

Modelling multi-domain large deformation problems using an Eulerian monolithic approach in a massively parallel environment

Fadi El Haddad

▶ **To cite this version:**

Fadi El Haddad. Modelling multi-domain large deformation problems using an Eulerian monolithic approach in a massively parallel environment. Modeling and Simulation. Ecole Nationale Supérieure des Mines de Paris, 2015. English. ⟨NNT : 2015ENMP0011⟩. ⟨tel-01228591⟩

HAL Id: tel-01228591

https://pastel.hal.science/tel-01228591v1

Submitted on 13 Nov 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

MINES ParisTech

École doctorale n° 364 : Sciences Fondamentales et Appliquées

Doctorat ParisTech
T H È S E

pour obtenir le grade de docteur délivré par

l'École nationale supérieure des mines de Paris

Spécialité " Mécanique Numérique "

Fadi EL HADDAD

Soutenue publiquement

le 29 Mai 2015

Modelling multi-domain large deformation problems using an Eulerian monolithic approach in a massively parallel environment

~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

Modélisation des problèmes de grandes déformations multi-domaines par une approche Eulérienne monolithique massivement parallèle

Directeur de thèse : **Thierry COUPEZ**

Maître de thèse : **Hugues Digonnet**

Jury

Pr. ROUX François-Xavier, Professeur Université Pierre et Marie Curies –Paris 6, France	Rapporteur
Dr. BRUCHON Julien, Maître Assistant, HDR Ecole Nationale Supérieure des Mines de Saint Étienne, France	Rapporteur
Pr. CESAR DE SÁ José Manuel de Almeida, Professeur Université de Porto, Portugal	Président
Dr. PERCHAT Etienne, Docteur Transvalor, France	Examinateur
Dr. DIGONNET Hugues, Chargé de recherche, Ecole Centrale de Nantes, France	Examinateur
Pr. COUPEZ Thierry, Professeur Ecole Centrale de Nantes, France	Examinateur

MINES ParisTech
CEMEF – UMR CNRS 7635
1, rue Claude Daunesse, CS 207 – 06904 Sophia Antipolis Cedex, France

T H È S E

To My Family

Acknowledgements

This work has been a very special project. I am deeply greatful to my family and to all whose enthusiasm and energy transformed my dream into reality.
I would like to express my sincere thanks :

- To my supervisors Professor Thierry Coupez and Doctor Hugues Digonnet, for their time, help, guidance and patience.
- To the personnel in CEMEF, especially Marie-Françoise Guenegan and Françoise Trucas for their moral support.
- To my friends in CEMEF, Ugo, Koffi, Ana-laura, Hong Chau, Carole, Nadine and Lionel for their kindness and for the good time we spent together
- To a special person who made my journey pleasant and joyful, my dear Rebecca
- And most of all, to my wonderful parents Joesph and Geneviève and my sisters Josiane and Theresa for their presence, support, motivations and kind words.

Contents

Introduction **1**

1 Industrial Motivation and Aim of this study . . . 2

2 The layout of this thesis . . . 3

Bibliography . . . 5

1 State of the art **7**

1.1 Introduction . . . 9

1.2 Lagrangian Description . . . 9

1.2.1 Spatial discretization . . . 11

1.2.2 Temporal discretization . . . 12

1.2.3 Contact . . . 13

1.2.3.1 Contact search algorithms . . . 14

1.2.3.2 Contact detection . . . 17

1.2.3.3 Parallel strategies in contact detection . . . 20

1.2.3.4 Contact resolution . . . 22

1.3 Eulerian Description . . . 24

1.3.1 Interface modeling . . . 25

1.3.2 Contact formulation . . . 25

1.4 Arbitrary Lagrangian Eulerian Description . . . 27

1.4.1 Coupled Resolution . . . 29

1.4.2 Split operator . . . 29

1.5 Discussion and Conclusion . . . 31

Résumé en Français . . . 33

Bibliography . . . 34

2 Monolithic approach tools **41**

2.1 Introduction . . . 43
2.2 Interface Capturing . . . 44
2.2.1 Distance function . . . 44
2.2.2 Level Set Method . . . 45
2.2.2.1 Local Distance function . . . 46
2.2.2.2 Convected level set method . . . 47
2.2.2.3 Weak form . . . 48
2.3 Mesh Generation . . . 49
2.4 Anisotropic mesh adaptation . . . 50
2.4.1 Edge based error estimator . . . 51
2.4.2 Metric construction . . . 52
2.4.3 Extension to multi-component field . . . 55
2.5 Mechanical problem . . . 56
2.5.1 Governing equations . . . 56
2.5.2 Weak formulation . . . 58
2.5.3 Monolithic system . . . 59
2.5.4 Finite Element Formulation . . . 59
2.5.4.1 Spatial discretization . . . 59
2.5.4.2 Variational Multi-scale Stabilization . . . 60
2.5.4.3 Stabilization parameter . . . 63
2.6 Mixture laws . . . 64
2.6.1 Linear mixture law . . . 65
2.6.2 Novel mixture law: Quadratic law . . . 66
2.7 New additions and Numerical Applications . . . 67
2.7.1 Test cases and proposed ameliorations . . . 67
2.7.1.1 Optimal parameter choices . . . 67
2.7.1.2 Weight of mixture laws . . . 69
2.7.1.3 Transport stabilization . . . 74
2.7.2 Air trapping . . . 77
2.7.3 Porous boundary conditions . . . 79
2.8 Conclusion . . . 81
Résumé en Français . . . 83
Bibliography . . . 84

3 Parallel Computing **89**
3.1 Introduction . . . 91
3.2 Concepts and Terminology . . . 91
3.2.1 Flynn's Classical Taxonomy . . . 92
3.2.2 Memory classification and access . . . 93
3.3 Performance of a Parallel Code . . . 95
3.3.1 Hardware performance . . . 95
3.3.2 Software performance . . . 96
3.3.3 Limits and Costs of Parallel Programming . . . 97
3.3.3.1 Serial fraction . . . 97
3.3.3.2 Communications between cores . . . 99
3.3.3.3 Load balancing . . . 100
3.4 Parallel computing in CIMLib . . . 100
3.4.1 S.P.M.D parallelization of the Navier-Stokes Solver . . . 101
3.4.2 Dynamic load balancing . . . 103
3.5 Applications . . . 104
3.5.1 Case presentation . . . 104
3.5.2 Navier-Stokes vs Stokes . . . 106
3.5.3 Coarse mesh: MPI Version and Bind to Core Option . . . 107
3.5.4 Coarse mesh: Hardware limitation and Round Robin Option . . . 111
3.5.5 Refined mesh . . . 113
3.5.6 Anisotropic mesh: Adaptation Scalability . . . 115
3.6 Conclusion . . . 118
Résumé en Français . . . 119
Bibliography . . . 120

4 Applications **123**
4.1 Introduction . . . 124
4.2 Industrial applications . . . 124
4.2.1 Connecting rod . . . 125
4.2.2 Crankshaft . . . 128
4.3 Multi-domain applications . . . 132
4.3.1 Ten Parallelepiped . . . 132
4.3.2 256 spheres . . . 137
4.4 Conclusions and Discussions . . . 140
Résumé en Français . . . 142
Bibliography . . . 143

5 Friction and contact **145**
5.1 Introduction . . . 146
5.2 Friction: from Lagrangian to Eulerian description . . . 146
5.2.1 Lagrangian Friction law . . . 147
5.2.2 Eulerian friction law . . . 148
5.2.3 Friction validation . . . 149
5.3 Directional Solver . . . 157
5.3.1 Weak formulation . . . 159
5.3.2 Spatial discretization . . . 159
5.3.3 Variational Multi-scale Method . . . 162
5.3.4 Applications . . . 163
5.3.4.1 Free flow . . . 163
5.3.4.2 Towards contact modeling . . . 165
5.3.4.3 Contact . . . 167
5.4 Conclusion . . . 169
Résumé en Français . . . 170
Bibliography . . . 171

Conclusions **173**
1 Conclusions . . . 173
2 Recommendation and Perspectives . . . 175

List of Figures

1 *Partition of a domain formed by* 10 *deformable bodies using a Lagrangian approach.* 1

1.1 *Mesh deformation when using a Lagrangian approach.* 10
1.2 *Illustration of the multi-scale time step adopted in explicit dynamic* [Bourel, 2006]. 13
1.3 *Improvement of the spatial search in the contact analysis [Yastrebov, 2011].* . . . 15
1.4 *Improvement of the spatial search suggested by [Yastrebov, 2011].* 16
1.5 *Contact spatial search analysis in Forge®.* 17
1.6 *Illustration of the gap function.* 18
1.7 *Construction of contact fictitious elements* 20
1.8 *SPMD strategie for parallel computing used in Forge®.* 21
1.9 *Mesh deformation when using an Arbitrary Lagrangian Eulerian approach.* . . . 28
1.10 *Decoupled resolution using an ALE approach* 30
1.11 *Time gain for different algorithm* 32

2.1 *A sphere immersed in a cubic computational domain* Ω. *The signed distance function isovalues of the central slice are projected onto the bottom of the domain.* 45
2.2 *Comparison between a signed distance function (blue) and a hyperbolic tangent function (red)* 47
2.3 *Different weight of the shape functions using a Galerkin approximation and a Streamline Upwind Petrov-Galerkin* 49
2.4 *Mesh generation by local optimization using "MTC"* 50
2.5 *The length distribution tensor for the node i is determined from the collected edges* 53
2.6 *The anisotropic mesh obtained when using the edge based error estimator metric: (a-b) for a cube, (c) for the upper tool of the connecting rod.* 55

2.7 *Illustration of a 2d flat bunch simulation involving two rigid bodies (tools) and a deformable body.* . . . 57
2.8 *Different Types of mixture laws: (a) P0-mixture law and (b) P1-mixture law* . . 65
2.9 *Comparison between a linear mixture law (blue) and a quadratic mixture law (red)* 66
2.10 *The effect of using a variable time step on the mass conservation..* . . . 68
2.11 *The effect of using P1mixture law instead of a P0 mixture law on the mass conservation.* . . . 68
2.12 *The effect of imposing the velocity on the inner of the dies* . . . 68
2.13 *Flat bunch simulation using a linear coefficient for mixture law.* . . . 70
2.14 *Representation of the mixture law for a flat bunch simulation using different coefficient weights: (a) linear - (b) inverse - (c) geometric - (d) logarithmic* . . . 71
2.15 *Flat bunch simulation using a inverse coefficient for mixture law.* . . . 73
2.16 *Flat bunch simulation using a logarithmic coefficient for mixture law.* . . . 73
2.17 *Top view of the zero iso-value of a flat bunch simulation in 3d showing the mixture law when using: a- linear coefficient, b- logarithmic coefficient.* . . . 74
2.18 *Instability of the levellerT solver.* . . . 74
2.19 *Two different ways to compute the mesh size* . . . 75
2.20 *Comparison between two methods to compute the mesh size in* CIMLib*: (a) using the* $Edge_{min}$ definition and (b) using the $Altitude_{min}$ *definition.* . . . 75
2.21 *A 3d flat bunch simulation after the ameliorations (logarithmic weigh, new mesh size definition ...).* . . . 77
2.22 *Domain considered using a porous upper tool to show the Eulerian approach capacity to detect trapped air* . . . 78
2.23 *Initial mesh and mixture law for the problem in Figure 2.22.* . . . 78
2.24 *The effect given by the imposition of the kinematic on the inner of the upper tool.* 79
2.25 *The evolution of the porous boundary condition in time.* . . . 80
2.26 *The evolution of the zero iso-value of the deformable body when using the porous boundary condition.* . . . 81

3.1 *The Flynn classification of computer architectures: (a) SISD, (b) MISD, (c) SIMD and (d)MIMD.* . . . 92
3.2 *The memory classification of parallel computers: (a) shared memory, (b) distributed memory and (c) hybrid memory.* . . . 94
3.3 *Illustration of a node containing bi-processors with three level of cache memory.* 95
3.4 *Top five super computers june 2013. [www.top500.org,]* . . . 96
3.5 *Different types of speed up when running a parallel simulation : the common speed up (blue); the linear speed up (red); the super-linear speed up (green)* . . . 97
3.6 *Example of partitioning a global mesh in two sub-meshes* . . . 101

3.7 *Local matrix assembly of the two sub-domains: red and blue (a-b) and their projection into the global matrix (c)* . . . 102
3.8 *Physical domain used to study the parallel performance of the monolithical approach.* . . . 105
3.9 *Speed up (a-c) and efficiency (b-d) for the mechanical and transport solvers and different important part of the code obtained on the coarse mesh using the 1.2.6 version of MPI.* . . . 108
3.10 *Speed up (a-c) and efficiency (b-d) for the mechanical and transport solvers and different important part of the code obtained on the coarse mesh using the 1.4.3 version of MPI with the bind to core option.* . . . 110
3.11 *Speed up (a) and efficiency (b) for different part of the code obtained on the coarse mesh using the openmpi round robin option.* . . . 113
3.12 *Using the refined mesh: Speed up (a-c) and efficiency (b-d) for the mechanical and transport solvers and different important part of the code using the 1.4.3 version of MPI with the bind to core option* . . . 115
3.13 *Speed up (a) and efficiency (b) for the anisotropic mesh adaptation technique used in* CIMLib. . . . 116
3.14 *Dynamic load balancing algorithm used in* CIMLib. . . . 117

4.1 *Examples of a connecting rod (a) and a crankshaft (b).* . . . 124
4.2 *Geometry of the connecting rod immersed in the computational domain and adapted using an anisotropic mesh ($t = 0\,s$).* . . . 125
4.3 *The contours of the deformable body and tools presented on a longitudinal section in the thickness at t=0 s* . . . 126
4.4 *From left to right: the isovalue zero of the connecting rod respectively for $t = 0\,s$ and $t = 2.6\,s$.* . . . 126
4.5 *Comparison of the results obtained with* CIMLib *and a Forge® simulation at $t \approx 1.6\,s$. The same longitudinal section in the width is used in both softwares: (a) and (b) illustrate the mesh, (c) and (d) illustrate the pressure, (e) and (f) illustrate the velocity component v_x and (g) and (h) illustrate the velocity component v_y.* . . . 127
4.6 *Comparison of the the velocity component v_z obtained with* CIMLib *(a) and Forge® (b) at $t \approx 1.6\,s$. The same longitudinal section in the thickness is used in both softwares.* . . . 128
4.7 *Geometry of the crankshaft immersed in the computational domain between the tools represented by the anisotropic mesh ($t = 0\,s$).* . . . 129
4.8 *Mesh projected on the zero iso-value of the upper tool* . . . 129
4.9 *Geometry of the crankshaft at $t \approx 0.9\,s$* . . . 130

4.10 *(a), (b) and (c) represent respectively the velocity components v_x, v_y and v_z on the whole crankshaft geometry at $t \approx 0.9\,s$.* . . . 130
4.11 *(a) and (b) represent respectively the contours of the crankshaft between the tools at $t \approx 0.9\,s$ and $t \approx 0.96\,s$. A growing penetration of the upper tool in the deformable piece is noticeable* . . . 131
4.12 *The zero-isovalue of the connecting rod between the tools at $t \approx 1.6s$. No penetration between the different bodies is noted.* . . . 131
4.13 *Initial geometry of the 10 parallelepiped immersed in the computational domain.* 133
4.14 *The uniform mesh of the domain presented on a section: Inside the virtual box, the mesh size is almost ten times smaller than the external one.* . . . 134
4.15 *The deformed geometries when using an isotropic mesh with a mixture law thickness equal to 0.9 mm.* . . . 135
4.16 *The initial mesh adapted anisotropically around the deformable parallelepiped (on the left) with a zoom (on the right).* . . . 135
4.17 *The deformed parallelepiped using an anisotropic adapted mesh. (a), (b), (c) and (d) represent respectively the maps of the velocity components v_x, v_y, v_z and the pressure.* . . . 136
4.18 Forge® *failed attempt to partition the domain containing 148 spheres on 8 cores.* 137
4.19 *From left to right: The partition of a domain containing 148 spheres in Forge® taking into account the above improvment using respectively 4 and 8 cores.* . . . 138
4.20 *From left to right: The partition of a domain containing 148 spheres in* CIMLib *using respectively 4 and 8 cores.* . . . 138
4.21 *Initial partition of a domain containing 256 spheres on 64 cores.* . . . 139
4.22 *Velocity components v_x and v_z are respectively represented on a longitudinal section in the thickness in (a) and (c). Velocity component v_y represented on a longitudinal section in the width in (b) ($t = 0.64\,s$).* . . . 140

5.1 *Graphic representation of the different friction laws.* . . . 147
5.2 *Schematic representing the boundary layer.* . . . 148
5.3 An illustration of the initial ring geometry along with the symmetry planes . . . 150
5.4 The deformed ring geometry at $t = 3.467\,s$ . . . 151
5.5 *(a) illustrates the mixture law on a cutting plane in the thickness. (b) to (e) represent the different velocity components and magnitude. (f) illustrates the pressure repartition. The different maps are presented on a forth of the ring at $t = 3.46\,s$.* . . . 152
5.6 The deformed ring outline *for* $y = 0$ at $t = 3.467\,s$ . . . 153

5.7 *Comparison of the velocities obtained with sticking friction in both Forge® and* CIMLib *on a cutting plane in the thickness for* $y = 0$ *at* $t = 3.46\,s$ *: (a) and (b) compare the velocity* v_x *plotted on the outer outline using two different mixture thickness* $\varepsilon = 0.5\,mm$ *(on the left) and* $\varepsilon = 0.25\,mm$ *(on the right). (c) compares the velocity* v_z *plotted on the outer outline using a mixture thickness* $\varepsilon = 0.25\,mm$*. (d) superposes the velocity* v_x *plotted on the inner outline of the ring using* $\varepsilon = 0.25\,mm$*.* 154

5.8 *Comparison of the velocities obtained with sliding friction in both Forge® and* CIMLib*. Curves are plotted at* $t = 3.46\,s$ *for* $\alpha_f = 0.3$ *on a cutting plane in the thickness for* $y = 0$ *: (a) and (b) compare the velocity* v_x *respectively plotted on the inner and outer outline using a mixture thickness* $\varepsilon = 0.25\,mm$*. (c) and (d) compares the velocity* v_z *respectively plotted on the inner and outer outline using a mixture thickness* $\varepsilon = 0.25\,mm$*.* 155

5.9 *(a) and (c) present a comparison of the velocities* v_x *obtained with sliding friction in both Forge® and* CIMLib *using respectively* $\varepsilon = 0.25\,mm$ *and* $\varepsilon = 0.5\,mm$ *. Curves are plotted at* $t = 3.46\,s$ *for* $\alpha_f = 0.03$ *on the outer outline of a cutting plane in the thickness for* $y = 0$*. (b) and (d) present a zoom of the quadratic mixture law using respectively* $\varepsilon = 0.25\,mm$ *and* $\varepsilon = 0.5\,mm$ *.* 156

5.10 *Initial position of the anisotropic material.* 163

5.11 *Maps of the velocity components* v_x *and* v_y *at* $t \approx 1\,s$ 164

5.12 *Side and upper views of the final position of the material corresponding to* $t \approx 2.1\,s$*.* 164

5.13 *The computational domain formed by two phases: the deformable body and the lubricant.* 165

5.14 *The impact of a variant lubricant anisotropic consistency on the velocity profile* v_x*.* 166

5.15 *At* $t = 0\,s$*, a superposition between the anisotropic consistency (on the left) and the consistency using the quadratic mixture law (on the right).* 168

5.16 *At* $t = 0.014\,s$*, a superposition between the material flow (illustrated by arrows) using an anisotropic consistency (on the left) and a quadratic mixture law (on the right).* 168

List of Tables

1.1 *Efficiency of the contact analysis algorithm in Forge®* 22

2.1 *Parameters used in the flat bunch simulation* 67
2.2 Comparison between the different definitions to compute the simplex size. . 76

3.1 *The simulations parameters used for speed up tests* 106
3.2 *Time (in s) spent to assemble, solve the mechanical problem with two different solvers (Navier-Stokes vs Stokes) using the 1.2.6 version of MPI* 106
3.3 *Time (in s) spent to assemble, solve the mechanical and transport solver and to accomplish the simulation using the 1.2.6 version of MPI* 107
3.4 *Time (in s) spent to accomplish other relevant part of the code during simulation using the 1.2.6 version of MPI* 108
3.5 *Time (in s) spent to assemble, solve the mechanical and transport solver and to accomplish the simulation using the 1.4.3 version of MPI with the bind to core option* 109
3.6 *Time (in s) spent to accomplish other relevant part of the code during simulation using the 1.4.3 version of MPI with the bind to core option* 109
3.7 *Size of the Output files in parallel computing* 111
3.8 *Time (in s) spent to assemble, solve the mechanical and transport solver and to accomplish the simulation using the 1.4.3 version of MPI with the opemmpi round robin option* 112
3.9 *Time (in s) spent to accomplish other relevant part of the code during simulation using the 1.4.3 version of MPI with the openmpi round robin option* 112
3.10 *Using the refined mesh: Time (in s) spent to assemble, solve the mechanical and transport solver and to accomplish the simulation using the 1.2.6 version of MPI* 114

3.11 *Using the refined mesh: Time (in s) spent to accomplish other relevant part of the code during simulation using the 1.4.3 version of MPI with the bind to core option* 114
3.12 *Times (in s) spent to accomplish the anisotropic mesh adaptation and its different parts.* 116

Introduction

This present work is devoted entirely to modeling multi-domain large deformation problems using a monolithic Eulerian approach in a massively parallel environment.

Generally, these type of problems are rather treated using Lagrangian or Arbitrary Lagrangian Eulerian (ALE) formulations (check chapter 1). Lagrangian approaches are spread widely and have almost perfect management for large deformation problems. But when it comes to multi-domain parallel deformation problems, they often show some shortcoming points. These imperfections are usually caused by the contact search algorithms affecting the scalability of the whole parallel simulation (see discussion in chapter 1).

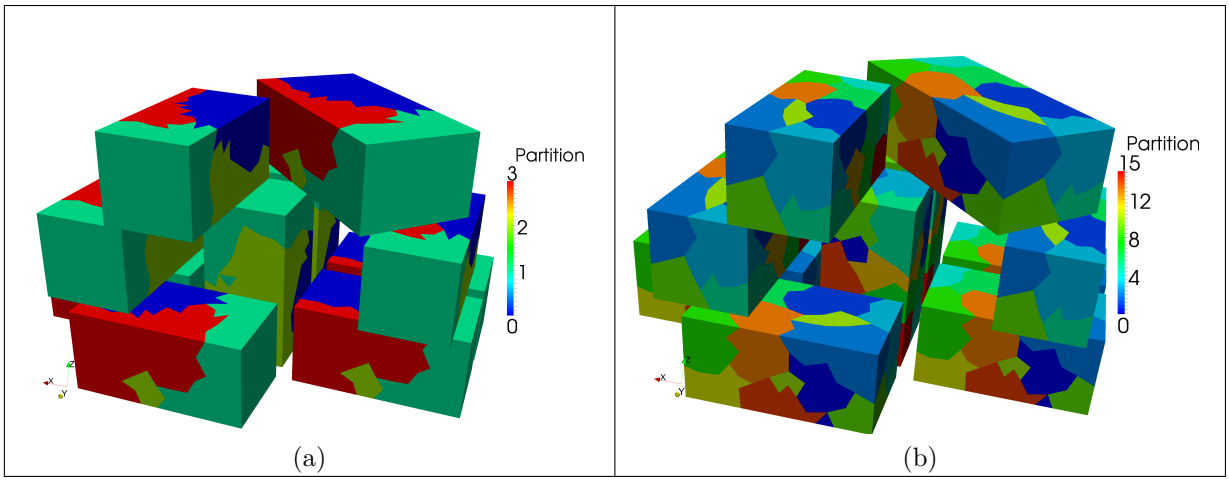


Figure 1: *Partition of a domain formed by* 10 *deformable bodies using a Lagrangian approach.*

If one considers the contact problem between 10 deformable bodies treated on 16 cores (illustrated in Figure 1), the computation domain associated to each core is no longer a physical domain. It is rather a numerical domain containing parts of each deformable body. This means that every time contact search algorithm needs to identify a node in

contact, communication between different cores is absolutely required. In other words, the communication cost is directly dependent of both the number of cores and interacting bodies.

In Cemef, we dispose of a library based on an Eulerian formulation fully functional in a parallel environment. In this context, our approach relies on a single meshed domain where different bodies are immersed and identified via the Level Set method. The different bodies are no longer individual domains but rather heterogeneity in the global computational one. We should note that one single mechanical set of equations is solved all over the domain. From the parallel point of view, it is expected to show great scalability even when using hundred cores since it depends of one unique mesh. In addition. this approach represents the advantage of not requiring any contact search algorithm. It is managed automatically via the Level Set method (see chapter 2).

Now that all has been said, we propose using this Eulerian monolithic approach to try and prevent the previously mentioned problems which leads us to the motivations of this work.

1 Industrial Motivation and Aim of this study

Transvalor, house of Forge2 and Forge3, owns in partnership with Armines a C++ Library developed in Cemef called CIMLib developed by [Digonnet and Coupez, 2003], [Digonnet et al., 2007], [Mesri et al., 2009]. This library is based on an Eulerian description and is fully parallelized.

CIMLib is mostly used for computational fluid dynamics and it was never been used for large deformation problems (treated normally using Lagrangian formulations). To prevent difficulties confronted in Lagrangian approaches such as the ones mentioned earlier (multi-domain large deformation problems, the complicated search contact algorithms, etc.), an Eulerian approach can offer some insight and eventually solutions for these problems.

Transvalor, interested in the capacity of this monolithic Eulerian approach considers utilizing this library for application similar to the ones carried-out in Forge. To do so, a proposition for the present thesis was conceived.

Though this monolithic Eulerian approach acquires impressing potential , it was never applied to these kind of problems before. Thus, it requires further developments and manipulations to be fully operational. So, the aim of this work can be summarized by the following headlines:

◇ Study the feasibility of multi-domain deformation problem using an Eulerian approach in a massively parallel environment.

◇ Identifying along this study several points such as where the approach excels and

when it falls short.

◇ Proposing improvements and new developments to enable applications to large deformation problems

In other words, the fundamental objective is to *i)* formulate a general idea if this monolithic approach is fit for large deformation problems, *ii)* to offer further needed developments and *iii)* to conclude if it shows promise to be industrialized on the long haul.

2 The layout of this thesis

This thesis is divided into five main chapters.

◇ In chapter 1 a litterature survey for modeling large deformation problems is presented (Lagrangian, Eulerian and Arbitrary Lagrangian Eulerian formulations). It focuses mainly on previous works using Lagrangian formulations evoking both advantages and difficulties (contact search algorithms, mesh distorsions,..). The latter justify our choice leading to utilizing an Eulerian approach.

◇ In chapter 2, the CIMLib library [Digonnet and Coupez, 2003] developed in a C++ parallel Eulerian framework is presented. First the Eulerian formulation, basis of this library, is detailed. Then, the different needed tools are introduced : from the level set method and the front convection, to the mixture laws and the anisotropic adapted mesh.
Last but not least, several tips and improvements are proposed such as introducing new mixture laws, new boudary conditions to treat air trapping and new definitions to handle some numerical instabilities. In addition, direct applications are offered to help readers understand how this existent approach can be adapted to modelling large deformation problems.

◇ In chapter 3 the parallel enviroment in CIMLib is detailed such as the adopted parallel strategy, the solvers parallelization and the mesh partitionning. A full study dealing with the performance and scalabillity of different parts of the codes is presented. The overall performance was found satisfactory.

◇ In chapter 4, advanced applications are displayed using all the previous developpment combined. Applications are divided into two halfs. The first models large deformations of complicated industrial pieces along with a comparison with Forge simulations. The second half is mainly dedicated to multi-domain deformation problem in a massively parallel enviroment. The approach performance is tested for upto 250

deformable bodies using couple hundreds of cores. We take advantage as well to point out some difficulties to be improved.

◇ In chapter 5 the reasearch is pushed further by investigating the feasability of several concepts usefull to large deformation problems.
A first attempt to introduce friction notion using an Eulerian description is detailled. Direct applications and comparison with Forge simulation are presented. A new directional solver is developed as a first pursue to simulate contact resolution in an Eulerian formulation.
The different results and limitations of the model developments are discussed. This offers areas of further development along with ways to improve the proposed models for new research topics (not necessarly directly related to this present study)

◇ Finally, general conclusions, perspectives and recommendations for futur works are provided.

Bibliography

[Digonnet and Coupez, 2003] Digonnet, H. and Coupez, T. (2003). Object-oriented programming for fast and easy development of parallel applications in forming processes simulation. In *2nd MIT Conference on Computational Fluid and Solid Mechanics*, page 1922.

[Digonnet et al., 2007] Digonnet, H., Silva, L., and Coupez, T. (2007). Cimlib: a fully parallel application for numerical simulations based on components assembly. In *MATERIALS PROCESSING AND DESIGN; Modeling, Simulation and Applications; NUMIFORM'07; Proceedings of the 9th International Conference on Numerical Methods in Industrial Forming Processes*, volume 908, pages 269–274. AIP Publishing.

[Farhat et al., 1998] Farhat, C., Chen, P.-S., Mandel, J., and Roux, F. X. (1998). The two-level {FETI} method part ii: Extension to shell problems, parallel implementation and performance results. *Computer Methods in Applied Mechanics and Engineering*, 155(1-2):153 – 179.

[Mesri et al., 2009] Mesri, Y., Digonnet, H., and Coupez, T. (2009). Advanced parallel computing in material forming with cimlib. *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique*, 18(7-8):669–694.

CHAPTER 1

State of the art

Contents

1.1 Introduction **9**

1.2 Lagrangian Description **9**

1.2.1 Spatial discretization 11

1.2.2 Temporal discretization 12

1.2.3 Contact 13

1.2.3.1 Contact search algorithms 14

1.2.3.1.1 All-to-All algorithm 15

1.2.3.1.2 Bucket sort algorithm 16

1.2.3.1.3 Spherical sorting algorithm 16

1.2.3.2 Contact detection 17

1.2.3.2.1 Contact between rigid and deformable bodies. 18

1.2.3.2.2 Contact between deformable bodies. 19

1.2.3.3 Parallel strategies in contact detection 20

1.2.3.4 Contact resolution 22

1.2.3.4.1 *Lagrange Multiplier Method* 22

1.2.3.4.2 *Penalty Technique* 23

1.2.3.4.3 *Augmented Lagrangian Method* 23

1.3 Eulerian Description **24**

1.3.1 Interface modeling 25
1.3.2 Contact formulation 25
1.4 Arbitrary Lagrangian Eulerian Description 27
1.4.1 Coupled Resolution 29
1.4.2 Split operator 29
1.5 Discussion and Conclusion 31
Résumé en Français 33
Bibliography 34

1.1 Introduction

Modeling large deformation problems, such as forging and rolling processes, is one of the complicated problems in solid mechanics.

Their complexity requires a great care to details, specially the consideration of all interacting aspects in order to insure the precision of the simulation. We cite important aspects such as the material and geometrical non-linearities, the good manipulation of the contact and friction problems. For all these reasons, the use of numerous numerical techniques is a must.

This subject has been widely discussed in the literature. A survey shows that the Lagrangian description is the most used dealing with such problems. The Arbitrary Lagrangian Eulerian (ALE) description comes in second. Finally, the Eulerian approach is rarely used for this type of problems (in solid mechanics). It is due to the lack of works presenting complete models dedicated to large solid deformation problems. Actually, the Eulerian approach is usually used in fluid mechanics.

In the present chapter, we will present

- A quick description of the Lagrangian framework focusing on the contact algorithm.
- A brief overview of the Eulerian formulation. Note that this part will not be discussed in detail since chapter 2 will be entirely dedicated for it.
- A small review of the Arbitrary Lagrangian Eulerian description.
- A discussion summarizing the relevant points and justifying our choice selecting the Eulerian approach.

1.2 Lagrangian Description

The Lagrangian description of a deformable moving body consists on identifying the deformation of a material point (on an instant t) from its initial position (on instant t_0).

If a deformable body is occupying the domain Ω_0 in its reference configuration and Ω_t in its current one (See Figure 1.1), a point deforming with the body is called:

- M_0 at $t = 0$ and occupies the position $X = (X_1, X_2, X_3)$
- M_t at t and moves to the position $x = (x_1, x_2, x_3)$

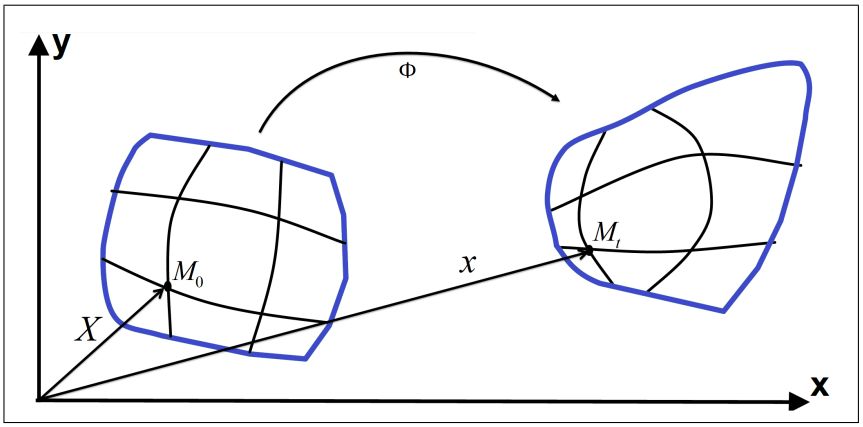


Figure 1.1: *Mesh deformation when using a Lagrangian approach.*

The new position x is defined as a function of X and t:

$$\begin{array}{rccl} \phi: & \Omega_0 \times T & \rightarrow & \Omega_t \\ & (X,t) & \rightarrow & \phi(X,t) = x \end{array} \tag{1.1}$$

where ϕ is continuous and bijective. Its inverse $\Phi = \phi^{-1}$ is continuous and bijective as well.

The function Φ exists and is bijective means that two distinct points initially remains distinct through the entire problem. Using Φ the position x can be computed as follows:

$$\begin{array}{rccl} \Phi: & \Omega_t \times T & \rightarrow & \Omega_0 \\ & (x,t) & \rightarrow & \Phi(x,t) = X \end{array} \tag{1.2}$$

In the Lagrangian description, all physical quantities (position, velocity, density, strain, stress, etc.) are expressed as functions of X and t:

The displacement or position vector is written as

$$OM_t = x = \phi(X,t) \quad \text{where} \quad \phi(\text{X},0)=\text{X} \tag{1.3}$$

The velocity of M_t takes the following form:

$$v(X,t) = \frac{\partial x}{\partial t} = \frac{\partial \phi(X,t)}{\partial t} \tag{1.4}$$

The same goes for the acceleration of M_t:

$$\gamma(X,t) = \frac{\partial V}{\partial t} = \frac{\partial^2 \phi(X,t)}{\partial t^2} \tag{1.5}$$

To finish this section, one should at least present the constitutive equations to be solved in Lagrangian description:

$$\begin{array}{lll} \rho\dfrac{\partial v}{\partial t} + div\sigma = 0 & & \text{dynamic equilibrium equation} \\ div(v) = 0 & & \text{continuity equation} \\ \text{behaviour law} & + & \text{boundary conditions} \end{array} \tag{1.6}$$

Remarks and notations:

X, called the material variable, is used in Lagrangian description.
x, called the spatial variable, is used in Eulerian description.
All quantities, for instance F, can be expressed as functions of X and t in Lagrangian description or as function of x and t in Eulerian description:
$F = F(X,t)$ value of F experienced by the initial particle of position X at instance t (Lagrangian description).
$F = F(x(X,t),t)$ value of F experienced by the particular of position x instantaneously (Eulerian description).

$$\frac{dF}{dt} = \left[\frac{\partial F}{\partial t}\right]_X = \frac{\partial F}{\partial t} + \frac{\partial F}{\partial x}\frac{\partial x}{\partial t} = \frac{\partial F}{\partial t} + (v.\nabla)\,F$$

This derivative will simply help transforming equations from Lagrangian description to the Eulerian description once needed in sec. 1.3.

1.2.1 Spatial discretization

To describe the deformation of a body, it is necessary to characterize the variations of lengths and angles. Since an analytical solution is quite impossible to find, a weak formulation is required. The transition to the weak formulation resort in introducing Sobolev spaces. Then the weak formulation is obtained by multiplying the equilibrium equation (1.6) by the test function and integrating the product over the entire area in the current configuration Ω_t:

Find $(v,p) \in \mathcal{V} \times \mathcal{P}$

$$\begin{cases} \int_{\Omega_t} \rho\dfrac{\partial v}{\partial t} w + \int_{\Omega_t} s(v) : \varepsilon(w)\,dV - \int_{\Omega_t} p div\, w\,dV + \int_\Gamma \tau.w dS = 0 \\ \int_{\Omega_t} q div\, v dV = 0 \\ \forall (w,q) \in \mathcal{V}_0 \times \mathcal{P} \end{cases} \tag{1.7}$$

where s represents the deviatoric part of Cauchy stress tensor, τ the friction contribution and:

$$\mathcal{V} = \{v \in (H^1(\Omega))^3 / (v - v_r).n = 0 \, over \, the \, contact \, surface\}$$
$$\mathcal{V}_0 = \{v \in (H^1(\Omega))^3 / v.n = 0 \, over \, the \, contact \, surface\}$$
$$\mathcal{P} = \{p \in L^2(\Omega)\}$$

1.2.2 Temporal discretization

When solving the system of equations (1.7), the choice of the time integration scheme varies mostly between an explicit scheme or an implicit scheme. By adopting an explicit scheme, the numerical system is the following at the increment i:

$$\mathbb{M}^i a^i + F^i_{int} = F^i_{ext} \tag{1.8}$$

where $\mathbb{M}$, the mass matrix, is diagonal, F_{int} is the internal vector forces and F_{ext} is the external vector forces. Using the former scheme, the stiffness matrix is updated at the end of the incremental procedure. Furthermore the stiffness matrix obtained by adopting such scheme is diagonal, therefore the resolution of the numerical system is often given by the inversion of $\mathbb{M}$. Moreover, no convergence loop is needed because the resolution is directly. This method is well suited for fast dynamics (shocks, crash, explosion...) where the physical phenomenon is very violent and induces strong non linearities over a short time. However, this scheme needs many small increments for good accuracy and it is time consuming. Whereas, implicit schemes have interest in slow dynamics (such as vibration in structure problems...). The stiffness matrix is no longer diagonal.This scheme allows the use of bigger time steps. However in non-linear problems, Newton-Raphson (N-R) algorithm is often needed to insure equilibrium. This leads to one important drawback which is reconstructing the stiffness matrix at each N-R iteration.

To overcome the restriction on the time step when using an explicit scheme, some works as in [Bourel, 2006] propose to couple the multi-domain decomposition technique with a multi-scale time step. When adopting the latter technique, the time steps used are different on each sub-domain. Since the equilibrium equations are locally verified at each sub-domain at different times, the challenge is to develop a method capable of managing the exchange among sub-domains, even if they are never at equilibrium in the same time.

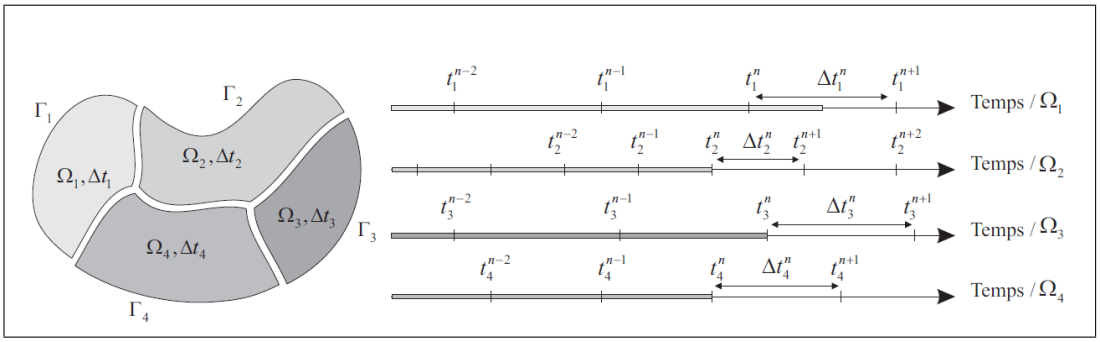


Figure 1.2: *Illustration of the multi-scale time step adopted in explicit dynamic* [Bourel, 2006].

1.2.3 Contact

In large deformation problems, contact algorithms have great impact on both solution quality and computational cost. In fact for such type of problems, the interacting geometries deform drastically leading to significant changes in the contact surface. The authors [Barboza, 2004] and [Yastrebov, 2011] recommend to update the contact surface at each time step. In [Mocellin, 1999], the contact surface is updated at each Newton-Raphson iteration. Thus, the contact problem should be treated carefully. In other words, a poor treatment of the contact (e.g. non-smooth contact surface discretization) yields to an undesired numerical solution. In addition, as the number of nodes increases; the detection of nodes in contact affects the computational cost of the simulation specially when the contact surface should be updated frequently.

In what follows, we concentrate on frequently used numerical techniques dealing with contact problems. Theoretical aspects related to the existence and the uniqueness of the contact problem can be found in [Benson, 1992, Martins and Oden, 1987] and [Sofonea and Matei, 2012]. A large number of reviews on contact algorithms related to solid mechanics are presented in [Aliabadi and Brebbia, 1993, Laursen, 2002, Hughes, 2012]. Additional publications devoted to investigate contact algorithms applied to metal forming can be found in [Chenot et al., 2002], [Fourment et al., 1999] and [Hachani and Fourment, 2013].

As mentioned earlier, when using a Lagrangian formulation the boundaries of a body are represented by a set of boundary cells and nodes. In other words, the contact is highly dependent of the moving mesh which brings us to the main objective of this section.

In every contact problem, we can distinguish two major parts. The first one is devoted to contact detection in which a spatial search retrieving nodes in/or may possibly go in contact is established. Note that in most large deformation problems, the contact surface cannot be predicted and has to be detected at each time step. The second part is related to the resolution of contact problem in which several schemes can be used . It aims to eliminate the penetration between different bodies by applying repulsive forces.

In a large number of reviews, contact takes place between a rigid body and a deformable one. Note that in a Lagrangian framework, one should take into consideration self contact e.g. different parts of the same body are in contact, which is the case in a large deformation problems. Publications in this direction can be found in [Oliveira et al., 2008]. In addition, multiple contact problems can occur in many applications containing multi-body geometries. Work investigating the multiple contact problem can be found in [Chenot et al., 2002] , [Barboza, 2004] and [Habraken and Cescotto, 1998].

1.2.3.1 Contact search algorithms

Numerical treatment of contact problems confront many difficulties. The algorithms needed can consume a great part of the computational time and require in general a significant computer resources. These difficulties are related to three different topics: the geometry discretization, the contact detection and the spatial search.

The literature provides three different type of geometry discretization. The simplest method is called node to node discretization [Francavilla and Zienkiewicz, 1975]. It is recommended in case of small deformation problems. In fact, this technique consists on forming contact pair nodes. These pairs are formed by nodes belonging to a surface matched with nodes on the opponent surface. Once contact nodes are defined, it does not change during the time step. The construction of contact pairs and the assumption taken in consideration does not allow large deformation of the mesh. The second technique is called node to segment discretization. It demands more effort put into the implementation. Contact elements are formed by first searching for the closest nodes in two surfaces (the slave and the master). Then, the slave node forms with the segment containing the corresponding node of the master surface a contact element. The final technique is the segment to segment discretization. It requires even more effort in implementation since detection algorithms based on surface topologies are needed.

In large deformation problems, the node to segment discretization is frequently used since it is well adapted for a good representation of sliding problems. In other words, it is easier to represent sliding between a node and a segment then between two nodes or two segments [Fourment et al., 2003].

Next, we will address the contact algorithms. As mentioned earlier, contact can occur at non-predectible surfaces. If an update is needed at each time step, which is always the case in our problem, then it is necessary to insure the efficiency of these algorithms. Generally, in every contact algorithm, two phases are distinguished : the global algorithm representing the spatial search and the local one representing the contact detection [Wriggers, 2002].

Among the spatial search algorithms, we cite the *all-to-all* algorithm, the *bucket sort* algorithm and the *spherical sorting* algorithm.

1.2.3.1.1 All-to-All algorithm is one of the first contact searching algorithm. It is easy to implement. However for problems with great numbers of nodes, this algorithm presents an essential drawback. The number of operations required to identify the contact pairs is proportional to the square of the number of boundary elements or nodes [Bourago and Kukudzhanov, 2005]. To reduce the computational time needed to identify the contact surface, several optimizations can be found in the literature. The main idea of these optimizations is to introduce a parameter d_{max} - called maximal distance -. It is used to determine which nodes are relevant for identifying the contact surface e.g. all nodes lying at a distance greater than d_{max} are not considered among the nodes involved in contact analysis. The choice of d_{max} is discussed in [Yastrebov, 2011]. A poor choice of d_{max} leads to a wrong contact surface definition as shown in Figure 1.3 (left part). According to [Yastrebov, 2011] work, the d_{max} should be computed as:

$$d_{max} = max \left\{ \underset{i=1,j=1,k=j+1}{max^{i=N_m,j=N_n^i-1,k=N_n^i}} \left| r_j^i - r_k^i \right| ; 2max_{i=1}^{N_c} |\Delta r_i| \right\} \tag{1.9}$$

where N_m is the number of the master segments, N_n^i is the number of master nodes forming the $i-th$ master segment, N_c is the total number of slave and master nodes and r_i is the coordinate of the $i-th$ node and Δr_i its displacement (Figure 1.3). In case of re-meshing, it is recommended to compute d_{max} after each re-meshing step.

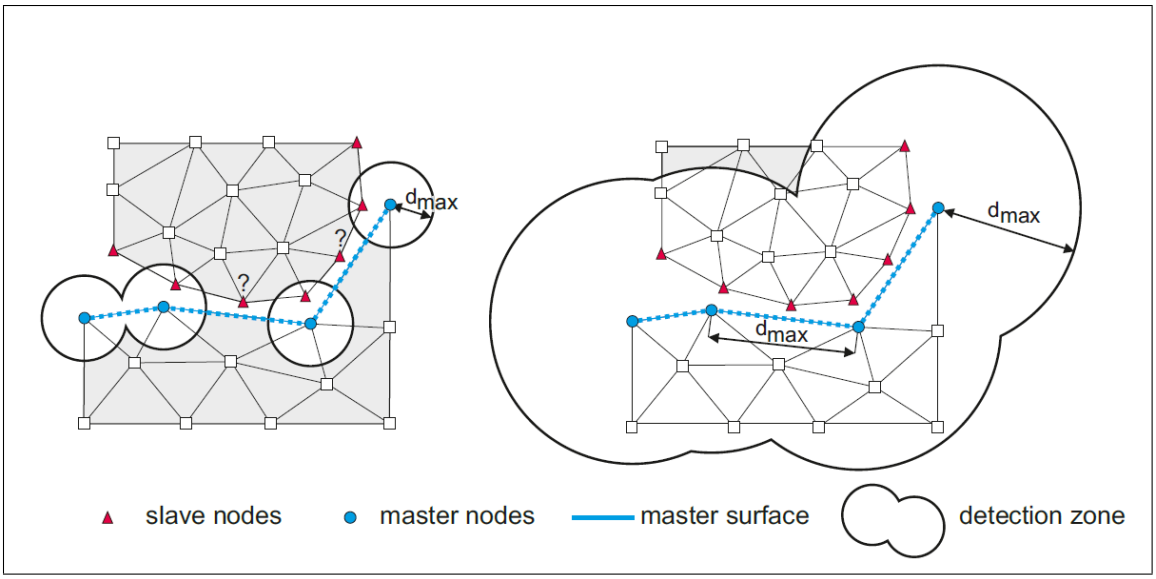


Figure 1.3: *Improvement of the spatial search in the contact analysis [Yastrebov, 2011].*

Even with the optimization proposed above, the number of nodes for the contact analysis remains significant. An additional improvement was brought by [Yastrebov, 2011]. The main idea is to determine a bounding box on each master and slave boundary surface. Then the nodes taken into consideration in the contact detection are reduced to the intersection of the two bounding boxes (see Figure 1.4). More details about the

construction of these bounding boxes can be found in [Yastrebov, 2011].

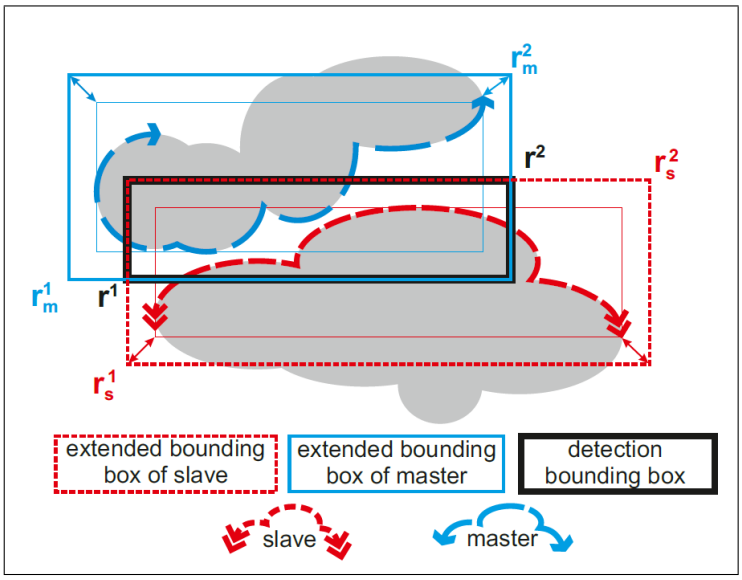


Figure 1.4: *Improvement of the spatial search suggested by [Yastrebov, 2011].*

1.2.3.1.2 Bucket sort algorithm is an alternative technique used for spatial search. It was proposed and applied for modeling self-contact surfaces in [Benson and Hallquist, 1990] and [Belytschko and Neal, 1991]. Several works relying on this algorithm can be found such as the works of [Heinstein et al., 2000] and [Paik et al., 2006]. Each of the previous cited works adapted and presented new variant of the algorithm responding to their own needs. In what follows, only the technique described in [Yastrebov, 2011] will be presented briefly.

Similar to the all-to-all algorithm, the preliminary phase is to define an appropriate d_{max}. The author recommends to choose the biggest segment of the master surface. Then the bounding boxes are defined. Finally an internal bucket grid should be created, where the size w of each bucket should be chosen properly. For instance, w should be greater than $\sqrt{2}d_{max}$ when using linear elements. If the bucket size is smaller, contact spatial search can be mistreated. In this case, one should consider treating the contact detection on several bucket layers.

1.2.3.1.3 Spherical sorting algorithm was used in [Papadopoulos and Taylor, 1993]. This algorithm is used as well in *Forge*® for the contact spatial search. To identify areas where contact may occur, several spheres are created. The latter are defined for every boundary element. Its center is the barycenter of the element and its radius is the biggest distance connecting the barycenter with the nodes of the elements.

Once all the sphere are determined, the set of neighbor spheres is transformed into a new bigger sphere covering the whole set. This step can be repeated several times

depending on the complexity of the geometry. The initial spheres are called the first-level-spheres, the bigger spheres englobing them are called the second-level-spheres and so on. *Forge*® considers only three spheres levels (see Figure 1.5).

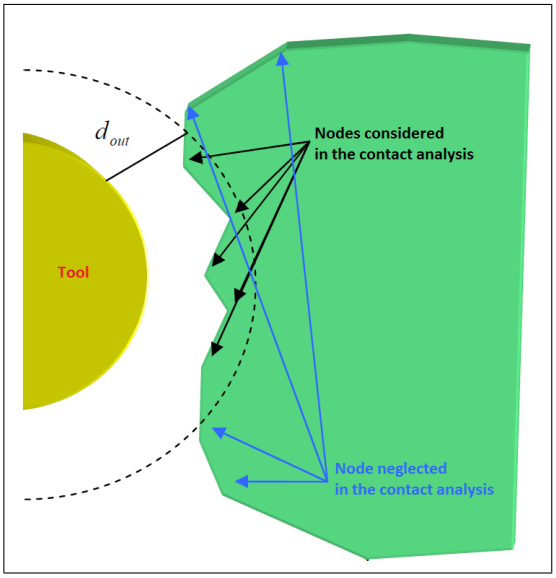


Figure 1.5: *Contact spatial search analysis in Forge*®.

Once this hierarchy is established, the search algorithm consists on finding for every node (belonging to the slave surface) the closest third-level-sphere. The selected sphere englobes smaller ones. So another search is need to determine the closest level-two-sphere. The latter is formed by the surface elements of the master. This means that the set of the closest surface elements is known. The last step consists on defining the closest element belonging to this set.

1.2.3.2 Contact detection

After having found the areas where contact may occur, the local search is performed. Moreover, in contact detection algorithms, one should distinguish a contact between rigid and deformable body from contact between two deformable bodies. Several algorithms can be used to accomplish this detection such as the pinball algorithm [Belytschko and Neal, 1991] and the gap function [Bathe and Brezzi, 2001], [El-Abbasi and Bathe, 2001]. Other algorithms are inspired from computer graphics as the aircraft's shadow projection method. According to [Yastrebov, 2011], the latter will eliminate some drawbacks of the gap function.

Herein we present briefly a description of the gap function used in *Forge*®. Recall that a master-slave formulation is put in place to describe the contact dynamics.

1.2.3.2.1 Contact between rigid and deformable bodies. Let us denote by Ω_r the rigid body and by Ω_d the deformable body. Γ_r and Γ_d represent respectively the set of the boundary nodes of the rigid and deformable body. We begin by introducing the contact zone as an intersection between the boundary of two distinct bodies:

$$\Gamma_c = \Gamma_r \cap \Gamma_d \tag{1.10}$$

Moreover, the contact should fulfill the no-penetration condition and the equilibrium condition on the interface. Those conditions are expressed by equations (1.11) and (1.12).

$$\Omega_r \cap \Omega_d = 0 \tag{1.11}$$

$$\sigma_{n1} + \sigma_{n2} = 0 \quad with\, \sigma_k = \sigma_k n_k \, \forall k \in \{1, 2\} \tag{1.12}$$

The two previous conditions when combined provide Signorini law which determine the type of contact between the different domains. For the theoretical aspects, the readers can refer to [Wriggers, 2002]. The simplest way to express the no-penetration condition is by introducing the gap function :

$$g(x_2, t) = \min_{x_1 \in \Gamma_1} ||x_2 - x_1|| sign(n_1.(x_1 - x_2)) \leqslant 0 \quad \forall x_2 \in \Gamma_2 \tag{1.13}$$

where n_1 is the unit outward normal at the surface Γ_1(see Figure 1.6).

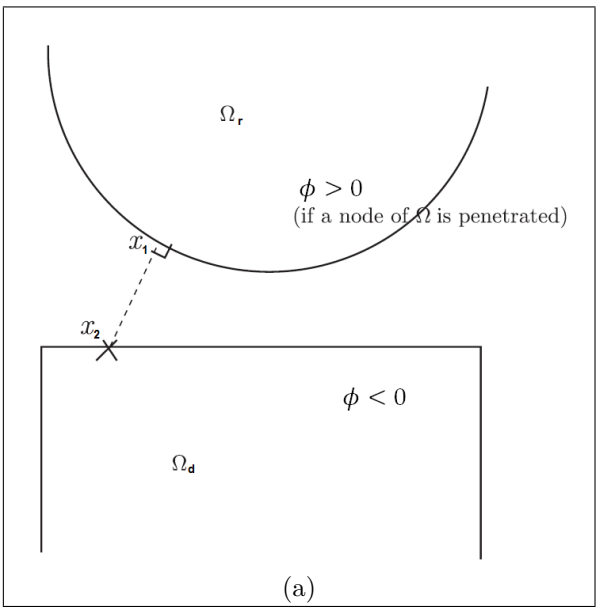


Figure 1.6: *Illustration of the gap function.*

1.2.3.2.2 Contact between deformable bodies. In many applications, specially in a multi-domain cases, multiple contact between deformable bodies occur. A broad range of algorithms allowing to fulfill the task can be found in the literature [Flickinger et al., 2013], [Hahn and Wriggers, 2003] and [Iwai et al., 1999]. Similar to the contact between rigid and deformable body, a contact graph is created. A review of different algorithms and implementations are found in [Erleben, 2004]. Note that in multi-domain algorithms, nodes do not always represent a physical entity. For instance, it can be a virtual entity used to trigger the contact. These nodes will form fictitious elements used for contact analysis. The latter concept is applied in *Forge*®.

Herein a brief description is presented. For more details, the readers can refer to [Barboza, 2004]. The following algorithm can be considered as a generalization of the contact analysis between a rigid and a deformable body. The global search relies on the spherical sorting algorithm presented above.

As for the local detection algorithm, let Ω_k and $\Omega_{k'}$ denote two different deformable bodies, where Γ_k and $\Gamma_{k'}$ represent respectively their boundaries. For each node x_k of the slave surface, the contact element is constructed as follows:

$$\begin{array}{rccc} \pi: & \partial\Omega_h^k & \longrightarrow & \partial\Omega_h^{k'} \\ & x_k & \longmapsto & x_{k'} = \pi(x_k) \end{array}$$

where π is the orthogonal projection. The projection determines the location of $x_{k'}$ on the master surface e.g. $x_{k'}$ belonging to which element. The position of $x_{k'}$ is computed using the element's barycenter and nodes $\{A^1, A^2, A^3\}$ (see Figure 1.7). The distance between the two bodies is given by :

$$g_n(x_k) = |x_k x_{k'}| \tag{1.14}$$

$$n(x_k) = \begin{cases} -\dfrac{1}{g_n(x_k)}\overrightarrow{x_{k'}x_k} & if\ g_n(x_k) \neq 0 \\ n_{F^{k'}} & if\ g_n(x_k) = 0 \end{cases} \tag{1.15}$$

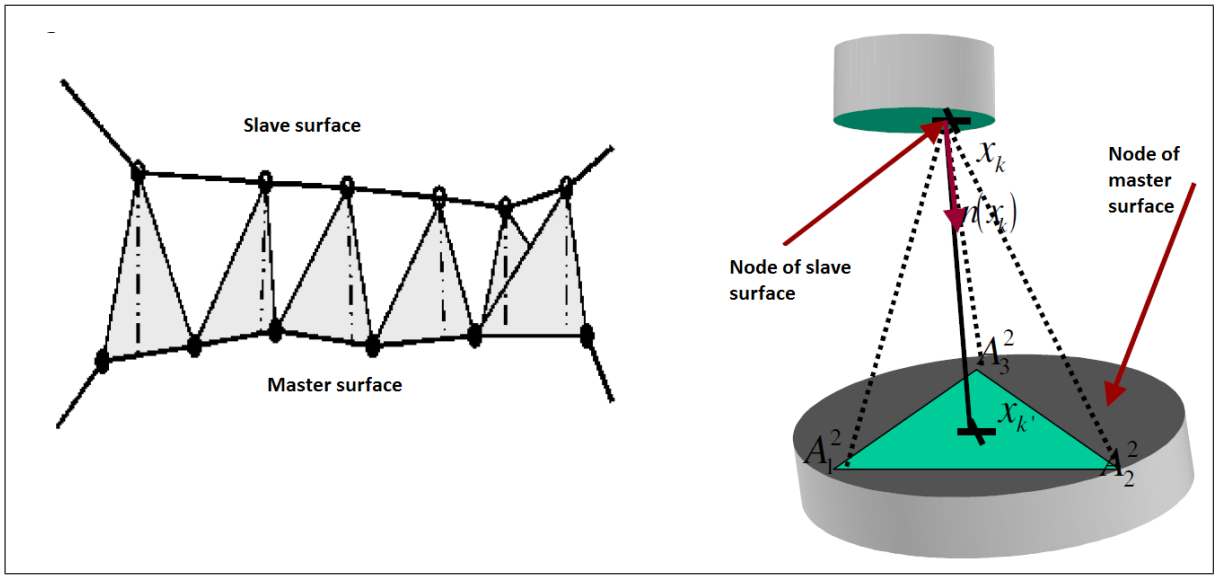


Figure 1.7: *Construction of contact fictitious elements*

Before proceeding into the parallel strategies, we would like to mention that the literature proposes alternatives methods to make solving large contact problems a possibility. We cite for instance the domain decomposition method presented in [Sassi et al., 2008] [Roux and Garaud, 2009]. Many variants can be found also as in [Farhat and Mandel, 1998, Farhat et al., 1998]. In general the computational domain is divided into non-overlapping sub-domains. A part of the solution is evaluated on a sub-domain then a gluing phase is needed to enforce field continuity [Roux et al., 2005]. These decomposition algorithms make solving large problem a possibility without resorting to parallel algorithm.

1.2.3.3 Parallel strategies in contact detection

Nowadays, most problems are oriented towards industrial applications which sometimes require complex geometries. The latter will involve a large number of contact nodes.

Even with all simplifications introduced to replace a full scale contact detection, the execution time of such algorithms remains important. To optimize such algorithms, developers used parallelism as a mean to decrease the required computed time.

In what follows, the parallel strategy used in *Forge*® is presented briefly. The focus will be on the generation of fictitious elements in multi-domain cases. For sake of clarity, two deformable bodies Ω^1 and Ω^2 are taken into account next. To begin with, we should mention that the parallelism of contact search algorithm relies on a S.P.M.D strategy (see chapter 3 for more details). Every body is partitioned into n_p sub-domains :

$$\Omega^1 = \bigcup_{p=1}^{n_p} \Omega_p^1 \ , \ \Gamma_{pl}^1 = \Omega_p^1 \cap \Omega_l^1 \tag{1.16}$$

$$\Omega^2 = \bigcup_{p=1}^{n_p} \Omega_p^2 \ , \ \Gamma_{pl}^2 = \Omega_p^2 \cap \Omega_l^2 \tag{1.17}$$

where n_p is the number of cores, Γ_{pl}^1 and Γ_{pl}^2 are respectively the interfaces between sub-domains of Ω^1 and Ω^2 . After partitioning Ω^1 and Ω^2, the sub-domains are seen as the sub-meshes formed by each core :

$$\Omega = \Omega^1 \cup \Omega^2 = \bigcup_{p=1}^{n_p} \Omega_p \ , \ \Omega_p = \Omega_p^1 \cup \Omega_p^2 \tag{1.18}$$

$$\Gamma^{pl} = \Gamma_{pl}^1 \cup \Gamma_{pl}^2 = \left(\Omega_p^1 \cup \Omega_p^2\right) \cap \left(\Omega_l^1 \cup \Omega_l^2\right) \tag{1.19}$$

After redefining the computational domain Ω as in equation (1.18), the simulation is launched on each core. The contact surface Γ_c is duplicated on each core eliminating any needed communications. Using this strategy, the contact analysis is carried out between each sub-domain Ω_p^1 and the global surface $\partial\Omega^2$.

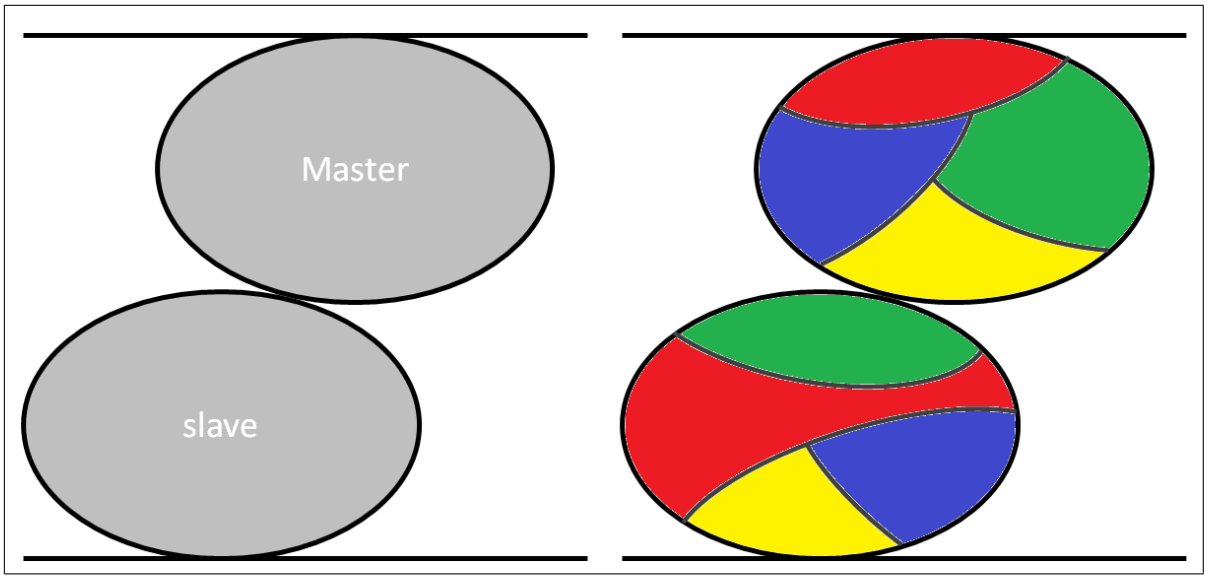


Figure 1.8: *SPMD strategie for parallel computing used in Forge®.*

As for the parallelization of the fictitious element algorithm generation, the mesh of both the domains and the sub-domains are identified via the following numbering:

$$\mathcal{M}(\Omega) = \{1, ..., nbnoe\} \text{ where } \Omega = \Omega^1 \cup \Omega^2 \tag{1.20}$$

$$\mathcal{M}(\Omega^p) = \{1, ..., nbnoe(p)\} \text{ where } \Omega^p = \Omega_p^1 \cup \Omega_p^2 \tag{1.21}$$

in which $nbnoe$ is the total number of nodes and $nbnoe(p)$ the number of nodes assigned to a core p.

Recalling paragraph 1.2.3.2.2, the contact analysis requires to compute the distance between a node $j \in \mathcal{M}(\Omega^p)$ and a surface $A = \{A_1, A_2, A_3\} \in \Omega$ (using a node to segment technique). The set of node (j, A_1, A_2, A_3) form a contact element which does not belong necessary to Ω^p. As a solution, [Barboza, 2004] proposes to add each node $A_{i=\{1,2,3\}}$ to the core containing the node j. In fact, the interface Γ^{pl} should be updated frequently by adding all nodes in contact (belonging to Ω^l) to the core p (e.g. to the set on nodes associated to the core p). This technique is not optimal in term of memory consumption. This strategy is tested in a triaxial compression simulation. Table 1.1 shows the efficiency of this strategy adopted by *Forge®* . We can notice that when using 4 cores, the efficiency decrease to 0.65.

cores	Cpu time (in s)	Efficiency
1	2 153	1.0
2	1 037	1.03
4	822	0.65

Table 1.1: *Efficiency of the contact analysis algorithm in Forge®*

1.2.3.4 Contact resolution

Numerous methods can be found in the literature dedicated to the contact resolution problem. A survey on this subject confirms that the most used methods are the Lagrange multipliers, the penalty method and the augmented Lagrangian method.

Though other methods can be used as well, such as the invariant point algorithm [Chabrand et al., 1998, Duvaut and Lions, 1972], the below descriptions will be only restricted to the previously cited methods.

1.2.3.4.1 ***Lagrange Multiplier Method*** This method consists on introducing new terms representing contact forces [Kikuchi and Oden, 1988] [Wriggers and Rieger, 1999]. These terms are called Lagrange multipliers. Thus the set of equations (1.7) is replaced by (1.22) introducing $\lambda.h$ the Lagrange multiplier.

$$\begin{cases} \int_\Omega s(v) : \varepsilon(w) dV - \int_\Omega p div\, w dV - \int_{\Gamma_c} \lambda h(w) dS = 0 \\ \int_\Omega q div\, v dV = 0 \\ -\int_{\Gamma_c} \lambda^* h(v) dS \leq 0 \end{cases} \tag{1.22}$$

where (1.22.c) represents the non-penetration criterion (to be respected) and $h(v) = (v - v_r).n - \frac{g}{\Delta t}$. This technique presents both drawbacks and advantages. Though it increases the number of unknowns (i.e. the computational time), it respects exactly the contact condition (or what we called non-penetration condition). In addition, it has no need for further parameter manipulation -normally confronted in penalty method- (see section below)

1.2.3.4.2 ***Penalty Technique*** This is one of the firsts techniques treating contact problems. It relies on the fact that numerical solutions are the approximation of the actual solution. So the contact interface is identified with a certain tolerance ε_{prec}. Respecting the contact criterion depends of this precision ε_{prec} as well.

This method consists on introducing penalty coefficient ρ to approximate the contact law. The set of equations (1.7) is now transformed into:

$$\begin{cases} \int_\Omega s(v) : \varepsilon(w) dV - \int_\Omega p div\, w dV - \rho \sum\limits_{x_k \in \Gamma_c} \left[h\left(v\left(x_k\right)\right)\right]^+ S\left(x_k\right)\left(n\left(x_k\right).v\right) = 0 \\ \int_\Omega q div\, v dV = 0 \end{cases} \tag{1.23}$$

where ρ is the penalty coefficient, $S(x_k)$ is the surface associated to the node k and verifies $S(x_k) = \int_{\partial\Gamma} N^k dS$. Thus, σ_n the normal contact stress is approximated by $\sigma_n(x_k) = -\rho h(v_h(x_k))$.

Note that the precision $\varepsilon_{prec} \approx O\left(\frac{1}{\rho}\right)$. So, to respect exactly the non-penetration conditions $\varepsilon_{prec} \to 0$ and $\rho \to +\infty$ must be verified.

Though the penalty method does not increase the unknowns number, it cannot respect exactly the non-penetration conditions unless $\rho \to +\infty$. This generates further complications regarding the badly conditioned matrices (affecting the algorithm convergence).

1.2.3.4.3 ***Augmented Lagrangian Method*** As presented previously for resolution methods, a choice between two alternatives has to be made. Using the "Lagrangian multiplier method" insures the respect of the contact criterion but increases the unknowns. Using the "penalty method" offers exactly the opposite. Which brings us to a new compromise: The "augmented Lagrangian method".

As proposed in [Fortin and Glowinski, 1985], a hybrid method called augmented Lagrangian is introduced. The concept is to solve the problem using the Lagrange multiplier and adding a penalty contribution as well. This method combines the advantages of the latter ones. The Lagrange multiplier contribution helps verifying the non-penetration

condition and the penalty contribution eases the algorithm convergence. For more details, the readers are invited to check [Kuss, 2008] [Chamoret, 2002].

1.3 Eulerian Description

Although the physical fields are the same, the Eulerian and Lagrangian descriptions are quite different. Rather than following each material point, the evolution of fields properties are recorded at every point in space as time varies. In other words, two reference frames are used to evaluate the physical fields. This approach is very common in fluid mechanics. Equation (1.1) represents the transformation function ϕ in the Lagrangian description. As mentioned earlier, an inverse transformation function $\Phi = \phi^{-1}$ can be defined. The initial position of a material point M_t can now be defined as follows:

$$X = \Phi(x, t) \tag{1.24}$$

where x is the spatial variable describing the current point position.

By using (1.24), the deformation is no longer expressed in terms of the Lagrangian variable (X, t). Instead it is expressed in terms of the Eulerien spatial variables (x, t). Let us take the velocity for instance:

$$v(x,t) = \frac{dx}{dt} = \frac{\partial \phi(X,t)}{\partial t} = \frac{\partial \phi(\Phi(x,t))}{\partial t} \tag{1.25}$$

For this reason, when applying a derivative in the Eulerian description two terms show up according to the remark on page 11:

$$\frac{dF}{dt} = \frac{\partial F}{\partial t} + (v.\nabla)F \tag{1.26}$$

$$\begin{cases} \rho\left(\dfrac{\partial v}{\partial t} + v.\nabla v\right) = \nabla.\sigma + \rho b & = & \rho g \\ \nabla.v & = & 0 \\ \sigma & = & behavior\,law \end{cases} \tag{1.27}$$

In fact, by adopting an Eulerian description the mesh is considered fixed in space. The fixed mesh does not represent a physical characteristic. Therefore in a multi-domain problem, tools representing each sub-domain are required; which brings us to the next point: Interface modeling.

1.3.1 Interface modeling

A literature survey shows the existence of several methods for dealing with the interface modeling in Eulerian formulation. Mainly, these methods can be divided into two categories: the *interface tracking* based on a Lagrangian approach and the *interface capturing* based on an Eulerian approach.

For interface tracking, two methods stand out the most: the volume tracking and the front tracking. the first was introduced by [Harlow and Welch, 1965]. It consists on adding particles to represent the volume. The efficiency of this method depends highly on the particles number and their distribution. As a result, it can be expensive in term of memory and computing time. The front tacking method was presented as an improvement of the latter in [Daly and Pracht, 1968]. Instead of distributing particles all over the volume, they are only distributed over the interface. This change plays a key-role in decreasing the memory cost. For these methods, the main drawbacks are the need to reposition particles at each time step and the lack of information regarding the topology changes.

As for interface capturing, two methods exist as well : The volume of Fluid [Hirt and Nichols, 1981] and the Level set method [Osher and Sethian, 1988]. Both methods rely on a scalar parameter to describe the domains interface. The volume of fluid method uses a fraction function C determining if a mesh element is fully or partially filled. If the interface is beyond an element -i.e. element fully filled-, the fraction function is equal to one. If the interface does not reach the element, C takes a zero value. Otherwise, C is equal to the filling of the element volume ratio. Whereas, the level set method uses a signed distance function to describe the interface of a domain. The interface is captured for a time t as the zero of the level set function ϕ (i.e., $\Gamma(t) = \{x|\phi(x,t) = 0\}$. Then, ϕ is chosen to be positive inside the domain and negative outside. The interface motion in both methods is given by convecting the corresponding scalar parameter with the velocity $\vec{v}$. We describe the level set method in detail in the next chapter of this manuscript.

1.3.2 Contact formulation

After reviewing the literature, rare are the works dedicated to contact problems in Eulerian framework. To our knowledge, the only existing work is presented in [Bruchon et al., 2009]. Herein, we present a brief description.

Considering a computational domain Ω formed by a deformable body Ω_d , a rigid tool Ω_s and the air Ω_a. The deformable body and the tool are represented by two distinct level set ϕ_d and ϕ_s. However the air is deduced by complementarity. We should point out that in a Eulerian framework, the contact is not defined explicitly. Therefore, the authors define the contact zone as follows:

$$\Gamma_c = \{x \in \Omega; |\phi_d(x,t)| + |\phi_s(x,t)| = 0\} \tag{1.28}$$

Thus, the no-penetration condition is expressed by:

$$\phi_d(x,t) + \phi_s(x,t) \leqslant 0 \quad \forall (x,t) \in \Gamma_c \times \mathbb{R}^+ \tag{1.29}$$

If v and v_s denote respectively the velocity of the deformable body and the tool, the authors propose linearizing the previous inequality in the following manner:

$$g(\phi_d, \phi_s) = \frac{\phi_d}{\Delta t} + \frac{\phi_s}{\Delta t} + (v - v_s).n_s \leqslant 0 \quad \forall (x,t) \in \Gamma_c \times \mathbb{R}^+ \tag{1.30}$$

where g is the gap function defined using the level set functions and the normal vector n_s. Proceeding by analogy with a Lagrangian formulation, the authors wrote the contact term in the weak formulation using a penalty method.

$$\int_{\Gamma_c} r \left[g(\phi_d, \phi_s)\right]^+ w.n_s d\Gamma \tag{1.31}$$

where r is the penalty parameter.

However, this surface integral is not computable within an Eulerian framework. In fact, the contact surface is not defined explicitly (see equation (1.28)). Thus, surface integral (1.31) has to be converted into a volume integral.

At first, the authors of [Bruchon et al., 2009] introduce the function $\xi : \mathbb{R} \to \mathbb{R}$ such as :

$$\xi(y) = \begin{cases} \frac{1}{2}(1 + cos(\pi y)) & if\ |y| < 1 \\ 0 & if\ |y| \geqslant 1 \end{cases} \tag{1.32}$$

Using the function defined in equation (1.32), the authors approximate the Dirac mass by the right term in equation (1.33) when ε tends towards zero:

$$\delta_{\Gamma_c} \approx \frac{1}{\varepsilon} \xi \left(\frac{|\phi_d| + |\phi_s|}{\varepsilon} \right) |\nabla |\phi_d| + \nabla |\phi_s|| \tag{1.33}$$

Since the contact term is written in the vicinity of the zero isovalue of the level set function, the previous equation is simplified using the property of distance function.

$$\delta_{\Gamma_c} \approx \frac{2}{\varepsilon} \xi \left(\frac{|\phi_d| + |\phi_s|}{\varepsilon} \right) \tag{1.34}$$

Finally, the contact term added to the Eulerian weak formulation is given by:

$$\int_{\Omega} \frac{2r}{\varepsilon} \left[g(\phi_d, \phi_s) \right]^{+} \xi \left(\frac{|\phi_d| + |\phi_s|}{\varepsilon} \right) w.n_s d\Omega \tag{1.35}$$

1.4 Arbitrary Lagrangian Eulerian Description

This section presents the theoretical aspects of Arbitrary Lagrangian Eulerian formulation (ALE) and the different resolution techniques used in solid mechanics. This approach was designed to address problems confronted by adopting a Lagrangian approach while maintaining its advantages. When the ALE approach was introduced, each subdomain was either purely Lagrangian or purely Eulerian. Then the approach was extended to the first arbitrary relative movements between the mesh and material. Nowadays, ALE formulations are widely used treating fluid structure interactions problems [Richter and Wick, 2010, Kucharik et al., 2010]. Nevertheless, some works use such approach in solids mechanics [Wang and Gadala, 1997]. A review of material forming processes using the ALE formulation can be found in [Chenot and Bellet, 1995].

From the mathematical point of view, an area located in the space can be identified using three different frames : a material, a spatial and a reference one. For each frame, a system of coordinates is associated. Position vectors are respectively denoted X, x and χ. The reference frame can be chosen properly to facilitate solving the numerical problem. In a Lagrangian approach, the reference frame coincides with the material one. While in an Eulerian approach, it coincides with the spatial frame. For the ALE approach the reference frame is chosen arbitrary and the material moves with a velocity different from the mesh velocity. The separation of these two velocities requires a new bijective function ψ joining the material frame with the referential one :

$$\chi = \psi(X, t) \tag{1.36}$$

The vector position χ can be expressed using the functions ϕ and ϕ^* crossing from the Lagrangian to the Eulerian frame then to the ALE frame (Figure 1.9) :

$$\chi = \phi^* \circ \phi(X, t) \tag{1.37}$$

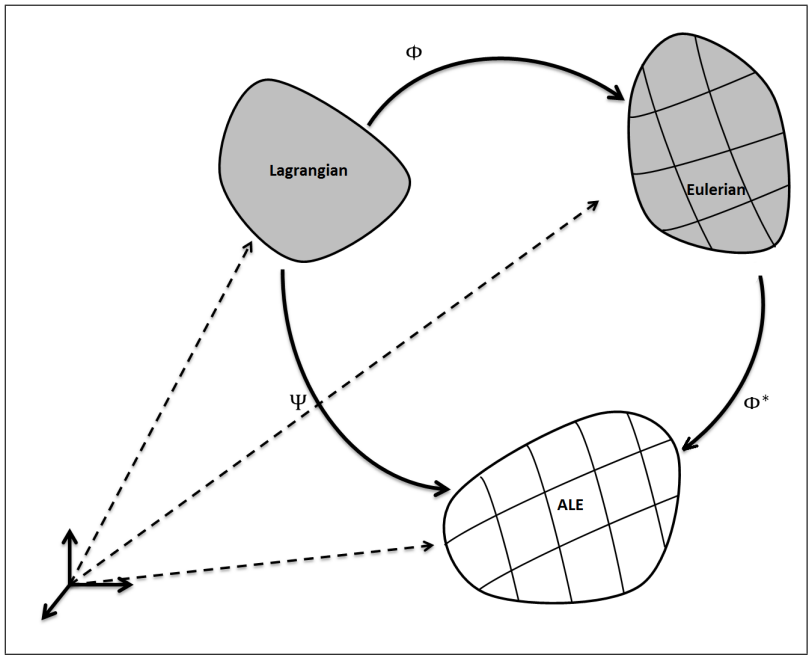


Figure 1.9: *Mesh deformation when using an Arbitrary Lagrangian Eulerian approach.*

To transform equations from the Lagrangian to the ALE description the following relation is used for every characteristic $F(\chi) = F(\chi(x,t)) = F(\chi(x(X,t),t))$:

$$\frac{dF}{dt} = \left[\frac{\partial F}{\partial t}\right]_X = \frac{\partial F}{\partial t} + \frac{\partial F}{\partial \chi}\frac{\partial \chi}{\partial t} = \frac{\partial F}{\partial t} + \frac{\partial F}{\partial x}\frac{\partial x}{\partial \chi}\frac{\partial \chi}{\partial t} = \frac{\partial F}{\partial t} + (c.\nabla)\, F \tag{1.38}$$

in which $c = v - \bar{v}$ is the convective velocity. v is the velocity of the material and $\bar{v}$ is the velocity of the grid. It represents the relative velocity between the material frame and the grid (the spatial frame).

Remark:

- In a Lagrangian approach, since the material is represented by the mesh, the mesh velocity will be equal to the material velocity : $v = \bar{v} \Longrightarrow c = 0$.
- In an Eulerian approach, since the mesh is fixed, the mesh velocity is given by $\bar{v} = 0$. Equation (1.38) will be expressed as follows:

$$\frac{\partial \nu}{\partial t} = \left.\frac{\partial \nu}{\partial t}\right|_X - v\nabla_X.\nu \tag{1.39}$$

Finally, the conservation laws are transformed using equation (1.38). This formulation is also called quasi-Eulerian formulation, since these equations are convected with the relative velocity c.

$$\left.\frac{\partial \rho}{\partial t}\right|_{\chi} + c.\nabla\rho + \rho\nabla.v = 0 \tag{1.40}$$

$$\rho\left(\left.\frac{\partial v}{\partial t}\right|_{\chi} + (c.\nabla)\, v\right) = \nabla.\sigma + \rho b \tag{1.41}$$

$$\rho\left(\left.\frac{\partial e}{\partial t}\right|_{\chi} + c.\nabla e\right) = \sigma : D + \rho r + \nabla.q \tag{1.42}$$

where ρ is the density, v is the velocity, σ is the Cauchy stress tensor, D is the strain rate tensor, b is the body force, e is the internal energy.

1.4.1 Coupled Resolution

Solving the previous equations using a finite element discretization requires the introduction of a variational formulation. The principle of the latter formulation is similar to the one explained in section (sec. 1.2.1).

$$\begin{aligned}\int \rho \left.\frac{\partial v_i}{\partial t}\right|_{\chi} w_i d\Omega + \int \rho c_j \frac{\partial v_i}{\partial x_j} w_i d\Omega + \int \frac{\partial w_i}{\partial x_j}\sigma_{ij} d\Omega \\ = \int \rho b_i w_i d\Omega + \int \sigma_{ij} n_j w_i d\Gamma\end{aligned} \tag{1.43}$$

In a coupled resolution, a convective term appears in equation (1.43). This requires stabilization techniques. The literature proposes several techniques such as SUPG - SCPG are the most used. To solve this type of problem, two methods are mentioned in the literature. The first is to consider that the mesh movement is known a priori [Liu et al., 1986]. This technique makes the mesh velocity independent from the material velocity which remains the only unknown of the problem. However, in most cases the definition of the mesh motion is not obvious. Therefore another method to solve this problem is needed where the mesh motion is unknown. Under the latter assumption, the introduction of new equations describing the mesh motion is required [Schreurs et al., 1986]. For a more elaborate bibliography on this matters the readers are invited to check [Boman, 2010, Philippe, 2009].

1.4.2 Split operator

In solids mechanics, the decoupled approach is more used. The following explained method is extracted from the work of [Benson, 1997, Benson, 1998a]. This method eliminates the convective term of the previous formulation (1.43). It consists on solving the

equations in two consecutive steps. The system of equations (1.44) to (1.46) presented previously is written in the following conservative form:

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho v) = 0 \tag{1.44}$$

$$\frac{\partial \rho v}{\partial t} + \nabla.(\rho v \otimes v) = \nabla.\sigma + \rho b \tag{1.45}$$

$$\frac{\partial \rho e}{\partial t} + \nabla.(\rho e.v) = \sigma : D + \rho b.v \tag{1.46}$$

where ρ is the density, v is the velocity, σ is the Cauchy stress tensor, D is the strain rate tensor, b is the body force and e is the internal energy.

Denoting $\phi = (\rho, \rho v, \rho e)$, $F = (\rho v, \rho v \otimes v, e.v)$ the flux function and $S = (0, \nabla.\sigma + \rho b, \sigma : D)$ the source term, equations (1.44-1.46) are rewritten as :

$$\frac{\partial \phi}{\partial t} + \nabla.F = S \tag{1.47}$$

The splitting Operator divides equation (1.47) into two equations:

$$\frac{\partial \phi}{\partial t} = S \tag{1.48}$$

$$\frac{\partial \phi}{\partial t} + \nabla.F = 0 \tag{1.49}$$

This method consists in solving the equations in two alternating steps. The first solves the equation using purely a Lagrangian approach where all the above definitions remains valid. The second step, called Eulerian, transfers all necessary fields from the deformed Lagrangian mesh to the fixed Eulerian mesh. Figure 1.10 depicts this two step with the mesh evolution.

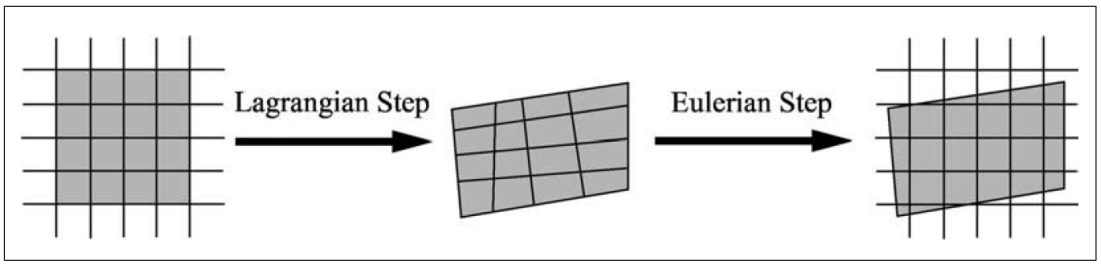


Figure 1.10: *Decoupled resolution using an ALE approach .*

The decoupled method offers several advantages over the coupled method. First, this method is faster and more robust. In addition, the time step is independent of the convective velocity. Finally, the decoupled approach is very easy to implement in a

Lagrangian code. It only requires the addition of a module for determining the speed of the mesh and a transport module. It is at this point that each increment is completed. Several algorithms are available to complete the task. The most intuitive transfer method is to perform an interpolation between the two meshes. This type of techniques is also used after a complete re-meshing for which the number of nodes and the number of elements changes [Philippe, 2009]. An alternative transport technique is represented by solving (1.49).

As mentioned earlier, the ALE approach helps overcoming the problem of mesh distortion encountered in a Lagrangian approach. The mesh motion is given by re-positioning algorithms. According to the nodes nature, these algorithms can be very different. For instance, a node on the mesh surface is generally constrained to remain thereon, while an internal node can be moved with more freedom. These algorithms are divided into two categories: i) mono-material algorithms known as Simple ALE [Benson, 1989] and ii) multi-material [Benson, 1998b]. Multi-material algorithms require sophisticated techniques to determine the boundaries between the involved bodies and then manage the interactions between them [Benson, 1997, Benson and Okazawa, 2004].

Though this section does not cover all aspects in an ALE approach, we should mention that it exhibits the same drawbacks as the Lagrangian approach regarding the efficiency of contact detection algorithms in a parallel environment.

1.5 Discussion and Conclusion

For large deformation problems, the most widespread and most developed methods rely on Lagrangian formulations [Benson, 1997]. Though the mesh suffers a distortion, a re-meshing algorithm is enough to fix this problem and insure the solution accuracy. Some works choose to use an Arbitrary Lagrangian Eulerian to prevent the mesh distortion. With the increasing number of industrial applications including multiple interacting components and complex geometries, a parallel environment is a must. Lagrangian approaches present drawbacks mainly for multi-domain problems in a parallel framework. In particular, the contact remains the topical issue where the spatial search algorithm affects highly the scalability of the whole computation. In *Forge*®, this defect is presented in Table 1.1. Almost half the efficiency is lost when using four cores. In addition, the authors of [Paik et al., 2006] present a study on this particular topic using the bucket sort algorithm. Figure 1.11 confirms that the loss of efficiency is mainly due to the contact calculation.

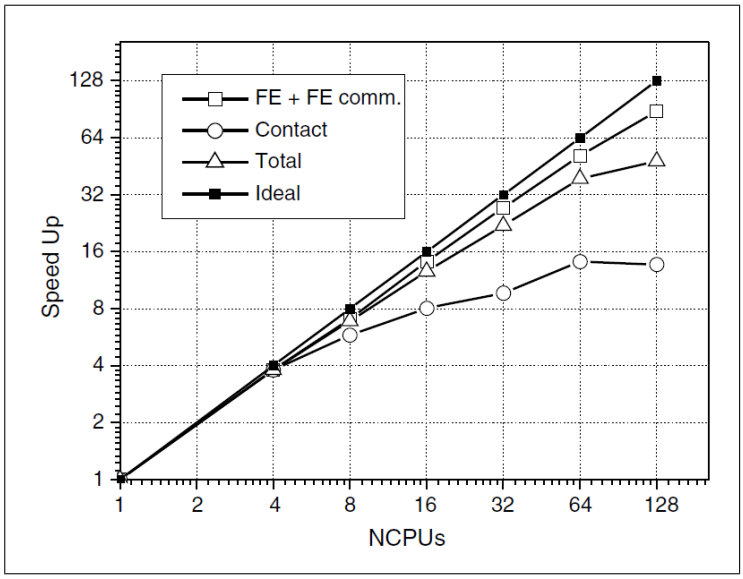


Figure 1.11: *Time gain for different algorithm .*

A worth mentioning work dealing with the parallel contact detection is presented in [Li et al., 2001]. The authors introduce a mesh-free method presenting several improvement such as a simple implementation and a better scalability.

Facing all these needs, the Eulerian approach seems to be a good candidate. This formulation uses a single fixed mesh even in presence of different interacting bodies. It presents the advantage of automatically detection contact. All these advantages does not mean that the formulation will not present limitations. For instance, it opens doors on how to treat friction and contact in such problems. Due to the small number of works treating the large deformation problem in solid mechanics, it shows a lack of development and unclear insight on what will or will not work. This is where our work comes in handy. It studies the feasibility of the Eulerian approach to model solid mechanics applications.

Résumé en Français

Dans ce chapitre, une enquête citant différentes approches disponibles dans la littérature dédiée pour la modélisation des problèmes de grandes déformations multi-domaines a été présentée.

Pour des problèmes de grandes déformations, les approches les plus utilisées et les plus développées se reposent sur des formulations Lagrangiennes où le maillage peut subir des distorsions. Quoique ces distorsions puissent être résolues en utilisant des algorithmes de remaillage, plusieurs travaux se servent des approches Lagrangiennes Arbitrairement Eulériennes comme une alternative.

Avec la demande croissante de modéliser des applications industrielles incluant de multiples géométries (complexes) interagissant au cours du procédé, le travail dans un environnement parallèle est prouvé nécessaire. Les approches lagrangiennes confrontent principalement des difficultés pour les problèmes multi-domaines dans un cadre parallèle. Plus particulièrement, le problème de contact reste le sujet d'actualité où l'algorithme de recherche spatiale affecte fortement l'efficacité du calcul entier.

Face à ces besoins, l'approche Eulérienne semble être un bon candidat. Cette formulation utilise un maillage unique même en présence de différents corps interagissant durant la simulation. La détection de contact est gérée automatiquement et une très bonne scalabilité est attendue. Vis-à-vis à ces avantages, l'utilisation de cette approche mène à de nouvelles questions préalablement non traitées telles que la modélisation du frottement et du contact.

En raison du petit nombre de travaux dédiés à la modélisation des problèmes de grandes déformations de la mécanique solide, les idées sont peu claires sur la direction dont nous devons mener les développements.

Ceci est où notre travail arrive à point nommé. Il étudie la faisabilité de l'approche Eulérienne en proposant les atouts et les développements additionnels nécessaires pour modéliser des problèmes de grandes déformations multi-domaines.

Bibliography

[Aliabadi and Brebbia, 1993] Aliabadi, M. H. and Brebbia, C. A. (1993). Computational methods in contact mechanics. *NASA STI/Recon Technical Report A*, 93:54575.

[Barboza, 2004] Barboza, J. A. P. (2004). *Traitement du contact entre corps déformables et calcul parallèle pour la simulation 3D du forgeage multicorps.* PhD thesis, École Nationale Supérieure Des Mines De Paris.

[Bathe and Brezzi, 2001] Bathe, K. and Brezzi, F. (2001). Stability of finite element mixed interpolations for contact problems.

[Belytschko and Neal, 1991] Belytschko, T. and Neal, M. O. (1991). Contact-impact by the pinball algorithm with penalty and lagrangian methods. *International Journal for Numerical Methods in Engineering*, 31(3):547–572.

[Benson, 1989] Benson, D. J. (1989). An efficient, accurate, simple ale method for nonlinear finite element programs. *Computer Methods in Applied Mechanics and Engineering*, 72(3):305 – 350.

[Benson, 1992] Benson, D. J. (1992). Computational methods in lagrangian and eulerian hydrocodes. *Comput. Methods Appl. Mech. Eng.*, 99(2-3):235–394.

[Benson, 1997] Benson, D. J. (1997). A mixture theory for contact in multi-material eulerian formulations. *Computer Methods in Applied Mechanics and Engineering*, 140(1 - 2):59 – 86.

[Benson, 1998a] Benson, D. J. (1998a). Eulerian finite element methods for the micromechanics of heterogeneous materials: Dynamic prioritization of material interfaces. *Computer Methods in Applied Mechanics and Engineering*, 151(3 - 4):343 – 360. Containing papers presented at the Symposium on Advances in Computational Mechanics.

[Benson, 1998b] Benson, D. J. (1998b). Stable time step estimation for multi-material eulerian hydrocodes. *Computer Methods in Applied Mechanics and Engineering*, 167(1-2):191 – 205.

[Benson and Hallquist, 1990] Benson, D. J. and Hallquist, J. O. (1990). A single surface contact algorithm for the post-buckling analysis of shell structures. *Computer Methods in Applied Mechanics and Engineering*, 78(2):141 – 163.

[Benson and Okazawa, 2004] Benson, D. J. and Okazawa, S. (2004). Contact in a multi-material eulerian finite element formulation. *Computer Methods in Applied Mechanics and Engineering*, 193(39 - 41):4277 – 4298. The Arbitrary Lagrangian-Eulerian Formulation.

[Boman, 2010] Boman, R. (2010). *Développement d'un formalisme Arbitraire Lagrangien et Eulérien tridimensionnel en dynamique implicit.Application aux opérations de mise en forme.* PhD thesis, Université de Liège.

[Bourago and Kukudzhanov, 2005] Bourago, N. G. and Kukudzhanov, V. N. (2005). A review of contact algorithms. Technical report, The Institute for problems in mechanics of RAS.

[Bourel, 2006] Bourel, B. (2006). *Calcul Multi-domaines et approches Multi-Échelles pour la simulation numérique de crashs automobiles.* PhD thesis, LaMCoS - INSA de Lyon.

[Bruchon et al., 2009] Bruchon, J., Digonnet, H., and Coupez, T. (2009). Using a signed distance function for the simulation of metal forming processes: Formulation of the contact condition and mesh adaptation. from a lagrangian approach to an eulerian approach. *International Journal for Numerical Methods in Engineering*, 78(8):980–1008.

[Chabrand et al., 1998] Chabrand, P., Dubois, F., and Raous, M. (1998). Various numerical methods for solving unilateral contact problems with friction. *Mathematical and Computer Modelling*, 28(4-8):97 – 108. Recent Advances in Contact Mechanics.

[Chamoret, 2002] Chamoret, D. (2002). *Modélisation du contact: nouvelles approches numériques.* PhD thesis, École Centrale de Lyon.

[Chenot and Bellet, 1995] Chenot, J. and Bellet, M. (1995). The ale method for the numerical simulation of material forming processes. *Proc. NUMIFORM, eds, S.-F. Shen and P.R. Dawson, Balkema A.A., Rotterdam*, pages 39–48.

[Chenot et al., 2002] Chenot, J.-L., Fourment, L., and Mocellin, K. (2002). Numerical treatment of contact and friction in {FE} simulation of forming processes. *Journal of Materials Processing Technology*, 125-126(0):45 – 52.

[Daly and Pracht, 1968] Daly, B. J. and Pracht, W. E. (1968). Numerical study of density current surges. *Physics of Fluids (1958-1988)*, 11(1):15–30.

[Duvaut and Lions, 1972] Duvaut, G. and Lions, J.-L. (1972). Les inÃ©quations en mÃ©canique et en physique.

[El-Abbasi and Bathe, 2001] El-Abbasi, N. and Bathe, K.-J. (2001). Stability and patch test performance of contact discretizations and a new solution algorithm. *Computers & Structures*, 79(16):1473 – 1486.

[Erleben, 2004] Erleben, K. (2004). Contact graphs in multibody dynamics simulation.

[Farhat and Mandel, 1998] Farhat, C. and Mandel, J. (1998). The two-level {FETI} method for static and dynamic plate problems part i: An optimal iterative solver for biharmonic systems. *Computer Methods in Applied Mechanics and Engineering*, 155(1-2):129–151.

[Flickinger et al., 2013] Flickinger, D., Williams, J., and Trinkle, J. (2013). What's wrong with collision detection in multibody dynamics simulation? In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 959–964.

[Fortin and Glowinski, 1985] Fortin, M. and Glowinski, R. (1985). Augmented lagrangian methods: Applications to the numerical solution of boundary-value problems. *Journal of Applied Mathematics and Mechanics*, 65(12):622–622.

[Fourment et al., 2003] Fourment, L., Barboza, J. A. P., and Popa, S. (2003). Master/slave algorithm for contact between deformable bodies and axial symmetries - application to 3D metal forging. In *COMPUTATIONAL FLUID AND SOLID MECHANICS 2003, VOLS 1 AND 2, PROCEEDINGS*, pages p.269–272 – ISBN 0–08–044046–0, Cambridge, Royaume-Uni.

[Fourment et al., 1999] Fourment, L., Chenot, J., and Mocellin, K. (1999). Numerical formulations and algorithms for solving contact problems in metal forming simulation. *International Journal for Numerical Methods in Engineering*, 46(9):1435–1462.

[Francavilla and Zienkiewicz, 1975] Francavilla, A. and Zienkiewicz, O. C. (1975). A note on numerical computation of elastic contact problems. *International Journal for Numerical Methods in Engineering*, 9(4):913–924.

[Habraken and Cescotto, 1998] Habraken, A. and Cescotto, S. (1998). Contact between deformable solids: The fully coupled approach. *Mathematical and Computer Modelling*, 28(4-8):153 – 169. Recent Advances in Contact Mechanics.

[Hachani and Fourment, 2013] Hachani, M. and Fourment, L. (2013). A smoothing procedure based on quasi-c1 interpolation for 3d contact mechanics with applications to metal forming. *Computers & Structures*, 128(0):1 – 13.

[Hahn and Wriggers, 2003] Hahn, C. and Wriggers, P. (2003). An explicit multi-body contact algorithm. *PAMM*, 3(1):280–281.

[Harlow and Welch, 1965] Harlow, F. H. and Welch, J. E. (1965). Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids*, 8(12):2182–2189.

[Heinstein et al., 2000] Heinstein, M. W., Mello, F. J., Attaway, S. W., and Laursen, T. A. (2000). Contact-impact modeling in explicit transient dynamics. *Computer Methods in Applied Mechanics and Engineering*, 187(3-4):621 – 640.

[Hirt and Nichols, 1981] Hirt, C. and Nichols, B. (1981). Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201 – 225.

[Hughes, 2012] Hughes, T. J. (2012). *The finite element method: linear static and dynamic finite element analysis*. Courier Dover Publications.

[Iwai et al., 1999] Iwai, T., Hong, C.-W., and Greil, P. (1999). Fast particle pair detection algorithms for particle simulations. *International Journal of Modern Physics C*, 10(05):823–837.

[Kikuchi and Oden, 1988] Kikuchi, N. and Oden, J. T. (1988). *Contact problems in elasticity: a study of variational inequalities and finite element methods*. Philadelphia : SIAM.

[Kucharik et al., 2010] Kucharik, M., Garimella, R. V., Schofield, S. P., and Shashkov, M. J. (2010). A comparative study of interface reconstruction methods for multi-material ale simulations. *Journal of Computational Physics*, 229(7):2432 – 2452.

[Kuss, 2008] Kuss, F. (2008). *Méthodes duales pour les problèmes de contact avec frottement*. PhD thesis, Université de Provence.

[Laursen, 2002] Laursen, T. (2002). *Computational Contact and Impact Mechanics: Fundamentals of Modeling Interfacial Phenomena in Nonlinear Finite Element Analysis*. Springer-Verlag, Heidelberg.

[Li et al., 2001] Li, S., Qian, D., Liu, W. K., and Belytschko, T. (2001). A meshfree contact-detection algorithm. *Computer Methods in Applied Mechanics and Engineering*, 190(24-25):3271 – 3292. Advances in Computational Methods for Fluid-Structure Interaction.

[Liu et al., 1986] Liu, W. K., Belytschko, T., and Chang, H. (1986). An arbitrary lagrangian-eulerian finite element method for path-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, 58(2):227 – 245.

[Martins and Oden, 1987] Martins, J. and Oden, J. (1987). Existence and uniqueness results for dynamic contact problems with nonlinear normal and friction interface laws. *Nonlinear Analysis: Theory, Methods and Applications*, 11(3):407 – 428.

[Mocellin, 1999] Mocellin, K. (1999). *Contribution à la simulation numérique tridimensionnelle du forgeage à chaud : Étude du contact et calcul multigrille*. PhD thesis, École nationale supérieure des Mines de Paris.

[Oliveira et al., 2008] Oliveira, M., Alves, J., and Menezes, L. (2008). Algorithms and strategies for treatment of large deformation frictional contact in the numerical simulation of deep drawing process. *Archives of Computational Methods in Engineering*, 15(2):113–162.

[Osher and Sethian, 1988] Osher, S. and Sethian, J. (1988). Fronts propagating with curvature dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49.

[Paik et al., 2006] Paik, S. H., Moon, J. J., Kim, S. J., and Lee, M. (2006). Parallel performance of large scale impact simulations on linux cluster super computer. *Computers & Structures*, 84(10-11):732 – 741.

[Papadopoulos and Taylor, 1993] Papadopoulos, P. and Taylor, R. (1993). A simple algorithm for three-dimensional finite element analysis of contact problems. *Computers & Structures*, 46(6):1107 – 1118.

[Philippe, 2009] Philippe, S. (2009). *Dévéloppement dd'un formulation Arbitraire LaLagrangien Eulérienne pour la simulation tridimensionnelle du laminage de produits plats*. PhD thesis, École Nationale Supérieure des Mines de Paris.

[Richter and Wick, 2010] Richter, T. and Wick, T. (2010). Finite elements for fluid-structure interaction in ale and fully eulerian coordinates. *Computer Methods in Applied Mechanics and Engineering*, 199(41-44):2633 – 2642.

[Roux and Garaud, 2009] Roux, F.-X. and Garaud, J.-D. (2009). Domain decomposition methodology with robin interface matching conditions for solving strongly coupled fluid-structure problems. *International Journal for Multiscale Computational Engineering*, 7(1).

[Roux et al., 2005] Roux, F.-X., Magoulés, F., Series, L., and Boubendir, Y. (2005). Approximation of optimal interface boundary conditions for two-lagrange multiplier feti method. In Barth, T., Griebel, M., Keyes, D., Nieminen, R., Roose, D., Schlick, T., Kornhuber, R., Hoppe, R., PÃ©riaux, J., Pironneau, O., Widlund, O., and Xu, J., editors, *Domain Decomposition Methods in Science and Engineering*, volume 40 of *Lecture Notes in Computational Science and Engineering*, pages 283–290. Springer Berlin Heidelberg.

[Sassi et al., 2008] Sassi, T., Ipopa, M., and Roux, F. (2008). Generalization of lions' nonoverlapping domain decomposition method for contact problems. In Langer, U., Discacciati, M., Keyes, D., Widlund, O., and Zulehner, W., editors, *Domain Decomposition Methods in Science and Engineering XVII*, volume 60 of *Lecture Notes in Computational Science and Engineering*, pages 623–630. Springer Berlin Heidelberg.

[Schreurs et al., 1986] Schreurs, P., Veldpaus, F., and Brekelmans, W. (1986). Simulation of forming processes, using the arbitrary eulerian-lagrangian formulation. *Computer Methods in Applied Mechanics and Engineering*, 58(1):19 – 36.

[Sofonea and Matei, 2012] Sofonea, M. and Matei, A. (2012). *Mathematical Models in Contact Mechanics*. Cambridge University Press. Cambridge Books Online.

[Wang and Gadala, 1997] Wang, J. and Gadala, M. (1997). Formulation and survey of {ALE} method in nonlinear solid mechanics. *Finite Elements in Analysis and Design*, 24(4):253 – 269.

[Wriggers, 2002] Wriggers, P. (2002). *Computational Contact Mechanics*. JOHN WILEY & SONS, LTD.

[Wriggers and Rieger, 1999] Wriggers, P. and Rieger, A. (München, Germany, August 31-Septembre 3 1999.). Adaptive methods for contact problems, recent developments. In *In European Conference and Computational Mechanics*.

[Yastrebov, 2011] Yastrebov, V. A. (2011). *Computational contact mechanics: geometry, detection and numerical techniques*. PhD thesis, École Nationale Supérieure des Mines de Paris.

CHAPTER 2

Monolithic approach tools

Contents

2.1 Introduction 43

2.2 Interface Capturing 44

2.2.1 Distance function 44

2.2.2 Level Set Method 45

2.2.2.1 Local Distance function 46

2.2.2.2 Convected level set method 47

2.2.2.3 Weak form 48

2.3 Mesh Generation 49

2.4 Anisotropic mesh adaptation 50

2.4.1 Edge based error estimator 51

2.4.2 Metric construction 52

2.4.3 Extension to multi-component field 55

2.5 Mechanical problem 56

2.5.1 Governing equations 56

2.5.2 Weak formulation 58

2.5.3 Monolithic system 59

2.5.4 Finite Element Formulation 59

2.5.4.1 Spatial discretization 59

2.5.4.2 Variational Multi-scale Stabilization 60
2.5.4.3 Stabilization parameter 63
2.6 Mixture laws 64
2.6.1 Linear mixture law 65
2.6.2 Novel mixture law: Quadratic law 66
2.7 New additions and Numerical Applications 67
2.7.1 Test cases and proposed ameliorations 67
2.7.1.1 Optimal parameter choices 67
2.7.1.2 Weight of mixture laws 69
2.7.1.3 Transport stabilization 74
2.7.2 Air trapping 77
2.7.3 Porous boundary conditions 79
2.8 Conclusion 81
Résumé en Français 83
Bibliography 84

2.1 Introduction

This chapter is devoted to the numerical approach used in this work to model large deformations problems. This approach is based on an Eulerian description of the monolithic system governing a multi-domain deformation problem. The used approach proved to be a powerful tool modeling CFD and FSI [Feghali et al., 2010] problems. Despite its potential, this approach was rarely if never applied to large deformation problems such as metal forming processes (treated later on in this study).

Unlike Lagrangian formulations, in a Eulerian one the mesh itself does not represent the material neither its evolution. This fact leads us to introduce first interface capturing techniques such as distance function. The evolution of a certain domain is followed by solving the convective problem corresponding to the level set function.

Next, the interface capture method is coupled with an heterogeneous anisotropic mesh adaptation technique presented in [Coupez, 2011]. The main goal is to obtain a high precision solution while maintaining the lowest possible number of nodes. It is insured since the mesh is only adapted where the important physical phenomenons appear, while it remains coarse elsewhere. Thus, a large scale simulations can be handled at an affordable cost (i.e. computing time).

To simplify the different steps leading to the monolithic system describing a multi-domain large deformation problem, we depict them using a simple flat simulation problem (containing two tools and one deformable sub-domains to generalize later on). One single mesh is used and the different sub-domains are represented as heterogeneities in the global domain. Since our aim is to solve one set of equations and since the different sub-domains can easily have different physical characteristics, mixture laws come in handy at this stage to introduce global characteristics defined all over the computational domain using the local ones corresponding to each sub-domain. New additions to this monolithic approach are proposed to help adapt it to large deformation problems. For instance, a novel quadratic law is introduced that can be helpful to represent an additional phase (detecting the presence of a lubricant for example). Air trapping is studied as well. Two options are offered to treat this topic depending on the problem need. In addition, some ameliorations are brought to the existent Eulerian approach to insure better results. For every improvement, academic applications are presented to understand both capacities and limitations of this approach.

The chapter is ended with a quick discussion covering the confronted difficulties, what has to be done to prevent them and what remains unsolved to have a more-or-less complete approach treating every aspect of large deformation problems.

2.2 Interface Capturing

Aiming to present a "fully" Eulerian approach, the interface capturing should rely on an Eulerian description. Volume of fluids method (VOF) was the most often used. It is based on a mass conservation and consists on convecting the characteristic function over the whole mesh. However, it faces several difficulties such as reconstructing the interface, at each step, specially in $3\,d$ geometries. Later on, Osher introduced the level set method in [Osher and Sethian, 1988]. In addition to its fluency taking into account every slight topology changes; this method has proven not only its effectiveness to describe precisely the interface but also a quick and smooth transition from $2d$ to $3d$ without any further development and complications. This method and other variant ones will be presented next.

2.2.1 Distance function

Consider $\Omega \subset \mathbb{R}^n$ the fixed Eulerian computational domain, in which all different bodies are immersed. A standard level set method consists on defining a signed distance function ϕ verifying (2.1).

$$|\phi(\vec{x},t)| = d(\vec{x}) = \underset{\vec{x_0}\in\Gamma(\vec{x})}{min}(|\vec{x}-\vec{x}_0|) \quad \forall\, \vec{x} = (x_1,\ldots,x_n) \in \Omega \tag{2.1}$$

where $\Gamma(\vec{x}) = \{\vec{x} \in \Omega; /\phi(\vec{x},t) = 0\}$ is the interface of the studied domain. In this work, ϕ is chosen to be positive inside the domain and negative otherwise.

If i different bodies $(\Omega_i)_i$ are immersed in Ω, for each body $\Omega_i(t)$ a signed distance function $\phi_i(t)$ is associated and Ω is chosen to cover all bodies:

$$\cup\, \Omega_i(t) \subseteq \Omega \tag{2.2}$$

Both the immersed body and its distance function can depend of the time variable t, since they are supposed to evolve during simulations later on. Note that we do not assign a level set to the air domain. Instead it is deduced by complementarity as follows:

$$\Omega_a = \Omega - \cup\Omega_i \tag{2.3}$$

To illustrate this concept, we take the example of one sphere, denote Ω_s, immersed in the computing domain Ω. Figure 2.1 depicts the immersed sphere along with a horizontal slice. The isovalues of this slice $-0.4 < iso < 0.285$ are projected on the bottom of Ω. The latter is a mean to clarify how the distance function associated to the sphere is defined. The projected isovalues confirm that they are negative outside the sphere and positive inside. Thus, the isovalue zero corresponds to the interface of this sphere.

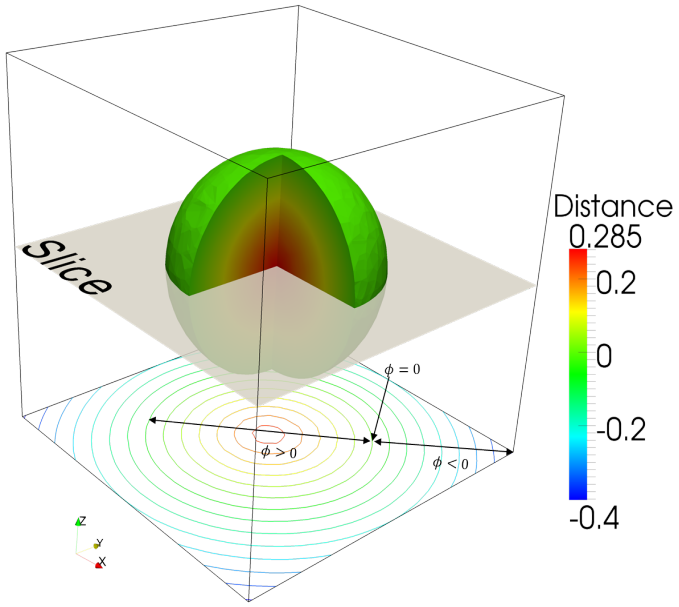


Figure 2.1: *A sphere immersed in a cubic computational domain* Ω*. The signed distance function isovalues of the central slice are projected onto the bottom of the domain.*

Now that the distance function is defined, the integral on the sub-domain itself can be defined as well. Let f be an integrable function on Ω. The integral of f on Ω_S can be expressed as the integral of the function $f.H$ on Ω:

$$\int_{\Omega_S} f d\Omega_S = \int_{\Omega} f H(\phi_s) d\Omega \tag{2.4}$$

where H is the Heaviside function :

$$H(\phi) = \begin{cases} 1 & if\ \phi > 0 \\ 0 & if\ \phi < 0 \end{cases} \tag{2.5}$$

2.2.2 Level Set Method

Assuming that the signed distance function is advected by a given velocity field $\vec{v}$, the distance function convection is equivalent to solving the following system on Ω:

$$\begin{cases} \phi_t + \vec{v}.\nabla\phi &= 0 \\ \phi(t=0,\vec{x}) &= \phi_0(\vec{x}) \end{cases} \tag{2.6}$$

In fact, the equation (2.6) describes the evolution of the level set function in time. To conserve its stability during the convection, the following feature should be always verified:

$$|\nabla\phi| = 1 \tag{2.7}$$

Despite all extensive research and improvements made in stabilization technique; the equation (2.7) was not insured by simply solving (2.6). Since maintaining this gradient property is a must to guarantee the stability of the numerical system, it is then necessary to restore it by solving a Hamilton-Jacobi equation (called also the reinitialization equation):

$$\begin{cases} \phi_\tau + s(\phi)(|\nabla\phi| - 1) &= 0 \\ \phi(\tau = 0, \vec{x}) &= \phi(t, \vec{x}) \end{cases} \tag{2.8}$$

where τ is a virtual time and $s(\phi)$ is the sign function defined as follows:

$$s(\phi) = \begin{cases} 1 & if\ \phi > 0 \\ 0 & if\ \phi = 0 \\ -1 & if\ \phi < 0 \end{cases} \tag{2.9}$$

The system (2.8) does not affect the position of the zero-value of the level set function; it simply repositions the other isovalues to verify the equation (2.7). The previous reinitialization algorithm was introduced in [Sussman et al., 1994]. The frequency of reinitialization after solving the system (2.6) was found a delicate matter, it can affect considerably the computing time. Authors like [Osher and Sethian, 1988, Sussman et al., 1994] urge to reinitialize ϕ after each time step, others find it sufficient to reinitialize it every other step.

2.2.2.1 Local Distance function

The level set advection though required, it is not necessary on the whole domain. In computations, the important information are located in the vicinity of the zero isovalue i.e. on the body interface. Therefore, it could be handy to use a function that crops all isovalues in the interface surroundings. Several functions exist in the literature. For example, [Ville et al., 2011] used a sinusoidal function. In this work, a hyperbolic tangent function $\hat{\phi}$ is chosen to narrow the distance function ϕ:

$$\hat{\phi} = \varepsilon.tanh\frac{\phi}{\varepsilon} \tag{2.10}$$

where ε represents the thickness of the interface and depends on the mesh size. Figure 2.2 shows that both functions ϕ and $\hat{\phi}$ match completely in the vicinity of zero. Thus, the use of the hyperbolic filter $\hat{\phi}$ instead of the distance function ϕ in the interface proximity does not affect at all the results precision. On the contrary, it can be beneficial decreasing the computing domain and time.

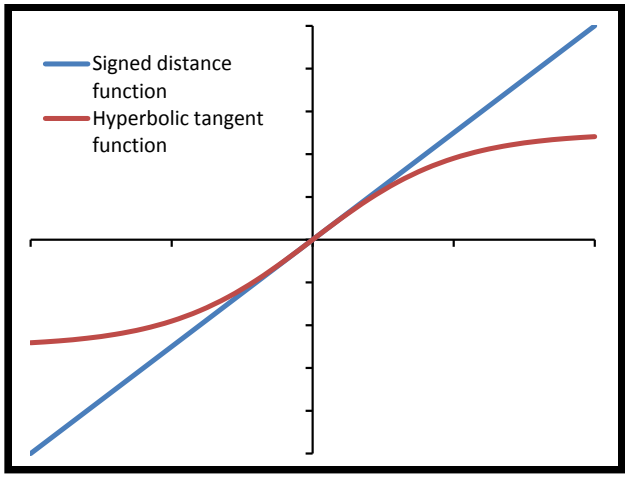


Figure 2.2: *Comparison between a signed distance function (blue) and a hyperbolic tangent function (red)*

2.2.2.2 Convected level set method

To summarize, a classical level set approach depends on solving both (2.6) and (2.8) systems. Recall that the system (2.8) is not linear. [Peng et al., 1999] introduced in his work a linearization based on evaluating the gradient at the previous virtual time:

$$|\nabla\phi| = \frac{|\nabla\phi|^2}{|\nabla\phi|} \approx \frac{\nabla\phi.\nabla\phi^-}{|\nabla\phi^-|} \tag{2.11}$$

Substituting the gradient norm in the first equation of the system (2.8) by the formula (2.11) , the reinitialized convection equation is transformed into:

$$\frac{\partial\phi}{\partial\tau} + s(\phi^-)\frac{\nabla\phi^-}{|\nabla\phi^-|}\nabla\phi = s(\phi^-) \tag{2.12}$$

Notice that equation (2.12) is the convection equation of ϕ with a velocity field $\vec{u} = s(\phi^-)\dfrac{\nabla\phi^-}{|\nabla\phi^-|}$. The convective reinitialization was introduced by Coupez in [Coupez, 2006]. In addition, he showed that equation (2.12) along with equation (2.6) leads to a convected level set method. In fact, the virtual time step can be decomposed as follows:

$$\frac{\partial\phi}{\partial t} = \frac{\partial\phi}{\partial\tau} \times \frac{\partial\tau}{\partial t} \tag{2.13}$$

$\dfrac{\partial\tau}{\partial t}$ will be approximated by $\dfrac{h}{\Delta t}$ and noted λ, where h is the mesh size and Δt is the time step [Ville et al., 2011]. Using equation (2.13), equation (2.12) is rewritten to obtain:

$$\frac{\partial\phi}{\partial t} + \lambda s\left(\phi\right)\left(|\nabla\phi| - 1\right) = 0 \tag{2.14}$$

Adding equations (2.14) to (2.6), the resulting equation is called the convective level set equation:

$$\frac{\partial \phi}{\partial t} + (v + \lambda u)|\nabla \phi| = \lambda s\,(\phi) \tag{2.15}$$

Finally, the signed distance function is substituted by the local distance one defined in (2.10), but for the sake of clarity we maintain the same notation ϕ:

$$\begin{aligned} \frac{\partial \phi}{\partial t} + (v + \lambda u).\nabla \phi &= \lambda s(\phi) g(\phi) \\ \phi(t = 0, \vec{x}) &= \phi_0(\vec{x}) \end{aligned} \tag{2.16}$$

where $g(\phi) = 1 - \left(\frac{\phi}{\varepsilon}\right)^2$.

2.2.2.3 Weak form

Until now, we presented a convective level set method. The remaining part is discretizing the corresponding system. In this section, the finite element discretization is presented briefly.

In order to approximate the level set function as a linear finite element, a space approximation must be determined :

$$\mathcal{W}_h = \left\{ w_h \in \mathcal{C}^0(\Omega_h), w_{h|K} \in P^1(K), \forall K \in \mathcal{T}_h(\Omega) \right\} \tag{2.17}$$

Multiplying the first equation of the system (2.16) by $w_h \in \mathcal{W}_h$, the temporal-spatial discretization of the weak formulation is given by :

$$\begin{aligned} \int_\Omega \frac{\phi_h}{\Delta t} w_h + \int_\Omega \frac{\phi_h^-}{\Delta t} w_h = \theta \left[\int_\Omega \lambda s g(\phi_h) w_h - \int_\Omega (v + \lambda u).\nabla \phi_h w_h \right] \\ + (1 - \theta) \left[\int_\Omega \lambda s g(\phi_h^-) w_h - \int_\Omega (v + \lambda u).\nabla \phi_h^- w_h \right] \end{aligned} \tag{2.18}$$

where $\phi^- = \phi(x, t-1)$ and θ is a parameter used to choose the type of the temporal scheme. In this work, we use a semi-implicit scheme (i.e $\theta = \frac{1}{2}$).

As we are dealing with a convection problem, a stabilization method is needed. Streamline Upwind Petrov-Galerkin method (SUPG) is one of the most popular used methods. Therefore, the space $\mathcal{W}_h$ is replaced by $\overline{\mathcal{W}}_h$, where the shape function is modified as mentioned in (2.19)

$$\overline{\mathcal{W}}_h = \left\{ \bar{w}_h, \bar{w}_{h|K} = w_h + \tau_{SUPG} v.\nabla w_h, \forall K \in \mathcal{T}_h(\Omega) \right\} \tag{2.19}$$

In other words, replacing $\mathcal{W}_h$ by $\overline{\mathcal{W}}_h$ means giving the shape functions an additional artificial weight in the flow direction (see Figure 2.3).

For the determination of the parameter stabilization τ_{SUPG} the reader can refer to [Brooks and Hughes, 1982].

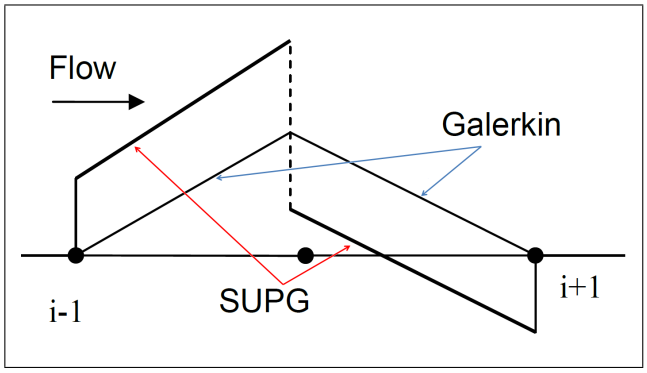


Figure 2.3: *Different weight of the shape functions using a Galerkin approximation and a Streamline Upwind Petrov-Galerkin .*

To maintain the highest precision possible in this eulerian approach, the anisotropic mesh adaptation is required. Next, we will present the technique implemented in CIMLib to generate and adapt the mesh.

2.3 Mesh Generation

A large effort has been devoted to develop mesh generation algorithms. A survey of the literature offers several algorithms capable of generating an unstructured triangle/tetrahedral mesh. The three major algorithms are: *Quadtree/Octree* described in [Tchon et al., 2003, Shephard, 1988, Shephard and Georges, 1991], *Frontal Method* presented in [Bonet and Peraire, 1991, Lo, 1985] and finally the *Delaunay Triangulation* detailed in [Frey and George, 1999]. The quality of the mesh is crucial for all computations. To ensure an optimal mesh quality, these three algorithms require an optimization step after the generation step.

However, this work lies on a topological mesh. It was first introduced in [Coupez et al., 1991] and proved in [Coupez, 2000]. Unlike the previous algorithms, the topological mesh optimizes the mesh automatically while its generation. It uses the below definition to construct the mesh and the minimal volume theorem to optimize its quality.

Definition 2.3.0.1. *Let $(\mathcal{N}, \mathcal{T})$ denotes respectively the nodes and the set of simplex whose vertices belong to $\mathcal{N}$. Considering $\mathcal{F}$ the set of faces of the simplex, $\mathcal{T}$ is a mesh topology of Ω if and only if:*

(i) *Each face of $\mathcal{F}$ shares at most two elements.*

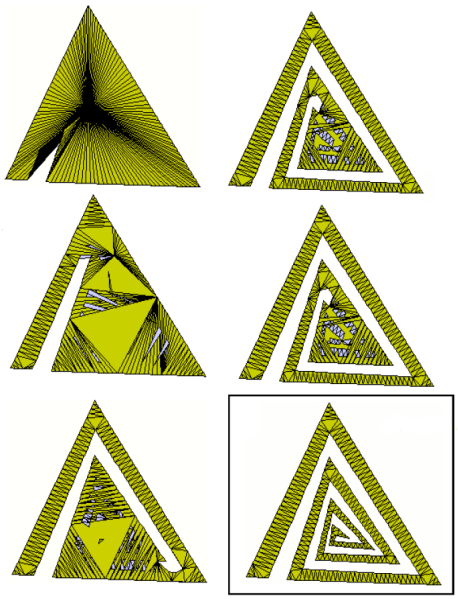


Figure 2.4: *Mesh generation by local optimization using "MTC"*

(ii) $(\mathcal{N}, \partial\mathcal{T})$ *is a mesh of* $\partial\Omega$.

Theorem 2.3.0.1. *Let* $\mathcal{T}$ *be a mesh topology on a finite node set* $\mathcal{N}$ *in* Ω. *Then* $(\mathcal{N}, \mathcal{T})$ *is a mesh of* Ω *if and only if the simplices of* $\mathcal{T}$ *are non-degenerated and*

$$\sum_{T \in \mathcal{T}} |T| = |\Omega| \tag{2.20}$$

The demonstration can be found in [Coupez, 2000, Gruau and Coupez, 2005] .

2.4 Anisotropic mesh adaptation

Working in an Eulerian framework, an anisotropic mesh adaptation is an efficient means to ensure an accurate prediction of the numerical solution. In fact, an uniform global mesh increases the number of degrees of freedom and may not lead to an improvement of the obtained solution. Starting from a coarse initial mesh, this technique consists on refining it locally in the regions where physical informations requires more nodes. Coupled with the Level set method, the anisotropic mesh adaptation is capable of handling complex geometries. The purpose of this section is to describe briefly the edge based error estimator and the metric construction algorithm implemented in CIMLib. Further details of the metric calculation and the error estimator used to control the generation of the anisotropic mesh are available in [Coupez, 2011, Coupez and Hachem, 2013].

2.4.1 Edge based error estimator

In large deformations problems, the arise of a new arbitrary surface is common. As mentioned earlier, an isotropic mesh leads to a loss in the accuracy of the desired solution, due to the arbitrary evolution of the interfaces of the different sub-domains. Therefore, an anisotropic mesh adaptation is required. As a matter of fact, discretization errors occur during simulations. The goal of the error estimator is to convert the numerical error into an additional information used to refine locally the mesh where needed. In [Coupez, 2011], the author proposed an edge based error estimator.

Let us begin with recalling some useful notations and definitions.
Consider that $v \in \mathcal{C}^2(\Omega) = \mathcal{V}$ and $\mathcal{V}_h$ the $P1$ finite element approximation such that : $\mathcal{V}_h = \{v_h \in \mathcal{C}^0(\Omega),\, v_h|_k \in P^1(K), K \in \mathcal{K}\}$, where $\mathcal{K}$ represents the set of element of the mesh. Furthermore let us denote by $\mathbf{X} = \{\mathbf{X^i} \in \mathbb{R}^\mathbf{d},\, \mathbf{i} = \mathbf{1}, \cdots, \mathbf{N}\}$ the set of nodes, and by $\mathbf{U}^i$ the projection of the solution $\mathbf{U}$ at the node $\mathbf{X}^i$ given by the Lagrange interpolation. Finally let $\mathbf{X}^{ij} = \mathbf{X}^j - \mathbf{X}^i$ and $\mathbf{U}^{ij} = \mathbf{U}^j - \mathbf{U}^i$ be respectively the length between the nodes i and j and the variation between the solution evaluated on these nodes.

Since a $P1$ approximation is used, the solution is continuous and can be written as:

$$\mathbf{U}^j = \mathbf{U}^i + \nabla u_h \mathbf{X}^{ij} \tag{2.21}$$

so,

$$\nabla u_h \mathbf{X}^{ij} = \mathbf{U}^{ij} \tag{2.22}$$

Notice that the gradient of u_h is a piecewise constant vector i.e. constant vector on each element. Though, the gradient field is discontinuous from element to element, it is continuous along the element edges since it depends only on the variation of the field value at the extremities (see equation (2.22)). Most error estimators are based on the Hessian approximation, nevertheless the author of [Coupez, 2011] proposes an error based on the gradient.

Using identities (2.21 - 2.22) and the mean value theorem, the following inequality is determined :

$$\left\| \nabla u_h . \mathbf{X}^{ij} - \nabla u(X^i) . \mathbf{X}^{ij} \right\| \leq \max_{Y \in [X^i, X^j]} \left| \mathbb{H}(u)(Y) \mathbf{X}^{ij} . \mathbf{X}^{ij} \right| \tag{2.23}$$

where $\mathbb{H}(u)$ is the Hessian solution. Equation (2.23) is the main reason behind the choice of the following gradient based error estimator:

$$e_{ij} = \mid G^{ij} X^{ij} \mid \tag{2.24}$$

where $G^{ij} = G^i - G^j$ is the difference between the gradient defined respectively on the nodes i and j.
Since the gradient is only determined on the elements, an additional step is needed to recover its values on the nodes. It is accomplished by solving the following equation:

$$G^i = \underset{G}{argmin} \left(\sum_{j \in \Gamma(i)} \left| (G - \nabla u_h) \, . X^{ij} \right|^2 \right) \tag{2.25}$$

where $\Gamma(i) = \{ j \in \mathcal{N},\ \exists K \in \mathcal{K},\ X^{ij} \in K \}$.

2.4.2 Metric construction

To pilot such a technique, a metric is needed. A metric is a symmetric defined positive tensor. It can be transformed into a diagonal one using:

$$\mathbb{M} = \mathbb{R} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_d \end{bmatrix} \mathbb{R}^T \tag{2.26}$$

where $\mathbb{R}$ is a rotation tensor containing the eigenvectors of $\mathbb{M}$ and $(\lambda_i)_{i=1,\cdots,d}$ represent the eigenvalues of the metric. The eigenvalues $(\lambda_i)_{i=1,\cdots,d}$ are strongly related to the shape and the size of the obtained mesh. It will be inversely proportional to the square of the mesh size. The orientation of the mesh elements will be determined by both $\mathbb{R}$ and $\mathbb{R}^T$. During computations, a generated mesh formed by equilateral triangles is the most suitable. That is why the metric notion will introduce a distance between the set of the nodes with the aim of maintaining the norm close to one :

$$\left\| X^{ij} \right\|_M = \left(\mathbb{M} X^{ij}, X^{ij} \right) = 1 \tag{2.27}$$

in which X^{ij} is the vector formed by the nodes i and j.

Following the work of [Coupez, 2011] , the proposed metric is constructed on each node of the mesh verifying (2.27). In fact, the author demonstrates that a unit metric can be built via an affine transformation $\mathbb{A}^{ij}$. The latter transformation will only adjust each edge length to be equal to 1. Therefore, the metric can be written as $\mathbb{M} = {}^t\mathbb{A}^{ij}\mathbb{A}^{ij}$ (defined on the node i).

When summing (2.27) on $\Gamma(i) = \{ j \in \mathcal{N},\ \exists K \in \mathcal{K},\ X^{ij} \in K \}$, we obtain the following identity:

$$\mathbb{M} : \left(\sum_{j \in \Gamma(i)} X^{ij} \otimes X^{ij} \right) = |\Gamma(i)| \tag{2.28}$$

The tensor in (2.28) on the node i is called the length distribution tensor defined (see Figure 2.5) :

$$\mathbb{X}^i = \frac{1}{|\ \Gamma(i)\ |} \left(\sum_{j \in \Gamma(i)} X^{ij} \otimes X^{ij} \right) \tag{2.29}$$

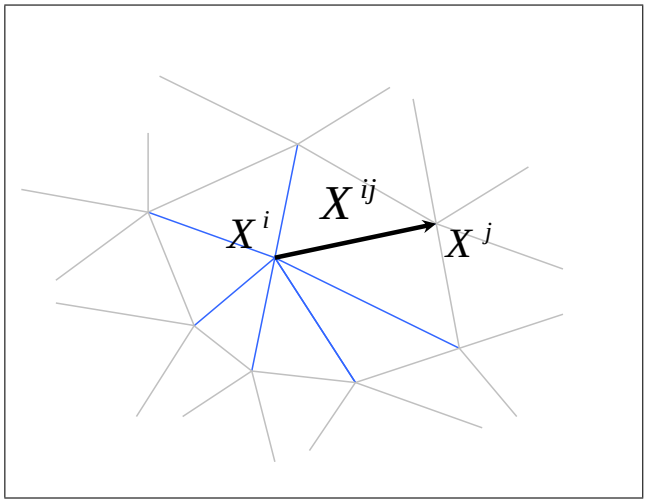


Figure 2.5: *The length distribution tensor for the node i is determined from the collected edges*

Since $\mathbb{X}^i$ is symmetric defined positive, when at least d non aligned edge vectors exist the metric is defined as :

$$\mathbb{M} = \frac{1}{d} \left(\mathbb{X}^i \right)^{-1} \tag{2.30}$$

In order to introduce the error variations calculated in (2.24) to the metric defined in (2.30), a new edge length is defined based on a stretch factor s_{ij}. The introduction of the latter factor leads to a new definition of the error given by $\widetilde{e}_{ij} = s_{ij}^2 e_{ij}$. More precisely, the stretch factor leads to generate a mesh under the constraint of a fixed number of edges. In fact, by scaling the edges by s_{ij}, the number of edges is multiplied by s_{ij}^{-p}. Thus for a given number of edges A, the proposed stretching factor is computed as follows:

$$s_{ij} = \left(\frac{\lambda}{e_{ij}} \right)^{\frac{1}{p}} \tag{2.31}$$

in which

$$\lambda = \left(\frac{\left(\sum_i \sum_{j \in \Gamma(i)} e_{ij}^{\frac{p}{p+2}} \right)}{A} \right)^{\frac{p+2}{p}}$$

Finally the optimized metric is given by:

$$\mathbb{M}^i = \frac{1}{d} \left(\sum_{j \in \Gamma(i)} s_{ij}^2 X^{ij} \otimes X^{ij} \right)^{-1} \tag{2.32}$$

A full review of the metric construction, lemma, propositions and proofs can be found in [Coupez, 2011]. Figure 2.6 shows the mesh obtained when using the technique described above for two different geometries. In Figure 2.6 (a-b) we choose a cube, it will be used latter on to study the case of a flat bunch simulation. To prove that this technique can handle more complex geometries, Figure 2.6 (c) shows the adapted mesh of the upper tool of a connecting rod. We point out that in these cases the mesh is adapted on the level set ϕ.

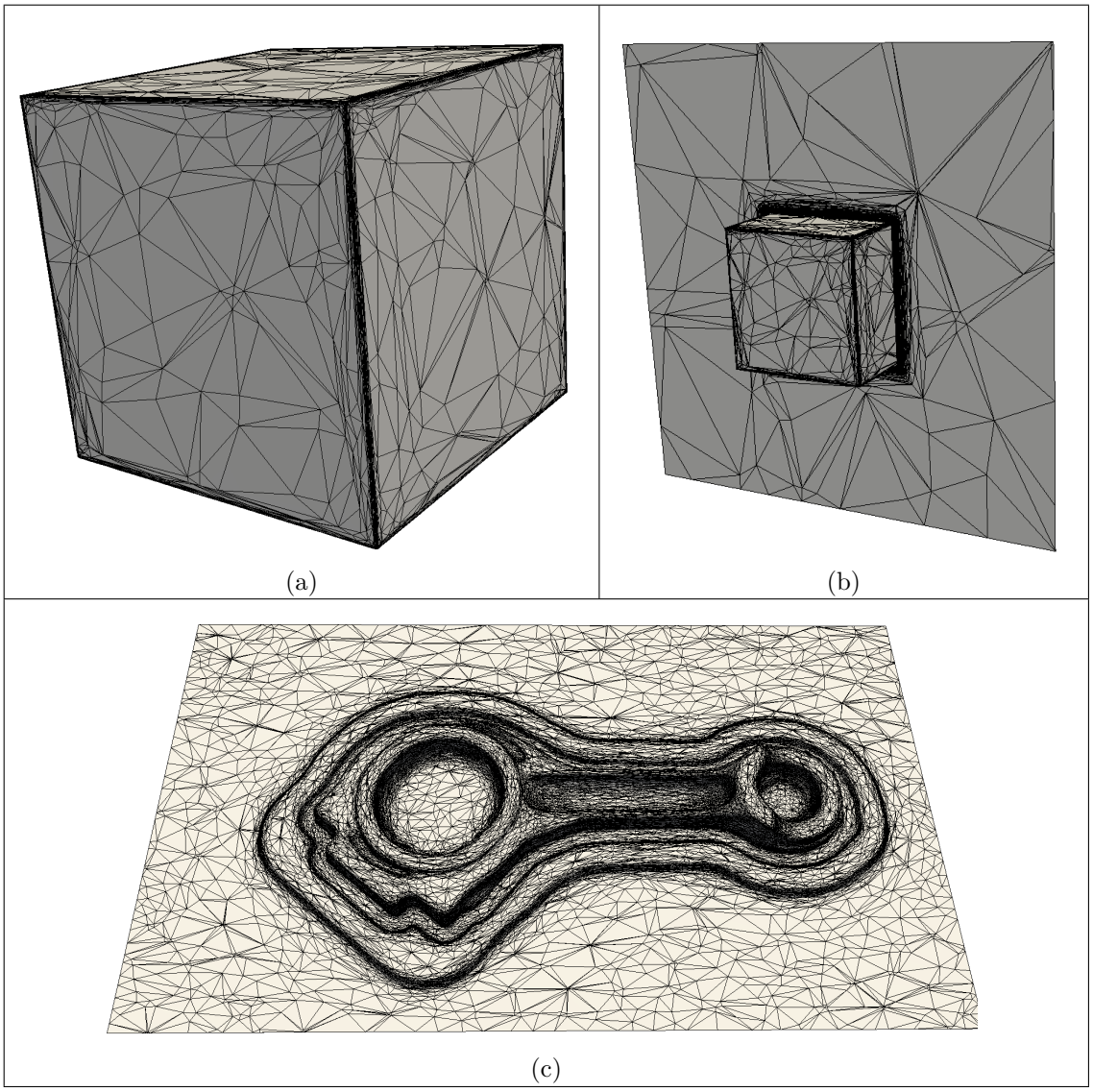


Figure 2.6: *The anisotropic mesh obtained when using the edge based error estimator metric: (a-b) for a cube, (c) for the upper tool of the connecting rod. .*

2.4.3 Extension to multi-component field

Many simulations require to adapt the mesh over a multi-field. Even though, the most important aspect is to capture the evolution of the new arbitrary surface generated by the level set motion; it come in handy to adapt the mesh over the mixture law and/or the velocity field. Though metric intersection methods are commonly used to define one metric when several ones are present [Loseille, 2008]; the author of [Coupez and Hachem, 2013] chose to extend the above theory to suit a multi-component field adaptation. This approach was validated with a driven cavity problem. the mesh was adapted not only using the velocity norm, but also using all velocity vector components.

Mainly this approach requires a global vector containing all fields to be adapted.

Denote $u = \{u_1, \cdots, u_m\}$ the vector containing those fields. Using the error estimator described in (sec. 2.4.1), the computed error is now a vector given by:

$$e_{ij} = \left\{e_{ij}^1, e_{ij}^2, \ldots, e_{ij}^m\right\} \tag{2.33}$$

Each component of the error vector is obtained by applying the already described theory on each component of u. To provide a scalar value to the error estimator; [Coupez and Hachem, 2013] proposes to evaluate the vector e_{ij} by its L^2 or L^∞ norm.

2.5 Mechanical problem

This section is devoted to elaborate an appropriate mathematical model capable of describing the large deformation problem. The desired model is written based on an Eulerian formulation of the fundamental mechanics equations. First, the system is adapted to suit this problem needs. The monolithic system is established from the weak formulation written in a velocity/pressure variables. Finally, the Finite Element formulation is explained using a Variational Multi-Scale method for stabilization.

2.5.1 Governing equations

In the framework of large deformation problem, let us consider a simple forging process illustration for a better understanding of the mechanical problem (see Figure 2.7). Since an Eulerian formulation is adopted, the computational domain $\Omega \subset \mathbb{R}^d\,(d = 2\,or\,3)$ is chosen in a way to cover all the present sub-domains, even the air. The deformable body, denoted by Ω_d, is deformed when the upper die moves uniformly in the $-z$ direction. Usually, the lower die remains fixed but nothing prevent it to move in the z direction. In this work, the temperature is assumed to be constant during the simulation. As a result, the system of equations is governed by the mass conservation equation and the momentum equation, omitting the heat transfer equation. The mass conservation equation is expressed by:

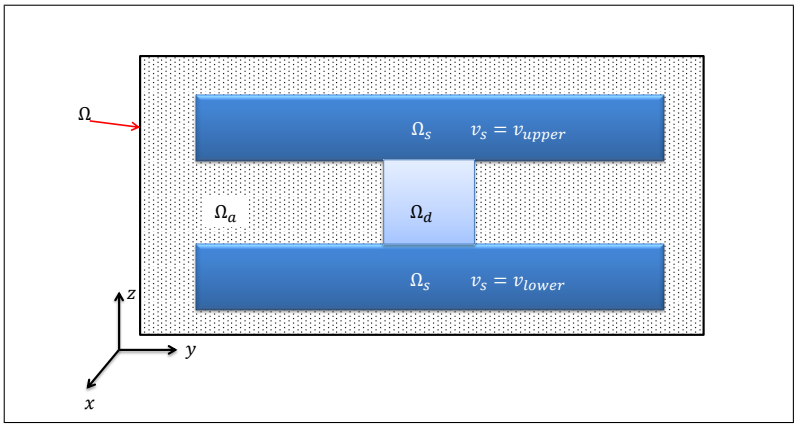


Figure 2.7: *Illustration of a 2d flat bunch simulation involving two rigid bodies (tools) and a deformable body.*

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho v) = 0 \tag{2.34}$$

where ρ is the material density and v represents the velocity. Since the computational domain is assumed to be incompressible, the equation (2.34) is reduced to:

$$\nabla.v = 0 \tag{2.35}$$

The momentum equation is given by:

$$\rho\left(\frac{\partial v}{\partial t} + v.\nabla v\right) - \nabla.\sigma = \rho g \tag{2.36}$$

in which g is the gravity and $\sigma = s - p\mathbb{I}_d$ is the Cauchy stress tensor, where p is the pressure and $\mathbb{I}_d$ is the identity matrix. The deviatoric component s is defined in terms of each sub-domain Ω_i. In Ω_d, s follows a Norton-hoff law:

$$s = 2K_d\left(\sqrt{3}\dot{\bar{\varepsilon}}\right)^{m-1}\varepsilon(v) \tag{2.37}$$

in which K_d is the consistency of the deformable body and $m\,(0 < m \leqslant 1)$ is the strain rate sensitivity coefficient. The strain rate tensor $\varepsilon(v) = \frac{1}{2}(\nabla v + {}^t\nabla v)$, while $\dot{\bar{\varepsilon}} = \sqrt{\frac{2}{3}\varepsilon(v) : \varepsilon(v)}$. In Ω_a which denotes the air, s obeys to a Newtonian fluid law:

$$s = 2\eta_a\varepsilon(v) \tag{2.38}$$

where η_a represents the air viscosity. In the rigid bodies the following equation should be verified:

$$\varepsilon(v) = 0 \tag{2.39}$$

In fact, equation (2.39) is not considered within the system. Nonetheless, a different procedure is taken into account to verify this equation. In fact, to be consistent with what was presented above, the stress deviatoric part of the rigid bodies is written as

$$s = 2K_s\varepsilon(v) \tag{2.40}$$

in which K_s, the tools consistency, should respect $K_s \gg K_d$. In addition to the previous restriction, the signed distance function ϕ_r is computed at each time step. Thus the upper die is placed into it right place. Under the latter conditions, equation (2.40) is equivalent to equation (2.39).

2.5.2 Weak formulation

In the previous section, the equations governing the physical problem were presented. Now the aim is to combine all these sub-systems in order to obtain the monolithic system defined all over the domain. To begin with, the computational domain Ω is defined to cover all heterogeneous environment including the air (see sec. 2.2.1). To obtain the monolithic system, using the weak formulation is mandatory.

Thus, we must first define Sobolev spaces in which the solution is approximated:

$$\mathcal{V} = \left\{ v,\ v \in \left(H^1(\Omega)\right)^d | v = g\ on\ \Gamma \right\} \tag{2.41}$$

$$\mathcal{Q} = \left\{ p,\ p \in L^2(\Omega), \int_\Omega p d\Omega = 0 \right\} \tag{2.42}$$

$$\mathcal{W} = \left\{ w,\ w \in \left(H^1(\Omega)\right)^d | w = 0\ on\ \Gamma \right\} \tag{2.43}$$

In the spaces defined above, the scalar product is given by:

$$(v, w)_\Omega = \int_\Omega v : w \tag{2.44}$$

A variational formulation lies mostly on the Green formula. The problem amounts to find $(v, p) \in \mathcal{V} \times \mathcal{Q} / \ \forall w \in \mathcal{W},\ \forall \Omega_i \subset \Omega$:

$$\begin{cases} \left(\rho \dfrac{\partial v}{\partial t}, w\right)_{\Omega_i} + (\rho v.\nabla v, w) + (s : \varepsilon(w))_{\Omega_i} - (p, \nabla.w)_{\Omega_i} &= (\rho g, w)_{\Omega_i} \\ (\nabla.v, q)_{\Omega_i} &= 0 \end{cases} \tag{2.45}$$

2.5.3 Monolithic system

At this stage, the monolithic system is defined all over the computational domain Ω by adding the previous weak formulations corresponding to each sub-domain Ω_i.

Find $(v, p) \in \mathcal{V} \times \mathcal{Q} / \ \forall w \in \mathcal{W}$:

$$\left(\underbrace{\rho_s H(\phi_r) + \rho_d H(\phi_d) + \rho_a H(\phi_a)}_{\rho}\right)\left(\frac{\partial v}{\partial t} + v.\nabla.v, w\right) + \left(2\left(\underbrace{K_s H(\phi_r) + K_d\left(\sqrt{3}\dot{\bar{\varepsilon}}\right)^{m-1} H(\phi_d) + \eta_a H(\phi_a)}_{\eta}\right)\varepsilon(v) : \varepsilon(w)\right)_{\Omega} - (p, \nabla.w)_{\Omega} = (\rho g, w)_{\Omega} \tag{2.46}$$

where the introduction of the Heaviside function allows to preserve the physical characteristic of each heterogeneity. In fact the terms ρ and η in equation (2.46) denotes the mixture law, it will be detailed and discussed in (sec. 2.6) and (sec. 2.7.1.2).

2.5.4 Finite Element Formulation

Since solving system (2.46) by finding an analytical solution is quite impossible, it is necessary to approximate its solution by a space and time discretization.

2.5.4.1 Spatial discretization

A finite element discretization of the monolithic system above begins by approximating the functional infinite spaces (2.41 -2.43) by finite size sub-spaces. Let $\mathcal{V}_h \subset \mathcal{V}, \mathcal{Q}_h \subset \mathcal{Q}, \mathcal{W}_h \subset \mathcal{W}$ a family of sub-spaces defined as follows:

$$\mathcal{V}_h = \left\{v_h |\, v_h \in \left(\mathcal{C}^0(\Omega)\right)^d,\ v_{h|k} \in \left(P^1(K)\right)^d, \forall K \in \mathcal{T}_h\right\} \tag{2.47}$$

$$\mathcal{Q}_h = \left\{p_h |\, p_h \in \left(\mathcal{C}^0(\Omega)\right)^d,\ p_{h|k} \in P^1(K), \forall K \in \mathcal{T}_h\right\} \tag{2.48}$$

$$\mathcal{W}_h = \left\{w_h,\ w_h \in \left(\mathcal{C}^0(\Omega)\right)^d | w_h = 0\, on\, \Gamma_h\right\} \tag{2.49}$$

In the theory of finite elements, the approximation of the solution v (for instance) on an element is determined using only its nodes values (called degrees of freedom). Then, the solution is approximated by

$$\forall K \in \mathcal{T}_h \; v_{h|k} = \sum_{i=1}^{D} v_i N^i \qquad p_{h|k} = \sum_{i=1}^{D} p_i N^i \tag{2.50}$$

in which i represents the node and N^i the shape function on this node. The numerical system is given by inserting equations (2.50) in the system (2.46). However the solution obtained by just solving this system is not accurate and represents many oscillations due to the convective terms.

2.5.4.2 Variational Multi-scale Stabilization

The accurate simulation of such problem remains a major challenge. Indeed, the convective term in (2.46) disrupts highly the solution. A variational multi-scale approach is chosen to stabilize the solution. This stabilization technique was originally proposed by [Hughes et al., 1998] . The following description of this technique is for a finite element formulation using a tetrahedral mesh. Several development and publications of VMS methods adapt this technique to other type of formulation such as the finite volume for example [Dahmen et al., 2011]. Inspired by the work of [Codina et al., 2007], [Hachem et al., 2010] developed a rigorous derivation of the VMS method in 2009. Here, a brief description of the VMS method implemented in CIMLib is presented.

The anatomy of the method relies on a two-scale decomposition of the velocity/pressure fields. Thus the following sub-spaces are introduced:

$$\widetilde{\mathcal{V}}_k = \left\{ v_h |\, v_{h|k} \in \left(P^1(K)\right)^d \cap \left(H_0^1(K)\right)^d, \forall K \in \mathcal{T}_h \right\} \tag{2.51}$$

$$\widetilde{\mathcal{Q}} = \left\{ p_h |\, p_{h|k} \in P^1(K) \cap H_0^1(K), \forall K \in \mathcal{T}_h \right\} \tag{2.52}$$

The solution is decomposed into a coarse scale and a fine scale:

$$v_h + \widetilde{v} \quad and \quad p_h + \widetilde{p} \tag{2.53}$$

Due to the orthogonality of spaces, the system of equations is split into two parts called coarse/fine scale. After Integrating by parts and considering Dirichlet boundary conditions, the coarse scale is written as :

$$\rho \left(\frac{\partial \left(v_h + \widetilde{v}\right)}{\partial t}, w \right)_\Omega + \rho \left(\left(v_h + \widetilde{v}\right) . \nabla \left(v_h + \widetilde{v}\right), w \right)_\Omega + \left(2\eta\varepsilon\left(v_h\right) : \varepsilon\left(w\right)\right)_\Omega$$
$$- \left(\left(p_h + \widetilde{p}\right), \nabla . w \right)_\Omega = \left(\rho g, w\right)_\Omega \tag{2.54}$$

$$\left(\nabla.\left(v_h+\widetilde{v}\right),q\right)=0 \tag{2.55}$$

while the fine scale is given by:

$$\rho\left(\frac{\partial\left(v_h+\widetilde{v}\right)}{\partial t},\widetilde{w}\right)_\Omega+\rho\left(\left(v_h+\widetilde{v}\right).\nabla\left(v_h+\widetilde{v}\right),\widetilde{w}\right)_\Omega+\left(2\eta\varepsilon\left(\widetilde{v}\right):\varepsilon\left(\widetilde{w}\right)\right)_\Omega$$
$$-\left(\tilde{p},\nabla.(w_h+\widetilde{w})\right)_\Omega=\left(\rho g,(w_h+\widetilde{w})\right)_\Omega \tag{2.56}$$

$$\left(\nabla.\left(v_h+\widetilde{v}\right),\tilde{q}\right)=0 \tag{2.57}$$

Solving the previous system of equations (2.54) and (2.56) is similar to the one used by a Mini Element formulation [Basset, 2006]. It normally begins by solving the fine scale on the sub-grid without being tracked in time. The VMS method in CIMLib adopts many assumptions to simplify the problem.

The non-linear term in equation (2.56) is approximated using only the coarse scale part:

$$\left(v_h+\widetilde{v}\right).\nabla\left(v_h+\widetilde{v}\right)\approx v_h.\nabla v_h+v_h.\nabla\widetilde{v} \tag{2.58}$$

Tough the resolution of the fine scale equation using both variables (v,p) provides a better stabilization, [Hachem, 2009] follows the work of [Franca and Oliveira, 2003] and approximates the pressure in equation (2.56) by:

$$\tilde{p}=-\tau_c\nabla.v \tag{2.59}$$

For the definition of τ_c, he adopts the one proposed by [Codina and Principe, 2007]

$$\tau_c=\left[\nu^2+\left(\frac{c_2|v|_k}{c_1h}\right)^2\right]^{1/2} \tag{2.60}$$

where c_1 and c_2 are two constants, independent of h. h is the characteristic length of an element and ν represents the kinematic viscosity. Now equation (2.59) can be integrated directly into the coarse scale equation (2.54). Thus, the fine scale equation is reduced to the following:

$$\begin{array}{rcll}\rho\left(v_h.\nabla\widetilde{v},\widetilde{w}\right)_\Omega+\left(2\eta\varepsilon\left(\widetilde{v}\right):\varepsilon\left(\widetilde{w}\right)\right)_\Omega & = & \left(\mathcal{R}_m,\widetilde{w}\right)_\Omega & \forall\widetilde{w}\in\widetilde{\mathcal{W}}\\ \left(\nabla.\widetilde{v},\tilde{q}\right) & = & \left(\mathcal{R}_c,\tilde{q}\right)_\Omega & \forall\tilde{q}\in\widetilde{\mathcal{Q}}\end{array} \tag{2.61}$$

where $\mathcal{R}_c$ designs the term $-\nabla.v_h$ and $\mathcal{R}_m$ gathers all terms considered in the second

member of the equation :

$$\mathcal{R}_m = \left(\rho g - \rho\frac{\partial v_h}{\partial t} - \rho\left(v_h + \widetilde{v}\right).\nabla v_h - \nabla p_h, \widetilde{v}\right) \tag{2.62}$$

Many methods were developed capable of solving the latter system. A full review can be found in [Hachem, 2009]. In CIMLib, the used approach consists on employing the bubble functions as the shape functions on each element. Moreover since the problem is a convection problem, the shape functions are modified to an upwind shape functions. Thus, the solution of (2.61) is given by :

$$\widetilde{v} = \frac{1}{\rho\left(v^{i-1}.\nabla b, b^*\right) + \left(2\eta\varepsilon\left(b\right) : \varepsilon\left(b\right)\right)}(\mathcal{R}_m, b) \tag{2.63}$$

where b represents the bubble shape functions.

As for the coarse scale equation, the non linear term is approximated by keeping the terms of the first order at the i^{th} iteration as:

$$\left(v.\nabla v, w\right) = \left(v^i.\nabla v^{i-1}, w\right) + \left(v^{i-1}.\nabla v^i, w\right) - \left(v^{i-1}.\nabla v^{i-1}, w\right) \tag{2.64}$$

where u^{i-1} is the previous know Newton-Raphson's iteration. After extracting $\tilde{v}$ the solution of the fine scale problem, equation (2.54) is integrated by parts and $\tilde{v}$ is substituted by it value, the coarse scale is expressed by equations (2.65 - 2.66).

$$\begin{aligned}
&\underbrace{\rho\left(\frac{\partial v_h}{\partial t}, w_h\right)_\Omega + \rho\left(v_h.\nabla v_h, w_h\right)_\Omega + \left(2\eta\varepsilon\left(v_h\right) : \varepsilon\left(w_h\right)\right)_\Omega - \left(p_h, \nabla.w_h\right)_\Omega + \left(\nabla.v_h, q_h\right) - \left(\rho g, w_h\right)}_{A} \\
&+ \underbrace{\sum_{K\in\mathcal{T}_h}\left(\tau_k\rho\frac{\partial v_h}{\partial t} - \rho v_h.\nabla v_h + \nabla p_h - \rho g, \rho v_h^{i-1}\nabla.w_h\right)_\Omega}_{B} \\
&+ \underbrace{\sum_{K\in\mathcal{T}_h}\left(\tau_k\rho\frac{\partial v_h}{\partial t} - \rho v_h.\nabla v_h + \nabla p_h - \rho g, \nabla.q_h\right)_\Omega}_{C} \\
&+ \underbrace{\sum_{K\in\mathcal{T}_h}\left(\tau_c\nabla v_h, \nabla.w_h\right)_\Omega}_{D} = 0
\end{aligned} \tag{2.65}$$

$$\left(\nabla.v_h, q_h\right) - \sum_{K\in\mathcal{T}_h}\left(\tau_k\mathcal{R}_m, \nabla.q_h\right)_\Omega = 0 \tag{2.66}$$

Comparing with a standard Galerkin formulation, the terms denoted A in equation

(2.65) refers to a standard Galerkin formulation, the terms denoted B represents the upwind stabilization terms and the terms denoted C refers to the pressure stabilization terms. Where the last term denoted by D is induced by the enrichment of the pressure spaces. The appearance of this new term will not interfere with the existence and unicity of the solution i.e. equation (2.65) satisfies the inf-sup condition for the velocity and pressure interpolations.

2.5.4.3 Stabilization parameter

Following the same analogy of [Hachem, 2009], this section presents the different evaluation of the stabilization parameter τ_k. A comparison will be presented between a condensed Mini Element formulation and the VMS stabilization technique. To eliminate all ambiguities matrix notations shown are similar to those in a Mini Element formulation. Using a the latter formulation, the local matrix formulation of the problem presented above is given by:

$$\begin{bmatrix} A_{vv} & A_{vb}^t & A_{vp}^t \\ A_{vb} & A_{bb} & A_{bp}^t \\ A_{vp} & A_{bp} & 0 \end{bmatrix} \begin{bmatrix} v_h \\ \widetilde{v}_h \\ p_h \end{bmatrix} = \begin{bmatrix} B_v \\ B_b \\ B_p \end{bmatrix} \tag{2.67}$$

when using a regular bubble function, (i.e. the bubble function is centered in the element) the terms in velocity/bubble will be vanished. Solving (2.67) consists into expressing $\widetilde{v}_h$ in function of v_h and p_h. Then by injecting it in the third and first line, the final matrix system is obtained. Therefore the inversion of the matrix A_{bb} is quite necessary when adopting such formulation. The genuinely of the VMS method is it capacity to tune automatically the stabilization parameter. In fact, the matrix formulation of the problem presented above is given by:

$$\begin{bmatrix} A_{vv} & A_{vp}^t \\ A_{vp} & A_{pp} \end{bmatrix} \begin{bmatrix} v_h \\ p_h \end{bmatrix} = \begin{bmatrix} B_v \\ B_p \end{bmatrix} \tag{2.68}$$

where the additional terms appearing in equations (2.65 - 2.66) are explicitly present in the matrix according to their correspondence. Reasoning by analogy, the A_{pp} contribution in the VMS formulation is equivalent to the $A_{bp}A_{bb}^{-1}A_{bp}^t$ contribution after the condensation of the system (2.67). In what follows, the choice of τ_k is discussed. This choice has been the subject of several studies. Many works evaluate τ_k by computing the local Reynolds number. In addition, it depends highly of the problem type: convection or diffusion [Medic and Mohammadi, 1999]. In CIMLib, the adopted parameter is computed from Fourier analysis proposed by [Codina, 2002]. It combines both regimes and evaluated as follows:

$$\tau_k = \left[\left(\frac{c_1 \nu}{h^2} \right)^2 + \left(c_2 \frac{| v |_k}{h} \right)^2 \right]^{-\frac{1}{2}} \tag{2.69}$$

where c_1 and c_2 are two algorithmic constants chosen equal to 4 and 2 respectively for linear elements. Notice that the previous stabilization parameter is set to be a static formulation. For a time dependent problem, [Tezduyar and Osawa, 2000a] proposed the following stabilization term

$$\tau_k = \left[\left(\frac{2}{\Delta t} \right)^2 + \left(\frac{c_1 \nu}{h^2} \right)^2 + \left(c_2 \frac{| v |_k}{h} \right)^2 \right]^{-\frac{1}{2}} \tag{2.70}$$

For anisotropic meshes with highly stretched elements, the definition of the mesh size within each element is still an open problem [Harari and Hughes, 1992]. Different definitions are proposed in [Principe and Codina, 2010] and [Tezduyar and Osawa, 2000a] for instance. In CIMLib, the mesh size element is computed in terms of the velocity v, its norm and the derivative of the shape function N as follows:

$$h = \sum_i^{n_e} \left| \frac{v}{\|v\|} \frac{\partial N}{\partial x_i} \right| \tag{2.71}$$

where n_e denotes the number of nodes per element.

2.6 Mixture laws

After defining all sub-domains geometrically with a distance function, the physical characteristics of each sub-domain are determined by mixture laws. In this section, different types of mixture laws are described briefly. The impact of using each of these mixture laws is discussed in the next section. In fact, to represent density and viscosity discontinuities over the interface a Heaviside function is needed. It is equal to 1 if the mesh element is fully filled and equal to 0 if it is not. If the material interface passes through an element (i.e. mesh element partially filled), the Heaviside function $H(\phi)$ is computed as follows:

$$H(\phi)_{|k} = \frac{1}{2} \left(1 + \frac{\phi_k^+}{|\phi|_k} \right) \quad then\ H \in [0, 1] \tag{2.72}$$

where ϕ_k^+ is the sum of all signed distance functions evaluated at each node of the element k and $|\phi|_k$ is the sum of the absolute value of ϕ.

By choosing to calculate the Heaviside function in this manner, the mixture law will be similar to the Volume of Fluid Method (VOF) where the fraction of the volume will

be H. Then, the mixture law is given by:

$$\eta = \sum_{i=1}^{n} H(\phi)_{|k} \eta_i \tag{2.73}$$

2.6.1 Linear mixture law

In computations, to achieve numerical robustness, a smoothed Heaviside is often used:

$$H_\varepsilon(\phi) = \begin{cases} 1 & if\ \phi > \varepsilon \\ \frac{1}{2}\left(1 + \frac{\phi}{\varepsilon}\right) & if\ \mid \phi \mid \leqslant \varepsilon \\ 0 & if\ \phi < -\varepsilon \end{cases} \tag{2.74}$$

where generally ε is taken as the interface thickness interface (see sec. 2.2.2.1). The regularized Heaviside will ensure a lighter transition between the different values of η corresponding to the different heterogeneity. Notice that, in Figure 2.8, the difference in the transition is very clear to the naked eye.

On the left, the P0-mixture law (2.73) is computed relatively per element. Thus depends highly on the refinement of the mesh. On the right, the mixture law is computed on each node. It does not relay on the mesh refinement as the P0-mixture law does. It is called the Linear mixture law and is given by:

$$\eta(H_\varepsilon) = \sum_{i=1}^{n} \eta_i \times H_\varepsilon(\phi_i) \tag{2.75}$$

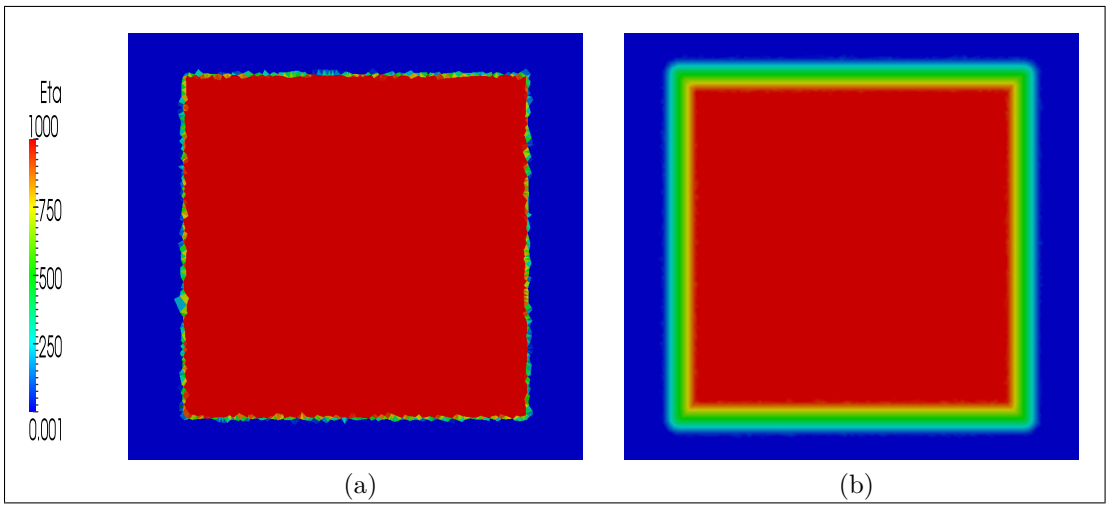


Figure 2.8: *Different Types of mixture laws: (a) P0-mixture law and (b) P1-mixture law*

2.6.2 Novel mixture law: Quadratic law

In this work, a new mixture law is implemented in CIMLib. This section will just describe the method, while the utility of this law will be discussed in chapter 5. It will be used to manage the sliding between two different heterogeneities. The main idea is to introduce a third viscosity between the two other heterogeneities. The newly added viscosity represents the viscosity of the lubricant. The advantage of this method is that lubricant is added implicitly. Thus no need to deal with it displacement. It is given by the mixture law.

To obtain the desired mixture law, one must choose a function admitting a minimum. Therefore, the general form of a quadratic function is considered (see equation (2.76)). This minimum will represent the viscosity of the lubricant.

$$\begin{array}{rccl} \Phi: & [0,1] & \rightarrow & \mathbb{R} \\ & H_\varepsilon & \rightarrow & \eta(H_\varepsilon) = a \times H_\varepsilon^2 + b \times .H_\varepsilon + c \end{array} \tag{2.76}$$

where

$$\begin{array}{rcl} a & = & \left[-2(\eta_L - \eta_0) + \sqrt{(\eta_L - \eta_0)(\eta_L - \eta_1)} + (\eta_1 - \eta_0)\right] \\ b & = & \left[2(\eta_L - \eta_0) - \sqrt{(\eta_L - \eta_0)(\eta_L - \eta_1)}\right] \\ c & = & \eta_0 \end{array} \tag{2.77}$$

In equation (2.77), η_0 and η_1 represents respectively the consistence of the deformable body and the tool, while η_L represents the lubricant viscosity. Figure 2.9 depicts the difference between a linear and a quadratic mixture law.

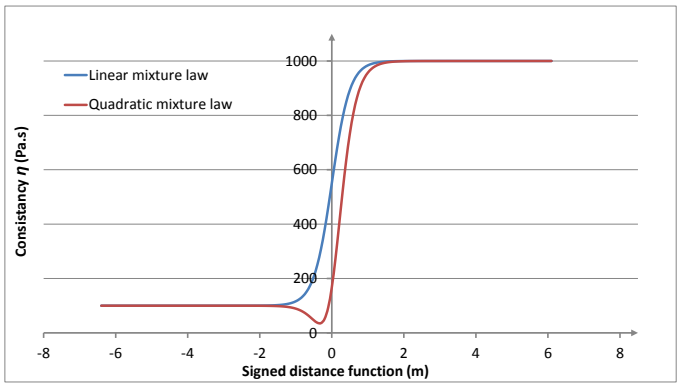


Figure 2.9: *Comparison between a linear mixture law (blue) and a quadratic mixture law (red)*

Parameters	Values
ρ_r	5000
ρ_d	5000
ρ_a	1
η_r	10^7
η_d	10^6
η_a	0.1

Table 2.1: *Parameters used in the flat bunch simulation*

2.7 New additions and Numerical Applications

2.7.1 Test cases and proposed ameliorations

2.7.1.1 Optimal parameter choices

This simulation consists on compressing a workpiece between two flat dies, as described in (Figure 2.7). The upper die presses the workpiece with a constant velocity v_{imp}=0,25 m/s, while the lower die stays fixed. As mentioned earlier, the material obeys the Norton-Hoff behavior law with a strain rate sensitivity coefficient fixed equal to 1. Table 1 shows the different parameters used in this simulation.

The initial height of the workpiece is $0.5\,m$.

After several tests, some deficiencies were noticed such as mass loss. In order to solve this deficiency, a short parametric analysis is presented next. Several parameters are proved to have a great impact on the mass loss appearance: the time step, the mixture law choice, and the boundary condition imposition.

Several improvements are introduced and presented next highlighting their positive effect on minimizing the mass loss:

1. Since the use of a really small time step is out of question do to the time of the simulation, we introduced a variable time step in CIMLib.

The heterogeneous time step is introduced using a formula in terms of the workpiece initial height h_0, the workpiece height h_i at the current increment i and the initial time step Δt_0:

$$\Delta t_i = \frac{h_i}{h_0}\Delta t_0$$

Figure 2.10 shows better mass conservation around 14% using the variable time step.

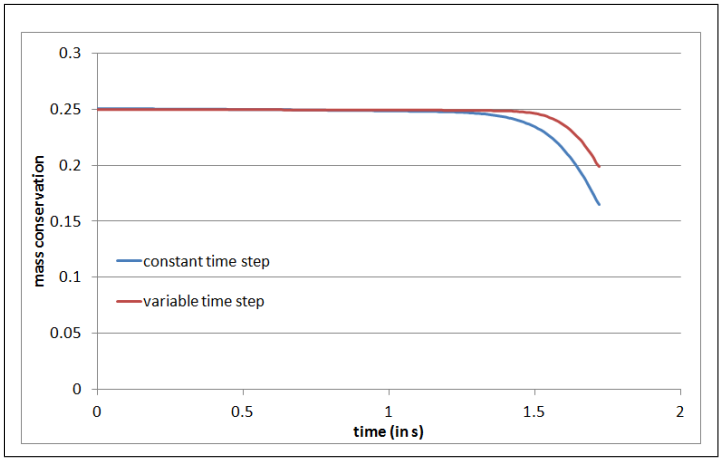


Figure 2.10: *The effect of using a variable time step on the mass conservation..*

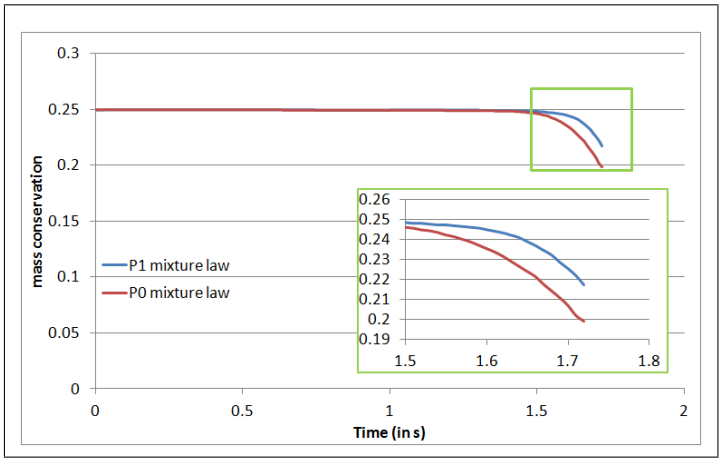


Figure 2.11: *The effect of using P1mixture law instead of a P0 mixture law on the mass conservation.*

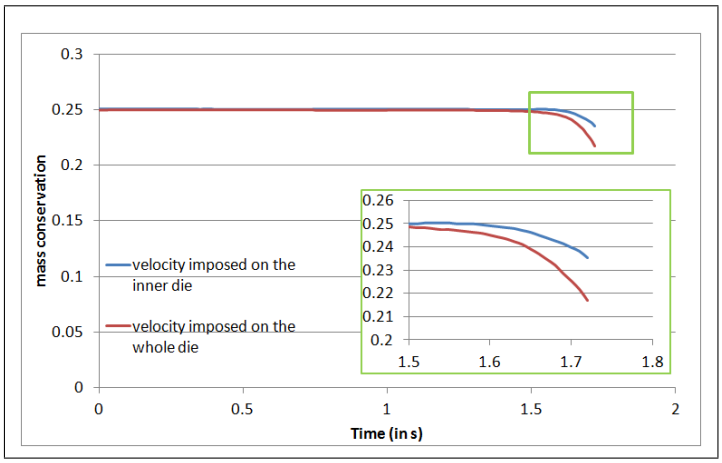


Figure 2.12: *The effect of imposing the velocity on the inner of the dies*

2. we changed the mixture law from P0 to P1 interpolation. Figure 2.11 shows the result of this change. An improvement of 8% is noted.

3. Last but not least, we treated the imposition of the dies boundary condition differently. Since a P1 mixture law is used, a critical transition zone is present when the contact is established. That is why, an inner part of the die is defined taking into account the transition zone. For every die defined on $L \times l$, its inner part is defined at an appropriate distance of the interface e.g. $(L - \varepsilon) \times (l - \varepsilon)$ where ε is the same parameter used for smoothing the Heaviside function in the mixture law. Now, the velocity is no longer imposed on the whole die, it is only imposed on its inner part.

All these improvements combined, minimize remarkably the mass loss. From now on and unless mentioned otherwise, they will be used in all applications presented in this work.

2.7.1.2 Weight of mixture laws

The issue remaining is the shape of the forged piece. As Figure 2.13 shows, using equation (2.75) to compute the mixture law affects the form obtained. In fact using such law allows higher consistency to outweighs the smaller one. It leads to preserving the initial trace of the deformable body on the tools. In addition, it prevents the deformable body from entering perfectly in contact with the tools during the simulation as shown in Figure 2.13. Therefore, we need to consider new mixture laws by changing the weight of the coefficients η_i (associated to the sub-domain i).

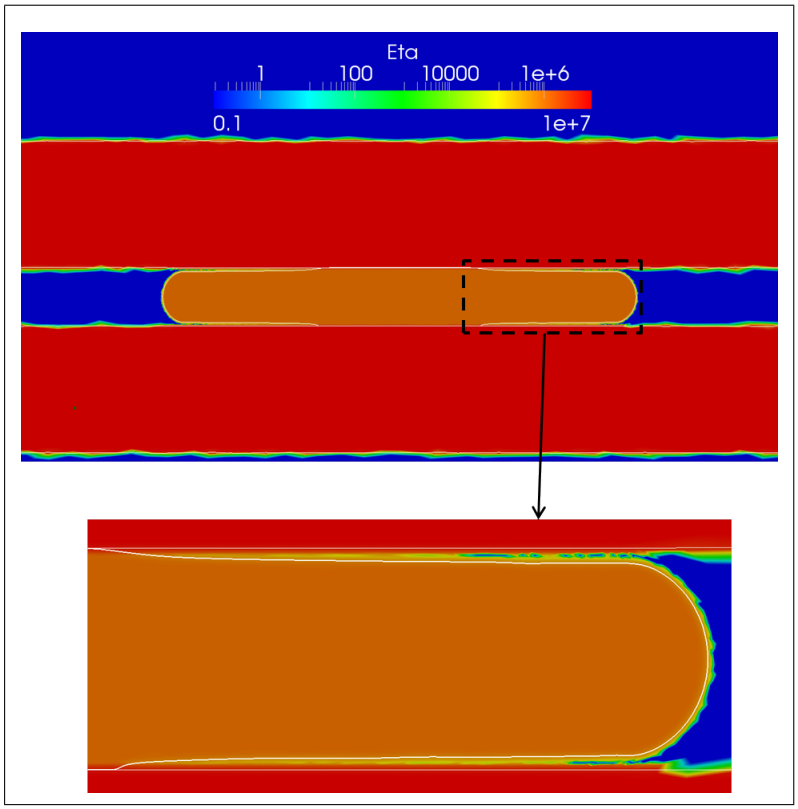


Figure 2.13: *Flat bunch simulation using a linear coefficient for mixture law.*

Three weights are presented : **i)** the inverse coefficients $\frac{1}{\eta_i}$, **ii)** the geometric mean $\sqrt[n]{\eta_i}$ where n is the number of phase and **iii)** the logarithmic coefficients $log_{10}\eta_i$. The new mixture laws are written as in (2.78 - 2.80). These new mixture laws modify the consistency only in the vicinity of the interface of the different sub-domains by narrowing the gap between their consistencies.

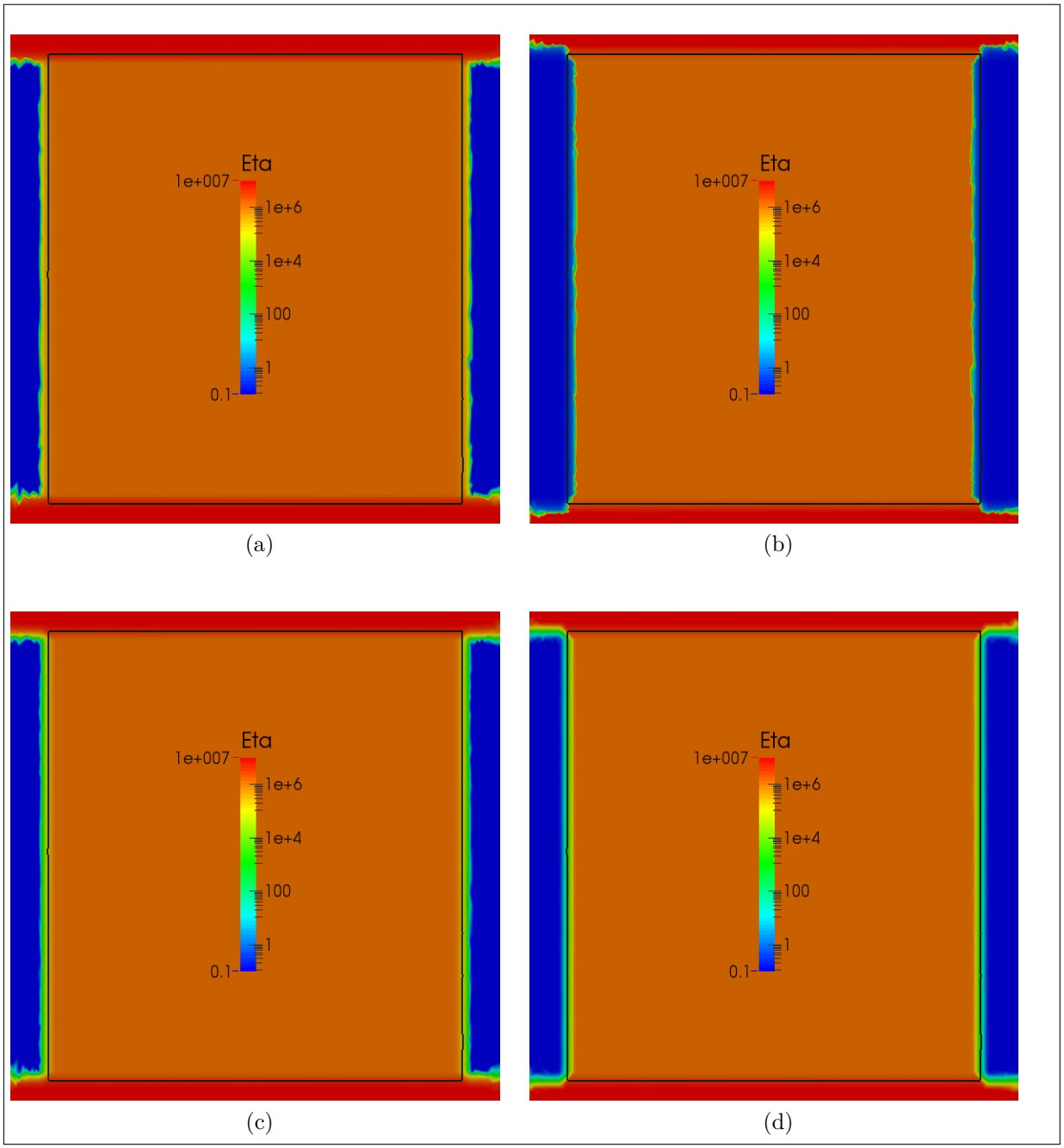


Figure 2.14: *Representation of the mixture law for a flat bunch simulation using different coefficient weights: (a) linear - (b) inverse - (c) geometric - (d) logarithmic .*

Figure 2.14 shows a comparison between the four mixture law presented above. As mentioned previously when using linear coefficients, the body with the higher consistency outweighs the body with the smaller one. But when using $\frac{1}{\eta_i}$, the smaller consistency outweighs the higher consistency. The geometric mean and the logarithmic mixture laws are the best reducing the gap between the different consistencies. Still, the logarithmic weight is the only one capable of centering the transition between two different values on the interface.

$$\frac{1}{\eta(H_\varepsilon)} = \sum_{i=1}^{n} \frac{1}{\eta_i} \times H_\varepsilon(\phi_i) \tag{2.78}$$

$$\sqrt[n]{\eta(H_\varepsilon)} = \sum_{i=1}^{n} \sqrt[n]{\eta_i} \times H_\varepsilon(\phi_i) \tag{2.79}$$

$$log_{10}\eta(H_\varepsilon) = \sum_{i=1}^{n} log_{10}\eta_i \times H_\varepsilon(\phi_i) \tag{2.80}$$

Next, we decide to launch the flat bunch simulation using only three different mixture laws defined in (2.75), (2.78) and (2.80). The geometric mean coefficient is not used since it is very similar to the logarithmic coefficient.

Figure 2.15 shows clearly that the inverse coefficient mixture law is poorly adapted to our case. When the deformable body does not perfectly touch the tool (dashed lines) i.e. the gap is filled with air. Knowing that the air has the smallest viscosity the inverse mixture law will give it advantage on the other sub-domains. Numerically, this means that these gaps will be preserved during the whole simulation and will affect the form of the final piece (which is not the case in reality).

Figure 2.16 confirms that the logarithmic weighs of the mixture law is the one with the best results. It offers a smooth transition without any remarkable imperfections. The transition between different consistencies are perfectly centered on the interface between the sub-domains. In addition, Figure 2.17 depicts the top view of the deformable body obtained in a 3d flat bunch simulation. A comparison between the classical linear mixture law and the logarithmic one displayed on the zero isovalue confirms that the contact problem confronted using the linear mixture law (Figure 2.13) faded away when using the logarithmic one : At the end of the simulation the deformable body is perfectly flat.

For all these reasons, the logarithmic weighs mixture law will be used in all our simulations.

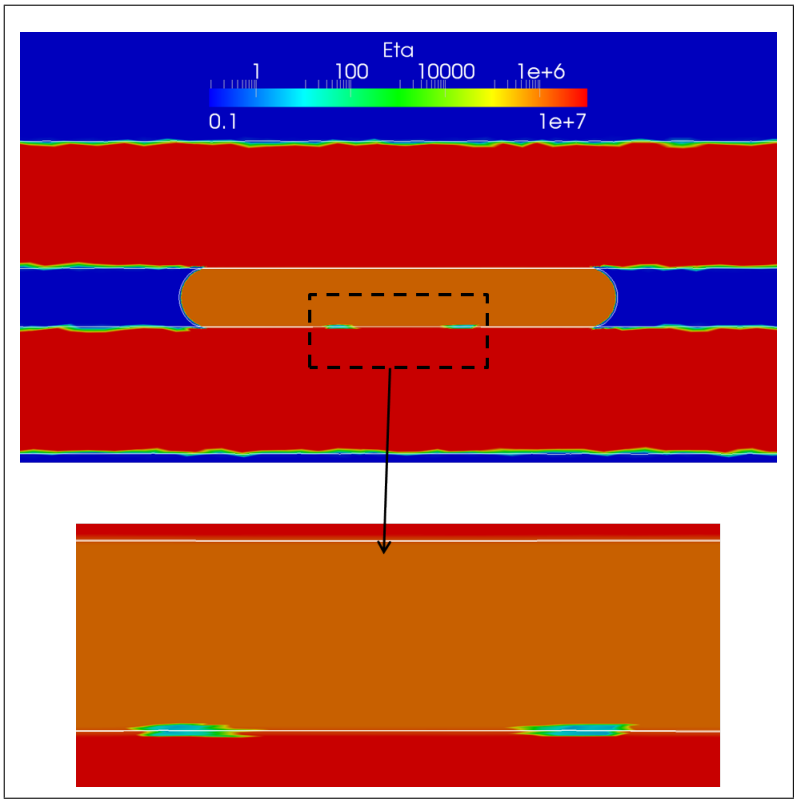


Figure 2.15: *Flat bunch simulation using a inverse coefficient for mixture law.*

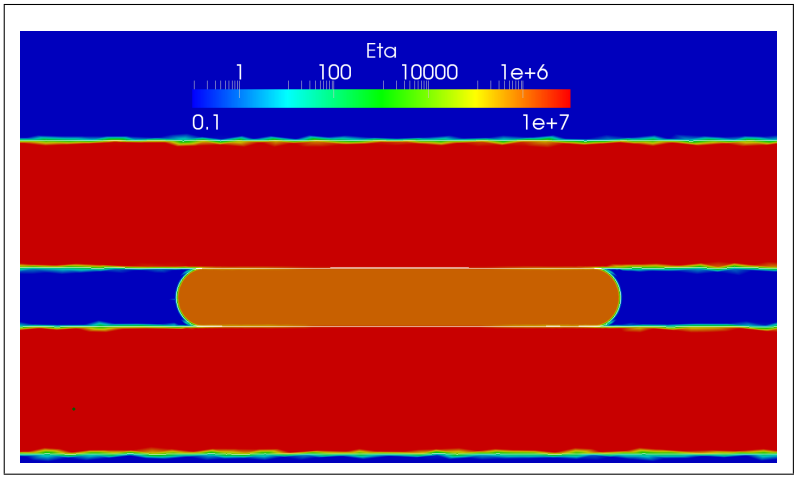


Figure 2.16: *Flat bunch simulation using a logarithmic coefficient for mixture law.*

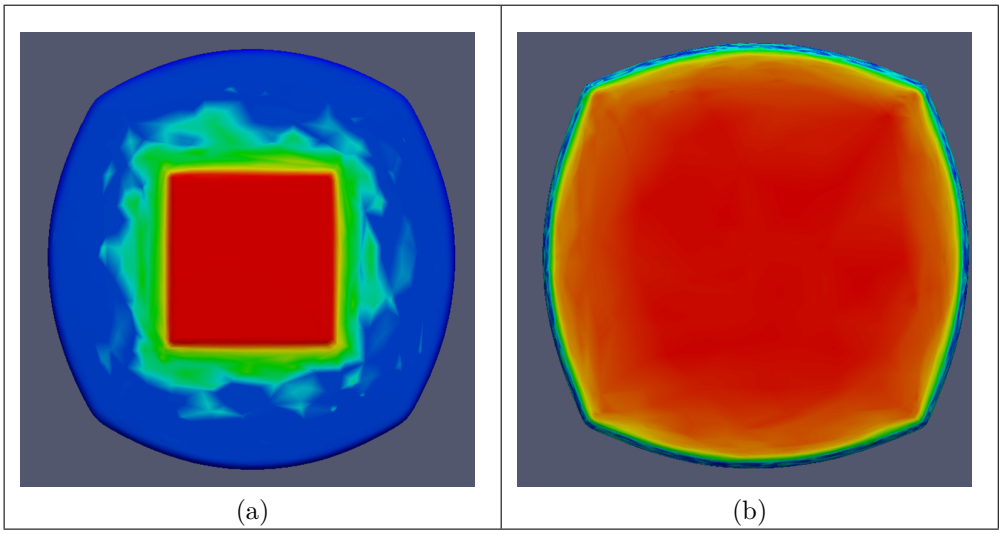


Figure 2.17: *Top view of the zero iso-value of a flat bunch simulation in 3d showing the mixture law when using: a- linear coefficient, b- logarithmic coefficient.*

2.7.1.3 Transport stabilization

In more than one occasion, stabilization problems were confronted in the convection solver *levellerT*. They were more noticeable in cases displaying highly anisotropic meshes. These numerical oscillations are mainly represented by positive level set values, indicating new material appearance in unexpected places. Figure 2.18 shows one example exhibiting instabilities in the computational domain. The physical characteristics are not mentioned since the problem is purely numerical.

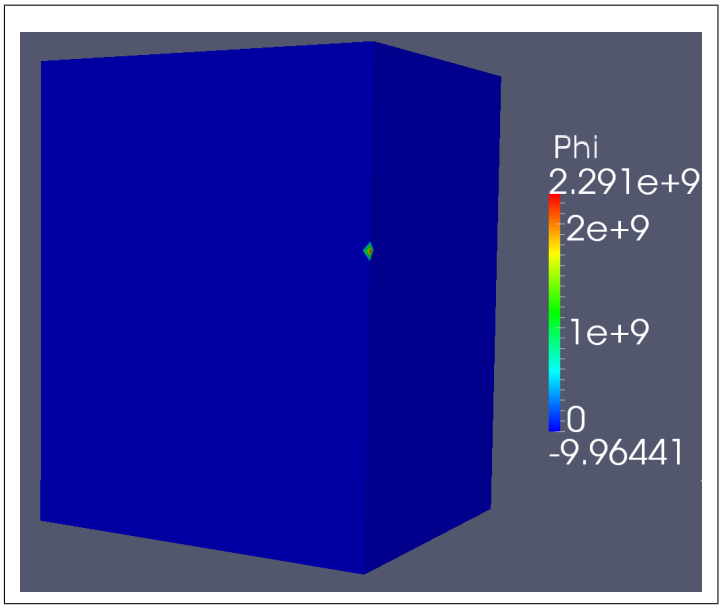


Figure 2.18: *Instability of the levellerT solver.*

After lengthy analysis, we found the origin of this instability. It is nothing other than the mesh size defined in the solver. As shown in Figure 2.19, in CIMLib, the size of every stretched element is determined as the smaller edge of the circumscribed rectangular. We propose another definition for the mesh size noted h previously. It can be defined as the element altitude (Figure 2.19). Knowing that it is not the most optimal definition, it remains logical since it is determined in the material flow direction.

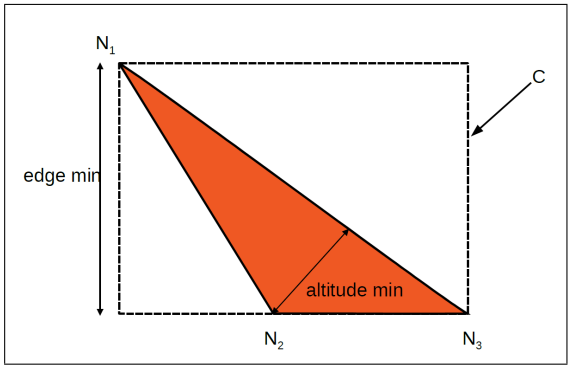


Figure 2.19: *Two different ways to compute the mesh size .*

To clarify the impact of the new definition, we use a 2D-unity domain containing 6 stretched elements illustrated in Figure 2.20. The different mesh elements were computed and presented in both Figure 2.20 and Table 2.2. Notice that the new value $Altitude_{min}$ is always smaller or equal to the older $Edge_{min}$ value. It is particularly smaller when the element is stretched (for the elements numbered 2 and 5 for instance).

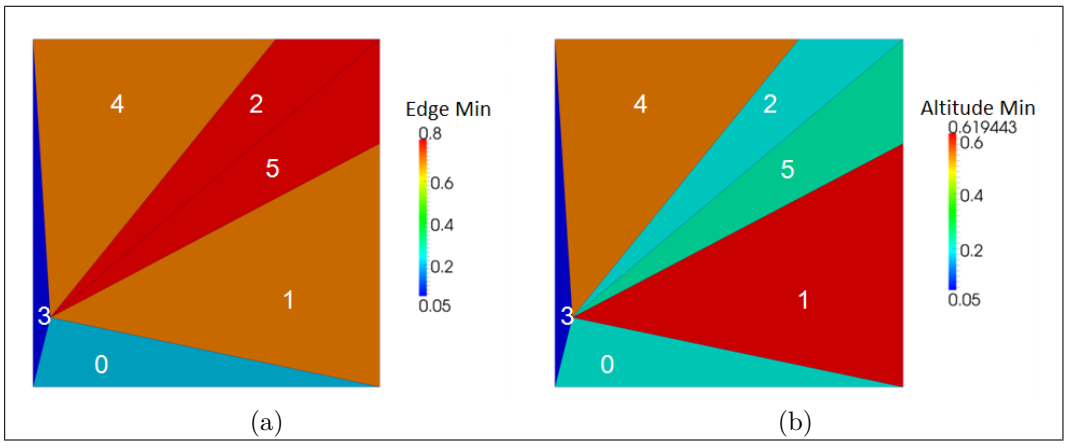


Figure 2.20: *Comparison between two methods to compute the mesh size in* CIMLib*: (a) using the $Edge_{min}$* definition and (b) using the *$Altitude_{min}$ definition.*

Simplex Id	Edge min	Altitude min
0	0.2	0.2
1	0.7	0.619443
2	0.8	0.193241
3	0.05	0.05
4	0.7	0.54328
5	0.8	0.229473

Table 2.2: Comparison between the different definitions to compute the simplex size.

The new smaller mesh size $Altitude_{min}$ stabilize the solver by respecting better the relationship binding h and the virtual time step τ (defined earlier in the reinitialized convected level set equation in sec. 2.2.2.2). After this adjustment, the same 3D case (Table 2.1) presenting instabilities in Figure 2.18, is launched again to check if the problem persists. A logarithmic weight is used for the mixture law coefficients.

Figure 2.21 illustrates the evolution of the zero iso-value of the deformable body at different time steps.

Notice that, the mesh is anisotropically adapted/refined on the interface of the body and is coarse elsewhere. The body is deformed uniformly everywhere; we should mention though that at $t = 2.975\,s$, the deformable body shows an impression and a sort of out of plane displacement. This caused by the small tools along with the boundary conditions imposed : $\vec{v}.\vec{n} = 0$. The deformable body has no way to move but in this manner. As approved in Figure 2.21 and even though the anisotropic mesh presents highly stretched elements, no numerical oscillations were detected.

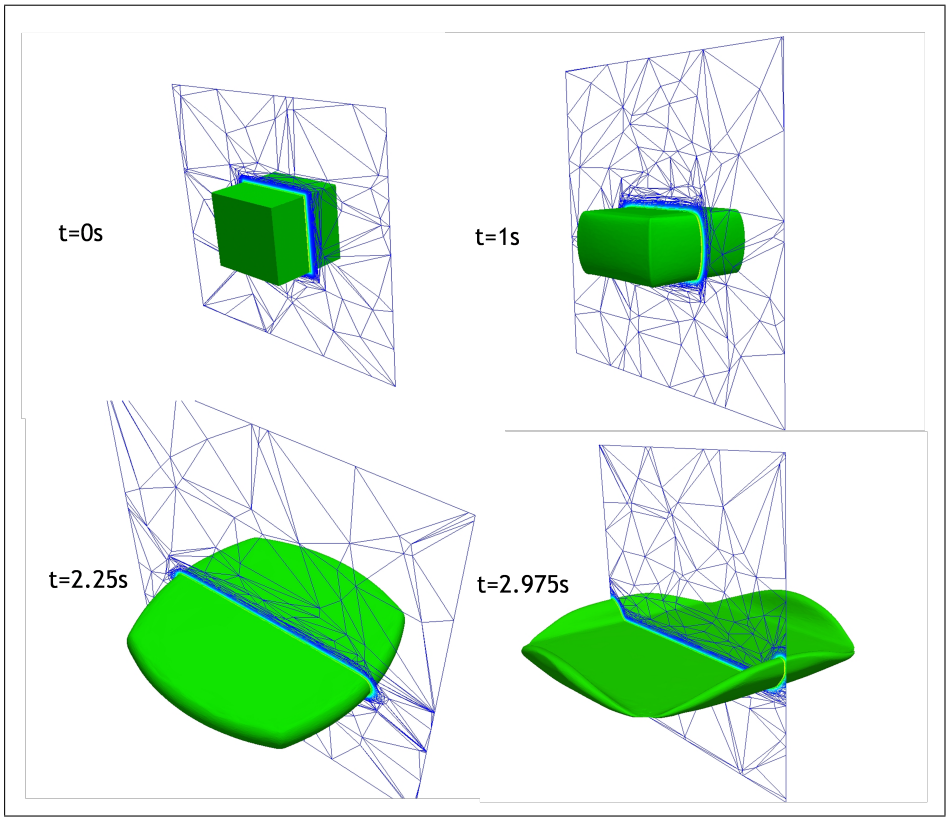


Figure 2.21: *A 3d flat bunch simulation after the ameliorations (logarithmic weigh, new mesh size definition ...).*

Note that this new mesh size definition will be applied in all following simulations to prevent any instabilities.

2.7.2 Air trapping

During a multi-domain problem, the lubricant or the air can be trapped between different interacting bodies. This section addresses the trapping issue between a rigid body and a deformable one. Figure 2.22 illustrates a problem in which the upper tool presents a porosity. Notice that the air (blue domain) fills the porosity between the upper tool (red domain) and the deformable body (orange domain) as presented in Figure 2.23 .

To treat this topic in a Lagrangian approach, an additional function must be introduced to the code, allowing to locate pockets of air or lubricant. The purpose of such a function is to take into account the impact exerted by the volume variation of the lubricant or the trapped air. In fact, the volume variation introduces a pressure variation exerted to the deformable body preventing it to fill the porous tools.

However in our Eulerian approach, their is no need to implement a function defining the air presence. The computational domain is defined to cover all heterogeneity including

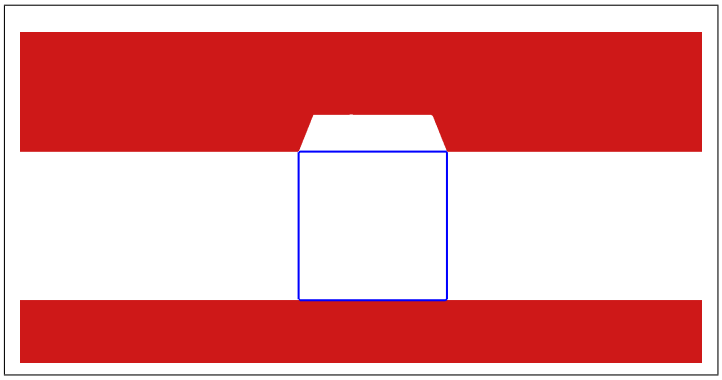


Figure 2.22: *Domain considered using a porous upper tool to show the Eulerian approach capacity to detect trapped air .*

the air or the lubricant if existent.

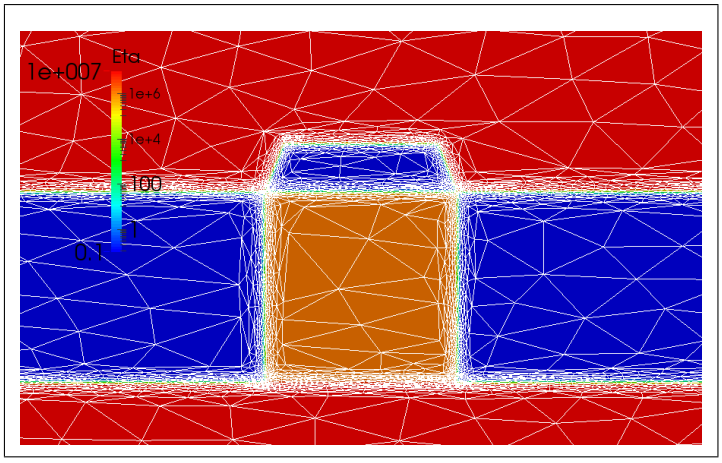


Figure 2.23: *Initial mesh and mixture law for the problem in Figure 2.22.*

Similar to the previously presented case, the kinematic of the tools is imposed on their inner defined as follows :

$$\Gamma_t^{inner}(x) = \left\{ \vec{x} \in \Omega, \phi_s(\vec{x}, t) > \frac{\varepsilon}{2} \right\} \tag{2.81}$$

Figure 2.24 shows the evolution of the isovalue zero of ϕ_d when considering the boundary conditions (2.81). This imposition generates a great pressure in the trapped air. Combined with the incompressible assumption, they prevent the deformable body from taking the tool shape. In addition, the air pockets are clearly detected in Figure 2.24(c-d). When simulating industrial cases with complex geometries, these problems (trapped air and pockets) are faced frequently.

Following this discussion, two paths can be proposed depending on what is more interesting for the study. The first focuses on the pockets formation due to the trapped air. The second, discussed in the next section, consists on creating vents to ensure the

filling of the tools.

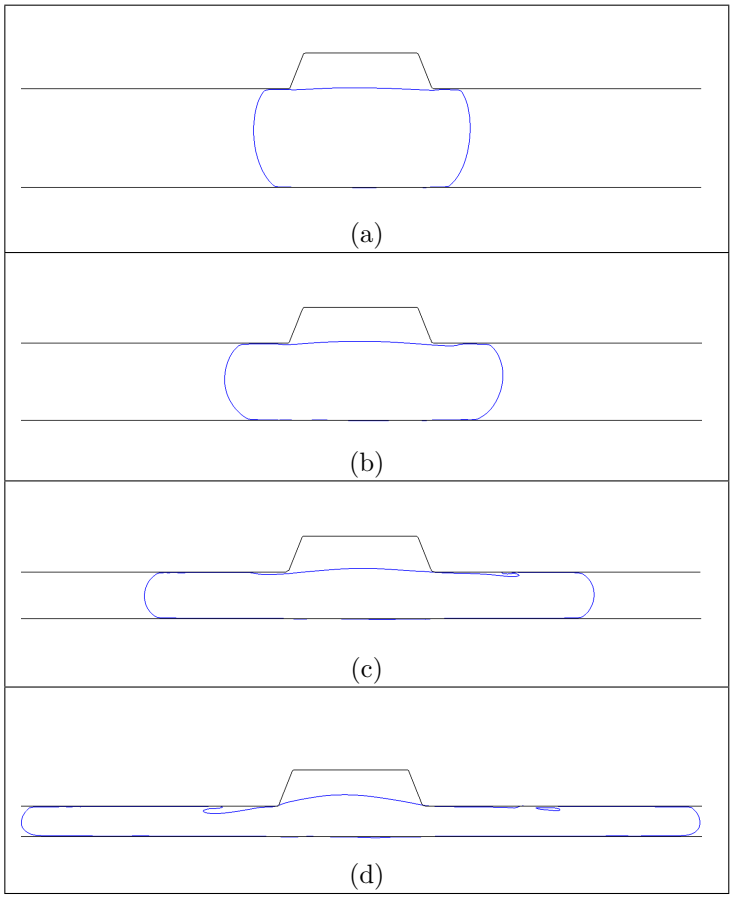


Figure 2.24: *The effect given by the imposition of the kinematic on the inner of the upper tool.*

2.7.3 Porous boundary conditions

To overcome the trapping effect, a new boundary condition called porous boundary condition, is defined. The kinematic of the tools is imposed accordingly to the new boundary condition allowing the escape of the air.

To define the porous boundary condition, we define the following set of nodes denoted Γ_t^ϕ:

$$\Gamma_t^\phi(x) = \{\vec{x} \in \Omega, \phi_d(\vec{x}, t) > -0.9 \times \varepsilon\} \tag{2.82}$$

Γ_t^ϕ represents a fictitious domain slightly bigger than the actual deformable body. The next step is to define the porous boundary condition as the intersection of Γ_t^ϕ and the classical boundary condition:

$$\Gamma_p = \Gamma_t^{inner} \cap \Gamma_t^\phi \tag{2.83}$$

Using equation (2.83) is equivalent to impose the kinematic only on the vicinity of the contact zone and letting the air escape otherwise.

To test the efficiency of the proposed porous boundary condition, the same case treated earlier is used. This time the kinematic are imposed on Γ_p instead of Γ_{inner}.

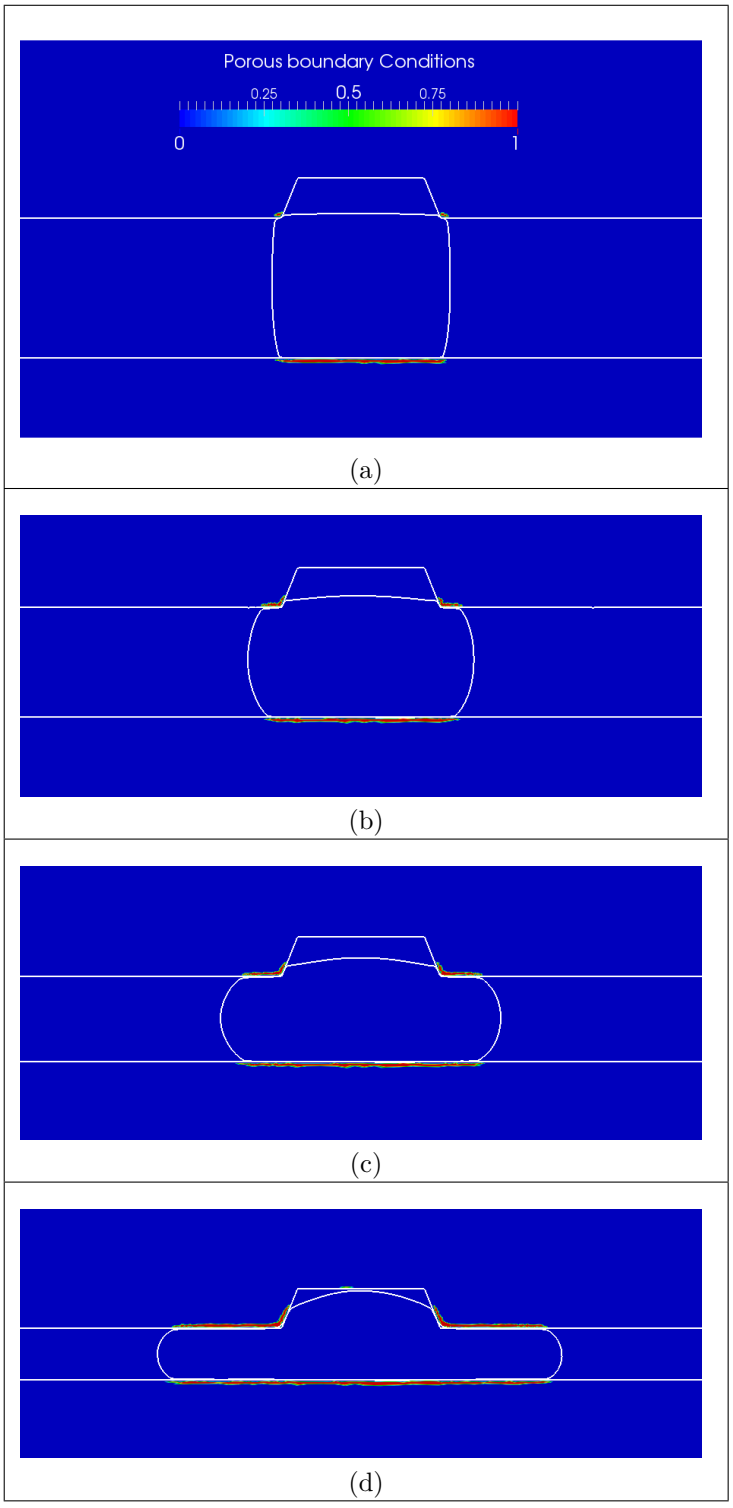


Figure 2.25: *The evolution of the porous boundary condition in time.*

Figure 2.25 shows that the set Γ_p evolves in time. The evolution is expected since the contact zone changes as well. This porous boundary condition shows a great capacity to adapt with the changing geometry of the deformable body.

To complete this discussion, the evolution of the deformable body is followed in Figure 2.26. The new boundary condition has proven to be a good fit to our problem. The air escapes allowing the body to deform properly filling the tool porosity.

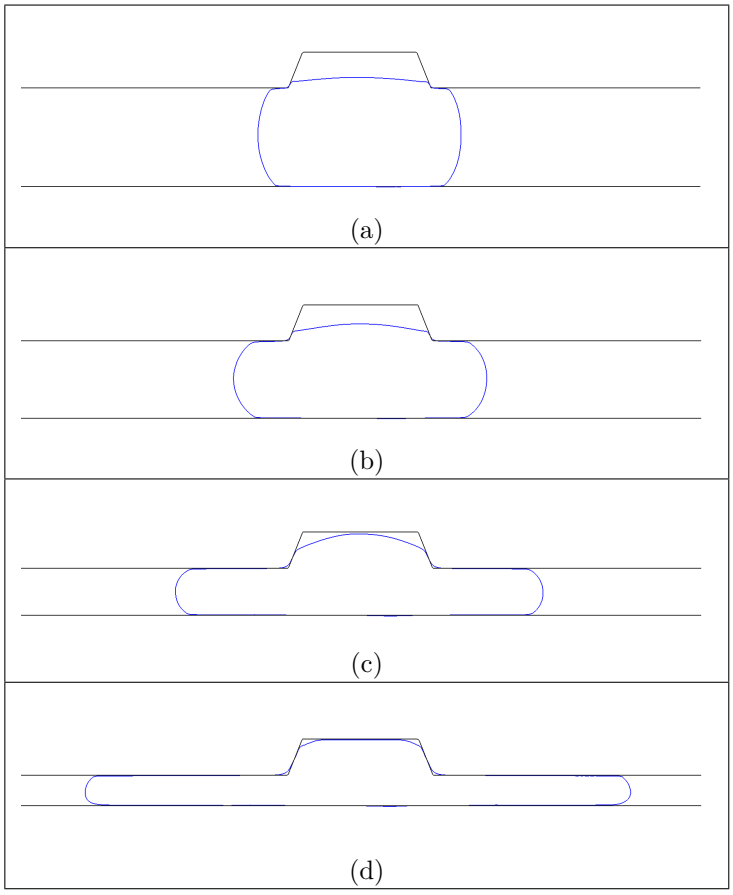


Figure 2.26: *The evolution of the zero iso-value of the deformable body when using the porous boundary condition.*

2.8 Conclusion

The present chapter is divided into two main parts. The first details the used Eulerian monolithic approach. The different tools are presented beginning by the convected level set method insuring the tracking of the interface. The mesh generator is described as well resulting in a single anisotropic mesh all over the global domain guaranteeing better precision. Since one mesh is applied per computational domain, the different interacting bodies are represented as heterogeneities via mixture laws. This part is ended by the mechanical problem coupling the Navier-Stokes equations with the level set convection

equations.

The second part is mostly dedicated to our contribution and improvements to this existent approach. A new quadratic mixture law is introduced. It plays a key role in large deformation problems exhibiting additional phases like the presence of a lubricant for instance. Several technical ameliorations are proposed also such as i) the choice of the mixture laws weight, ii) the choice of interpolation (P0 vs. P1), iii) the choice of an adaptive time step and iv) a new choice of the mesh size. These enhancements are proven to be of a great importance when it comes to stabilization and volume conservation. A last addition deals with the way to treat the existent air in our domain. The method by default traps the air inside. We offer the option to let it clear out using a new manner to impose boundary conditions. It is equivalent to creating vents in the tools for instance. The air trapping condition remains available if one wants to follow bubbles formation in the domain.

To conclude, the existent approach needed several additions to be adapted to large deformation problems involving multiple interacting bodies. Moreover, a great care is needed to choose the optimal parameters since it shows a high sensitivity. Combined with the new improvements, it shows a great capacity to respond to our needs. From now and unless specified otherwise, all ameliorations proposed in this chapter will be used for every application to ensure the optimal results.

Note that several development remains necessary. Even though the contact detection is managed automatically, the contact resolution is not taken into account. Sticking friction and sliding friction should be studied likewise. The latter developments will be treated in details in chapter 5.

Since one of the objectives is to work in a highly scalable parallel environment, an exhaustive study is presented in the next chapter. It is entirely dedicated for the parallelism of the CIMLib library where the performance of this approach is inspect closely.

Résumé en Français

Le présent chapitre est divisé en deux parties principales. La première détaille l'approche Eulerienne monolithique utilisée. Les différents outils sont présentés commençant par la méthode de convection de Level Set (assurant le suivi de l'interface). La génération d'un seul maillage anisotrope défini sur le domaine de calcul garantissant une meilleure précision est décrite aussi. Puisqu'un maillage est appliqué sur tout le domaine, les différentes pièces interagissant sont représentées comme des hétérogénéités par les lois de mélange. Cette partie prend fin avec la description du problème global couplant les équations de Navier-Stokes avec les équations de convection de la Level Set. La deuxième partie est principalement dédiée à nos contributions et améliorations proposées sur cette approche existante. Une nouvelle loi de mélange quadratique est introduite. Elle joue un rôle clé dans les problèmes de grandes déformations présentant des phases supplémentaires, comme la présence d'un lubrifiant par exemple. Plusieurs améliorations techniques sont proposées également telles que i) le choix du poids des lois de mélange, ii) le choix de l'interpolation (P0 vs P1), iii) le choix d'un pas de temps adaptatif et iv) un nouveau choix de la taille de maille. Ces améliorations sont prouvées être d'une grande importance en ce qui concerne la stabilisation et la conservation du volume. Un dernier ajout traite la modélisation de l'air dans notre domaine de calcul. La méthode par défaut emprisonne l'air à l'intérieur. Nous offrons la possibilité de le laisser évacuer le domaine à l'aide de nouvelles conditions aux limites. C'est équivalent à la création des évents dans les outils par exemple. La condition de piégeage de l'air reste disponible si l'on veut suivre la formation de bulles au cours du temps. Pour conclure, l'approche existante avait besoin de plusieurs ajouts pour être adaptées aux problèmes de grandes déformations. En outre, un grand soin est nécessaire pour choisir les paramètres optimaux. Combinée avec les nouvelles améliorations, l'approche montre une grande capacité à répondre à nos besoins. Notez que plusieurs développements restent nécessaires. Bien que la détection de contact soit gérée automatiquement, la résolution de contact n'est pas prise en compte. La modélisation du frottement glissant et collant doivent être étudiés de même. Ces derniers développements seront traités en détails dans le chapitre 5. Puisque l'un des objectifs est de travailler dans un environnement massivement parallèle une étude exhaustive est présentée dans le chapitre suivant. Il est entièrement dédié au parallélisme de la bibliothèque CimLib où la performance de cette approche est inspectée de près.

Bibliography

[Basset, 2006] Basset, O. (2006). *Numerical simulation of multi-fluid flows on a computational grid*. PhD thesis, Ecole Nationale Supérieure des Mines de Paris.

[Bonet and Peraire, 1991] Bonet, J. and Peraire, J. (1991). An alternating digital tree (adt) algorithm for 3d geometric searching and intersection problems. *International Journal for Numerical Methods in Engineering*, 31:1–17.

[Brooks and Hughes, 1982] Brooks, A. N. and Hughes, T. J. (1982). Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32(1-3):199 – 259.

[Codina, 2002] Codina, R. (2002). Stabilized finite element approximation of transient incompressible flows using orthogonal subscales. *Computer Methods in Applied Mechanics and Engineering*, 191(39-40):4295 – 4321.

[Codina and Principe, 2007] Codina, R. and Principe, J. (2007). Dynamic subscales in the finite element approximation of thermally coupled incompressible flows. *International Journal for Numerical Methods in Fluids*, 54(6-8):707–730.

[Codina et al., 2007] Codina, R., Principe, J., Guasch, O., and Badia, S. (2007). Time dependent subscales in the stabilized finite element approximation of incompressible flow problems. *Computer Methods in Applied Mechanics and Engineering*, 196(21-24):2413 – 2430.

[Coupez, 2000] Coupez, T. (2000). Génération de maillage et adaptation de maillage par optimisation locale. *Revue européenne des éléments finis*, 9:403–423.

[Coupez, 2006] Coupez, T. (2006). Rénitialisation convective et locale des fonctions level set pour le mouvement de surfaces et d'interfaces. In *Journées Activités Universitaires de Mécanique*.

[Coupez, 2011] Coupez, T. (2011). Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing. *Journal of Computational Physics*, 230(7):2391 – 2405.

[Coupez and Hachem, 2013] Coupez, T. and Hachem, E. (2013). Solution of high-reynolds incompressible flow with stabilized finite element and adaptive anisotropic meshing. *Comput. Methods Appl. for Mech. Engrg.*, 267:65–85.

[Coupez et al., 1991] Coupez, T., Soyris, N., and Chenot, J.-L. (1991). 3-d finite element modelling of the forging process with automatic remeshing. *Journal of Materials Processing Technology*, 27(1-3):119 – 133.

[Dahmen et al., 2011] Dahmen, W., Gotzen, T., Müller, S., and Schäfer, R. (2011). *Adaptive Multiresolution Finite Volume Discretization of the Variational Multiscale Method: General Framework*. Citeseer.

[Feghali et al., 2010] Feghali, S., Hachem, E., and Coupez, T. (june 2010). Stable/stabilized mixed formulation for fluid-structure interaction: theory and application. In *GDR Interaction Fluide Structure, CNRS 2902, UTC CompiÃšgne*.

[Franca and Oliveira, 2003] Franca, L. P. and Oliveira, S. P. (2003). Pressure bubbles stabilization features in the stokes problem. *Computer Methods in Applied Mechanics and Engineering*, 192(16-18):1929 – 1937.

[Frey and George, 1999] Frey, P. and George, P.-L. (1999). *Maillages - applications aux elements finis*. Hermes.

[Gruau and Coupez, 2005] Gruau, C. and Coupez, T. (2005). 3d tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and anisotropic mesh generation with adaptation to natural and multidomain metric. *Computer Methods in Applied Mechanics and Engineering*, 194:4951–4976.

[Hachem, 2009] Hachem, E. (2009). *Stabilized finite element method for heat transfer and turbulent flows inside industrial furnaces*. PhD thesis, Ecole Nationale Supérieure des Mines de Paris.

[Hachem et al., 2010] Hachem, E., Rivaux, B., Kloczko, T., Digonnet, H., and Coupez, T. (2010). Stabilized finite element method for incompressible flows with high reynolds number. *Journal of Computarional Physics.*, 229(23):8643–8665.

[Harari and Hughes, 1992] Harari, I. and Hughes, T. J. (1992). What are c and h?: Inequalities for the analysis and design of finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 97(2):157 – 192.

[Hirt and Nichols, 1981] Hirt, C. and Nichols, B. (1981). Volume of fluid (vof) method for the dynamics of free boundaries. *Journal Of Computational Physics*, 39:201–225.

[Hughes et al., 1998] Hughes, T. J., Feijóo, G. R., Mazzei, L., and Quincy, J.-B. (1998). The variational multiscale method—a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 166(1-2):3 – 24. Advances in Stabilized Methods in Computational Mechanics.

[Lo, 1985] Lo, S. (1985). A new mesh generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering*, 21:1403–1426.

[Loseille, 2008] Loseille, A. (2008). *Adaptation de maillage anisotrope 3D multi-échelles et ciblÃ©e à une fonctionnelle pour la mécanique des fluides. Application à la prédiction haute-fidélité du bang sonique*. PhD thesis, Université Pierre et Marie Curie - Paris VI.

[Medic and Mohammadi, 1999] Medic, G. and Mohammadi, B. (1999). an incompressible navier-stokes solver for unstructured meshes. Technical report, INRIA Research Report 3644.

[Osher and Sethian, 1988] Osher, S. and Sethian, J. (1988). Fronts propagating with curvature dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49.

[Peng et al., 1999] Peng, D., Merriman, B., Osher, S., Zhao, H., and kang, M. (1999). A pde-based fast local level set method. *Journal of Computational Physics*, 155:410.

[Principe and Codina, 2010] Principe, J. and Codina, R. (2010). On the stabilization parameter in the subgrid scale approximation of scalar convection-diffusion-reaction equations on distorted meshes. *Computer Methods in Applied Mechanics and Engineering*, 199(21-22):1386 – 1402. Multiscale Models and Mathematical Aspects in Solid and Fluid Mechanics.

[Shephard and Georges, 1991] Shephard and Georges (1991). Autommesh three-dimensional mesh generation by the finite octree technique. *International journal of numerical methods in engenering*, 32:709–740.

[Shephard, 1988] Shephard, M. (1988). Approaches to the automatic generation anf control of finite element meshes. *Applied Mechanics Reviewa*, 41:169–185.

[Sussman et al., 1994] Sussman, M., Smereka, P., and Osher, S. (1994). A level set approche fort computing solutions to incompressible two phase flow. *Journal of Computational Physics*, 114:146–159.

[Tchon et al., 2003] Tchon, K.-F., Khachan, M., Guibault, F., and Camarero, R. (2003). Constructing anisotropic geometric metrics using octrees and skeletons. In *12th International Meshing Roundtable, Santa Fe,*, pages 293–304.

[Tezduyar and Osawa, 2000a] Tezduyar, T. and Osawa, Y. (2000a). Finite element stabilization parameters computed from element matrices and vectors. *Computer Methods in Applied Mechanics and Engineering*, 190(3-4):411–430.

[Ville et al., 2011] Ville, L., Silva, L., and Coupez, T. (2011). Convected level set method for the numerical simulation of fluid buckling. *International Journal for Numerical Methods in Fluids*, 66(3):324–344.

CHAPTER 3

Parallel Computing

Contents

3.1 Introduction 91

3.2 Concepts and Terminology 91

3.2.1 Flynn's Classical Taxonomy 92

3.2.2 Memory classification and access 93

3.3 Performance of a Parallel Code 95

3.3.1 Hardware performance 95

3.3.2 Software performance 96

3.3.3 Limits and Costs of Parallel Programming 97

3.3.3.1 Serial fraction 97

3.3.3.1.1 Amdahl's law 98

3.3.3.1.2 Gustafson law 98

3.3.3.2 Communications between cores 99

3.3.3.3 Load balancing 100

3.4 Parallel computing in CIMLib 100

3.4.1 S.P.M.D parallelization of the Navier-Stokes Solver 101

3.4.2 Dynamic load balancing 103

3.5 Applications 104

3.5.1 Case presentation 104
3.5.2 Navier-Stokes vs Stokes 106
3.5.3 Coarse mesh: MPI Version and Bind to Core Option 107
3.5.4 Coarse mesh: Hardware limitation and Round Robin Option . 111
3.5.5 Refined mesh 113
3.5.6 Anisotropic mesh: Adaptation Scalability 115
3.6 Conclusion 118
Résumé en Français 119
Bibliography 120

3.1 Introduction

The last few years has experienced a significant progress in computer technology. This progress has influenced mostly the processor power. Modern processors are formed by multi-cores. Hence, the execution of multiple instructions is available.

Simultaneously, the complexity of mathematical models has increased as well. These models can describe complex physical problem such as weather forecast, aerospace applications, crash test vehicles, material forming and much more applications. The resulting high computational demands inspired scientific programmers to introduce parallel computing concept.

The use of parallelism in scientific computing reduces considerably computational time when comparing it to the sequential time. Thus, handling larger scale physical problems are now affordable. For instance, the number of freedom can be increased to obtain a more accurate solution without a drastic increase of the computing time.

To be beneficial, parallelizing algorithms has to be treated with a great care. In fact, the efficiency of parallelism depends highly on the computer architecture and other software aspects such as the load balancing.

This chapter provides a quick overview of the parallelism in general and focuses on CIMLib's parallel environment in particular. We begin by explaining some important concepts and terminologies to simplify the topic for non-initiative readers. For more details, the readers are invited to consult [Magoulès and Roux, 2013], [Eijkhout et al., 2014] and [Rauber and Rünger, 2010]. Next, a brief guide on how to judge a parallel performance is presented highlighting hardware and software limitations. Then, the parallelism in CIMLib is described restricting to solvers and dynamic load balancing (consuming the bigger part of the computing time). Finally, the efficiency of the parallel library is inspected via numerical applications followed by a conclusive discussion.

3.2 Concepts and Terminology

A first step into understanding parallel programming begins by introducing some concepts and terminologies. Even though, parallel algorithms play a key role to obtain a high performance computing application, it should be adapted to our computer/cluster architecture. Since many different possibilities of algorithms decomposition exist for the same application, choices should be made carefully. For computer architectures, several classifications exists. Herein, we present in sec. 3.2.1 the most popular taxonomy defined in [Flynn, 1972]. Flynn's classification scheme is based on the notion of a stream of information and instructions. Memory is another important factor affecting performance. Not only the memory size but also the speed in which the information is accessible impact

greatly the [Eijkhout et al., 2014]

3.2.1 Flynn's Classical Taxonomy

1. **Single-Instruction, Single-Data (SISD)** : It uses one processing element to load a single instruction and access the corresponding data. Then this processing element executes the instruction and stores back its result into the data storage.

2. **Multiple-Instruction, Single-Data (MISD)** : It uses multiprocessing element to execute multiple instructions accessing the same data storage. Each processing element has a private memory but shares one common access to the global memory. More precisely, each processing element loads a different instruction from its private memory and executes it using the data received from the global memory storage.

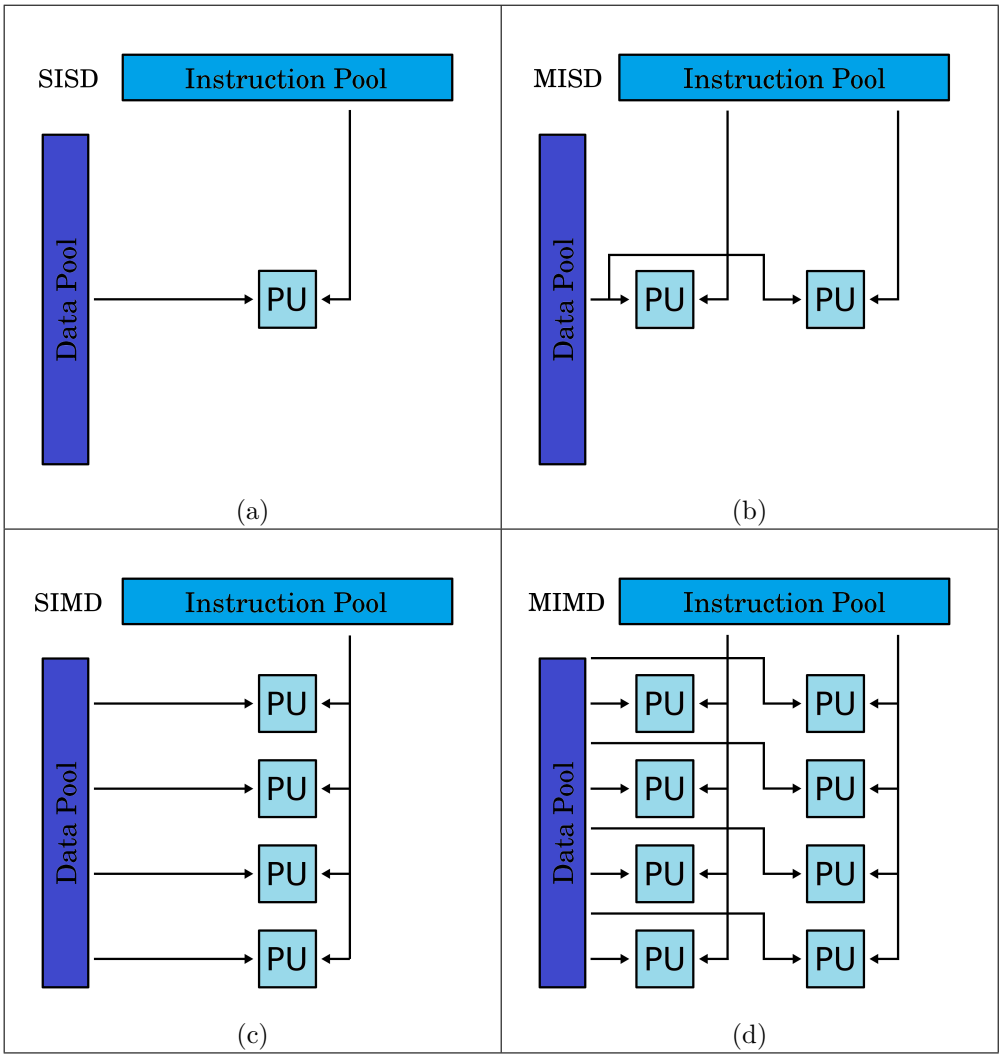


Figure 3.1: *The Flynn classification of computer architectures: (a) SISD, (b) MISD, (c) SIMD and (d)MIMD.*

3. **Single-Instruction, Multiple-Data (SIMD)** : It uses multiprocessing element in order to execute one single instruction using multiple data storage. In fact, each processing element obtains the same instruction to execute, but loads a different and separate data. This type of architecture is compatible with a shared or distributed memory.

4. **Multiple-Instruction, Multiple-Data (MIMD)** : It uses multiprocessing element capable of executing multiple instruction for separate data memory units. Each processing element loads a different instruction, executes it to a different data memory and stores the result back into the corresponding storage.

To clarify the previous notations, one must mention the following points:

- A typical example of the SISD is the conventional sequential computer.
- Using the MISD represents a big challenge due to its nature executing different instructions on a single data memory. Thus it can be very restrictive and tricky.
- The SIMD and MIMD are the most used approaches. For instance, the SIMD is widely spread in multimedia/games applications especially to generate 3d prospects. While the MIMD is mostly used in multiprocessors systems or clusters. Between the SIMD and MIMD, it should be mentioned that the parallel algorithm for the former is considerably easier to implement.

3.2.2 Memory classification and access

As for the memory, the conventional classification consists in dividing it into two major types: a shared memory (Figure 3.2a) and a distributed memory (Figure 3.2b).

- For a shared memory system (called also global memory system), the memory is considered as a pool in which all the problem data are stored. All processors access the data via this global memory. In term of parallel computing, a shared memory machine is associated with threads based on a Compiler directives (OpenMP, TBB)

- Distributed memory machines contain a number of processing elements called nodes and interconnection network. A node is formed by a processor and a local memory. Each processing element has its own local memory. The latter is private and can be accessed by one and only one processing element. The interconnections between processing elements -not sharing memory- are ensured via a network. In term of parallel computing, a distributed memory machine is associated with a process using a message passing interface to communicate between nodes (MPI or Parallel Virtual Machine). Note that a third category can be found as well: a hybrid memory combining shared and distributed memory in one configuration (see Figure 3.2c).

Another important factor affecting a simulation computing time is the time needed to access the memory. In a shared memory machine, the access time to the data storage depends highly on CPU characteristics in a NUMA systems (No Uniform Memory Access). Further details dealing with the memory management can be found in [Hager and Wellein, 2010]. Nowadays all processors has a cache memory. A cache is a small memory formed at most by three levels : L1, L2 and L3 (Figure 3.3). The L2 cache is the most important one. Actually, when requesting data, the processor will check $L1$, followed by $L2$ and $L3$. If the requested data is found in one of these levels, a cache hit is obtained, if not (i.e. the data is found in the memory) a cache miss is obtained. The more a cache hits faster the simulation will go. Thus the choices made can affect significantly the computing time.

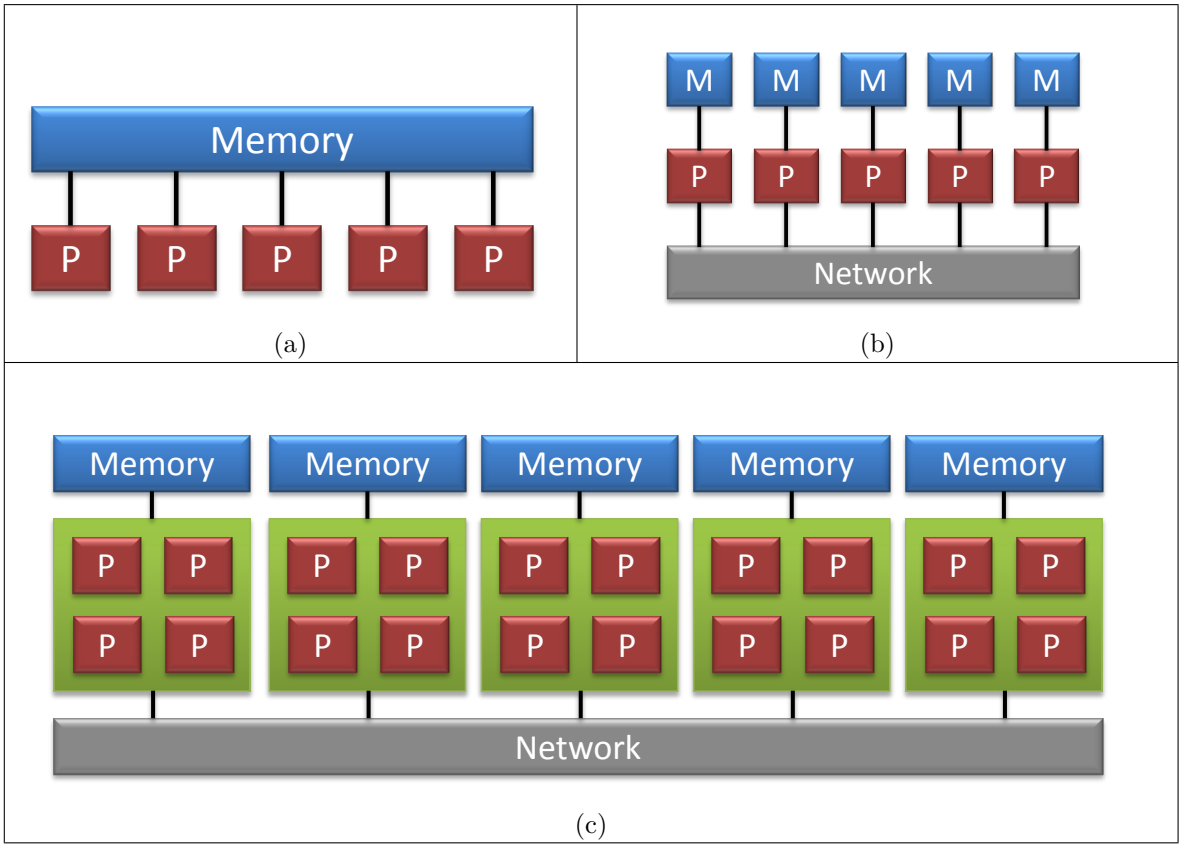


Figure 3.2: *The memory classification of parallel computers: (a) shared memory, (b) distributed memory and (c) hybrid memory.*

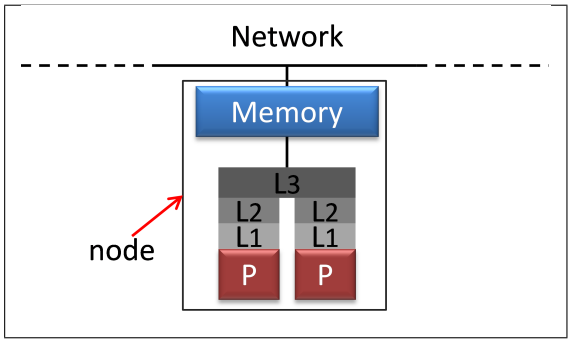


Figure 3.3: *Illustration of a node containing bi-processors with three level of cache memory.*

3.3 Performance of a Parallel Code

3.3.1 Hardware performance

Processors are the most important components in a computer. They affect significantly the hardware performance. Since 2005, manufacturers like *Intel* and *AMD* deliver processors containing two or more cores (multicore processors). The processor performance is measured using several parameters. According to *[www.top500.org,]*, the most important ones are :

- $\#Proc$: Number of processors (Cores)
- R_{peak}: Theoretical peak performance provided by the manufacturer or the seller. This performance indicates the number of instructions treated by all cores per second (FLOPS) .
- R_{max}: The maximum value of R_{peak} which cannot be exceeded by the cores. The real performance of the cores is given by R_{max}. It is measured by the LINPACK Benchmark. The LINPACK Benchmark was introduced by *Jack Dongarra*. It consists to solve a dense system of Linear equations. A detailed description is available in [Dongarray et al., 2001]. A parallel implementation of the Linpack benchmark can be found in [Petitet et al., 2008] .
- N_{max}: Problem size for achieving R_{max}
- $N_{1/2}$: Problem size for achieving half of R_{max}

Figure 3.4 shows the top 5 of the super Calculators in June 2013 *[www.top500.org,]* . At Cemef, the Cluster contains 1 480 cores distributed in 150 nodes. The nodes are divided as : 48×4 cores, 40×8 cores and 62×16 cores.

NAME	SPECS	SITE	COUNTRY	CORES	R_{MAX} PFLOP/S	POWER MW
Tianhe-2 (Milkyway-2)	NUDT, Intel Ivy Bridge (12C, 2.2 GHz) & Xeon Phi (57C, 1.1 GHz), Custom interconnect	NUDT	China	3,120,000	**33.9**	17.8
Titan	Cray XK7, Opteron 6274 (16C, 2.2 GHz) + Nvidia Kepler (14C, .732 GHz), Custom interconnect	DOE/SC/ORNL	USA	560,640	**17.6**	8.3
Sequoia	IBM BlueGene/Q, Power BQC (16C, 1.60 GHz), Custom interconnect	DOE/NNSA/LLNL	USA	1,572,864	**17.2**	7.9
K computer	Fujitsu SPARC64 VIIIfx (8C, 2.0GHz), Custom interconnect	RIKEN AICS	Japan	705,024	**10.5**	12.7
Mira	IBM BlueGene/Q, Power BQC (16C, 1.60 GHz), Custom interconnect	DOE/SC/ANL	USA	786,432	**8.16**	3.95

Figure 3.4: *Top five super computers june 2013. [www.top500.org,]*

3.3.2 Software performance

Despite all progress on recent multiprocessors and their performance, parallel algorithms remain essential to obtain an efficient parallel code. The performance is judged mainly by analyzing three conventional parameters: the execution time, the speed up and the efficiency. The execution time of any code is a primary concept in measuring the parallel implementation efficiency. Consider a size N problem running on p cores, the execution time adopts the following notation:

$$T_p = t(N, p) \tag{3.1}$$

The main criterion to evaluate a parallel algorithm is "Speed up". It refers to how much a parallel algorithm is faster than a sequential one; it is defined by the following formula:

$$S_p = \frac{T_1}{T_p} \tag{3.2}$$

where p is the number of used cores, T_1 is the execution time of the sequential algorithm and T_p is the execution time of the parallel algorithm using p cores.

Theoretically, the linear or ideal Speed up is obtained if $S_p = p$, i.e. when executing an algorithm using p cores, the parallel computing time should be equal to the sequential time divided by p. Speed up could be classified into three categories: sub-linear, linear, super-linear (see Figure 3.5). Even though we aim for a linear speed up, the commonly reached is a sub-linear one. Nevertheless sometimes if the problem is small enough to be contained fully in the cache memory, a super linear speed up is obtained. In this case, the simulation runs the fastest.

The efficiency of a parallel algorithm is given by the ratio:

$$E = \frac{S_p}{p} \tag{3.3}$$

Equation (3.3) will precise the scalability of the parallel algorithm: if the speed up is linear ($S_p = p$), the efficiency is equal to 1 i.e. algorithm is as efficient running on 10

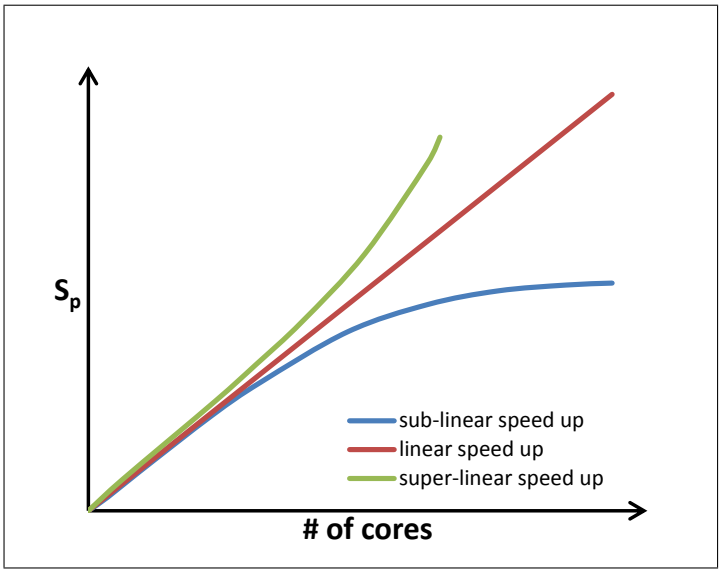


Figure 3.5: *Different types of speed up when running a parallel simulation : the common speed up (blue); the linear speed up (red); the super-linear speed up (green)*

cores as on 1000 and does not require any further improvements. We should note that T_p, the execution time of the parallel algorithm, plays a key-role impacting the algorithm efficiency.

3.3.3 Limits and Costs of Parallel Programming

Summarizing what has been discussed previously, the ideal speed up and efficiency verify:

$$S_p = p \quad and \quad E = 1 \tag{3.4}$$

In practice, it can not be easily reached. Many reasons affect the scalability of the code (such as the serial fraction, inter-cores communication and the load balancing) and will be presented next.

3.3.3.1 Serial fraction

In practice, several parts of the code cannot be parallelized and remain sequential. These algorithms are called the serial fraction of the code. [Amdahl, 1967] and [Gustafson, 1988] laws provide a theoretical approximation of speed up in codes containing a serial fraction.

Consider a problem of size N . The execution time needed to solve the problem using one core is noted by $t(N, 1)$. The algorithm contains both sequential and parallel parts. Thus, the execution time is divided into two parts as well: $t(N, 1) = t_{seq} + t_{par}$

where t_{seq} and t_{par} represent respectively the execution time of the sequential and parallel algorithms.

The serial fraction is defined as follows:

$$f_{seq} = \frac{t_{seq}}{t_{seq} + t_{par}} = \frac{t_{seq}}{t(N,1)} \tag{3.5}$$

3.3.3.1.1 Amdahl's law

This law evaluates the computational time of a fixed size problem according to the number of cores p and the serial fraction f_{seq} [Amdahl, 1967]:

$$t(N,p) = t_{seq} + \frac{t_{par}}{p} \tag{3.6}$$

Using (3.6) , Amdahl's law offers an approximation of the maximum speed-up that can be reached by the code:

$$\begin{aligned} S_{p(max)} &= \frac{t(N,1)}{t(N,p)} = \frac{t_{seq} + t_{par}}{t_{seq} + \frac{t_{par}}{p}} \\ &\leq \frac{t_{seq} + t_{par}}{t_{seq}} = \frac{1}{f_{seq}} \end{aligned} \tag{3.7}$$

Equation (3.7) confirms that the presence of a serial fraction affects the scalability of a parallelized code. The speed-up is limited by a threshold inversely proportional to f_{seq} . If one aims to improve the code scalability (i.e. speed up closer to p), the serial fraction must be decreased as much as possible (i.e. replacing sequential algorithms by parallel ones) .

3.3.3.1.2 Gustafson law

The Gustafson law consists on evaluating the speed up for a particular problems where the problem size varies linearly with the number of cores p (N is replaced by pN for $p \geq 1$) [Gustafson, 1988]. The Amdahl law is replaced by the Gustafson law in which the the parallel execution time is taken as the reference :

$$t(pN,p) = t_{seq} + t_{par} \tag{3.8}$$

If t_{par} is the execution time on p processors (size problem equals pN), the sequential processing time (size problem equals N) is :

$$t(N,1) = t_{seq} + pt_{par} \tag{3.9}$$

Then the corresponding Speed up is :

$$S_p = \frac{t_{seq} + pt_{par}}{t_{seq} + t_{par}} = \frac{t_{seq}}{t_{seq} + t_{par}} + p\frac{t_{par}}{t_{seq} + t_{par}} = f_{seq} + p(1 - f_{seq}) \tag{3.10}$$

Gustafson's law offers a linear speed, a better estimation of the one proposed by Amdahl. The main difference is that the total execution time is now dependent of p.

With these kind of problems, the smartest step is to simulate the largest cases possible. In other word, if the data size increases the sequential part decreases relatively. According to equation (3.10) , the speed up increases with the number of cores and tends towards it (if f_{seq} is small enough then $S_p \to p$).

3.3.3.2 Communications between cores

Inter-cores communication limitations are confronted when using a distributed memory machines: the communication between cores is required and the data exchange is done via an interconnection network. These operation lead to a loss of efficiency mainly related to:

1. Task synchronization: It is compulsory to synchronize the task management when the simulation runs using multiprocessors. If a processor needs information residing in the memory of another processor, it is necessary that the latter is available for communication. If the processor requesting information must wait, we are missusing these resources leading to loss of efficiency and computing time. Thus, it is necessary to have a good coordination between processors to insure the best synchronized communication possible.

2. Transfer time: Another important reason affecting the algorithm scalability is the time needed to transmit information from one node to another. The cost is assessed as follows:

$$t_{comm} = t_{latency} + t_{pack} + \frac{N}{v_{transfert}} + (d-1)t_{transit} \tag{3.11}$$

where :

- $t_{latency}$ represents the time needed to establish a connection. It is equivalent to the time needed for sending a zero length message.
- t_{pack}: represents the time needed to load the information into a buffer.
- $v_{transfert}$: represents the speed of data transfer in the interconnection network.
- t_{tansit}: represents the time needed to transfer a message via a network if using distributed memory machines. The time of transfer, using a network of d internodes connections, is assessed equal to $(d-1)t_{transit}$. For instance, in a shared

memory machine, using a UMA, d is equal to 1 i.e. one direct connection. Thus, no transit time what so ever is needed.

Notice that the speed of data transfer depends highly on the hardware (used interconnection) as well as on the message transfer protocol. In order to minimize the communication time between nodes, it is recommended to send one long message rather than several short ones.

3.3.3.3 Load balancing

In scientific computing, load balancing algorithms are used to distribute the amount of work (i.e. the workload) as equally as possible on the number of cores in a parallel environment. These algorithms can be roughly divided into two different categories : static and dynamic ones. The static algorithms distribute the load evenly between cores at the beginning of the simulation. Such algorithms can affect the scalability since the current state of the simulation is not taken into account. Thus, a dynamic load balancing algorithm is recommended to fix this limitation. For instance, a finite element problem using a remeshing or mesh adaptation technique will present eventually a load imbalance. Their use increases locally the number of nodes on a particular core leading to an imbalance in distributing the workloads. An other example of imbalance workload occurs in contact search algorithms in Lagrangian approaches (see chapter 1 for more details). An introduction and discussion related to this topic can be found in [Hendrickson and Devine, 2000]. An overview on load balancing algorithms in parallel environment can be found in [Jimack, 1999].

For dynamic algorithms, a literature survey splits the existent methods into two major families. The first consists on partitioning the domain into sub-domains recursively using a geometric or topological criteria [Williams, 1991b] and [Van Driessche and Roose, 1995]. These methods are known as the direct methods. The others, called iterative, uses a diffusion algorithm. They start from an initial partition and exchange nodes between neighbor cores in order to improve the balance of the workload. The dynamic load balancing used in CIMLib, implemented by [Digonnet, 2001], will be described in sec. 3.4.2

3.4 Parallel computing in CIMLib

Though CIMLib is a fully parallelized library, we only highlight two important parts: the mechanical solver and the Dynamic load balancing. The first represents a large part of the simulation time. Whereas the second is considered as a crucial component to insure the work equilibrium since we use a mesh adaptation technique. Therefore their

parallelization has a great importance.

3.4.1 S.P.M.D parallelization of the Navier-Stokes Solver

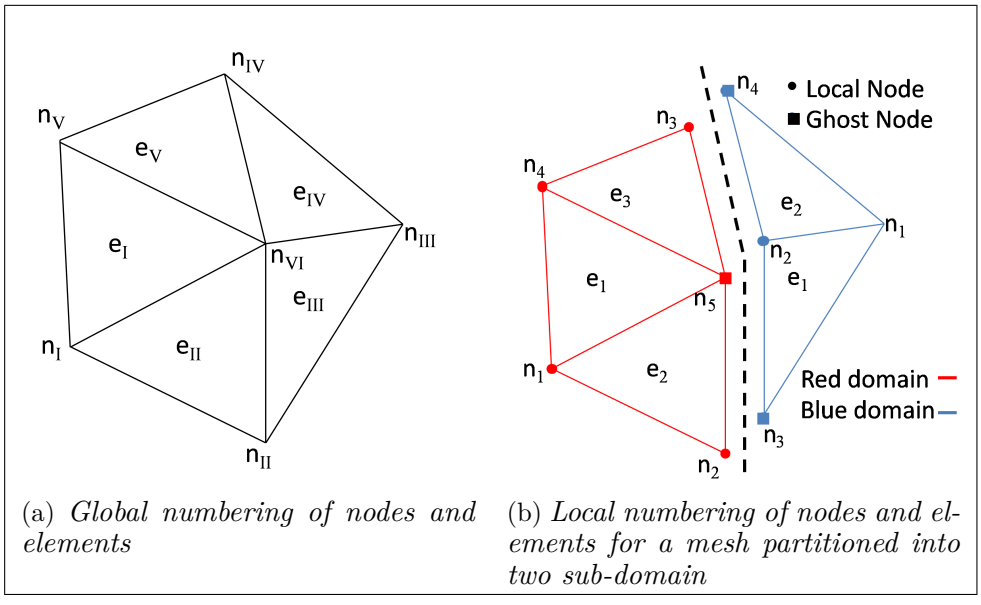
Let $\mathcal{M} = (\mathcal{N}, \mathcal{T})$ be the mesh discretizing Ω where $\mathcal{N}$ and $\mathcal{T}$ are respectively the set of nodes and simplexes forming the mesh. Note $\mathcal{P}$ the set of cores used for running the simulation. If $\mathcal{M}_i$ is a partition of $\mathcal{M}$ then it can be expressed as follows:

$$\mathcal{M} = \bigcup_{i=1}^{card(\mathcal{P})} \mathcal{M}_i \tag{3.12}$$

Every sub-mesh $\mathcal{M}_i = (\mathcal{N}_i, \mathcal{T}_i)$ is associated to a particular core p_i; where $\mathcal{T}_i$ is the set of simplexes assigned to a core p_i and $\mathcal{N}_i$ is the set of nodes forming those simplexes. Note that a simplex can only belong to one and only one sub-domain. Nevertheless the set of nodes $\mathcal{N}_i$ can be shared between several cores. The shared nodes form an interface between the different cores :

$$\Gamma_{ij} = \{\mathcal{M}_i \cap \mathcal{M}_j / p_i \neq p_j\} \tag{3.13}$$

As illustrated in (Figure 3.6), the interface nodes are not replicated. It means that if two cores share the same nodes, they are stocked on one core and ghost nodes are created and stocked on the other one.



(a) *Global numbering of nodes and elements*

(b) *Local numbering of nodes and elements for a mesh partitioned into two sub-domain*

Figure 3.6: *Example of partitioning a global mesh in two sub-meshes*

Once the partition is created, each core p_i reads the data related to the associated sub-mesh. Note that for now, no communications between cores is needed. Parallelizing

the solver is summarized by two steps: The assembly of the stiffness matrix and the iterative resolution of the problem:

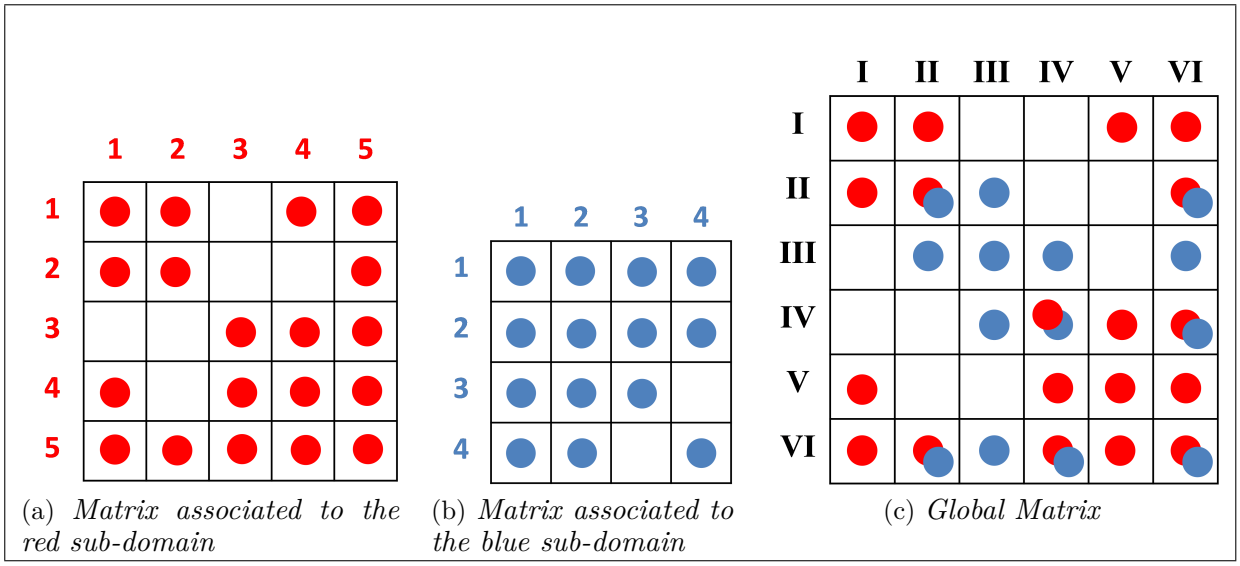
$$\mathbf{A}x = \mathbf{b} \tag{3.14}$$

where $\mathbf{A}$ is the global stiffness matrix, x and $\mathbf{b}$ are two global vectors (i.e. defined on the whole domain Ω).

To determine the global matrix $\mathbf{A}$, each core p_i must first assemble its own local matrix $\mathbf{A}_i$. We should mention that the dimension of each $\mathbf{A}_i$ varies between cores depending on the number of nodes in the sub-mesh $\mathcal{M}_i$. The local rigidity matrix is assembled on each core. Then, the global stiffness matrix $\mathbf{A}$ is written as follows:

$$\mathbf{A} = \sum_{i=1}^{card(\mathcal{P})} \hat{\mathbf{A}}_i \tag{3.15}$$

where $\hat{\mathbf{A}}_i$ is the extension of the local matrix $\mathbf{A}_i$. This extension is determined by resizing the matrix $\mathbf{A}_i$ to fit the global dimension of $\mathbf{A}$. It is accomplished by adding zeros for nodes belonging to only one core respecting the relations between the local and global positions of the variable nodes via a binding matrix (see Figure 3.6 and Figure 3.7).



(a) *Matrix associated to the red sub-domain* (b) *Matrix associated to the blue sub-domain* (c) *Global Matrix*

Figure 3.7: *Local matrix assembly of the two sub-domains: red and blue (a-b) and their projection into the global matrix (c)*

Now that the stiffness matrix $\mathbf{A}$ is determined, equation (3.14) can be solved. CIM-Lib uses the Portable Extension Toolkit for Scientific Computation (PETSc) [Balay et al., 1997], [Balay et al., 2014a] and [Balay et al., 2014b] for an iterative resolution method. Such methods are based on a matrix-vector product:

$$y = \mathbf{A}r \tag{3.16}$$

y and r are two global vectors defined on the whole Ω.

Let us define a local vector $\bar{r}_p$ representing the restriction of r on the core p. So the product matrix-vector is written as follows:

$$y = \mathbf{A}r = \left(\sum_{p\in\mathcal{P}} \hat{\mathbf{A}}_p\right) r = \sum_{p\in\mathcal{P}} \widehat{(\mathbf{A}_p \bar{r}_p)} = \sum_{p\in\mathcal{P}} \widehat{(\mathbf{A}r)_p} \tag{3.17}$$

With locally $y_p = (\mathbf{A}r)_p = \mathbf{A}_p r_p$.

Nevertheless, the product y_p does not represent the restriction of $\left(\bar{\mathbf{A}r}\right)_p$ onto the core p. It represents the contribution of the latter product to the $p-th$ core . Thus, the global vector y is given by:

$$y = \sum_p \hat{y}_p \tag{3.18}$$

where $\hat{y}_p$ represents the local vector resized as we mentioned before. So to use an iterative method we should define the restriction $\bar{y}_p$ as :

$$\bar{y}_p = y_p + \sum_{q\in\Gamma_{pq}} y_{qp} \tag{3.19}$$

3.4.2 Dynamic load balancing

In CIMLib, the dynamic load balancing is treated as a renumbering of the mesh (nodes and simplexes) under constraints [Digonnet, 2001]. In fact, the nodes/simplexes are numbered using a numbering couple (n_1, n_2) :

- $n_1 \in \mathbb{N}$ is the number of the nodes/simplexes.
- $n_2 \in \mathbb{N}$ is the number of the domain p.

The above technique does not change the definition of the mesh, it simply introduce a new numbering of its nodes and simplexes. It is obvious that the numbering is not unique. Therefore a cost function is needed to compare the different numbering sets. The cost function assessing the computational time spent on a core p can be decomposed into:

$$\phi = \underset{p\in\mathcal{P}}{max}\, t_p + \underset{p\in\mathcal{P}}{max}\, c_p \tag{3.20}$$

where t_p is the computational time required by the core p and c_p the sum off all the communication related to the core p including the synchronization time. The previous cost function uses the maximum criterion. It generates difficulties when searching for the

optimal numbering. Therefore the author introduces locally an order relation leading to optimize the cost function (i.e. minimizing the cost function globally).

In a parallel framework, the procedure above is based on an iterative algorithm coupling two processors at a time. As mentioned earlier, optimizing a partition can be achieved by optimizing the corresponding numbering. To manage this task, the author introduces three numbering operators R_{pp}, R_{pq} and I_p. The operator $R_{pp} = (r^n_{pp}, r^s_{pp})$ acts locally (on a processor p) renumbering both the nodes and simplexes. The operator $R_{pq} = (r^n_{pq}, r^s_{pq})$ acts between processors p and q renumbering the nodes and simplexes. The operator I_p is an identity operator leaving the numbering of a processor p intact. Repartitioning can be conceived as a succession of the renumbering operators R_{pp}, R_{pq}, I_p.

This renumbering strategy can be repeated as long as the re-meshing evolves in order to maintain both an optimal partitioning and cost function.

3.5 Applications

This section is dedicated to examine the performance of the monolithic Eulerian approach implemented in CIMLib. Through all this study, a $3d$ generic flat bunch simulation is used to inspect closely the parallel scalability. In fact, neither the type of the chosen application nor the complexity of the geometry should affect the obtained speed up. It should only increase the computational time of the simulation. This is true since the approach uses one mechanical solver all over the domain and each deformable body is tracked via the level set technique. In other words, the parallel scalability (i.e. speed up) is not affected regardless of the number of deformable bodies and their geometry (even if complex).

3.5.1 Case presentation

To begin with, we define the domain $\Omega = 40.5\,mm \times 33.5\,mm \times 51\,mm$ where the deformable body of dimensions $27\,mm \times 20\,mm \times 15\,mm$ is completely immersed in Ω (see Figure 3.8).

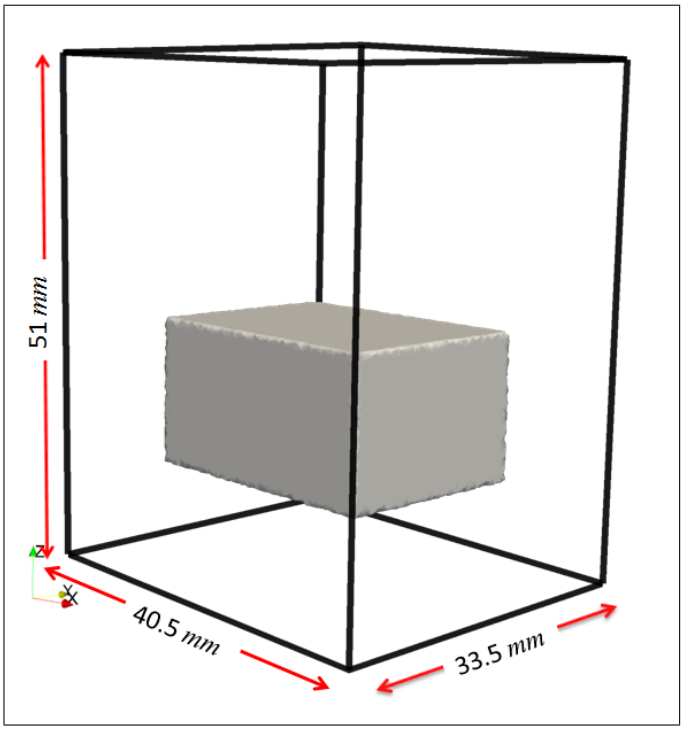


Figure 3.8: *Physical domain used to study the parallel performance of the monolithical approach.*

The work domain Ω is meshed using one of the following meshes:

- a coarse isotropic mesh containing 65 017 nodes and 377 522 elements where the mesh size is equal to 1.25 mm;
- a refined isotropic mesh containing 491 055 nodes, 2 879 257 elements and a mesh size of 0.625 mm;
- an anisotropic mesh of approximately the same size of the coarse isotropic one.

The forging process is stopped when the forged piece reaches a deformation of 10% of its initial height. Thus, it is equivalent to 32 increments on the coarse mesh using a time step of $5.10^{-3}s$ and 64 increments on the refined mesh using a time step of $25.10^{-4}s$. Table 3.1 summarizes the simulations parameters.
The simulations are launched locally on the nodes. A computing node is formed by a CPU bi-processors "*AMD Opteron™ Processor* 6134". The used processor contains 8 cores with a speed of 2.3 Ghz. Each core is characterized by a $L1$ memory of size 128 kb; $L2$ memory of size 512 kb with a speed of 2.3 Ghz and a $L3$ memory of size 12288 kb.

mesh	isotropic coarse	isotropic refined	anisotropic
number of nodes	65 017	491 055	72 767
number of elements	377 552	2 879 257	425 863
number of increments	32	64	32
mesh size	1.25	0.625	-
time step	5.10^{-3}	25.10^{-4}	-

Table 3.1: *The simulations parameters used for speed up tests*

The scalability of the parallel code is tested versus several variables. Once the results are established, the refined mesh is used to highlight its effects if any on the scalability. The anisotropic mesh is used, at the end, to inspect the adaptation scalability used in CIMLib.

3.5.2 Navier-Stokes vs Stokes

We begin by a quick comparison between two different solvers in CIMLib : i) Navier-Stokes and ii) Stokes using the isotropic coarse mesh. The comparison is done using 2^n where $n \in \{0, 1, \cdots, 5\}$ cores. Table 3.2 shows the time needed to assemble and solve both Navier-Stokes and Stokes solvers.

number of cores	Navier-Stokes		Stokes	
	Assembly	Resolution	Assembly	Resolution
1	352.9625	455.8391	177.2075	438.8229
2	184.3498	235.1301	93.0619	241.1997
4	94.4416	152.1518	48.5017	126.6885
8	47.9751	86.8955	25.0818	72.7976
16	24.5011	50.7569	13.0989	47.4226
32	12.8818	30.1226	7.3847	30.2918

Table 3.2: *Time (in s) spent to assemble, solve the mechanical problem with two different solvers (Navier-Stokes vs Stokes) using the 1.2.6 version of MPI*

In terms of resolution both solvers consume approximately the same amount of time even though they use completely different types of resolution : i) Generalized minimal residual algorithm (GMRES) for Navier-Stokes and ii) Conjugate Gradient for the Stokes solver. As for the assembly time, Stokes solver takes approximately half the time needed

by Navier-Stokes solver. This behavior is expected since less terms are in play. For sake of convenience, Stokes solver will be used through all the study presented in this chapter.

3.5.3 Coarse mesh: MPI Version and Bind to Core Option

Using the 1.2.6 version of MPI and the coarse mesh, the simulations were launched on 2^n cores where $n \in \{0, 1, \ldots, 5\}$. As mentioned in sec. 3.4.1, solvers are the ones consuming most of the computational time. For example, the Stokes solver represents 62.41% of the computational time when the simulation runs using one core. As for the convection solver, levellerT, it represents 15.5%. Thus, the time consumed by both solvers takes up to $\approx 78\%$ of the computational time.

Note that the assembly time required by the LevellerT is very comparable to the mechanical solver assembly. Even though the local stiffness matrix to be assembled (size 4×4) is smaller then Stokes (size 16×16). Thus, we recommend optimizing the levellerT assembly to improve the global computational time.

For sake of clarity, we separate the assembly time from the resolution time. Table 3.3 and Table 3.4 summarize respectively the time used by the solvers (Stokes solver and the convection solver -LevellerT-) and by other relevant algorithms of the simulation (generating output files, computing generalized strain, etc.). These numbers confirm that the same comment (for the sequential computing) is valid for parallel computing as well. Stokes and LevellerT solvers consume roughly 75% of the computational time whether in sequential or parallel simulations.

number of cores	Stokes		LevellerT		Total Computational Time
	Assembly	Resolution	Assembly	Resolution	
1	177.2075	438.8229	143.9945	9.0872	987.0089
2	93.0619	241.1997	77.2412	5.8230	544.9218
4	48.5017	126.6885	39.5058	2.4788	305.4120
8	25.0818	72.7976	19.7558	1.3304	221.6395
16	13.0989	47.4226	10.0588	0.7003	218.0213
32	7.3847	30.2918	5.6219	0.3675	126.0812

Table 3.3: *Time (in s) spent to assemble, solve the mechanical and transport solver and to accomplish the simulation using the 1.2.6 version of MPI*

To begin, the evolution of speed up and efficiency with the increasing number of cores are presented in Figure 3.9.

Figures 3.9a and 3.9b show that, for the assembly part, a good scalability of 0.8 is obtained when using 32 cores. For the resolution part, a good scalability is obtained for

number of cores	Written Output Files	Generalized strain rate	Total without Output & initial partition	Remaining Time
1	82.3429	37.8287	900.5514	131.4393
2	56.2207	19.8565	482.8176	65.4918
4	46.6388	10.3177	251.2557	34.0809
8	71.9221	4.9659	137.9408	18.9752
16	124.9519	2.6144	82.4550	11.1744
32	61.0540	1.7141	52.7528	9.0869

Table 3.4: *Time (in s) spent to accomplish other relevant part of the code during simulation using the 1.2.6 version of MPI*

a small problem such as the convection problem. However, we notice a loss of scalability for the Stokes resolution. Figures 3.9c and 3.9d show the performance of the remaining part of the simulation. The speed up of the function implemented in CIMLib to compute the generalized strain rate is acceptable. But, when it comes to the rest, a big loss of scalability is shown. The sources of this loss should be investigated in order to improve the overall scalability of the simulations.

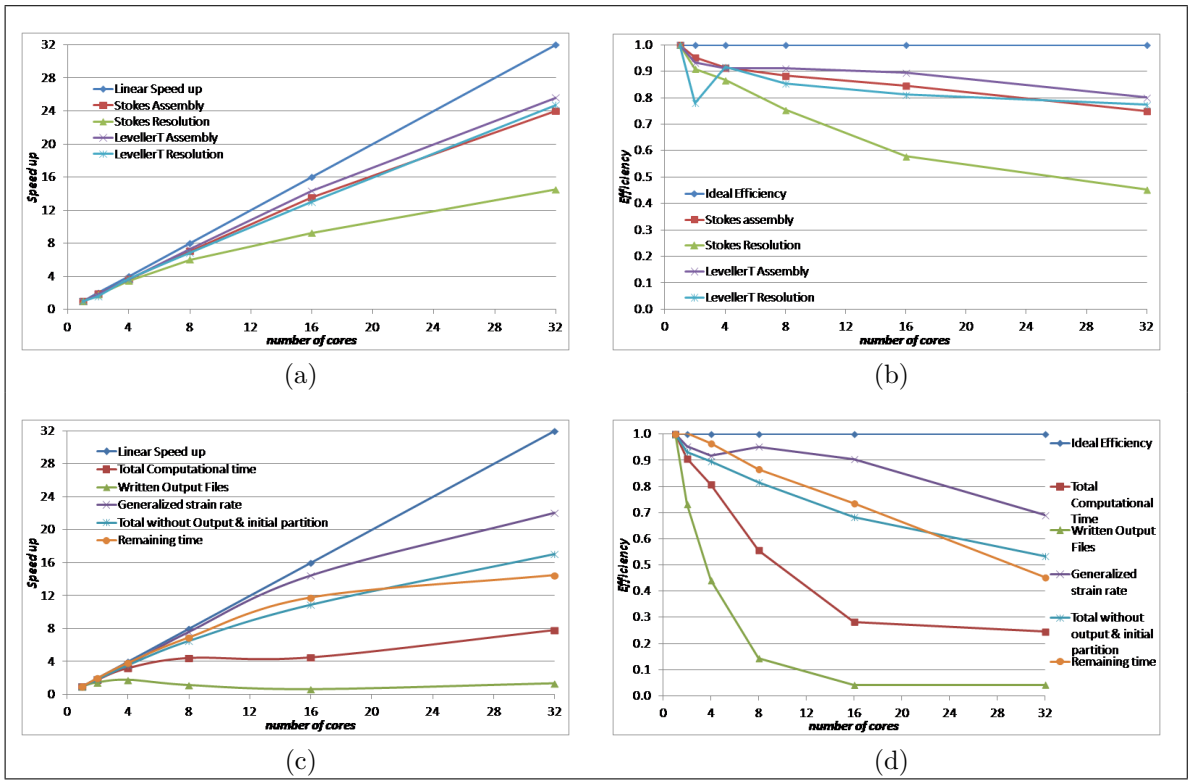


Figure 3.9: *Speed up (a-c) and efficiency (b-d) for the mechanical and transport solvers and different important part of the code obtained on the coarse mesh using the 1.2.6 version of MPI.*

Recent processors allow each core to communicate with its neighbors trying to optimize the attached process. Analyzing the results, we suspect that these communications can lead to an important loss of efficiency. Thus, we decided to migrate towards the 1.4.3 version of MPI. This newer version of MPI offers an option called "bind to core". This option is used to attach a process to a core throughout the whole simulation and hopefully guarantees a better performance.

number of cores	Stokes		LevellerT		Total Computational Time
	Assembly	Resolution	Assembly	Resolution	
1	177.5655	439.8508	142.0285	9.0568	984.0464
2	94.7523	270.4347	75.7627	6.5459	568.8039
4	50.7460	179.8544	39.9512	3.4741	351.5527
8	25.4583	90.4282	19.8727	1.6952	232.6729
16	12.9823	46.1396	10.0351	0.6594	220.7195
32	6.9151	24.7715	5.3670	0.3563	119.2165

Table 3.5: *Time (in s) spent to assemble, solve the mechanical and transport solver and to accomplish the simulation using the 1.4.3 version of MPI with the bind to core option*

number of cores	Written Output Files	Generalized strain rate	Total without Output & initial partition	Remaining Time
1	82.9321	37.1710	896.9749	128.4733
2	49.9486	19.6609	512.8915	65.3959
4	35.0572	9.7928	308.7822	34.7565
8	63.6903	5.2728	156.9266	19.4722
16	125.9468	2.6567	84.1534	14.3370
32	59.3700	1.3556	47.8224	10.4125

Table 3.6: *Time (in s) spent to accomplish other relevant part of the code during simulation using the 1.4.3 version of MPI with the bind to core option*

The same tests are relaunched but this time using the "bind to core" option. Again, the time consumed by the different parts of the code are presented in Table 3.5 and Table 3.6. The "bind to core" option does not change noticeably the final computational time. When examining the speed-up and efficiency presented in Figure 3.10, we notice that the efficiency of the Stokes resolution is improved from 0.43 to 0.58 (for 32 cores). Nevertheless, the loss of efficiency is mainly visible for less than 16 cores. We find that

the latter is reasonable when eliminating the communication between cores (i.e. "bind to core" option) and when utilizing less than the full cores of a computing node.

To summarize, we can say that the 'bind to core' option has proven to be essential to insure a good scalability. The overall efficiency is satisfactory, it decreases linearly when using more than 4 cores. However we can notice a loss of efficiency when using less than 4 cores. This behavior is clear in Figure 3.9. This local loss can be attribute to the architecture of the multi-core processor. The next section is the key to prove this point. When iterating between different computing nodes (i.e using Round Robin option), we gain respectively 34 s and 15 s when using 2 and 4 cores (check Table 3.8). So we consider this loss (for $p \leq 4$) as a direct outcome of the hardware limitation. Accordingly, these algorithms especially the resolution algorithm are scalable.

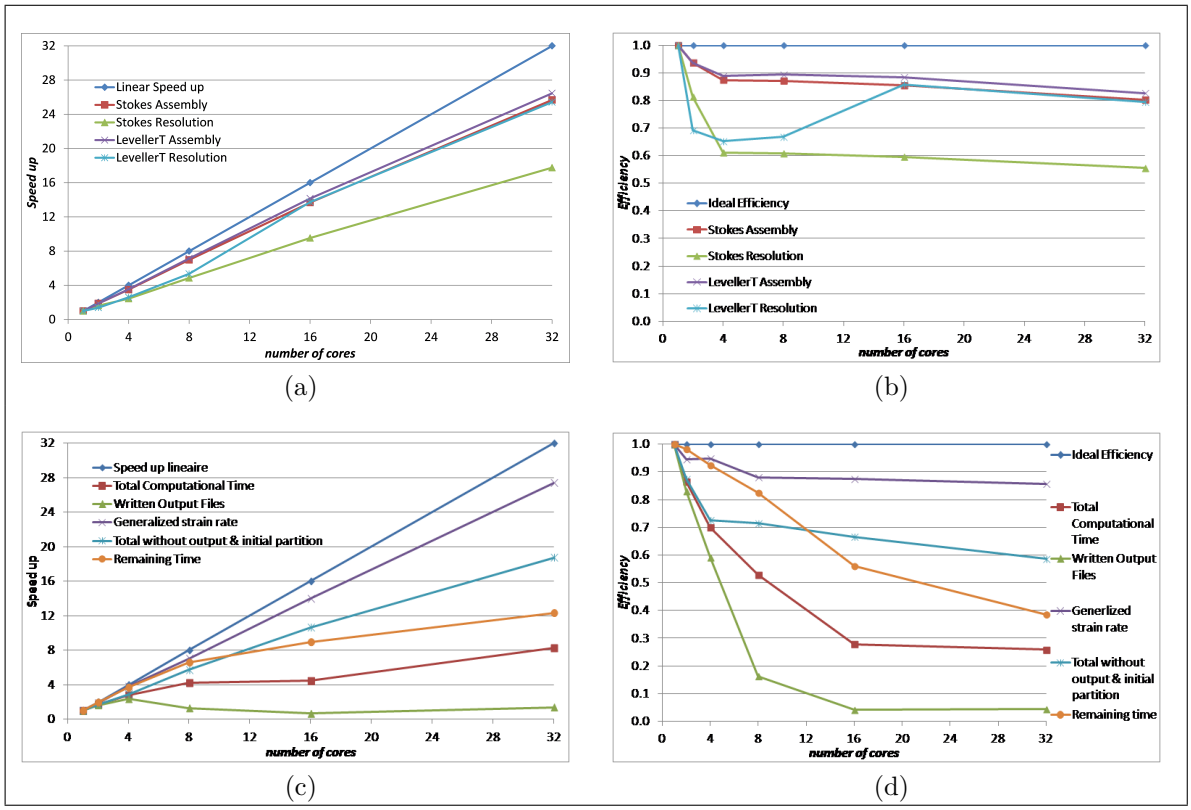


Figure 3.10: *Speed up (a-c) and efficiency (b-d) for the mechanical and transport solvers and different important part of the code obtained on the coarse mesh using the 1.4.3 version of MPI with the bind to core option.*

If leaving aside the scalability of the output generating algorithm, we prove that the scalability of our code, is mostly limited by the scalability of the Stokes system.

The scalability loss of the output files generation is investigated next. The aim is to determine how the hardware can be manipulated to obtain the best results our machines can offer.

3.5.4 Coarse mesh: Hardware limitation and Round Robin Option

When analyzing the efficiency of our code in the previous sections, it was noticeable that the scalability of files generation algorithms was not satisfactory (Figure 3.9c). This result was expected since it is directly related to the used hardware. To understand better, we should explain that the used hard disc is formed by 3 platters and 3 heads which enable it to write in parallel. The speed up was obtained until reaching a specific number of cores (equal to 4 cores in Figure 3.9c) which means until all cores of a node are used. Afterward, the hard disc fails to manage properly writing in parallel.

This point is confirmed in Table 3.7: Though the size of the results files are equally divided on the number of used cores (i.e. each core writes the same amount of data), the hardware is limited and fails once more than 4 cores are used.

So the reasons behind the loss of scalability for output files generation is i) simulations are launched locally on nodes while results files are written in a parallel way (using vtk file format ".pvtu") and ii) the hard disc head's cannot manage data when different cores are in the process of writing output files.

number of cores	Size of Output files pvtu (Kb)	vtu (Mb)
1	1.5	442
2	1.5	219
4	1.6	108
8	1.7	53
16	2.1	27
32	2.7	14

Table 3.7: *Size of the Output files in parallel computing*

To try and extract the best of what this hardware can offer, the use of "Round Robin" option is recommended. This option can be handy to promote the scalability of the written output files.

The "Round Robin" option works in the following manner: it iterates on the available nodes attempting to distribute all processes equally on each node. This means that as soon as the code is launched on more than two cores, it uses two different nodes and accordingly two Hard Discs. Next, the same simulations are repeated using a "Round Robin" option.

The results presented in Table 3.8 and Table 3.9 confirms that nothing is affected

but the output algorithms. In addition, Table 3.9 and Figure 3.11 show that both time and speed-up of the output algorithms are improved. By redistributing processes equally on each nodes, the "Round Robin" option increased the limit number of cores from 4 to 8 cores (i.e. for which the hard disc stop writing properly afterward). Still, after 8 cores the time used to write output files increases and the speed up drops again.

number of cores	Stokes		LevellerT		Total Computational Time
	Assembly	Resolution	Assembly	Resolution	
1	178.1416	440.9533	146.1969	9.0786	904.0554
2	93.0375	236.6903	78.0407	5.842	480.9263
4	49.3545	143.6863	39.6819	2.9108	269.7911
8	25.355	90.8500	20.0234	1.6955	156.3066
16	13.3082	46.3393	10.1175	0.6729	81.5517
32	6.9328	24.6166	5.3414	0.3682	47.9026

Table 3.8: *Time (in s) spent to assemble, solve the mechanical and transport solver and to accomplish the simulation using the 1.4.3 version of MPI with the opemmpi round robin option*

number of cores	Written Output Files	Generalized strain rate	Total without Output & initial partition	Remaining Time
1	82.7806	37.6434	904.0554	129.6850
2	43.2563	20.4194	480.9263	67.3158
4	27.9493	9.9888	269.7911	34.1576
8	19.3887	4.9998	156.3066	18.3827
16	28.6615	2.6957	81.5517	11.1138
32	61.0352	1.3785	47.9026	10.6436

Table 3.9: *Time (in s) spent to accomplish other relevant part of the code during simulation using the 1.4.3 version of MPI with the openmpi round robin option*

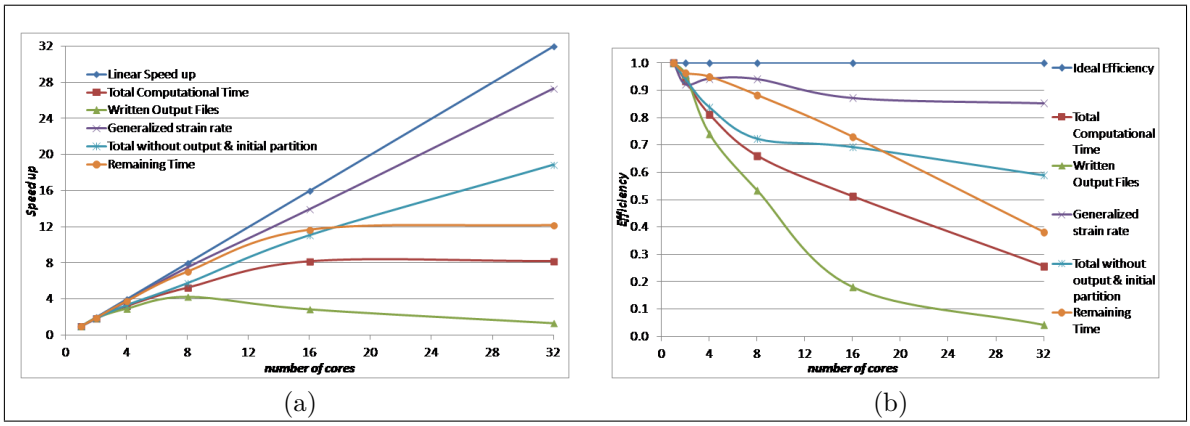


Figure 3.11: *Speed up (a) and efficiency (b) for different part of the code obtained on the coarse mesh using the openmpi round robin option.*

To summarize, the output generation scalability is bound by the used hardware limitation. "Round Robin" option proved to be a great way to boost the scalability but still narrowed by the hard disc capacities.

3.5.5 Refined mesh

Using a refined mesh is expected to give better scalability. In fact, the workload of a core will increase. Thus, the ratio workload/communication using a refined mesh will increase as well leading to a better scalability. The same simulation is launched using up-to 128 cores. The results for time consuming are summarized in Table 3.10 and Table 3.11. The simulation time is considerably greater since more nodes are used in the new refined mesh. The overall behavior of the solver remains the same: the solvers consume the bigger part of the total computing time.

number of cores	Stokes		LevellerT		Total Computational Time
	Assembly	Resolution	Assembly	Resolution	
1	3162.2112	25497.1773	2492.3585	427.7585	34844.7487
2	1679.9621	13875.3077	1389.4449	275.8599	19218.6489
4	850.2076	9572.5892	679.7318	129.1397	12424.3732
8	411.8736	4127.7119	329.8567	57.4261	6291.4682
16	211.6616	2100.1478	166.1422	25.2834	4556.1730
32	106.0528	1021.6278	81.7341	10.4183	2597.8566
64	57.3246	508.7427	42.2413	6.9786	1242.0377
128	33.0700	287.6470	22.8955	5.8589	828.2126

Table 3.10: *Using the refined mesh: Time (in s) spent to assemble, solve the mechanical and transport solver and to accomplish the simulation using the 1.2.6 version of MPI*

number of cores	Written Output Files	Generalized strain rate	Total without Output & initial partition	Remaining Time
1	1193.1272	629.5313	33616.8978	2037.3923
2	833.5299	334.6115	18324.3467	1103.7721
4	516.2075	165.2059	11812.8972	581.2289
8	982.5606	80.9494	5217.2592	290.3909
16	1712.9052	40.1335	2706.2488	203.0138
32	1160.9286	21.0772	1325.9360	106.1030
64	444.2012	10.3300	704.9010	89.6138
128	291.3921	5.5548	417.4245	67.9531

Table 3.11: *Using the refined mesh: Time (in s) spent to accomplish other relevant part of the code during simulation using the 1.4.3 version of MPI with the bind to core option*

Figure 3.12 completes this discussion. The Stokes resolution remains the least scalable with an efficiency of 0.8. It is interesting to mention that after 32 cores LevellerT changes behavior and becomes less scalable. It is totally expected for small problems such as the convection problem (i.e LevellerT). Increasing the number of cores is efficient to a certain limit; afterward the scalability decreases due to an increasing communication between cores.

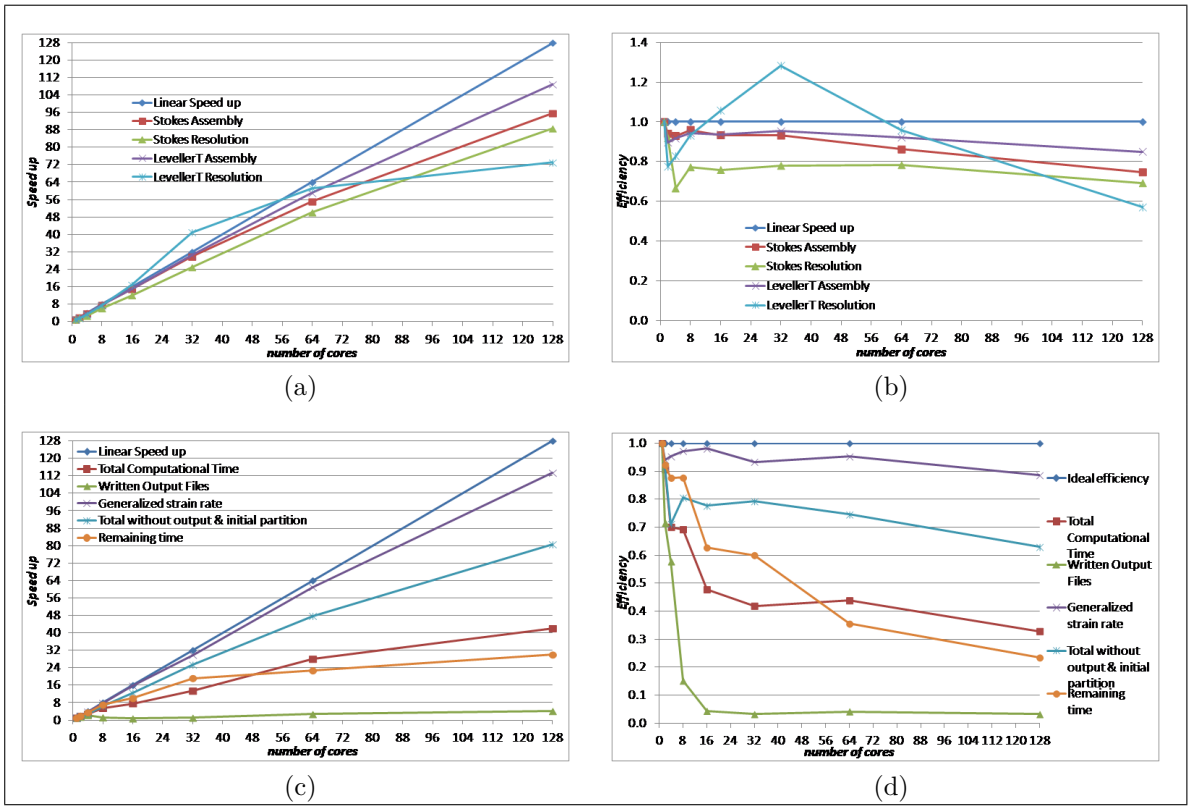


Figure 3.12: *Using the refined mesh: Speed up (a-c) and efficiency (b-d) for the mechanical and transport solvers and different important part of the code using the 1.4.3 version of MPI with the bind to core option*

To conclude, the study presented hereby proves the scalability of the code using up to 128 cores. The global behavior is very consistent with the results obtained on the coarse mesh. Nevertheless an undeniable improvement was detected: Increasing the number of nodes improves enormously the obtained speed up (check the speed up using 32 cores for instance). The behavior using refined mesh was proven to be satisfactory and consequently, only hardware limitations can be held accountable for any loss of efficiency.

3.5.6 Anisotropic mesh: Adaptation Scalability

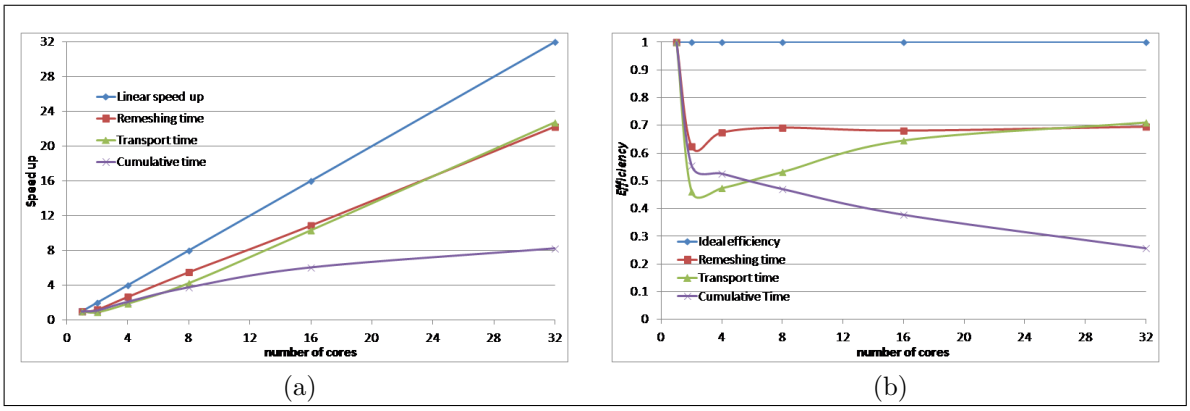
Anisotropic mesh adaptation, described in chapter 2, is an important part of our approach. Its scalability highly matters in our framework. Its performance will be tested in this section in order to know if any improvements are needed.

Recall that the anisotropic mesh adaptation used is a h-adaptation. Thus at each adaptation step, three algorithms should be executed : i) the re-meshing algorithm where the mesh generator "MTC" generates the mesh according to the metric constructed at each node. ii) The transport algorithm where the solution and all essential fields are transfered from the previous mesh to the newly generated one in order to continue the

computation. In other words, this step relies simply on interpolating the solution between the old mesh and the new one. iii) The dynamic load balancing algorithm to ensure a good workload distribution between different cores. Table 3.12 and Figure 3.13 show explicitly the results (computation time, speed up, efficiency) for both re-meshing and transport algorithms. The total adaptation time contains in addition the time spent executing the dynamic load balancing algorithm. Figure 3.13b shows that the scalability is steady over 4 cores. Even the transport caught up with the remeshing part when using 32 cores. The loss of scalability between 1 and 2 cores is due to the number of time the adaptation scheme is used depending of the computation environment (whether parallel or sequential). In a parallel execution the adaptation scheme is repeated two times to ensure a good quality of the mesh.

number of cores	Remeshing Time	Transport Time	Cumulative Time
1	252.512	31.812	1418.736
2	202.375	34.482	1364.411
4	93.631	16.783	753.328
8	45.648	7.476	419.367
16	23.178	3.081	248.605
32	11.355	1.400	185.432

Table 3.12: *Times (in s) spent to accomplish the anisotropic mesh adaptation and its different parts*.



(a) (b)

Figure 3.13: *Speed up (a) and efficiency (b) for the anisotropic mesh adaptation technique used in* CIMLib.

The loss of efficiency visible for the total adaptation time is due to the iterative method used in CIMLib. The algorithm starts with a given initial partitioned mesh. At first, the nodes shared between cores are fixed; and the re-meshing is done on nodes

belonging to one and only one core. Then the dynamic load balancing algorithm is executed. This process is repeated until the couple mesh/solution has converged. This iterative scheme is illustrated in Figure 3.14. For more details, we encourage the readers to refer to [Digonnet et al., 2007, Digonnet et al., 2013]. Finally, It is important to mention that the time spent to adapt the mesh is equal to the time needed for Navier Stokes solver when adapting every 10 time steps (adaptation frequency = 10). In other words, the time is doubled. Despite this fact, we recommend using the adaptation technique since it improves drastically the obtained precision (see chapter 4).

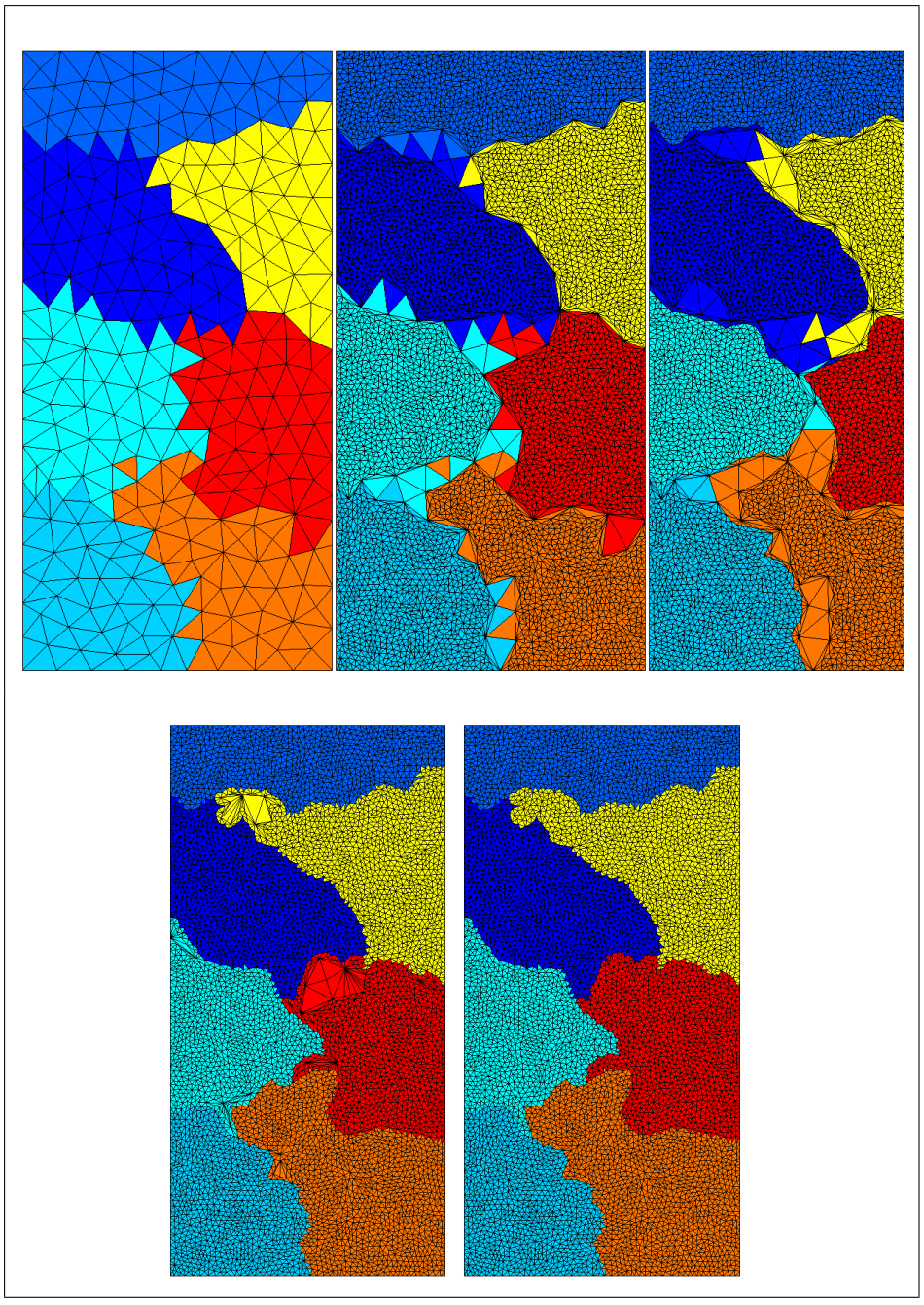


Figure 3.14: *Dynamic load balancing algorithm used in* CIMLib.

3.6 Conclusion

In this chapter, we presented a general description of parallelism in modern computers. We focused essentially on the parallel environment in CIMLib. The main objective was to test the response of the library in a massively parallel environment to determine if any improvements are needed. In particular, important solver such as the Stokes and the convection solver LevellerT were inspected closely. The study has shown the need to optimize the latter solver in term of assembly. An MPI migration to newer version was recommended to insure better performance when exceeding 16 cores. As a result, the solvers performance was improved and found satisfactory using up to 128 cores. The efficiency of the mesh adaptation was tested as well. A loss of efficiency was noticed in the resulting dynamic load balancing. Though not crucial, we should note that it is mainly attributed to the used iterative partitioning method. Output generation exhibited a loss of efficiency also, it is for the most part associated the our hardware limitation and does not reflect programing deficiency.

To conclude and despite the casual loss, the overall scalability is found satisfactory. The performance of the major parts including the solvers and mesh adaptation was judged solid.

Advanced applications combining information and recommendations from both chapter 2 and chapter 3 are the focal point of the next chapter. The results are assessed and confronted with *Forge*® simulations.

Résumé en Français

Dans ce chapitre, nous avons présenté une description générale du parallélisme dans les ordinateurs modernes. Nous nous sommes concentrés essentiellement sur l'environnement parallèle au CIMLib. L'objectif principal était de tester la réponse de la bibliothèque dans un environnement massivement parallèle pour déterminer si des améliorations sont nécessaires. En particulier, le solveur mécanique de Stokes et le solveur de convection LevellerT ont été inspectés de près. L'étude a montré que c'est recommandé d'optimiser le solveur de convection en termes d'assemblage. Une migration de MPI vers une nouvelle version a été recommandée aussi. Ainsi, meilleures performances sont assurées lorsque nous utilisons plus que 16 cœurs au cours d'une simulation. Par conséquent, la performance des solveurs a été améliorée et jugée satisfaisante en utilisant jusqu'à 128 cœurs. L'efficacité de l'adaptation de maillage a été testée aussi bien. Une perte d'efficacité est remarquée dans l'équilibrage de charge dynamique. Bien que non essentiel, il faut noter qu'il est principalement attribuable à la méthode de partitionnement itérative utilisée. La génération de sorties présentait une perte d'efficacité aussi. C'est associé à la limitation du matérielle utilisé et ne reflète pas de carence en programmation. Pour conclure et malgré la perte occasionnelle, la performance globale est jugée satisfaisante. Dans ce qui suit, des applications avancées combinant les recommandations des chapitre 2 et 3 sont le point focal du prochain chapitre. Les résultats sont évalués et confrontés à des simulations de Forge ®.

Bibliography

[Amdahl, 1967] Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485. ACM.

[Balay et al., 2014a] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Rupp, K., Smith, B. F., and Zhang, H. (2014a). PETSc users manual. Technical Report ANL-95/11 - Revision 3.5, Argonne National Laboratory.

[Balay et al., 2014b] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Rupp, K., Smith, B. F., and Zhang, H. (2014b). PETSc Web page. http://www.mcs.anl.gov/petsc.

[Balay et al., 1997] Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F. (1997). Efficient management of parallelism in object oriented numerical software libraries. In Arge, E., Bruaset, A. M., and Langtangen, H. P., editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press.

[Digonnet, 2001] Digonnet, H. (2001). *Repartitionnement dynamique, mailleur parallèle et leurs applications à la simulation numérique en mise en forme des matériaux*. PhD thesis, école nationale supérieure des mines de paris.

[Digonnet et al., 2007] Digonnet, H., Bernacki, M., Silva, L., and Coupez, T. (2007). Adaptation de maillage en parallèle, application à la simulation de la mise en forme des matériaux. *18ème Congrès Français de Mécanique (Grenoble 2007)*.

[Digonnet et al., 2013] Digonnet, H., Silva, L., Coupez, T., et al. (2013). Massively parallel computation on anisotropic meshes. *Adaptive Modeling and Simulation 2013-Proceedings of the 6th International Conference on Adaptive Modeling and Simulation, ADMOS 2013*.

[Dongarray et al., 2001] Dongarray, J. J., Luszczeky, P., and Petitetz, A. (2001). The linpack benchmark: Past, present, and future. Technical report, University of Tennessee, Department of Computer Science.

[Eijkhout et al., 2014] Eijkhout, V., Chow, E., and de Geijn, R. V. (2014). *Programming on Parallel Machines: GPU, Multicore, Clusters and More*. Lulu.com.

[Flynn, 1972] Flynn, M. (1972). Some computer organizations and their effectiveness. *IEEE Trans. Comput.*

[Gustafson, 1988] Gustafson, J. L. (1988). Reevaluating amdahl's law. *Communications of the ACM*, 31:532–533.

[Hager and Wellein, 2010] Hager, G. and Wellein, G. (2010). *Introduction to High Performance Computing for Scientists and Engineers*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition.

[Hendrickson and Devine, 2000] Hendrickson, B. and Devine, K. (2000). Dynamic load balancing in computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 184(2-4):485 – 500.

[Jimack, 1999] Jimack, P. (1999). An overview of parallel dynamic load-balancing for parallel adaptive computational mechanics codes.

[Magoulès and Roux, 2013] Magoulès, F. and Roux, F.-X. (2013). *Calcul scientifique parallèle - Cours, exemples avec Openmp et Mpi, exercices corrigés*. Dunod.

[Petitet et al., 2008] Petitet, A., Whaley, R. C., Dongarra, J., and Cleary, A. (2008). A portable implementation of the high-performance linpack benchmark for distributed-memory comput. http://www.netlib.org/benchmark/hpl/.

[Rauber and Rünger, 2010] Rauber, T. and Rünger, G. (2010). *Parallel Programming-For Multicore and Cluster Systems*. Springer Berlin.

[Van Driessche and Roose, 1995] Van Driessche, R. and Roose, D. (1995). An improved spectral bisection algorithm and its application to dynamic load balancing. *Parallel Comput.*, 21(1):29–48.

[Williams, 1991b] Williams, R. (1991b). Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency*, 3:457–481.

[www.top500.org,] www.top500.org. http://www.top500.org/.

CHAPTER 4

Applications

Contents

4.1 Introduction 124

4.2 Industrial applications 124

4.2.1 Connecting rod 125

4.2.2 Crankshaft 128

4.3 Multi-domain applications 132

4.3.1 Ten Parallelepiped 132

4.3.2 256 spheres 137

4.4 Conclusions and Discussions 140

Résumé en Français 142

Bibliography 143

4.1 Introduction

This chapter is devoted to the different applications of the Eulerian monolithic approach described in chapter 2. Two sets of applications are presented. In the first, industrial applications such as the forging processes of rods and crankshafts are highlighted. The second is dedicated to the deformation of more simple geometries (such as spheres and Parallelepiped) but in a multi-domain context. In other words, the numerous deformable bodies (up to 250 bodies) will be stacked in the computational domains where there evolution will be followed during the simulation.

The aim of these cases will be testing both the capacity and the reliability of the method for complex geometries (in the industrial applications) in multi-domain situations.

Both the advantages and the limitations will be detailed and discussed. Consequent improvements will be proposed as well leading to a decisive conclusion if the approach is fit to these kind of applications.

4.2 Industrial applications

Besides pistons, crankshafts and connecting rods are the main components of the reciprocating piston engine used in cars for instance.

The connecting rod (Figure 4.1a) is the element joining the piston to the rotating crankshaft (Figure 4.1b).



(a) (b)

Figure 4.1: *Examples of a connecting rod (a) and a crankshaft (b).*

Whether casted, forged or machined the production of the connecting rod and crankshaft remains critical. Connecting rods should sustain complex loads such as the

cycle of high compressive loads followed by high tensile ones; while cranks should be hardened and strengthen to endure bending and twisting during full functioning.

For these reasons alone, such pieces are the topics of numerous researches including ours. In the aim of providing the industrial partners a simulation tool responding to their needs, different cases are explored next, using the monolithic Eulerian approach presented in chapter 2. This attempt is first of a kind since these problems are typically treated using fully Lagrangian approach

4.2.1 Connecting rod

In this section, the method capacity to model the deformation of a complex geometries is tested. For instance, the forging of a connecting rod is illustrated next. Figure 4.2 shows the position of the initial geometry (orange) between the two tools for $t = 0s$. Notice that, the details of the initial geometry of the deformable body and the tools are perfectly captured due to the anisotropic mesh. The used mesh is adapted on the gradient of both deformable body and tools level set. It contains 119 000 nodes and 687 784 elements. The simulation is launched using 16 cores.

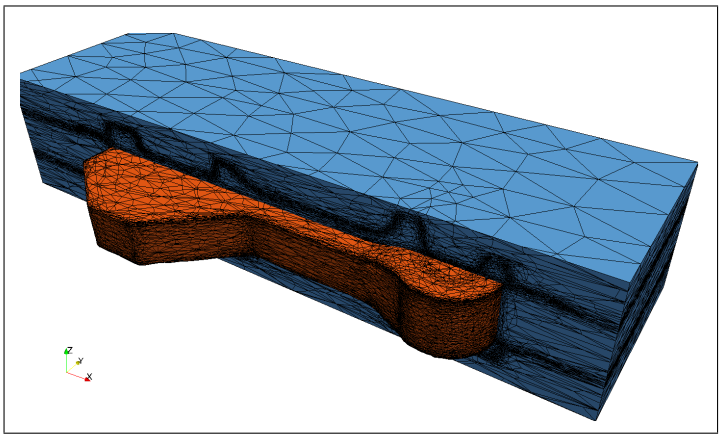


Figure 4.2: *Geometry of the connecting rod immersed in the computational domain and adapted using an anisotropic mesh ($t = 0\,s$).*

For a better visualization, a longitudinal section is represented in Figure 4.3. Both the contour of the deformable body (orange) and the tools (blue) are depicted and are perfectly smooth. The lower tools is fixed ($v = 0\,mm/s$), whereas the upper tools moves with a velocity equal to $-10\,mm/s$ along the Z axis. The tools consistency and the density are respectively equal to $1000\,MPa.s$ and $8\,g/cm^3$. The deformable body consistency and density are respectively equal to $352\,MPa.s$ and $2.8\,g/cm^3$. As for the air, the viscosity is equal to $1 \times 10^{-7}\,MPa.s$ and the density is equal to $1.2\,g/cm^3$.

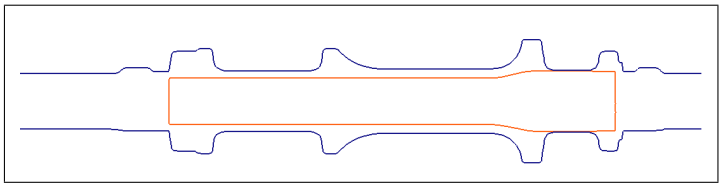


Figure 4.3: *The contours of the deformable body and tools presented on a longitudinal section in the thickness at t=0 s*

Figure 4.4 represents the zero iso-value of the deformable body for $t = 0\,s$ (on the left) and $t = 2.6\,s$ (on the right). We mention that, based on the same analogy presented in chapter 2, the air is allowed to evacuate by imposing a porous boundary conditions. The latter is equivalent to defining vents in the tools.

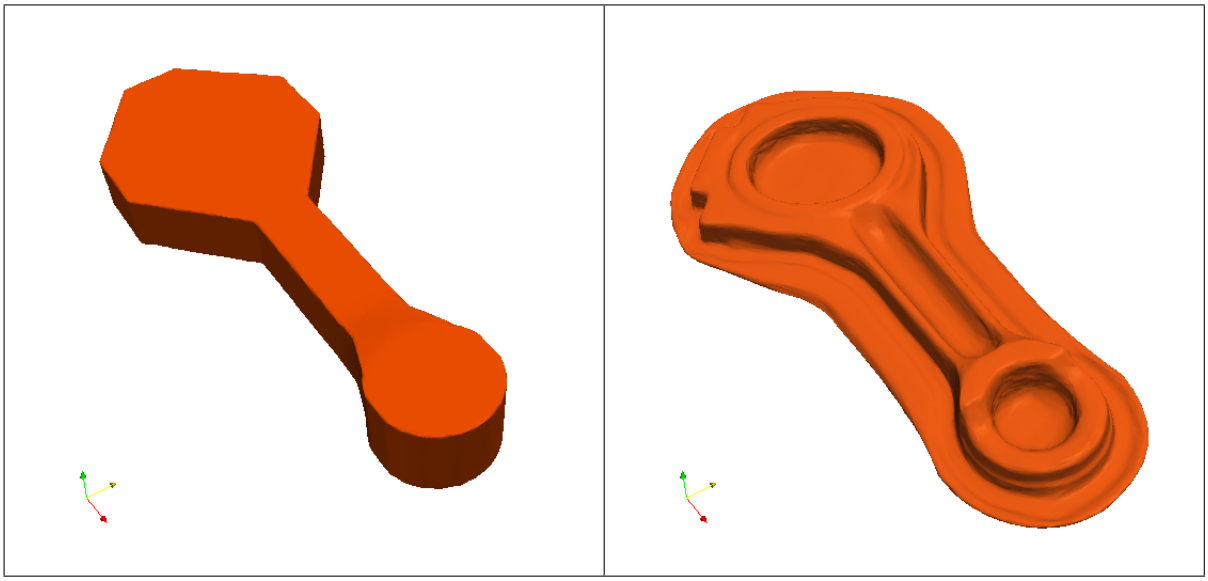


Figure 4.4: *From left to right: the isovalue zero of the connecting rod respectively for* $t = 0\,s$ and $t = 2.6\,s$.

Qualitatively, our simulation gives a good results. To validate quantitatively our approach, the same case is simulated using Forge®. The Lagrangian mesh contains 37 483 nodes 42 438 elements. A comparison between the different results is presented in Figure 4.5 and Figure 4.6.

Figure 4.5 illustrates the pressure and velocities maps for both software for $t \approx 1.6\,s$. The chosen cut corresponds to the mid-plane of the deformable body. Results are very similar and represent the same features. The main difference, thought not significantly important, is mostly located in the center. For instance, the velocity along the x-axis represents a gap of $1.7\,mm/s$. This is directly related to the meshes illustrated in Figures 4.5a and 4.5b. In the CIMLib case, lesser nodes are used in the center leading to this difference. The same reason is behind the difference for the pressure and the velocity

along the y-axis. Though, the Eulerian cut contains approximately half the nodes of the Lagrangian one (2900 nodes vs. 6000 nodes), the results remains similar and satisfactory.

Figure 4.6 completes the comparison. The velocity component v_z is represented on a longitudinal section in the thickness. Once again, the results are found very comparable.

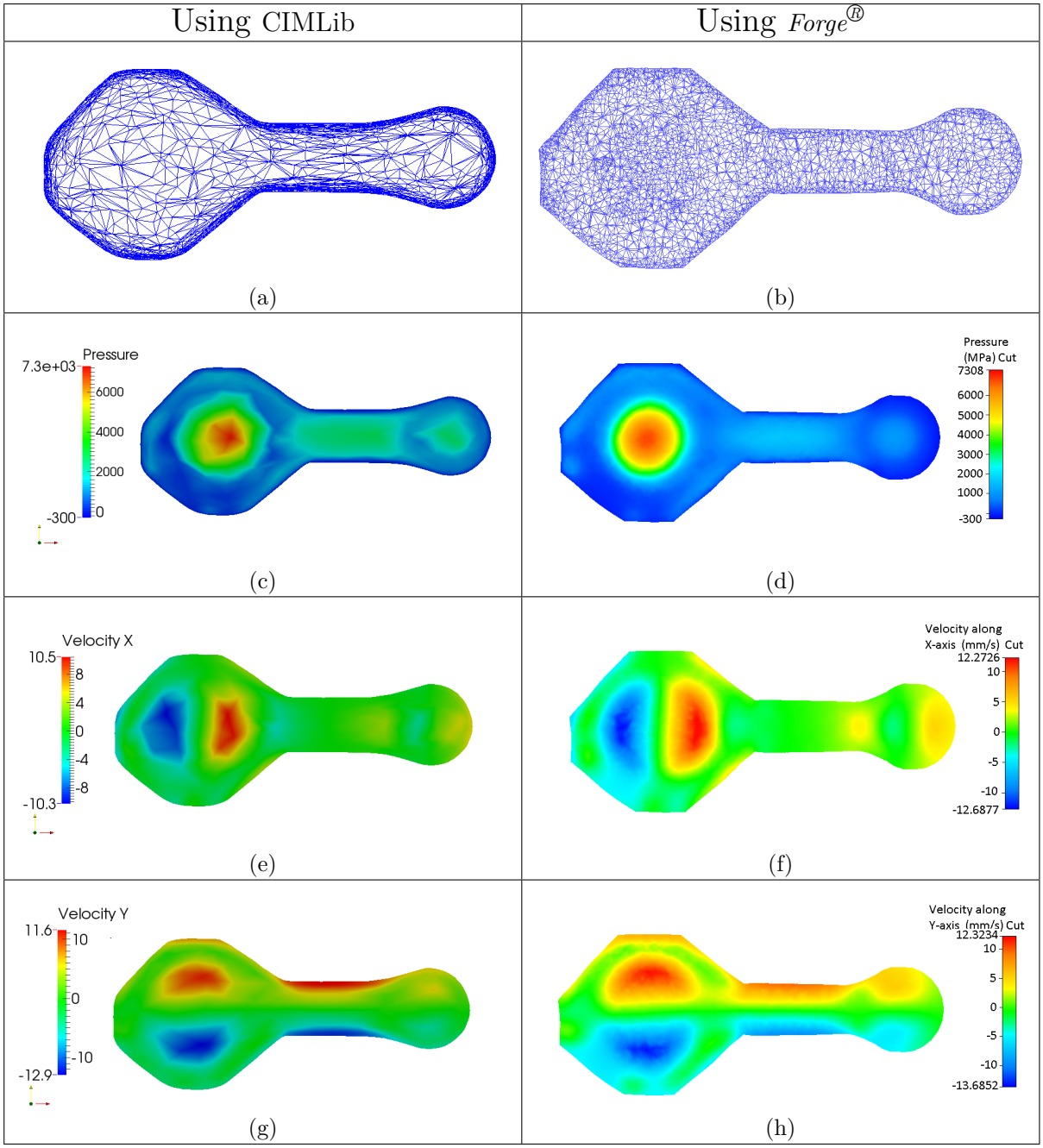


Figure 4.5: *Comparison of the results obtained with* CIMLib *and a Forge® simulation at* $t \approx 1.6\,s$. *The same longitudinal section in the width is used in both softwares: (a) and (b) illustrate the mesh, (c) and (d) illustrate the pressure, (e) and (f) illustrate the velocity component* v_x *and (g) and (h) illustrate the velocity component* v_y.

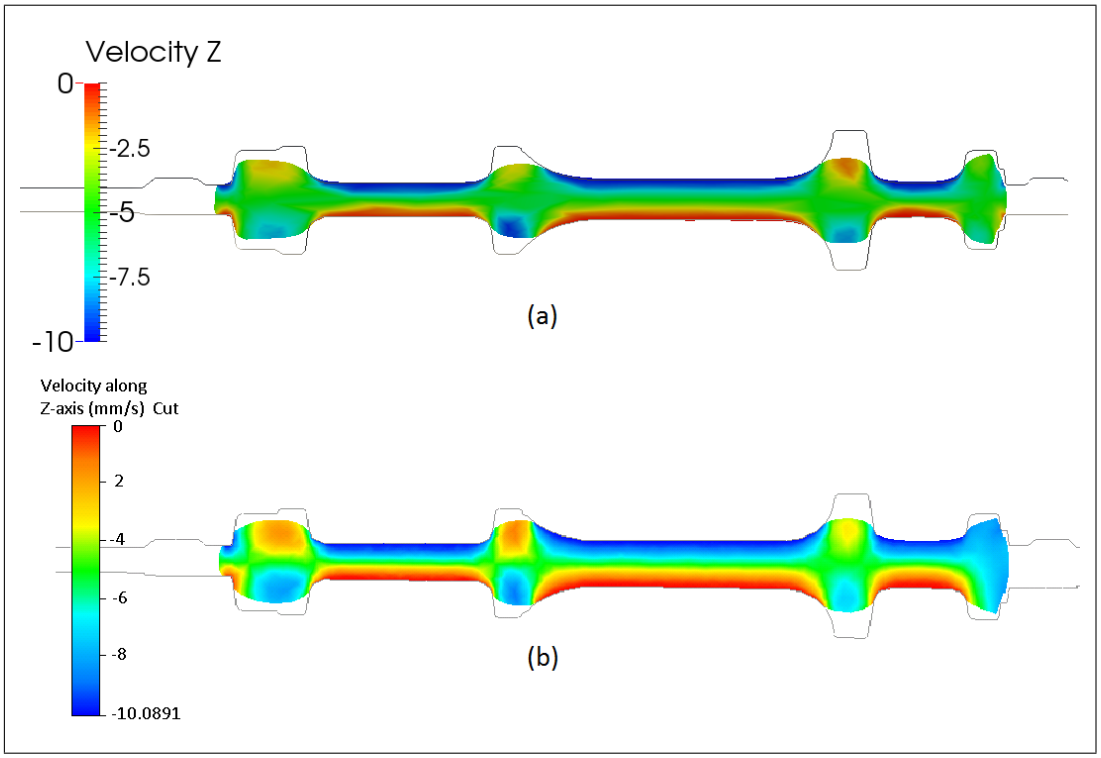


Figure 4.6: *Comparison of the the velocity component v_z obtained with* CIMLib *(a) and Forge® (b) at $t \approx 1.6\,s$. The same longitudinal section in the thickness is used in both softwares.*

To conclude, the Eulerian monolithic approach gives good results when compared with the Lagrangian approach. The slight differences shown in Figure 4.5 and Figure 4.6 are the outcome of the different used re-meshing techniques (an anisotropic adaptation technique in the Eulerian approach versus a re-meshing technique in the Lagrangian approach). Figure 4.5a shows that the nodes concentration in the Eulerian approach is located on the border of the deformable body. Whereas the re-meshing in the Lagrangian approach is based on the deformation and thus distributes the nodes appropriately in the discretized deformable body. Therefore the maps in the right part of Figure 4.5 are smoother then the left one.

4.2.2 Crankshaft

The aim of the next case is to test the program performance for even more complex geometries such as the crankshaft. To push the evaluation even further, the same program used in sec. 4.2.1 for the connecting rod is used again. Only the tools and piece geometries are replaced. The same characteristics are used as well . The mesh remains intact. In other words, the same number of nodes and elements is used.

At $t = 0\,s$, the deformable piece is placed between the two tools (illustrated in Figure 4.7). The upper tool moves with a velocity equal to $v_z = -80\,mm/s$ while the

lower tool remains fixed ($v = 0\,mm/s$). Though the tools show more intricate details than the last ones, the mesh seems to captures every aspect. In Figure 4.8, the perfectly meshed upper tool confirms that the current mesh is versatile.

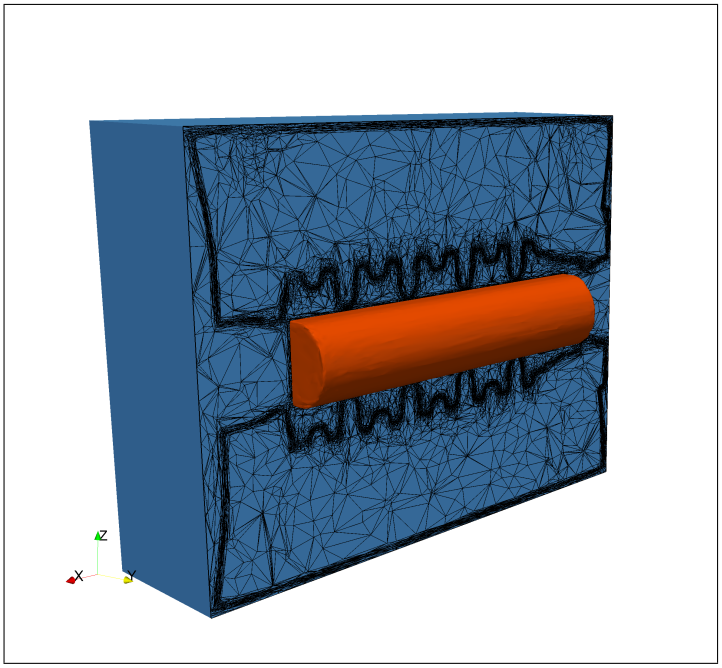


Figure 4.7: *Geometry of the crankshaft immersed in the computational domain between the tools represented by the anisotropic mesh ($t = 0\,s$).*

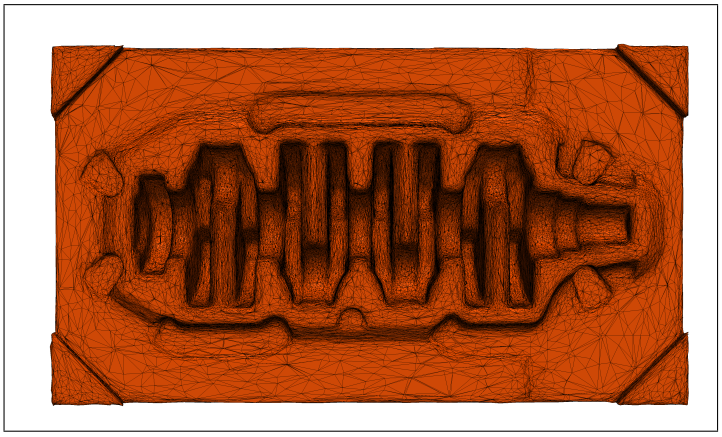


Figure 4.8: *Mesh projected on the zero iso-value of the upper tool*

At $t \approx 0.9\,s$, the isovalue of the deformable piece is illustrated in Figure 4.9. The crankshaft seems to take well the tools form. To complete the discussion, the velocities maps all over the deformable piece are presented in Figure 4.10. The velocities components v_x and v_y are bigger on the edges depicting the direction of the material flow. On the top, v_z takes the upper tool velocity value $80\,mm/s$ which is expected. The overall behavior seems satisfactory.

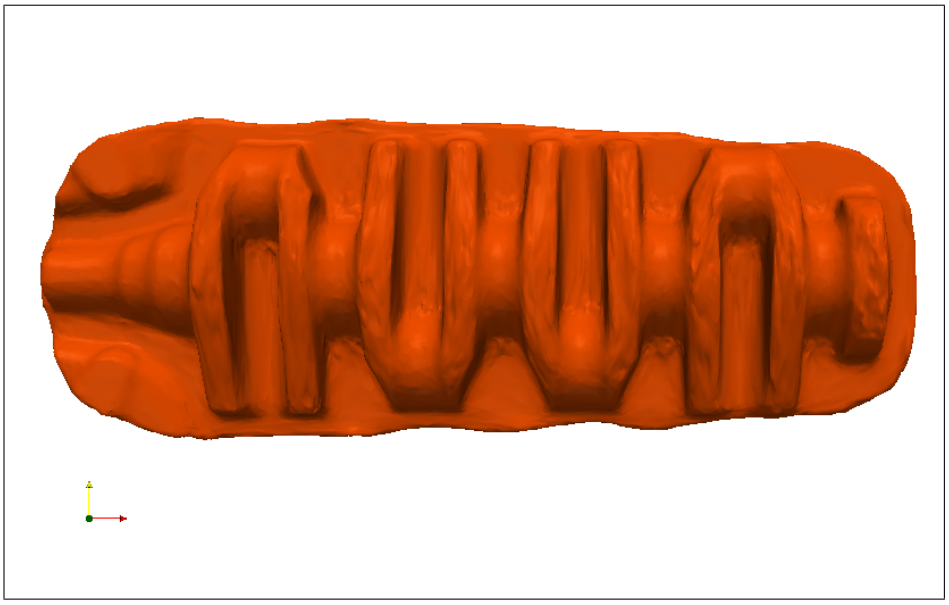


Figure 4.9: *Geometry of the crankshaft at* $t \approx 0.9\,s$

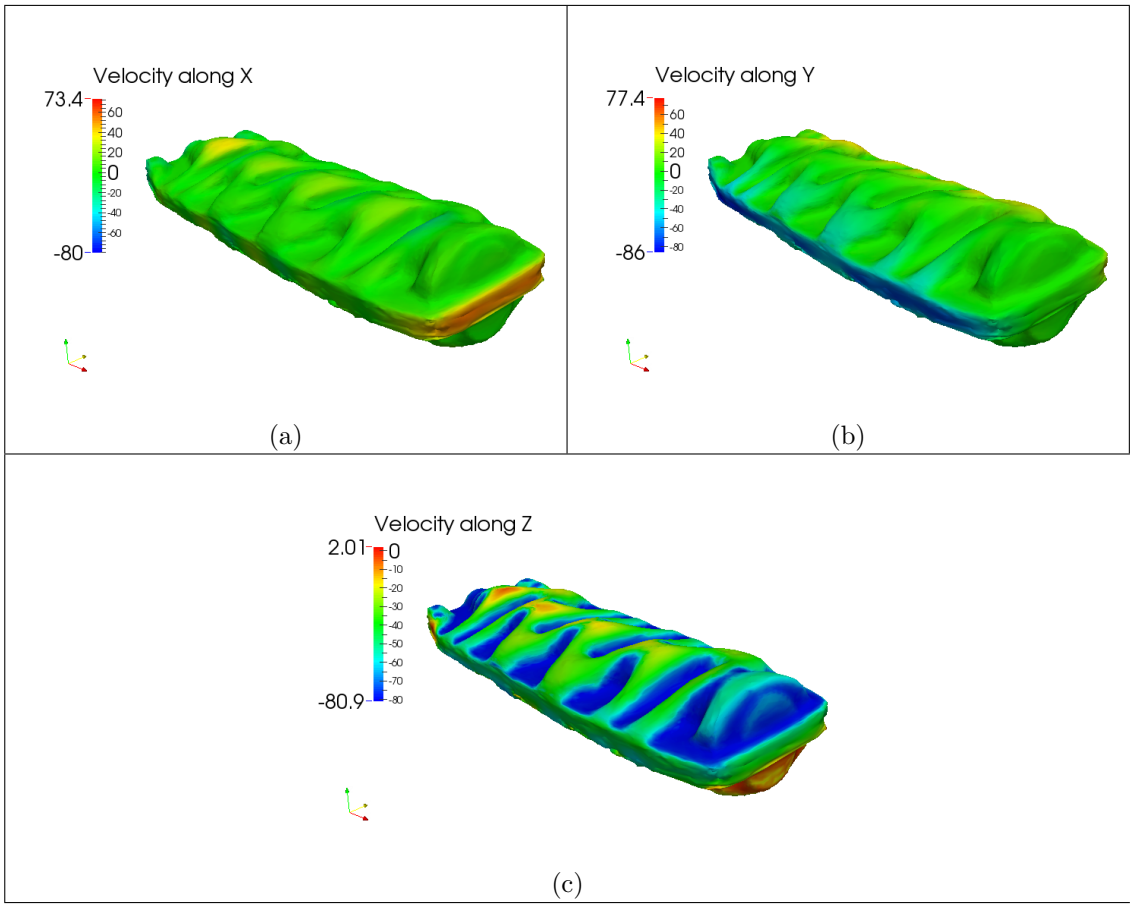


Figure 4.10: *(a), (b) and (c) represent respectively the velocity components* v_x, v_y *and* v_z *on the whole crankshaft geometry at* $t \approx 0.9\,s$.

Nevertheless, further inspection shows some lack of precision. Figure 4.11 plots the zero-isovalue of the tools and the deformable piece on a longitudinal section in the thickness. We have noticed an increasing penetration between the upper tool and the deformable piece. This suggests that even though the current mesh is sufficient to perfectly represent the initial geometries; it needs more nodes to capture the complex shape of the deformable piece during time. Notice that the mesh generator tried to salvage the situation by migrating nodes from the edge to the contact zone, but fell short. Figure 4.12 confirms this argument. The mesh was able to capture less complicated forms (such as the connecting rod) and thus no penetration was noticeable.

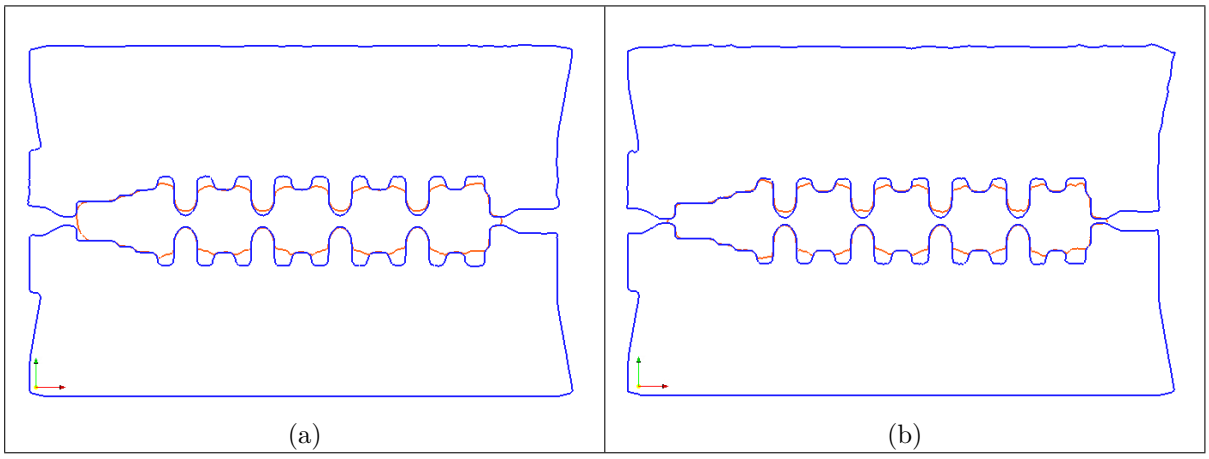


Figure 4.11: *(a) and (b) represent respectively the contours of the crankshaft between the tools at* $t \approx 0.9\,s$ *and* $t \approx 0.96\,s$*. A growing penetration of the upper tool in the deformable piece is noticeable*

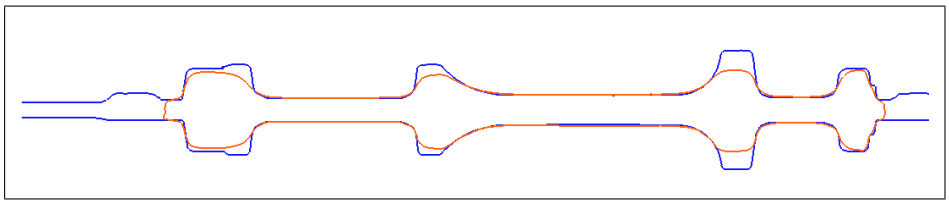


Figure 4.12: *The zero-isovalue of the connecting rod between the tools at* $t \approx 1.6s$ *. No penetration between the different bodies is noted.*

In addition, the upper tool is translated analytically along the $z-$axis while the level set of the deformable body is convected using the velocity (solution of Navier-Stokes). In other words, the movement of the upper tool does not depend of the mesh since it is analytically determined at every time step. This can result in a clash of the bodies for lack of precision. This is why no penetration occurs with the lower tool. The latter remains fix while the deformable piece takes the corresponding form accordingly. This can be another reason behind this anomaly.

To reenforce our argument, we can compare the penetration of the deformed material into the two contact surfaces in Figure 4.11 (the upper tool surface and the lower tool surface). More penetration is visible with the upper tool. This is not surprising since the lower tool is fixed during the whole simulation, the mesh is more stable resulting in less (to no) penetration.

To summarize, the overall behavior is acceptable but the mesh size was proven to have a crucial impact on the precision maintain during the simulation progress.

4.3 Multi-domain applications

In the following section, the method capacity is tested when using a great number of deformable bodies at once. Two major applications are proposed. The first handles 10 parallelepiped deformed simultaneously using both anisotropic and uniform meshes. The second stacks up to 256 deformable spheres in the computational domain. In both, computations are treated in a highly parallel environment.

4.3.1 Ten Parallelepiped

A first test handling more than two deformable bodies at once is presented hereby. In a computational domain $\Omega = [0; 100] \times [0; 100] \times [0; 51]\, mm^3$, we immerse 10 identical parallelepiped of dimensions $5\, mm \times 5\, mm \times 10 mm$ (see Figure 4.13). The deformable pieces are placed between two flat rigid tools. The lower tool remains fixed and the upper tool moves with a velocity equal to $-10\, mm/s$. The deformable pieces viscosity is equal to $353\, MPa.s$ while the rigid tools consistency is equal to $1000\, MPa.s$. Note that it is the same case mentioned in the General Introduction page 1 .

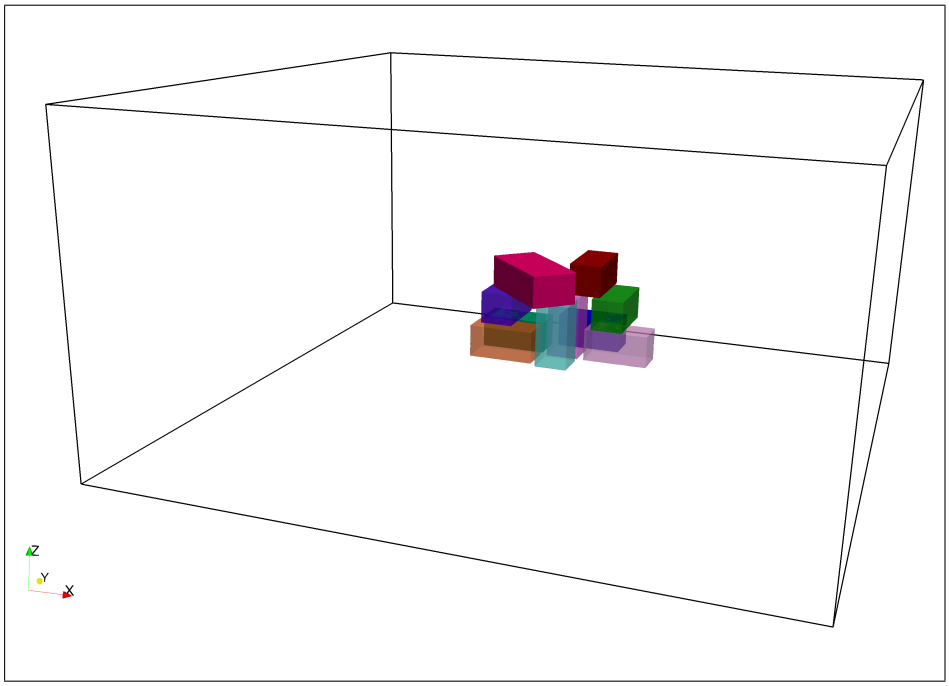


Figure 4.13: *Initial geometry of the* 10 *parallelepiped immersed in the computational domain.*

We chose to use a uniform mesh. Since the computational domain is relatively big in comparison with the parallelepiped dimension, we created a new virtual box en-globing the deformable bodies but is still smaller than the domain. Though the mesh is uniform, we impose two different mesh sizes h_{int} and h_{ext}. Inside the virtual box, the size mesh h_{int} is equal to $0.3\,mm$ whereas outside the box h_{ext} is equal to $2.5\,mm$ (see Figure 4.14). This choice enable us to use less mesh nodes and elements. The overall mesh contains 2 282 109 nodes and 13 445 963 elements. The simulation is launched on 256 cores. We should indicate as well that 6 different level set functions were defined and associated manually to these 10 deformable pieces.

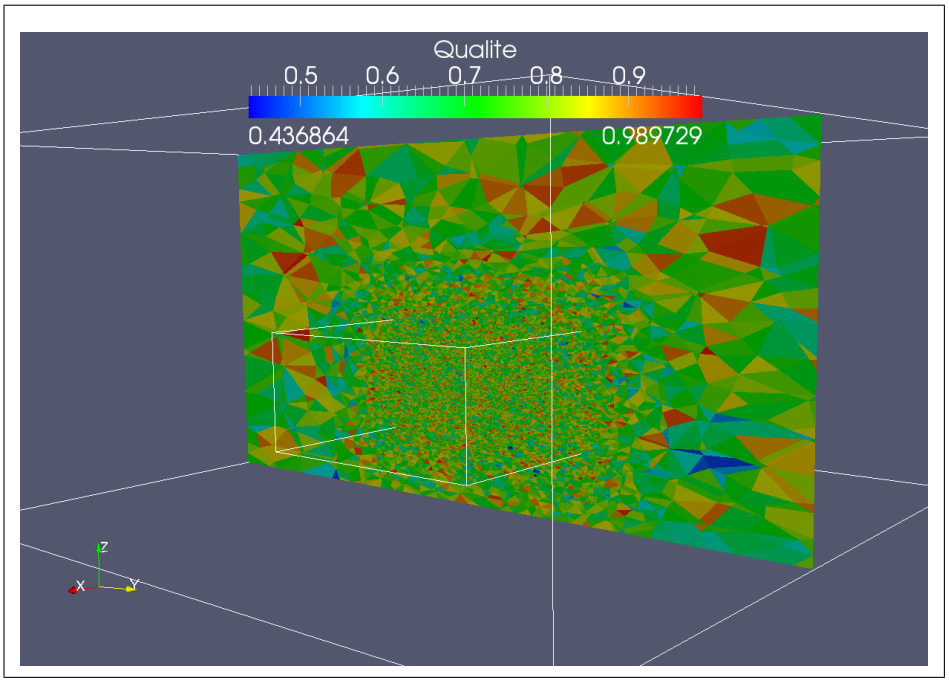


Figure 4.14: *The uniform mesh of the domain presented on a section: Inside the virtual box, the mesh size is almost ten times smaller than the external one.*

Figure 4.15 represents the deformed geometries at $t = .0.7\,s$ The simulation was carried out on 256 cores with no problem what so ever. Nonetheless, the results were found unsatisfactory. The pieces did not deform properly: their top remained intact during the whole simulation. When examined closely, this irregularity is associated to the mesh size h_{int}. Even though it is small, it remains significantly large when compared with the mixture law thickness $\approx 0.9\,mm$. If one decreases even more the mesh size, the problem will be prevented but it will be costly in computing time and the number of cores needed to run the simulation.

Although the results were inadequate for this particular case, we learned interesting information regarding the solvers performance. Considering the big number of unknowns ($> 8\,000\,000$) to be treated in this particular case, we were expecting an unrealistic number of iteration to insure the convergence. In fact, an average of 1 500 iterations were enough.

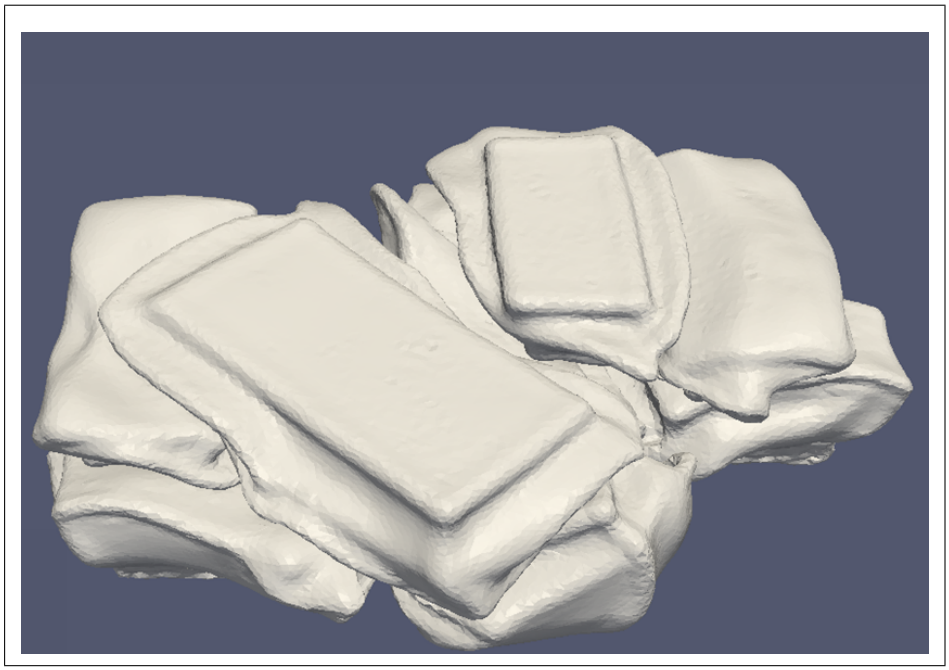


Figure 4.15: *The deformed geometries when using an isotropic mesh with a mixture law thickness equal to 0.9 mm.*

We were brought once again to use the anisotropic mesh generator. Leaving all the other components intact, only the mesh is changed. It is an anisotropic adaptive mesh formed by 147 500 nodes and 856 442 elements (illustrated in Figure 4.16). The simulation is now launched on only 16 cores. Remark that both the elements/nodes and the cores numbers were divided by 16.

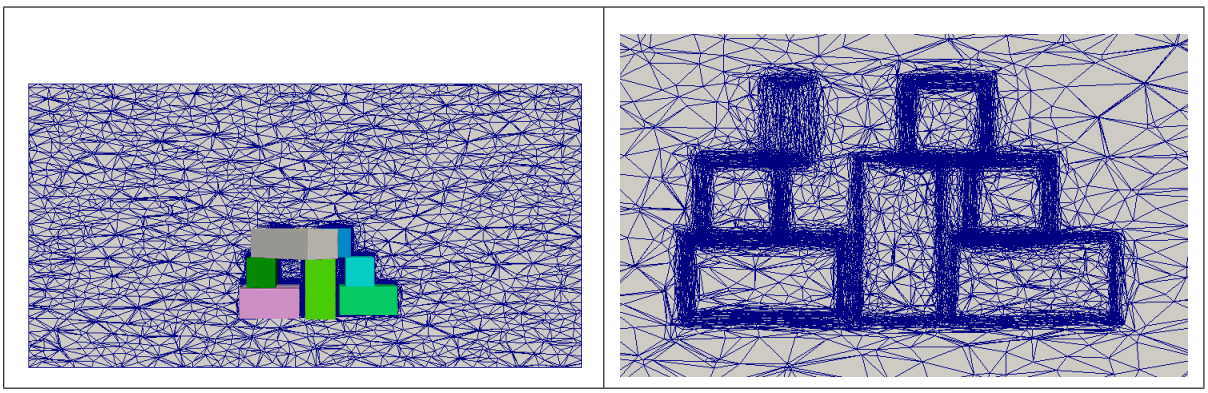


Figure 4.16: *The initial mesh adapted anisotropically around the deformable parallelepiped (on the left) with a zoom (on the right).*

Not only less nodes/elements and cores were needed for the exact simulation, but also the results were much better. Figure 4.17 completes the discussion. The deformable geometries are acceptable. In addition, the velocity components and pressure predict a good behavior. v_x and v_y are bigger on the edges whereas v_z is the highest on top. In

addition, $|v_z|$ takes the value of the upper tool velocity on top ($10\,mm/s$) and the lower tool on the bottom ($0\,mm/s$). The pressure anticipates an adequate performance: it takes the maximal value on top where the upper tool compresses the most.

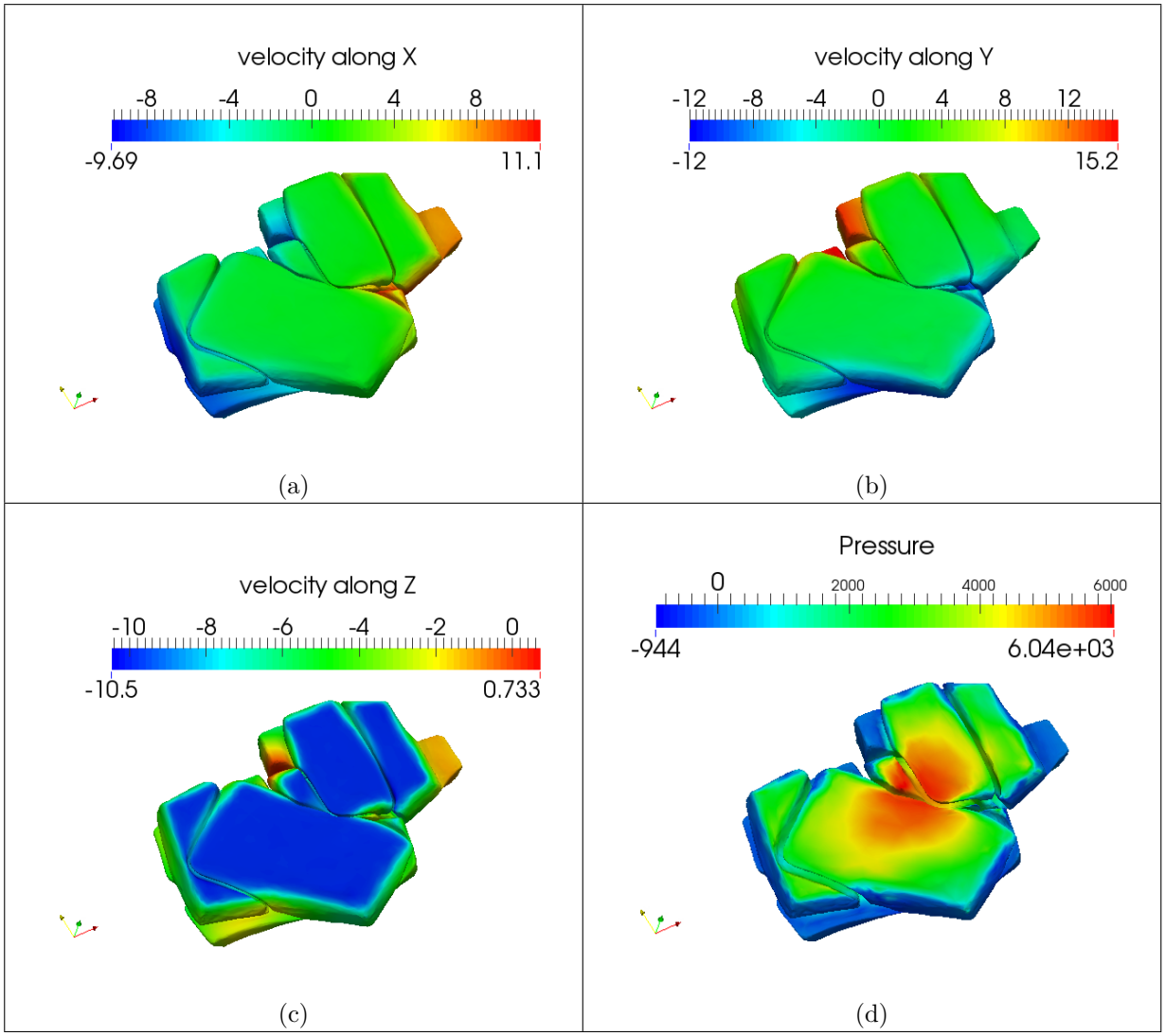


Figure 4.17: *The deformed parallelepiped using an anisotropic adapted mesh. (a), (b), (c) and (d) represent respectively the maps of the velocity components v_x, v_y, v_z and the pressure.*

Until now, we proved more than once that even though the overall performance of the approach is impressing, it remains extremely dependent of the mesh. It is an approach involving a big spectrum of parameters dependent one of the other (such as the mesh size, the thickness of the mixture law, ...).

In conclusion, important points should be retained from this section. In multi-domain problems and up to 2 000 000 nodes and 256 cores, no suspicious behavior in the solver was detected. However, the use of an adapted anisotropic mesh were found crucial to insure the precision of the simulation.

4.3.2 256 spheres

To increase even more the number of the deformable bodies, we begin by simulating a problematic case proposed by Transvalor. In both Forge and CIMLib, 148 spheres are stacked to form a cubic domain partitioned on 8 cores. Before imposing any mechanical problem, the partition is inspected closely. In Forge®, the partitioning algorithm is unable to converge and the tasks are not equally divided between the 8 cores. Figure 4.18 shows that the workload is almost handled by only one core which affects the scalability of the parallel computation (see chapter 3). Whereas, CIMLib confronted no difficulties partitioning the computational domain on 8 cores. As shown in Figure 4.20 , The workloads seem to be almost equally divided between the different cores. Note that the lack of smoothness is only due to the unrefined mesh used for this test.

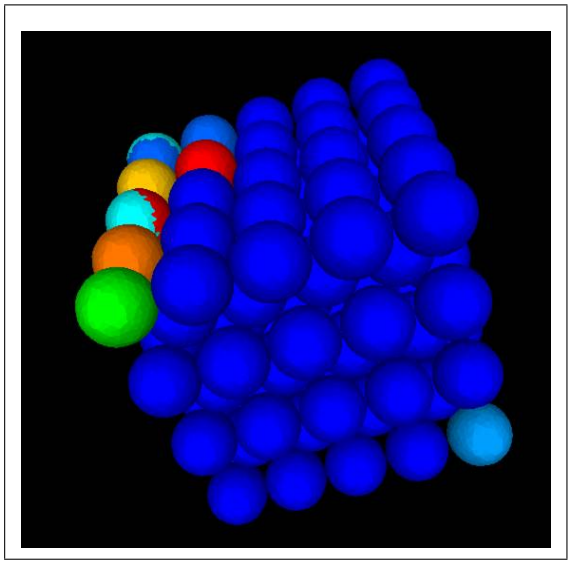


Figure 4.18: Forge® *failed attempt to partition the domain containing* 148 *spheres on* 8 *cores.*

As mentioned earlier, the dynamic load algorithm does not allow to exchange nodes between unneighbored domains (i.e. does not share a node).

In order to generalize the algorithm, an exception should be added for the initial partitioning when dealing with a sequential mesh. Thus, the exchange between two domains -not sharing an interface- is allowed only for the first step in the dynamic load algorithm. A node is chosen and is affected to an empty core then the classical algorithm is executed.

However, when considering a mesh formed by multiple connected geometries, the classical algorithm fails to partition properly the domain. A random node is chosen to begin with. Once the connected geometry containing this node is handled by the algorithm, it is unable to handle another one. This is logic since the core is no longer empty and the domain has no neighbors. Thus, the dynamic load algorithm stops even

though the imbalance remains.

To overcome this limitation, a simple yet effective modification is introduced. the core is allowed to handle new nodes when it has no neighbor and even if it is not empty. The latter modification allows to improve the imbalance shown in Figure 4.18 (see Figure 4.19)

Note that this is the best partitioning we can offer considering the fact that no geometrical notions are present in the algorithm.

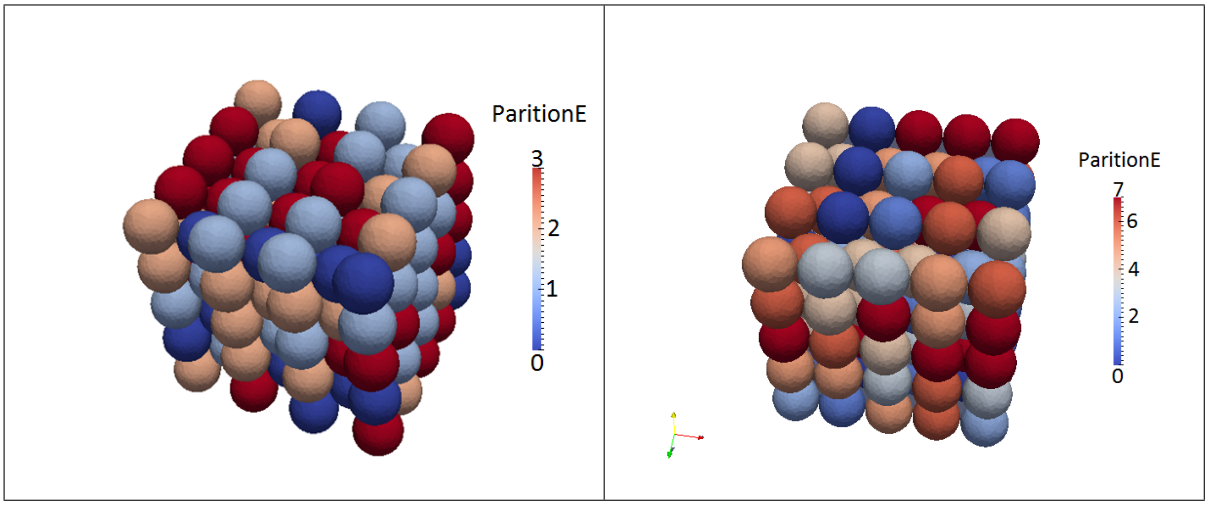


Figure 4.19: *From left to right: The partition of a domain containing* 148 *spheres in Forge® taking into account the above improvment using respectively* 4 *and* 8 *cores.*

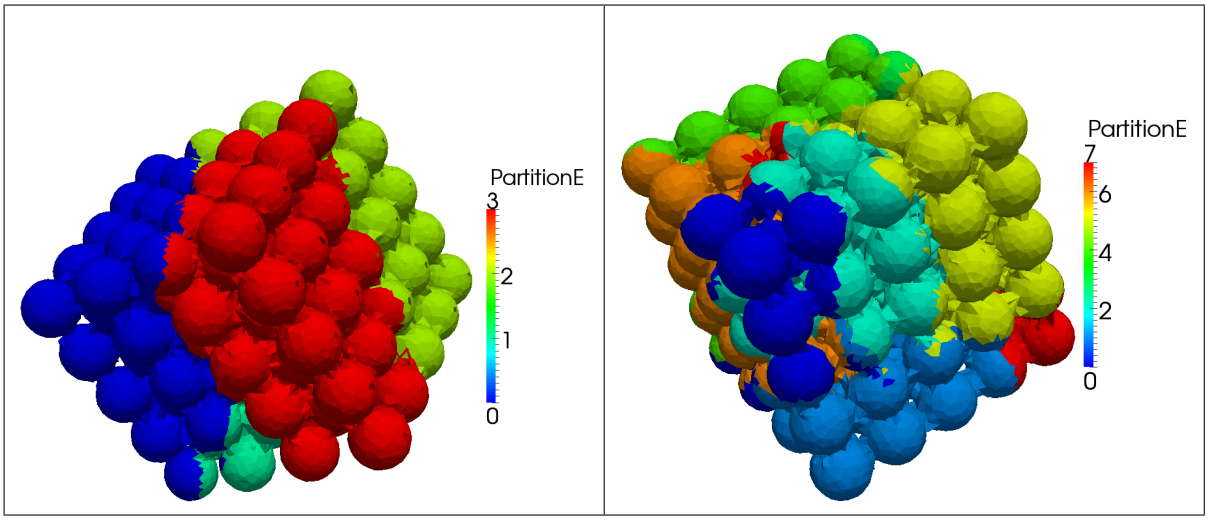


Figure 4.20: *From left to right: The partition of a domain containing* 148 *spheres in* CIMLib *using respectively* 4 *and* 8 *cores.*

Since CIMLib had no problems what so ever partitioning a domain containing 148 spheres on 8 cores, why not push this study further to test the limits of this approach.

To do so, 256 identical spheres of a radius equal to $0.1\,m$ are placed in a computational domain of dimension $[0;4]\times[0;4]\times[0;1.63]\,m^3$ to form a cube. Then, the spheres are placed between two rigid flat tools with a consistency of $10^6\,Pa.s$. The lower tool is fixed while the upper one moves with a velocity equal to $1\,m/s$. The whole computational domain is anisotropically meshed using 252 204 nodes and 1 440 718 elements. Since the deformable spheres will interact at one stage of the simulation, we cannot associate one level set for all the 256 spheres. Instead we should associate one level set for each deformable body. This can be hard to accomplish due to the high number of deformable bodies. To pull-off this step, we utilize an existent technique based on a coloration algorithm developed by [Hitti, 2011] . This coloring algorithm enable us to associate one level set to one or more deformable spheres with only one restriction. Two different spheres can share the same level set if and only if they never interact during the simulation. In other words, the same level set cannot be associated to neighbor spheres. Applying this technique, we were able to define only 40 level set functions to all 256 spheres. The spheres consistencies are chosen arbitrary between $10^5\,Pa.s$ and $7\times10^5\,Pa.s$. The simulation is launched on 64 cores. Figure 4.21 illustrates the whole domain partition. Again, CIMLib confronted no difficulties dividing the workloads properly on 64 cores.

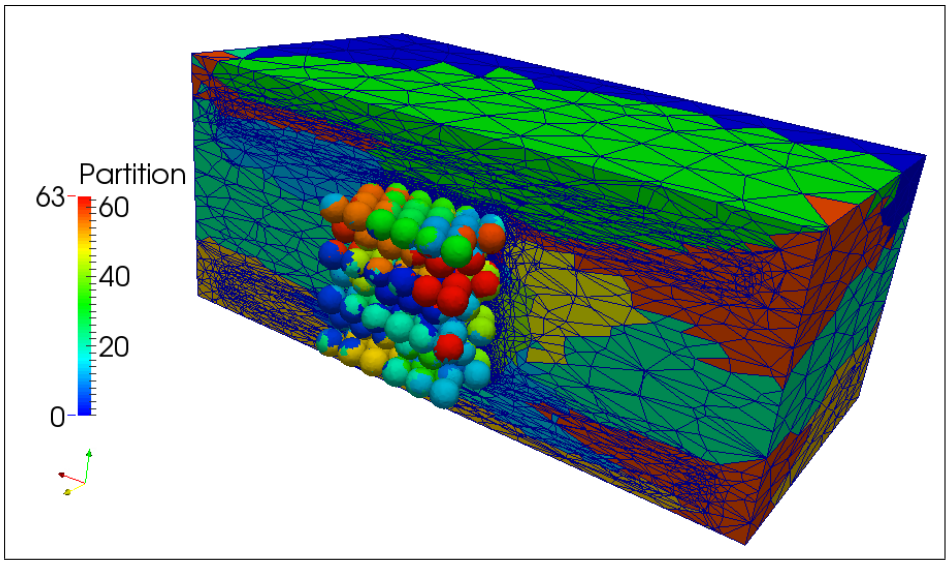


Figure 4.21: *Initial partition of a domain containing* 256 *spheres on* 64 *cores.*

Note that we could of used less level set functions. In-spite of this fact, we chose to define 40 functions to test the limits of the program . We expected to confront a memory clash but we did not. For a great number of level sets, the resulting data will be normally overwhelming and could lead to a lack of memory space. In this case, we recommend to compress the unused data to increase the remaining space. Once the data is needed, it can be uncompressed. This can be a solution when significant number of deformable pieces is needed in future works.

Finally, the deformed domain and the different velocity components are illustrated in Figure 4.22 at $t = 0.64\,s$. We notice that every sphere deforms differently inside the domain (due to their different characteristics) and interacts with neighbor spheres to fill properly the gaps. v_x and v_z are represented on a longitudinal section in the thickness, while v_y is represented on a longitudinal section in the width. These maps portray a good behavior. $|v_x|$ and $|v_y|$ are bigger on the edges and are almost symmetrical suggesting the direction of the material flow. $|v_z|$ is bigger on the top and takes the value of the moving upper tool.

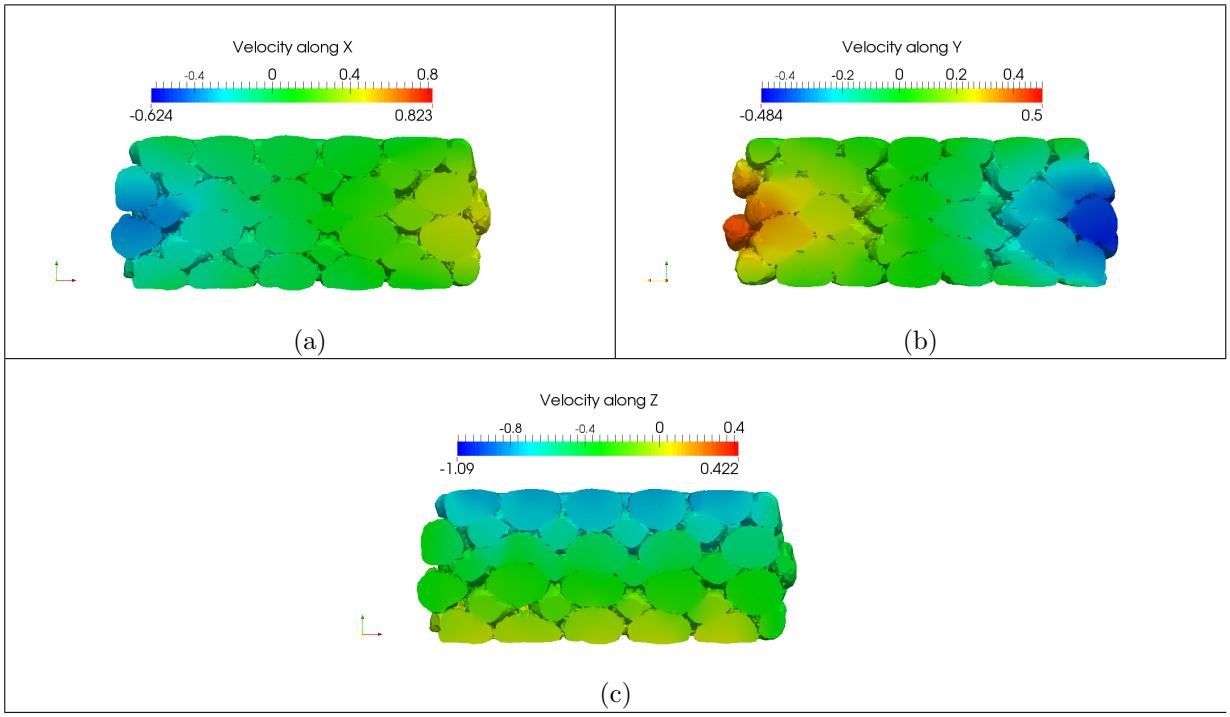


Figure 4.22: *Velocity components v_x and v_z are respectively represented on a longitudinal section in the thickness in (a) and (c). Velocity component v_y represented on a longitudinal section in the width in (b) ($t = 0.64\,s$).*

The Eulerian monolithic approach was proven more resilient than *Forge*® when undertaking multi-domain problems containing significant number of deformable bodies. In particular, the partitioning algorithm was found more efficient. In addition, the results issued by the mechanical problem give a good description of the deformation evolution with time.

4.4 Conclusions and Discussions

This chapter is entirely dedicated to numerical applications using the monolithic Eulerian approach along with the proposed improvements (described in chapter 2). All

applications were carried out in a scalable parallel environment on at least 16 cores. The efficiency of the method was studied in more than one context using complex geometries (such as crankshaft and connecting rod) or simple deformable geometries stacked in great numbers (up to 256 spheres).

The results were judged via comparison with simulations launched in Forge and found satisfactory. The overall performance of this approach was solid. It was even considered more resilient than *Forge®* in some cases; in particular when undertaking multi-domain problems containing significant number of deformable bodies.

Although the performance is generally impressive, some points can be improved even more. The approach itself involve a big spectrum of parameters dependent one of the other that need to be mastered (such as the mesh size, the thickness of the mixture law, ...). In more than one occasion, we proved that the method precision remains extremely dependent of the mesh. When dealing with evolving geometries exhibiting details more visible with time, the mesh felt short and problems such penetration were detected. In addition, when comparing with Forge, we noticed that our results exhibited less precision in the center than on the edge. It is mainly because more nodes are distributed on the edge respecting the level set gradient.

Following these observations several upgrades/enhancement can be proposed:

- An increasing number of nodes can be used as the simulation progress. It can be determine using a relationship between the geometry surface and the mesh size. This can easily solve the problem of penetration when using complex deformable geometries.

- The mesh adaptation can be based not only on the level set gradient but on additional fields such as the velocity. This can improve the precision in the center (for instance) to match the outer precision depending of the application type.

To this stage, this approach was proven effective. Furthermore, the superposition with *Forge®* was reassuring: two different softwares based on different approaches provided not only promising results but extremely comparable ones as well. The fact that a recent approach like the Eulerian one can compete with Forge -a well known software perfected for at least 20 years- drives us to enrich the method and search for new answers. Hence, new ways to model friction and contact in an Eulerian context are proposed and tested in the next chapter.

Résumé en Français

Ce chapitre est entièrement dédié aux applications numériques utilisant l'approche Eulérienne monolithique ainsi que les améliorations proposées dans le chapitre 2. Toutes les applications ont été réalisées dans un environnement parallèle utilisant au moins 16 cœurs. L'efficacité de la méthode a été testée dans plusieurs contextes en utilisant des géométries complexes (telles que le vilebrequin et la bielle) ou des géométries simples empilées en grand nombre (jusqu'à 256 sphères par exemple). Les résultats ont été jugés par comparaison avec des simulations lancées en Forge et trouvés satisfaisants. La performance globale de cette approche est solide. Elle est même considérée plus pertinente que Forge® dans certains cas; en particulier dans les cas multi-domaines avec un grand nombre de corps déformables. Bien que la performance soit généralement impressionnante, certains points peuvent être améliorés. L'approche elle-même implique un grand spectre de paramètres dépendants l'un de l'autre, qui doivent être maîtrisés (comme la taille de la maille, l'épaisseur de la loi de mélange, ...). A plusieurs reprises, nous avons prouvé que la précision de la méthode reste extrêmement dépendante de la taille de maille. En traitant l'évolution d'une géométrie présentant de plus en plus de détails au cours du temps, le maillage ne semble pas entièrement satisfaisant et entraine des problèmes de pénétration par exemple. De plus, suite à des comparaisons avec Forge pour ces cas-là, nous avons remarqué que nos résultats présentent moins de précision au centre que sur les bords. C'est principalement dû à la distribution des nœuds plutôt localisés au bord de la pièce déformable. Suite à ces observations plusieurs améliorations peuvent être proposées: • Un nombre croissant de nœuds peut être utilisé vis-à-vis l'avancement de la simulation. Ça peut être déterminé en utilisant une relation entre la surface de la géométrie et le maillage. Cela peut facilement résoudre le problème de pénétration entre des pièces déformables complexes et les outils. • L'adaptation du maillage peut se baser non seulement sur le gradient de la Level Set mais sur des champs supplémentaires tels que la vitesse. Cela certainement améliorera la précision des solutions au centre de la pièce.

Pour le moment, cette approche a été prouvée efficace. En outre, la superposition avec Forge® est rassurante: deux logiciels différents basés sur différentes approches ne fournissent pas seulement des résultats prometteurs, mais très comparables aussi. Le fait que l'approche Eulérienne, une approche relativement nouvelle pour telles applications, puisse rivaliser avec Forge nous motive à enrichir la méthode même plus. Par conséquent, de nouvelles façons à modéliser le frottement et le contact dans un contexte Eulérien sont proposées et testées dans le chapitre suivant.

Bibliography

[Hitti, 2011] Hitti, K. (2011). *Direct numerical simulation of complex Representative Volume Elements (RVEs): Generation, Resolution and Homogenization.* Theses, École Nationale Supérieure des Mines de Paris. Co-encadrement de la thèse : Marc Bernacki.

CHAPTER 5

Friction and contact

Contents

5.1 Introduction 146

5.2 Friction: from Lagrangian to Eulerian description 146

5.2.1 Lagrangian Friction law 147

5.2.2 Eulerian friction law 148

5.2.3 Friction validation 149

5.3 Directional Solver 157

5.3.1 Weak formulation 159

5.3.2 Spatial discretization 159

5.3.3 Variational Multi-scale Method 162

5.3.4 Applications 163

5.3.4.1 Free flow 163

5.3.4.2 Towards contact modeling 165

5.3.4.3 Contact 167

5.4 Conclusion 169

Résumé en Français 170

Bibliography 171

5.1 Introduction

Friction and contact are two important features in every large deformation problem. As pointed several times through out our study, Eulerian approaches are rarely used in this context. Thus, a lack of information handling this topic is detected in the literature. This chapter offers first steps towards friction and contact modeling in an Eulerian environment.

The chapter is mainly divided into two parts. The first part is dedicated to friction modeling. We propose different manners to numerically imitate friction depending on the simulation. A linear mixture law emulates a perfectly sticking friction, whereas a quadratic mixture law introduces, via a boundary layer, a lubricant to simulate sliding friction. The latter requires additional manipulation to determine the appropriate lubricant consistency for each case. An identification technique superposing both Lagrangian and Eulerian formulations is used. For each friction coefficient in the Lagrangian approach corresponds a lubricant consistency spread along a certain thickness. Followed the technique description, 3D test cases are launched and results are validated with Forge® simulations.

Recalling the literature survey in chapter 1, the contact detection algorithm affect enormously the parallel scalability of the code. In Eulerian approaches, contact detection is managed automatically. Still, no way to model different types of contacts is available. To our knowledge, there are very few works in the literature dealing with this topic. The only work found is proposed by [Bruchon et al., 2009]. In the second part of this chapter, a daring proposition is presented to model the contact problem. Searching for inspiration in other domains, we decided to apply the same reasoning used in geology and oceanography. A directional solver is a great candidate and may hold the answers to our needs. The formulation of the anisotropic solver is detailed then followed by different applications. Its capacity is assessed ending in a small discussion and conclusions.

5.2 Friction: from Lagrangian to Eulerian description

It is important to model friction in large deformation problems. For instance in forging process, friction leads to heat and tools wear. Thus, it can affect the final shape and quality of the product.

In reality, to control the effects of friction, lubrication techniques are used. Generally in an Eulerian context, modeling surface forces (such as friction) consists on crossing from surface to volume integral using Dirac function [Bruchon et al., 2009] (see chapter 1).

In this work, a simpler approach based on boundary layer is used. A new viscosity corresponding to the lubricant is introduced. To determine this appropriate viscosity, a

similar approach to [Foudrinier, 2007] and [Mondalek, 2012] is applied hereby. The first worked on reactive molding processes. Whereas the work presented in [Mondalek, 2012] treats modeling of spark plasma sintering process.

Based on a Lagrangian description of friction, this approach relies on finding a new formula describing friction in the Eulerian context. The detailed methodology is presented next.

5.2.1 Lagrangian Friction law

For every multi-domain problem, the physical phenomena in the contact zone is modeled using friction laws. Friction can be described as the force acting tangentially along surfaces in contact and resisting the motion of solid surfaces.

For instance, in a case containing a tool and a deformable body in contact. Let us denote respectively v_t and v the velocities of the tool and the deformable body. The sliding velocity v_s is computed in terms of v and v_{tool}:

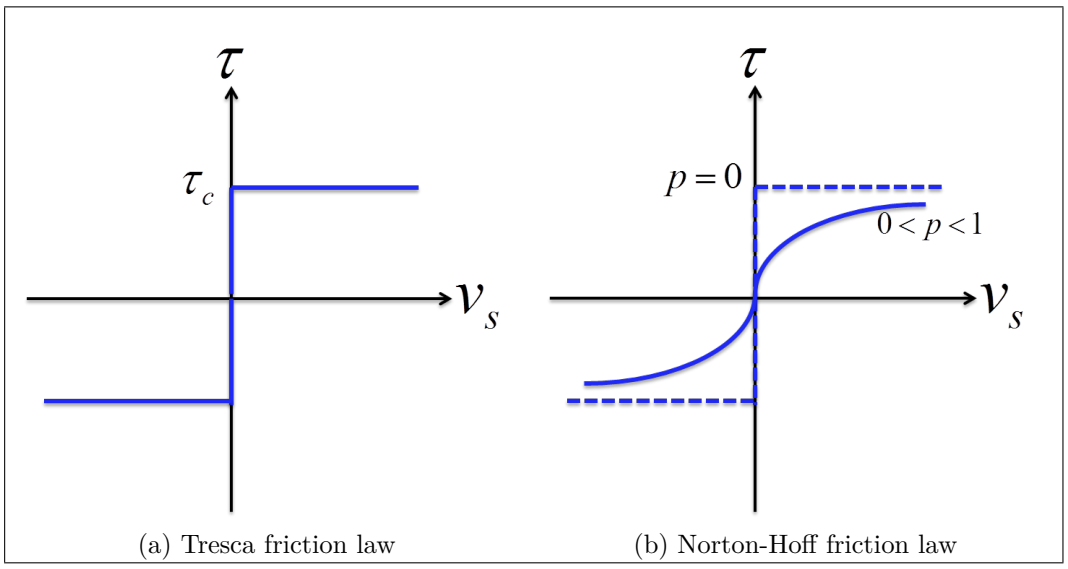
$$\vec{v}_s = (\vec{v} - \vec{v}_{tool}) - [(\vec{v} - \vec{v}_{tool}) .\vec{n}]\, \vec{n} \tag{5.1}$$

where $\vec{n}$ is the normal to the surface.

In addition, the shear stress is given by:

$$\vec{\tau} = \sigma\vec{n} - (\sigma\vec{n}.\vec{n})\, \vec{n} \tag{5.2}$$

In literature, numerous behavior laws are introduced. These laws describe the relation between the sliding velocity v_s and $\vec{\tau}$ and are divided into two families.



(a) Tresca friction law (b) Norton-Hoff friction law

Figure 5.1: *Graphic representation of the different friction laws.*

The first family is based on a threshold concept such as in Tresca and Coulomb friction laws (Figure 5.1a); where below this threshold no sliding can occurs (equation (5.3)). For Coulomb law, the threshold τ_c is proportional to the normal stress σ_n.

$$\begin{cases} \tau < \tau_c & no\ slip \\ \tau = \tau_c & slip \end{cases} \tag{5.3}$$

The second family includes regularized law such as Norton-Hoff law. In our study, Norton-Hoff friction law is used due to its continuity and regularized form (Figure 5.1b). Using the latter law, the relation between the shear stress τ and the sliding velocity v_s is expressed by:

$$\vec{\tau} = -\alpha_f \eta . |v_s|^{p-1} \vec{v}_s \tag{5.4}$$

where η is the material consistency and α_f the friction coefficient (chosen depending on the material). p can be considered equal to m the strain rate sensitivity coefficient since the lubricant is not considered as a third body.

5.2.2 Eulerian friction law

To apply the boundary layer approach, we begin by explaining its concept. We consider the existence of a boundary layer located between the two bodies in contact (tool/deformable body). This new layer represents the lubricant with its own viscosity (see Figure 5.2).

Unlike other works [Mondalek, 2012], where the boundary layer is explicitly represented, this viscosity once determined helps representing implicitly the boundary layer via quadratic mixture law presented in chapter 2.

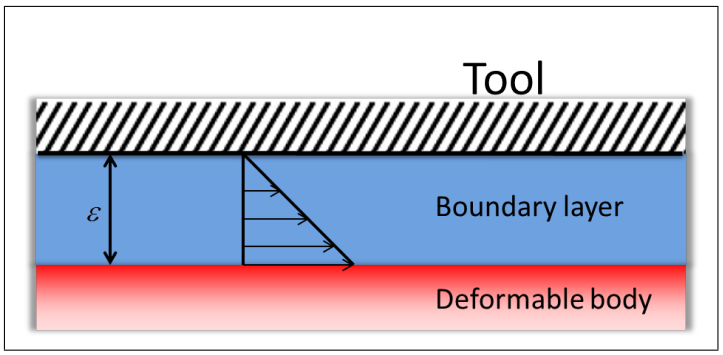


Figure 5.2: *Schematic representing the boundary layer.*

To proceed in determining the layer viscosity, we must first express the shear stress in the Eulerian context. Then by identification, the viscosity is determined from the Lagrangian law (equation 5.4).

In the boundary layer, the shear rate is written as :

$$\dot{\gamma} = \frac{|\vec{v}_s|}{\varepsilon} \tag{5.5}$$

where $\vec{v}_s$ is the tangential velocity defined in equation (5.1) and ε is the thickness of the boundary layer (see Figure 5.2). Evaluating the strain rate tensor in the latter layer, it can expressed as :

$$\dot{\varepsilon}(v) = \begin{bmatrix} 0 & \frac{v_s}{2\varepsilon} & 0 \\ \frac{v_s}{2\varepsilon} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.6}$$

Thus the equivalent strain rate tensor is computed in the following manner :

$$\begin{aligned} \dot{\bar{\varepsilon}}(v) &= \sqrt{\left(\frac{2}{3}\dot{\varepsilon}(v) : \dot{\varepsilon}(v)\right)} \\ &= \frac{|\vec{v}_s|}{\varepsilon\sqrt{3}} \end{aligned} \tag{5.7}$$

Using the definition of the shear stress given in equation (5.2) and substituting $\dot{\varepsilon}$ and $\dot{\bar{\varepsilon}}$ by their new form (5.6) and (5.7) computed in the boundary layer, the shear stress is expressed as :

$$\begin{aligned} \tau &= -2\eta_{lub}.\left(\frac{\left|\sqrt{3}\vec{v}_s\right|}{\varepsilon\sqrt{3}}\right)^{m-1}\left(\frac{\vec{v}_s}{2\varepsilon}\right) \\ &= -\frac{\eta_{lub}}{\varepsilon^m}.|\vec{v}_s|^{m-1}\vec{v}_s \end{aligned} \tag{5.8}$$

where η_{lub} represents the lubricant viscosity.

Both equations (5.4) and (5.8), describe the form of the shear stress. One in a Lagrangian formulation and the other in an Eulerian one. These formulations must be equivalent and it is ensured via an identification technique :

$$\eta_{lub} = \alpha_f \eta \varepsilon^m \tag{5.9}$$

The layer viscosity η_{lub} depend highly on the chosen thickness and the friction coefficient. η_{lub} will be finally transferred into equation (2.76) (see chapter 2) to represent this heterogeneity in the monolithic approach.

5.2.3 Friction validation

In large deformation problems, especially the ones involving multiple materials in contact, friction is considered an important component. It interferes in the deformation

evolution and affects the final shape of the piece (in this case the ring). Therefore this section illustrates the numerical results of friction in an Eulerian formulation as presented in the previous description. In the computational domain $[0; 75] \times [-75; 75] \times [0; 55]\, mm^3$, we immersed a perfectly symmetrical ring illustrated in Figure 5.3 between two rigid tools. The inner and outer radius denoted respectively r_{int} and r_{ext} are equal to $40\, mm$ and $50\, mm$. The half-height h is equal to $40\, mm$. Since the ring is symmetrical, only one fourth of the piece is deformed to decrease the computational time. Imposing a symmetry is equivalent to verifying the condition $\sigma.\vec{n} = 0$. The mesh of a quarter ring contains 39 000 nodes. Note that the lower tool is fixed while the upper tool moves with a velocity of $-5\, mm/s$. The rigid tools consistency η_r and density ρ_r are respectively equal to $1\,000\, MPa.s$ and $8\,000\, kg/m^3$. As for the deformable body, η_d and ρ_d are equal to $66\, MPa.s$ and $2\,800\, kg/m^3$. The air viscosity and density are respectively fixed to $10^{-7}\, MPa.s$ and $1.2\, kg/m^3$.

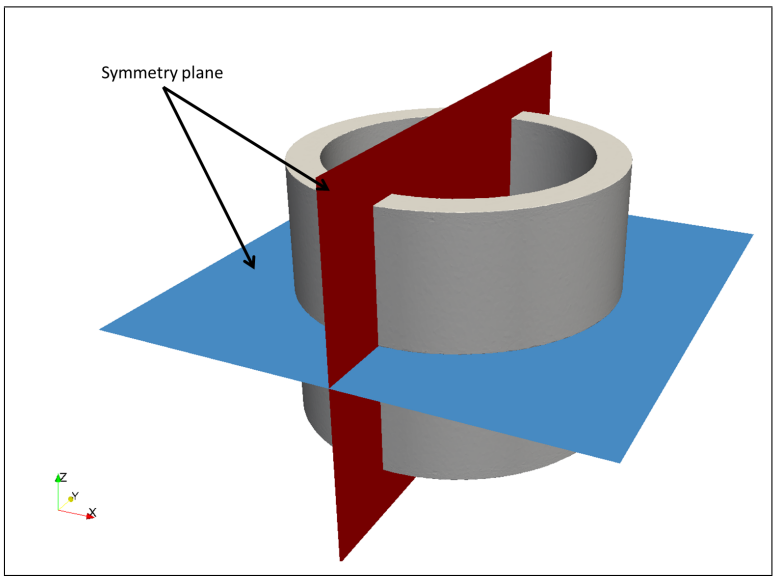


Figure 5.3: An illustration of the initial ring geometry along with the symmetry planes .

Different scenarios demonstrating how to represent friction in this Eulerian description are considered. We start with a perfectly sticking friction and finish by a sliding friction. The results are superposed with *Forge*® simulations. Both advantages and limitations are highlighted before ending with a small discussion.

To begin with, we impose a perfectly sticking friction. In our approach this can be accomplished by using a linear mixture law illustrating the different domains. In fact, since the ratio η_r/η_d exceeds 100, the interface consistency is closer to the rigid body consistency. Therefore, it can be considered as an excessive numerical friction or "sticking friction". The latter is illustrated on a cutting plane in the thickness in Figure 5.5a.

The evolution of the ring in time is followed. Figure 5.4 illustrates the deformed

ring at $t = 3.467\,s$. The half-height of the deformable ring is equal to $22.7\,mm$. The inner and outer radius for $h = 0$ are approximately equal to $54.89\,mm$ and $66.34\,mm$. The velocities and pressure maps are presented as well for $t = 3.467\,s$ in Figure 5.5. The overall behavior seems satisfactory. The velocity component v_z is equal to $-5\,mm/s$ on the top conform with the upper tools velocity. The velocity components v_x and v_y take a maximum of approximately $4.77\,mm/s$ on the lower edge, which suggests that the deformation is symmetric as expected.



Figure 5.4: The deformed ring geometry at $t = 3.467\,s$.

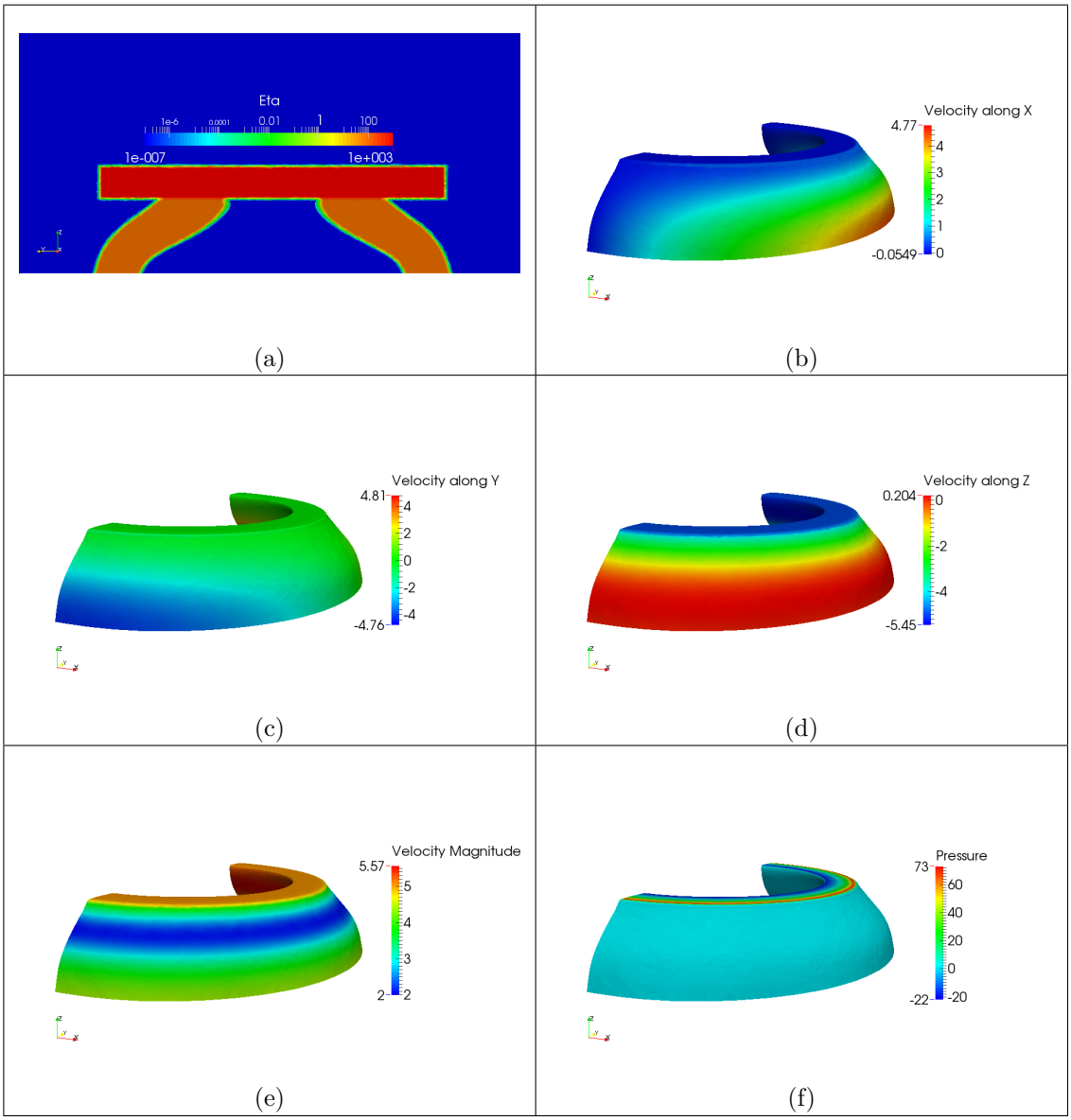


Figure 5.5: *(a) illustrates the mixture law on a cutting plane in the thickness. (b) to (e) represent the different velocity components and magnitude. (f) illustrates the pressure repartition. The different maps are presented on a forth of the ring at* $t = 3.46\,s$.

To validate the results, the same simulation is launched in Forge taking a sticking friction. Since the deformable geometry is hollow inside, we did not plot the curve on a vertical section. Instead, we took a cutting plane in the thickness for $y = 0$ in both interfaces (Paraview for CIMLib and Glview for *Forge*®).

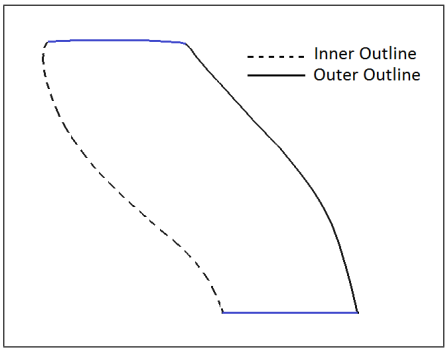


Figure 5.6: The deformed ring outline *for* $y = 0$ at $t = 3.467\,s$.

The chosen nodes corresponds to the inner (dashed line) or outer (solid line) outline as illustrated in Figure 5.6. All curves are plotted as functions of the vertical distance z $(0 \leq z \leq h)$. The different results are superposed in Figure 5.7.

Important information can be extracted. First, by analyzing Figures 5.7a and 5.7b we notice that the velocities v_x plotted on the outer outline in both softwares are globally very similar. Still, decreasing the mixture law thickness from $\varepsilon = 0.5\,mm$ to $\varepsilon = 0.25\,mm$ affects locally our results to fit better the Forge simulation on the top $(z = h)$. Since this contact region is of a great importance, the parameter ε should be mastered to find the optimal results. We should mention that decreasing the mixture law thickness is not that simple. Smaller the chosen value is, more difficulties will be confronted to mesh this region. In other words, more stretched elements will be generated and smaller time steps will be needed. The end game is to find a thickness value ε small enough to increase the precision but big enough to prevent meshing difficulties. We should mention as well that small thickness values ε can not represent the boundary layer when treating sliding friction. This topic will be treated in details in the following cases.

Using the smaller mixture law thickness $\varepsilon = 0.25\,mm$, v_z and v_x are respectively plotted on the outer and inner outlines. Once again the curves are similar and the average error does not exceed $2\,\%$. This slight difference can be attributed to the different used meshes as well as the different interpolations used by Glview and Paraview.

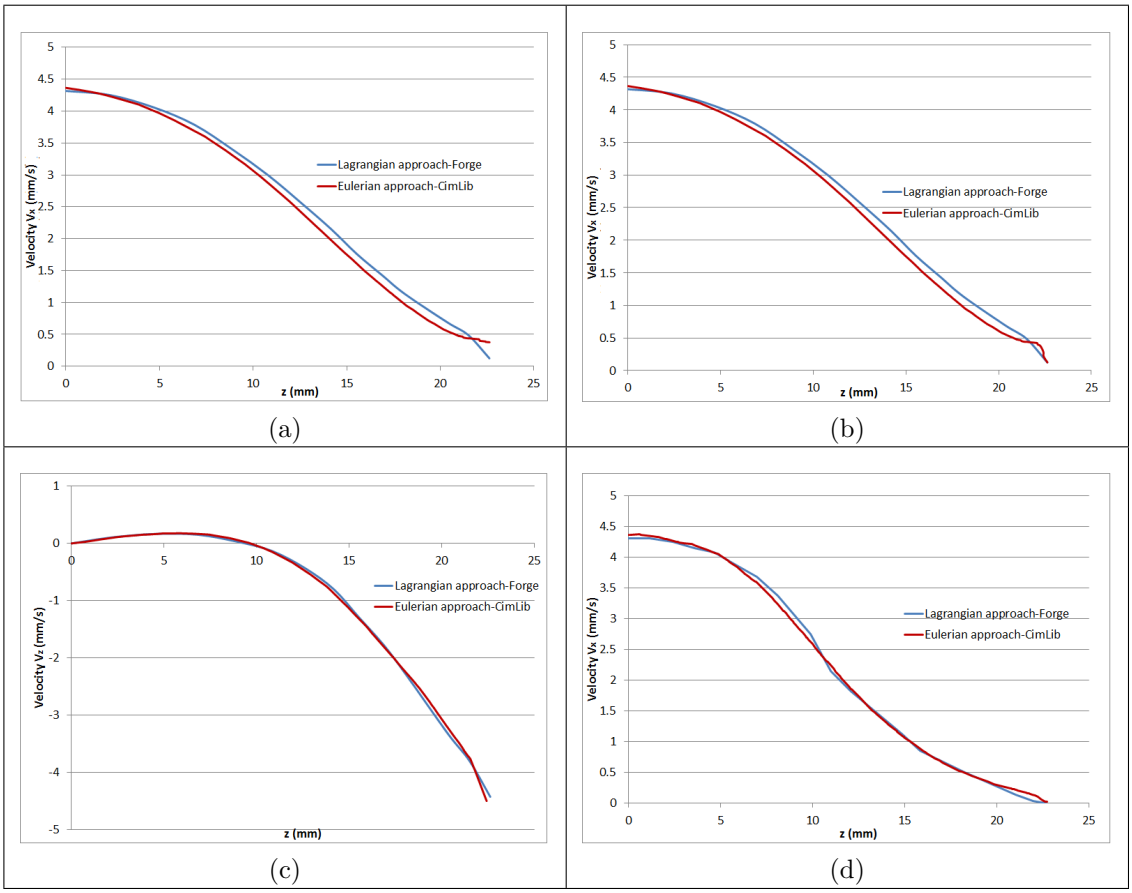


Figure 5.7: *Comparison of the velocities obtained with sticking friction in both Forge® and* CIMLib *on a cutting plane in the thickness for* $y = 0$ *at* $t = 3.46\,s$ *: (a) and (b) compare the velocity* v_x *plotted on the outer outline using two different mixture thickness* $\varepsilon = 0.5\,mm$ *(on the left) and* $\varepsilon = 0.25\,mm$ *(on the right). (c) compares the velocity* v_z *plotted on the outer outline using a mixture thickness* $\varepsilon = 0.25\,mm$*. (d) superposes the velocity* v_x *plotted on the inner outline of the ring using* $\varepsilon = 0.25\,mm$*.*

The second part of this study is dedicated to sliding friction. As mentioned earlier, "sliding friction" is equivalent to using the quadratic mixture law introduced in equation (2.76) (chapter 2). In other words an additional phase is added. It is called the boundary layer and it represents the presence of a lubricant with η_{lub} (Eulerian formulation). The consistency of this lubricant is chosen to verify equation (5.9). To be more precise, the couple $(\eta_{lub}, \varepsilon)$ should be chosen to give the same effects as the coefficient α_f in the Norton-Hoff friction Law (Lagrangian formulation). For instance, if we need to simulate a case with sticking friction using $\alpha_f = 0.3$ in a Lagrangian approach, for a fixed mixture law thickness $\varepsilon = 0.25\,mm$ we should use $\eta_{lub} = 4.95\,MPa.s$ in our monolithic Eulerian approach. This means that the couple $(\eta_{lub}, \varepsilon) = (4.95, 0.25)$ in CIMLib should give the same results as when using $\alpha_f = 0.3$ in *Forge®*.

The curves are plotted in the same manners as before. The results are compared in Figure 5.8. Mainly, the curves show good agreement in both softwares. The overall

average error does not exceed 4%.

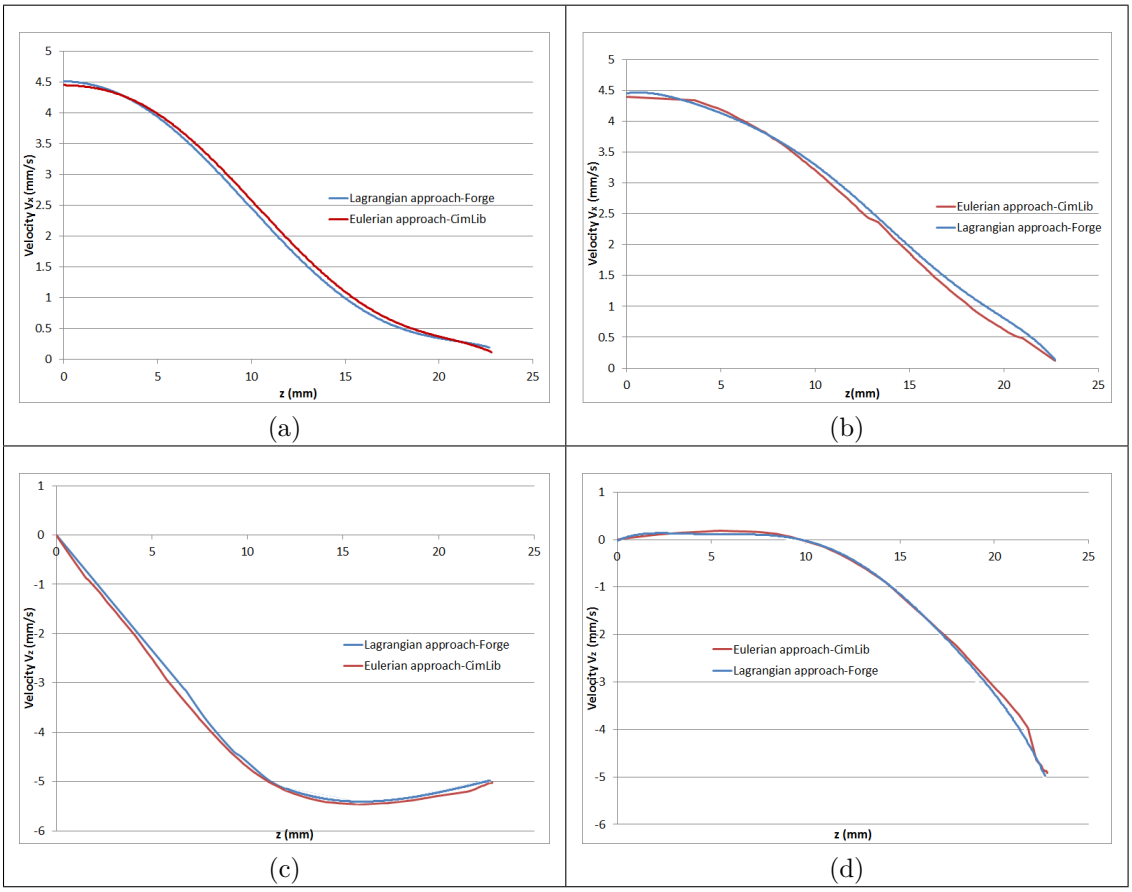
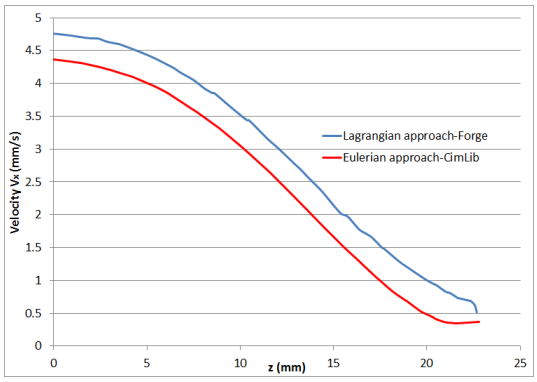


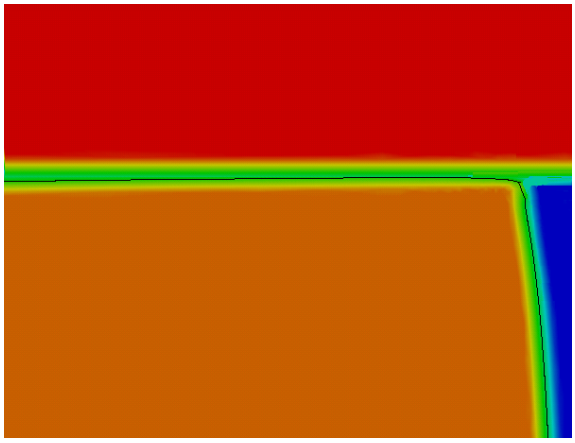
Figure 5.8: *Comparison of the velocities obtained with sliding friction in both Forge® and* CIMLib. *Curves are plotted at* $t = 3.46\,s$ *for* $\alpha_f = 0.3$ *on a cutting plane in the thickness for* $y = 0$ *: (a) and (b) compare the velocity* v_x *respectively plotted on the inner and outer outline using a mixture thickness* $\varepsilon = 0.25\,mm$*. (c) and (d) compares the velocity* v_z *respectively plotted on the inner and outer outline using a mixture thickness* $\varepsilon = 0.25\,mm$*.*

To assess even more the capacity of this Eulerian method to impose sliding friction, we attempt to model the same case by decreasing drastically α_f to 0.03. For the same thickness $\varepsilon = 0.25\,mm$, the corresponding η_{lub} is now equal to $0.495\,MPa.s$. On the same cutting plane, the results were inspected. In Figure 5.9(a), the superposition of curves plotted in both softwares reveals this time a relatively big gap between the results. They seem almost parallel one to another. The velocity v_x obtained in CIMLib is almost exactly the same found for sticking friction. When examining in details the cause of this difference, we noticed that for this particular α_f, η_{lub} =0.495 $MPa.s$ is very small in comparison with $\eta_d = 66\,MPa.s$. And since we are using a small thickness mixture equal to $0.25\,mm$, the exact value of η_{lub} was not reached in the boundary layer. Actually, the latter is almost non-existent as illustrated in Figure 5.9(b). This can be solved by increasing the mixture thickness to $\varepsilon = 0.5\,mm$ for instance. The curves are compared

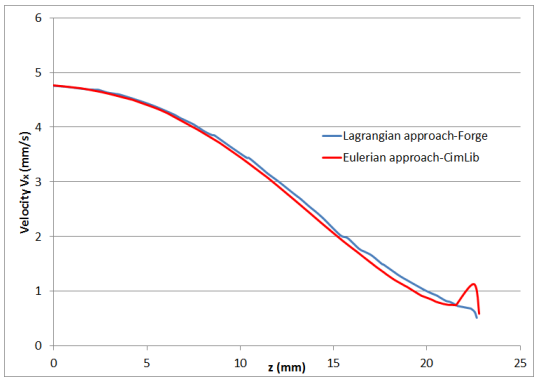
again in Figure 5.9(c). The curves fit almost perfectly except on the edge where an odd hill is visible in the vicinity of $z = h$. Figure 5.9(d) presents the reason behind this bump: even though the boundary layer is perfectly represented, a miss-functioning is detected on the upper corner. Since η_{lub} =0.99 $MPa.s$ (corresponding to the new mixture thickness 0.5 mm) is very small in comparison with $\eta_d = 66\, MPa.s$. The resulting viscosity is much smaller than η_d as well and is translated in bigger displacement on the top (for $z = h$).



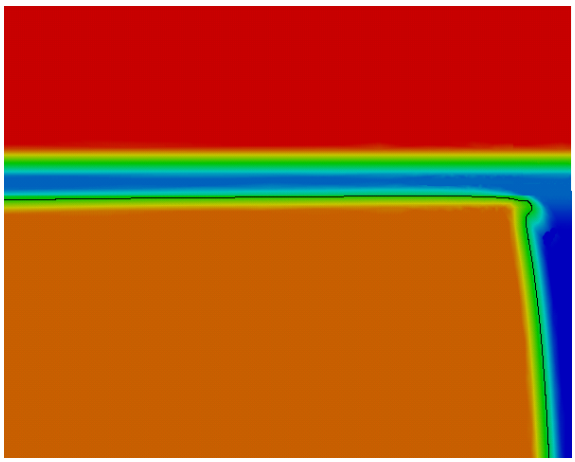
(a) *using a mixture thickness* $\varepsilon = 0.25\, mm$.



(b) *using a mixture thickness* $\varepsilon = 0.25\, mm$.



(c) *using a mixture thickness* $\varepsilon = 0.5\, mm$.



(d) *using a mixture thickness* $\varepsilon = 0.5\, mm$.

Figure 5.9: *(a) and (c) present a comparison of the velocities* v_x *obtained with sliding friction in both Forge® and* CIMLib *using respectively* $\varepsilon = 0.25\, mm$ *and* $\varepsilon = 0.5\, mm$ *. Curves are plotted at* $t = 3.46\, s$ *for* $\alpha_f = 0.03$ *on the outer outline of a cutting plane in the thickness for* $y = 0$*. (b) and (d) present a zoom of the quadratic mixture law using respectively* $\varepsilon = 0.25\, mm$ *and* $\varepsilon = 0.5\, mm$ *.*

Following the last analysis, a major limitation was noticed. The proposed method gives altogether great results for the chosen α_f values . Nevertheless, it cannot be generalized for the whole range of α_f (respectively $(\eta_{lub}, \varepsilon)$) especially for smaller values. The mixture thickness ε was proven to be a key-parameter affecting significantly the results.

It should be big enough to represent properly the boundary layer but remains insufficient for very small values of α_f.

To conclude, we can say that the boundary layer should be a good candidate to simulate fiction in an Eulerian framework. However, a proper rpresentation of the boundary layer was proven to be a delicate matter as shown in Figure 5.9.

Next, we present a directional solver attempting to model bilateral sliding friction.

5.3 Directional Solver

To model the contact problem, we propose in this section developing an anisotropic solver. A long reasoning was behind this choice. Two types of contact are usually available: bilateral and unilateral contact. The bilateral contact can be expressed as an extension of the formulation proposed in the previous section. We need to impose a linear mixture law in the normal direction and a quadratic mixture law in the tangential direction. The first insures a sticking contact in the normal direction whereas the second allows sliding in the tangential direction. If we successfully represent a directional tool, the same results obtained in Figure 5.9 are expected but with a slight improvement in the critical zones (in the vicinity of the corners). This is the main focus of this section. We should note that an anisotropic solver can be exploited in many ways.

After breaking these facts, developing an anisotropic solver seems to be the right candidate to our approach. We began searching in the literature for similar works or at least for any directional notion that we can exploit to our favor. Two particular methods stand out. **The first category** is used frequently in oceanography and geology modeling [Bellec et al., 2013][Iftimie and Planas, 2006][Zhang and Fang, 2008]. They generally proposes to represent the consistencies anisotropy using a diagonal tensor. In other words, the scalar consistency is replaced by a tensor, where the different consistencies are directionally dependent. They frequently suppose that the consistency in the $z-$direction is a small value tending toward zero:

$$\eta = \begin{bmatrix} \eta_x & 0 & 0 \\ 0 & \eta_y & 0 \\ 0 & 0 & \eta_\varepsilon \end{bmatrix} \tag{5.10}$$

where η_x, η_y and η_ε are respectively the consistencies in x, y, and z directions with $\eta_\varepsilon \to 0$.

Using this tensor, the Navier-Stokes equations are rewritten:

$$\begin{aligned} \frac{\partial v}{\partial t} - \left(\eta_x \frac{\partial^2}{\partial x^2} + \eta_y \frac{\partial^2}{\partial y^2} + \varepsilon \frac{\partial^2}{\partial z^2} \right) v + v.\nabla v = -\nabla p \\ div\, v = 0 \end{aligned} \tag{5.11}$$

Several simplifications are usually used to reduce the equations into Saint-Venant equations (known as shallow water equations as well).

The second category based on the fiber theory proposes a more general form to portray the anisotropic viscosity by introducing an orientation tensor. Based on the work of [Folgar and Tucker, 1984], [Redjeb, 2007] proposes a new decomposition using the strain rate tensor $\varepsilon(v)$. As a result a new behavior law taken into account the directional viscosity is presented.

Note that several work were dedicated to prove the existence and unicity of the solution in the anisotropic equations [Paicu, 2005] [Iftimie and Planas, 2006]. We highlight in particular, the work presented in [Besson et al., 1990] proving the existence and unicity of the anisotropic solution in the same Sobolev spaces used in our standard formulation (described in sec. 2.5.4.1).

In this work, we rather chose the second approach to express the anisotropic consistency. Inspired by their decomposition, the stain rate tensor $\varepsilon(v)$ is multiplied by an orientation tensor a to express the anisotropy. The orientation tensor will determine mainly in which direction the deformation/velocity is greater.

The deviatoric stress tensor can be expressed as follows:

$$s = [\varepsilon(v).a + a.\varepsilon(v)] \tag{5.12}$$

in which a is the orientation tensor depending of the anisotropic consistency η.

If $\eta = \begin{pmatrix} \eta_x & 0 & 0 \\ 0 & \eta_y & 0 \\ 0 & 0 & \eta_z \end{pmatrix}$ denotes the diagonal consistency tensor with η_x, η_y and η_z respectively the consistencies in the x, y and z directions, the orientation tensor a can take the following form:

$$a = {}^{\mathbf{t}}\mathbf{R}\eta\mathbf{R} \tag{5.13}$$

where $\mathbf{R}$ is a rotation matrix .

Equation (5.13) guaranties that the couple $(\eta, \mathbf{R})$ offers a general description of the anisotropy (i.e. in all directions).

Combining equations (5.12) and (5.13), the deviatoric tensor is now

$$s = \left[\varepsilon(v).{}^{t}\mathbf{R}\eta\mathbf{R} + {}^{t}\,\mathbf{R}\eta\mathbf{R}.\varepsilon(v)\right] \tag{5.14}$$

Since $\mathbf{R}$ is antisymmetric and η is diagonal then the orientation tensor a is symmetric (${}^{t}a = {}^{t}\,({}^{t}\mathbf{R}\eta\mathbf{R}) = {}^{t}\,\mathbf{R}\eta\mathbf{R} = a$). By that, we insure that the deviatoric part of the Cauchy tensor remains symmetric as well.

Note that if $\mathbf{R}$ is equal to the identity tensor $\mathbb{I}$ and all the directional consistencies are

equal (i.e. $\eta_x = \eta_y = \eta_z$), we recover the classic form of the deviatoric tensor $s = 2\eta\varepsilon(v)$.

To conclude, the system to be solved is summarized by:

$$\left\{\begin{array}{rcl} \nabla.\sigma & = & 0 \\ \nabla.v & = & 0 \\ \sigma & = & [\varepsilon(v).a + a.\varepsilon(v)] - p\mathbf{I} \end{array}\right. \tag{5.15}$$

in which the inertia term are dropped for sake of simplicity.

5.3.1 Weak formulation

To retrieve the weak formulation of (5.15), the same steps as in chapter 2 are used. The constitutive equations are multiplied by the test function couple (w, q). The only noticeable difference is replacing s by $\varepsilon(v).a + a.\varepsilon(v)$. This substitution leads to new terms in the variational formulation. The integral $(2\eta\varepsilon(v) : \varepsilon(w))_{\Omega_i}$ will be replaced by the following one $(\varepsilon(v).a : \varepsilon(w))_{\Omega_i} + (a.\varepsilon(v) : \varepsilon(w))_{\Omega_i}$:

$$\left\{\begin{array}{rcl} (\varepsilon(v).a : \varepsilon(w))_{\Omega_i} + (a.\varepsilon(v) : \varepsilon(w))_{\Omega_i} - (p, \nabla.w)_{\Omega_i} & = & 0 \\ (\nabla.v, q)_{\Omega_i} & = & 0 \end{array}\right. \tag{5.16}$$

5.3.2 Spatial discretization

In this section, we recall some basics about the spatial discretization of the incompressible Stokes equations by the finite element method. The emphasis will be on the new terms introduced in system (5.15). Then we describe the implementation of our code in CIMLib. In order to simplify the problem into a practical for computation, we will be working locally on each simplex K.

Let $(N^i)_{i=1,\dots,D}$ denote the nodal shape functions. Then the solution v and the test function are written as follows :

$$v_K = \sum_{i=1}^{D} v^i N^i \quad and \quad w_K = \sum_{j=1}^{D} w^j N^j$$

As mentioned earlier, the only modifications will concern the local matrix denoted A_{vv}^{loc} (see equation (2.68) in chapter 2). It is the matrix form of the following term found in the classic solver :

$$\int_K \varepsilon\left(N_k^i\right) : \varepsilon\left(N_l^j\right) dK = \begin{cases} \dfrac{1}{2}\displaystyle\int_K \frac{\partial N^i}{\partial x_l}\frac{\partial N^j}{\partial x_k} dK \quad if\, k \neq l \\ \dfrac{1}{2}\displaystyle\int_K \sum_m^d \frac{\partial N^i}{\partial x_m}\frac{\partial N^j}{\partial x_m} dK + \frac{1}{2}\int_K \frac{\partial N^i}{\partial x_l}\frac{\partial N^j}{\partial x_l} dK \quad if\, k = l \end{cases} \tag{5.17}$$

where N_k^i is the k^{th} component of the shape function on the i^{th} node.

However computing the new A_{vv}^{loc} in system (5.16) proves to be more expensive than the classical one. For instance,

$$\varepsilon_k^i.a = \begin{pmatrix} \dfrac{\partial N^i}{\partial x_1}a_{k1} & \cdots & \dfrac{\partial N^i}{\partial x_1}a_{kk} & \cdots & \dfrac{\partial N^i}{\partial x_1}a_{kd} \\ \dfrac{\partial N^i}{\partial x_2}a_{k1} & \cdots & \dfrac{\partial N^i}{\partial x_2}a_{kk} & \cdots & \dfrac{\partial N^i}{\partial x_2}a_{kd} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \displaystyle\sum_m^d \frac{\partial N^i}{\partial x_m}a_{m1} + \frac{\partial N^i}{\partial x_k}a_{k1} & \cdots & \displaystyle\sum_m^d \frac{\partial N^i}{\partial x_m}a_{m1} + \frac{\partial N^i}{\partial x_k}a_{kk} & \cdots & \displaystyle\sum_m^d \frac{\partial N^i}{\partial x_m}a_{m1} + \frac{\partial N^i}{\partial x_k}a_{kd} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \dfrac{\partial N^i}{\partial x_d}a_{k1} & \cdots & \dfrac{\partial N^i}{\partial x_d}a_{kk} & \cdots & \dfrac{\partial N^i}{\partial x_d}a_{kd} \end{pmatrix} \tag{5.18}$$

Instead, we write $(\varepsilon_k^i.a + a.\varepsilon_k^i) : \varepsilon_k^i$ using the gradient decomposition. The new A_{vv}^{loc} is now computed using the following form:

$$A_{vv}^{loc} = \frac{1}{4}\int_K \left[\left(\nabla N_k^i.a + {}^t\nabla N_k^i.a\right) + \left(a.\nabla N_k^i + a.{}^t\nabla N_k^i\right)\right] : \left(\nabla N_l^j + {}^t\nabla N_l^j\right) dK \tag{5.19}$$

Equation (5.19) gives eight contraction terms leading to eight local matrices. This increases the computational time. To overcome this increase, we compute only four matrices defined by equations (5.20) to (5.23) .

$$A_1^{loc} = \int_K \nabla N_k^i.a : \nabla N_l^j dK \tag{5.20}$$

$$A_2^{loc} = \int_K \nabla N_k^i.a : {}^t\nabla N_l^j dK \tag{5.21}$$

$$A_3^{loc} = \int_K {}^t\nabla N_k^i.a : \nabla N_l^j dK \tag{5.22}$$

$$A_4^{loc} = \int_K {}^t\nabla N_k^i.a : {}^t\nabla N_l^j dK \tag{5.23}$$

The other four are deducted by computing their transpose (5.24) :

$$A_5^{loc} = {}^tA_1^{loc}, \quad A_6^{loc} = {}^tA_2^{loc}, \quad A_7^{loc} = {}^tA_3^{loc}, \quad A_8^{loc} = {}^tA_4^{loc} \tag{5.24}$$

The local matrix A_{vv}^{loc} is now the sum of all the individual matrices defined above:

$$A_{vv}^{loc} = \frac{1}{4}\left(A_1^{loc} + A_2^{loc} + \cdots + A_7^{loc} + A_8^{loc}\right) \tag{5.25}$$

Term by term decomposition:

$$\left(\nabla N_k^i.a\right)_{n,m} = \begin{cases} 0 & if\ n \neq k \\ \sum_{p=1}^{m} \dfrac{\partial N^i}{\partial x_p} a_{pm} & if\ n = k\ \forall m = 1, \ldots, d \end{cases} \tag{5.26}$$

$$\left({}^t\nabla N_k^i.a\right)_{n,m} = \begin{pmatrix} \dfrac{\partial N^i}{\partial x_1}a_{k1} & \cdots & \cdots & \cdots & \dfrac{\partial N^i}{\partial x_1}a_{kd} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \dfrac{\partial N^i}{\partial x_n}a_{km} & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \dfrac{\partial N^i}{\partial x_d}a_{k1} & \cdots & \cdots & \cdots & \dfrac{\partial N^i}{\partial x_d}a_{kd} \end{pmatrix} \tag{5.27}$$

$$\nabla N_k^i.a : \nabla N_l^j = \begin{cases} 0 & if\ k \neq l \\ \sum_m \left[\left(\sum_n \dfrac{\partial N^i}{\partial x_n} a_{nm}\right) \dfrac{\partial N^j}{\partial x_m}\right] & if\ k = l \end{cases} \tag{5.28}$$

$$\nabla N_k^i.a : {}^t\nabla N_l^j = \left(\sum_p \frac{\partial N^i}{\partial x_p} a_{pl}\right) \frac{\partial N^j}{\partial x_k} \tag{5.29}$$

$${}^t\nabla N_k^i.a : \nabla N_l^j = \sum_m^d \left[\left(\frac{\partial N^i}{\partial x_l} a_{km}\right) \frac{\partial N^j}{\partial x_m}\right] \tag{5.30}$$

$${}^t\nabla N_k^i.a : {}^t\nabla N_l^j = \sum_n^d \left[\left(\frac{\partial N^i}{\partial x_n} a_{kl}\right) \frac{\partial N^j}{\partial x_n}\right] \tag{5.31}$$

5.3.3 Variational Multi-scale Method

Similar to the classic Navier-Stokes Solver, the anisotropic Stokes solver is unstable as well. A stabilization technique should be put in place. A decision had to be made between the P1+/P1 and the VMS technique. An equivalence between these two methods was established in [Brezzi et al., 1997]. The choice fell on the latter. It is more general (it can be used in many cases) and offers a simpler yet efficient stabilization to our problem. The stabilization is insured via one parameter called τ_c . In other words, the condensation needed in P1+/P1 is avoided and instead automatically managed using τ_c. In what follows, an extension of the VMS stabilization technique presented in chapter 2 is adapted to our system of equation(5.12). The upside of this choice is our ability to use the same VMS technique by only replacing the stabilization parameter with the new one.

Stabilization techniques are used for numerous problems nowadays, the literature offers many eligible stabilization parameters [Tezduyar and Osawa, 2000a]. We mainly were interested in the work presented in [Principe, 2008] where a new stabilization parameter taking into account the anisotropy is proposed.

The same reasoning is used as before. The solution spaces V and Q are respectively decomposed into two spaces V_h and Q_h (resolvable space) and $\tilde{V_k}$ and $\tilde{Q}_k$ (sub-grid scale space): $(V, Q) = (V_h \oplus \widetilde{V_k}, Q_h \oplus \widetilde{Q_k})$. Thus the unknowns of the problem (v, p) are replaced by $(v_h + \tilde{v}, p_h + \tilde{p})$. The weak formulation is written using the same decomposition. Rather than solving it directly, the author of [Principe, 2008] transforms the equation to the reference domain, then a Fourier analysis of the sub-scale problem is used as proposed in [Codina, 2002].

After a lengthy calculation, a 2D stabilization parameter is computed in terms of the velocity, viscosity and the mesh size.

We adapted it to our needs by supposing that the viscosity and velocity take respectively the following forms $\eta = \begin{pmatrix} \eta_1 & \eta_{12} & \eta_{13} \\ \eta_{12} & \eta_2 & \eta_{23} \\ \eta_{13} & \eta_{23} & \eta_3 \end{pmatrix}$, $v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$.

The new stabilization term, customized for anisotropic consistency and mesh orientation can be expressed as:

$$\tau_c = (4M\eta + c)^{-1/2} \quad where \quad c = 4\left(\frac{v_1^2}{h_1^2} + \frac{v_2^2}{h_2^2} + \frac{v_3^2}{h_3^2}\right) \tag{5.32}$$

in which M is the mesh metric and h_1, h_2 and h_3 are the mesh sizes in the different directions. Note that since we are solving anisotropic Stokes equations, the convection term c is dropped in the following applications.

5.3.4 Applications

As explained previously, the main focus behind the anisotropic solver was to recreate a bilateral sliding contact. Nevertheless, this solver can be of a great importance for other applications in particular for anisotropic flows of viscoplastic materials. We begin by applying the anisotropic solver on a free flow case for two reasons: First, to insure that the solver is perfectly running and second to give a general idea on how it can be used in other applications. Afterward, the solver is used to model bilateral sliding contact in large deformation problems. The center of attention is to evaluate this solver e.g if it is general enough to offer the needed information or if it requires additional manipulation.

5.3.4.1 Free flow

In the computational domain $[0;1] \times [0;1] \times [0;1]\, m^3$, an anisotropic material is filled initially in the back corner of the domain to take the form of a quarter of a cylinder (symmetries are applied on both sides). The radius R and height H are respectively equal to $0.4\, m$ and $0.85\, m$ (see Figure 5.10).

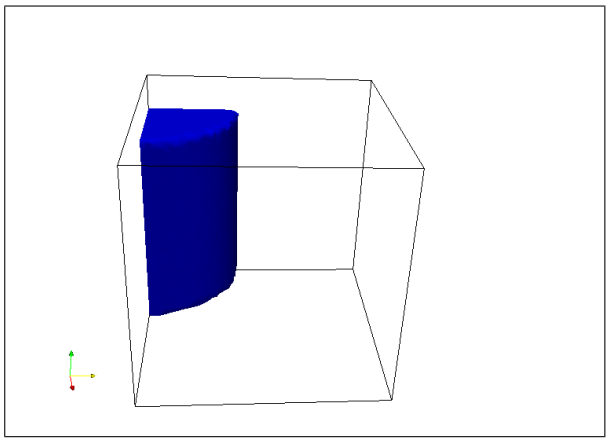


Figure 5.10: *Initial position of the anisotropic material.*

The remaining space is supposed to be filled with air . The air and the material densities are respectively $\rho_{air} = 1\, kg/m^3$ and $\rho_m = 5000\, kg/m^3$. The air viscosity is $\eta_{air} = 10^{-5}\, Pa.s$. As for the anisotropic material, its viscosity is no longer similar in all directions. It is expressed as a tensor $\eta_m = \begin{pmatrix} 10^6 & 0 & 0 \\ 0 & 10^4 & 0 \\ 0 & 0 & 10^4 \end{pmatrix}$ where the diagonal terms represent the viscosities in the different directions. The material flows due to gravity with no additional external forces. The evolution is followed during time. The global domain is meshed using 29 000 nodes. The simulation is launched on 4 cores.

Since the viscosity in the $x-$direction is bigger than the one in the $y-$direction, the material is thicker and should move slower in this direction. Figure 5.11 confirms this

fact. The velocities maps in both directions are presented at $t \approx 1\,s$. It is clear that the velocity v_y is bigger than the velocity v_x, the material moves forward faster in the $y-$direction .

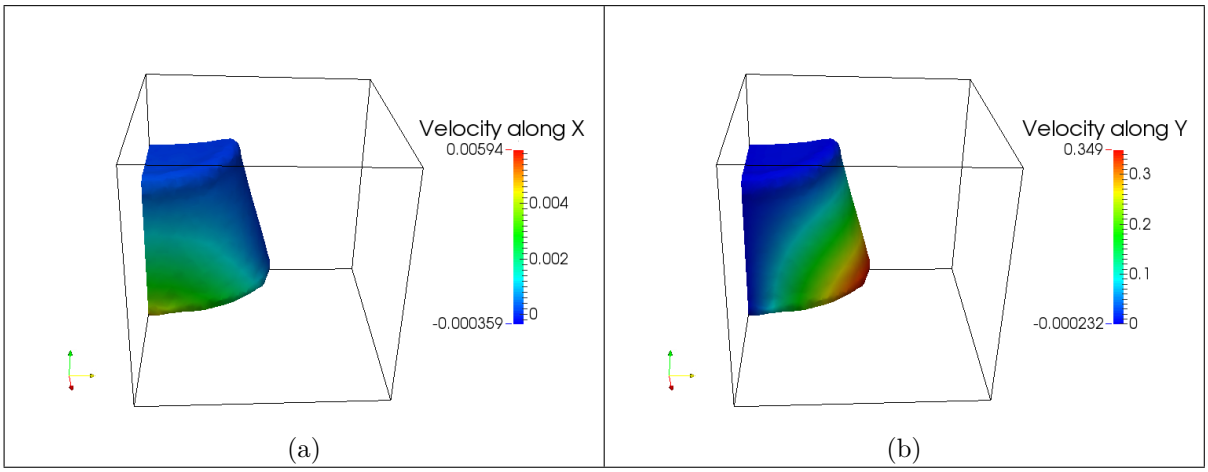


Figure 5.11: *Maps of the velocity components v_x and v_y at $t \approx 1\,s$*

Figure 5.12 completes this discussion. To visualize even more the progress of the anisotropic material, the zero-isovalue is presented at $t \approx 2.1\,s$. The evolution is indeed more visible in the $y-$direction. When inspected closely, we noticed an uneven top (Figure 5.12a). This is not considered as an unusual behavior. In fact, since the material advances more in one direction under nothing other than gravity, less material is located in the center which means less support. This can generate a slack region in the center (the corner vicinity).

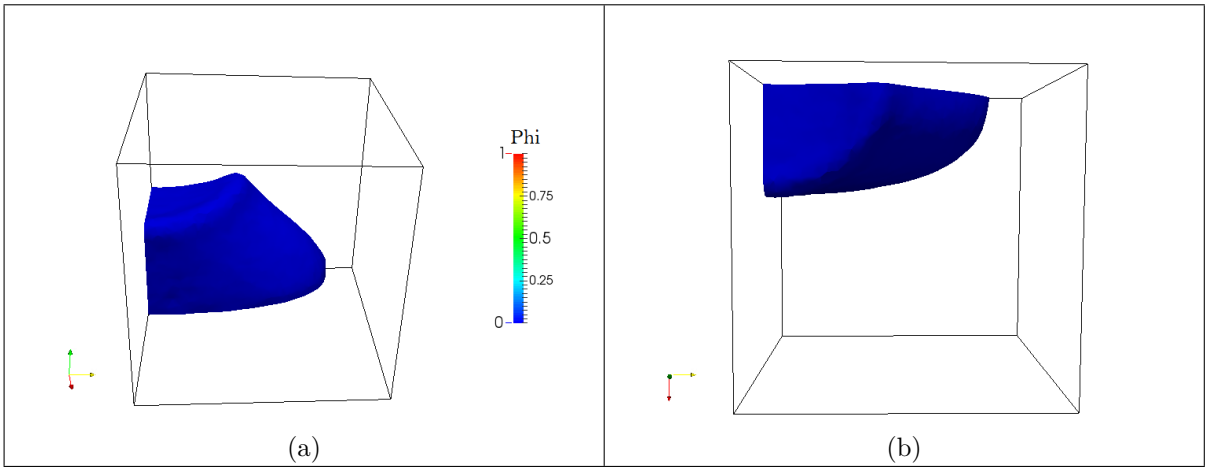


Figure 5.12: *Side and upper views of the final position of the material corresponding to $t \approx 2.1\,s$.*

Though this case is simple and maybe considered academic, it is the perfect example to illustrate the performance of the anisotropic solver. Its behavior is overall satisfactory.

5.3.4.2 Towards contact modeling

An intermediate step before applying the anisotropic solver to contact problem is presented hereby. The computational domain $[0;1] \times [0;1] \times [0;1]\, m^3$ is formed by two sub-domains, the deformable material (in red) and the lubricant (in blue) covering the top and the bottom (Figure 5.13). Note that the lubricant in this case is no longer represented implicitly via a quadratic mixture law. Instead it is explicitly defined as a sub-domain of thickness $0.1\, m$. On the domain's upper and lower boundaries, we impose velocities of respectively $5\, m/s$ and $-5\, m/s$. These boundary conditions emulate in the simplest way a contact problem without actual contact between tools and the deformable body.

Note that the chosen lubricant thickness is somewhat larger than what is expected in reality, but this case is purely academic to test the behavior of the anisotropic solver.

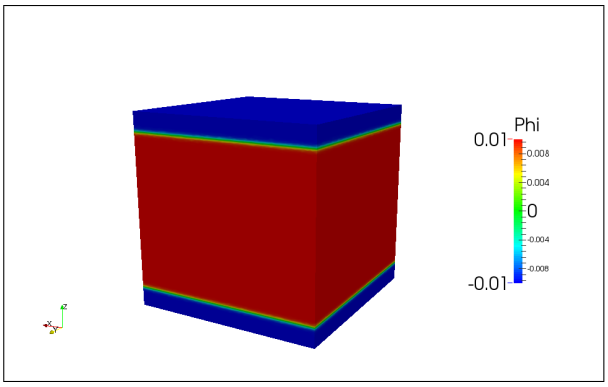


Figure 5.13: *The computational domain formed by two phases: the deformable body and the lubricant.*

The consistencies are defined as diagonal tensors. The deformable body consistency is isotropic. The anisotropy is expressed in the lubricant consistency. Thus, $\eta_{d_1} = \eta_{d_2} = \eta_{d_3} = \eta_d\, (Pa.s)$ and the consistency tensors are expressed as follows:

$$\overline{\overline{\eta}}_d = \begin{pmatrix} \eta_d & 0 & 0 \\ 0 & \eta_d & 0 \\ 0 & 0 & \eta_d \end{pmatrix} = \begin{pmatrix} 10^7 & 0 & 0 \\ 0 & 10^7 & 0 \\ 0 & 0 & 10^7 \end{pmatrix} \quad \text{and} \tag{5.33}$$

$$\overline{\overline{\eta}}_{lub} = \begin{pmatrix} \eta_{lub} & 0 & 0 \\ 0 & \eta_{lub} & 0 \\ 0 & 0 & \eta_d \end{pmatrix} \tag{5.34}$$

In this case, we are using an Eulerian orthonormal frame. The rotation matrix $\mathbf{R}$ is

the identity tensor (no rotation is needed):

$$\mathbf{R} = \mathbb{I} \qquad \vec{n} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{5.35}$$

Note that we fixed $\eta_{lub_3} = 10^7\, Pa.s$, and we control the anisotropy by changing η_{lub_1} and η_{lub_2}. For instance, η_{lub_1} and η_{lub_2} takes three different values: $10^7\, Pa.s$, $10^5\, Pa.s$ and $10^3\, Pa.s$. For the first value, the lubricant consistency is isotropic. The resulting velocity profile should be the same as for a sticking friction. The latter values suggest that the anisotropic ratios are respectively 10^2 and 10^4. We should expect more sliding impacting the velocity profiles. Figure 5.14 illustrates the velocities plotted on an outer section in the thickness positioned on the middle of the cube side.

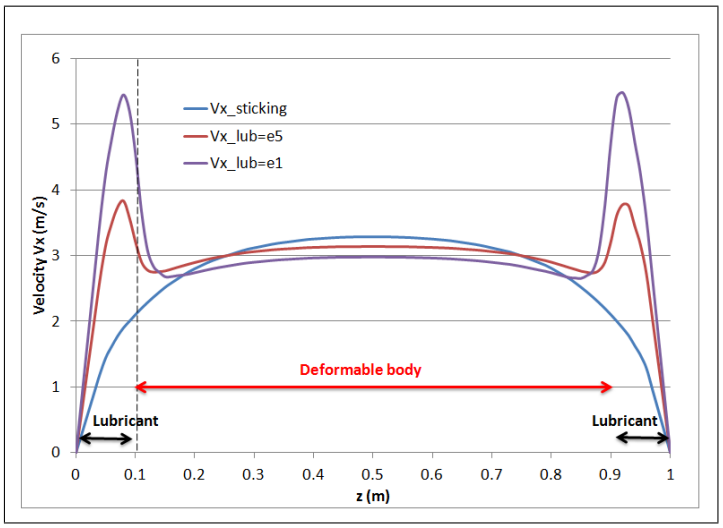


Figure 5.14: *The impact of a variant lubricant anisotropic consistency on the velocity profile* v_x.

Notice that for $\eta_{lub_1} = \eta_{lub_2} = 10^7\, Pa.s$, the velocity profile is perfectly parabolic. This is the kind of response we get when sticking contact is modeled. When decreasing the lubricant consistency, the anisotropy ratio increases and more sliding is allowed on the edges of the deformable body. More sliding results in bigger velocities on the edges and the velocity profile (in the material) is more and more flat with the decreasing consistency. The velocity curves in the deformable body confirm our analysis. Note that the velocity picks are located in the lubricant domain which is perfectly normal. Smaller the lubricant consistency, lesser it resists to displacement (e.g bigger the velocity) . To conclude, the behavior of the anisotropic solver is once again proven logic and satisfactory.

5.3.4.3 Contact

Now that the well-behavior of the anisotropic solver is approved, we attempt using it to model a 2D-contact problem. The lubricant phase is presented implicitly using the quadratic mixture law (contrary to the previous case). Combined together, we await an imitation of a bilateral sliding contact. The monolithic anisotropic consistency tensor $\overline{\overline{\eta}}$ defined all over the domain is obtained in a delicate manner. In the normal direction, it is recovered using a linear mixture law. Whereas in the tangential directions, a quadratic mixture law enabling η_{lub} is used:

$$\overline{\overline{\eta}} = \begin{pmatrix} \eta_{quad} & 0 \\ 0 & \eta_{lin} \end{pmatrix} \tag{5.36}$$

where η_{quad} and η_{lin} are the quadratic and linear mixture laws defined all over the domain.

To fix the ideas, we should mention that the anisotropy in $\overline{\overline{\eta}}$ is a direct outcome from the lubricant anisotropy: $\begin{pmatrix} \eta_{lub} & 0 \\ 0 & \eta_{lin} \end{pmatrix}$

Equation (5.36) expresses the tensor in the Cartesian frame. To generalize its form, we apply a rotation matrix $\mathbf{R}$.

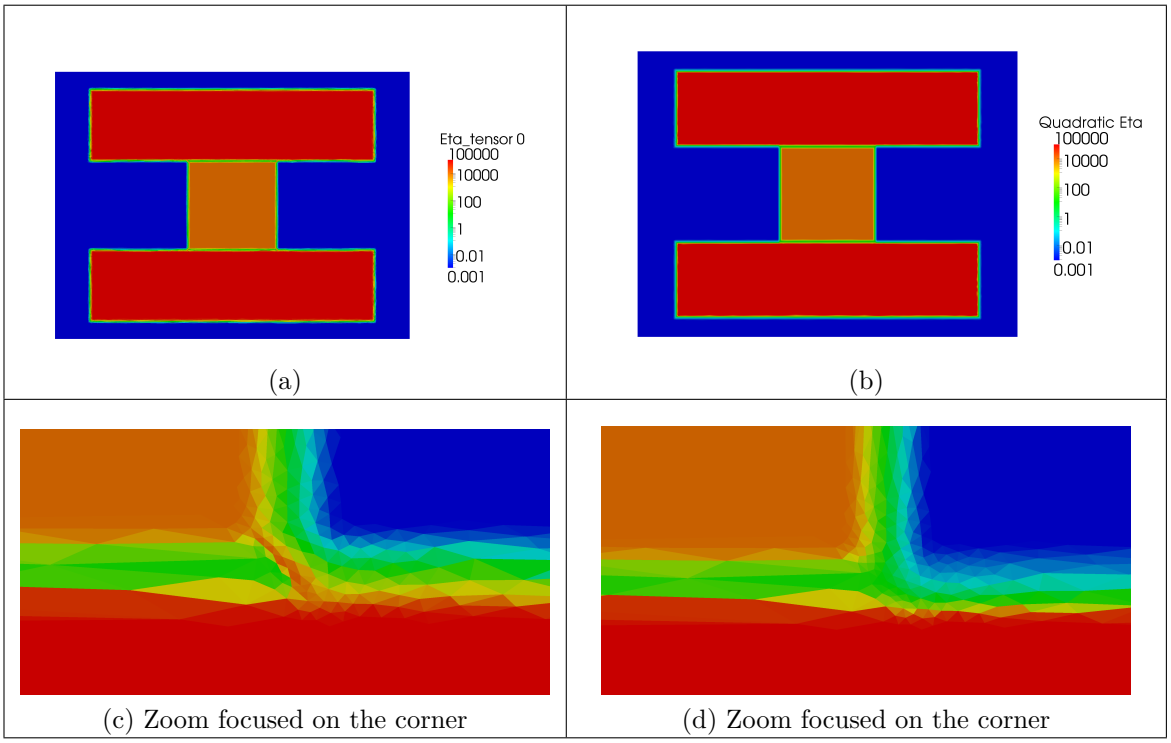
The rotation matrix $\mathbf{R}$ is determined using $\vec{n}$, the normal vector to the level set function ϕ:

$$\mathbf{R} = (n, n^{\perp}) \qquad \vec{n} = \begin{pmatrix} n_x \\ n_y \end{pmatrix} = \frac{\nabla\phi}{||\nabla\phi||} \tag{5.37}$$

In this 2D contact problem, the upper tool moves downward with a velocity of $0.3\,m/s$. The lubricant, deformable body and tools consistencies are respectively equal to $700\,Pa.s$, $10^4\,Pa.s$ and $10^5\,Pa.s$.

The results were inspected and did not meet our expectations. The overall results issued from this case were similar to the ones obtained using a linear mixture law e.g. sticking contact.

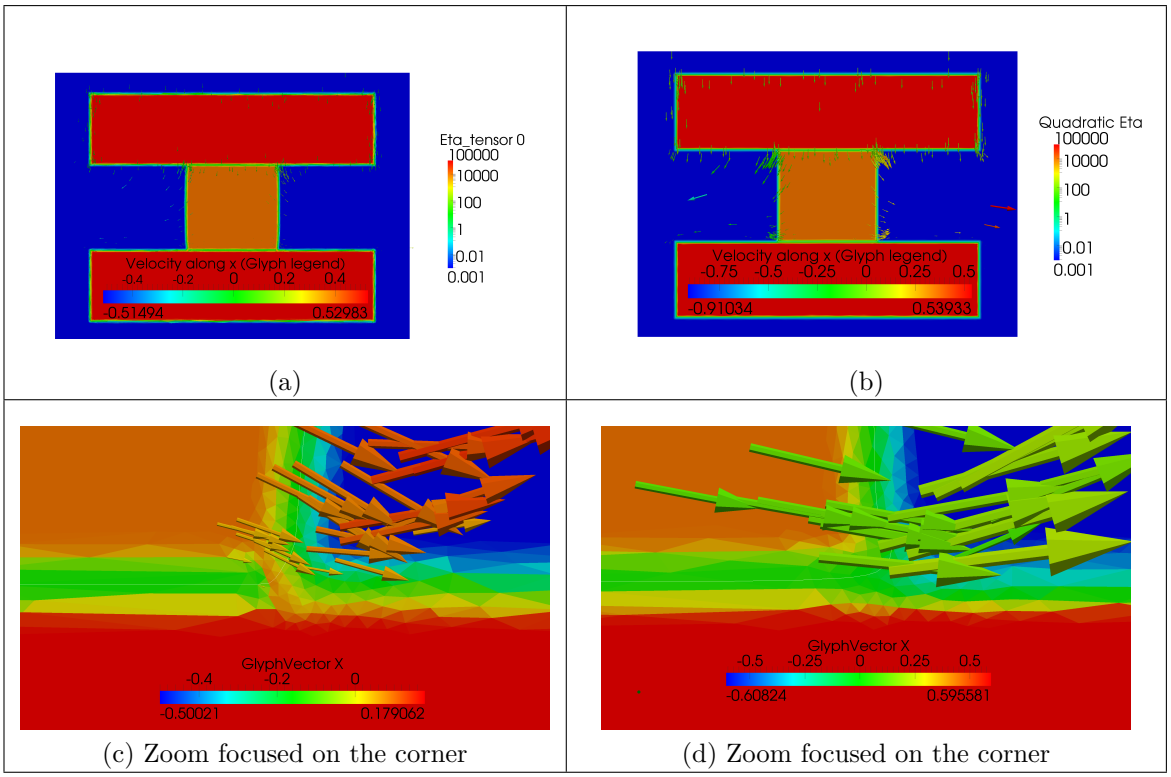
The cause was immediately identified. Figure 5.15 illustrates the consistency computed all over the domain using the anisotropic solver and a quadratic mixture law. A zoom focused on the corner uncovers an irregular zone in which the consistency is unexpectedly large. To complete the discussion, the flow material presented using arrows is illustrated in Figure 5.16 at $t = 0.014\,s$. In the corner, when using the anisotropic solver the deformable body is unable to slide properly due to the barrier created by the large consistency. The flow should be more like the one issued when using the quadratic law.



(a) (b)

(c) Zoom focused on the corner (d) Zoom focused on the corner

Figure 5.15: *At $t = 0\,s$, a superposition between the anisotropic consistency (on the left) and the consistency using the quadratic mixture law (on the right).*



(a) (b)

(c) Zoom focused on the corner (d) Zoom focused on the corner

Figure 5.16: *At $t = 0.014\,s$, a superposition between the material flow (illustrated by arrows) using an anisotropic consistency (on the left) and a quadratic mixture law (on the right).*

Though the behavior of the anisotropic solver was proven satisfactory for other applications, it remains limited when it comes to contact modeling. This limitation arises due to the level set gradient. The computed gradient shows singularities and leads to an ill-defined consistency tensor on the corners. It leads to rotated consistency in the x direction forming a barrier preventing the material from sliding freely in the tangential direction.

5.4 Conclusion

This chapter attempts to model friction and contact in an Eulerian framework. We presented a quadratic mixture law to introduce a lubricant implicitly between the tool and the deformable body. The lubricant consistency and thickness are deduced by an identification technique with the friction parameter in a Norton-Hoff friction law. We carry out a ring test case and the results were compared with Forge®. They were found comparable but highly dependent of the boundary layer thickness. For a small thickness, the lubricant consistency is overpowered by the tools/deformable body consistencies and consequently the boundary layer is not properly presented. However for a large thickness, an odd behavior is detected near the corners.

The development of the directional solver was meant to model bilateral sliding contact. A VMS stabilization technique was used to insure the stability. The new solver was validated with two different cases: i) in a free flow simulation and ii) an explicit contact case. However when applying the directional solver to a 2d case, where the friction is represented implicitly using a quadratic mixture law, some difficulties were confronted. The gradient has proven to be the problem, it introduces a rotated consistency values on the corner of the deformable body. It acts like a barrier preventing the sliding of the deformable body. An alternative method, non dependent of the mesh (here we mean the interface thickness of the level set), is the use of Navier boundary conditions. The latter boundary condition is a combination between a Dirichlet conditions and the Newman ones. However to apply it, we should compute explicitly the contact surface between the tool and the deformable body.

Résumé en Français

Ce chapitre est dédié à la modélisation du frottement et du contact dans un cadre Eulérien. Nous avons présentés une loi de mélange quadratique pour introduire implicitement un lubrifiant entre l'outil et le corps déformable. La viscosité et l'épaisseur du lubrifiant sont déduites par une technique d'identification avec le paramètre de frottement dans une loi Norton-Hoff. En utilisant le cas de déformation d'un anneau, les résultats de notre approche Eulérienne utilisant la loi quadratique ont été comparés ave Forge. Les résultats sont satisfaisants mais fortement dépendant de l'épaisseur de la couche limite. Pour une petite épaisseur, les consistances de l'outil et du corps déformable emportent sur la viscosité du lubrifiant. Par conséquent, la couche limite n'est pas proprement représentée. Pour surmonter ce problème, le maillage est la réponse-clé pour une propre représentation. Dans le but de modéliser un problème de contact glissant bilatéral, nous avons proposé un solveur directionnel stabilisé par la technique VMS. Le nouveau solveur a été validé avec deux cas différents : i) dans une simulation d'un écoulement libre d'un matériau anisotrope et ii) un cas de contact explicite. Cependant en appliquant le solveur directionnel à cas 2D, où le frottement est représenté implicitement en utilisant une loi de mélange quadratique, quelques difficultés ont été confrontées. Le gradient s'avérant être l'origine, il agit comme une barrière empêchant le glissement du corps déformable. Pour surmonter cette difficulté, nous devons revisiter la définition des normales dans notre formulation directionnelle.

Bibliography

[Bellec et al., 2013] Bellec, S., Colin, M., and Ricchiuto, M. (2013). Sur des modèles asymptotiques en océanographie. Research Report RR-8361.

[Besson et al., 1990] Besson, O., Laydi, M., and Touzani, R. (1990). Un modèle asymptotique en océanographie. *Comptes Rendus De L Academie Des Sciences Serie I-Mathematique*, 310(9):661–665.

[Brezzi et al., 1997] Brezzi, F., Franca, L., Hughes, T., and Russo, A. (1997). b = $\int$ g. *Computer Methods in Applied Mechanics and Engineering*, 145(3-4):329 – 339.

[Bruchon et al., 2009] Bruchon, J., Digonnet, H., and Coupez, T. (2009). Using a signed distance function for the simulation of metal forming processes: Formulation of the contact condition and mesh adaptation. from a lagrangian approach to an eulerian approach. *International Journal for Numerical Methods in Engineering*, 78(8):980–1008.

[Codina, 2002] Codina, R. (2002). Stabilized finite element approximation of transient incompressible flows using orthogonal subscales. *Computer Methods in Applied Mechanics and Engineering*, 191(39):4295–4321.

[Folgar and Tucker, 1984] Folgar, F. and Tucker, C. L. (1984). Orientation behavior of fibers in concentrated suspensions. *Journal of Reinforced Plastics and Composites*.

[Foudrinier, 2007] Foudrinier, E. (2007). *3D Computation of reactive moulding processes*. PhD thesis, École Nationale Sup/'erieure des Mines ParisTech.

[Iftimie and Planas, 2006] Iftimie, D. and Planas, G. (2006). Inviscid limits for the navier-stokes equations with navier friction boundary conditions. *Nonlinearity*, 19(4):899.

[Mondalek, 2012] Mondalek, P. (2012). *Numerical modeling of the spark plasma sintering process*. PhD thesis, École Nationale Sup/'erieure des Mines ParisTech.

[Paicu, 2005] Paicu, M. (2005). Équation anisotrope de navier-stokes dans des espaces critiques. *Rev. Mat. Iberoamericana*, 21(1):179–235.

[Principe, 2008] Principe, J. (2008). *Subgrid scale stabilized finite elements for low speed flows*. PhD thesis.

[Redjeb, 2007] Redjeb, A. (2007). *Simulation numérique de l'orientation de fibres en injection de thermoplastique renforcé*. PhD thesis, École de Mines ParisTech.

[Zhang and Fang, 2008] Zhang, T. and Fang, D. (2008). Global wellposed problem for the 3-d incompressible anisotropic navier-stokes equations. *Journal de Mathématiques Pures et Appliquées*, 90(5):413 – 449.

1 Conclusions

This work is one of the first attempts studying the feasibility of multi-domain large deformation problems using an unusual approach, "the monolithic Eulerian technique". The resulting model is desired to be able of *i)* following complex geometries evolution during time, *ii)* describing multi-domain problems, *iii)* modeling friction and contact and *iv)* giving results more or less comparable to real applications (using *Forge*® as the base of our superpositions). All the mentioned objectives has to be accomplished in a highly parallel scalable environment.
In other words, this work is the first step toward an Eulerian *Forge*®:

- ◇ It has to assess if the current Eulerian approach -available in CIMLib- is fit to large deformation modeling.
- ◇ It has to propose and test feasible development and recommend answers to unsolved aspects for future works.
- ◇ It has to conclude if the resulting approach has industrializing potential on the long run.

After an overview of the different approaches fit for large deformation modeling, the literature confirmed that the Eulerian approach is rarely used for these kind of problems (chapter 1). Therefore, adopting the Eulerian technique will require an outside of the box thinking to find answers to our problems. However, the undeniable potential of this method makes it a great candidate (e.g the capacity of managing automatically contact detection). The monolithic approach -base of the CIMLib- is described in details in the first part of chapter 2.

To adapt the standard version to large deformation problems, additional developments and manipulations are needed . The second part of chapter 2 is fully dedicated to these improvements:

◇ A new quadratic mixture law was implemented. It offers the option of adding a new phase between two interacting bodies to simulate the presence of a lubricant for instance. Note that this is a great deal when attempting large deformation modeling. Especially when there is a need to model friction between different domains.

◇ A new manner to impose boundary conditions was presented. It enables the users to let the air escape from the computational domain emulating actual vents.

◇ Technical ameliorations were proposed as well. We recommend an optimal choice of parameters such as *i)* mixture law weight, order of interpolation and a variable time step to improve volume conservation and *ii)* mesh size choice for the stabilization of the convection solver.

chapter 3 and chapter 4 were entirely dedicated to numerical applications. The first is devoted for the parallel environment in CIMLib and its efficiency. The overall performance was found satisfactory. The second presents different applications to evaluate the capacity of the approach in several scenarios: *i)* deforming complex geometries such as crankshafts and connecting rod, *ii)* deforming simple geometries but stacked in great numbers (up to 256 deformable geometries). The results were validated via *Forge*® simulations and found very satisfactory. In some cases (with high numbers of deformable bodies), the performance was even more resilient than *Forge*®. After these comparisons, we were reassured that this approach exhibits real potential to compete with an industrial software. To push even further, attempts to model friction and contact problems in an Eulerian approach were the subject matter of chapter 5. Simple yet effective, propositions were formed:

◇ To model a sticking friction, a linear mixture law is the answer. The transition between a high consistency (for the solid) and a lower consistency (for the deformable body) results with a high resistance emulating numerically friction. The results were validated by deforming a ring between two rigid tools in both CIMLib and *Forge*®. The results were a perfect match.

◇ To control friction, a quadratic mixture law is used enabling a fictitious boundary layer imitating the presence of a lubricant between the different bodies. The lubricant viscosity is determined in terms of the deformable body consistency and the boundary layer thickness. This relationship was established via a superposition with Lagrangian friction to make sure the same cases are modeled. Again, a comparison

with simulations carried out in *Forge®* were established. The results showed the same behavior but exhibited some limitations as well for certain cases. The results were highly dependent of the boundary layer thickness. If the needed thickness is very small, the mesh may not detect the boundary layer and the lubricant consistency is not reached. In this case, the same results as in sticking friction were found. If the thickness is increased to prevent this problem but a big gap is present between the different consistencies, an unrealistic behavior is detected in the vicinity were the different values meet (on the corners).

Even though the quadratic law was proven effective for some cases in controlling friction, the overall performance cannot be generalized.

◇ The directional solver is introduced to model bilateral sliding contact. The consistency is expressed as a tensor. Although our intentions were to model contact in a general way, the solver can be used for other high-demand applications such as anisotropic polymer flow. To test the well-functioning of the solver, it was utilized for a simple anisotropic free flow case. Its behavior was found impressive. High hopes were expected for contact applications. When testing it in a simple 2D case a draw-back was noticed. The directional formulation does not take into account the critical zone (on the corners) where points may have an ill-defined normals. A rotated consistencies resulted near the corner and limited the proper progress and sliding of the material. To avoid this matter, this crucial zone should be properly managed. Several steps are proposed such as a new formulation using Navier slipping conditions.

To conclude, the Eulerian approach express high potential to compete, in the long run, with a perfectly mastered software such as *Forge®*. Additional developments are needed. Still, its capacity is undeniable especially when taking into consideration that few works deal with these large deformation problems.

Note that during this study only materials with Newtonian behavior law were considered. Thermal effects were neglected as well. Which brings us to the next point: the recommendations and perspectives.

2 Recommendation and Perspectives

To recapitulate, future works should be directed towards :

◇ Studying the feasibility of multi-domain contact problem in this Eulerian approach:

- The directional solver should be revisited and explored to asses if it will be ever able to describe properly contact problems. Defining the normal using the level set of the deformable body leads to a rotated consistency on the corners

preventing the sliding. Redefining the normal in a new manner could be the solution to this limitation.

- In term of friction, we were able to represent sliding. However, representing the consistency in the boundary layer was challenging. The adopted method relies heavily on the lubricant consistency and thickness (η_{lub}, ε). An alternative method is to introduce Navier slip/no-slip conditions. It does not depend of the mesh (especially the thickness interface ε). However the use of such method requires an explicit representation of the contact zone. In other words, the zero isovalue should be reconstructed to be defined on nodes as in Lagrangian formulations.

◇ Varying Norton-Hoff laws with a sensibility parameter m different then 1.

We should note that in order to pursue this point, the Norton-Hoff law may need need regularization. It can be accomplished using a hyperbolic tangent function for instance.

◇ Taking into account thermal effects by coupling diffusion-convection equations with the current mechanical system.

In an Eulerian framework, coupling the thermal and mechanical problem can be more beneficial than in a Lagrangian formulation. It takes into account different interacting heterogeneities and enables us to study/follow the thermal transfers easier.

◇ Optimizing different algorithm affecting the computing time and the parallel efficiency such as the convection solver assembly and the balancing load algorithm.

- The assembly time of the LevellerT solver is very comparable to a Stokes problem. In fact, the convection problem is relatively small when compared to the mechanical problem (Stoke or Navier-Stokes) and should not consume a large amount of time. We recommend thus, rewriting the solver in order to optimize the assembly time.
- The parallel strategy of the adaptation algorithm should be improved as well. In a parallel framework, the adaptation is executed twice. In addition, the mesh generator acts locally on nodes and therefore needs an excessive interface displacement (interfaces between cores). The latter strategy should be revisited to reduce its high cost.

Modélisation des problèmes de grandes déformations multi-domaines par une approche Eulérienne monolithique massivement parallèle

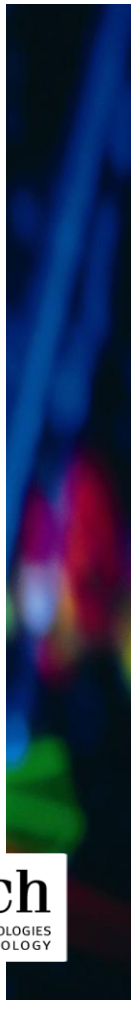
RESUME : La modélisation des problèmes multi-domaine est abordée dans un cadre purement Eulérien. Un maillage unique, ne représentant plus la matière, est utilisé. Les différentes frontières et leur évolution sont décrites via des outils numériques tels que la méthode Level Set. Les caractéristiques locales de chaque sous domaines sont déterminées par des lois de mélange.
Ce travail est une des premières tentations appliquant une approche Eulérienne pour modéliser de problèmes de grandes déformations. Dans un premier temps, la capacité de l'approche est testée afin de déterminer les développements nécessaires.
Le frottement entre les différents objets est géré par un lubrifiant ajouté dans une couche limite. Combinée avec une technique d'identification, une nouvelle loi de mélange quadratique est introduite pour décrire la viscosité du lubrifiant. Des comparaisons ont été effectuées avec Forge® et les résultats sont trouvés satisfaisants. Pour traiter le contact entre les différents objets, un solveur directionnel a été développé. Malgré que les résultats soient intéressants, il reste le sujet de nouvelles améliorations. La scalabilité de l'approche dans un environnement massivement parallèle est testée aussi. Plusieurs recommandations ont été proposées pour s'assurer d'une performance optimale. La technique du maillage unique permet d'obtenir une très bonne scalabilité. L'efficacité du parallélisme ne dépend que de la partition d'un seul maillage (contrairement aux méthodes Lagrangiennes). La méthode proposée présente des capacités indéniables mais reste loin d'être complète. Des pistes d'amélioration sont proposées en conséquence.

Mots clés : approche Eulérienne monolithique, Level Set, adaptation de maillage anisotrope, loi de mélange quadratique, solveur directionnel, calcul parallèle.

Modelling multi-domain large deformation problems using an Eulerian monolithic approach in a massively parallel environment

ABSTRACT : Modeling of multi-domain problems is addressed in a Purely Eulerian framework. A single mesh is used all over the domain. The evolution of the different interacting bodies is described using numerical tools such as the Level Set method. The characteristics of the subdomains, considered as heterogeneities in the mesh, are determined using mixture laws.
This work is one of the first attempts applying fully Eulerian Approach to Model large deformation problems. Therefore, the capacity of this approach is tested to determine necessary developments. The friction between the different objects is managed by adding a boundary layer implying the presence of a lubricant. Combined with an identification technique, a new quadratic mixture Law is introduced to determine the lubricant viscosity. Comparisons have been performed with Forge® and results were found satisfactory. To treat the contact problem between the different objects, a directional solver was developed. Despite the interesting results, it remains the topic of further improvements. The scalability of the approach in a massively parallel environment is tested as well. Several recommendations were proposed to ensure an optimal performance. The technique of a single mesh guarantees a very good scalability since the efficiency of parallelism depends of the partition of a single mesh (unlike the Lagrangian Methods). The proposed method presents undeniable capacities but remains far from being complete. Ideas for future Improvements are proposed accordingly.

Keywords : monolithic approach, Level Set, anisotropic mesh adaptation, quadratic mixture law, directional solver, parallel computing.



MINES ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY