 & \frac{\partial P}{\partial \alpha} \\ 0 & 0 & \rho u & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \rho u & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \rho u & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \rho u & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho u & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho u & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho u & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho u & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho u \end{bmatrix} \quad (5.99)$$

with the associated eigenvalues:

$$\lambda_{1-9} = u, \quad \lambda_{10,11} = u \pm \sqrt{\frac{\partial P}{\partial \rho}} = u \pm c_F \quad (5.100)$$

where  $c_F$  is the frozen sound speed

$$c_F^2 = \sum_k^2 Y_k c_k^2, \quad c_k^2 = \frac{\partial p_k}{\partial \rho_k} \quad (5.101)$$

### 5.5.2 Recasting of the energy conservation for numerical implementation purposes

In Cordesse (2020), in order to improve the numerical results, it is mentioned that the phasic entropy equations eq. (5.96d) fail at “preserving a uniform pressure and velocity flow”. Therefore, they propose to replace the entropy equations with the internal energy equations, via the Gibbs relation of thermodynamics which provides:

$$\frac{\partial \alpha_k \rho_k e_k}{\partial t} + \nabla \cdot (\alpha_k \rho_k e_k \mathbf{u}) - \alpha_k p_k \nabla \cdot \mathbf{u} = -p_k D_t \alpha_k \quad (5.102)$$

Nonetheless eq. (5.102) contains non-conservative terms that destroy the strict conservation of the total energy due to the loss of Rankine-Hugoniot jump conditions. Therefore Cordesse proposes the addition of a redundant total energy equation, in the same spirit as Saurel, Petitpas, et al. (2009), *i.e.*

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot \left[ \left( \rho E + p - \sigma \|\nabla \alpha\| - \tilde{\sigma} \tilde{\Sigma} + \sigma \frac{\nabla \alpha \otimes \nabla \alpha}{\|\nabla \alpha\|} \right) \mathbf{u} \right] = 0 \quad (5.103)$$

### 5.5.3 Instantaneous relaxation of the phasic pressures

The pressure relaxation when neglecting capillarity effects,

$$p_1(\rho_1, s_1) = p_2(\rho_2, s_2) \quad (5.104)$$

can be recasted, as per Saurel, Petitpas, et al. (2009), in terms of a non-linear volume fraction algebraic equation that is imposed instantaneously via a non-linear Newton-Rhapson solver after the time step update,

$$\alpha_1^{n+1} = \alpha_1^n - \frac{p_1^n - p_2^n}{\left(\frac{dp_1}{d\alpha_1}\right)^n + \left(\frac{dp_2}{d\alpha_2}\right)^n} \quad (5.105)$$

After the relaxation, we obtain new values for the  $\alpha'$ ,  $\Sigma'$ ,  $\rho'_k$ . As described in section 5.5.2, the total energy needs to be corrected because of the non-conservative terms that appears in the phasic internal energies equations  $\rho_k \alpha_k e_k$ . The fundamentals of the procedure lie in the recalculation of an equilibrium pressure from the equation:

$$\left( \rho E - \frac{1}{2} \rho \mathbf{u} \cdot \mathbf{u} \right) = \sum_k^2 (\alpha_k \rho_k)^0 e_k(p, \rho'_k) \quad (5.106)$$

where the superscript  $\bullet^0$  indicates fields before the relaxation procedure. The energy correction returns a new value of the equilibrium pressure  $p'$  that is then used to recompute  $e_k(p', \rho'_k)$ .

### 5.5.4 The final system

The final system to be solved, neglecting capillarity effects, is:

$$\left\{ \begin{array}{ll} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) & = 0 \quad (5.107a) \\ \frac{\partial Y_k}{\partial t} + \mathbf{u} \cdot \nabla Y_k & = 0 \quad (5.107b) \\ \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot [(\rho \mathbf{u} \otimes \mathbf{u} + p \mathbf{I})] & = 0 \quad (5.107c) \\ \frac{\partial \alpha_k \rho_k e_k}{\partial t} + \nabla \cdot (\alpha_k \rho_k e_k \mathbf{u}) - \alpha_k p_k \nabla \cdot \mathbf{u} & = 0 \quad (5.107d) \\ \frac{\partial \rho E}{\partial t} + \nabla \cdot [(\rho E + p) \mathbf{u}] & = 0 \quad (5.107e) \\ \frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega & = 0 \quad (5.107f) \\ \frac{\partial H}{\partial t} + \mathbf{u} \cdot \nabla H & = 0 \quad (5.107g) \\ \frac{\partial \tilde{\Sigma}}{\partial t} + \mathbf{u} \cdot \nabla \tilde{\Sigma} & = 0 \quad (5.107h) \\ \frac{\partial \alpha_k}{\partial t} + \nabla \cdot (\alpha_k \mathbf{u}) - \alpha_k \nabla \cdot \mathbf{u} & = 0 \quad (5.107i) \end{array} \right.$$

to which we couple the instantaneous relaxation procedure we describe in section 5.5.3. Please note that the equation on the volume fraction eq. (5.107i) is recasted in order to expose a non-conservative contribution in terms of  $\nabla \cdot \mathbf{u}$ . Additionally, the system 5.107 as presented in the original work of Cordesse (2020), features transport equations for all the mass fractions (eq. (5.107b)) that are redundant with the global mass conservation equation (eq. (5.107a)). Similarly for the volume fractions (eq. (5.107i)). In our implementation, we respected these implementation choices to produce a result as closest as possible to the original one.

Recalling the reference numerical template equation

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) + \mathbf{B}(\mathbf{q}) \cdot \nabla \mathbf{q} - \nabla \cdot (\mathbf{K}(\mathbf{q}) \cdot \nabla \mathbf{q}) + s(\mathbf{q}) = \mathbf{0}$$

with the state

$$\mathbf{q} = (\rho, \rho u, \rho v, \rho w, \rho E, \omega, \rho \Sigma, \rho H, \rho Y_1, \alpha_1, \rho_1 \alpha_1 e_1, \rho Y_2, \alpha_2, \rho_2 \alpha_2 e_2; \varphi_k) \quad (5.108)$$

where the auxiliary fields are:

$$\varphi_k = (u, v, w, c_F, p_1, c_1, p_2, c_2) \quad (5.109)$$

We also define a modified non-conservative multiplier such that  $\mathbf{B}(\mathbf{q}) \cdot \nabla \mathbf{q} = \overline{\mathbf{B}}(\mathbf{q}) \cdot \nabla \mathbf{u}$ . The modified  $\overline{\mathbf{B}}$  pre-multiplies only the gradient tensor of the fluid velocity instead of the entire state: this allows to have a smaller matrix and therefore a lower storage footprint in the software for this specific system. The convective flux is:

$$\mathbf{F}(\mathbf{q}) = \begin{bmatrix} \rho u & \rho v & \rho w \\ \rho u^2 + p & \rho uv & \rho uw \\ \rho vu & \rho v^2 + p & \rho vw \\ \rho wu & \rho wv & \rho w^2 + p \\ (\rho E + p)u & (\rho E + p)v & (\rho E + p)w \\ \omega u & \omega v & \omega w \\ \rho H u & \rho H v & \rho H w \\ \rho \Sigma u & \rho \Sigma v & \rho \Sigma w \\ \rho Y_1 u & \rho Y_1 v & \rho Y_1 w \\ \alpha_1 u & \alpha_1 v & \alpha_1 w \\ \rho_1 \alpha_1 e_1 u & \rho_1 \alpha_1 e_1 v & \rho_1 \alpha_1 e_1 w \\ \rho Y_2 u & \rho Y_2 v & \rho Y_2 w \\ \alpha_2 u & \alpha_2 v & \alpha_2 w \\ \rho_2 \alpha_2 e_2 u & \rho_2 \alpha_2 e_2 v & \rho_2 \alpha_2 e_2 w \end{bmatrix} \quad (5.110)$$

and the non-conservative multiplier,

$$\begin{aligned}
 \overline{\mathbf{B}}_{ab0}(\mathbf{q}) &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\omega & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\alpha_1 & 0 & 0 \\ -\alpha_1 p_1 & 0 & 0 \\ 0 & 0 & 0 \\ -\alpha_2 & 0 & 0 \\ -\alpha_2 p_2 & 0 & 0 \end{bmatrix} & \overline{\mathbf{B}}_{ab1}(\mathbf{q}) &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -\omega & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -\alpha_1 & 0 \\ 0 & -\alpha_1 p_1 & 0 \\ 0 & 0 & 0 \\ 0 & -\alpha_2 & 0 \\ 0 & -\alpha_2 p_2 & 0 \end{bmatrix} & \overline{\mathbf{B}}_{ab2}(\mathbf{q}) &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\omega \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\alpha_1 \\ 0 & 0 & -\alpha_1 p_1 \\ 0 & 0 & 0 \\ 0 & 0 & -\alpha_2 \\ 0 & 0 & -\alpha_2 p_2 \end{bmatrix} \quad (5.111)
 \end{aligned}$$

### 5.5.5 HLLC with pressure relaxation and energy correction

In addition to the pressure relaxation algorithm discussed previously, the convective flux is approximated using an HLLC scheme analogous to the one described in section 5.3.3.3 for the Euler system.

## 5.6

## The system featuring a governing equation for the interfacial area density

We already discussed in a fairly detailed manner the properties of the systems in section 3.6. Here we will limit ourselves to express the terms in the framework of the generalized equation we introduced in this chapter. The generalized template of the equations we can support in `josi` is:

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) + \mathbf{B}(\mathbf{q}) \cdot \nabla \mathbf{q} - \nabla \cdot (\mathbf{K}(\mathbf{q}) \cdot \nabla \mathbf{q}) + s(\mathbf{q}) = \mathbf{0}$$

The convective flux is:

$$F(q) = \begin{bmatrix} \rho u & \rho v & \rho w \\ (\rho u^2 + p) & (\rho uv) & (\rho uw) \\ (\rho vu) & (\rho v^2 + p) & (\rho vw) \\ (\rho uw) & (\rho vw) & (\rho w^2 + p) \\ \rho Yu & \rho Yv & \rho Yw \\ \rho Y\omega u & \rho Y\omega v & \rho Y\omega w \\ \rho \alpha u & \rho \alpha v & \rho \alpha w \\ \rho zu & \rho zv & \rho zw \end{bmatrix} \quad (5.112)$$

and the source terms vector respectively is,

$$s(q) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{p_2 - p_1}{\sqrt{m}} \\ -\frac{\rho Y \omega}{\sqrt{m}} \\ 0 \end{bmatrix} \quad (5.113)$$

### 5.6.1 The Rusanov scheme

In order to retain a certain stability of the system, since the phasic densities on which the phasic pressure depends  $p_k = \rho_k$  can lead to stiffness in the source terms, we choose as numerical scheme the Rusanov scheme discussed in section 5.3.3.1. The computation of the intercell flux is analogous to the Euler case, taking into consideration that the effective sound velocity is:

$$c_z^2 = \sum_k^2 Y_k c_k^2 + \rho (Y\omega)^2 - \frac{2}{9} \sigma \left( \frac{z}{\rho} \right)^{\frac{2}{3}}$$

## 5.7 A set of verification test cases

In this section we present a set of verification cases obtained with `josiepy`, a Python library aimed at fast prototyping of problems governed by PDEs. In section 6.1 we discuss the phi-

losophy behind the development of the software, together with the structure of the code. Here, a set of tests have been performed on different systems, notably the Euler System, a NS pipe flow configuration, the Baer-Nunziato system and the system proposed in Cordesse (2020) for normal pulsations (*i.e.* a formal evolution of the system described in section 3.5.2) and we highlight the level of accuracy we can achieve with the code.

### 5.7.1 A series of Riemann Problems (RPs) for the Euler system

The Euler system is an ideal candidate to benchmark the correct implementation of schemes, data structures, boundary conditions implementation and all the machinery underlying a solver code like `josiepy`. That is why in this section we present the implementation of a Rusanov, HLL and HLLC schemes for the Euler system as briefly explained in section 5.3. Hence, the set of tests proposed by Eleuterio F. Toro (2009) are used.

#### 5.7.1.1 Euler exact Riemann Problem solver

Before delving in the discussion of the results, a note on the exact curves against which we compare our results: all the reference data shown in Figures 5.16 to 5.20 are computed using the exact Riemann solver for the Euler system we presented in section 5.3.2. In particular, the case #2 (Fig. 5.17) features a situation where the density can reach 0 or even negative values if the non-linear iterations in the Riemann solver are not bounded. In section 5.3.2 we discussed the usage in one of the non-linear loop of a bounded root-finding algorithm to ensure robustness of the resolution algorithm and we correctly obtain the theoretical curve with the general-EoS RP solver. The exact solver is also available in the `josiepy` library.

#### 5.7.1.2 Benchmark tests from Eleuterio F. Toro

Table 5.1: Left and right initial data for the Euler test cases

Test	$\rho_L$	$u_L$	$p_L$	$\rho_R$	$u_R$	$p_R$	CFL
1	1.0	0.0	1.0	0.125	0.0	0.1	0.5
2	1.0	-2.0	0.4	1.0	2.0	0.4	0.45
3	1.0	0.0	1000	1.0	0.0	0.01	0.5
4	1.0	0.0	0.01	1.0	0.0	100	0.5
5	5.999 24	19.5975	460.894	5.992 42	-6.196 33	46.0950	0.5

The reference curves for testing are taken from the tests described in Eleuterio F. Toro (2009, Section 4.3.3); those tests are RP configurations with a selection of different initial states. All

the results are obtained assuming a specific heat ratio  $\gamma = 1.4$ , on a 1D mesh of 500 cells. The left and right [Boundary Conditions](#) are mostly irrelevant, since the simulation is stopped before then the state could interact with the boundaries. Effectively, both Neumann and Dirichlet conditions have been imposed with no tangible effect on the robustness of the resolution or execution speed. More in details,

- Test 1 is a [Sod \(1978\)](#) test problem. The solution of this problem consists of a left rarefaction, a contact and a right shock. The resulting curves are plotted at the time instant  $t = 0.25s$ . The results are shown in [Fig. 5.16](#).
- Test 2 has a solution consisting of two strong rarefactions and a stationary contact discontinuity. This test is probably among the most challenging ones because the density reaches values close to zero and this can lead to problems in the non-linear root finders. Results in [Fig. 5.17](#) at time  $t = 0.15s$ .
- Test 3 is a severe problem whose solution is made of a left rarefaction, a contact and a right shock. It is the left half of the blast wave problem of [Woodward and Colella \(1984\)](#). The results are shown in [Fig. 5.18](#) at time  $t = 0.012s$ .
- Test 4 is the corresponding right half of the blast wave problem from [Woodward and Colella \(1984\)](#). The solution contains a left shock, a contact discontinuity and a right rarefaction as shown in [Fig. 5.19](#). The figures are shown for time  $t = 0.035s$ .
- The final test 5 is made up of the right and left shocks emerging from the solution to test 3 and 4. The solution features a left facing shock, a right traveling contact discontinuity and a right traveling shock wave. The simulation data are shown in [Fig. 5.20](#) at time  $t = 0.035s$ .

All the tests configurations are summarized in [table 5.1](#) and the mesh size is 500 cells for all the trials. We will discuss the results of the tests globally, since the general performance of the schemes is generally repeatable on each configuration. The three different first-order schemes tested, *i.e.* the Rusanov, HLL and HLLC, perform substantially very similarly. The accordance with the exact solutions of the [RP](#) is generally good considering the low order of the schemes, featuring a noticeable amount of numerical diffusion specially detected in high-gradient zones. As an example, the sudden increase of the density field in [Figures 5.18c, 5.19c and 5.20c](#) is quite diffused having the peaks smeared out by the numerical diffusion. Inspecting [Fig. 5.20c](#) for instance, we can appreciate how the HLLC scheme is less diffusive than the other two since the peak is closer to the analytical one. This behavior can be verified in all the other figures.



The implementation in `josi` allows to retrieve accurate results of some classical test cases for the Euler system that validate the underlying implementation and opens up interesting possibilities for the quick testing of numerical strategies for the Euler system. Notably, the results for the verification tests we list in this section are part of the testing suite of the library that is executed at each code commit, allowing an agile advancement in the numerical computing for the classical Euler system.

### 5.7.1.3 Higher order

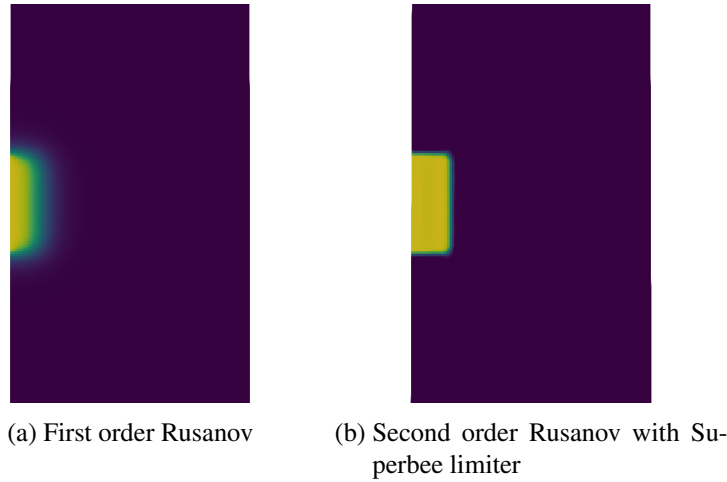
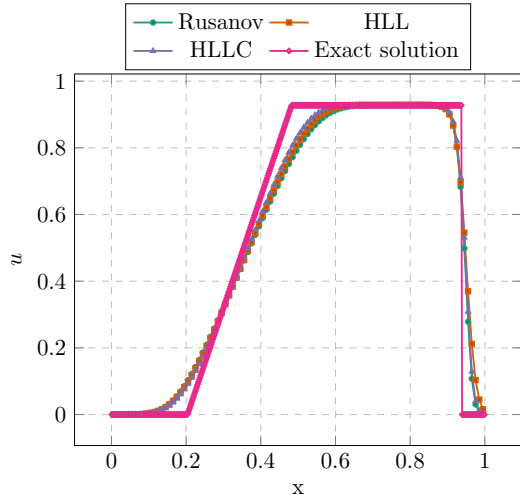


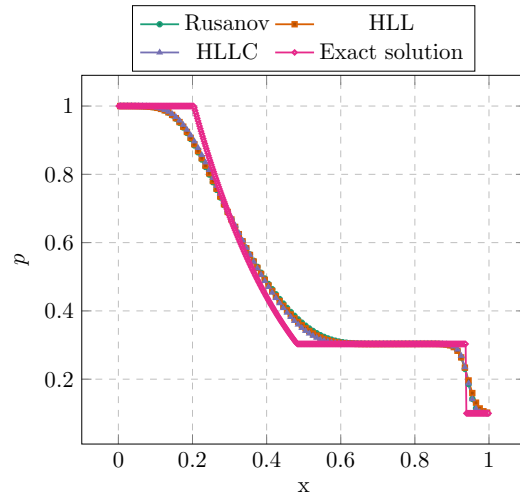
Figure 5.15: Density field comparison of first and second order Rusanov schemes for an Euler jet at time  $t = 0.03s$

`josi` also includes higher-order extensions via the MUSCL-Hancock methodology. In the context of this work, we focused especially on the flexibility of the library to allow the fast introduction of several different type of solvers and the associated underlying phenomena. Nonetheless, the PhD of [Loison \(2023\)](#) and the post-doc of [Ait Ameer \(2020\)](#) are ongoing efforts to implement high-order methods for the simulation of two-phase flows, especially Discontinuous Galerkin schemes. Without the presumption of being complete, in this section we just showcase a simple test case of a 2D “round” jet simulated using the Euler system. The configuration of the case is the following: a box of dimensions  $0.25 \times 1m$  is meshed with  $100 \times 400$  cells. The internal field is at rest  $u_0 = 0, v_0 = 0, \rho_0 = 1, p_0 = 1000$ . The left boundary contains the inlet at the position  $y = 0.5m$ , of radius  $0.05m$ , which is configured as a Dirichlet BC with state  $u_{jet} = 1, v_{jet} = 0, \rho_{jet} = 1000\rho_0, p_{jet} = 1000$ . The rest of the left boundary is set with a zero-gradient Neumann BC. The top and bottom walls have a fixed value BC with the same state as the internal field at  $t = 0$ . The right wall instead has a complete zero-gradient

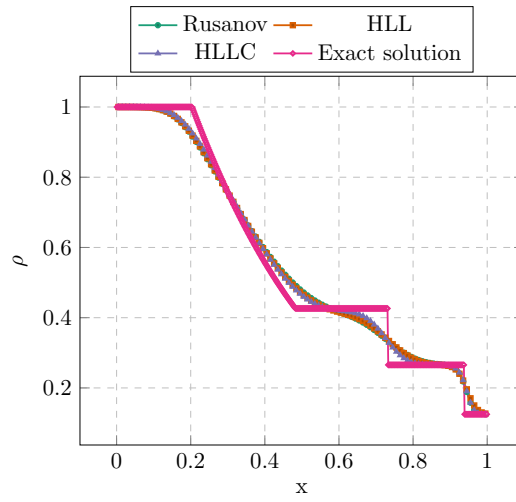
Neumann BC. The scheme we show here is the Rusanov scheme and its extension to second order with the MUSCL-Hancock strategy, with a SUPERBEE limiter (LeVeque 1990; LeVeque 2002; Eleuterio F. Toro 2009; Bouchut 2004). josiemy nonetheless contains second order extensions for HLL and HLLC with more limiters to choose from. Fig. 5.15 shows a comparison of the  $\rho$ -field between the first order, very diffusive, Rusanov scheme and its corresponding second order limited version, where the chosen limiter is the SUPERBEE (Eleuterio F. Toro 2009).



(a) Velocity profile

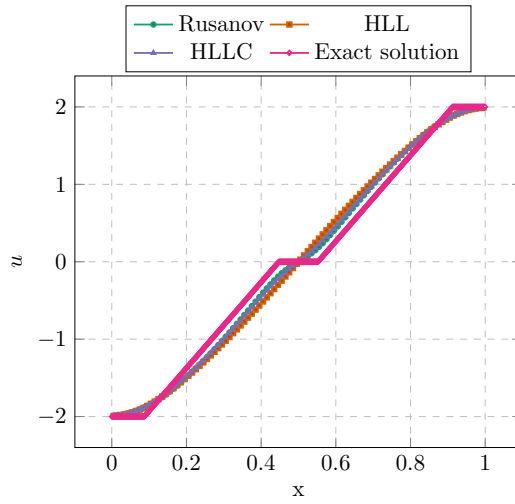


(b) Pressure profile

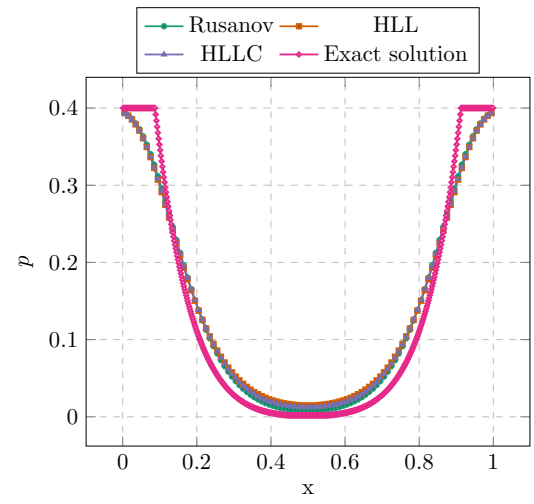


(c) Density profile

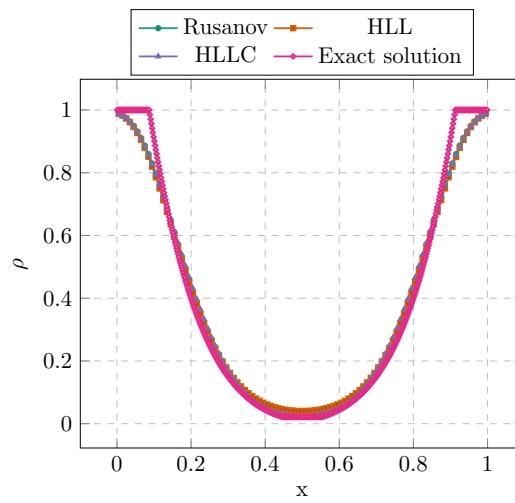
Figure 5.16: Test 1 at  $t = 0.25s$  of Eleuterio F. Toro (2009)



(a) Velocity profile

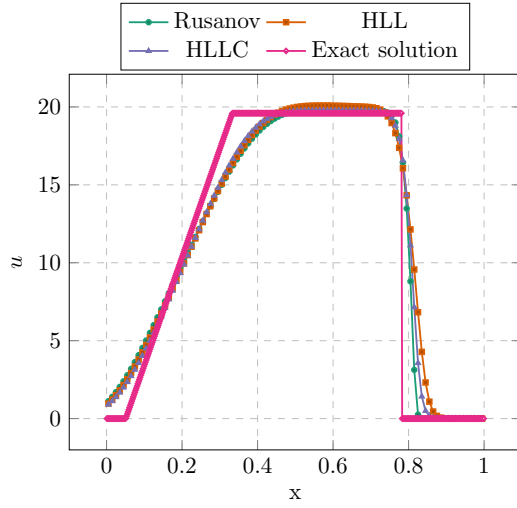


(b) Pressure profile

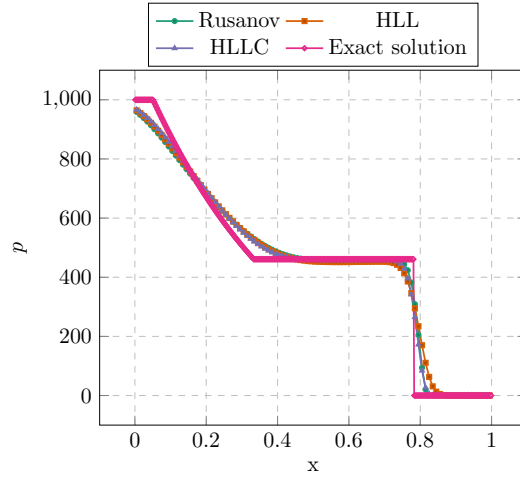


(c) Density profile

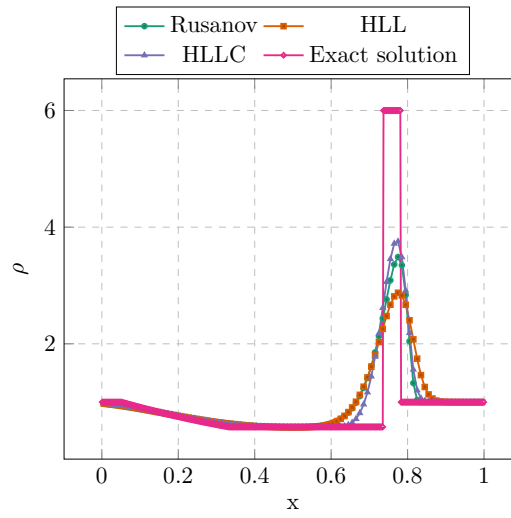
Figure 5.17: Test 2 at  $t = 0.15s$  of Eleuterio F. Toro (2009)



(a) Velocity profile

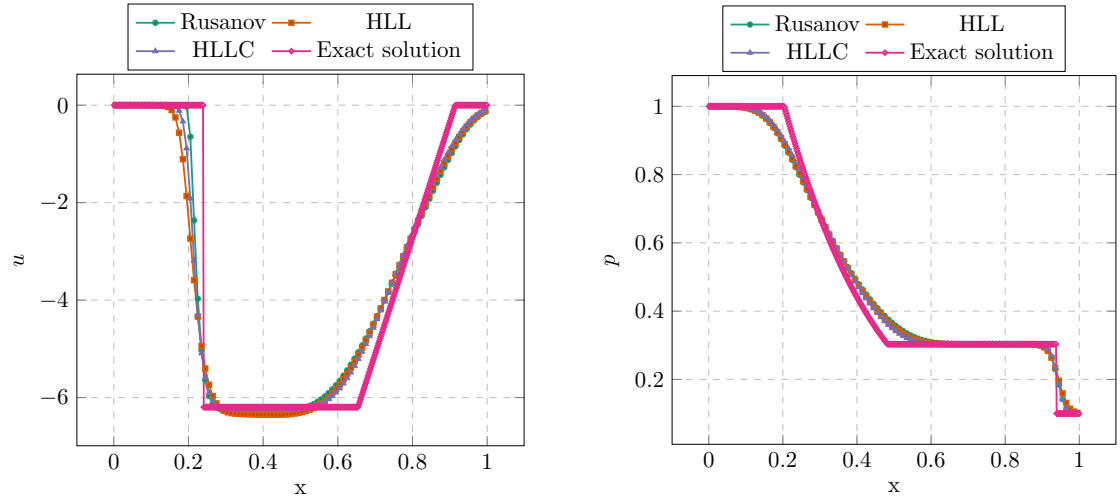


(b) Pressure profile



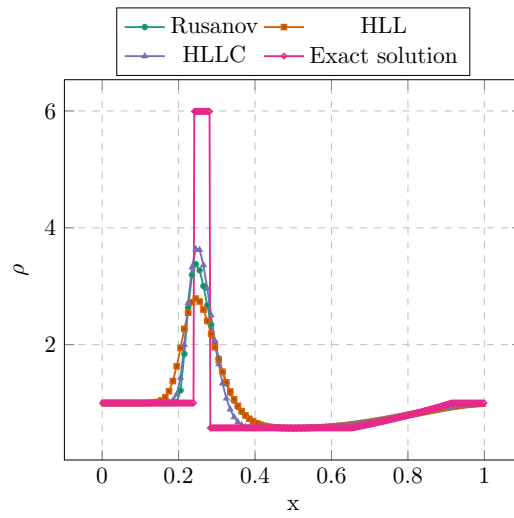
(c) Density profile

Figure 5.18: Test 3 at  $t = 0.012s$  of Eleuterio F. Toro (2009)



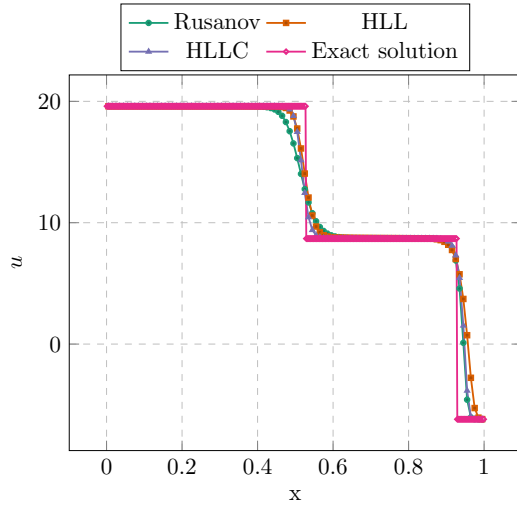
(a) Velocity profile

(b) Pressure profile

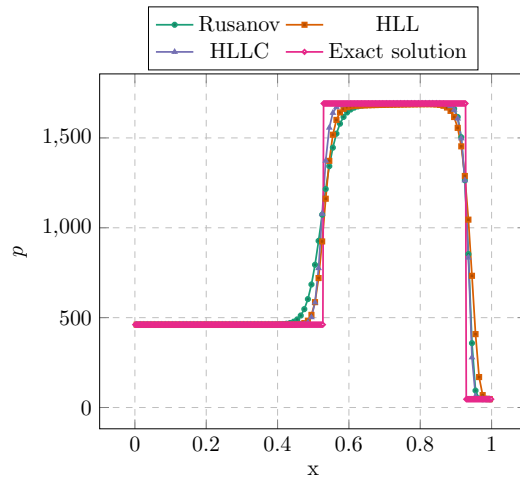


(c) Density profile

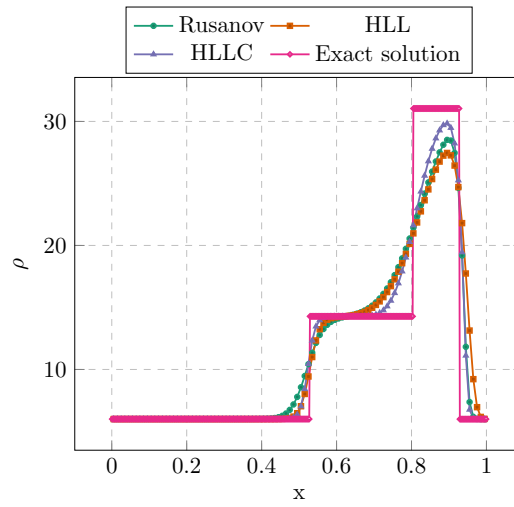
Figure 5.19: Test 4 at  $t = 0.035s$  of Eleuterio F. Toro (2009)



(a) Velocity profile



(b) Pressure profile



(c) Density profile

Figure 5.20: Test 5 at  $t = 0.035s$  of Eleuterio F. Toro (2009)

### 5.7.2 A series of Riemann Problems for the Baer-Nunziato (B-N) system

The B-N model (Baer and Nunziato 1986) is a classical model in the two-phase flow community that features phase disequilibrium in all aspects, that is in terms of velocity, pressures and temperatures. It is used in different contexts that could lie even far from the original purpose of simulating two-phase flow explosives for which it was originally conceived. For this reason, `josi` contains an implementation of this model with associated numerical schemes. In order to assess the correctness of the implementation, we decided to benchmark the code against the tests that are suggested in S. A. Tokareva and E. F. Toro (2010). We also add two additional “artificial” tests with the aim of reproducing the RP results for the Euler system (see section 5.7.1), or a simple advection on the volume fraction  $\alpha$  field, but with the complete implementation of the B-N solver. All the tests are solved on a 1D mesh of 100 points, the simulation is stopped before the solution could interact with the boundaries hence the choice of the boundary conditions has no effect (for the sake of completeness, Dirichlet BCs are imposed with fixed value equal to the left/right state of the RP). The scheme used is a Rusanov scheme for all the results shown here and an upwinding scheme is used for the non-conservative term.

#### 5.7.2.1 Test 1: retrieving the Euler system results

Table 5.2: Left and right initial data for the B-N test 1

Phase	$\alpha_L$	$\alpha_R$	$\rho_L$	$u_L$	$p_L$	$\rho_R$	$u_R$	$p_R$	CFL	$\gamma$
1	0.5	0.5	1.0	0.0	1.0	0.125	0.0	0.1	0.5	1.4
2			1.0	0.0	1.0	0.125	0.0	0.1	0.5	1.4

The first “artificial” test is created in order to reproduce the results of the first test of the Euler system (see table 5.1, Test 1). Therefore, the volume fraction field is set to a constant value of  $\alpha_{LR} = 0.5$  on the entire domain, while the left and right state are the same as those of table 5.1, Test 1, for both phases. Table 5.2 summarize the initial conditions for the simulation. The results are shown in Fig. 5.21 in which the correct behaviour is retrieved, the two phases fields effectively reproduce the Euler system results and the volume fraction field stays constant as expected.



Table 5.3: Left and right initial data for the B-N test 2

Phase	$\alpha_L$	$\alpha_R$	$\rho_L$	$u_L$	$p_L$	$\rho_R$	$u_R$	$p_R$	CFL	EOS
1	0.8	0.3	1.0	0.0	1.0	1.0	0.0	1.0	0.45	$(\gamma = 1.4)$
2			1.0	0.0	1.0	0.125	0.0	0.1	0.5	$(\gamma = 1.4)$

### 5.7.2.2 Test 2: Pure interface advection

The second test is crafted in order to assess the correct implementation of the non-conservative term discretization, as explained in section 5.2.3. The idea is to set constant values across the domain for all the fields except the volume fraction, that is set to a step function, going from the value 0.8 on the left side of the domain to 0.3 on the right, the onset point for the step is at  $x = 0.5$ . The closure for the interfacial velocity and pressure  $u_I, p_I$  is set such that  $p_I = 0$ . Fig. 5.22 shows the results. As expected, only the volume fraction field evolves as a transported scalar field, the slope of the curve also highlights the high diffusivity of the Rusanov scheme, being the mesh quite coarse and the scheme first-order.

### 5.7.2.3 Tests 3-7: A set of RPs from S. A. Tokareva and E. F. Toro

Table 5.4: Left and right initial data for the B-N test from S. A. Tokareva and E. F. Toro, phase  $k = 1$ 

Test	$x_0$	$\alpha_L$	$\alpha_R$	$\rho_L$	$u_L$	$p_L$	$\rho_R$	$u_R$	$p_R$	CFL	$\gamma$	$p_0^\infty$
1	0.5	0.8	0.3	1.0	0.0	1.0	1.0	0.0	1.0	0.45	1.4	
2	0.5	0.2	0.9	1900	0.0	10	1950	0.0	1000	0.45	3.0	3400
3	0.5	0.8	0.3	1.0	0.75	1.0	0.125	0.0	0.1	0.45	1.4	
4	0.5	0.8	0.5	1.0	-2.0	0.4	1.0	2.0	0.4	0.45	1.4	
6	0.8	0.7	0.2	1.0	-19.5975	1000	1.0	-19.5975	0.01	0.45	3.0	100

Table 5.5: Left and right initial data for the B-N test 3-8, phase  $k = 2$ 

Test	$x_0$	$\alpha_L$	$\alpha_R$	$\rho_L$	$u_L$	$p_L$	$\rho_R$	$u_R$	$p_R$	CFL	$\gamma$	$p_0^\infty$
1	0.5	0.2	0.7	0.2	0.0	0.3	1.0	0.0	1.0	0.45	1.4	
2	0.5	0.8	0.1	2	0.0	3	1.0	0.0	1.0	0.5	1.35	
3	0.5	0.2	0.7	1.0	0.75	1.0	0.125	0.0	0.1	0.45	1.4	
4	0.5	0.2	0.5	1.0	-2.0	0.4	1	2.0	0.4	0.45	1.4	
6	0.8	0.3	0.8	1.0	-19.5975	1000	1.0	-19.5975	0.01	0.45	1.4	

The set of tests presented in this section are proposed in the work S. A. Tokareva and E. F.

Toro (2010). The six test problems are characterized by the same structure: they are **RP** for the **B-N** system in which a discontinuity placed at the coordinate  $x = x_0$  separates two constant states, (the “left” and “right” state). The initial conditions are summarized in tables 5.4 and 5.5 and the results are showcased in Figures 5.23 to 5.27. The phases can be routinely equipped with a Perfect Gas **EoS**

$$p = (\gamma - 1)\rho e \quad (5.114)$$

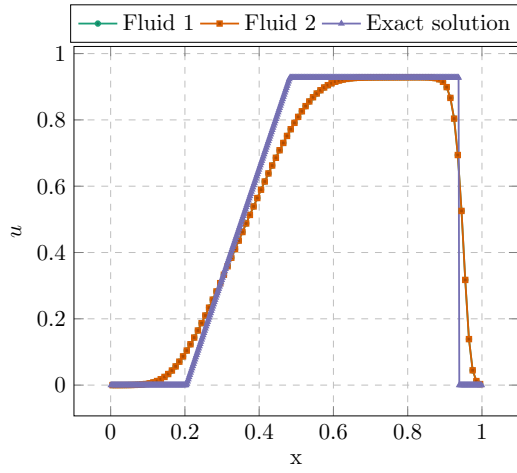
or a Stiffened Gas **EoS** to model liquid phases,

$$p = (\gamma - 1)\rho e - \gamma p_0^\infty$$

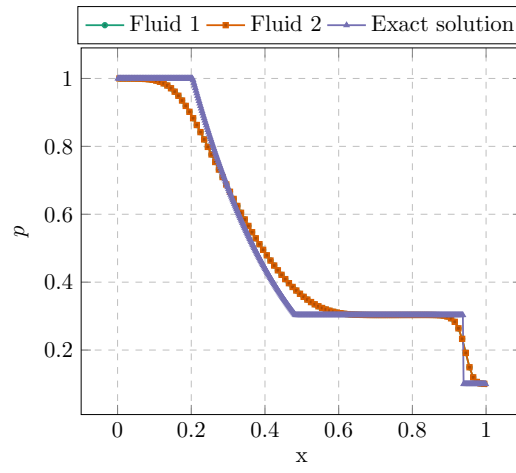
Tables 5.4 and 5.5 include one coefficient  $\gamma$  in the first case and two coefficients  $\gamma, p_0^\infty$  in the second case. The convective scheme employed is a first-order Rusanov scheme.

- Test 1 consists in a left rarefaction, right shock and a right traveling solid contact for the phase 1, and a left rarefaction, a contact and a right shock for the phase 2.
- Test 2 features larger variations *w.r.t.* Test 3, hence being more demanding.
- Test 3 consists in a right shockwave, a right traveling contact discontinuity and a left sonic rarefaction wave for both phases.
- Test 4 showcases for both phases two symmetric rarefaction waves, bringing the density close to vacuum, and a trivial stationary contact wave.
- Test 6 solution contains for both phases, a right traveling strong shockwave, a contact discontinuity and a left rarefaction wave. The jump of pressure is very large  $\Delta p_k = 1000$

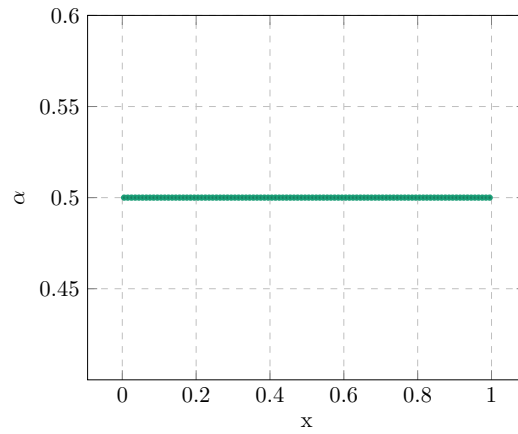
For all the cases the first-order Rusanov scheme is prohibitely diffusive. Even if the general solution behavior is somehow recovered, confirming the correctness of the implementation, this scheme with the chosen coarse resolution is not suitable. Even without the need of dealing with higher order schemes, it is apparent that a more suitable choice like the HLLC solver from S. A. Tokareva and E. F. Toro (2010) must be enforced in order to obtain accurate predictions. Nonetheless, this suite of tests is included in `josiepy` as a guiding structure to implement more complex approximation strategies for the **B-N** model and it is indeed useful as **CI** test to detect the introduction of bugs in the underlying structures. Since the **B-N** is not a direct topic of this work, we decided to forego the implementation of higher order schemes for the **B-N** model referring to, among others, the future work of Ait Ameer (2020); Loison (2023).



(a) Velocity profile



(b) Pressure profile



(c) Volume fraction profile

Figure 5.21: A test with constant  $\alpha$  to retrieve same results as Fig. 5.16,  $t = 0.25$ s

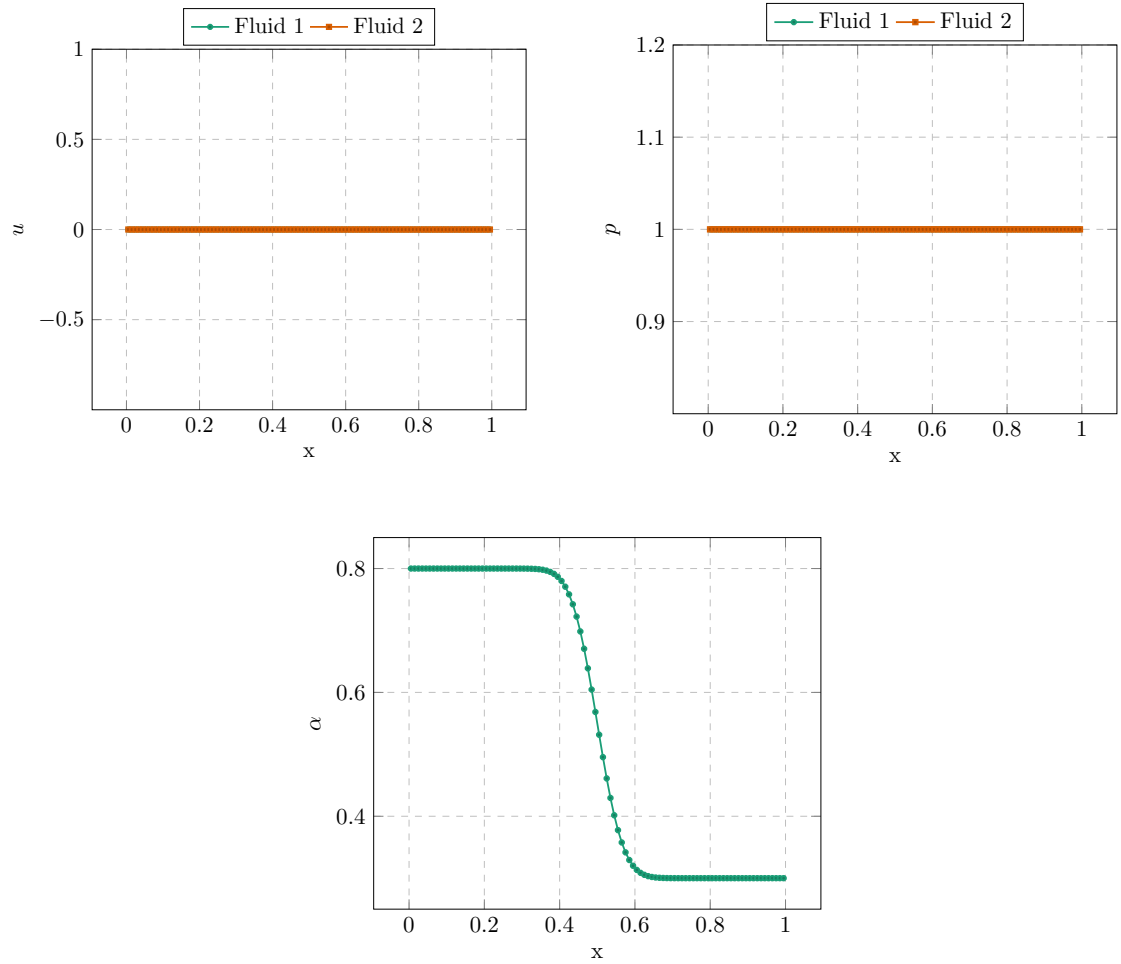


Figure 5.22: An artificial test in which the convective solver is disabled and only the non-conservative term is approximated, *i.e.* an advection problem

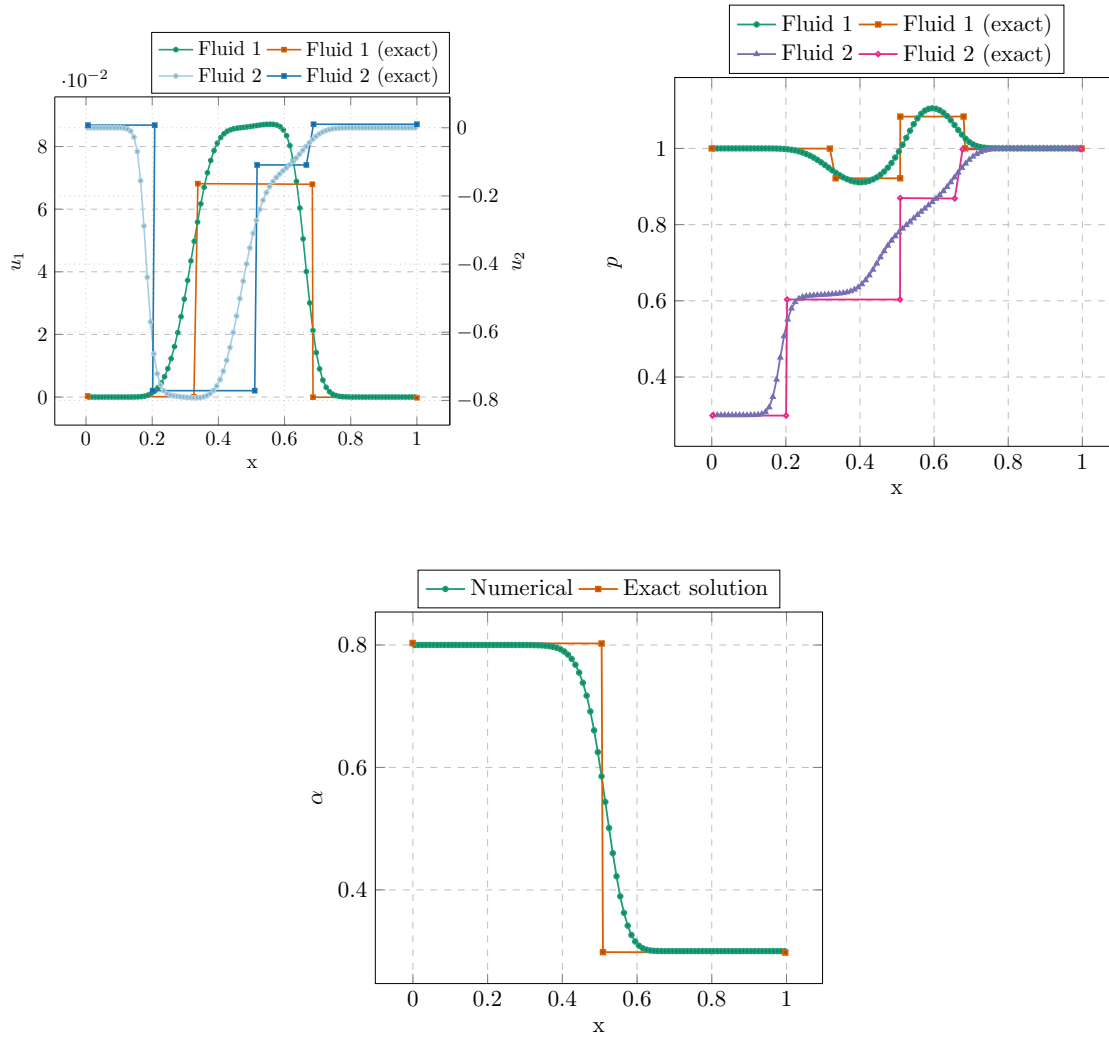


Figure 5.23: Test 1 from S. A. Tokareva and E. F. Toro (2010)

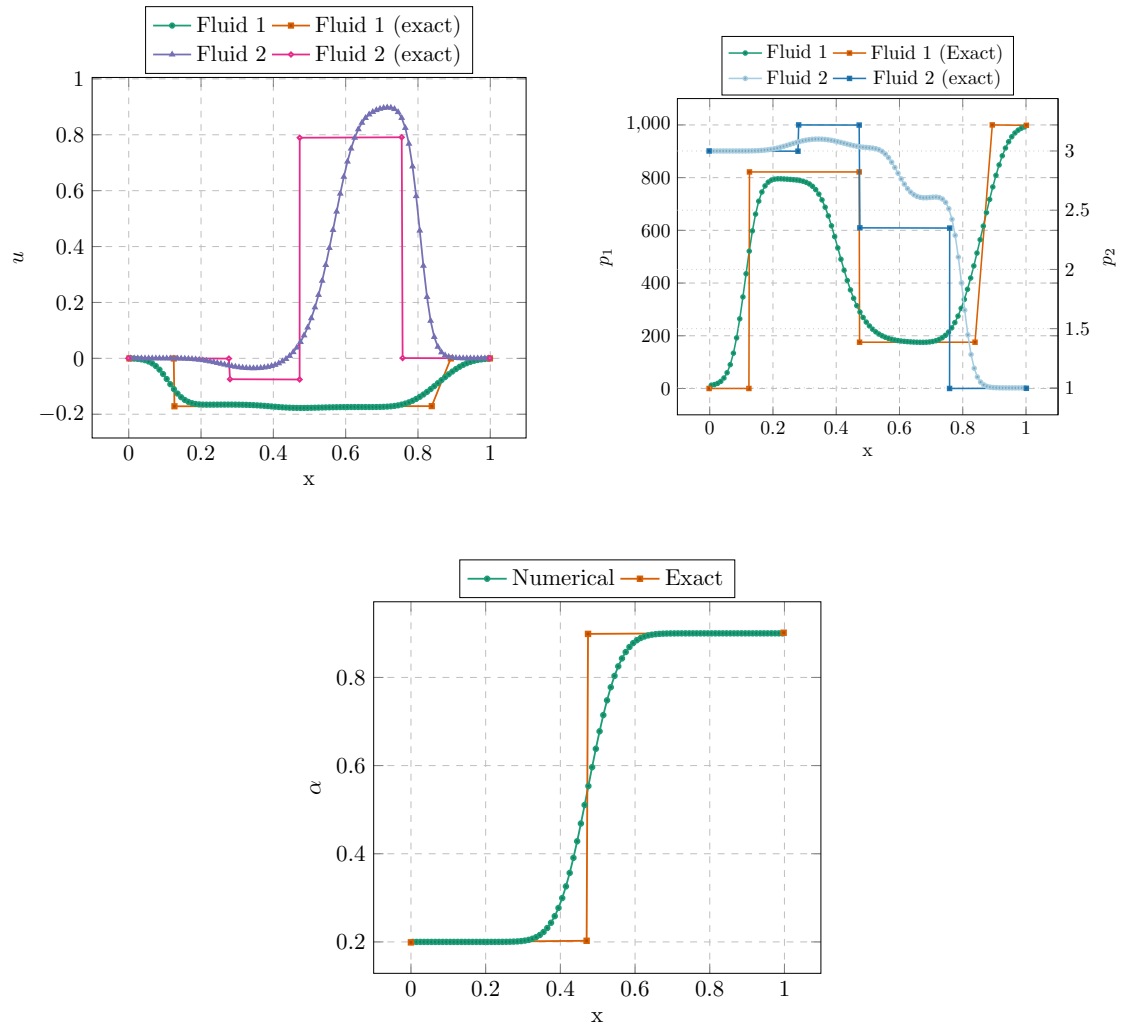


Figure 5.24: Test 2 from S. A. Tokareva and E. F. Toro (2010)

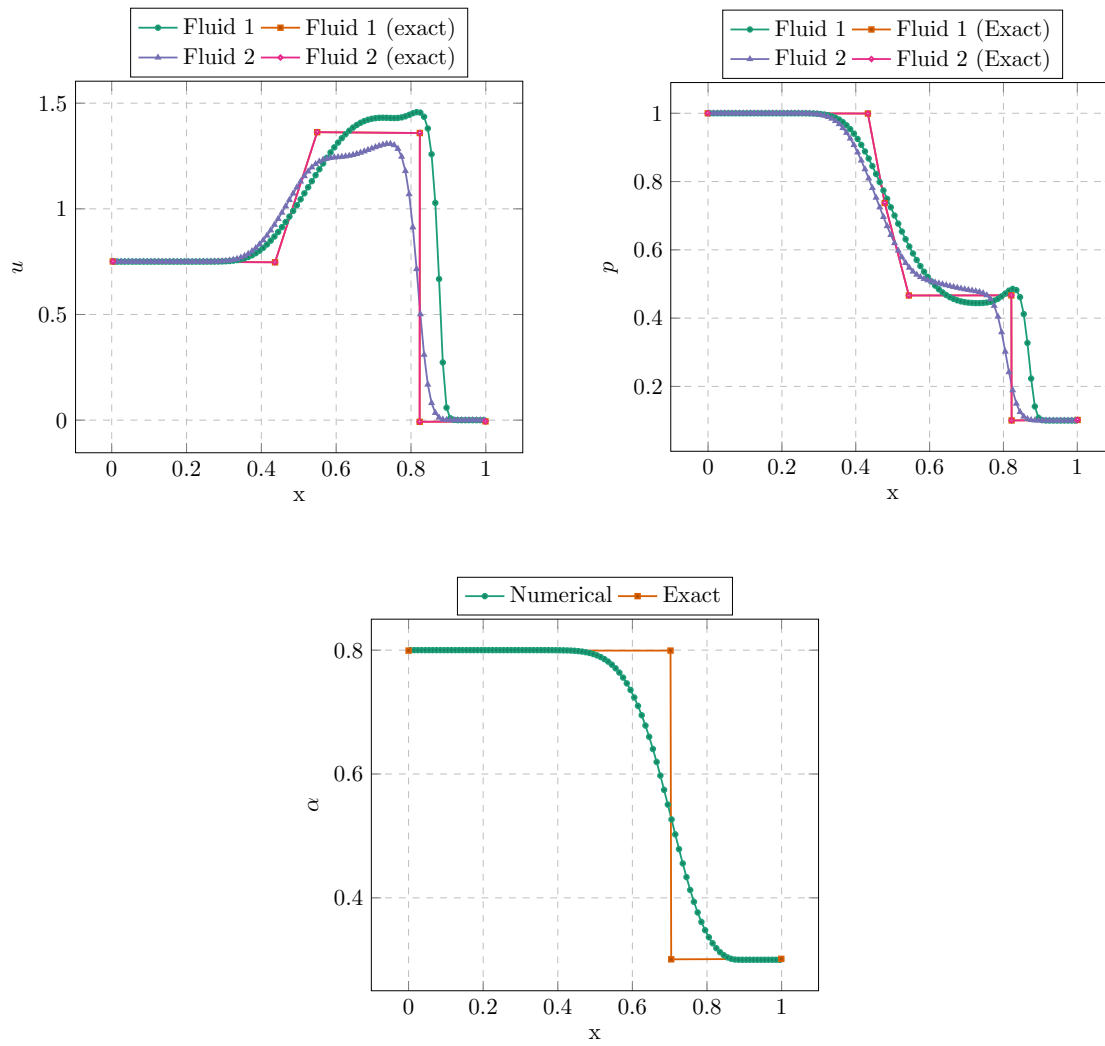


Figure 5.25: Test 3 from S. A. Tokareva and E. F. Toro (2010)

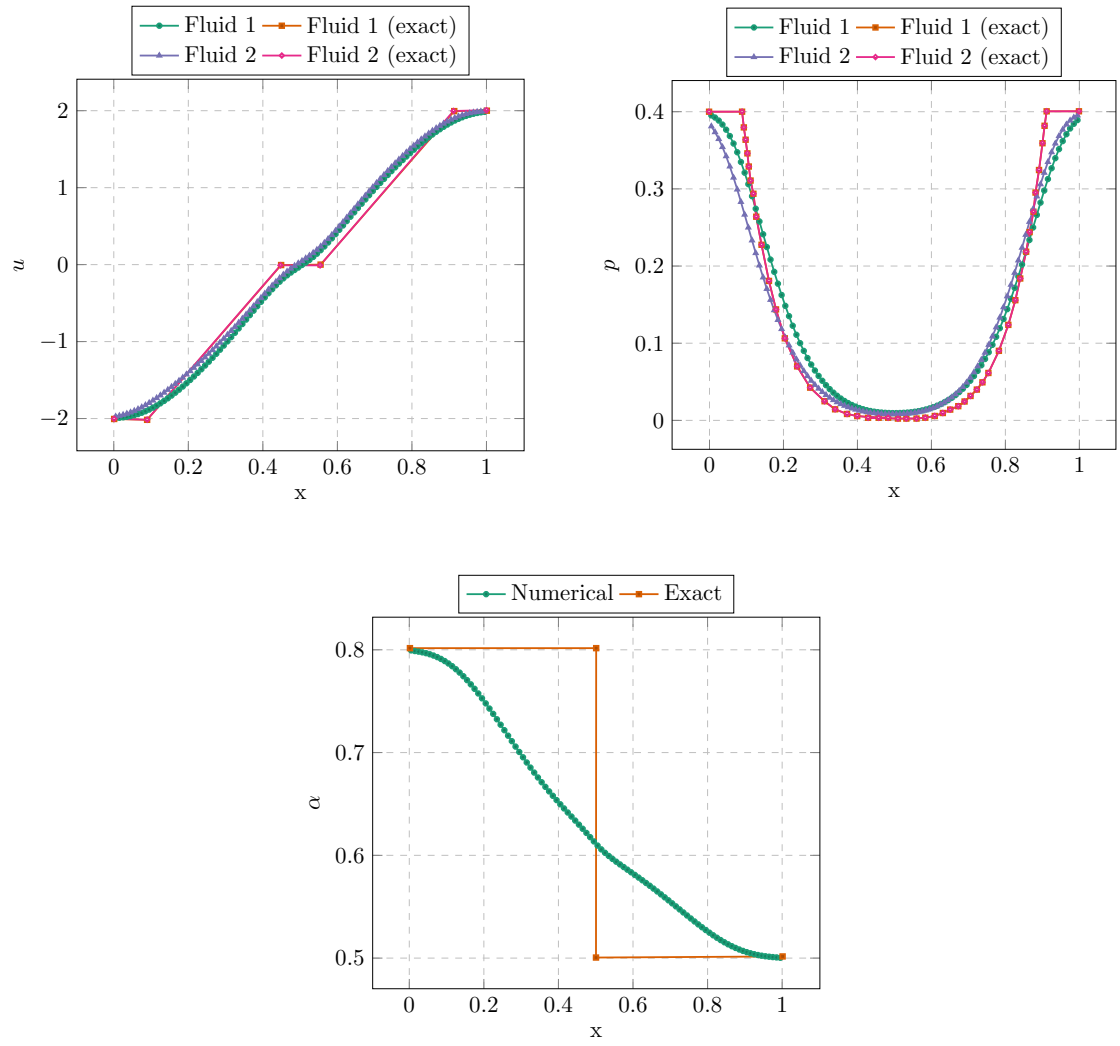


Figure 5.26: Test 4 from S. A. Tokareva and E. F. Toro (2010)



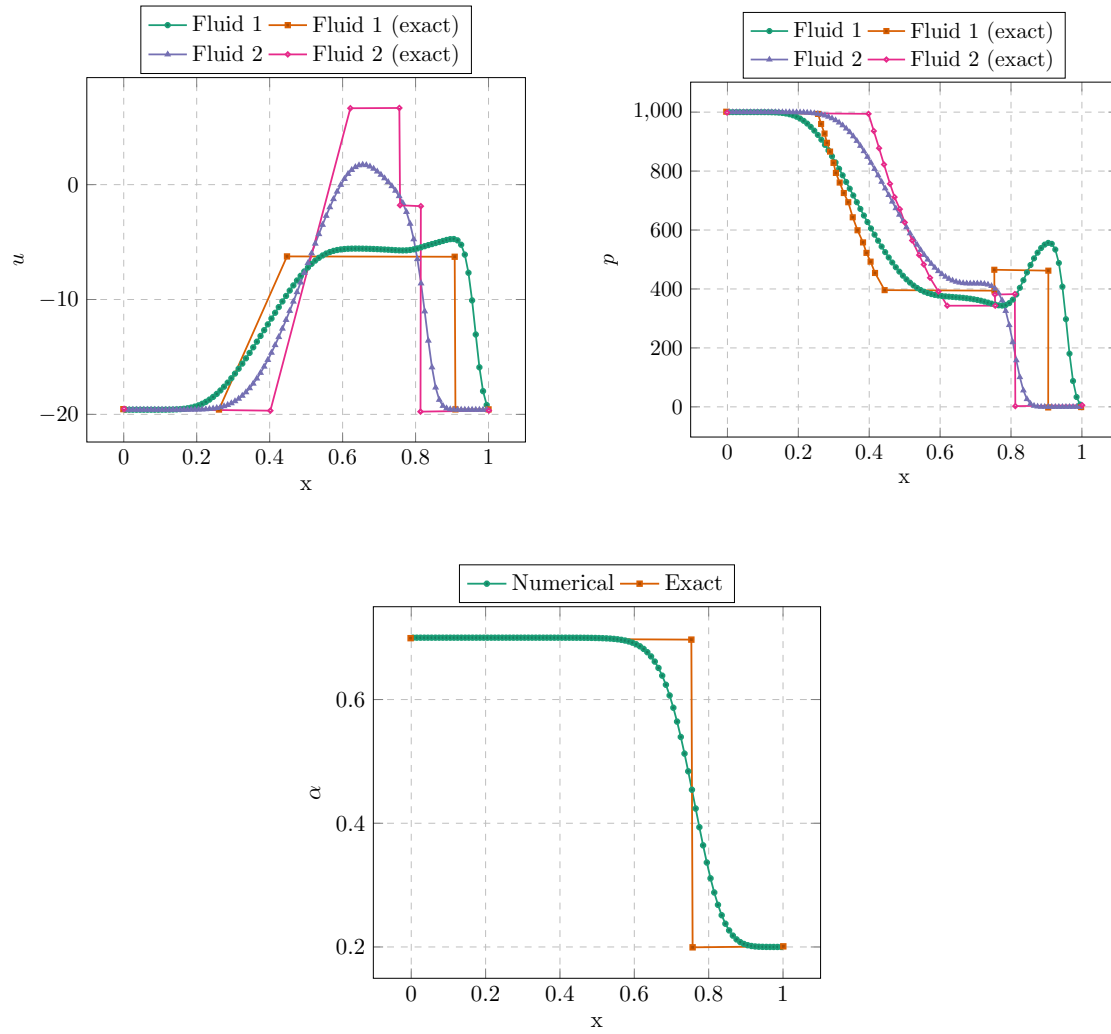


Figure 5.27: Test 5 from S. A. Tokareva and E. F. Toro (2010)

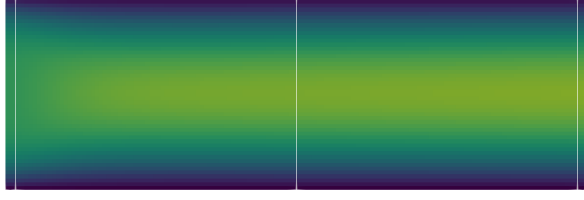
### 5.7.3 Hagen-Poiseuille flow for the Navier-Stokes (NS) system

In order to test the implementation of the diffusive schemes in *josiepy*, we present here a case in which the compressible NS equations have been implemented and a Hagen-Poiseuille laminar pipe flow configuration is simulated. The configuration of the case is the following: a box of dimension  $3 \times 1$  m is meshed with  $150 \times 50$  cells. For what concerns the BCs, differently from what we usually do in incompressible flow, the compressible NS system allows different ways of imposing BCs, Rudy and Strikwerda (1981) for example performs a thorough study on the different possibilities. The BCs we impose for this situation are:

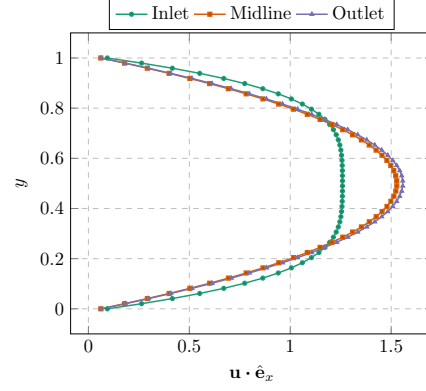
- On the left boundary of the domain, an “inlet” BC is imposed, that is the velocity and the internal energy of the flow are imposed ( $\mathbf{u} = (u(y), 0) \text{ m s}^{-1}$ ,  $e = 300 \text{ J kg}^{-1}$ ), while for the pressure a zero-gradient condition is enforced. The profile imposed for the x-component of velocity is  $u(y) = -20((y - 0.5)^4) + 1.25$ .
- On the right boundary of the domain, an “outflow” BC is assigned, that is, the pressure is imposed to a fixed value  $p = p^{\text{eos}}(p = 1, e = 300)$  and the rest of the quantities have a zero-gradient Neumann condition.
- On the top and bottom walls, the velocity is set to a fixed zero value  $\mathbf{u} = \mathbf{0}$ , while all the other quantities have a zero-gradient Neumann BC

The internal field is initialized with the fields  $\mathbf{u} = (1, 0) \text{ m s}^{-1}$ ,  $e = 300 \text{ J kg}^{-1}$ ,  $\rho = 1 \text{ kg m}^{-3}$ , the dynamic diffusivity of the fluid is set to  $\mu = 1.8\text{e-}5 \text{ Pa s}$ , the bulk viscosity is set to zero and the thermal diffusivity  $k_T = 2.1\text{e-}5 \text{ m}^2 \text{ s}^{-1}$ .

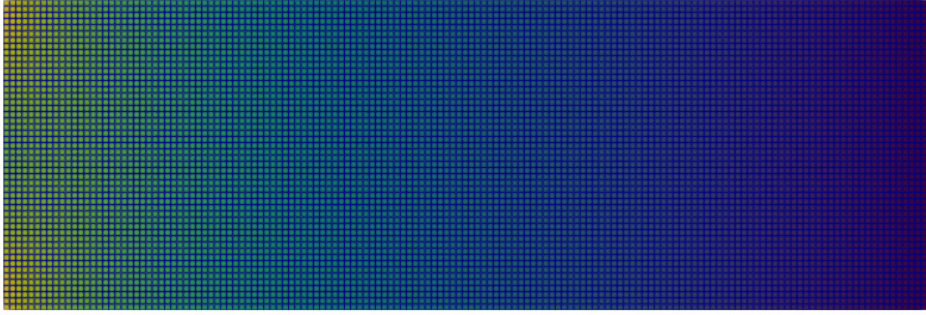
Fig. 5.28 shows the final outcome at  $t = 10$  s. In particular Fig. 5.28b shows the velocity profiles at three different stations of the pipe (the stations are highlighted in Fig. 5.28a). We can effectively retrieve a Hagen-Poiseuille velocity profile ensuring that the diffusion-related schemes and data structure are performing as expected.



(a) Velocity field  $\mathbf{u} \cdot \hat{\mathbf{e}}_x$ , [1.8 m s<sup>-1</sup> to 6.1e-2 m s<sup>-1</sup>]. The white vertical lines define stations where a velocity profile is extracted



(b) Velocity profile  $\mathbf{u} \cdot \hat{\mathbf{e}}_x$  for the three vertical stations shown in Fig. 5.28a



(c) Pressure field, [130 Pa to 120 Pa]

Figure 5.28: Poiseuille Flow

#### 5.7.4 Convective subsystem and relaxation procedure from Cordesse system

In this section we present the first three validation cases from Cordesse (2020, Section 4.3). These validation cases verify the first subsystem and the relaxation procedure used in a splitting strategy to solve a two-phase flow system featuring capillarity force. The aim of these tests, apart from the mere validation, is to put in place a first block to reproduce the results retrieved by Cordesse with the proprietary CEDRE software, in a FOSS context as the software josiepy. The tests are classical shock-tube problems found in literature. We use a Stiffened Gas EoS,

$$p_k = (\gamma_k - 1)\rho_k e_k - \gamma_k p_k^\infty$$

where  $\gamma_k$  is the ratio of the isentropic specific heat coefficients  $\gamma = C_p/C_V$ ,  $p_k^\infty$  is the residual pressure at zero temperature used to enforce liquid or solid behavior. The corresponding expression for the internal energy is:

$$e_k = C_{V,k} T_k \frac{p_k + \gamma_k p_k^\infty}{p_k + p_k^\infty}$$

with  $C_V$  the specific heat at constant volume. The chosen values for the coefficients of the EoS are:

$$\begin{cases} \text{Air: } \gamma_1 = 1.4, & p_1^\infty = 0, & C_{V,1} = 1000 \\ \text{Water: } \gamma_2 = 4.4, & p_2^\infty = 6.8 \times 10^8, & C_{V,2} = 4180 \end{cases}$$

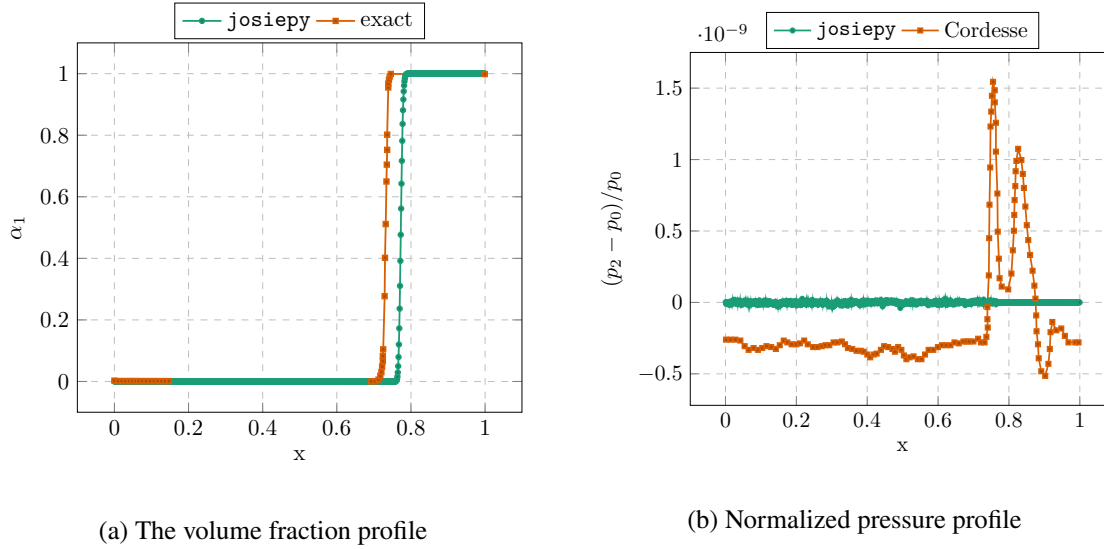
A major difference we want to highlight *w.r.t.* the configuration by Cordesse is that we use a 1<sup>st</sup> order HLLC scheme with the same relaxation procedure described in the original work for simplicity, while the original study employs a second order multislope HLLC scheme (Le Touze et al. 2014).

#### 5.7.4.1 Pure interface advection

Table 5.6: Pure interface advection ( $\epsilon = 1e-8$ ,  $\rho_1 = 10 \text{ kg m}^{-3}$ ,  $\rho_2 = 1000 \text{ kg m}^{-3}$ )

$x$	$\alpha_1[-]$	$u[\text{m s}^{-1}]$	$p_1[\text{Pa}]$	$p_2[\text{Pa}]$
$[0, 0.5[$	$\epsilon$	100	1e5	1e5
$[0.5, 1]$	$1 - \epsilon$			

The first benchmark case is a 1D pure interface advection. The domain is 1 m long. The interface between the two phases is initially placed at  $x = 0.5$ , and it separates two zones of the domain hosting the individual phase almost completely ( $\alpha_k = 1 - \epsilon$ ,  $\epsilon \ll 1$ , the volume fraction is not exactly one to avoid zero division). The domain is discretized in 1000 cells and the BCs are fixed values on the left and right boundaries, with the values of the left and right fields summarized in table 5.6 (even if it is not incredibly important since the simulation is stopped before the BCs can interact with the solution). Fig. 5.29a shows the volume fraction profile. The general slope of the curve is well reconstructed even if the onset of the transition is captured a bit later *w.r.t.* the exact solution, probably due to the 1<sup>st</sup> order scheme; Fig. 5.29b shows the relative pressure ratio. The pressure in this case is supposed to stay constant and in facts Cordesse (2020) explains the oscillating behavior with accumulating rounding errors in the energy computation, errors that we do not encounter in our implementation.


 Figure 5.29: Advection test at instant  $t = 229 \mu\text{s}$  from Cordesse (2020)

#### 5.7.4.2 Moderate density water-air shock tube

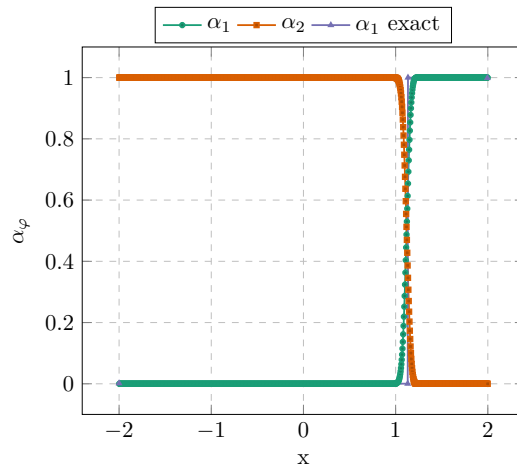
 Table 5.7: Water-air shock tube with moderate density ratio ( $\epsilon = 1\text{e-}8$ ,  $\rho_1 = 50 \text{ kg m}^{-3}$ ,  $\rho_2 = 1000 \text{ kg m}^{-3}$ )

$x$	$\alpha_1[-]$	$u[\text{m s}^{-1}]$	$p_1[\text{Pa}]$	$p_2[\text{Pa}]$
$[-2, 0.75[$	$\epsilon$	0	1e9	1e9
$[0.75, 2]$	$1 - \epsilon$		1e5	1e5

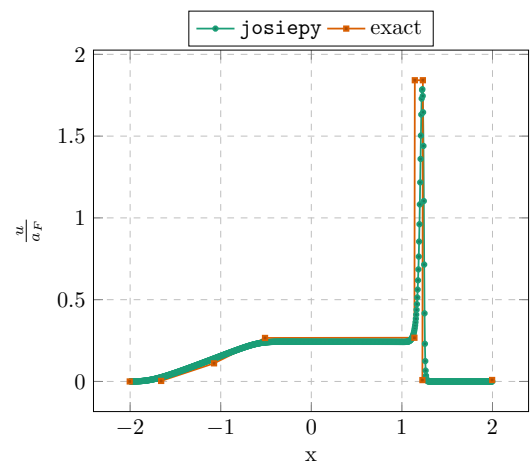
The second test we propose is a water-air shock tube happening in a domain of length 4 m. The interface lies initially at 0.7 m and it separates pressurized water in a nearly pure volume fraction (the usual  $\epsilon \ll 1$  to avoid numerical problems) at  $10^9$  Pa, from a zone of the domain with almost pure air at  $10^5$  Pa. The complete initial states are reported in table 5.7. Despite the lower order scheme, we find a very good agreement with the theoretical data. The only under optimal outcome is related to the temperature profile shown in Fig. 5.30d. The higher numerical diffusion creates smoother gradients and undershooting in the zone of the shock.

#### 5.7.4.3 High-Density water-air shock tube

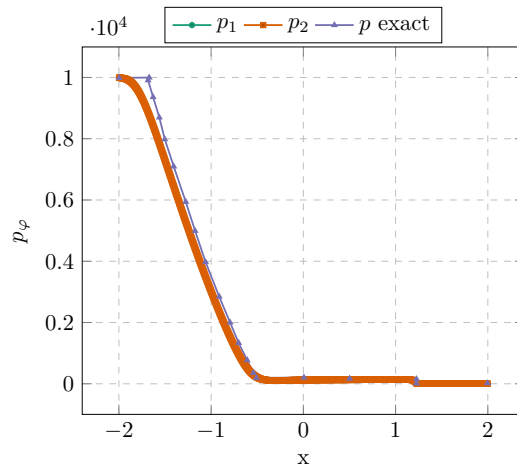
The last test case features another shock-tube configuration, but with higher initial densities (the ratio is raised to 1000) as shown in table 5.8. The length of the domain is 1 m, and the interface appears at  $x = 0.75$ , the water phase has  $10^9$  Pa pressure on the left of the domain,



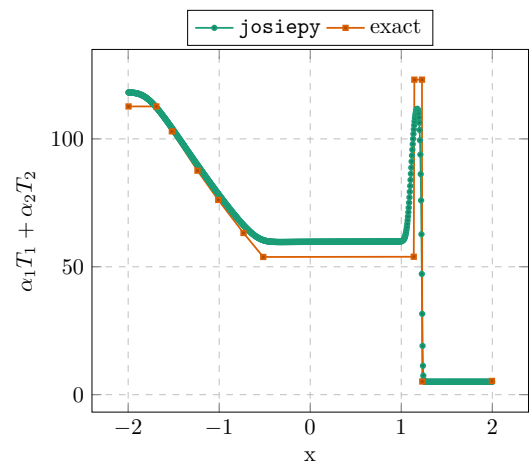
(a) The volume fraction profile



(b) The velocity profile



(c) Pressure profile



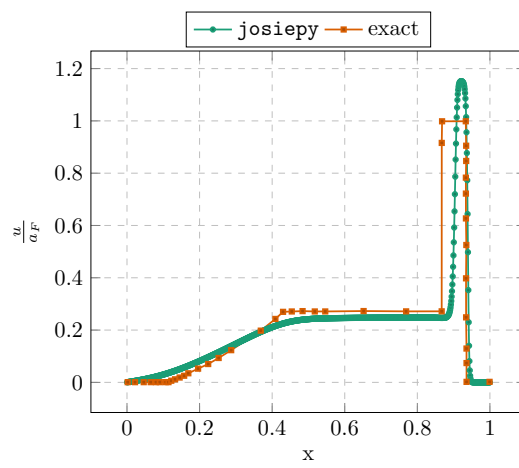
(d) Temperature profile

 Figure 5.30: Shock tube at moderate density ratio at instant  $t = 900 \mu\text{s}$  from Cordesse (2020)

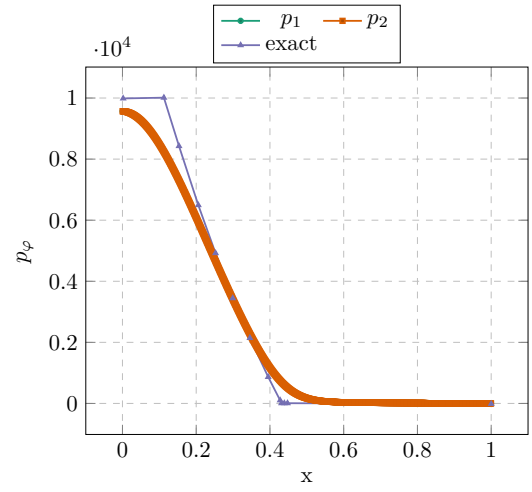
 Table 5.8: Water-air shock tube with high density ratio ( $\epsilon = 1\text{e-}8$ ,  $\rho_1 = 1 \text{ kg m}^{-3}$ ,  $\rho_2 = 1000 \text{ kg m}^{-3}$ )

$x$	$\alpha_1[-]$	$u[\text{m s}^{-1}]$	$p_1[\text{Pa}]$	$p_2[\text{Pa}]$
$[-2, 0.75[$	$\epsilon$	0	1e9	1e9
$[0.75, 2]$	$1 - \epsilon$		1e5	1e5

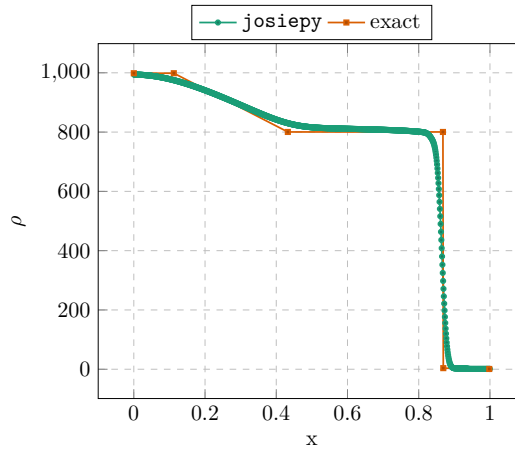
the air phase is at  $10^5$  Pa on the right side. As always, phases are almost pure except for a small  $\epsilon$  value to avoid numerical difficulties. In this more challenging scenario, the first order HLLC schemes shows higher diffusion in all the fields shown in Fig. 5.31 *w.r.t.* the second order scheme used by Cordesse. In the velocity profile, the shock peak is overshoot and delayed *w.r.t.* the theoretical slope. The results are in any case in good agreement.



(a) The velocity profile



(b) Normalized pressure profile



(c) Density profile

Figure 5.31: Shock tube at high density ratio at instant  $t = 240 \mu\text{s}$  from Cordesse (2020)



## 5.8

## Results for the convective subsystem featuring an interfacial area density equation

Table 5.9: Initial conditions for the system with the interfacial area density equation,  $P = 0.71$ 

$y$ [m]	$\alpha$ [-]	$\omega$ [m s <sup>-1</sup> ]	$\mathbf{u}$ [m s <sup>-1</sup> ]	$z$ [kg <sup>1/2</sup> ]
$> \tilde{y}(x) + \Delta x$	0.8	0	$(-0.25, 0)$	0
$< \tilde{y}(x) - \Delta x$	0.2	0	$(0.25, 0)$	0
$\tilde{y}(x) \pm \Delta x$	0.5	0	$(0, 0)$	$1e-2$

In this section we present the simulation we prepared in order to numerically experiment with the convective part of the system described in section 3.6, *i.e.* eq. (3.159). In this section we present the details of a simulation the purpose of which is to show the physical outcomes the model. The chosen test case is a variation of the classical Kelvin-Helmoltz instability. The initial domain is a  $1 \times 1$  m square meshed with  $1500 \times 1500$  and  $500 \times 500$  cells and the numerical schemes employed are respectively a 1<sup>st</sup> order Rusanov scheme for the finest case and a MUSCL-Hancock extension of the same scheme with a MINMOD limiter (see section 5.7.1.3 for details) for the coarse case. The BCs are configured as follows:

- Periodic BCs on the left and right boundary
- On the top and bottom boundaries a zero-gradient condition for all the fields except the vertical component of the velocity  $\mathbf{u} \cdot \hat{\mathbf{n}} = \pm v$ , that is set to zero  $v \triangleq 0$ .

The fields are initialized at  $t = 0$  dividing the original domain in two vertical sections, separated by a an interface of thickness  $\Delta x$ , lying at the coordinates  $y = \tilde{y}(x) \pm \Delta x$ , where  $\Delta x$  is the cell spacing that is taken uniform for the entire mesh. The midline of the interface is described by the function  $\tilde{y}(x)$ , and for the current case it has the following expression:

$$\tilde{y}(x) = \begin{cases} 0.5, & x \in [0, x_a[ \cup ]x_b, 1] \\ K \sin\left(\pi \frac{x - x_a}{x_b - x_a}\right), & x \in [x_a, x_b] \end{cases} \quad (5.115)$$

That is, in the interval  $[x_a, x_b]$ , a sinusoidal perturbation of the interface is introduced in order to induce destabilization of the interface faster than what would happen in a classical case with a completely flat interface (also, the model of section 3.6 does not feature viscosity, therefore

the distortion of the interface would be caused only by numerical dissipation hence resulting in very slow time scales and longer simulations would be needed). See Fig. 5.32 for a plot of the initial fields at  $t = 0$  of the simulation and table 5.9 for the actual initial values. The EoS for both phases is barotropic,

$$p_k = C_k \rho_k^{\gamma_k}, c_k = C_k \gamma_k \rho_k^{\gamma_k-1} \quad (5.116)$$

and the parameters are summarized in table 5.10. Figures 5.33 and 5.34 show the evolution

Table 5.10: Barotropic EoS coefficients

$k$	$C_k$	$\gamma_k$
1	1	1.4
2	1	1.65

of the case in time for the two time instants  $t = 2.99$  s, 4.99 s, for both the very refined case solved with the first-order scheme, and the less-refined test with a second-order scheme. It is possible to appreciate the correct onset of the typical Kelvin-Helmoltz vortices enhanced by the sinusoidal instability. In particular, looking at Figures 5.33b, 5.33e, 5.34b and 5.34e, we see the  $z(\rho, \Sigma)$  field, that is function of the interfacial area density. We can appreciate the higher concentration of  $z$  in the zones which feature the smallest vortices length scales, intuitively the most likely to feature the highest distortion, creation of ligaments and satellite objects, hence higher  $\Sigma$ . This intuitive results is a positive outcome giving optimism on the mathematical soundness of the model presented in section 3.6. Figures 5.33c, 5.33f, 5.34c and 5.34f show the evolution of the field  $\omega$ , that we can interpret as a sort of small scale interface pulsation indicator. The highest gradient of  $\omega$  are an indication of interface stretching and small scale inclusions pulsation.

In terms of comparison between the first order and second order schemes, we can say that the higher order scheme, despite the less resolved mesh, is capable of capturing the evolution at even finest scales *w.r.t.* the first order implementation. In terms of speed of execution, the reduction in size of the mesh translates on a 92% reduction in computational time *w.r.t.* the highly resolved case, in the context of this specific testing scenario. This allows meaningful comparisons and fast iteration on the scheme development to be performed in very reasonable times, justifying the effort of providing a second-order scheme for this problem.

The simulation of this test case shows the flexibility of the `josiepy` library<sup>1</sup> and it is a first attempt at numerically benchmarking the theoretical outcomes of section 3.6. The current choice of destabilizing the flow with a localized sinusoidal perturbation is a first simple way

<sup>1</sup>The entirety of the code for this test case fits a 500 lines Jupyter notebook

of performing such task. The sinusoidal “bump” connects to the rest of the constant interface in a way that perturbrates all the spatial frequency of the system. Future refining of the case might envisage a perturbation that stimulates only a specific set of frequencies as a continuous sinusoidal interfacial wave.

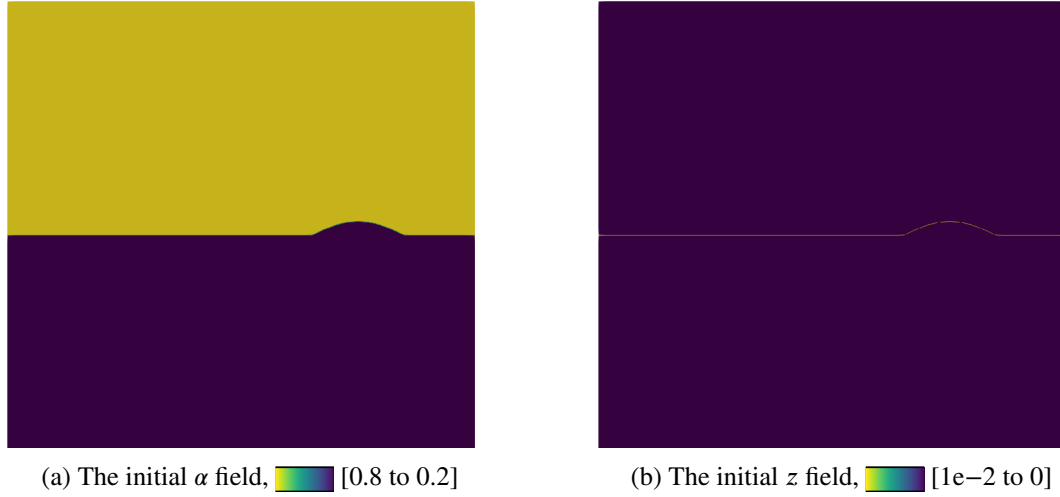


Figure 5.32: Initial condition for the Kelvin-Helmoltz case

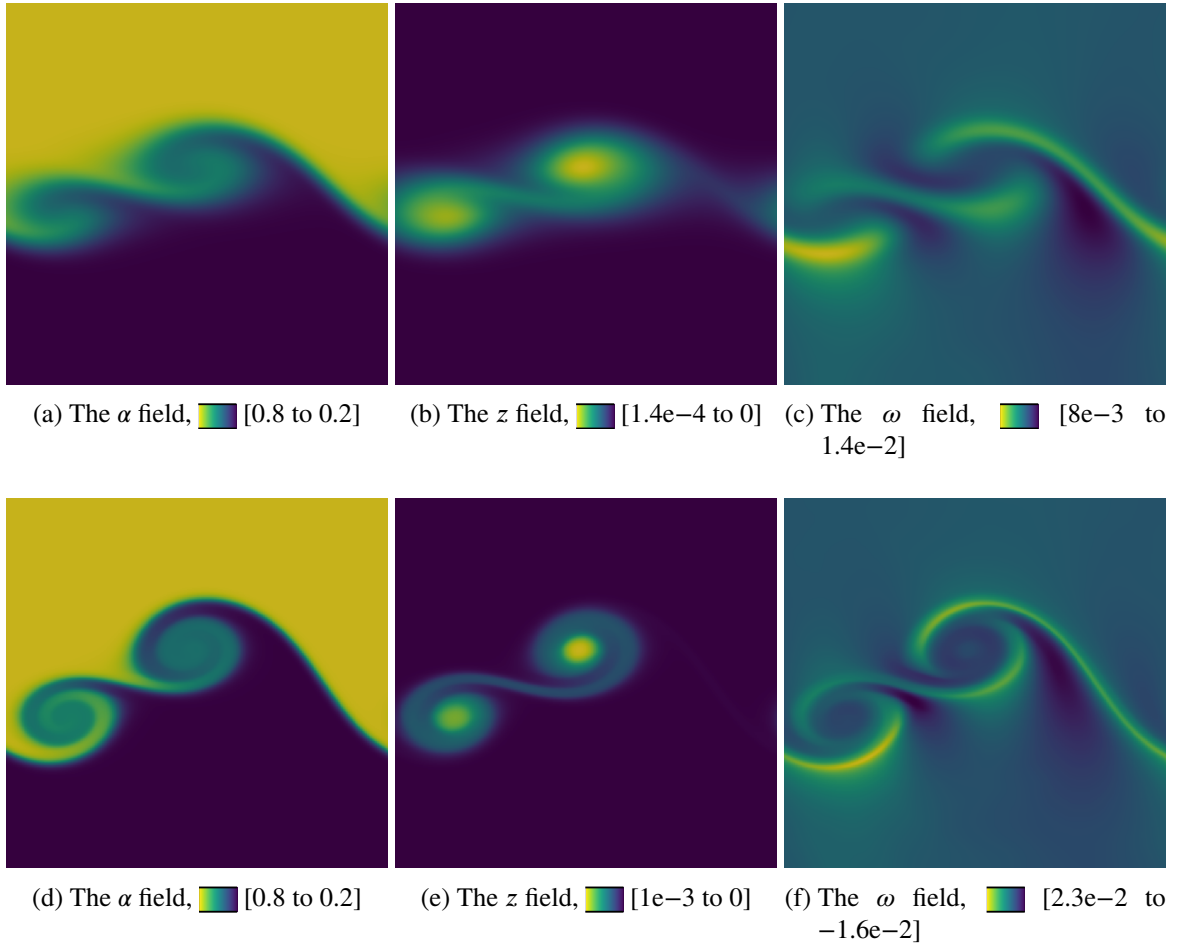


Figure 5.33: State of the Kelvin-Helmoltz case for  $t = 2.99$  s. Top row with the fist-order scheme on a  $1500 \times 1500$  mesh. Bottom one with MUSCL-Hancock MINMOD scheme on a  $500 \times 500$  mesh

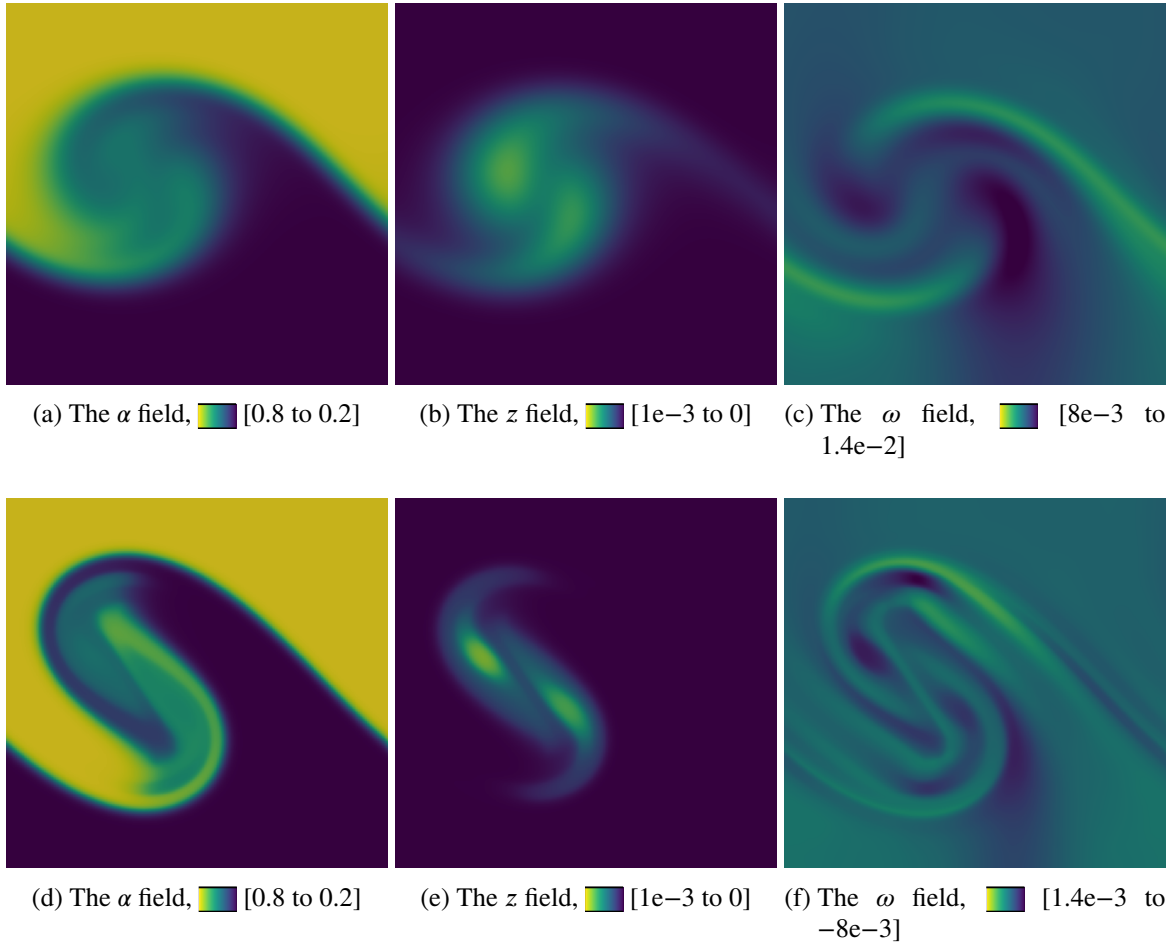


Figure 5.34: State of the Kelvin-Helmoltz case for  $t = 4.99$  s. Top row with the first-order scheme on a  $1500 \times 1500$  mesh. Bottom one with MUSCL-Hancock MINMOD scheme on a  $500 \times 500$  mesh

## 5.9 Conclusions and perspectives

---

The implementation of a generic numerical framework implemented with an accessible programming language as Python into the library `josiepy` has been presented in this chapter. The general aspects of the numerical strategy allow to decouple the implementation of specific terms in isolated modules granting agility in the implementation of numerical aspects from the perspective of the modeler. In facts, at the current date, `josiepy` features a wide selection of different solvers and schemes that showcase the flexibility of this choice and allow to simulate classical systems like the Euler equations with exact and approximated solvers, two-phase flow advanced problems as the ones governed by the B-N (Baer and Nunziato 1986), complex two-phase flow systems necessitating of elaborate numerical strategies as the sub-system from Cordesse (2020). In addition, the simulation of the brand new model we introduced in section 3.6. All the schemes are moreover easily usable with a second-order extension via a MUSCL-Hancock approach that is also already available in the library. In facts the `josiepy` proved to be a very useful tool to ascertain the numerical simulability of the theoretical model we developed with the SAP and to produce a set of testing scenarios that can be easily adapted to future implementations in an agile way.

Future work is currently ongoing, especially with the post-doc of Ait Aneur (2020) and the PhD thesis of Loison (2023), in order to extend the capabilities to high-order DG schemes and to include additional models, such as the complete system from Cordesse, Di Battista, Chevalier, et al. (2020) with large and small scale capillarity effects. On the scientific computing side we are also actively working on the parallelization of the code for modern architectures, we talk about this subject more specifically in chapter 6.

# 6

## Software architecture for scientific computing

Most of current scientific endeavor nowadays could not be achieved without the aid of a computer. A lot of high-level science discoveries are obtained thanks to well-optimized software. Two-phase flow modeling and simulation is no exception. The long-term success of a software depends not only on the scientific outcome it can provide, but also on the easiness of accessing its codebase, the level of testing and associated coverage, the readiness of its API, the cross-compatibility of its code, the efficiency of its build system, the rapidity of bug fixes and new features implementations. While those aspects are now common knowledge in the applied computer science world, scientific software tends to neglect the aspects that are peculiar of modern software development that are also undervalued in publications, where code is hardly accessible. We believe that the free access to high-quality software stacks that follow modern practices for Software Architecture is paramount to long-term success of scientific efforts. For this reason, in this chapter we present the two libraries that have been developed in the context of this thesis: a FVM-based PDE solver written in Python named **josiepy**, whose target is to be a fast-prototyping tool for modelers, and **Mercur(v)e**, a C++ library that implements the details of the geometrical post-processing routine that has been discussed in chapter 4. Both libraries are FOSS, and accessible freely on the Internet.

### 6.1

#### josiepy: A PDE solver written in Python without sacrificing (too much) performance

---

The situation of simulation codes in the framework of physics systems governed by PDEs is characterized by the co-existence of two kinds of software: industrial codes that are extremely flexible and performant, with enhanced multiphysics capabilities. Notable examples are closed commercial or enterprise specific suites like CEDRE from ONERA (Le Touze et al. 2014), products from NUMECA, ANSYS, ConvergeCFD, COMSOL just to cite few. FOSS examples are also available like OpenFOAM (Greenshields 2015) or FreeFem++ (Hecht 2012). These advanced suites are often optimized for the execution time on CPU-centric virtual Non-Uniform Memory Access (NUMA) architectures on HPC clusters. On the other side, academic codes that work on simplified configurations, sometimes limited to low dimensionalities, that assume lots of hypotheses on the mesh structure (*e.g.* undeformed, regular Cartesian meshes

with fixed spacing) and can have limited applicability on real scenarios. For the first category, often the chosen implementation language is low-level and compiled, often being C++, with a steep learning curve and a real difficulty of integrating localized modifications to specific part of the code. Testing different numerical approaches to solve a complex PDE system can be long, cumbersome and sometimes frustrating, if the solver you need to embed your schemes into is written in a complex, compiled, programming language such as C++ or you have limited access to the entire code base or documentation. Despite C++ being invented to produce pain, and pain being cathartic sometimes, *when half way through the journey of your life, you find yourself in a gloomy wood, because the path which led aright was lost*<sup>1</sup>, having an easy framework to test your numerical experiments without the need of a savvy guide like Virgil, can be useful. Jokes apart, having C++ a clear established role in the field, we believe that there is room for a mid-ground solution that can be optimized on the global “Time to Market” of a simulation including in the evaluation metrics also the time a developers needs to understand and get acquainted with a certain code base in order to set up a reasonably complex case. In addition to that, nowadays, the Python language is used in all sort of HPC fields, like Data Analysis and AI, its ecosystem is equipped with powerful compiled extensions like NumPy (Harris et al. 2020), that allow fast debugging cycles without sacrificing too much performance in the execution.

Here comes *josiepy* (Di Battista 2019), a Python library that allows to easily solve multi-dimensional<sup>2</sup> PDE systems encoded in the same spirit as described in section 5.2. It is heavily based on the solid fundamentals of the NumPy API—that incidentally allows to use different backends like `DistArray` (Ganger 2008), for distributed arrays allowing parallel execution on CPUs, or `cupy` (Preferred Infrastructure, Inc. 2021), for execution on GPUs, or even Bauer and Garland (2019) for hybrid large scale workflows— and it basically allows to “program” your test case in Python, without the need of cryptic and limited configuration files. In addition to that, *josiepy* also ships a basic structured mesher based on TFI (Thompson et al. 1998, Chap. 3) that allows to “program” your meshes directly in Python, without the need of interfacing with other meshing tools. The simulation results can be exported in common files such as XDMF and inspected in post-processing tools such as Paraview. The framework aims at providing a fast prototyping tool granting the possibility to have all the required tools for a simulation in one place: a structured mesh generator, a selection of space and time schemes, high-order extensions of those schemes, a selection of explicit time integrators, everything easily installable using Python packaging tools and accessible with a familiar programming language API. The

---

<sup>1</sup>Adapted from Alighieri (n.d.)

<sup>2</sup>At this time, the dimensionality of the problem is limited to 2D. Future plans are to extend to 3D block-structured meshes



numerical schemes can be implemented directly in Python and they can run as fast as NumPy API allows, that in most cases is close to what compiled high performance languages like C and C++ allow (Harris et al. 2020).

### 6.1.1 Architecture of the code

`josiepy` has been thought as a fast prototyping tool that would allow to experiment on a single aspect of the resolution of a PDE problem, without the need of knowing how things are done for the other modules. A representation of the core modules is shown in Fig. 6.1. We can see that the code is logically arranged in three macro-areas:

- Mesh Generation
- Physics of the problem to be solved
- Numerics

The three aspects are then combined and used in the `solver` module the aim of which is to orchestrate the simulation exposing the mesh geometry to the chosen scheme, advance the simulation in time, and store all the metadata about the simulation. The `io` module operates on the solver object in order to serialize simulation results to the disk. All the underlying data structures for storing mesh-related quantities or the problem fields evolved in time during the simulation are based on the NumPy array API. This allows to define operations on the entire block of values, exploiting modern processor optimizations for large float vectors operations (SSE, AVX2, AVX512) that NumPy ships internally via the linking to optimized linear algebra implementations, but also other type of low-level containers that are not stored locally on a machine, as for the bare NumPy ndarrays, but potentially on distributed machines and GPUs (Harris et al. 2020; Preferred Infrastructure, Inc. 2021; Bauer and Garland 2019).

#### 6.1.1.1 Physics of the Problem

The model template which `josiepy` supports is,

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) + \mathbf{B}(\mathbf{q}) \cdot \nabla \mathbf{q} - \nabla \cdot (\mathbf{K}(\mathbf{q}) \cdot \nabla \mathbf{q}) + s(\mathbf{q}) = \mathbf{0} \quad (6.1)$$

in order to take care of most of the problems governed by PDEs that concern the underlying scientific context of this work with a logic separation of the various type of contributions (convective, non-conservative, diffusive and source terms) together with time integration implemented with the Method of Lines (MoL) semi-discretization technique. This is indeed an

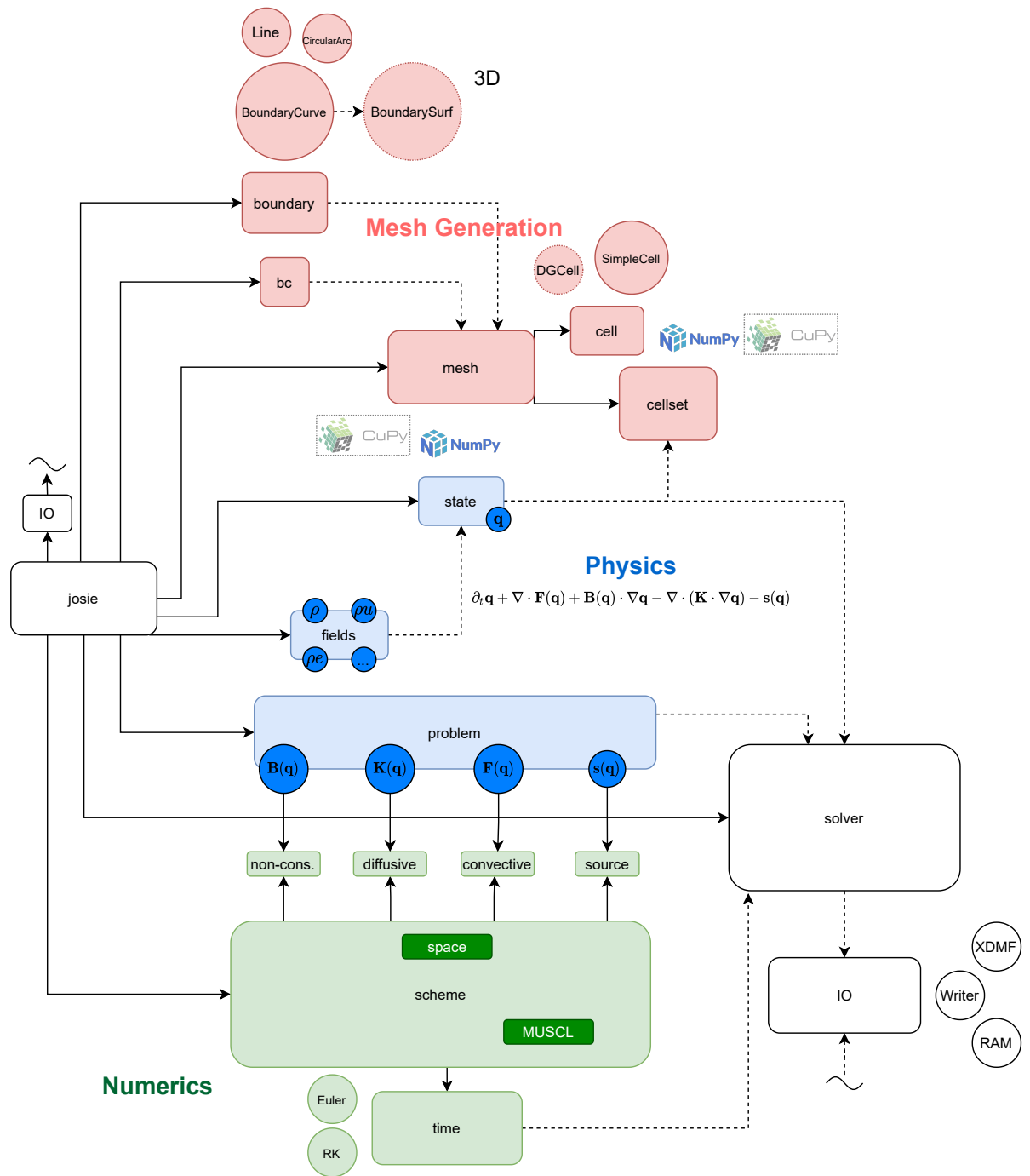


Figure 6.1: The architecture of the josi solver

opinionated choice, but it appeared flexible enough for the purpose of this work. The dynamic nature of the Python language allows faster refactoring *w.r.t.* C++ nonetheless.

**The State object** The first thing to do is to define the phase space containing the problem variables. For example, to create a solver for the Euler system (section 3.4.1.7), the fields are defined as,

```
from josie.fluid.fields import FluidFields

class EulerFields(FluidFields):
    # Conservative fields
    rho = 0
    rhoU = 1
    rhoV = 2
    rhoE = 3

    # auxiliary fields
    U = 4
    V = 5
    rhoe = 6
    p = 7
    c = 8
```

Strictly speaking, the fields that are present in the equations are the conservative ones, but to compute the other properties of the system, notably using the [EoS](#), it might be necessary to access other auxiliary fields, arbitrarily defined (increasing the memory footprint of the running simulation and the corresponding size of the saved file on disk). Once the fields of the problem are defined, they can be assigned to a State object,

```
class EulerState(State):
    fields = EulerFields
```

which, under the hood, is a familiar NumPy array. In facts it is possible to *cast* a NumPy array into a State object, and access it with integer indices or using its fields,

```
rnd_state = np.random.random(len(EulerFields)).view(EulerState)
fields = rnd_state.fields
assert rnd_state[0] == rnd_state[fields.rho]
```

The class `EulerFields` is actually an enumeration of integers, that is fast to access and with small memory footprint, that can be used to index the `State` array without the need of remembering the index of the desired field. `State` can also be multidimensional (as it is within the solvers namespace), and the `Ellipsis [...]` Python object allows to retrieve all the values of a set of fields for all the cells of a mesh irrespective of the dimensionality (1D, 2D or 3D) of the problem.

```
rnd_state = np.random.random((100, 100, len(EulerFields))).view(EulerState)
U = rnd_state[..., EulerFields.U]
```

There is also a functional `API` to define a state class, that can allow the definition of a state in a slightly less verbose fashion,

```
MyStateClass = StateTemplate("rho", "rhoU", "rhoV")
zero = np.zeros(10).view(MyStateClass)
```

**The Problem object** The Problem is the “continuous” representation of the problem the user is willing to simulate. The Problem class implements the corresponding methods to the terms  $F(q)$ ,  $K(q)$ ,  $B(q)$ ,  $s(q)$  of the reference eq. (6.1). As an example, the implementation of the `josie.euler.EulerProblem` needs to provide the implementation of the convective flux for the Euler system, that is

```
class EulerProblem(Problem):
    def __init__(self, eos: EOS):
        self.eos = eos

    def F(self, cells: CellSet) -> np.ndarray:
        values: Q = cells.values.view(Q)
        fields = values.fields

        num_cells_x, num_cells_y, _ = values.shape

        # Flux tensor
        F = np.empty(
            (num_cells_x, num_cells_y, len(ConsFields), MAX_DIMENSIONALITY)
        )
```

```
rhoU = values[..., fields.rhoU]
rhoV = values[..., fields.rhoV]
rhoE = values[..., fields.rhoE]
U = values[..., fields.U]
V = values[..., fields.V]
p = values[..., fields.p]

rhoUU = np.multiply(rhoU, U)
rhoUV = np.multiply(rhoU, V)
rhoVV = np.multiply(rhoV, V)
rhoVU = rhoUV # np.multiply(rhoV, U)

F[..., fields.rho, Direction.X] = rhoU
F[..., fields.rho, Direction.Y] = rhoV
F[..., fields.rhoU, Direction.X] = rhoUU + p
F[..., fields.rhoU, Direction.Y] = rhoUV
F[..., fields.rhoV, Direction.X] = rhoVU
F[..., fields.rhoV, Direction.Y] = rhoVV + p
F[..., fields.rhoE, Direction.X] = np.multiply(rhoE + p, U)
F[..., fields.rhoE, Direction.Y] = np.multiply(rhoE + p, V)

return F
```

The `Problem.F` method operates on all the cell meshes (`cells`) as an entire entity to promote vectorized operations driven by NumPy. To understand better the data structure for the mesh and the values contained in each mesh cell, we will now discuss the mesh generation.

#### 6.1.1.2 Mesh Generation

In section 5.2.1 we discussed the mathematical background that backs our implementation of the structured mesher integrated in *josiepy*. Here we will discuss the choices we made for the [API](#). The mesh generation starts defining the boundaries of the domain, directly in Python

```
from josie.boundary import Line, CircleArc

left = Line([0, 0], [0, 1])
bottom = CircleArc([0, 0], [1, 0], [0.5, 0.5])
```

```
right = Line([1, 0], [1, 1])
top = Line([0, 1], [1, 1])
```

to be sure that the defined domain coincides exactly with what the user had in mind, without the need of saving the mesh in another format and then opening in a post-processing tool, it is possible to simply plot the curves that constitute the domain boundary,

```
for curve in [left, bottom, right, top]:
    curve.plot()
```

and the resulting image is shown in Fig. 6.2.

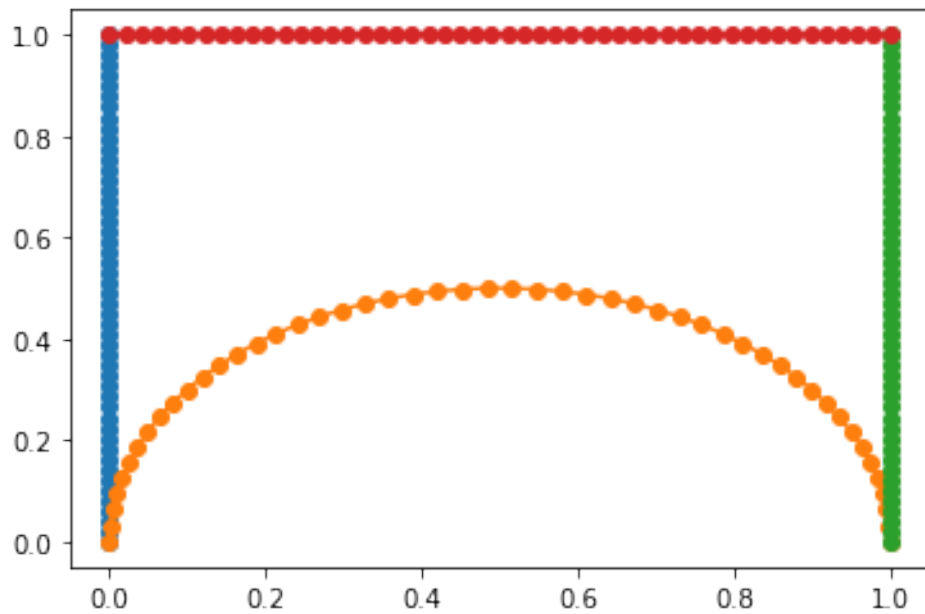


Figure 6.2: The plotted boundary of the domain

A problem governed by a system of PDEs requires the imposition of BCs on all the domain boundaries. With `josiepy` it is possible to directly assign the BCs to the actual boundaries of the domain. The generic module `josi.bc` provides the `Dirichlet` and `Neumann` base classes that the user can use respectively to impose a fixed value for the state on a boundary or a gradient value. As an example, let us impose a zero value on the left, bottom, and right boundary and a zero-gradient BC on the top boundary:

```
from josi.bc import Dirichlet, Neumann
```

```
Q_zero = np.zeros(len(EulerState.fields)).view(EulerState)
```

```
dQ_zero = Q_zero

left.bc = Dirichlet(Q_zero)
top.bc = Neumann(dQ_zero)
bottom.bc = Dirichlet(Q_zero)
right.bc = Dirichlet(Q_zero)
```

This way of defining the BCs is easy and allows to assign to the entire state the same BC. More often it is needed to set different conditions on the individual fields of a state. This is indeed also possible with josiepy,

```
from josie.bc import BoundaryCondition

Q_bc = EulerState(rho=Dirichlet(1), rhoU=Neumann(0), rhoV=Dirichlet(1),
                  rhoE=Dirichlet(1), rhoe=Dirichlet(3), U=Neumann(0),
                  V=Neumann(0), p=Dirichlet(1), c=Dirichlet(10))

left.bc = BoundaryCondition(Q_bc)
```

What is happening here is that we can assign per each field of the state a Dirichlet or Neumann condition. In facts, the boundary conditions objects act a bit magically based on the given input. If the given input value is an entire State object, then the BC is set on all the fields of the problem. If the input value is just a float, then the returned BC object is a ScalarBC, and it just sets the condition on one individual (that is “scalar”) field. As boundary values, not only constant values are possible: in order to impose space and time dependent boundary values, also Callable objects can be given as argument to the BCs. For reference, the josie.ns.bc module implements some specific BCs for the Navier-Stokes problem that make use of the Callable input.

Internally, the Boundary Conditions are applied assigning a value to the ghost cells of the mesh. Fig. 6.3 shows a sample mesh with the ghost shell next to each boundary highlighted. The entire mesh including the ghost cells is allocated in a contiguous memory region such that much of the NumPy slicing operations are just memory views and not copies of the same data. The corner values are unused in the current implementation and NaNs are stored in those locations. The Dirichlet BC implementation currently imposes the boundary value as the arithmetical mean of the neighboring points,

$$\varphi_D = \frac{\varphi_i + \varphi_G}{2} \quad (6.2)$$

where  $\varphi_D$  is the value to impose on the boundary face for the generic field  $\varphi$ . Therefore this is translated on a ghost value to be imposed:

$$\varphi_G = 2\varphi_D - \varphi_i \quad (6.3)$$

Similarly, the Neumann condition is imposed as:

$$\nabla\varphi_N \cdot \hat{\mathbf{n}} = \frac{\varphi_G - \varphi_i}{\Delta\mathbf{x}_{Gi} \cdot \hat{\mathbf{n}}} \triangleq g_N \quad (6.4)$$

hence the value of the field on the ghost cell is evaluated (at each time step) as:

$$\varphi_G = g_N \Delta\mathbf{x}_{Gi} \cdot \hat{\mathbf{n}} + \varphi_i \quad (6.5)$$

being  $g_N$  the value of the gradient to impose on the boundary for the generic field  $\varphi$  and  $\Delta\mathbf{x}_{Gi}$  is the relative distance vector between the boundary cell  $i$  and the corresponding ghost  $G$ .

### 6.1.1.3 The Numerics

Again referring to Fig. 6.1, we talked about the mesh generation module in section 6.1.1.2 and of the physical properties configuration in section 6.1.1.1. We are now going to introduce the fundamentals for the implementation of a numerical scheme well suited to a specific problem. The core object of our discussion is going to be the `josie.scheme.Scheme` class.

**The Scheme object** The Scheme object is an **Abstract Base Class (ABC)**, a sort of interface for those more comfortable with C-family languages vocabulary, that exposes methods that act like “hooks” that are called at specific moments during the simulation lifetime. Fig. 6.4 shows a graphical representation of how the `josie.solver.Solver` object interacts with the Scheme. In order to modularize even further the code, allowing very precise editing of the scheme implementation in order to implement exactly what the mathematical modeler has in mind, the classes `ConvectiveScheme`, `NonConservativeScheme`, `DiffusiveScheme` and `SourceScheme` are direct children mixins<sup>3</sup> of the parent Scheme and they expose the interface to implement the terms explained in chapter 5 for the discretization of the reference equation

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) + \mathbf{B}(\mathbf{q}) \cdot \nabla \mathbf{q} - \nabla \cdot (\mathbf{K}(\mathbf{q}) \cdot \nabla \mathbf{q}) + s(\mathbf{q}) = \mathbf{0}$$

<sup>3</sup>A mixin is a separate class that implements a specific functionality that can be then plugged into other classes via multi-inheritance to enrich their [API](#)



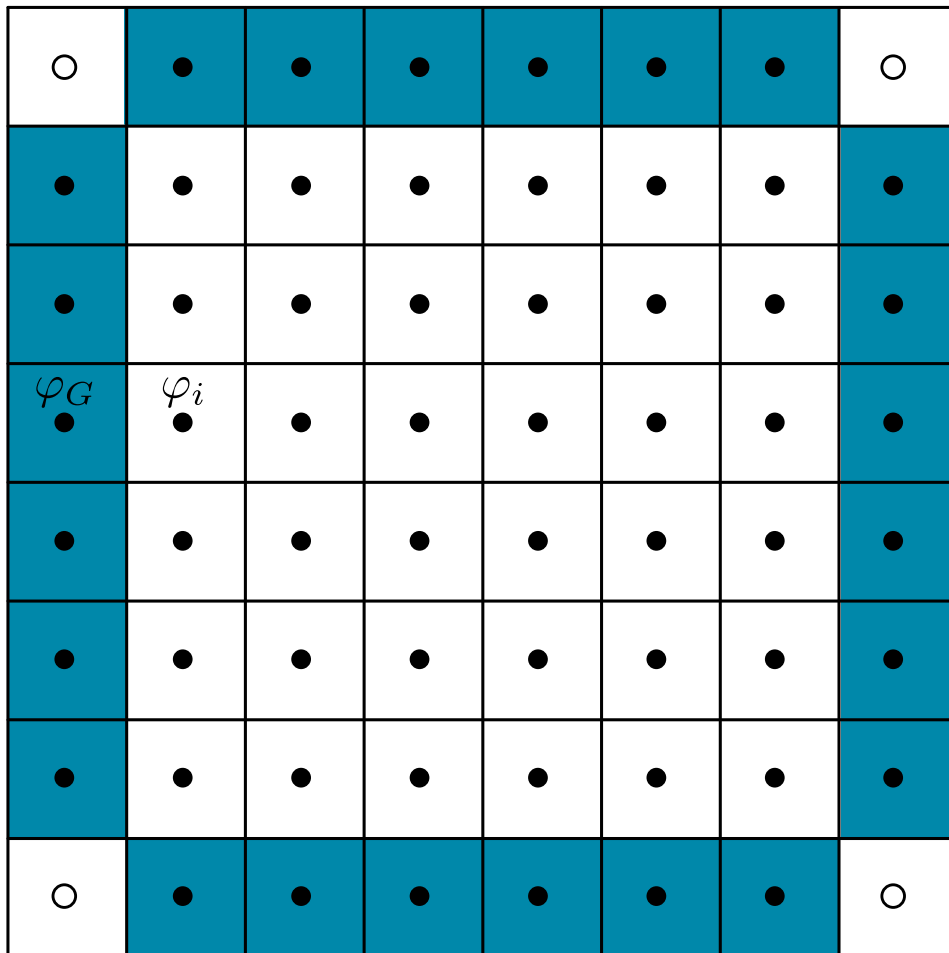


Figure 6.3: The mesh data structure including the ghost cells

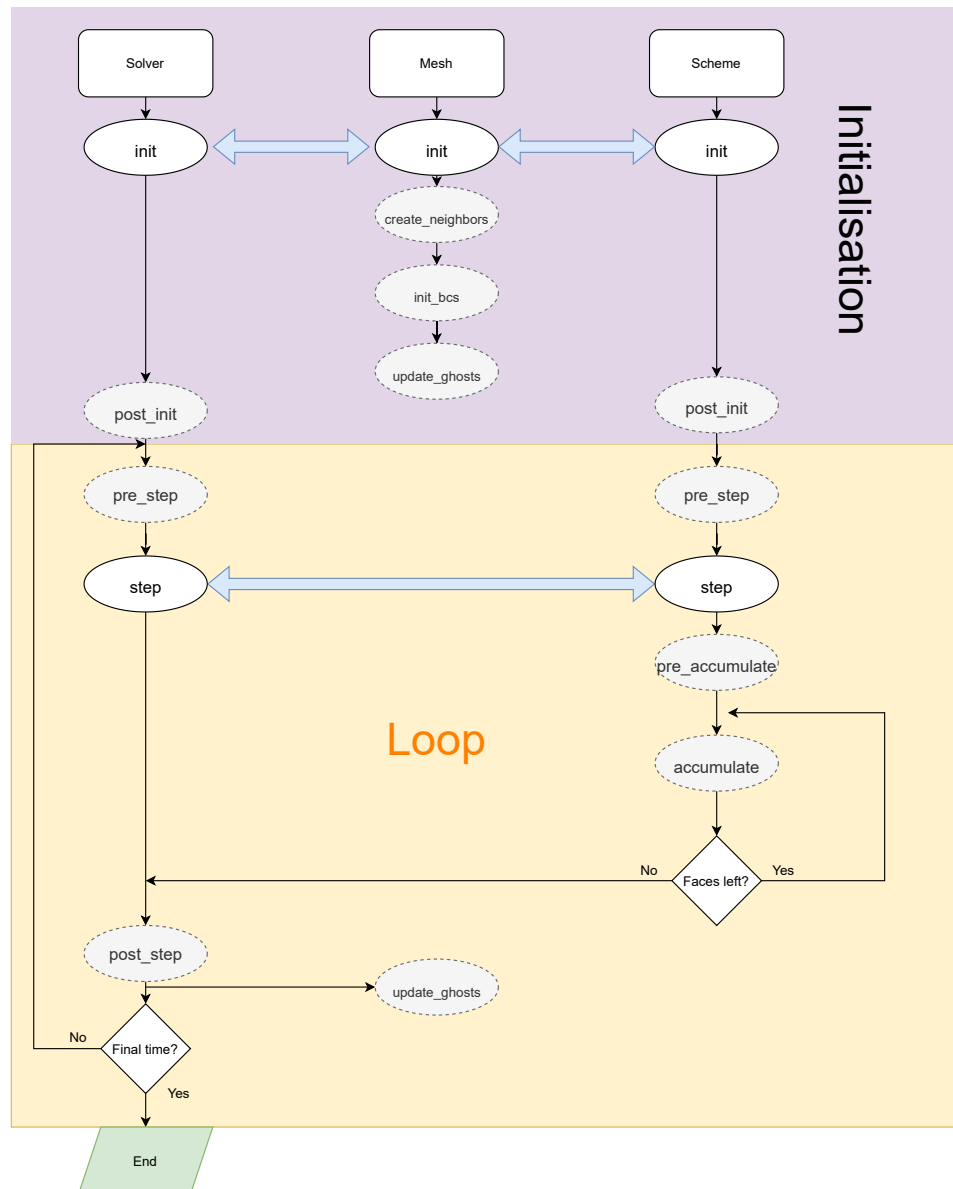


Figure 6.4: The flow of operations driven by the Solver object

that are:

- The convective term approximation (see section 5.2.2),

$$\sum_f^{N_{\text{faces}}} [F(q) \cdot \hat{n}]_f S_f, \quad \text{ConvectiveScheme.F}$$

- The non-conservative term approximation (see section 5.2.3),

$$\langle B(q) \rangle_{\Omega_i} \cdot \sum_f^{N_{\text{faces}}} [q \otimes \hat{n}]_f S_f, \quad \text{NonConservativeScheme.G}$$

- The diffusive term approximation (see section 5.2.4),

$$\sum_f^{N_{\text{faces}}} [\nabla q \cdot \hat{n}]_f S_f, \quad \text{DiffusiveScheme.D}$$

- The source term approximation (see section 5.2.5),

$$\langle s(q) \rangle_{\Omega_i} |\Omega_i|, \quad \text{SourceScheme.s}$$

In addition the `TimeScheme` class provides the abstract interface to implement time schemes (without needing to know how the other terms have been implemented). Some of those terms can be problem specific, notably the convective term implementation that depends on the eigenstructure of the problem, others may be reused as-is for most schemes. That is why a meta-module named `josie.general` contains all the schemes that can be used “vanilla” for all scheme implementations, notably different gradient schemes (see section 5.2.4) stored in `josie.general.schemes.diffusive`, the different time schemes (see section 5.2.6) in `josie.general.schemes.time`, and the source schemes (see section 5.2.5) in `josie.general.schemes.source`. Fig. 6.5 shows graphically this modularized organization.

As a final note, let us propose as an example the implementation of the Rusanov scheme for the Euler system as explained in section 5.3.3.1. This implementation is directly extracted from the `josie.euler.schemes` package.

```
class Rusanov(EulerScheme):
    @staticmethod
    def compute_sigma(
```

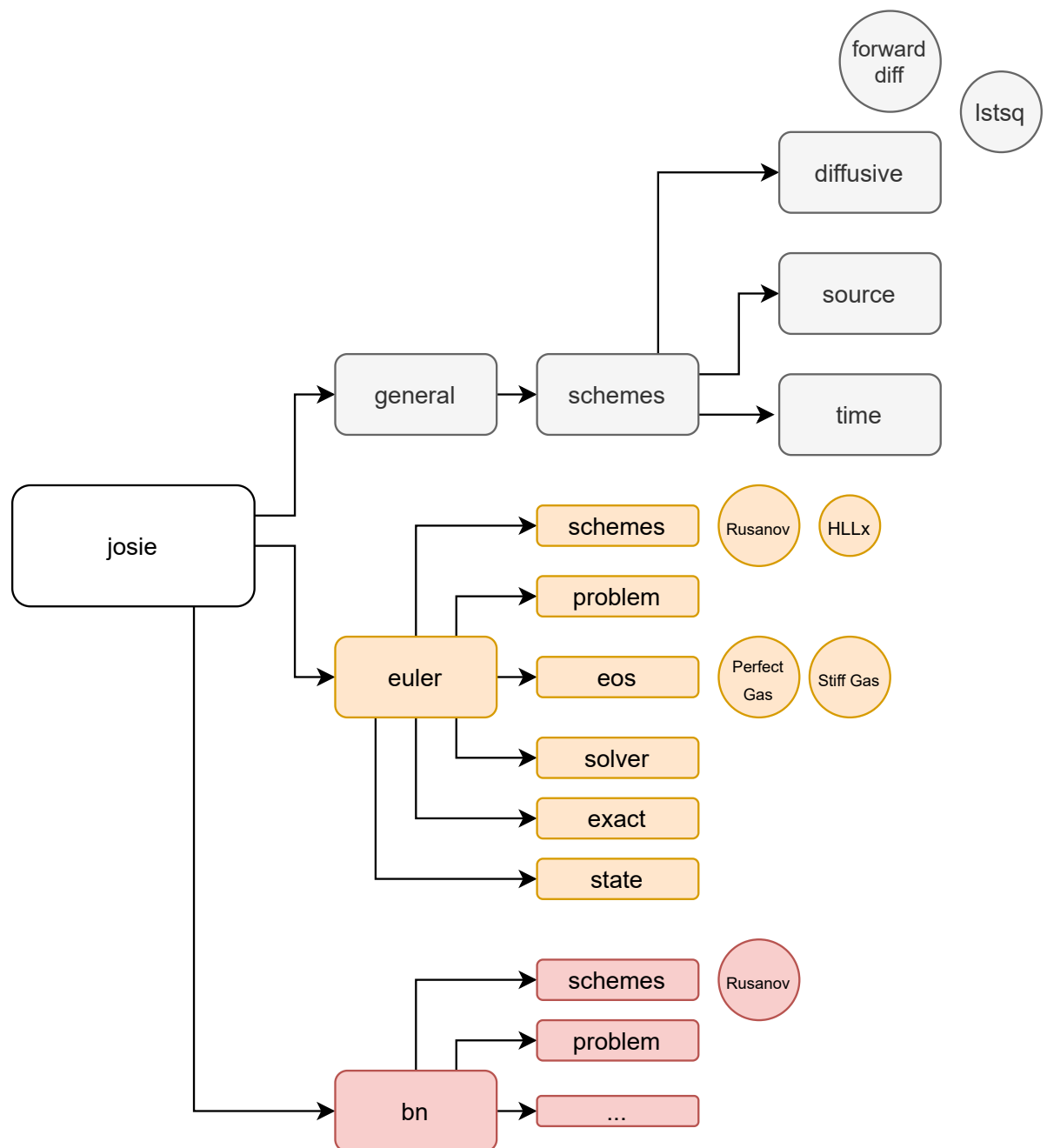


Figure 6.5: Each solver can have its own specific implementation of objects or share the `general` ones

## 6.1 josiepy: A PDE solver written in Python without sacrificing (too much) performance

```
U_L: np.ndarray, U_R: np.ndarray, c_L: np.ndarray, c_R: np.ndarray
) -> np.ndarray:
    sigma_L = np.abs(U_L) + c_L[..., np.newaxis]

    sigma_R = np.abs(U_R) + c_R[..., np.newaxis]

    # Concatenate everything in a single array
    sigma_array = np.concatenate((sigma_L, sigma_R), axis=-1)

    # And then we found the max on the last axis (i.e. the maximum value
    # of sigma for each cell)
    sigma = np.max(sigma_array, axis=-1, keepdims=True)

    return sigma

def F(self, cells: MeshCellSet, neighs: NeighboursCellSet):
    Q_L: EulerState = cells.values.view(EulerState)
    Q_R: EulerState = neighs.values.view(EulerState)

    fields = EulerState.fields

    FS = np.zeros_like(Q_L).view(EulerState)

    # Get normal velocities
    U_L = self.compute_U_norm(Q_L, neighs.normals)
    U_R = self.compute_U_norm(Q_R, neighs.normals)

    # Speed of sound
    c_L = Q_L[..., fields.c]
    c_R = Q_R[..., fields.c]

    sigma = self.compute_sigma(U_L, U_R, c_L, c_R)

    DeltaF = 0.5 * (self.problem.F(cells) + self.problem.F(neighs))
```

```

# This is the flux tensor dot the normal
DeltaF = np.einsum("...kl,...l->...k", DeltaF, neighs.normals)

# First four variables of the total state are the conservative
# variables (rho, rhoU, rhoV, rhoE)
Q_L_cons = Q_L.get_conservative()
Q_R_cons = Q_R.get_conservative()

DeltaQ = 0.5 * sigma * (Q_R_cons - Q_L_cons)

FS.set_conservative(
    neighs-surfaces[... , np.newaxis] * (DeltaF - DeltaQ)
)

return FS

```

As it is possible to see, the implementation is very dense and abstracted. EulerScheme is a child of the general mixin ConvectiveScheme that makes easier to set up an Euler problem (e.g. providing EoS implementation stored in `josie.euler.eos`). The important aspect is the implementation of the ConvectiveScheme.F method. This method acts on **all** the cells of a mesh and the corresponding neighbors (as shown graphically in Fig. 5.3), to be treated as a whole vector to benefit of the NumPy acceleration. Once the “space part” of the scheme implementation is ready, the user can actually plug it with whatever time scheme they like, exploiting multi-inheritance, as for example:

```

from josie.euler.schemes import Rusanov
from josie.general.schemes.time import RK2

class MyRusanov(Rusanov, RK2):
    pass

```

That is all it is needed to do to define a Rusanov scheme integrated in time with a RK 2.

#### 6.1.1.4 Wrapping things up, the Solver object

Once all the aspects for the definition of the case to simulate are ready, that is the physical description of the problem (section 6.1.1.1), the mesh (section 6.1.1.2), the Numerics (section 6.1.1.3), then everything is wrapped into the Solver object. As reference, let us consider

Fig. 6.4, it shows the interoperability among the different principal objects that take a role during a simulation, that are the Solver, Mesh, Scheme classes. The Problem object is embedded into Scheme in a composition approach. Two main “hooks” are available:

- `init` method that is called once at the beginning of the simulation where the Mesh and Scheme object can initialize their local data structures, storing mesh geometry information and scheme-specific data respectively, and the Solver objects applies the initial condition to the whole domain. For certain objects, the `init` method is decomposed in multiple sub-steps that are shown in Fig. 6.4 as dashed circles.
- `step` method that is called routinely at each time step. This hook is decomposed in several different sub-steps within the Scheme object, notably the `pre_accumulate` method that exposes access to all the neighbors of the mesh cells as a whole, in order to apply operations that need to access all the neighbors simultaneously (as for example the Least Square method to approximate fields gradient described in section 5.2.4.3).

#### 6.1.1.5 I/O Control

The last element we want to discuss in this section is about controlling the I/O of a test case. The user certainly does not want to save everything at each time step and, for different simulations, often different serialization strategies are required. For this reason a hierarchy of objects are stored in the `josie.io.write` package. Notably it is possible to choose a `WriteStrategy`, that imposes (tautologically) which strategy the user wants to use to serialize the results on disk, *e.g.* every  $N$  time steps or every  $d$  seconds. Those strategies are then taken as input argument by a concrete implementation of a `Writer` object, that takes care of actually serializing the data to disk. This structure allows to implement very easily and independently different serialization strategies and different serialization drivers. At the date of editing of this manuscript, the `josie.io.write.writer` module allows to serialize simulation results into XDMF (*XDMF Model and Format - XdmfWeb 2021*), memory (but be careful to not fill up your machine RAM!), and nowhere (that is always a sound option). Other drivers are indeed very easy to implement at need. As a final note we show how to finally run a simulation saving data every 0.01 s in a XDMF file,

```
from josie.io.write.strategy import TimeStrategy
from josie.io.write.writer import XDMFWriter
writer = XDMFWriter("euler.xdmf",
    TimeStrategy(dt_save=0.01, animate=True),
```

```
solver, final_time=1, CFL=0.2)
writer.solve()
```

Finally a complete test case “main” file, from top to bottom, is shown in section 6.A. The test case is a 2D jet governed by the Euler system of equations as presented in section 3.4.1.7. The code is quite verbose in order to explain each step of the code, but a complete non-trivial simulation for an Euler jet can be written in about 100 lines.

### 6.1.2 Future plans and perspectives

The main focus of *josiepy* as we already mentioned is not to replace or to compete with established **Computational Fluid Dynamics (CFD)** software that can encompass extremely varied configurations, mesh types, and industrial-specific needs. The vision behind the library is to provide an agile framework in a familiar language that allows to test the implementation of different aspects of a simulation, notably numerical schemes, Boundary Conditions, Equation of States, and so on, *without sacrificing too much performance*. That is why the envisioned roadmap is the following:

- Better integration of the NumPy API that will allow to leverage different backends, potentially on GPUs like *CuPy*; “Legate NumPy.”
- Extending the integrated mesh generator to allow the creation of block-structured meshes
- The addition of 3D capability both for mesh generation and simulation
- The addition of more advanced mesh generation algorithms, like the one based on elliptic and hyperbolic equations (see section 5.2.1)
- The addition of modern turbulence models
- Improvement of the infrastructure to facilitate interoperability with **HPC** clusters (notably an agile, automatic checkpointing ability)

The code is available with a very permissive license at the link <https://gitlab.com/rubendibattista/josiepy>, all the interested individuals are encouraged to join the GitLab project and exchange ideas and code to improve the current state of *josiepy*. The current workable tasks and encountered bugs are readily reported at <https://gitlab.com/rubendibattista/josiepy/issues> and the official documentation can be found at [josiepy.rdb.is](https://josiepy.rdb.is), where few tutorials can be executed directly on the browser thanks to Jupyter ([jupyter.org](https://jupyter.org)).



org). The ongoing work of Ait Ameer (2020); Loison (2023) will leverage the capabilities of josiépy. Most of the results presented in chapter 5 are promptly available in the repository as Jupyter notebooks or integration tests that are automatically executed at each commit, ensuring the correct functioning of the code while it gets improved. At the current date, the testing suite of the software counts 480 tests with a code coverage of 93% of the total code base, all executed automatically at each commit.

## 6.2 **Mercur(v)e: A post-processing library to perform analysis on two-phase flow simulations using differential geometry**

---

One of the main aspect discussed in this work is the sound derivation of reduced order models that can be used for situations where high-fidelity simulations are not feasible. Notably, in chapter 3 we largely discuss the different methods one can use to derive a set of governing equations that accurately represent the phenomenon of injection we are focusing here. Notably, section 3.5.3 introduces the derivation of a model that features a small-scale submodel that supposes oscillating ellipsoidal inclusions. In the formulation of the Lagrangian in facts, we faced the necessity of choosing the right set of energies that were correctly representing the phenomenon we wanted to represent. In order to do that, we decided to leverage the information a DNS could provide and we developed the Mercur(v)e library in order to perform the post-processing of these high-fidelity simulations and extract geometrical information from the triangulated interface. The details are thoroughly explained in chapter 4 in terms of formal algorithms and related results. The library offers a higher-level abstractions on top of the VTK graphic toolkit (Will Schroeder et al. 2006), therefore the underlying data structures are for the most part borrowed from VTK.

### 6.2.1 The architecture of the code

Mercur(v)e can actually be used in two ways: as a standalone executable (named hgve), or as a library to be used in your own code. This double mode allows the software to be used by user that are only interested to the post-processing functionality offered by the software via the standalone executable, while providing full access to the API to people interested to customize a specific behavior. The general architecture (that is not listing specific low-level data structures) is shown in Fig. 6.6. We will start our presentation from the entry point, that

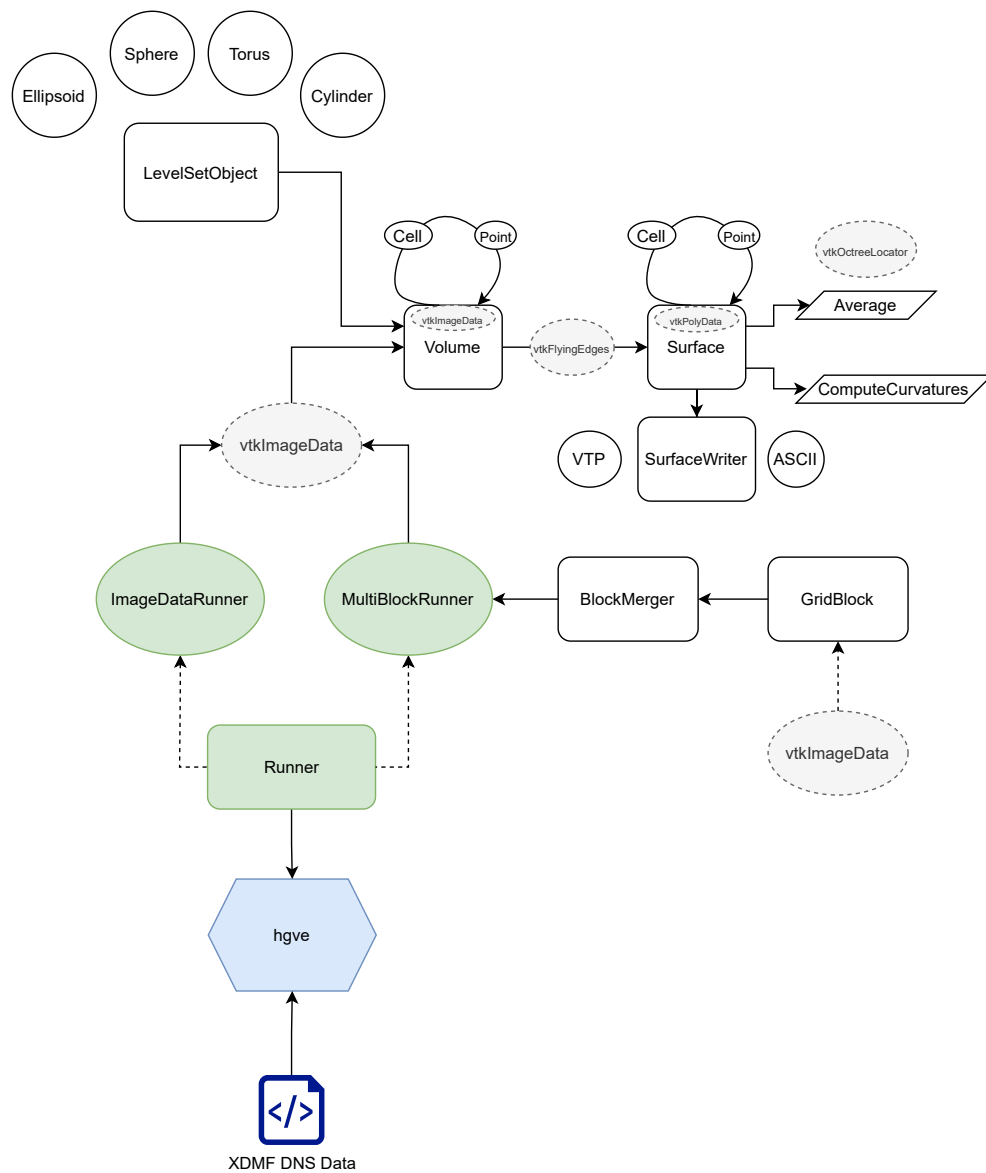


Figure 6.6: Mercur(v)e architecture

is the hgve executable.

### 6.2.1.1 The standalone executable

Usage:

```
hgve [--mirror=<axis> ...] [options] <file>
```

Options:

-D, --output_dir=<dir>	Directory where to store output files [default: .]
-h, --help	Print this help message
-a, --average=<timesDx>	Average points within the radius equal to <timesDx> times the spacing of the original grid (computed as the average of the three axes)
-s, --write_stats	Write point data to file
--nooverlap	Flag to set for a multiblock input file in order to remove first layer of points that are overlapping
--merge-only	Execute the merge step only
--mirror=<axis>	Before computing the curvature, mirror the along the specified axis. Possible values of <axis> are `+X`, `-X`, `+Y`, `-Y`, `+Z`, `-Z`, the option to mirror along multiple axis (in order).
--sanitize	True if you need to sanitize the resulting triangulated surface. Check <code>Surface::Sanitize</code> method for more information

The hgve program exposes most of the library functionality via an easy-to-use [Command Line Interface \(CLI\)](#). It takes as input an XDMF (*XDMF Model and Format* - [XdmfWeb 2021](#)) file containing the level-set field computed via the DNS software (*e.g.* ARCHER [Vaudor et al. 2017](#)), on a Cartesian mesh with uniform spacing<sup>4</sup>. Once the file is read (using VTK I/O classes), a polymorphic behavior is encountered: based on the size of the simulation, the resulting data can be

---

<sup>4</sup>This is not a hard requirement. Strictly speaking the input data can be whatever is supported by the VTK iso-surfacing routines

included in just a single rectangular grid, or in a set of blocks, each one being a Cartesian grid. In order to resolve correctly the problem, the executable dispatches the decision to the Runner object that based on the nature of the input files, can actually exploit the `MultiBlockRunner` to perform the merging of the blocks with the algorithm presented in algorithm 4.A.1. The final outcome of the Runner execution is then a single rectangular block containing the level-set field, and stored in a `vtkImageData` low-level data structure provided by VTK. This low-level structure is wrapped by the Volume object.

### 6.2.1.2 The Volume and Surface objects

The Volume objects wraps the rectangular grid exposing methods to retrieve general statistics (as the total number of points and the spacing), but most of all it allows the triangulation of the interface between the two phases present in the provided input file. `Volume::Triangulate` is the method used to obtain the resulting surface. At the current date, the iso-surfacing algorithm used to obtain the triangulated surface representing the interface is hardcoded to be the Flying Edges algorithm (William Schroeder et al. 2015), since from our experimentation, it is the one providing the best results with the most efficient computation. A further modularization of the code might easily allow to use different algorithms like the MC (Lorensen and Cline 1987; Chernyaev 1995). Once the triangulation is completed, a Surface object is returned.

The Surface object is a high-level wrapper over the VTK `vtkPolyData` container. This low-level container can store a polygonal surface connectivity or relatively arbitrary form, but in the context of this application, a triangulated surface is employed. This implementation also exposes a polymorphic iterator interface that allows to loop over the Points,

```
// Get norm of the position vector associated
// to the iterated point
for (auto p : surface.IterateOver<Point>()) {
    auto norm = p.Norm();
}
```

respectively the Cells, of the triangulated surface very easily.

```
// Sum cell areas
double A = 0;
for (auto c : surface.IterateOver<Cell>()) {
    A += c.Area();
}
```

In addition, the object encapsulates the functionality to actually perform the computation using the “1-ring averaging algorithm” discussed in chapter 4 via the `Surface::ComputeCurvatures` method. It also allows the topology-preserving averaging we present in section 4.4.2.1 via the `Surface::Average`, that takes as input the radius around which we are keen on performing the averaging. Both the `Surface` and `Volume` objects expose a way to serialize their data on disk, notably in VTK format. The `Surface` object can be also serialized in a human-readable format via the `SurfaceASCIIWriter` helper (opposed to the `SurfaceVTKWriter` for VTK format serialization) and it is easily extensible.

A `Volume` cannot only be created reading the output data of an actual simulation. Sometimes it could be useful to have the possibility of creating artificial level-set fields that yield interesting surface objects. Actually a distance function like a level-set embodies interesting properties that allow a sort of “set operations” like addition or subtraction between basic, canonical, objects described by level-sets. *Mercur(v)e* offers an easy way of performing those operations to create arbitrary objects to experiment with, *i.e.* the `LevelSetObject`.

### 6.2.1.3 Playing with distance functions to create artificial surfaces

The functionality described in this section is heavily based on the classical work in the framework of Computer Graphics and rendering. We shall not discuss the mathematics behind the composition of distance functions, the interested reader can refer to *Inigo Quilez :: Fractals, Computer Graphics, Mathematics, Demoscene and More (2018)* for an hands-on description. The `LevelSetObject` is the abstract class that overrides the `+`, `-`, `^` operators to allow respectively the union, difference and intersection of other generic `LevelSetObject`s and the additional `LevelSetObject::Elongate` transform that creates derived objects via elongation of the basic ones. The concrete implementations of `LevelSetObject` are topological objects that feature analytical or easily approximable expressions for their level-set function, *i.e.* `LevelSetSphere`, `LevelSetEllipsoid`, `LevelSetCylinder`, `LevelSetTorus`. Each individual object can then be unioned, subtracted, intersected or elongated to obtain complex objects with composition of transforms. As an example, we can create a “Mickey Mouse”-shaped object (Fig. 6.7) and compute curvatures on it with the code shown in section 6.B.

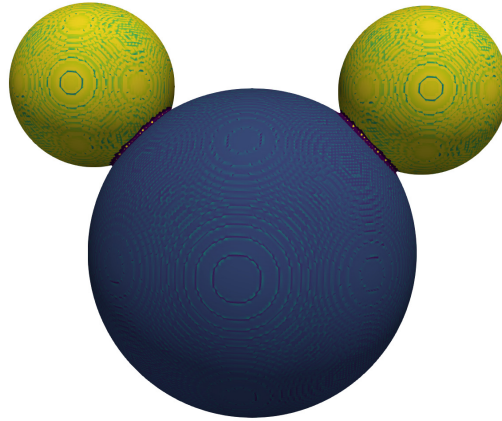



Figure 6.7: A “Mickey Mouse” object made by the composition of level-set spheres. The  maps the Gauss curvature value

### 6.2.2 Future plans and perspectives

The `Mercur(v)e` library was an invaluable tool that allowed the sound closure of the model described in section 3.6. Nonetheless, the library can still be advanced to become a complete post-processing tool for two-phase flow DNS and beyond. The following point of improvement can be identified in the short and medium term:

- Making the curvatures computation algorithms pluggable: at the current date, the only algorithm that is possible to use is the one described in chapter 4 that exploits the “1-ring averaged values” over the surface. Other techniques are indeed available (see for example Bermejo-Moreno and Pullin (2008); Bermejo-Moreno, Pullin, and Horiuti (2009)). It would be highly beneficial to allow the user to easily choose among a set of possibilities via the implementation of a pluggable interface like the one exposed by the `Surface` object for the serialization to disk
- Allowing the curvatures computation on unclosed objects: currently, the computation of the curvatures via the algorithms described in chapter 4 can only be performed on closed or symmetrical objects. In order to allow the post-processing of unclosed surfaces, a fix for those calculations is required
- Performing the curvature computation at runtime during a simulation and not only as post-processing: this point of improvement actually requires the previous one to be implemented first, since a DNS simulation that has an actual value must be performed on a

partitioned domain in parallel, each block may contain an unclosed section of the interface. The interest of performing a runtime computation for the curvatures is to benefit of the topology-preserving averaging kernel discussed in section 4.4.2.1 to compute the surface tension in a simulation. In the ARCHER code for example, in the parts of the domain where strong topological variations undergo, the computed values of the mean curvatures that are needed for the surface tension estimation can spike to extreme values, therefore a manual and somehow arbitrary threshold is applied. The averaging kernel we presented in this thesis might lift this restriction acting as a regularization strategy in these difficult part of the domain. This runtime curvature estimation however may require a higher computing cost associated to the triangulation, together with the need of projecting back the computed values of curvature to the rectangular volumetric mesh, maintaining the topological invariance.





# Appendices

## 6.A

A “main” file to run an Euler jet simulation with  
josiepy

---

```
import numpy as np

from josie.bc import Dirichlet, Neumann, NeumannDirichlet
from josie.boundary import Line
from josie.euler.eos import PerfectGas
from josie.euler.solver import EulerSolver
from josie.euler.state import Q

from josie.mesh import Mesh
from josie.mesh.cell import SimpleCell

# Definition of the domain
left = Line([0, 0], [0, 1])
bottom = Line([0, 0], [0.25, 0])
right = Line([0.25, 0], [0.25, 1])
top = Line([0, 1], [0.25, 1])

# Choose your gas EOS
eos = PerfectGas(gamma=1.4)

# Conditions of the inlet jet and its position
JET_CENTER = 0.5
JET_RADIUS = 0.05
```

```
# Inlet
U_JET = 1
V_JET = 0
RHO_JET = 1000
P_JET = 1e3

RHOe_JET = eos.rhoe(RHO_JET, P_JET)
E_JET = RHOe_JET / RHO_JET + 0.5 * (U_JET ** 2 + V_JET ** 2)
C_JET = eos.sound_velocity(RHO_JET, P_JET)
Q_JET = Q(
    RHO_JET,
    RHO_JET * U_JET,
    RHO_JET * V_JET,
    RHO_JET * E_JET,
    RHOe_JET,
    U_JET,
    V_JET,
    P_JET,
    C_JET,
)

# Conditions for the rest of the domain at t=0
U_INIT = 0
V_INIT = 0
RHO_INIT = RHO_JET / 1000
P_INIT = P_JET
RHOe_INIT = eos.rhoe(RHO_INIT, P_INIT)
E_INIT = RHOe_INIT / RHO_INIT + 0.5 * (U_INIT ** 2 + V_INIT ** 2)
C_INIT = eos.sound_velocity(RHO_INIT, P_INIT)

Q_INIT = Q(
    RHO_INIT,
    RHO_INIT * U_INIT,
    RHO_INIT * V_INIT,
    RHO_INIT * E_INIT,
```

```

    RHOe_INIT,
    U_INIT,
    V_INIT,
    P_INIT,
    C_INIT,
)

# Zero-gradient value
dQ = np.zeros(len(Q.fields)).view(Q)

# Partition function that sets the part of the left boundary on which to impose
# the inlet conditions
def partition_fun(centroids: np.ndarray):
    yc = centroids[..., 1].flatten()

    # Partition cells of the inlet
    idx = np.where((yc - JET_CENTER) ** 2 < JET_RADIUS ** 2)

    return idx

# Assign BC to boundaries
left.bc = NeumannDirichlet(
    dirichlet_value=Q_JET, neumann_value=dQ, partition_fun=partition_fun
)
top.bc = Dirichlet(Q_INIT)
right.bc = Neumann(dQ)
bottom.bc = Dirichlet(Q_INIT)

# Generate your mesh
mesh = Mesh(left, bottom, right, top, SimpleCell)
mesh.interpolate(25, 100)
mesh.generate()
mesh.write("mesh.xdmf")

```

```
# Definition of the Scheme
from josie.euler.schemes import Rusanov
from josie.general.schemes.time import ExplicitEuler

# Choose your convective scheme and time integration via inheritance
class MyScheme(Rusanov, ExplicitEuler):
    pass

scheme = MyScheme(eos)

# Define the Solver object
solver = EulerSolver(mesh, scheme)

# Init function that initializes the domain at t=0
def init_fun(solver):
    solver.values[...] = Q_INIT

# Perform the initialization
solver.init(init_fun)

from josie.io.write.strategy import TimeStrategy
from josie.io.write.writer import XDMFWriter

# Run the simulation and write the results to file every 0.01s of simulated
# time
writer = XDMFWriter(
    "euler.xdmf",
    TimeStrategy(dt_save=0.01, animate=True),
    solver,
    final_time=1,
    CFL=0.2,
)
```

```
writer.solve()
```

## 6.B

### A “Mickey Mouse” shaped object obtained with the composition of basic level-set objects using `Mercur(v)e`

---

```
#include "CompositeLevelSetObject.h"
#include "LevelSetSphere.h"
#include "Volume.h"

using namespace hgve;

int main(int argc, char *argv[]) {
    Volume domain(250, 250, 250, SimpleVector(4, 4, 4));

    // Generate the central sphere
    double radius = 1.0;
    SimpleVector center(2, 2, 2);
    LevelSetSphere sphere(radius, center);

    // Generate the ears spheres
    LevelSetSphere ear1(radius / 2, SimpleVector(1, 1, 2));
    LevelSetSphere ear2(radius / 2, SimpleVector(3, 1, 2));

    // Combine objects
    auto mickeyMouse = sphere + ear1 + ear2;

    // Materialize in the domain
    domain.AddLevelSetObject(mickeyMouse);

    // Save the level-set fields
    domain.Write("mickey-mouse.vti");

    // Triangulate the domain
```

```

    auto surface = domain.Triangulate();

    // Compute curvatures and save
    surface->ComputeCurvatures();
    surface->Write("mickey-mouse.vtp");

    // Test Gauss Bonnet
    double GA = 0;
    for (auto p : surface->IterateOver<Point>()) {
        auto curv = p.OneRingProperties();
        double G = curv.G();
        double A = curv.A();
        GA += G * A;
    }

    std::cout.precision(std::numeric_limits<double>::max_digits10);
    std::cout << "Gauss Bonnet: " << GA / 4 / vtkMath::Pi();
}

```

# 7

## General conclusions and future perspectives

The endeavor of this thesis has been focused on improving the current state of the art regarding the modeling and simulation of two-phase systems, with a particular interest towards space propulsion systems. The general picture we aim at providing at the end of this work is the one of a mathematically sound methodology that allows the derivation of two-phase flow models that have desirable mathematical properties, notably a hyperbolic convective structure and, where needed, source terms that respect the second principle of thermodynamics dissipating entropy in a correct manner. This modeling strategy is then coupled with a flexible computing environment in which Free and Open Source Software (FOSS) libraries act as useful tools that help the modeler to close their modeling assumptions and also to benchmark the most suitable numerical schemes for the specific model under investigation. The final, global target, of this effort is the creation of an accessible ecosystem to perform accurate predictions of flow behavior for systems that are intrinsically complex and feature multiple scales. In facts, the major difficulty that makes performing accurate simulations is this inherent multiscale nature and the necessity of being able to simulate both the separated phase regime and the disperse phase regime, that are often co-existent in a single case. The current state-of-the art strategies that account only for the disperse phase regime, neglecting the correct resolution of the liquid film that actually produces the final spray droplet distribution fail at providing a satisfying outcome. The approach we presented in this work is therefore centered at providing the building elements of a unified modeling approach that allows the resolution of the liquid film close to the injector, together with an Eulerian modeling of the spray. This objective has been tackled on different levels:

- On the modeling side, we have proposed a unified approach based on a variational mathematical tool called Stationary Action Principle (SAP). The current presentation of the method is the coronation of a lot of years of investigation and it stands on the shoulder of other studies performed within the research team, notably [Druil \(2017\)](#); [Essadki \(2018\)](#); [Cordesse \(2020\)](#). The main idea of the method is based on the injection of additional ge-

ometrical properties such as the interfacial area density and the average mean and Gauss curvatures into the set of fields featured in the modeling equations. These properties are injected in the set of equations assuming a specific behavior of the unresolved small scales of the interface dynamics, as for example the oscillation of an ellipsoidal inclusion at constant volume or its normal pulsation complying with the Weyl's tube formula [Weyl \(1939\)](#). The behavior is translated in a set of small scale energies that are featured in the global Lagrangian of the system and their evolution is constrained by specific governing equations. The optimization of the Hamiltonian Action, which is the integral over the available phase space of the chosen Lagrangian functional, under the constraints as for example the conservation of the mass, phase mass fractions or a specific governing equation for the interfacial density area, yields the set of the governing equations. The novelty of this work resides in the possibility of addressing the interfacial area density equation, the source terms of which are derived using averaging techniques and sometimes closed with empirical correlations drawn from experiments or physical considerations, as described in [Drew \(1990\)](#); [Morel et al. \(1999\)](#); [Daniel Lhuillier \(2004\)](#). In particular we provide a possible methodology to consider these source terms in the framework of the [SAP](#), we introduce them in a way that complies with the correct dissipating structure mandated by the second principle of thermodynamics.

- Sometimes, as in the case of the oscillating ellipsoidal inclusion in the context of this thesis, the right modeling assumptions to be taken are not straightaway clear. In order to validate a certain choice of Lagrangian small scale energies we leverage the higher density of information contained in high-fidelity simulations such as [DNS](#) to guide the choice. In order to do that, we have developed a library called `Mercur(v)e` following the tracks of [Essadki \(2018\)](#) the aim of which is to offer an easy way to implement a post-processing procedure based on the triangulation of the interface between the two phases and the associated topological invariant calculation of the local and averaged surface related quantities as the interfacial area density, mean and Gauss curvatures. The discretized estimation of these quantities is performed with an algorithm that conserves the Gauss-Bonnet theorem. Moreover, for situations in which the interface undergoes strong topological deformations, we also provide an averaging kernel that smooths out non-physical peaks of values for the geometrical fields, always respecting the Gauss-Bonnet theorem. This framework allowed in the context of this thesis to perform energetic choices on the modeling side that lead to the development of a model that partially overcomes the limitations of the model of [Drui \(2017\)](#), extending it for objects homeomorphic to a sphere and not only strictly spherical. Also, compared to more classical



methods that do not enforce topological invariance, such as for example the curvature estimation routine of the [DNS](#) code ARCHER ([Vaudor et al. 2017](#)), the approach presented in this work show higher accuracy at same level of refinement of the mesh.

- Reduced order models are developed to be then simulated in [HPC](#) codes and used in industry to drive design choices. Sometimes these industrial codes can be complex to manipulate and can translate in long development times to implement and bug fix a new feature. In this thesis we propose a [FOSS](#) software library aimed at simulating [PDEs](#)-driven problems named [josiepy](#), which is based on the [API](#) of familiar language like Python and its ecosystem of high-performance extensions like NumPy ([Harris et al. 2020](#); [Bauer and Garland 2019](#)). The dynamic nature of the language speeds up the development cycle and it allows a more comfortable test of innovative numerical schemes, [BCs](#) or conservation systems, without sacrificing (too much) performance thanks to the use of high-performance low-level compiled libraries that are accessible from Python via some form of binding. We verify the correctness of the results obtained using this library on classical testing cases for different systems and we also showcase its capabilities on the implementation of a system featuring the equation of the interfacial area density, with first and second order MUSCL-Hancock schemes.

The work we performed undoubtedly opens up different future perspectives for improvement. Most of these perspectives are discussed in fair detail in each specific chapter of the thesis. We will limit ourselves to repeat them in a prosaic fashion here:

- For what concerns the unified modeling framework we put in place, we succeeded in extending the small scale scenarios we are able to account for with the [SAP](#); still, more work is indeed necessary to achieve a reasonable accuracy for industrial cases. Notably, the highest order objects we are able to account are inclusions homeomorphic to spheres. After primary and secondary atomization, a non-negligible percentage of objects is non-homeomorphic to a sphere, therefore our modeling assumptions need further extension. The multi-dispersion nature of the disperse phase regime, in which distribution of droplets are present in each point of the domain, needs to be tackled with additional care. The possibility of exploiting geometrical fields such as the curvatures as moments of a [NDF](#) from which, through a realizable quadrature, is a promising path for the future. Future work to integrate the ideas of [Drew \(1990\)](#); [Vallet and Borghi \(1999\)](#); [Morel \(2015\)](#) in the variational [SAP](#) method is also a path of advancement. Both directions have been taken over by Loison in his PhD at CMAP ([Loison 2023](#)).

- The computation of the geometrical properties on triangulated interfaces we introduced and implemented in the library `Mercur(v)` is currently used only as a post-processing tool. But the possibility of employing the smoothing kernel at runtime, for example during an ARCHER simulation, might open interesting perspectives on the effect of the method used to estimate mean curvature (and then surface tension). In order to be able to perform such calculations, the algorithm needs to be corrected to operate also on un-closed objects surfaces. Also, the values of the curvatures are needed on the original volumetric mesh on which the original level set values are stored, therefore a projection method needs to be built which is able to keep the topological invariance benefits provided by the calculation on triangulated interfaces.
- On the numerical side, even if the `josiepy` library is not aimed at replacing multi-physics industrial codes, the recent advances on heterogeneous computing libraries featuring data structures that are storable on heterogeneous architectures made of CPUs and GPUs ([Bauer and Garland 2019](#)) open up interesting possibilities. A multi-block 3D PDE solver capable running on multiple GPUs is a target that is envisioned in the near future for the library. In parallel, the easy to use API we worked hard to deliver might also lead to new outcomes in the framework of numerical schemes and models simulation that could benefit also the higher level multi-physics computing platforms. So far, `josiepy` library has been chosen in the PhD of A. Loison, in the post-doctoral work of K Ait-Ameur, as well as by our colleagues at ONERA in the DMPE department in collaboration with VKI and CMAP (future PhD of W. Haegeman).

# Acronyms

ABC Abstract Base Class. 230

AI Artificial Intelligence. 143, 144, 222

API Application Programming Interface. 11, 33, 109, 221–223, 226, 227, 230, 239, 255

B-N Baer-Nunziato. xv, 161, 198–200, 220

BC Boundary Condition. 190–192, 198, 208, 210, 215, 228, 229, 255

CD Continuous Development. 12, 34

CFD Computational Fluid Dynamics. 238

CI Continuous Integration. 12, 34, 200

CLI Command Line Interface. 241

CLSVOF Coupled Level Set/Volume of Fluid. 109

CSM Apollo command and service module. 1, 17

DG Discontinuous Galerkin. 160, 220

DNS Direct Numerical Simulation. xvii, xxiii, 8, 10, 11, 13, 14, 30, 32–35, 72, 74, 103, 110, 114, 121, 137, 138, 141, 142, 239, 241, 244, 254, 255

EoS Equation of State. xv, 62, 82, 157, 168–170, 172, 189, 200, 209, 210, 216, 225, 236

FDM Finite Difference Method. 143

FE Flying Edges. 108, 109

FEM Finite Element Method. 143

FOSS Free and Open Source Software. 14, 35, 137, 209, 221, 255

FVM Finite Volume Method. 35, 143–146, 150, 156, 161, 166, 182, 221

H<sub>2</sub>O<sub>2</sub> Hydrogen Peroxide. 20

HNO<sub>3</sub> Nitric Acid. 17, 20

HPC High Performance Computing. 12, 34, 221, 238, 255

HRE Hybrid Rocket Engine. xi, 3, 21, 22, 28

HTPB Hydroxyl-terminated polybutadiene. 20, 21

IP Intellectual Property. 21

IVP Initial Value Problem. 155

LB Laplace-Beltrami. 110

LEM Apollo Lunar Excursion Module. 1, 17

LH<sub>2</sub> Liquid Hydrogen. 17

LHS Left-Hand Side. 85

LOX Liquid Oxygen. 17, 20, 21

LRE Liquid Rocket Engine. xi, 17, 20–22, 24

MC Marching Cubes. 109, 110, 118, 141, 242

MoL Method of Lines. 223

MoM Method of Moments. 31, 41

MTBF Mean time between failures. 5, 24

NDF Number Density Function. xi, 29, 31, 119, 121, 122, 124, 125, 137, 255

NS Navier-Stokes. 8, 30, 41, 189, 208

NTO Dinitrogen Tetroxide N<sub>2</sub>O<sub>4</sub>. 20

NUMA Non-Uniform Memory Access. 221

ODE Ordinary Differential Equation. 143, 156, 157, 170, 171, 175

PDE Partial Differential Equation. 35, 143, 144, 146–148, 157, 166, 168, 188, 221–223, 228, 255, 256

PDF Probability Density Function. 65, 114

RHS Right-Hand Side. 78, 84, 164, 166

RK Runge-Kutta. 166, 175, 236

RP Riemann Problem. ix, 156–158, 167, 170–173, 175, 189, 190, 198–200

RUD Rapid Unscheduled Disassembling. 21, 24

SAP Stationary Action Principle. 32, 35, 39, 41, 51, 55, 56, 60, 61, 63, 65, 67, 79, 81, 129, 137, 220, 254, 255

SRM Solid Rocket Motor. 20, 21

SSP Strong Stability Preserving. 167

TFI Trans-Finite interpolation. 146, 148, 222

TVD Total variation diminishing. 159, 160, 167

UDMH Unsymmetrical dimethylhydrazine. 17

VOF Volume Of Fluid. 8, 30



# Bibliography

- Ait Ameer, Katia (Nov. 9, 2020). “Contributions to the Parallel Simulation of Two-Phase Flows and Analysis of Finite Volume Schemes on Staggered Grids.” PhD thesis. Sorbonne Université. URL: <https://tel.archives-ouvertes.fr/tel-03191320>.
- Alighieri, Dante (n.d.). *The Divine Comedy of Dante Alighieri*/. 1265-1321. Houghton Mifflin.
- Baer, M. R. and J. W. Nunziato (1986). “A Two-Phase Mixture Theory for the Deflagration-to-Detonation Transition (DDT) in Reactive Granular Materials.” In: *International journal of multiphase flow* 12.6, pp. 861–889.
- Bauer, Michael and Michael Garland (Nov. 17, 2019). “Legate NumPy: Accelerated and Distributed Array Computing.” In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC ’19: The International Conference for High Performance Computing, Networking, Storage, and Analysis. Denver Colorado: ACM, pp. 1–23. ISBN: 978-1-4503-6229-0. URL: <https://dl.acm.org/doi/10.1145/3295500.3356175>.
- Bedford, A. (1985). *Hamilton’s Principle in Continuum Mechanics*. Pitman Advanced Publishing Program. 126 pp. ISBN: 978-0-273-08730-4.
- Bellotti, Thomas et al. (Mar. 2021). “Multidimensional Fully Adaptive Lattice Boltzmann Methods with Error Control Based on Multiresolution Analysis.” working paper or preprint. URL: <https://hal.archives-ouvertes.fr/hal-03158073>.
- Berdichevsky, Victor (2009). *Variational Principles of Continuum Mechanics*. Interaction of Mechanics and Mathematics. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-88466-8.
- Bermejo-Moreno, Iván and D. I. Pullin (May 2008). “On the Non-Local Geometry of Turbulence.” In: *J. Fluid Mech* 603. ISSN: 0022-1120, 1469-7645.
- Bermejo-Moreno, Iván, D. I. Pullin, and Kiyosi Horiuti (Feb. 2009). “Geometry of Enstrophy and Dissipation, Grid Resolution Effects and Proximity Issues in Turbulence.” In: *J. Fluid Mech.* 620, p. 121. ISSN: 0022-1120, 1469-7645.
- Bi, Désir-André Koffi et al. (2021). “A Review of Geometrical Interface Properties for 3D Front-Tracking Methods.” In: *Turbulence and Interactions*. Ed. by Michel Deville et al.

- Vol. 149. Notes on Numerical Fluid Mechanics and Multidisciplinary Design. Cham: Springer International Publishing, pp. 144–149. ISBN: 978-3-030-65819-9. URL: [http://link.springer.com/10.1007/978-3-030-65820-5\\_15](http://link.springer.com/10.1007/978-3-030-65820-5_15).
- Blanchard, Ghislain, Davide Zuzio, and Philippe Villedieu (May 2016). “A Large Scale Multi-Fluid/Dispersed Phase Approach for Spray Generation in Aeronautical Fuel Injectors.” In: *ICMF 2016*. FLORENCE, Italy. URL: <https://hal.archives-ouvertes.fr/hal-01441814>.
- Bobenko, Alexander I. and Peter Schröder (2005). “Discrete Willmore Flow.” In: *ACM SIGGRAPH 2005 Courses on - SIGGRAPH '05*. ACM SIGGRAPH 2005 Courses. Los Angeles, California: ACM Press, p. 5. URL: <http://portal.acm.org/citation.cfm?doid=1198555.1198664>.
- Bouchut, François (2004). *Nonlinear Stability of Finite Volume Methods for Hyperbolic Conservation Laws: And Well-Balanced Schemes for Sources*. Frontiers in Mathematics. Birkhäuser Basel. ISBN: 978-3-7643-6665-0.
- Brackbill, J. U., Douglas B. Kothe, and Charles Zemach (1992). “A Continuum Method for Modeling Surface Tension.” In: *Journal of computational physics* 100.2, pp. 335–354.
- Brent, Richard P. (Apr. 17, 2013). *Algorithms for Minimization Without Derivatives; Dover Books on Mathematics*. Illustrated edition. Mineola, N.Y: Dover Publications. 206 pp. ISBN: 978-0-486-41998-5.
- Butcher, J. C. (Mar. 1, 1996). “A History of Runge-Kutta Methods.” In: *Applied Numerical Mathematics* 20.3, pp. 247–260. ISSN: 0168-9274.
- Cahn, John W. and John E. Hilliard (1958). “Free Energy of a Nonuniform System. I. Interfacial Free Energy.” In: *The Journal of chemical physics* 28.2, pp. 258–267.
- Candel, Sebastien and Thierry Poinot (Mar. 1, 1990). “Flame Stretch and the Balance Equation for the Flame Area.” In: *Combustion Science and Technology* 70.1-3, pp. 1–15. ISSN: 0010-2202.
- Canu, Romain, Christophe Dumouchel, et al. (Sept. 6, 2017). “Where Does the Drop Size Distribution Come From?” In: *ILASS-Europe 2017*. URL: <https://hal-normandie-univ.archives-ouvertes.fr/hal-01621288>.
- Canu, Romain, Stefano Puggelli, et al. (Oct. 1, 2018). “Where Does the Droplet Size Distribution Come From?” In: *International Journal of Multiphase Flow* 107, pp. 230–245. ISSN: 0301-9322.
- Carmo, Manfredo P. do (Dec. 14, 2016). *Differential Geometry of Curves and Surfaces: Revised and Updated Second Edition*. Courier Dover Publications. 529 pp. ISBN: 978-0-486-80699-0.



- Castro, C. E. and E. F. Toro (2006). “A Riemann Solver and Upwind Methods for a Two-Phase Flow Model in Non-Conservative Form.” In: *International journal for numerical methods in fluids* 50.3, pp. 275–307.
- Chalons, Christophe et al. (2011). “Large Time-Step Numerical Scheme for the Seven-Equation Model of Compressible Two-Phase Flows.” In: *Finite Volumes for Complex Applications VI Problems & Perspectives*. Ed. by Jaroslav Fořt et al. Vol. 4. Springer Proceedings in Mathematics. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 225–233. ISBN: 978-3-642-20670-2. URL: [http://link.springer.com/10.1007/978-3-642-20671-9\\_24](http://link.springer.com/10.1007/978-3-642-20671-9_24).
- Chandrasekhar, S (1961). *Hydrodynamic and Hydromagnetic Stability*. International Series of Monographs on Physics.
- Chandrasekhar, Subrahmanyan (Jan. 1, 1981). *Hydrodynamic and Hydromagnetic Stability*. Courier Corporation. 720 pp. ISBN: 978-0-486-64071-6.
- Chantepredrix, Guilhem, Philippe Villedieu, and Jean-Paul Vila (2002). “A Compressible Model for Separated Two-Phase Flows Computations.” In: *Volume 1: Fora, Parts A and B*. ASME 2002 Joint U.S.-European Fluids Engineering Division Conference. Montreal, Quebec, Canada: ASME, pp. 809–816. ISBN: 978-0-7918-3615-6. URL: <http://proceedings.asmedigitallcollection.asme.org/proceeding.aspx?articleid=1577348>.
- Chernyaev, Evgeni (1995). *Marching Cubes 33: Construction of Topologically Correct Isosurfaces*.
- Chevalier, Quentin (2019). *Rapport de Stage : MODÉLISATION THERMODYNAMIQUE D’UNE INTERFACE LIQUIDE-GAZ*.
- Cockburn, Bernardo and Chi-Wang Shu (2001). “Runge–Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems.” In: *Journal of scientific computing* 16.3, pp. 173–261.
- Colella, Phillip and Harland M Glaz (June 1985). “Efficient Solution Algorithms for the Riemann Problem for Real Gases.” In: *Journal of Computational Physics* 59.2, pp. 264–289. ISSN: 00219991.
- Coquel, Frédéric, Thierry Gallouët, et al. (Apr. 30, 2002). “Closure Laws for a Two-Fluid Two-Pressure Model.” In: *Comptes Rendus Mathématique* 334.10, pp. 927–932. ISSN: 1631-073X.
- Coquel, Frédéric, Jean-Marc Hérard, and Khaled Saleh (2012). “A Splitting Method for the Isentropic Baer-Nunziato Two-Phase Flow Model.” In: *ESAIM: Proceedings*. Vol. 38. EDP Sciences, pp. 241–256.
- (Nov. 2016). “A Positive and Entropy-Satisfying Finite Volume Scheme for the Baer–Nunziato Model.” In: *Journal of Computational Physics* 330, pp. 401–435. ISSN: 00219991.

- Coquel, Frédéric, Jean-Marc Hérard, Khaled Saleh, and Nicolas Seguin (2014). “A Robust Entropy- Satisfying Finite Volume Scheme for the Isentropic Baer- Nunziato Model.” In: *ESAIM: Mathematical Modelling and Numerical Analysis* 48.1, pp. 165–206.
- Cordesse, Pierre (June 2020). “Contribution to the Study of Combustion Instabilities in Cryotechnic Rocket Engines: Coupling Diffuse Interface Models with Kinetic-Based Moment Methods for Primary Atomization Simulations.” PhD thesis. Université de Paris-Saclay.
- Cordesse, Pierre, Ruben Di Battista, Quentin Chevalier, et al. (2020). “A Diffuse Interface Approach For Disperse Two-Phase Flows Involving Dual-Scale Kinematics Of Droplet Deformation Based On Geometrical Variables.” In: *ESAIM: Proceedings*, p. 22. URL: <https://hal.archives-ouvertes.fr/hal-02879950v1>.
- Cordesse, Pierre, Ruben Di Battista, Florence Drui, et al. (2020). “Derivation of a Two-Phase Flow Model with Two-Scale Kinematics, Geometric Variables and Surface Tension Using Variational Calculus.” In: *Proceedings of the NASA Summer Program*. Nasa Technical Memorandum. URL: <https://hal.archives-ouvertes.fr/hal-02336996>.
- Cordesse, Pierre, Ruben Di Battista, Samuel Kokh, et al. (May 19–24, 2019). “Derivation of a Two-Phase Flow Model with Two-Scale Kinematics and Surface Tension by Means of Variational Calculus.” In: 10th International Conference on Multiphase Flow. Rio de Janeiro. URL: <https://hal.archives-ouvertes.fr/hal-02194951>.
- Cordesse, Pierre and Marc Massot (May 2020). “Entropy Supplementary Conservation Law for Non-Linear Systems of PDEs with Non-Conservative Terms: Application to the Modelling and Analysis of Complex Fluid Flows Using Computer Algebra.” In: *Communications in Mathematical Sciences* 18.2, pp. 515–534.
- Cordesse, Pierre, Angelo Murrone, and Marc Massot (2018). “Coupling a Hierarchy of Diffuse Interface Models with Kinetic-Based Moment Methods for Spray Atomization Simulations in Cryogenic Rocket Engines.” In: 14th International Conference on Liquid Atomization & Apray Aystems (ICLASS).
- Crouzet, Fabien et al. (July 2013). “Approximate Solutions of the Baer-Nunziato Model.” In: *ESAIM: Proceedings* 40, pp. 63–82.
- Dallet, Sophie (Dec. 2016). “A Comparative Study of Numerical Schemes for the Baer-Nunziato Model.” In: *International Journal on Finite Volumes* 13, pp. 1–37.
- Daru, V. and C. Tenaud (Jan. 20, 2004). “High Order One-Step Monotonicity-Preserving Schemes for Unsteady Compressible Flow Calculations.” In: *Journal of Computational Physics* 193.2, pp. 563–594. ISSN: 0021-9991.

- David, Etienne et al. (Nov. 23, 2004). “Fuel Injection System with Multipoint Feed.” U.S. pat. 6820425B2. Safran Transmission Systems SAS. URL: <https://patents.google.com/patent/US6820425B2/en>.
- De Chaisemartin, S. et al. (June 2009). “Eulerian Models for Turbulent Spray Combustion with Polydispersity and Droplet Crossing.” In: *Comptes Rendus Mécanique* 337.6-7, pp. 438–448. ISSN: 16310721.
- Deledicque, Vincent and Miltiadis V. Papalexandris (Mar. 1, 2007). “An Exact Riemann Solver for Compressible Two-Phase Flow Models Containing Non-Conservative Products.” In: *Journal of Computational Physics* 222.1, pp. 217–245. ISSN: 0021-9991.
- Delhay, Jean-Marc (May 2001a). “Some Issues Related to the Modeling of Interfacial Areas in Gas–Liquid Flows I. The Conceptual Issues.” In: *Comptes Rendus de l’Académie des Sciences - Series IIB - Mechanics* 329.5, pp. 397–410. ISSN: 16207742.
- (June 2001b). “Some Issues Related to the Modeling of Interfacial Areas in Gas–Liquid Flows, II. Modeling the Source Terms for Dispersed Flows.” In: *Comptes Rendus de l’Académie des Sciences - Series IIB - Mechanics* 329.6, pp. 473–486. ISSN: 16207742.
- (2013). *Thermohydraulique des réacteurs: Edition révisée*. EDP Sciences. ISBN: 978-2-7598-1147-2.
- Dell’Isola, Francesco and S. L. Gavriluk (2012). *Variational Models and Methods in Solid and Fluid Mechanics*. Vol. 535. Multiphase Flow Modeling via Hamilton’s Principle. Springer Science & Business Media.
- Deserno, Markus (2004). *Notes on Differential Geometry*. Dover, New York. [http://www.cmu.edu/biolphys/deserno/pdf/diff\\_geom.pdf](http://www.cmu.edu/biolphys/deserno/pdf/diff_geom.pdf).
- Desjardins, Olivier et al. (2013). “Direct Numerical and Large-Eddy Simulation of Primary Atomization in Complex Geometries.” In: *Atomization and Sprays* 23.11.
- Devassy, Bejoy Mandumpala (2014). “Atomization Modeling of Liquid Jets using an Eulerian-Eulerian Model and a Surface Density Approach.”
- Devassy, Bejoy Mandumpala, Chawki Habchi, and Eric Daniel (2015). “Atomization Modelling of Liquid Jets Using a Two-Surface-Density Approach.” In: *Atomization and Sprays* 25.1.
- Di Battista, Ruben (2018). *Mercur(v)e — A Library to Exploit Geometrical and Topological Properties to Allow Post-Processing of DNS Simulations (and Much More)*. URL: <https://gitlab.com/rubendibattista/mercurve>.
- (2019). *Josiepy — A 2D PDE Solver Written in Python without Compromising (Too Much) Performance*. URL: <https://gitlab.com/rubendibattista/josiepy>.

- Di Battista, Ruben, Iván Bermejo-Moreno, Thibaut Ménard, Stéphane de Chaisemartin, et al. (May 19–24, 2019). “Post-Processing of Two-Phase DNS Simulations Exploiting Geometrical Features and Topological Invariants to Extract Flow Statistics and Droplets Number Density.” In: International Conference on Multiphase Flow. Rio de Janeiro. URL: <https://hal.archives-ouvertes.fr/hal-02345825v1>.
- Di Battista, Ruben, Iván Bermejo-Moreno, Thibaut Ménard, and Marc Massot (2019). “Geometrical Characterization and DNS Post-Processing of the 3D Objects in Two-Phase Flow: Collision of Two Droplets.” In: *NASA Technical Memorandum*.
- Di Battista, Ruben, Thibault Ménard, et al. (2021). “A Computational Framework Based on the Discrete Estimation of Geometrical Properties over Triangulated Interfaces Preserving Topological Invariants to Design and Validate Two-Phase Flow Models.” In: *Fluids*. In preparation.
- Doisneau, François, Joel Dupays, et al. (2011). “Two-Way Coupled Simulation of Acoustic Waves in Polydisperse Coalescing Two-Phase Flows: Application to Solid Rocket Motor Instabilities.” In: *4th European Conference for Aerospace Sciences*, pp. 1–16.
- Doisneau, François, Alaric Sibra, et al. (2014). “Numerical Strategy for Unsteady Two-Way Coupled Polydisperse Sprays: Application to Solid-Rocket Instabilities.” In: *Journal of Propulsion and Power* 30.3, pp. 727–748.
- Dopazo, César (1977). “On Conditioned Averages for Intermittent Turbulent Flows.” In: *Journal of Fluid Mechanics* 81.03, pp. 433–438.
- Dopazo, Cesar et al. (July 2018). “Strain, Rotation and Curvature of Non-Material Propagating Iso-Scalar Surfaces in Homogeneous Turbulence.” In: *Flow, Turbulence and Combustion* 101.1, pp. 1–32. ISSN: 1386-6184, 1573-1987.
- Drew, Donald A. (1990). “Evolution of Geometric Statistics.” In: *SIAM Journal on Applied Mathematics* 50.3, pp. 649–666.
- Drew, Donald A. and Stephen L. Passman (2006). *Theory of Multicomponent Fluids*. Vol. 135. Springer Science & Business Media.
- Druil, Florence (June 2017). “Eulerian Modeling and Simulations of Separated and Disperse Two-Phase Flows: Development of a Unified Modeling Approach and Associated Numerical Methods for Highly Parallel Computations.” PhD thesis. Université de Paris-Saclay. URL: <https://tel.archives-ouvertes.fr/tel-01618320>.
- Druil, Florence, Alexandru Fikl, et al. (Mar. 2016). “Experimenting with the P4est Library for AMR Simulations of Two-Phase Flows.” In: *ESAIM: Proceedings and Surveys* 53. Ed. by Martin Campos Pinto and Frédérique Charles, pp. 232–247. ISSN: 2267-3059.

- Drui, Florence, Adam Larat, et al. (Oct. 2019). “Small-Scale Kinematics of Two-Phase Flows: Identifying Relaxation Processes in Separated- and Disperse-Phase Flow Models.” In: *Journal of Fluid Mechanics* 876, pp. 326–355. ISSN: 0022-1120, 1469-7645.
- Duarte, Max et al. (2012). “New Resolution Strategy for Multiscale Reaction Waves Using Time Operator Splitting, Space Adaptive Multiresolution, and Dedicated High Order Implicit/Explicit Time Integrators.” In: *SIAM Journal on Scientific Computing* 34.1, A76–A104.
- Dupif, Valentin (June 22, 2018). “Modélisation et Simulation de l’écoulement Diphasique Dans Les Moteurs-Fusées à Propergol Solide Par Des Approches Eulériennes Polydispersées En Taille et En Vitesse.” Sous la direction de Marc Massot, Frédérique Laurent et de Joël Dupays. Soutenue le 22-06-2018, à l’Université Paris-Saclay (ComUE), dans le cadre de École doctorale de mathématiques Hadamard (Orsay, Essonne ; 2015-....), en partenariat avec CentraleSupélec (2015-....) (établissement opérateur d’inscription) et de Centre de mathématiques appliquées (Palaiseau, Essonne) (laboratoire). These de doctorat. Université Paris-Saclay (ComUE). URL: <http://theses.fr/2018SACLC050>.
- Duret, B. et al. (Nov. 1, 2018). “A Pressure Based Method for Vaporizing Compressible Two-Phase Flows with Interface Capturing Approach.” In: *International Journal of Multiphase Flow* 108, pp. 42–50. ISSN: 0301-9322.
- Duret, Benjamin et al. (2012). “DNS Analysis of Turbulent Mixing in Two-Phase Flows.” In: *International Journal of Multiphase Flow* 40, pp. 93–105.
- Essadki, Mohamed (Feb. 2018). “Contribution to a Unified Eulerian Modeling of Fuel Injection: From Dense Liquid to Polydisperse Evaporating Spray.” PhD thesis. Université de Paris-Saclay. URL: <https://tel.archives-ouvertes.fr/tel-01928584>.
- Essadki, Mohamed et al. (Nov. 1, 2019). “Statistical Modeling of the Gas–Liquid Interface Using Geometrical Variables: Toward a Unified Description of the Disperse and Separated Phase Flows.” In: *International Journal of Multiphase Flow* 120, p. 103084. ISSN: 0301-9322.
- Evrard, Fabien (Dec. 2017). “Numerical Methods for Multi-Scale Interfacial Flows.” URL: <http://spiral.imperial.ac.uk/handle/10044/1/69775>.
- Evrard, Fabien, Fabian Denner, and Berend van Wachem (May 2020). “Height-Function Curvature Estimation with Arbitrary Order on Non-Uniform Cartesian Grids.” In: *Journal of Computational Physics: X*, p. 100060. ISSN: 25900552.
- Fedkiw, Ronald P. et al. (1999). “A Non-Oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (the Ghost Fluid Method).” In: *Journal of computational physics* 152.2, pp. 457–492.

- “France Moves to Ban Short-Haul Domestic Flights” (Apr. 12, 2021). In: *BBC News. Europe*.
- François, Laurent et al. (2020). *Solid Propellant Combustion in the Low Mach One-Dimensional Approximation: From an Index-One Differential-Algebraic Formulation to High-Fidelity Simulations through High-Order Time Integration with Adaptive Time-Stepping*.
- Froehly, Algiane, Charles Dapogny, and Pascal Frey (June 26, 2019). “Remailler Pour Améliorer La Simulation Numérique : Exemple Avec La Plateforme Mmg.”
- Furfaro, Damien and Richard Saurel (Apr. 2015). “A Simple HLLC-Type Riemann Solver for Compressible Non-Equilibrium Two-Phase Flows.” In: *Computers & Fluids* 111, pp. 159–178. ISSN: 00457930.
- Gallice, Gerard (June 11, 2002). “Approximation numérique de Systèmes Hyperboliques Non-linéaires Conservatifs ou Non-conservatifs.” thesis. Université de Bordeaux I. URL: <https://hal-cea.archives-ouvertes.fr/tel-01320526>.
- Gallouët, Thierry, Jean-Marc Hérard, and Nicolas Seguin (May 2004). “Numerical Modeling of Two-Phase Flows Using the Two-Fluid Two-Pressure Approach.” In: *Math. Models Methods Appl. Sci.* 14.05, pp. 663–700. ISSN: 0218-2025, 1793-6314.
- Ganger, Brian (2008). *DistArray*. DistArray. URL: <http://docs.enthought.com/distarray/>.
- Gavrilyuk, Sergey and Henri Gouin (1999). “A New Form of Governing Equations of Fluids Arising from Hamilton’s Principle.” In: *International Journal of Engineering Science* 37.12, pp. 1495–1520.
- Gavrilyuk, Sergey and Richard Saurel (Jan. 2002). “Mathematical and Numerical Modeling of Two-Phase Compressible Flows with Micro-Inertia.” In: *Journal of Computational Physics* 175.1, pp. 326–360. ISSN: 00219991.
- Glimm, J., D. Saltz, and D. H. Sharp (1998). “Two-Phase Modelling of a Fluid Mixing Layer.” In: *J. Fluid Mech.* 378, pp. 119–143. ISSN: 0022-1120, 1469-7645.
- Glimm, James et al. (July 25, 2006). “Robust Computational Algorithms for Dynamic Interface Tracking in Three Dimensions.” In: *SIAM Journal on Scientific Computing*.
- Godlewski, Edwige and Pierre-Arnaud Raviart (1991). *Hyperbolic Systems of Conservation Laws*. Ellipses. 262 pp.
- (1996). *Numerical Approximation of Hyperbolic Systems of Conservation Laws*. Red. by J. E. Marsden, L. Sirovich, and F. John. Vol. 118. Applied Mathematical Sciences. New York, NY: Springer New York. ISBN: 978-1-4612-6878-9.
- Godunov, Sergei Konstantinovich (1959). “A Difference Method for Numerical Calculation of Discontinuous Solutions of the Equations of Hydrodynamics.” In: *Matematicheskii Sbornik* 89.3, pp. 271–306.



- Gottlieb, Sigal and Chi-Wang Shu (1996). *Total Variation Diminishing Runge-Kutta Schemes*. NASA Contractor Report 201591. Hampton, VA 23681-0001: Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, pp. 73–85. URL: <http://www.ams.org/jourcgi/jour-getitem?pii=S0025-5718-98-00913-2>.
- (Jan. 1, 1998). “Total Variation Diminishing Runge-Kutta Schemes.” In: *Math. Comp.* 67.221, pp. 73–85. ISSN: 0025-5718, 1088-6842.
- Gouin, Henri and Tommaso Ruggeri (2009). *The Hamilton Principle for Fluid Binary Mixtures with Two Temperatures*.
- Greenshields, Christopher J. (2015). *OpenFOAM Programming Guide*.
- Guillemaud, Vincent (2007). “Modélisation et Simulation Numérique Des Écoulements Diphasiques Par Une Approche Bifluide à Deux Pressions.” PhD thesis. Université de Provence-Aix-Marseille I.
- Harris, Charles R. et al. (Sept. 2020). “Array Programming with NumPy.” In: *Nature* 585.7825 (7825), pp. 357–362. ISSN: 1476-4687.
- Harten, Ami (1997). “High Resolution Schemes for Hyperbolic Conservation Laws.” In: *Journal of computational physics* 135.2, pp. 260–278.
- Harten, Amiram, Peter D. Lax, and Bram van Leer (1983). “On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws.” In: *SIAM review* 25.1, pp. 35–61.
- Hecht, Frédéric (2012). “New Development in FreeFem++.” In: *Journal of numerical mathematics* 20.3-4, pp. 251–266.
- Herrmann, M. (Oct. 25, 2013). “A Sub-Grid Surface Dynamics Model for Sub-Filter Surface Tension Induced Interface Dynamics.” In: *Computers & Fluids*. USNCCM Moving Boundaries 87, pp. 92–101. ISSN: 0045-7930.
- Hirt, C.W and B.D Nichols (Jan. 1981). “Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries.” In: *Journal of Computational Physics* 39.1, pp. 201–225. ISSN: 00219991.
- Hybrid Propulsion for Space* (2021). URL: <https://hybrid-propulsion.space/>.
- Inigo Quilez :: Fractals, Computer Graphics, Mathematics, Demoscene and More* (2018). URL: <https://iquilezles.org/www/articles/distfunctions/distfunctions.htm>.
- Irvine, Tom (Oct. 2008). “Apollo 13 Pogo Oscillation.” In.
- Ishii, M., S. Kim, and J. Uhle (July 1, 2002). “Interfacial Area Transport Equation: Model Development and Benchmark Experiments.” In: *International Journal of Heat and Mass Transfer* 45.15, pp. 3111–3123. ISSN: 0017-9310.
- Ishii, M. and K. Mishima (1984). “Two-Fluid Model and Hydrodynamic Constitutive Relations.” In: *Nuclear Engineering and design* 82.2-3, pp. 107–126.

- Jay, S, F Lacas, and S Candel (Feb. 2006). “Combined Surface Density Concepts for Dense Spray Combustion.” In: *Combustion and Flame* 144.3, pp. 558–577. ISSN: 00102180.
- John W. Strutt (3rd Baron Rayleigh) (1883). “Investigation of the Character of the Equilibrium of an Incompressible Heavy Fluid of Variable Density.” In: *Proceedings of the London Mathematical Society*. Vol. 14, pp. 170–177. URL: <https://www.irphe.fr/~clanet/otherpaperfile/articles/Rayleigh/rayleigh1883.pdf>.
- Kah, D. et al. (May 1, 2015). “High Order Moment Method for Polydisperse Evaporating Sprays with Mesh Movement: Application to Internal Combustion Engines.” In: *International Journal of Multiphase Flow* 71, pp. 38–65. ISSN: 0301-9322.
- Kamm, James Russell (Mar. 5, 2015). *An Exact, Compressible One-Dimensional Riemann Solver for General, Convex Equations of State*. LA-UR-15-21616, 1172220. Los Alamos National Laboratory. URL: <http://www.osti.gov/servlets/purl/1172220/>.
- Karabeyoglu, Arif et al. (2004). “Scale-up Tests of High Regression Rate Paraffin-Based Hybrid Rocket Fuels.” In: *Journal of propulsion and power* 20.6, pp. 1037–1045.
- Kindlmann, G. et al. (Oct. 2003). “Curvature-Based Transfer Functions for Direct Volume Rendering: Methods and Applications.” In: *IEEE Visualization, 2003. VIS 2003*. IEEE Visualization, 2003. VIS 2003. Pp. 513–520.
- Klabnik, Steve and Carol Nichols (June 26, 2018). *The Rust Programming Language*. Illustrated edition. San Francisco: No Starch Press. 552 pp. ISBN: 978-1-59327-828-1.
- Lalanne, Benjamin, Sébastien Tanguy, and Frédéric Risso (2013). “Effect of Rising Motion on the Damped Shape Oscillations of Drops and Bubbles.” In: *Physics of Fluids* 25.11, p. 112107.
- Lamb, Horace (1916). *Hydrodynamics*. 4th ed. Cambridge.
- Lamb, Sir Horace (1895). *Hydrodynamics*. University Press. 632 pp. ISBN: 978-1-4086-1332-0.
- Lasheras, Juan C. and E. J. Hopfinger (2000). “Liquid Jet Instability and Atomization in a Coaxial Gas Stream.” In: *Annual review of fluid mechanics* 32.1, pp. 275–308.
- Laurent, Frédérique and Marc Massot (Dec. 2001). “Multi-Fluid Modelling of Laminar Polydisperse Spray Flames: Origin, Assumptions and Comparison of Sectional and Sampling Methods.” In: *Combustion Theory and Modelling* 5.4, pp. 537–572. ISSN: 1364-7830, 1741-3559.
- Laurent, Frédérique, Marc Massot, and Philippe Villedieu (Mar. 2004). “Eulerian Multi-Fluid Modeling for the Numerical Simulation of Coalescence in Polydisperse Dense Liquid Sprays.” In: *Journal of Computational Physics* 194.2, pp. 505–543. ISSN: 00219991.
- Le Touze, Clément (Dec. 3, 2015). “Couplage entre modèles diphasiques à « phases séparées » et à « phase dispersée » pour la simulation de l’atomisation primaire en combustion cry-



- otechnique.” PhD thesis. Université Nice Sophia Antipolis. URL: <https://tel.archives-ouvertes.fr/tel-01250527/document>.
- Le Touze, Clément, Angelo Murrone, and Herve Guillard (Dec. 2014). “Multislope MUSCL Method for General Unstructured Meshes.” In: *Journal of Computational Physics*, p. 44.
- Lebas, Romain et al. (2005). *Coupling Vaporization Model with the Eulerian-Lagrangian Spray Atomization (ELSA) Model in Diesel Engine Conditions*. SAE Technical Paper.
- Lee, Joosung J et al. (2001). “Historical and Future Trends in Aircraft Performance, Cost, and Emissions.” In: *Annual Review of Energy and the Environment*, p. 44.
- Lefebvre, Arthur H. and Vincent G. McDonell (Apr. 4, 2017). *Atomization and Sprays*. 2nd edition. Boca Raton: CRC Press. 300 pp. ISBN: 978-1-4987-3625-1.
- LeVeque, Randall J. (1990). *Numerical Methods for Conservation Laws*. Basel: Birkhäuser Basel. ISBN: 978-3-7643-2464-3.
- (Aug. 26, 2002). *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press. 582 pp. ISBN: 978-0-521-00924-9.
- LeVeque, Randall J. and Helen C. Yee (1990). “A Study of Numerical Methods for Hyperbolic Conservation Laws with Stiff Source Terms.” In: *Journal of computational physics* 86.1, pp. 187–210.
- Lewiner, Thomas et al. (Jan. 2003). “Efficient Implementation of Marching Cubes’ Cases with Topological Guarantees.” In: *Journal of Graphics Tools* 8.2, pp. 1–15. ISSN: 1086-7651.
- Lhuillier, D., C. H. Chang, and T. G. Theofanous (Sept. 1, 2013). “On the Quest for a Hyperbolic Effective-Field Model of Disperse Flows.” In: *Journal of Fluid Mechanics* 731, pp. 184–194. ISSN: 0022-1120.
- Lhuillier, Daniel (Mar. 2003). “A Mean-Field Description of Two-Phase Flows with Phase Changes.” In: *International Journal of Multiphase Flow* 29.3, pp. 511–525. ISSN: 03019322.
- (Feb. 2004). “Evolution of the Volumetric Interfacial Area in Two-Phase Mixtures.” In: *Comptes Rendus Mécanique* 332.2, pp. 103–108. ISSN: 16310721.
- Ling, Yue, Daniel Fuster, et al. (Jan. 23, 2017). “Spray Formation in a Quasiplanar Gas-Liquid Mixing Layer at Moderate Density Ratios: A Numerical Closeup.” In: *Phys. Rev. Fluids* 2.1, p. 014005.
- Ling, Yue, Stéphane Zaleski, and R. Scardovelli (2015). “Multiscale Simulation of Atomization with Small Droplets Represented by a Lagrangian Point-Particle Model.” In: *International Journal of Multiphase Flow* 76, pp. 122–143.
- Liu, Chun and Jie Shen (May 2003). “A Phase Field Model for the Mixture of Two Incompressible Fluids and Its Approximation by a Fourier-Spectral Method.” In: *Physica D: Nonlinear Phenomena* 179.3-4, pp. 211–228. ISSN: 01672789.

- Liu, Yujie (Sept. 11, 2013). “Contribution à la vérification et à la validation d’un modèle diphasique bifluide instationnaire.” PhD thesis. URL: <https://tel.archives-ouvertes.fr/tel-00864567>.
- Loison, Arthur (2023). “Unified Eulerian Modeling and High-Performance Numerical Simulation of Two-Phase Flows.” Preliminary Title.
- Lorensen, William E. and Harvey E. Cline (1987). “Marching Cubes: A High Resolution 3D Surface Construction Algorithm.” In: *ACM Siggraph Computer Graphics*. Vol. 21. ACM, pp. 163–169.
- Massot, Marc, Frédérique Laurent, et al. (2010). “A Robust Moment Method for Evaluation of the Disappearance Rate of Evaporating Sprays.” In: *SIAM Journal on Applied Mathematics* 70.8, pp. 3203–3234.
- Massot, Marc, Laurent Series, et al. (2020). “Introduction à l’analyse Numérique : Des Fondements Mathématiques à l’experimentation Avec Jupyter.” Polycopié. Polycopié. École Polytechnique.
- Ménard, Thibault, Sebastien Tanguy, and Alain Berlemont (2007). “Coupling Level Set/VOF/Ghost Fluid Methods: Validation and Application to 3D Simulation of the Primary Break-up of a Liquid Jet.” In: *Intern. J. Multiph. Flow* 33.5, pp. 510–524.
- Meyer, Mark et al. (2003). “Discrete Differential-Geometry Operators for Triangulated 2-Manifolds.” In: *Visualization and Mathematics III*. Ed. by Hans-Christian Hege and Konrad Polthier. Red. by Gerald Farin et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 35–57. ISBN: 978-3-642-05682-6. URL: [http://link.springer.com/10.1007/978-3-662-05105-4\\_2](http://link.springer.com/10.1007/978-3-662-05105-4_2).
- Mirjalili, Shahab, Christopher B. Ivey, and Ali Mani (July 2019). “Comparison between the Diffuse Interface and Volume of Fluid Methods for Simulating Two-Phase Flows.” In: *International Journal of Multiphase Flow* 116, pp. 221–238. ISSN: 03019322.
- Mitchell, Don and Pat Hanrahan (July 1, 1992). “Illumination from Curved Reflectors.” In: *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’92. New York, NY, USA: Association for Computing Machinery, pp. 283–291. ISBN: 978-0-89791-479-6. URL: <https://doi.org/10.1145/133994.134082>.
- Morel, Christophe (Oct. 2007). “On the Surface Equations in Two-Phase Flows and Reacting Single-Phase Flows.” In: *International Journal of Multiphase Flow* 33.10, pp. 1045–1073. ISSN: 03019322.
- (2015). *Mathematical Modeling of Disperse Two-Phase Flows*. Vol. 114. Fluid Mechanics and Its Applications. Cham: Springer International Publishing. ISBN: 978-3-319-20103-0.

- Morel, Christophe, Nicolas Goreaud, and Jean-Marc Delhay (1999). “The Local Volumetric Interfacial Area Transport Equation: Derivation and Physical Significance.” In: *International Journal of Multiphase Flow*, p. 30.
- Moukalled, F., L. Mangani, and M. Darwish (2016). *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab*. Fluid Mechanics and Its Applications. Springer International Publishing. ISBN: 978-3-319-16873-9.
- N’Guessan, Marc-Arthur (Mar. 11, 2020). “Space Adaptive Methods with Error Control Based on Adaptive Multiresolution for the Simulation of Low-Mach Reactive Flows.” PhD thesis. Université Paris-Saclay. URL: <https://tel.archives-ouvertes.fr/tel-02895792>.
- Nguyen Tri Nguyen (2018). “Numerical Methods for Compressible Multi-Phase Flows with Surface Tension.” In: *Unpublished*.
- O’Rourke, Peter J. and Anthony A. Amsden (Nov. 1, 1987). “The Tab Method for Numerical Calculation of Spray Droplet Breakup.” In: 1987 SAE International Fall Fuels and Lubricants Meeting and Exhibition. URL: <https://www.sae.org/content/872089/>.
- Perkel, Jeffrey M. (July 30, 2019). “Julia: Come for the Syntax, Stay for the Speed.” In: *Nature* 572.7767 (7767), pp. 141–142.
- Petit, Xavier et al. (2013). “Large-Eddy Simulation of Supercritical Fluid Injection.” In: *The Journal of Supercritical Fluids* 84, pp. 61–73.
- Phase-out of Fossil Fuel Vehicles (May 26, 2021). In: *Wikipedia*. URL: [https://en.wikipedia.org/w/index.php?title=Phase-out\\_of\\_fossil\\_fuel\\_vehicles&oldid=1025234434](https://en.wikipedia.org/w/index.php?title=Phase-out_of_fossil_fuel_vehicles&oldid=1025234434).
- Plümacher, D. et al. (June 2020). “On a Non-Linear Droplet Oscillation Theory via the Unified Method.” In: *Physics of Fluids* 32.6, p. 067104. ISSN: 1070-6631, 1089-7666.
- Preferred Infrastructure, Inc. (2021). *CuPy*. CuPy. URL: <https://cupy.dev/>.
- Primo | A Small Lightweight Nanolauncher for Your Small Sat (2018). Leaf Space. URL: <https://web.archive.org/web/20180802032946/https://leaf.space/primo/>.
- Prosperetti, Andrea (1980). “Free Oscillations of Drops and Bubbles: The Initial-Value Problem.” In: *Journal of Fluid Mechanics* 100.2, pp. 333–347.
- Puggelli, Stefano (Apr. 24, 2018). “Vers une approche unifiée pour la simulation aux grandes échelles d’écoulements réactifs, diphasiques et turbulents.” PhD thesis. Normandie Université ; Università degli studi (Florence, Italie). URL: <https://tel.archives-ouvertes.fr/tel-01820637>.
- Quartapelle, L. (2015). “Godunov Method and Related Schemes for Nonlinear Hyperbolic Problems.” Lecture Notes. Lecture Notes.

- Quartapelle, Luigi and Franco Auteri (Apr. 1, 2013). *Fluidodinamica incompressibile*. Rozzano: CEA. 400 pp. ISBN: 978-88-08-18539-6.
- Ransom, V H and D L Hicks (June 1982). “Hyperbolic Two-Pressure Models for Two-Phase.” In: p. 28.
- Reid, W. H. (Apr. 1960). “The Oscillations of a Viscous Liquid Drop.” In: *Quarterly of Applied Mathematics* 18.1, pp. 86–89. ISSN: 0033-569X, 1552-4485.
- Rekakavas, A. et al. (2015). “Experimental Visualizations of Entrainment Phenomena in Wax-Based Fuels for Hybrid Space Propulsion.” In: 6th European Conference for Aeronautics and Space Sciences (EUCASS).
- Remigi, Alberto (Mar. 2021). “Modélisation Numérique d’un Injecteur Aéromécanique: De l’écoulement Interne Au Spray Dispersé.” PhD thesis.
- Remigi, Alberto et al. (May 19–24, 2019). “Exploring Different Approaches for the Simulation of Multi-Scale Atomization Process.” In: International Conference on Multiphase Flow. Rio de Janeiro. URL: <https://hal.archives-ouvertes.fr/hal-02379257>.
- Roe, Philip L. (1981). “Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes.” In: *Journal of computational physics* 43.2, pp. 357–372.
- Rudy, David H. and John C. Strikwerda (Sept. 1981). “Boundary Conditions for Subsonic Compressible Navier-Stokes Calculations.” In: *Computers & Fluids* 9.3, pp. 327–338. ISSN: 00457930.
- Rumpf, Martin and Tobias Preußer (July 3, 2002). “A Level Set Method For Anisotropic Geometric Diffusion In 3D Image Processing.” In: *SIAM Journal of Applied Mathematics* 62, pp. 1772–1793.
- Rusanov, V.V (Jan. 1962). “The Calculation of the Interaction of Non-Stationary Shock Waves and Obstacles.” In: *USSR Computational Mathematics and Mathematical Physics* 1.2, pp. 304–320. ISSN: 00415553.
- Rusche, Henrik (2003). “Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions.” Imperial College London (University of London). URL: <http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.402087>.
- Russo, Giovanni and Peter Smereka (Sept. 2000). “A Remark on Computing Distance Functions.” In: *Journal of Computational Physics* 163.1, pp. 51–67. ISSN: 00219991.
- Saleh, Khaled (Dec. 26, 2012). “Analyse et Simulation Numérique par Relaxation d’Ecoulements Diphasiques Compressibles. Contribution au Traitement des Phases Evanescences.” PhD thesis. Université Pierre et Marie Curie - Paris VI. URL: <https://tel.archives-ouvertes.fr/tel-00761099>.

- Sanjosé, M. et al. (2011). “Fuel Injection Model for Euler–Euler and Euler–Lagrange Large-Eddy Simulations of an Evaporating Spray inside an Aeronautical Combustor.” In: *International Journal of Multiphase Flow* 37.5, pp. 514–529.
- Saurel, Richard and Rémi Abgrall (Apr. 10, 1999). “A Multiphase Godunov Method for Compressible Multifluid and Multiphase Flows.” In: *Journal of Computational Physics* 150.2, pp. 425–467. ISSN: 0021-9991.
- Saurel, Richard, Sergey Gavrilyuk, and François Renaud (Nov. 2003). “A Multiphase Model with Internal Degrees of Freedom: Application to Shock–Bubble Interaction.” In: *Journal of Fluid Mechanics* 495, pp. 283–321. ISSN: 1469-7645, 0022-1120.
- Saurel, Richard, Fabien Petitpas, and Ray A. Berry (2009). “Simple and Efficient Relaxation Methods for Interfaces Separating Compressible Fluids, Cavitating Flows and Shocks in Multiphase Mixtures.” In: *Journal of Computational Physics* 228.5, pp. 1678–1712.
- Schroeder, Will, Ken Martin, and Bill Lorensen (2006). *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware. 512 pp. ISBN: 978-1-930934-19-1.
- Schroeder, William, Rob Maynard, and Berk Geveci (Oct. 25–26, 2015). “Flying Edges: A High-Performance Scalable Isocontouring Algorithm.” In: 2015 IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV), pp. 33–40. URL: <http://ieeexplore.ieee.org/document/7348069/>.
- Schwendeman, Donald W., Christopher W. Wahle, and Ashwani K. Kapila (2006). “The Riemann Problem and a High-Resolution Godunov Method for a Model of Compressible Two-Phase Flow.” In: *Journal of Computational Physics* 212.2, pp. 490–526.
- Scoggins, James B., Jiequn Han, and Marc Massot (2021). “Machine Learning Moment Closures for Accurate and Efficient Simulation of Polydisperse Evaporating Sprays.” In: *AIAA Scitech 2021 Forum*. American Institute of Aeronautics and Astronautics. URL: <https://arc.aiaa.org/doi/abs/10.2514/6.2021-1786>.
- Seguin, Nicolas (Nov. 22, 2002). “Modélisation et Simulation Numérique Des Écoulements Diphasiques.” PhD thesis. URL: <https://tel.archives-ouvertes.fr/tel-00003139>.
- Sethian, J. A. (Feb. 20, 1996). “A Fast Marching Level Set Method for Monotonically Advancing Fronts.” In: *Proceedings of the National Academy of Sciences* 93.4, pp. 1591–1595. ISSN: 0027-8424, 1091-6490.
- Shinjo, J. and A. Umemura (2010). “Simulation of Liquid Jet Primary Breakup: Dynamics of Ligament and Droplet Formation.” In: *International Journal of Multiphase Flow* 36.7, pp. 513–532.

- Shu, Chi-Wang (2003). “High-Order Finite Difference and Finite Volume WENO Schemes and Discontinuous Galerkin Methods for CFD.” In: *International Journal of Computational Fluid Dynamics* 17.2, pp. 107–118.
- Shukla, Ratnesh K., Carlos Pantano, and Jonathan B. Freund (2010). “An Interface Capturing Method for the Simulation of Multi-Phase Compressible Flows.” In: *Journal of Computational Physics* 229.19, pp. 7411–7439.
- Sod, Gary A. (1978). “A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws.” In: *Journal of computational physics* 27.1, pp. 1–31.
- Taylor, Geoffrey Ingram (1963). *The Scientific Papers of Sir Geoffrey Ingram Taylor*. Ed. by G. K. Batchelor. Reissue edition. Vol. 3. 4 vols. Cambridge: Cambridge University Press. 457 pp. ISBN: 978-0-521-15879-4.
- Thiesset, F, C Dumouchel, et al. (2019). “Probing Liquid Atomization Using Probability Density Functions, the Volume-Based Scale Distribution and Differential Geometry.” In: ILASS. Paris, p. 7.
- Thiesset, F, T Ménard, et al. (2019). “A New Theoretical Framework for Characterizing the Transport of Liquid in Turbulent Two-Phase Flows.” In: p. 8.
- Thiesset, F., B. Duret, et al. (Jan. 10, 2020). “Geometry, Topology, Morphology of Liquid Structures.”
- Thiesset, F., T. Ménard, and C. Dumouchel (Apr. 2021). “Space-Scale-Time Dynamics of Liquid–Gas Shear Flow.” In: *Journal of Fluid Mechanics* 912. ISSN: 0022-1120, 1469-7645.
- Thompson, Joe F., Bharat K. Soni, and Nigel P. Weatherill (Dec. 29, 1998). *Handbook of Grid Generation*. Boca Raton, Fla: CRC Press. 1136 pp. ISBN: 978-0-8493-2687-5.
- Tiwari, Arpit, Jonathan B. Freund, and Carlos Pantano (Nov. 1, 2013). “A Diffuse Interface Model with Immiscibility Preservation.” In: *Journal of Computational Physics* 252, pp. 290–309. ISSN: 0021-9991.
- Tokareva, S. A. and E. F. Toro (May 20, 2010). “HLLC-Type Riemann Solver for the Baer–Nunziato Equations of Compressible Two-Phase Flow.” In: *Journal of Computational Physics* 229.10, pp. 3573–3604. ISSN: 0021-9991.
- (Oct. 2016). “A Flux Splitting Method for the Baer–Nunziato Equations of Compressible Two-Phase Flow.” In: *Journal of Computational Physics* 323, pp. 45–74. ISSN: 00219991.
- Toro, E.F. et al. (May 2020). “Low-Dissipation Centred Schemes for Hyperbolic Equations in Conservative and Non-Conservative Form.” In: *Journal of Computational Physics*, p. 109545. ISSN: 00219991.



- Toro, Eleuterio F. (2009). *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. 3rd ed. Berlin Heidelberg: Springer-Verlag. ISBN: 978-3-540-25202-3.
- (2019). “The HLLC Riemann Solver.” In: *Shock waves*, pp. 1–18.
- Trinh, E. and T. G. Wang (Sept. 1982). “Large-Amplitude Free and Driven Drop-Shape Oscillations: Experimental Observations.” In: *Journal of Fluid Mechanics* 122 (-1), p. 315. ISSN: 0022-1120, 1469-7645.
- Tryggvason, Grétar et al. (2001). “A Front-Tracking Method for the Computations of Multiphase Flow.” In: *Journal of computational physics* 169.2, pp. 708–759.
- Tsamopoulos, John A. and Robert A. Brown (Feb. 1983). “Nonlinear Oscillations of Inviscid Drops and Bubbles.” In: *Journal of Fluid Mechanics* 127 (-1), p. 519. ISSN: 0022-1120, 1469-7645.
- Vallet, Ariane and Roland Borghi (Sept. 1999). “Modélisation eulerienne de l’atomisation d’un jet liquide.” In: *Comptes Rendus de l’Académie des Sciences - Series IIB - Mechanics-Physics-Astronomy* 327.10, pp. 1015–1020. ISSN: 12874620.
- Vaudor, G. et al. (2017). “A Consistent Mass and Momentum Flux Computation Method for Two Phase Flows. Application to Atomization Process.” In: *Comput. Fluids* 152, pp. 204–216.
- Versteeg, H. and W. Malalasekera (Feb. 6, 2007). *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. 2nd edition. Harlow, England ; New York: Pearson. 520 pp. ISBN: 978-0-13-127498-3.
- Vignat, Guillaume et al. (2021). “A Joint Experimental and Large Eddy Simulation Characterization of the Liquid Fuel Spray in a Swirl Injector.” In: *Journal of Engineering for Gas Turbines and Power* 143.8, p. 081019.
- Vincent, S. et al. (June 6, 2019). *A Phase Inversion Benchmark for Multiscale Multiphase Flows*. URL: <http://arxiv.org/abs/1906.02655>.
- Wargnier, Quentin et al. (2019). “Numerical Treatment of the Nonconservative Product in a Multiscale Fluid Model for Plasmas in Thermal Nonequilibrium: Application to Solar Physics.” In.
- Weyl, Hermann (1939). “On the Volume of Tubes.” In: *American Journal of Mathematics* 61.2, pp. 461–472. ISSN: 0002-9327.
- Woodward, Paul and Phillip Colella (1984). “The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks.” In: *Journal of computational physics* 54.1, pp. 115–173.
- XDMF Model and Format - XdmfWeb (2021). URL: [https://www.xdmf.org/index.php/XDMF\\_Model\\_and\\_Format](https://www.xdmf.org/index.php/XDMF_Model_and_Format).

- Zamansky, Rémi et al. (2014). “Radiation Induces Turbulence in Particle-Laden Fluids.” In: *Physics of Fluids* 26.7, p. 071701.
- Zandian, Arash, William A. Sirignano, and Fazle Hussain (2018). “Understanding Liquid-Jet Atomization Cascades via Vortex Dynamics.” In: *Journal of Fluid Mechanics* 843, pp. 293–354.
- Zhang, Jun (1996). “Acceleration of Five-Point Red-Black Gauss-Seidel in Multigrid for Poisson Equation.” In: *Applied Mathematics and Computation* 80.1, p. 73.
- Zhao, Hongkai (May 21, 2004). “A Fast Sweeping Method for Eikonal Equations.” In: *Mathematics of Computation* 74.250, pp. 603–628. ISSN: 0025-5718.
- Zou, Ziqiang et al. (May 2019). “An Accurate Sharp Interface Method for Two-Phase Compressible Flows at Low-Mach Regime.” In: *International Conference on Multiphase Flow*. Rio de Janeiro, Brazil. URL: <https://hal.archives-ouvertes.fr/hal-02187217>.



**Titre :** Vers un cadre unifié de modélisation eulérienne pour les écoulements diphasiques : phénomènes géométriques à petite échelle et stratégies de calcul flexibles associées

**Mots clés :** écoulements diphasiques, modélisation de sous-échelle, densité surfacique d'interface, courbures, DNS, géométrie computationnelle, calcul scientifique

**Résumé :** Nous assistons actuellement à une « deuxième course à l'espace » : des entreprises privées comme SpaceX ouvrent la voie à une nouvelle génération de systèmes de lanceurs spatiaux optimisés pour leur rentabilité et leurs performances extrêmes, qui permettront à l'humanité d'atteindre Mars pour la première fois dans son existence. L'un des aspects essentiels de ces systèmes est d'offrir un niveau élevé de réutilisation, ce qui entraîne une baisse drastique des coûts de lancement. Cela se traduit par des systèmes de propulsion qui doivent fonctionner dans des enveloppes de vol plus larges, avec des paires d'ergols plus avantageuses comme le méthane et l'oxygène liquides, ce qui exige une conception plus rigoureuse des systèmes d'injection. Les injecteurs sont responsables de la nébulisation correcte des ergols et ils ont un impact direct sur les

performances des moteurs.

Ce travail identifie trois points principaux d'amélioration : le développement de modèles d'ordre réduit avec le principe d'action stationnaire, comportant un ensemble d'équations qui incluent des propriétés géométriques telles que la densité de la surface interfaciale et les courbures moyenne et de Gauss ; la mise en œuvre d'un outil de post-traitement géométrique pour les simulations à haute-fidélité utilisé pour recueillir des informations utiles afin d'élaborer un modèle d'ordre réduit précis, et le développement d'une bibliothèque Python qui agit comme un outil de prototypage rapide visant à tester rapidement des idées dans le contexte des schémas numériques, des conditions limites, des configurations de domaine, avec la possibilité d'exploiter des architectures de calcul modernes comme les GPU.

**Title :** Towards a unified Eulerian modeling framework for two-phase flows: geometrical small-scale phenomena and associated flexible computing strategies

**Keywords :** two-phase flows, subscale modeling, interface area density, curvatures, DNS, computational geometry, scientific computing

**Abstract :** In current times we are witnessing a « second space race »: private companies like SpaceX are paving the way to a new generation of space launcher systems optimized for cost effectiveness and extreme performances that will bring humankind to Mars for the first time in its existence. A key aspect of those systems is to provide a high level of reusability leading to a drastic drop of launch costs. This translates into propulsion systems that need to operate on wider flight envelopes, with more advantageous propellant pairs like cryogenic methane and liquid oxygen, therefore requiring tighter designs for the injection systems. The injectors are responsible for the correct nebulization of fuel and oxidizer and they have a direct impact on the performance of the engines. These kind of problems are shared across different applications and are somehow generic.

The current state of the art modeling strategies fail at predicting the correct distributions of droplets in the combustion chamber. Therefore, the target of this thesis is to contribute to the design of a unified modeling framework addressing the derivation of system of equations governing two-phase flow systems charac-

terized by a sound mathematical structure via a variational approach named Stationary Action Principle (SAP) coupled to the second principle of thermodynamics. This effort is backed by a tailored computational toolset that allows the rational choice of modeling assumptions and the effective simulations of the developed models, possibly on modern computing architectures.

This work identifies three main points of improvement: the development of reduced-order models via a variational procedure named the SAP featuring a set of equations that include geometrical properties such as the interfacial surface density and the mean and Gauss curvatures; the implementation of a geometric Direct Numerical Simulations (DNS) post-processing tool that is used to collect useful insight from high-fidelity simulations in order to craft an accurate reduced-order model, and the development of a Python library that acts as a prototyping playbook aimed at quickly testing ideas in the context of numerical schemes, boundary conditions, domain configurations, with the potential ability of leveraging modern computational architectures such as GPUs