Modeling and simulation of multi-fluid systems. Applications to blood flows

Vincent Doyeux

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Physique**

Arrêté ministériel : du 7 août 2006

Présentée par

Vincent Doyeux

Thèse dirigée par **Mourad Ismail**

préparée au sein **du laboratoire Interdisciplinaire de Physique**
et de **l'école doctorale de Physique de Grenoble**

Modélisation et simulation de systèmes multi-fluides. Application aux écoulements sanguins.

Thèse soutenue publiquement le **28 Janvier 2014**,
devant le jury composé de :

M. George Biros
Professeur, University of Texas at Austin, Rapporteur
M. Jeffrey Morris
Professeur, City University of New York, Rapporteur
M. Emmanuel Maitre
Professeur, Grenoble Institute of Technology, Examinateur
M. Bertrand Maury
Professeur, Université Paris Sud, Examinateur
M. Mourad Ismail
Maître de Conférences, Université Joseph Fourier, Directeur de thèse
M. Christophe Prud'homme
Professeur, Université de Strasbourg, Co-Directeur de thèse
M. Philippe Peyla
Professeur, Université Joseph Fourier, Co-Directeur de thèse

Table of Contents

Remerciements

Tout au long de la réalisation de cette thèse, j'ai pu compter sur le soutien de nombreuses personnes tant au niveau professionnel que personnel. Je tiens à les remercier ici.

Je voudrais tout d'abord remercier Mourad Ismaïl pour m'avoir fait confiance et proposé un stage en première année de master dans ce labo que je n'ai finalement plus voulu quitté. Merci pour m'avoir proposé une thèse dès ce moment là alors que je ne n'avais jamais imaginé cette option, puis, pour m'avoir accompagné jusqu'à la fin de mon doctorat en cherchant toujours à m'aider à faire les bons choix pour mon avenir professionnel. J'espère que nous pourrons nous revoir dans le futur pour prendre un verre ensemble, pour parler un peu de maths et beaucoup de tout le reste... J'en profite aussi pour remercier Alice, Gabriel, et Léonore que j'ai eu plaisir à croiser pendant ces 4 ans, sans oublier Maya que je connais depuis le début grâce à ses incursions au labo !

Je remercie aussi Christophe Prud'homme pour m'avoir suivi durant ma thèse, pour avoir été disponible très souvent pour répondre à mes questions et me guider face aux difficultés que j'ai pu rencontrer. Merci pour son enthousiasme qui a été un moteur certain pour moi. Merci aussi de m'avoir intégré dans l'équipe Feel++ . Grâce à lui j'ai pu acquérir une expérience extrêmement enrichissante en maths appli, programmation, éléments finis et en conception de logiciels, quelques uns des éléments de ma thèse qui m'ont le plus passionnés.

Merci aussi à Philippe Peyla pour les discussions scientifiques que nous avons pu avoir et pour lesquelles il a toujours été disponible. Merci aussi pour ses conseils d'ordre général durant le master.

Je remercie également les rapporteurs de cette thèse, Jeffrey Morris et George Biros, ainsi que les examinateurs Emmanuel Maitre et Bertrand Maury qui ont pris le temps de s'intéresser à mon travail et m'ont fait d'intéressantes suggestions. Je les remercie également d'avoir été présents, physiquement ou virtuellement le jour de ma soutenance.

Je tiens à remercier toute l'équipe DYFCOM, qui est une équipe dans laquelle j'ai eu beaucoup de plaisir à travailler. Être dans une équipe dynamique dans laquelle se côtoient expérimentateurs, théoriciens et numériciens venant principalement du monde de la physique mais aussi des mathématiques et de la biologie, donne lieu à de très belles collaborations ainsi qu'à des discussions extrêmement riches. J'espère pouvoir retrouver des environnements aussi diversifiés que cette équipe dans le futur.

Je voudrais également remercier particulièrement les personnes qui m'ont aidées à préparer l'oral pour obtenir ma bourse de thèse. En effet, dix jours après mon retour de stage aux États-Unis, il me fallait présenter un projet sous forme d'oral très court, un exercice difficile pour lequel j'étais un peu perdu au départ. Je suis reconnaissant à l'équipe d'avoir fait bloc et de m'avoir apporté une aide extrêmement précieuse à ce moment là. Pour cela, je remercie particulièrement Salima, Philippe, Gwennou, Thomas, Pierre-Yves, et Mourad.

Merci aussi à Thomas, Gwennou, Chaouqi, Marine et Alexander chercheurs et post docs du laboratoire avec qui j'ai pu avoir de nombreuses discussions scientifiques intéressantes tout au long de ma thèse. J'en profite pour remercier également Silvia Bertoluzza et Emmanuel Maitre avec qui j'ai pu également discuter de mon sujet à plusieurs reprises et qui m'ont donné de précieux conseils.

Une thèse de simulation s'effectue difficilement sans un bon support technique informatique. Je remercie Philippe Beys pour sa patience, sa réactivité et sa gentillesse pour toutes les (nombreuses) fois ou je suis venu lui demander d'installer des paquets manquant pour faire tourner mes applications.

Une des grandes forces de cette équipe est aussi son nombre important de thésards et post-docs. Il est tellement important de ne pas se sentir seul dans les moments de galère qu'on connaît tous durant une thèse, et pouvoir partager ses doutes et ses difficultés avec des personnes ayant les mêmes préoccupations est souvent un grand soulagement. J'ai beaucoup apprécié la bonne ambiance entre les thésards et post-docs du laboratoire tout le temps de ma thèse. Un grand merci aux doctorants de l'équipe DYFCOM ainsi qu'à ceux du LIPHY en général que j'ai pu croiser et qui m'ont beaucoup apporté, pour les bons moments passés à la cafèt, durant les repas, un match de foot, ou autour d'une chartreuse en Chartreuse... Ils sont trop nombreux pour que je les nomme tous ici, je sais qu'ils se reconnaîtront, alors encore une fois, un grand merci à eux tous.

Pour continuer avec les personnes m'ayant aidé moralement tous les jours au labo, je veux absolument remercier les personnes avec qui j'ai partagé le *bureau 330*, ou, *le bureau du fond* pendant ces quelques années. Les pauses café / thé, les pauses *potins* avec eux, ou simplement leur bonne humeur ont rendu ma thèse agréable au jour le jour. Pour tout ça, un grand merci à (dans leur ordre d'apparition au bureau) Aparna, Najim, Maeva, Xavier, Vassanti et Marie Cecilia.

Et pour finir avec mes remerciements au laboratoire, je me dois d'avoir une pensée spéciale pour les couchers de soleils sur la grande lance de Domène et le grand Colon enneigés. Il sera difficile dans mes futurs environnements professionnels de trouver un cadre aussi beau dans lequel travailler.

Je remercie également toute l'équipe Feel++ avec qui j'ai eu plaisir à interagir durant ces années, des petits coups de mains rapides instantanés aux longues recherches de bugs, j'ai toujours pu trouver une aide précieuse venant des autres développeurs. Merci aussi pour les bons moment passés lors de conférences, du CEMRACS, et de nos voyages à Strasbourg. Un grand merci à Vincent C., Christophe P., Abdoulaye, Cécile, Stéphane

V., Gonzalo, Ranine, Vincent H., Alexandre, Jean-Baptiste, Guillaume, Christophe T., Marcella et Stéphane P.

Je remercie mes colocs et amis Benj, Mariya et Mathilde avec qui j'ai pu passer de très bons moments. Merci aussi à mes amis de facs souvent devenus thésards eux aussi. Merci en particulier à Guillaume, Jeanne, Benj et Mariya (encore), autres labos, autres problématiques, différentes difficultés, mais qu'il est agréable de se plaindre autour d'une bonne bière ! Un grand merci aussi à Emilie, pour son amitié et son écoute. Après les parcs du Texas, les Ecrins, les Calanques, la Corse, j'espère qu'on créera de nouveaux souvenirs dans *ton* Colorado ou autour d'un concert de Blues dans *mon* Texas !

Je remercie également mes copains d'avant cette période, en particulier les *cond'laB200* qui m'ont toujours encouragé, en particulier pour mes moments de procrastination. J'espère qu'on pourra continuer encore longtemps les discussions cinéma et les sorties montagne.

Je dois un très grand merci à Sandrine qui m'a portée (et supporté) durant ces années. Je suis conscient de la patience qu'il faut certains jours pour vivre avec un geek qui passe autant de temps devant son pc au sacrifice quelques fois des loisirs communs. Merci de me rappeler de sortir la tête du travail et merci pour tous les bons moments qu'on a passés durant nos années de thèse, en espérant que ceux qui nous attendent soient encore meilleurs.

Enfin un énorme merci à toute ma famille qui m'a supporté et poussé tout au long de mes études et particulièrement durant les moments difficiles que nous avons pu connaître durant la période de ma thèse. Jamais je n'aurais pu arriver jusqu'ici sans ce soutien inconditionnel. Merci encore.

Notations

The Level set related quantities

Symbol		Description	Page
ϕ	:	the level set function	33
ϕ_h	:	the discrete level set function	36
ε	:	the half interface thickness	33
H_ε	:	the smoothed Heaviside function of thickness 2ε	33
δ_ε	:	the smoothed delta function of thickness 2ε	33
$\boldsymbol{n}$	:	the normal of the level set field	35
κ	:	the curvature of the level set field	35
$\sigma, \boldsymbol{\beta}, f$	:	the coefficients of the advection equation	36

The fluid, vesicles and flow related quantities

Symbol		Description	Page
R_e	:	the Reynolds number	64
$\boldsymbol{u}, p$	:	the velocity and pressure of the fluid	
$\boldsymbol{u}_h, p_h$	:	the discrete versions of the velocity and pressure	
$\boldsymbol{\sigma}$	:	the stress tensor	64
$\boldsymbol{D}(\boldsymbol{u})$	:	the strain tensor	64
μ	:	the viscosity of the fluid	64
ρ	:	the density of the fluid	70
μ_ϕ	:	the viscosity of the fluid defined according to the level set function	72
ρ_ϕ	:	the density of the fluid defined according to the level set function	72
σ	:	the surface tension	76
$\boldsymbol{f}_\phi$	:	the external forces defined according to the level set function	72
E_b	:	vesicle bending energy	106
k_b	:	bending modulus	106
$\boldsymbol{F}_\mathrm{b}$	:	bending force	106
E_el	:	the elastic energy of the membrane	114
$\boldsymbol{F}_\mathrm{el}$	:	the elastic force	115
λ	:	Lagrange multiplier insuring the inextensibility of the membrane	120
α	:	reduced area	122
C_n	:	the confinement	91, 133
Q_i	:	the flow rate in the branch i	170
N_i	:	the particle flow rate in the branch i	172

y_f	:	the fluid separating line	170
y_p	:	the particles separating line	174
Φ	:	the volume fraction of a suspension	217
μ_{eff}	:	effective viscosity of a suspension	217
(L, l)	:	size of the box where the effective viscosity is computed	223

Domains and spaces

Symbol		Description	Page
Ω	:	the whole domain	33
$\partial\Omega$	:	the boundary of Ω	
Ω_1	:	the *outside* domain, in which $\phi > 0$	33
Ω_2	:	the *inside* domain, in which $\phi < 0$	33
Γ	:	the interface $\phi = 0$	33
$L^2(\Omega)$	:	the square-integrable function space	65
$[L^2(\Omega)]^d$	:	the square-integrable function space of dimension d	65
$L^2_0(\Omega)$	:	the square-integrable function space with null mean pressure	65
$H^1(\Omega)$	:	the Sobolev function space	65
$[H^1(\Omega)]^d$	:	the Sobolev function space of dimension d	65
$H^1_0(\Omega)$	:	the Sobolev function space with vanishing value at boundary	65
C^0	:	the set of the continuous functions	
$\mathbb{R}^k_h$	:	the discrete finite element space depending on mesh size h and spanned by Lagrange polynomials of degree k	36

Mathematical and numerical symbols

Symbol		Description	Page
S_{gn}	:	the sign function	
$[\![f]\!]$	:	the jump of the function f across the face of an element	40
π_{L^2}	:	the L^2 projection	107
$[\mu]$	:	the physical dimensions of the quantity μ	
${}^t(\boldsymbol{v})$	:	the transposed of the vector $\boldsymbol{v}$	
$\boldsymbol{I}_{\text{d}}$	:	the identity matrix	262
$\nabla_{\text{s}} \cdot \boldsymbol{u}$	:	the surfacic divergence of $\boldsymbol{u}$	262
δ	:	the Dirac function	
χ	:	the characteristic function	51
h	:	the mesh size	
dt	:	the time step	
d	:	the space dimension, in this work, $d = 2$ or 3	

Introduction

Contents

i Introduction

Français

Le sang est un fluide au comportement complexe. Il est composé de plasma, un fluide dans lequel baignent des cellules : globules rouges, globules blancs et plaquettes. Ces différentes entités ayant des propriétés mécaniques riches, confèrent au sang des comportements très variés. Depuis très longtemps, les scientifiques essayent de décrire les écoulements du sang. Au 19$^{\text{ème}}$ siècle, Poiseuille étudiait l'écoulement du sang dans les veines et les capillaires. Il tenta de décrire le sang comme un fluide homogène, ce qui le conduisit à découvrir, en même temps que Hagen, la loi qui porte désormais leurs noms.

Cette loi décrit la vitesse d'un fluide dans une conduite cylindrique lorsqu'on lui applique une différence de pression. De plus, pour un rayon de tube donné, le rapport entre la différence de pression appliquée au tube et la vitesse maximum acquise par le fluide donne une définition de la viscosité. Même si cette loi est encore utilisée de nos jours pour des fluides homogènes, elle n'est pas suffisante pour décrire l'écoulement du sang dans de petits vaisseaux.

Plus tard, Fåhræus et Lindqvist découvrirent que la viscosité d'un même échantillon de sang était plus faible lorsqu'il circulait dans des vaisseaux de très petite taille. Cet effet a été expliqué par le fait que les globules rouges ont une tendance à migrer vers le centre des vaisseaux, créant une zone proche des parois dans laquelle le flux sanguin peut s'écouler facilement, ce qui réduit la résistance globale de l'écoulement et fait apparaître la viscosité plus faible que dans de grands vaisseaux où cette zone sans globules est négligeable. Cet effet très connu illustre parfaitement le problème de l'étude de l'écoulement du sang.

Les propriétés telles que la viscosité ou la vitesse d'écoulement dans un vaisseau sont grandement influencées par les propriétés mécaniques individuelles des globules rouges (99% des cellules présente dans le sang), les interactions entre les globules, les interactions des globules avec la paroi des vaisseaux sanguins, et même les interactions entre les globules rouges et les autres cellules. De nos jours, une grande partie des études réalisées pour la compréhension des écoulements sanguins s'attache à comprendre ces phénomènes au niveau microscopique pour ensuite les appliquer à une description à plus grande échelle. De plus, l'intérêt grandissant pour les écoulements de fluides biologiques dans des micro canaux artificiels dans le but de créer des laboratoires sur puce augmente encore le besoin de connaissance des comportements microscopiques de ces fluides. Pour cela, de nombreuses expériences, théories et simulations sont développées. C'est dans cette dernière catégorie que s'inscrit cette thèse. En effet, depuis la fin du 20$^{\text{ème}}$ siècle la simulation numérique a pris une importance croissante dans tous les domaines de la physique et des mathématiques. L'évolution constante de la puissance des ordinateurs conduit les physiciens et mathématiciens à repenser constamment leurs modèles et les faire évoluer en conséquence. S'il paraissait impossible il y a une dizaine d'année de résoudre en même temps les équations régissant un fluide, les coupler avec les équations gouvernant les propriétés mécaniques d'un globule rouge, le tout dans une géométrie complexe, l'arrivée de super calculateurs et la démocratisation du calcul parallèle rendent aujourd'hui ces simulations possibles dans certaines mesures. La recherche de nouvelles méthodes numériques couplée à des codes de calculs performants est donc un des défis auquel se confrontent les scientifiques aujourd'hui. De plus l'utilisation de ces codes de calcul pour extraire de nouvelles lois physique est aussi un axe de recherche prenant une importance grandis-

sante dans les laboratoire et conduit physiciens et mathématiciens à travailler en étroite collaboration.

C'est dans ce contexte que se place cette thèse. L'objectif est de développer un cadre de calcul générique pour la modélisation et la simulation d'écoulements sanguins, et de l'utiliser dans un contexte de simulation en interaction avec des expériences. Le but d'avoir un environnement de calcul très générique est qu'il ne soit pas simplement un outil de simulation, mais aussi un "laboratoire d'expérimentation de méthodes numériques". En effet, l'objectif est de pouvoir tester quelques-unes des méthodes de calcul les plus intéressantes et les différents modèles d'écoulement sanguins. Lors de ce travail, deux stratégies différentes ont par exemple été testées pour assurer l'inextensibilité de la membrane des objets en suspension. Par soucis de généricité, un effort particulier a été fait durant le développement pour que le même code de calcul soit utilisable en deux et trois dimensions, sur une machine de bureau ou un cluster de calcul, sur un seul ou plusieurs processeurs. Ceci est rendu possible par l'utilisation d'une librairie d'éléments finis utilisant les avancées des dernières technologies informatiques ([1]MPI, méta-programmation et dernière norme du C++).

Durant ce travail, les méthodes numériques développées ont été vérifiées grâce à des simulations tests dans lesquelles la solution numérique est connue. Elles ont également été validées sur des simulations dans lesquelles les résultats physiques étaient connus par expériences, théories ou d'autres simulations.

Bien que les parties numériques de ce travail fassent partie du projet Feel++ *interactions fluide structure*, les problèmes physiques étudiés sont liés aux recherches de l'équipe *Dynamique des fluides complexes* (DYFCOM) du laboratoire interdisciplinaire de physique dans lequel ce travail a été effectué. Le champ de recherche de cette équipe est le comportement des fluides complexes en général et celui des fluides complexes biologiques en particulier. Dans ce contexte, l'équipe étudie entre autre la rhéologie et l'écoulement du sang par l'intermédiaire d'expériences, de développements théoriques et bien entendu de simulations. Les travaux numériques effectués dans cette thèse, avaient comme but, l'application à des problèmes étudiés au sein de l'équipe DYFCOM.

Ainsi, l'étude de l'influence du confinement sur la fréquence des oscillations d'une bulle a été guidée par les expériences d'O.Vincent [111] sur la cavitation de bulles dans un hydrogel.

Puis, une méthode de simulation d'objets rigides a été appliqué au problème de la répartition d'une suspension diluée de particules rigides lors de son passage dans une bifurcation micro-fluidique. En collaboration avec les expérimentateurs G.Coupier et T.Podgorski, l'effet de l'accroissement de la concentration de particules dans la branche recevant le plus grand débit a été expliqué par un effet géométrique de la répartition des particules dans le canal d'entrée. De plus, nous avons mis à jour l'existence d'une force poussant les particules vers la branche de plus bas débit et entrant en concurrence avec le précédent effet.

Enfin, la rhéologie d'une suspension de disques rigides dans un écoulement de cisaillement confiné a été étudiée. Nous avons pu confirmer l'influence décroissante de l'interaction entre les particule sur la viscosité pour des grands confinements. Cet effet avait déjà été observé en 3D numériquement et expérimentalement et est maintenant con-

[1] http://fr.wikipedia.org/wiki/Message_Passing_Interface

firmé en 2D. Durant ce travail, l'influence de la position des particules d'une suspension diluée relativement aux bords du domaine a aussi été exploré.

Tous les problèmes étudiés en utilisant la méthode de simulation d'objets rigides on été adaptés à l'étude de d'objets déformables. Dans un futur proche, ces méthodes seront utilisées pour explorer l'influence de la déformabilité sur ces différents phénomènes.

English

Blood is a fluid having a complex behavior. It is composed of plasma, a fluid in which are flowing cells: red blood cells, white blood cells and blood platelets. These different cells having rich mechanical properties, confer to blood diverse behaviors. For a long time, scientists have tried to describe blood flows. In the 19^{th} century, Poiseuille studied blood flow in veins and capillaries. He tried to describe blood as a homogeneous fluid, which led him to discover, at the same time as Hagen, the law which carries their names.

This law describes the velocity of a fluid in a cylindrical channel when one applies a pressure different at the extremities. Moreover, for a given channel radius, the ratio between the pressure difference and the maximum velocity of the fluid gives a measure of the viscosity. Even if this law is still used nowadays for homogeneous fluids, it is not sufficient to describe the flow of blood in very small vessels.

Later, Fåhræus and Lindqvist discovered that the viscosity of a given sample of blood is smaller when it flows in very small vessels. This effect has been explained by the fact that red blood cells tend to migrate toward the center of the channel, making a cell free layer around the vessel walls in which the plasma can circulate more easily. This reduces the global resistance of the flow and makes the viscosity appear smaller than in large channels in which the free layer zone impact is negligible. This famous effect illustrates the problem of the study of blood flow.

The properties such as the viscosity or the flowing velocity in a channel are greatly influenced by the individual mechanical properties of red blood cells (99% of the cells present in the blood), the interaction between the red blood cells, the interaction between the cells and the vessel walls, and even the interactions between the red blood cells and the other cells. Nowadays, a large part of the study devoted to the understanding of blood flow is trying to understand these phenomenons at a microscopic scale to apply them to a macroscopic one. Moreover, the increasing interest for the biological flows in artificial microfluidic devices with the goal to create lab on chip tools increases the need of knowledge of the microscopic behaviors of these fluids.

To this goal, many experiments, theories and simulations have been developed. This thesis is consecrated to this last category. Indeed, since the end of the 20^{th} century the numerical simulation took an increasing importance in all the fields of physics and mathematics. The constant evolution of computers leads physicists and mathematicians to re-think their models and make them evolve. Although it seemed impossible ten years ago to solve at the same time the equations governing a fluid, couple them with the one of the mechanical properties of red blood cells and do so in a complex geometry, the rise of super computers and the ready availability of parallel computing make these simulations possible today in some measure. The research of new numerical methods coupled to powerful codes is thus one of the challenge faced by scientists today. Moreover the use of these codes to extract new physical laws is also a research axis growing in laboratories and leads physicists and mathematicians to work in close collaboration.

It is in this context that belongs this thesis. The objective is to develop a generic numerical framework for the model and simulation of blood flows, and to use it in a context of simulation in close collaboration with experiments. The goal to have a very generic framework environment is that it is not simply a numerical tool to make simulation, but also a laboratory to experiment new numerical methods. Indeed, the objective is to be able to test some of the most promising techniques and different models for blood flow simulation. For example, during this work, two different strategies have been tested to insure the inextensibility of the cells membrane. For general purposes, a particular care has been taken during the development in order to be able to use the same code in 2D or 3D, on a cluster or on a single computer, on a single or several processors. This has been possible thanks to the use of a finite element library using some of the latest programming tools ([2]MPI, meta-programming, and recent C++ norms).

During this work, the numerical methods have been verified thanks to test simulations in which the numerical solution is known. They have also been validated on simulations for which the physical result was known by experiment, theory or other simulations.

Although the numerical part of this thesis takes place in the context of the Feel++ *fluid structure interaction* project, the physical problems studied are related to the researches of the *Dynamics of Complex Fluids* (DYFCOM) team of the interdisciplinary laboratory of Physics in which this work has been made. The research subject of this team is the behavior of complex fluids in general and the one of biological related fluids in particular. In this context, blood rheology and circulation are studied in the team, experimentally, theoretically and of course numerically. The numerical developments made in this work had as goals applications in the fields of interest of the DYFCOM team.

As a matter of fact, the study of the influence of the confinement on the oscillation frequency of a bubble has been suggested by the experiments of O.Vincent [111] on cavitation of bubbles in an hydrogel.

Then, a solid disk simulation method has then been applied to the problem of the splitting of a suspension of particles when flowing in a micro fluidic bifurcation. In collaboration with the experimenters by G.Coupier and T.Podgorski, the effect of the increasing concentration of particles in the high flow rate branch has been explained by the geometrical distribution of particles in the inlet channel. Moreover, a force pushing particles toward the low flow rate branch balancing the previous effect has been discovered.

Finally, the rheology of a suspension of solid disks in a confined shear flow has been made. We were able to confirm the effect of the decreasing influence of the interactions between particles on the viscosity for strong confinements. This effect has already been seen in 3D numerically and experimentally and is now also confirmed in 2D. During this work, the influence of the position of the particles of a dilute suspension relative to the walls of a confined shear flow has also been explored.

All the problems studied with the solid disk simulation method have been adapted to soft object simulations and will be used in a near future to explore the influence of deformability on these phenomenons.

[2] http://fr.wikipedia.org/wiki/Message_Passing_Interface

ii Vesicles as a model for red blood cells

Red blood cells (**RBCs**) are cells measuring about 7 µm of diameter and having no nucleus. The inside of a RBC is principally composed of hemoglobin and a cytoskeleton. The surrounding membrane is a bi-layer of phospholipids. Many proteins compose the membrane in a minor manner compared to the phospholipids as one can see in figure 1. A

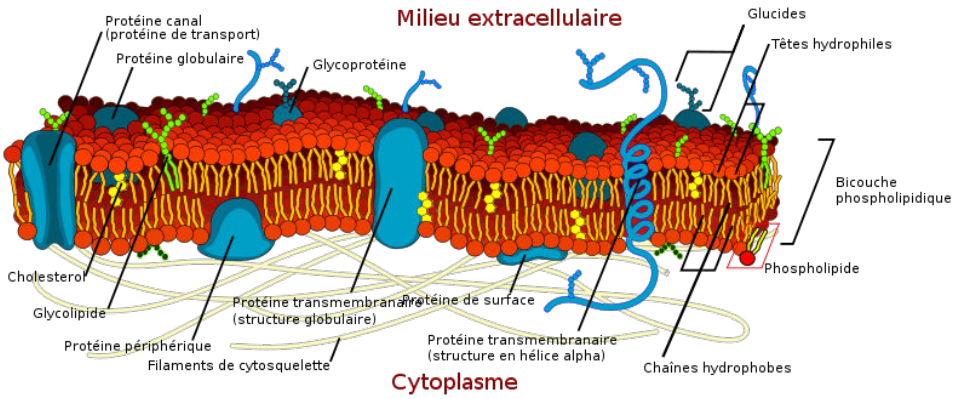


Figure 1: Scheme representing the membrane of a red blood cell.

common simple model of red blood cell (**RBC**) is a vesicle. A vesicle is a fluid drop surrounded by a bi-layer phospholipidic membrane as shown in figure 2. Generally, artificial vesicles are of the order of 10 µm. A vesicle does not include the cytoskeleton which adds

Figure 2: Scheme of a vesicle. The scale is not correct, the phospholipids are about 5 nm whereas the diameter of an artificial vesicle is about 10 µm

some elastic properties to the RBC. Nevertheless, a big part of the mechanical behavior of the RBCs can be reproduced by vesicles. Indeed, both vesicles and RBC membranes are resist strongly to stretching and both have the same curvature energy expression. One of the first theoretical works trying to model the vesicle goes back to 1982 with Keller and Skalak [99], and the first simulation arrived more than 10 years after that with [61] followed by the first experimental work one year after that [23]. Since then, many works have been done to understand the behaviors of these objects. From the numerical point

of view, the system of a single vesicle in a shear or Poiseuille flow has been studied widely and scientists start to get interested in many vesicle interacting systems or flowing in complex geometries. Many efforts are done in this sense, and many works are done to modify existing techniques. We give a short overview of the existing methods to simulate vesicles and their advantages or drawbacks.

iii State of the art of the simulation of vesicles

The simulation of a vesicle is a difficult task. It is part of the large field of the fluid structure interactions. Indeed, one has to deal at the same time with the fluid equations and the membrane of the vesicle having its own properties. Two fluids are involved in this kind of simulation. The fluids inside and outside the vesicle which are in the general case different. Thus, the fluid solver has to be able to deal with a viscosity ratio. The equations involved are the incompressible Navier-Stokes equations in the general case, which are nonlinear. Most of the time, in microcapillaries, one can consider the limit of the vanishing Reynolds number and solve the Stokes equations which are linear and thus easier to handle.
The model for the vesicle has three main properties. The first one is that the membrane resists stretching. It is modeled as inextensible, its local surface is conserved. The total volume of the vesicle is also conserved. The membrane can let solvent particles pass through it, but the osmotic pressure balance insures that in a small time scale, the same amount of particle is going in and out. Thus in a continuous description, the total volume of fluid inside the vesicle is conserved. The last property needed to be incorporated in the models is the fact that it costs some energy to bend the membrane. This energy has been derived by Canham and Helfrich respectively in [16, 43] and it is proportional to the square of the curvature of the membrane. Many different methods have been developed to deal with the membrane and the fluids. One can put them in three categories of methods. In the pure Lagrangian methods, only the membrane is discretized and one follows each discretization point individually. In Lagrangian / Eulerian methods, the membrane is discretized in a Lagrangian manner but the fluid has an Eulerian treatment. And finally, in the pure Eulerian methods both the membrane and the fluid are seen from an Eulerian point of view. Let us give a brief overview of some of these methods.

The pure Lagrangian methods

Principle of the boundary integral method

Under the pure Lagrangian methods, we can name all the variants of the boundary integral methods. They have been introduced for the vesicle problem by Pozrikidis in 2001 [85]. The principle is to use the Green tensor $\boldsymbol{G}$ of the Stokes equation and couple it to a force giving to the membrane all its properties. The Helfrich energy is derived to give the curvature force. A tension force is added with a huge energy value which makes the membrane unstreatchable, in other implementations, a Lagrange multiplier enforces the local area conservation exactly. A typical expression for the membrane force in 2D is given by:

$$\boldsymbol{f} = k_{\mathrm{B}} \left(\frac{d^2\kappa}{ds^2} + \frac{\kappa^3}{2} \right) \boldsymbol{n} - \xi\kappa\boldsymbol{n} + \frac{d\xi}{ds}\boldsymbol{t}$$

with κ the curvature of the membrane, $\boldsymbol{n}$ and $\boldsymbol{t}$ the normal and tangent unit vector to the membrane, s the arc-length coordinate and k_{B} the membrane bending rigidity. ξ is a tension accounting for the inextensibility. The velocity for a given point of the domain can be computed as an integral over the membrane boundary. Thus for each point $\boldsymbol{x}_0$ on which one wants to compute the velocity $\boldsymbol{u}(\boldsymbol{x}_0)$, it is given by:

$$\boldsymbol{u}(\boldsymbol{x}_0) = \boldsymbol{u}_0(\boldsymbol{x}_0) + \frac{1}{\mu}\int_{membrane} \boldsymbol{G}(\boldsymbol{x}-\boldsymbol{x}_0)\boldsymbol{f}ds$$

where $\boldsymbol{u}_0$ is the imposed velocity flow without vesicle. Here, $\boldsymbol{G}(\boldsymbol{x}-\boldsymbol{x}_0)\boldsymbol{f}$ is the product matrix-vector resulting in a vector. A more general formulation exists providing the ability to deal with a different viscosity inside and outside of the vesicle.

To have the evolution of a vesicle in a flow, one has to compute the velocity of each point of the membrane which depends on an integral of all the other points of the membrane. Then, the membrane position is updated by integration of the velocity. One of the advantages of this method is that for a 2D problem, the problem is reduced to only a 1D problem (since only the membrane has to be discretized) and only integrals have to be calculated. However, the fact that the velocity of each point of the membrane depends on all the other points makes the complexity of algorithms vary as N^2 where N is the number of discretization points. However, this complexity can be decreased to $N\log(N)$ thanks to a fast multi-pole method [96].

The tensor $\boldsymbol{G}$ can be given for an infinite Stokes fluid, or simple geometry cases like a semi-infinite fluid (with one wall). This does not make impossible to add other walls, but one has to integrate over all the walls by addition to the integration over the membrane which can be costly in the case of a geometry with many walls. The method is well suited for simulations where the positions and shapes of the particles are important since they are computed at every time step. They are also good to do rheology since the viscosity of a suspension can be computed with only the knowledge of the velocity on the walls of the domain. However, in a simulation where one would like to see for example the velocity field in all the domain in a complex geometry, one would have to compute an integral for each point of the domain, running on all points of the complex geometry boundary and all points of the membrane. This is one of the limitation of this method. It is however very accurate and efficient for simulation in simple flows and is probably the method which gave the most numerical results on vesicles until now. Let us present briefly some of the implementations and results obtained with this method.

The use of the boundary integral method in the literature

In 2009, Veerapaneni et al. [108] have used the boundary integral method where the Fourier transform of the arc-length is used to compute derivatives of the shape (normal and curvature). They performed a 2D simulation of 256 highly deformable vesicles in an unbounded Poiseuille flow. The method has been extended to the 3D case in [109] and [117, 116]. The boundary integral method has been one of the most used methods and led to many progress in the understanding of vesicle behaviors. In [54] the authors used a 2D boundary integral method to explore the lateral migration of a vesicle in a Poiseuille flow. In [57] the authors explored the different shapes of a vesicle in Poiseuille flow and

the influence of the relevant parameters. They were able to construct a phase diagram out of these simulations. The differences between 2D and 3D calculations in Poiseuille flow have been explored in [55]. The rheology of a single vesicle in a shear flow has also been explored in [37] where some comparisons with the phase field method have been made. The rheology of a dilute suspension of vesicles in a curved flow have been studied in [38].

Some other works have been using the Lagrangian point of view for both the fluid and the membrane. For example, in [79], the authors use a particle collision dynamics to explore the shapes of vesicles in capillary flows. In this method, the fluid is seen as many particles interacting by collision and the membrane is meshed independently.

The Lagrangian / Eulerian methods

The fact that the boundary integral method can not handle Navier-Stokes equation and the difficulty to use it in complex geometries lead researchers to try another strategy in which the fluid is seen in an Eulerian way which brings the possibility to use the huge literature on classical fluid solver. By keeping an explicit discretization of the membrane, the hybrid Lagrangian / Eulerian methods have been created. Indeed, in the Lagrangian / Eulerian methods, the membrane is discretized and followed explicitly in a Lagrangian manner, whereas the fluid equations are solved on a separate grid following an Eulerian point of view.

The immersed boundary method

Under these methods, the immersed boundary method is one of the most popular. It has been introduced by Peskin [82] in 1977 for general fluid structure interaction problems and has been applied to the flow of vesicles in 2007 by Zhang et al. [115]. The principle of the immersed boundary method is to have two different discretization grids for the fluid and the membrane. The membrane properties are computed on the membrane grid like the Helfrich force and the force responsible for the inextensibility. Then some smooth delta-like functions having a thickness bigger than the step of the fluid grid are used to distribute these forces on the fluid grid. Then the fluid problem is solved by taking into account the membrane through the forces applied on it. The method to solve the fluid problem can be taken as any usual fluid solver : Fourier Transform, lattice Boltzmann method, finite differences, finite element or finite volume. Finally, the velocity of each point of the membrane is computed by interpolation of the solution found by the fluid solver on the membrane grid. The membrane position is finally updated according to this velocity. This method can handle Stokes, Navier-Stokes or more complicated non Newtonian flows.
There is no limitation for the method to work in 2D and 3D. The method can also work in parallel if the fluid solver supports parallel computing. One can see for example [75] in which an immersed boundary method has been carried on in parallel. The fluid motion was in this case solved using the finite volume method. There can be many variants of the immersed boundary method, with different fluid solvers and schemes in integrating the position of the membrane and also different methods to calculate with accuracy the curvature of the membrane which is a crucial point. The method can also work in a complex geometry if the fluid solver provides this possibility. One of the drawback of

the immersed boundary method is that the interpolation step of the velocity on the interface grid leads to numerical error. This can cause a loss of mass of the vesicle. Special care has also to be taken when some particles are entering or leaving the domain. Indeed, since there is a Lagrangian knowledge of the interface, they have to be destroyed or reconstructed explicitly at the boundaries. The last drawback of this method is the difficulty of the method to handle a different viscosity inside and outside a vesicle. In such case, one has to explicitly locate the points of the fluid mesh being inside or outside the vesicle. This method gives a lot of results and many different implementations of it have been made.
In the first implementation [115] the authors used a lattice Boltzmann method as fluid solver and shown 2D simulations of a vesicle in Poiseuille and shear flows. Kim [60] used an immersed boundary method in which the fluid was solved using Fourier transforms which limits the simulation to periodic boundaries. They validated their results on the equilibrium shapes of vesicles and on the simulation of vesicles in a shear flow. They also performed simulation with 56 vesicles. In [35], the authors used a finite element solver and shown the ability of their code to work properly in 3D. In [53] the authors used a lattice Boltzmann method to solve the fluid equations in 2D and validated their code on the equilibrium shapes on vesicles. They have also shown the dependency of the tank treading angle on the confinement of the vesicle.

The other methods

There exist also some other methods, which can be classified as Lagrangian / Eulerian methods. In [11], a deformation field for the membrane is introduced and advected in time. The problem of finding the coupled velocity, pressure and deformation field while taking into account the particle is solved in an Eulerian manner. But at each time step, the interface is localized and the mesh is adapted so that mesh points are always present at the exact location of it. In this sense, one can say that there is a Lagrangian handling of the interface. In other works [107, 47], the coupling between the fluid and the membrane is not done by the forces exerted on the fluid, but by a force acting on the membrane discretization points. That is to say, at the step of updating the membrane points, a force is added to mimic the membrane properties. This is different from the immersed boundary method in which the force is also computed on the membrane points but is transmitted to the fluid thanks to the delta-like functions. In such implementation, the membrane points have generally some thickness and are seen as rigid particles embedded in the fluid. The membrane is thus seen as a particle *necklace*.

The pure Eulerian methods

To avoid the interpolation problem of the velocity at membrane points introduced by the immersed boundary method, to have the possibility to handle easily a viscosity ratio and to handle also more easily the vesicles at the boundary of the domain, researchers have introduced some pure Eulerian methods in which both the fluid and the membrane are lying on the same grid.

Phase field and level set methods

The two pure Eulerian methods which have been adapted to the simulation of vesicles are the *phase field* method and the *level set* methods. They both present many similarities. The principle is to define a field on the mesh which accounts for the position of the interfaces. From this field, delta-like and Heaviside-like functions are defined which make possible to differentiate the inner and outer fluid, distribute interfacial forces, and define different viscosities for each fluid. The normal and curvature of the interfaces are also computed from this position field.
The fluid equations are solved on the same mesh and the membrane is taken into account thanks to the forces added on it. The difference between the phase field method and the level set method resides on the choice of the function representing the position of the interfaces. In the phase field method, the function representing the two fluids is equal to 1 in one fluid, -1 in the other one and varies smoothly from one to the other. Whereas for the level set method, the representation of the interface is made by a signed distance function to the interface. The level set function carries more information than the phase field one, but it needs to be reset to a distance function often because the advection does not preserve the distance function property of the field. The methods can work in 2D and 3D and can be carried in principle on multiple processors. The fluid solver can be Stokes, Navier-Stokes or more complicated non-Newtonian solver. Complex geometries can be also handled since the fluid solver is able to deal with it. However, one of the drawbacks of pure Eulerian methods is that, since the membrane is not independently meshed, the number of discretization points around it might be low. Indeed, if the mesh size is the same everywhere in the domain, one has to use very small mesh size to have a lot of discretization points around the membrane which might be costly. An alternative is to use a mesh adaptation algorithm but the need to project a solution from a mesh to another one arises. Usually, to obtain a projection with a good accuracy, a multi step prediction - projection is needed [12, 62]. Another drawback which is in general attributed to the level set methods is that the advection and procedure to reset the level set to a signed distance field leads to some mass loss. This can be reduced by improving the discretization or using conservative schemes but it might still be an issue for long time simulations.

Eulerian methods for the simulation of vesicles

A phase field method to do simulation of vesicles has been introduced by Biben and Misbah in [7] for 2D simulations. It has been then extended to three dimensions in [8]. In these works, a tension field is searched to penalize the extensibility of the membrane. The problems were solved using finite difference methods. Later, in [20, 21], Cottet et al. used a level set field to simulate an inextensible object in flow. They showed that the stretching of the interface was recorded in the level set field and derived a force out of it to insure the inextensibility of the membrane. This force has been used in the PhD thesis of Milcent [76] combined with a force accounting for the curvature to simulate vesicles. The simulations have been made both in 2D and 3D. A comparison between level set and phase field methods for the simulation of vesicles has then been made in [70] showing all the similarities between them. Another level set strategy has been developed by Laadhari [62] during his PhD where the inextensibility of the membrane where insured by a Lagrange multiplier. The simulations were made in 2D with a finite element method for which a

mesh adaptation have been developed to follow more precisely the interface. A level set method in which both the level set and its gradient are advected has been presented in [92]. Moreover the authors presented a 4 step projection solving Navier Stokes equation while searching for a surface tension term imposing the inextensibility of the membrane.

iv Contributions and outline of this thesis

The goal of this thesis was to develop a generic framework for the simulation of vesicles in flow with a maximum of generality to be able to test different models in the future and compare them. A special care was also taken to make the framework efficient and easy to handle in order to be applied to real physical problems. Several verifications and validations have been made. The second goal of the thesis was to prepare some physical studies on a simpler model of rigid circular particle with the goal to be applied in the future to vesicles and red blood cells.

First part : the numerical methods

The first part of this thesis is dedicated to the description and the tests of the numerical methods we used.

First chapter

The numerical framework for the simulation of vesicles had to be able to capture vesicles with non-unit viscosity ratio, to be easily coupled with Stokes, Navier Stokes or other fluid solver, and to be easy to run in complex geometries. Following the previous state of the art on the numerical methods available for vesicles, we choose a level set method which includes all these ingredients. We also choose a finite element method because of the great number of theoretical results available in the literature compared to the finite difference or finite volume methods. Moreover, the finite element method allows us to use complex geometries. The promising work of A. Laadhari [62] on the simulation of a vesicle with level set method solved by finite element has also been an ingredient for our choice.

Consequently, the first chapter of this thesis is dedicated to the level set method. We present the strategy we used to create the level set framework, particularly the stabilization methods that we used for the advection of the level set field. We also present the different strategies of reinitialization of the level set field to a distance function, their limitations and advantages. We then show our results on a verification test which is a famous numerical benchmark: the solid rotation of a slotted disk. Finally we show a simple application of the level set which does not need any fluid solver: the detection of boundaries in an image.

Second chapter

For the sake of generality, we choose to decouple the fluid solver and the level set advection solver. That is to say, instead of solving the fluid equations and the advection of the level set field in the same problem (*i.e* in the same matrix for the algebraic point of view),

we choose to solve them one after the other. On the coding part, both fluid and level set framework are independent and separate C++ classes that we can plug into each other at will. The goal of having this separation is to be able to couple any fluid solver which is implemented in our library to the level set framework. For example during this thesis, two different fluid solvers have been used, a simple Stokes solver and a Navier-Stokes solver.
Thus, in the second chapter, we introduce the fluid equation and their discretization with the finite element method. Then we show how the coupling between the fluid framework and the level set framework is done to create a two fluid flow system. We extend this coupling to a multi-level set framework to show that it is also possible to have an N-fluid flow system with N greater than two. After that, we present a verification on a precise numerical benchmark for two fluid flow: the rising of a viscous drop in a fluid. Finally, we show a validation on a physical problem which is also a problem suggested by experimental physicists in our group: the oscillation frequency of a bubble in a fluid viscous fluid. Indeed, we have shown that the oscillation frequency of a bubble in a fluid is not influenced by its confinement.

Third chapter

Although the goal of this thesis was the simulation of vesicles, we wanted to show that it is also possible to deal with simpler objects with the same framework. Indeed, in a physical applications context, most of the time, to understand the precise role of the deformability of the vesicles, one would have to understand first the phenomenon with rigid spheres or disks. Thus, being able to simulate efficiently these kind of objects is a nice feature offered by our framework.
This is why the third chapter is dedicated to the simulation of solid objects in flow. We first make a brief review of the existing methods dedicated to this kind of simulations. Then we show that two of them are closely linked: the fluid particle dynamics and the penalty method. Finally we show that the two fluid flow framework presented in chapter two can be easily used for the simulation of solid objects. We also show that, from the fluid solver point of view, the method is equivalent to a fluid particle dynamics or a penalty method which brings all the theoretical development made for these methods to our framework. The only difference with these methods is that with level set method, the particles are followed in an Eulerian point of view. We then discuss the advantages and drawbacks compared to the classical Lagrangian point of view used in this context.

Fourth chapter

The fourth chapter is dedicated to our simulation method for vesicles. We start by describing the curvature force and show that it needs the knowledge of the curvature and its derivatives with a good accuracy. Thus, we present two strategies to obtain this information in finite element / level set context which are: increasing the polynomial order of approximation of the finite element bases, or smoothing the derivative fields. We then present in detail two different strategies to insure the inextensibility of the membrane which have been found in the literature and adapted to our framework. We finally make validation simulations on known results of vesicles which are the equilibrium shapes in

a fluid at rest, the tank treading motion, and the tumbling motion. Finally we compare both methods.

Second part : the implementation

The implementation of the numerical methods presented in the first part was an important piece of this work. The second part of this thesis is dedicated to the implementation of some particular ingredients needed to achieve the simulation of vesicles under flow.

Fifth chapter

The numerical framework that we developed has been based on the finite element library Feel++ . A special care has been taken to make all the different ingredients needed for the simulation of vesicles as general as possible both on the methodology and implementation level. By generality on the methodology level, we mean that we tried as much as possible to keep every development valid both in 2D and 3D, at order one and higher, and valid on a single or several CPU's. The library Feel++ being developed in this spirit, most of the time, the proper use of the standard features of the library leads to such genericity. However on few occasions, some specific development had to be made to keep the code valid. By generality at the implementation level, we mean that we tried to make the codes as re-usable as possible. In this spirit, most of the tools developed for this application are meant to become some part of the Feel++ library. Thus, a special design needed to be done properly to ease the usability and future maintainability of the codes. To this goal, the object oriented paradigm included in C++ has been a powerful tool. For example the solution of the advection equation framework, the level set framework, the fluid framework are all C++ classes. Moreover some special tools helped to build really independent frameworks, like the [3]`boost::program_options` which allows each level set or fluid solver object to have its own external options when such object is created.
The fifth chapter is dedicated to the implementation tools needed for the simulation of vesicles in flow and which have been designed to be included in Feel++ . We first describe the projection operator, which is a small useful tool to ease the use of several projection methods. Then we explain how the level set class works, what are the essential features and how to use them in a two fluid flow context. Finally, we explain how we created the multi-level set class which derives from the level set class making it easy to pool the resources of the many level set fields for a maximum efficiency.

Sixth chapter

In the sixth chapter, a description is made of some development done for the simulation of vesicles which can not be transposed directly to other works. In this chapter we first describe the problem of constructing efficiently the Lagrange multiplier in the method where the inextensibility of vesicles is insured by it. We also describe how we interfaced the vesicle application. Indeed, because of the high generality spirit of the development, many parameters have been allowed to be set by the user as external options. Thus, at

[3] http://www.boost.org/doc/libs/1_54_0/doc/html/program_options.html

the user point of view, it might be useful to have a small interface to deal at least with the inter dependent options.

Third part : Flow of object in micro capillary and rheology

The last part of this thesis is dedicated to the physical applications of the flow of objects in microcapillaries and their rheology. In order to understand the precise role of the deformability of the particles in such systems, it is important to start with the simpler case of rigid bodies. This is why, in this part, most of the applications are made on rigid disks.

Seventh chapter

One of the building blocks of any microfluidic or biological micro-circulation is the splitting of a channel into two daughter channels having different flow rates. Understanding the behavior of a suspension of particles in such a configuration is a big step forward in the understanding of the micro-circulation in general. The seventh chapter adresses study of the suspension in splitting channels. We started by the most simple case: a dilute suspension of rigid disks in a bifurcation. In collaboration with experimental physicists, we were able to explain the effect of the increasing concentration of the higher flow rate branch already known for several years as the Zweifach-Fung effect and often miss interpreted. We then show the possibility to extend this work to deformable particles and the way to measure properly the interesting quantities.

Eighth chapter

In the future, one issue we want to study is the rheology of a confined suspension of vesicles. To this goal, once again, it is necessary to understand the basic behaviors of rigid disks in order to be able to extract the role of the deformability of the particles. Moreover rheology study requires measurement of the viscosity of a suspension with a controlled accuracy. Finally, even if it has been studied for a long time, the basic behaviors of a suspension of rigid disks in a confined environment still have many open questions and its study is interesting to many groups in the world. For these reasons, the eighth chapter is dedicated to the study of the rheology of solid disks. We first show three methods to measure the effective viscosity of a suspension. We show that two of them always converge to the theoretical effective viscosity and explain the condition for the last one to converge. The errors made on the viscosity are also quantified as a function of the mesh size.
In the second part of the chapter, we study the contribution of a single particle to the effective viscosity and the contribution of a pair of particles. We finally use the study of these contributions to recover the effective viscosity of a suspension of semi dilute particles in a confined shear flow. We show that with this method, we are able to simulate the effective viscosity of a great number of different configurations for every suspension concentration studied in a relatively small time. Finally, we show that the phenomenon of the decrease of the second order viscosity of a suspension with the confinement which was a known effect of a 3D suspension still holds in 2D.

v The finite element library Feel++

The library Feel++

This work has been done using the library Feel++ which stands for *Finite Element Embedded Library in C++* [86, 87]. This is a C++ library solving partial differential equation (PDE) by finite element method. The goal of Feel++ is to provide a language close to the mathematical one to solve complex PDEs. The idea is to provide to scientists a framework in which they can express in a language close to the mathematics the strategy they propose for solving complex systems of PDE and generate a high performance code. Feel++ can be classified as a domain specific embedded language (DSEL), that is to say, it is a C++ code but provides a language which makes the interface between the mathematics written by the user of the library and the lower level high performances computer science domain. To this goal, Feel++ uses the last standards of C++ and meta-programming (through the *template* and boost meta-programming library) to have a maximum of generality. Indeed, the code written using the standard formulation of Feel++ is valid in 1D, 2D and 3D and works with an arbitrary polynomial order since the mathematical formulation allows it. Moreover, the code can run on a single or several processors [17]. It also uses external libraries as for example the mesh generator Gmsh [36], or the library Petsc [3] which provides a large class of method to solve numerical problems.

How does this work fit in Feel++ ?

Within Feel++ , this work is a part of the fluid structure interaction (FSI) project. The goal of this project is to develop strategies toward the simulation of complex blood circulation system. Blood flow is a complex subject and to simulate whole its complexity, one would have to deal with a complex geometry, the interaction between the elastic vessel walls and the blood plasma, and include the red blood cells and possibly other objects dealing with the plasma. During his PhD, Vincent Chabannes developed an arbitrary Lagrangian Eulerian (ALE) method [18] able to deal with the coupling between an elastic wall and a Newtonian fluid. Moreover, flows in complex geometries such as part of the cerebrovenous system, the aorta or an artery with an aneurysm have been investigated with Stokes flow in [15, 17]. This work is dedicated to the simulation of the blood vessels. It could be in the future coupled with the Eulerian part of the ALE method created to deal with the vessel walls to simulate realistic blood flows.

The tools created for the simulation of vesicles can be used to other purposes. For example, we have shown that our two fluid flow system can handle drop suspensions. Future development can be made using the large existing literature on level set methods [98] to create many different applications based on this framework.

Part I

Numerical Methods

Chapter 1

The level set method

Contents

In this chapter we will present the level set numerical methods we have used and some verification results. We will firstly remind some basic principles of level set methods. Then we will detail the formulations of the stabilization methods that we developed to stabilize the transport of the level set function. We will also detail the reinitialization methods available, especially the reinitialization by solving a Hamilton Jacobi equation and the fast marching method. We will discuss our strategy to make the fast marching method valid for high order polynomial approximations. The results of the famous benchmark of the rotation of a slotted disk will be presented. Both time and space convergence will be addressed. Finally, an application of the level set method which does not require a coupling with fluid equations will be presented to show the genericity of the method. This application is the contour detection of an image.

1.1 Principle of the method

1.1.1 The level set method

Many domains in scientific computing require knowledge of the position of an interface between two regions. The method called **level set** proposes a solution to this problem. It has been introduced by Sethian during his PhD on propagating flames in 1982 [97]. He has then kept developing the method and pushed its application fields. His book [98] and Osher's [101] are two major references for the introduction to the method and overview its possible applications. The principle of the method is the following: it defines a scalar function on all the computing space so that the 0 value of this function defines the interface between the two domains. This function is then transported respecting the equations of the system. At each moment, the value where this function is 0 represents the interface. This method, for which the interface is known implicitly, thanks to a function, is classified as an Eulerian method. This classification is made by opposition to Lagrangian methods in which the points on the interface are known explicitly. In the level set method, finding the interface is equivalent to find the 0 value of a function. This function is often taken as a signed distance function to the interface. Then, the function can be seen as the *altitude* on a map for which the interface would be the sea level. One can easily see where the name level set comes from.

One of the advantages using an implicit function to follow the interface is that the method can handle naturally topological changes. Indeed, if the velocity imposed to the level set field induces a sign change in a region, this leads naturally to the creation of a new interface. It is also possible to have an interface disappearing during a simulation. This is automatically taken into account in a level set context which is not the case in Lagrangian point of view. Another big advantage is that it is possible to handle many interfaces with the same level set function. Indeed, the two domains can be spread in many different places on the domain. Nothing forbids the level set function to have the value 0 at many different connected components. The computing time is then not necessarily impacted by the number of interfaces, as opposed to Lagrangian methods in which following one interface or several generally makes a difference from the computational cost point of view.

The method only introduces a scalar field and to use it, one needs to be able to solve

partial differential equations (**PDE**). It is often used in applications which already need PDE solvers (fluid mechanics, combustion, solid mechanics ...) and offers then a solution to follow interfaces without introducing a new tool. It can be used in finite difference **FD**, finite volume **FV** or finite element **FE** frameworks and going from two to three dimensions does not lead to methodological problems. For all these reasons the level set method has been adopted in the multi-fluid flows simulation community for example, but its application fields are much larger. In the book [98], it is applied to combustion, deposition and etching of thin layers on surfaces, noise removal and shape recognition in images, shape optimization, optimal path search, solidification and grid generation.

1.1.2 The level set function

Let us describe in detail the principle of the method. As already explained, the main characteristic of the level set function is that it has to be negative in one domain, positive in the other and vanishes between them. Any function satisfying this property could be used as a level set function. In practice, the function almost always used is the signed distance function to the interface. The choice of this function implies by definition that it has a unit gradient magnitude. As this quantity is used during the advection of the function, it is appreciable to have this property. Moreover, the signed distance function has the property to be regular at each side of the interface. Since the level set function is used as support for Dirac and Heaviside functions, it is important to have also this property. Let us define a function ϕ on a domain Ω composed of two subdomains Ω_1 and Ω_2. The interface between Ω_1 and Ω_2 is denoted Γ. The definition of the function ϕ is the following:

$$\phi(\boldsymbol{x}) = \begin{cases} \operatorname{dist}(\boldsymbol{x}, \Gamma), & \boldsymbol{x} \in \Omega_1 \\ 0, & \boldsymbol{x} \in \Gamma \\ -\operatorname{dist}(\boldsymbol{x}, \Gamma), & \boldsymbol{x} \in \Omega_2 \end{cases} \tag{1.1}$$

with

$$\operatorname{dist}(\boldsymbol{x}, \Gamma) = \min_{\boldsymbol{y} \in \Gamma}(|\boldsymbol{x} - \boldsymbol{y}|). \tag{1.2}$$

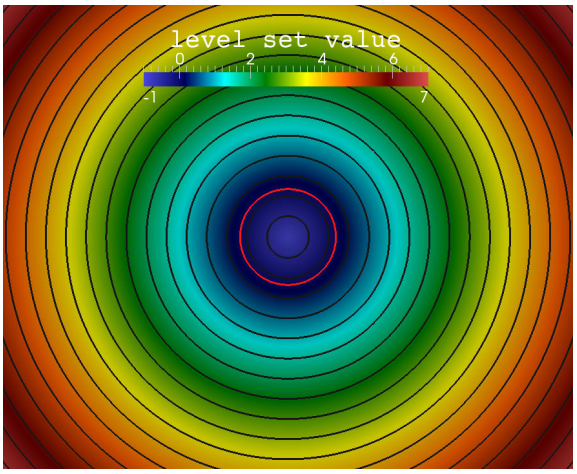
This function is represented in the figure 1.1. As we explained previously, this function has by definition the property $|\boldsymbol{\nabla}\phi| = 1$. We also define the regularized Heaviside function:

$$H_\varepsilon(\phi) = \begin{cases} 0, & \phi \leq -\varepsilon, \\ \dfrac{1}{2}\left(1 + \dfrac{\phi}{\varepsilon} + \dfrac{\sin\left(\dfrac{\pi\phi}{\varepsilon}\right)}{\pi}\right), & -\varepsilon \leq \phi \leq \varepsilon, \\ 1, & \phi \geq \varepsilon \end{cases} \tag{1.3}$$

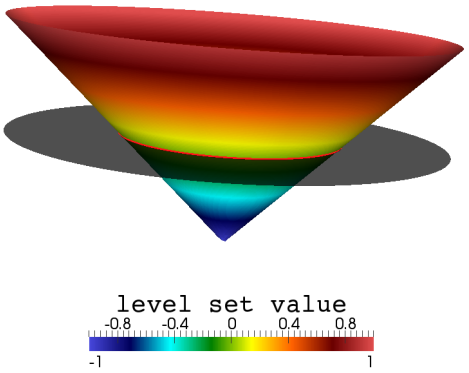
and the regularized Dirac function:

$$\delta_\varepsilon(\phi) = \begin{cases} 0, & \phi \leq -\varepsilon, \\ \dfrac{1}{2\varepsilon}\left(1 + \cos\left(\dfrac{\pi\phi}{\varepsilon}\right)\right), & -\varepsilon \leq \phi \leq \varepsilon, \\ 0, & \phi \geq \varepsilon. \end{cases} \tag{1.4}$$

These two functions are the *smoothed* versions of the classical Heaviside and Dirac functions. They have a thickness of 2ε. That is to say, the thickness of the zone in which



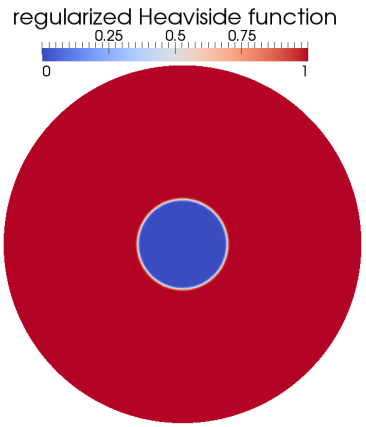
(a) Level set function on the plane. The color is the value of the level set. The iso 0 is represented in red. The iso values are represented as black line.



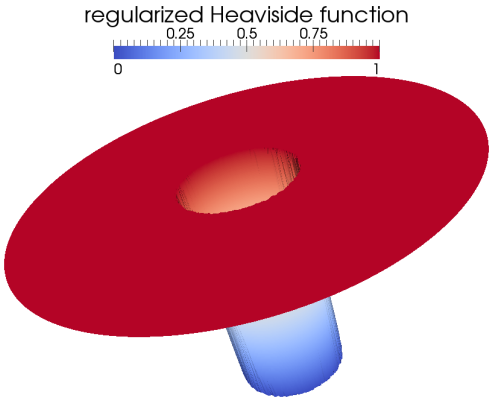
(b) Level set function represented in elevation. The elevation corresponds to the value of ϕ. The 0 plane is represented in grey. The iso 0 of ϕ is represented as a red line. Every points above the 0 plane are in Ω_1, every points below are in Ω_2.

Figure 1.1: Representations of the level set function ϕ.

the Heaviside goes from 0 to 1 and the one in which the Dirac function is not 0 is 2ε. This value has to be taken sufficiently small to approach the real functions and has to be large enough not to induce very brutal jumps in the quantities they define. A typical value often found in the literature is $\varepsilon = 1.5h$ with h the typical size of the mesh. An example of the regularized Heaviside function is given by figure 1.2 and an example of the regularized delta function is given figure 1.3. Both are represented for an interface being a circle.



(a) H_ε function represented in the plane.

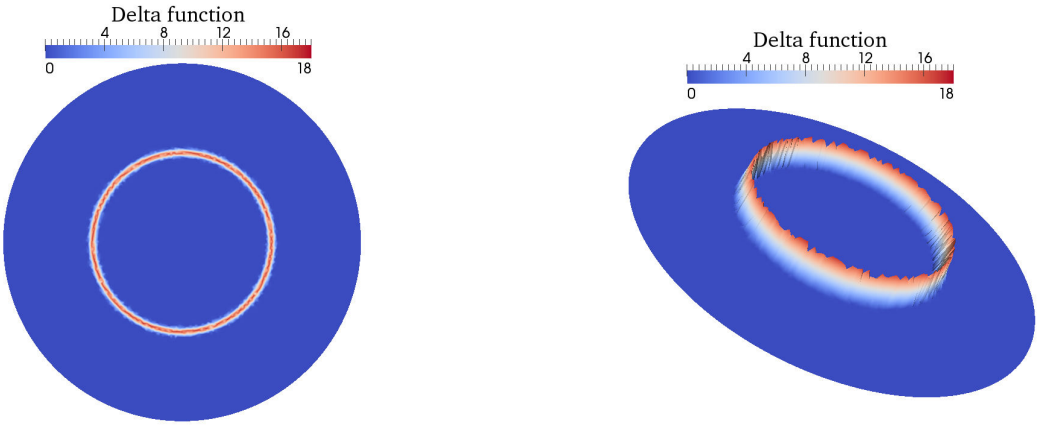


(b) H_ε function represented in elevation.

Figure 1.2: Representation of the regularized Heaviside function.

The function H_ε is used to define quantities having different values on each side of the interface. Let μ be a function having a value μ_1 in Ω_1 and μ_2 in Ω_2. In level set formulation, μ is defined as:

$$\mu = \mu_2 + (\mu_1 - \mu_2)H_\varepsilon. \tag{1.5}$$



(a) δ_ε function represented in the plane.

(b) δ_ε function represented in elevation.

Figure 1.3: Representation of the regularized delta function.

The function δ_ε is used to define quantities having a non 0 value only at the interface (for example interfacial forces). A function f which needs to be distributed on the interface can be distributed on all the domain Ω. Its action is restricted to the interface thanks to the function δ_ε:

$$\int_\Gamma f \simeq \int_\Omega f\ \delta_\varepsilon. \tag{1.6}$$

Thanks to these functions, one can also define the volume (area in 2D) inside each domain, so as the surface (perimeter in 2D) of the interface. The volume inside a domain defined according to the level set is often called the *mass* because of its use in fluid mechanics in which the volume inside the domain and the mass are proportional. One can thus define the *regularized* volume and a surface:

$$V_\varepsilon^+ = \int_\Omega H_\varepsilon \tag{1.7}$$

$$S_\varepsilon^\Gamma = \int_\Omega \delta_\varepsilon. \tag{1.8}$$

One also defines the unit outward normal vector to the interface as well as the curvature. These two quantities are given by:

$$\boldsymbol{n} = \frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|} \tag{1.9}$$

$$\kappa = \nabla \cdot \boldsymbol{n} = \nabla \cdot \left(\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|} \right). \tag{1.10}$$

The evolution of the level set function under the incompressible velocity field $\boldsymbol{u}$ is the advection equation:

$$\partial_t \phi + \boldsymbol{u} \cdot \boldsymbol{\nabla}\phi = 0. \tag{1.11}$$

This hyperbolic equation is a particular form of the advection reaction equation and presents some numerical difficulties to be solved. These issues are presented theoretically in [67]. Few schemes for its resolution are presented in [98] for finite difference methods.

We will present in section 1.2 the method we used to solve this equation in a finite element framework. The advection velocity $\boldsymbol{u}$ has to be equal to the velocity of the interface on Γ. It is thus given by the problem to solve (Navier-Stokes, combustion, shape detection in images). There is no a priori restriction to its value on the rest of the domain except its divergence has to vanish. We will see that usually there is a natural extension for $\boldsymbol{u}$ away from the interface, and if not, one can construct an extension making to advection easy to solve. The equation (1.11) does not conserve ϕ as a distance function which can bring some numerical problems. One needs then a way to bring regularly ϕ close to a distance function. The methods that we used are presented section 1.3.

1.2 Advection

In this section, we will describe how to handle the transport of the function ϕ. The governing equation is the advection equation (1.11) that we recall here:

$$\partial_t \phi + \boldsymbol{u} \cdot \boldsymbol{\nabla} \phi = 0.$$

This is an hyperbolic equation. After time discretization, this equation can be seen as a particular form of the advection reaction equation:

$$\sigma \phi + \boldsymbol{\beta} \cdot \boldsymbol{\nabla} \phi = f. \tag{1.12}$$

We choose to treat the general equation (1.12) and to apply it to the advection (1.11). In this manner, any time discretization scheme can be treated the same way, by discretizing the time derivative and make the terms depending on the previous iterations in the right hand side f. It is also possible to treat the case of other advection reaction equations like for example the one appearing in the reinitilization by Hamilton Jacobi equation framework (see section 1.3).

1.2.1 Discretization of the advection reaction equation

Spatial discretization

The variational formulation of the equation (1.12) is found by multiplying it by a test function ψ and integrating it on Ω. The problem becomes: find $\phi \in H^1(\Omega)$ so that $\forall \psi \in H^1(\Omega)$:

$$\int_\Omega \sigma \phi \psi + \int_\Omega (\boldsymbol{\beta} \cdot \boldsymbol{\nabla} \phi) \psi = \int_\Omega f \psi. \tag{1.13}$$

Dirichlet boundary conditions are added at the boundary of the domain. Let us introduce the discrete finite element space $\mathbb{R}_h^k$ depending on mesh size h and spanned by Lagrange polynomials of degree k. We also introduce $(\phi_h, \psi_h) \in (\mathbb{R}_h^k, \mathbb{R}_h^k)$ the discrete versions of (ϕ, ψ). The discrete form of the equation (1.13) becomes: find $\phi_h \in \mathbb{R}_h^k$ so that $\forall \psi_h \in \mathbb{R}_h^k$:

$$\int_\Omega \sigma \phi_h \psi_h + \int_\Omega (\boldsymbol{\beta} \cdot \boldsymbol{\nabla} \phi_h) \psi_h = \int_\Omega f \psi_h. \tag{1.14}$$

The advection equation being numerically unstable when solved by finite element (or finite difference), a stabilization term has to be added to the variational formulation. We will give more details about this term in section 1.2.2.

Temporal discretization

A framework to solve the problem (1.14) is implemented in Feel++ and will be presented in section 5.2. To be able to use the equation (1.11), one needs to do a temporal discretization of (1.11) and identify the coefficients in this equation to the coefficients $(\sigma, \boldsymbol{\beta}, f)$. The time derivative of the level set function is discretized by both Euler scheme of order 1 (1.15) and a Backward Differentiation Formula of order 2 **BDF2** (1.16). Let Δt be the time discretization step so that, at the iteration $n : t_n = n\Delta t$, we approximate the temporal derivative by:

$$\text{Euler}: \quad \partial_t \phi \approx \frac{\phi^{n+1} - \phi^n}{\Delta t} \tag{1.15}$$

$$\text{BDF2}: \quad \partial_t \phi \approx \frac{3\phi^{n+1} - 4\phi^n + \phi^{n-1}}{2\Delta t} \tag{1.16}$$

where ϕ^n is an approximation of $\phi(t_n)$. By multiplying the equation (1.11) by a test function, integrating over Ω, and replacing the time derivative by the approximations (1.15) and (1.16), we obtain, for the Euler scheme: find $\phi \in H^1(\Omega)$ so that $\forall \psi \in H^1(\Omega)$:

$$\int_\Omega \frac{\phi^{n+1}}{\Delta t}\psi + \int_\Omega (\boldsymbol{u}^{n+1} \cdot \boldsymbol{\nabla}\phi^{n+1})\psi = \int_\Omega \frac{\phi^n}{\Delta t}\psi \tag{1.17}$$

and for the BDF2 scheme: find $\phi \in H^1(\Omega)$ so that $\forall \psi \in H^1(\Omega)$:

$$\int_\Omega \frac{3\phi^{n+1}}{2\Delta t}\psi + \int_\Omega (\boldsymbol{u}^{n+1} \cdot \boldsymbol{\nabla}\phi^{n+1})\psi = \int_\Omega \frac{4\phi^n - \phi^{n-1}}{2\Delta t}\psi. \tag{1.18}$$

By identifying in these equations the coefficients of the equation (1.14), we find the coefficients given in table 1.1. When BDF2 is used, the first step is an Euler step since ϕ^{-1} does not exist. A similar special care has to be taken when a reinitialization is performed. Indeed, if a reinitialization has been made at iteration n, there is a break in the regularity between ϕ^{n-1} and ϕ^n. Thus, if a reinitialization step occurs at iteration n, ϕ^{n+1} is obtained by an Euler step.

	σ	$\boldsymbol{\beta}$	f
Euler	$\frac{1}{\Delta t}$	$\boldsymbol{u}^{n+1}$	$\frac{\phi^n}{\Delta t}$
BDF2	$\frac{3}{2\Delta t}$	$\boldsymbol{u}^{n+1}$	$\frac{4\phi^n - \phi^{n-1}}{2\Delta t}$

Table 1.1: Identification of the terms in equations (1.17, 1.18) to those of the advection reaction equation (1.14)

1.2.2 Stabilization methods

The transport equation (1.12) is difficult to solve numerically. More generally the resolution of the advection-reaction-diffusion equation:

$$\sigma\phi + \boldsymbol{\beta} \cdot \boldsymbol{\nabla}\phi + \varepsilon\Delta\phi = f \tag{1.19}$$

sees some spurious oscillations appearing in the convection dominated regime when no care is taken to stabilize the discretized problem. The equation (1.19) is general and the range of fields where it appears goes from propagating acoustic waves, shallow water equations, Navier-Stokes with incompressibility constraint or not, noise removal in images [98] and of course propagating interfaces. Thus, a lot of effort has been spent to stabilize this equation. In finite volume and finite difference methods, some special schemes have been written to overcome these oscillations. They are called Essentially non-oscillatory (**ENO**) or Weighted **ENO** (**WENO**) schemes [68] and are efficient to solve hyperbolic equations in **FV** and **FD** methods. The idea of these methods is to solve the equation using stencils so that the derivative is always taken in a *upwind* manner (in the same direction of the flow). One can show [10] that it is equivalent to add a small artificial diffusion term only in the direction of the flow. However, these methods are designed for regular grids and do not apply directly for finite element formulations. Many other methods have been written to stabilize equation (1.19) in finite element. We have implemented four of them with FEEL++ . In this section we describe all these methods, that we separated in two different classes. First, we will show three methods in which a perturbation term is added to the elements of the computational domain. Secondly, we will present a more recent method based on the stabilization of the internal faces of the domain.

Elements stabilization

In 1982, Brooks [10] introduced the *streamline upwind Petrov Galerkin* method (**SUPG**) in which one adds a perturbation term to each element of the domain. The perturbation has a weight depending on the upstream values of the flow and on the advection coefficients terms. The method is applied in [10] on 1D advection equation and 2D convection dominated Navier Stokes. One can actually see this perturbation as a way to modify the test functions so that the upstream values are favored compared to the downstream ones. In this sense, it is close to the methods used by finite difference to stabilize the convection dominated problem. Later, Hughes and Franca [44] proposed a formulation in which the advection equation was re-written as a minimization problem. The least square minimization leads to a new formulation of an element stabilization called the *Galerkin Least Square* formulation (**GLS**). Finally Hauke [42] presented a complete study of the *sub-grid scale* method **SGS** in which he developed an idea of Hughes and applied it to more general cases (negative source term f). A Matlab implementation of the **SGS** method is also given in [42]. Good reviews and presentations of those techniques are given in [34, 89, 9]. The way we present the stabilization methods follows these reviews (by using an operator $\mathcal{L}$). In the cited references, the authors present the way to stabilize equation (1.19) with a parameter ε very small compared to $|\boldsymbol{\beta}|$ (convection dominated regime). In the following, we will focus on the pure advection equation and thus we set $\varepsilon = 0$.

The advantage of using element stabilization is that it does not introduce new elements to the matrix associated to the algebraic representation of the discretized problem. Thus, the size of the stabilized problem is the same as without stabilization.
To have a general view of these methods, let us introduce the operator $\mathcal{L}$ and its adjoint

$\mathcal{L}^*$ defined by:

$$\begin{aligned}\mathcal{L} &= \boldsymbol{\beta}\cdot\boldsymbol{\nabla} + \sigma \\ -\mathcal{L}^* &= \boldsymbol{\beta}\cdot\boldsymbol{\nabla} - \sigma.\end{aligned}$$

With the operator $\mathcal{L}$, one can rewrite (1.12) as:

$$\mathcal{L}(\phi) = f.$$

so that we re-write the equation (1.14) using this notation and introducing a stabilization term, the problem becomes, find $\phi_h \in \mathbb{R}_h^k$ so that $\forall \psi \in \mathbb{R}_h^k$:

$$\int_\Omega \mathcal{L}(\phi_h)\psi_h + \sum_{K=1}^{N_{\text{elt}}} S_K(\phi_h, \psi_h) = \int_\Omega f\psi_h$$

where $S_K(\phi_h, \psi_h)$ denotes the stabilization term for a given element K of the discretized domain Ω and N_{elt} represents the total number of elements in Ω. Let us give the definitions of $S_K(\phi_h, \psi_h)$ for the methods **SUPG**, **GLS** and **SGS**. They are all defined the same way, it is the product of an element dependent parameter (τ_K), the residual of equation (1.12): $\mathcal{L}(\phi) - f$ and a linear form of the test function ψ. The whole product being integrated on each element of the mesh Ω_{e}. They read:

$$\begin{aligned} S^{\text{SUPG}} &= \int_{\Omega_{\text{e}}} \tau_K^{\text{SUPG}} \quad (\boldsymbol{\beta}\cdot\boldsymbol{\nabla}\psi_h) \quad (\mathcal{L}(\phi_h) - f) \\ S^{\text{GLS}} &= \int_{\Omega_{\text{e}}} \tau_K \quad \mathcal{L}(\psi_h) \quad (\mathcal{L}(\phi_h) - f) \\ S^{\text{SGS}} &= \int_{\Omega_{\text{e}}} \tau_K \quad -\mathcal{L}^*(\psi_h) \quad (\mathcal{L}(\phi_h) - f) \end{aligned} \tag{1.20}$$

with the associated coefficients:

$$\begin{aligned}\tau_K^{\text{SUPG}} &= \frac{h}{2\,|\boldsymbol{\beta}|} \\ \tau_K &= \frac{1}{\dfrac{2\,|\boldsymbol{\beta}|}{h} + |\sigma|}\end{aligned}$$

where h is the typical size of the concerned element Ω_{e}. Let us explain into more details the SUPG formulation. The stabilization term is proportional to the residual $(\mathcal{L}(\phi_h) - f)$ which insures that the consistency of the discrete equation will not be modified by the stabilization term. Indeed, when adding a stabilization term S_K to a discrete equation the consistency requires that $S_K \to 0$ when $h \to 0$ and $\Delta t \to 0$, which is the case of the residual term. The term $(\boldsymbol{\beta}\cdot\boldsymbol{\nabla}\psi)$ insures that the stabilization only occurs upwind where it is needed. Finally, the coefficient τ_K^{SUPG} controls the amount of diffusivity added. It vanishes with h. This value is not unique but we choose one of the most commonly admitted values for it.

Internal faces stabilization

Another kind of method has been proposed for the first time by Douglas and Dupont in 1976 [25] and a convergence study has been made long time after that (2004) by Burman and Hansbo in [13] on the advection-diffusion-reaction problem and has been applied to Navier Stokes by the same authors in [14]. The precise error analysis at high order has been made in [100]. Two implementations in the context of level set advection can be found in [81, 112]. The idea is to add a penalty term on the jump of the gradient of the solution. This avoids artificial higher gradients to appear. The advantage of this method is that it stays consistent even for higher order finite element. The drawback is that it adds new non-zero entries in the matrix and leads to a more difficult problem to solve.
Let us denote the internal faces of the mesh by Γ_{I}, being all the lines (surfaces in 3D) which are not on a boundary. We also need to define the jump of a quantity across two elements called K_0 and K_1. The jump of a scalar quantity f is a vector defined by:

$$[\![f]\!] = f_0 \boldsymbol{N}_0 + f_1 \boldsymbol{N}_1 \tag{1.21}$$

where $\boldsymbol{N}_i$ denotes the normal of the concerned face pointing outward from the element K_i. And the jump of a vector $\boldsymbol{f}$ is a scalar defined by:

$$[\![\boldsymbol{f}]\!] = \boldsymbol{f}_0 \cdot \boldsymbol{N}_0 + \boldsymbol{f}_1 \cdot \boldsymbol{N}_1 \tag{1.22}$$

Thus, in a configuration where K_0 and K_1 are neighbors, $\boldsymbol{N}_0 = -\boldsymbol{N}_1$ as shown in figure 1.4. The **CIP** stabilization consists of adding the following term to the discretized

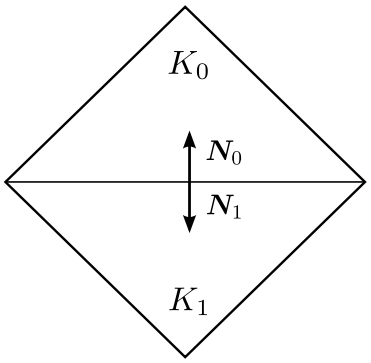


Figure 1.4: Normal of the touching faces of two neighbors elements.

weak formulation (1.14), with $(\phi, \psi) \in (H^1(\Omega), H^1(\Omega))$:

$$S^{\mathrm{CIP}} = \int_{\Gamma_{\mathrm{I}}} \gamma \frac{h_f^2}{k^{3.5}} |\boldsymbol{\beta} \cdot \boldsymbol{N}| \, [\![\boldsymbol{\nabla}\phi]\!][\![\boldsymbol{\nabla}\psi]\!] \tag{1.23}$$

with γ a small factor, h_f the length of the edge and $\boldsymbol{N}$ the outward normal of the edge.

1.3 Keeping the level set function close to a distance function: the reinitialization methods

The function ϕ transported by the equation (1.11) does not conserve the property to be a signed distance function ($|\boldsymbol{\nabla}\phi| = 1$). In the areas where the velocity $\boldsymbol{u}$ goes from a

big value to a smaller one, one sees an *accumulation* of the iso lines or iso surface of the level set function. The gradient magnitude of the function increases and the Heaviside and Dirac function supports are getting thinner. These functions can even have no more mesh element on which they are supported if $|\boldsymbol{\nabla}\phi|$ is too large. At the opposite, an area where $\boldsymbol{u}$ goes roughly from a small value to a big one sees the support of these functions getting larger and the extension of the interface becoming too large. Several solutions exist to this problem.

1.3.1 The advection by an extended velocity

As explained previously, the velocity $\boldsymbol{u}$ used in the equation (1.11) has to be equal to the velocity of the interface on Γ but no particular restriction on the rest of the domain is needed. It is then possible to define an *extended* velocity $\boldsymbol{u}_{\text{ext}}$ equal to the velocity of the interface on Γ and being defined elsewhere in a way so that the advection of ϕ with this velocity does not change the property $|\boldsymbol{\nabla}\phi| = 1$. An example of extended velocity is the one we have used in section 1.5. This velocity is defined by the equation (1.24), that is to say that the value of $\boldsymbol{u}_{\text{ext}}$ on a point of the domain is equal to its value on the closest point in Γ. In practice, searching for each point in the domain the closest point to it on the interface is costly and we choose not to use an extended velocity for multi-fluid applications.

$$\boldsymbol{u}_{\text{ext}}(\boldsymbol{x}) = \begin{cases} \boldsymbol{u}(\boldsymbol{x}) & \text{if} \quad \boldsymbol{x} \in \Gamma \\ \boldsymbol{u}(\boldsymbol{x}^*) & \text{else} \end{cases} \tag{1.24}$$

with

$$\underset{\boldsymbol{x}^* \in \Gamma}{\text{dist}}(\boldsymbol{x}, \boldsymbol{x}^*) = \text{dist}(\boldsymbol{x}, \Gamma)$$

Other ways exist to create extensional velocities, many are described in [98]. Nevertheless, they all must be calculated at every time iteration and for performance purposes we choose other methods for multi-fluid applications. This method could be really interesting when there is no obvious definition of the velocity in all the domain. For example, for a multi-fluid application, taking the velocity of the fluid on all the domain as the advection velocity is an obvious choice. For contour detection on an image (see section 1.5), such a velocity does not exist, thus an extended velocity is well suited for this application.

1.3.2 The interface local projection

When ϕ is not too irregular, it has been shown in [81, 21] that the function $\dfrac{\phi}{|\boldsymbol{\nabla}\phi|}$ is close to a distance function where ϕ vanishes, that is to say near the interface. So it is possible to use $\dfrac{\phi}{|\boldsymbol{\nabla}\phi|}$ as support for the functions H_ε and δ_ε which will then have a regular extension at each side of the interface. Nevertheless, for a long time simulation with a velocity field $\boldsymbol{u}(t)$ irregular, $|\boldsymbol{\nabla}\phi|$ can take extreme values. If $|\boldsymbol{\nabla}\phi|$ becomes too big, it is possible than $\dfrac{\phi}{|\boldsymbol{\nabla}\phi|}$ tends to 0 and with the numerical error changes of sign. In this case, a new artificial interface appears and is advected by the equation (1.11) as if it was a real interface. This method is not completely sufficient to solve the problem of keeping ϕ as a distance function but it is extremely useful for two purposes: i) it maintains a relative

regularity of the functions H_ε and δ_ε for a simulation time longer than if one uses the classical ϕ as a support. ii) It can be used as an exact value for a distance function close to the interface. This last advantage is particularly valuable in the case of a reinitialization by a fast marching method as we will see in section 1.3.4.

1.3.3 Reinitialization by solving a Hamilton Jacobi problem

Hamilton Jacobi equation leading ϕ to a distance function

It is possible to reset the function ϕ regularly to a signed distance function. The position of the 0 value of the new function has to be the same as the previous one, in this way, the interface does not move. A method based on an idea of [91], modified in [104] and used several times since (a non-exhaustive list is given by [103, 106, 31, 78, 110]) consists in solving a Hamilton Jacobi equation of the form:

$$\partial_t \phi = S_{\text{gn}}(\phi)(1 - |\boldsymbol{\nabla}\phi|) \tag{1.25}$$

S_{gn} being the sign function. When this equation is solved to equilibrium ($\partial_t \phi = 0$) the solution is a signed distance function (one has $|\boldsymbol{\nabla}\phi| = 1$). One can rewrite the equation (1.25) as:

$$\partial_t \phi - S_{\text{gn}}(\phi)(1 - \frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|} \cdot \boldsymbol{\nabla}\phi) = 0. \tag{1.26}$$

Equation (1.26) is a transport equation with a velocity equal to $S_{\text{gn}}(\phi)\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}$. One can then see this reinitialization as a *front* propagating and resetting ϕ to a distance function. The velocity $\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}$ is normal to the interface and is pointing out from the interface. The reinitialization front with such a velocity would always go from the interface Γ to the exterior of the domain Ω_1 (i.e toward $\phi > 0$). This is the reason for the introduction of the sign function as a factor to this velocity. This way, the front always goes from the interface Γ to both the domains Ω_1 and Ω_2. Thus, the areas close to the interface are always reinitialized first. This can be useful since the need of having a distance function is stronger close to the interface. Moreover, generally the initial function ϕ is not too far away from a signed distance function, the equation (1.25) is not too far from the equilibrium. One can generally use a large pseudo time step and few iterations are needed to reach the equilibrium state. The sign function is smoothed so that the numerical error does not make the front go to the wrong side. It is often found on the form initially given by [104]:

$$S_{\text{gn}}(\phi) = \frac{\phi}{\sqrt{\phi^2 + \varepsilon^2}} \tag{1.27}$$

with ε a small smoothing parameter. In our case, we prefer to use the formula given in [63] using the function H_ε. Thus, the smoothing parameter ε is the interface thickness. So we use:

$$S_{\text{gn}}(\phi) = 2H_\varepsilon - \frac{1}{2}. \tag{1.28}$$

The advantage of using this formulation, is that it does not need a new parameter ε, the interface thickness is used and the sign function is defined the same way as the Heaviside and Delta functions.

Discretization and resolution of the problem

The equation (1.25) is discretized in time using an Euler scheme and linearized by taking the velocity term as being the value of $\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}$ at the previous iteration. We obtain the equation:

$$\frac{\phi^{n+1}}{\Delta\tau} + \frac{S_{\text{gn}}(\phi^n)\boldsymbol{\nabla}\phi^n}{|\boldsymbol{\nabla}\phi^n|} \cdot \boldsymbol{\nabla}\phi^{n+1} = S_{\text{gn}}(\phi^n) + \frac{\phi^n}{\Delta\tau} \tag{1.29}$$

with $\Delta\tau$ the pseudo time step. The weak formulation associated is obtained by multiplying by the test function ψ and integrating the equation (1.29) on all the space. The problem obtained is then: find $\phi \in H^1(\Omega)$ so that $\forall\psi \in L^2(\Omega)$:

$$\int_\Omega \frac{\phi^{n+1}}{\Delta\tau}\psi + \int_\Omega \left(\frac{S_{\text{gn}}(\phi^n)\boldsymbol{\nabla}\phi^n}{|\boldsymbol{\nabla}\phi^n|} \cdot \boldsymbol{\nabla}\phi^{n+1}\right)\psi = \int_\Omega \left(S_{\text{gn}}(\phi^n) + \frac{\phi^n}{\Delta\tau}\right)\psi. \tag{1.30}$$

One recognizes an equation of the form (1.14) with the coefficients given by:

$$\begin{aligned}\sigma &= \frac{1}{\Delta\tau}\\ \boldsymbol{\beta} &= \frac{S_{\text{gn}}(\phi^n)\boldsymbol{\nabla}\phi^n}{|\boldsymbol{\nabla}\phi^n|}\\ f &= S_{\text{gn}}(\phi^n) + \frac{\phi^n}{\Delta\tau}\end{aligned}$$

One understands that the great advantage of this method is that it is simple to implement since it does not need other tools than the one already implemented for the advection of the function ϕ. Moreover, this formulation is very generic and no hypothesis has been done for the dimension of the spaces or the polynomial order of the finite element functions. It also works on one or many processors since the PDE solver used provides this possibility (as opposed to the fast marching, for which a special care has to be taken in the algorithm).
It is possible to create an indicator to know at each iteration if one needs to reinitialize or not. An example of indicator is the *distance to a distance function*. It measures the relative distance of ϕ to a signed distance function. It is given by:

$$e_{\text{dist to dist}} = \frac{1}{V}\int_\Omega \sqrt{(|\boldsymbol{\nabla}\phi| - 1)^2} \tag{1.31}$$

with V the volume of the domain. One can then choose a stop criterion for the reinitialization thanks to this indicator.
In practice, we prefer to fix a reinitialization frequency. At this frequency, we solve the equation (1.30) until we obtain $\frac{|\phi^{n+1} - \phi^n|}{|\phi^n|} < \varepsilon_{\text{tol}}$, with ε_{tol} a numerical tolerance imposed according to the needs of the application.

The reinitialization requires choosing correctly the parameters: reinitialization frequency, $\Delta\tau$ and ε_{tol}. These parameters depend on the space discretization h, on the time step Δt, on the precision needed for the application. Even if one can find relations between these parameters, there does not exist any absolute formula working for every

application, and, in practice, one needs to do many tests before finding optimal parameters. It is the principal drawback that one can find to this method. The other drawback is that in general, the interface slightly moves during the first iteration of (1.30), and for long time simulations, one can see large variations of the volume of the domains due to the implied numerical errors.

An example: the signed distance function to an ellipse

The figure 1.5 shows different iterations of the equation (1.30). The example is taken from chapter 2.4 for which this reinitialization method has been used. We do not know an analytic expression for a signed distance function to an ellipse. To initialize a level set function to an ellipse, we have to use the following function:

$$\phi_0^{\text{ellipse}} = \sqrt{\left(\frac{x-x_0}{a}\right)^2 + \left(\frac{y-y_0}{b}\right)^2} - 1 \tag{1.32}$$

with (x_0, y_0) the position of the center of the ellipse and a and b its semi-axes. This function is not a signed distance function, one can easily show that $\left|\boldsymbol{\nabla}\phi_0^{\text{ellipse}}\right| \neq 1$, but the iso surface of value 0 and the sign of each side of the interface are well positioned. ϕ_0 is an acceptable approximation of a distance function when the [1]eccentricity of the ellipse is very small. If one starts the simulation with this function, one sees that the Dirac and Heaviside functions will have a different thickness along the interface. Thus, we need to reinitialize this function to a signed distance function. The initial reinitialization is the one taking the longest time since, after that, we never let ϕ be so different from a distance function. Moreover, since we do not want to lose too much mass at the beginning of the simulation, a small pseudo time step is chosen. In this example, the mesh size is $h = 0.025$, the time step for the simulation is $\Delta t = 0.0125$ and the pseudo time step has been taken as $\Delta\tau = 0.08$. The equation is stabilized by a SUPG method. We see on the figure 1.5 the state of ϕ_0^{ellipse} every 10 iterations.

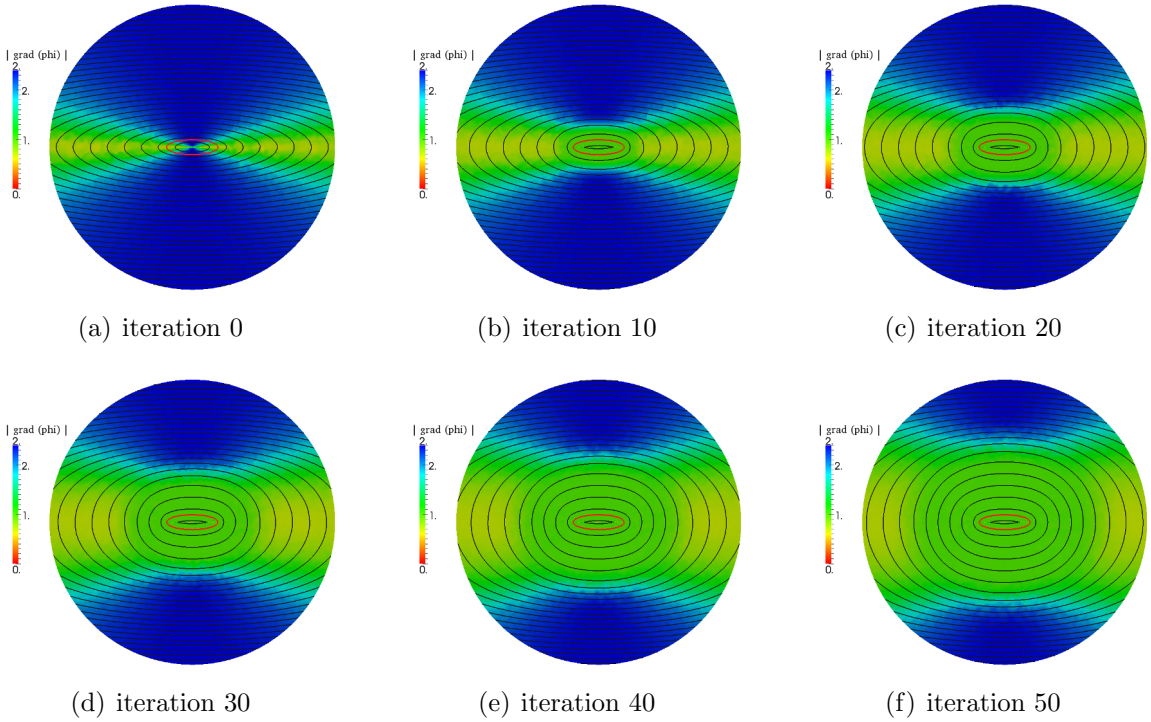
1.3.4 The fast marching method

Basic principle of the fast marching

Another way to reinitialize ϕ to a distance function is to modify one by one every degree of freedom of the system by calculating the new value of ϕ. Such a method has already been described in [98] and with more details in [72]. The method is named the *fast marching method* (**FMM**). This method is a way to efficiently iterate over the degrees of freedom and updating the visited points.

The idea is the following: it starts by an initialization phase, let us suppose that one already knows the value of ϕ as a distance function on a few degrees of freedom. One marks these points as being *accepted*, they will never be changed again during the reinitialization step. Then, one searches for all the closest neighbors to the *accepted* points and we mark them as *close*. All the other points are marked as *far*. A new value $\tilde{\phi}$ is

[1]The eccentricity is defined as $\sqrt{1-\left(\frac{b}{a}\right)^2}$. At the limit of a vanishing eccentricity, the ellipse is a circle.



(a) iteration 0 (b) iteration 10 (c) iteration 20

(d) iteration 30 (e) iteration 40 (f) iteration 50

Figure 1.5: Successive iterations of the discretized Hamilton Jacobi equation for the reinitialization of the function ϕ_0^{ellipse}. The line $\phi_0^{\text{ellipse}} = 0$ is represented in red and other values of level lines equally separated are represented in black. The background color represents $\left|\boldsymbol{\nabla}\phi_0^{\text{ellipse}}\right|$. One can see that $\left|\boldsymbol{\nabla}\phi_0^{\text{ellipse}}\right|$ expands to a value of 1 while the shape of the iso 0 stay preserved.

calculated for each point marked as *close*, this value is computed by using the values of ϕ of the points marked *accepted* for which one knows that ϕ is a correct distance function. The elements in *close* are then sorted by their values of $\left|\tilde{\phi}\right|$. They are stored in a heap so that the one having smallest value is on the top. Then follows the real marching procedure, it is reported in the algorithm 1.

With this method, we are sure that one point is accepted every iteration and the algorithm 1 has a complexity of $\mathcal{O}(N)$. The fact that the elements *close* are in a sorted heap is important, indeed, once the heap is sorted, it is easy to find the smallest value of $\tilde{\phi}$ and update the heap. The initialization procedure has a complexity of $\mathcal{O}(\log(N))$.
One understands the term *marching*. Indeed, the idea is never to modify an already accepted point. Thus, the algorithm always goes from known points to the unknowns. The method requires that some points are already known at the initialization step. For that, one uses the interface local projection method presented above 1.3.2. Thus, the elements close to the interface are known at the beginning of the marching algorithm. The way to calculate the new value of $\tilde{\phi}$ using the known values of ϕ is given in detail in [112] both for 2D and 3D, but it assumes monotonicity of ϕ between the nodes (only $\mathbb{P}^1$ elements). The fast marching method for FEEL++ has already been implemented during the PhD thesis of Christophe Winkelmann [112]. We used it without changes except adding the possibility to handle periodic boundary conditions. This implementation has been done

Algorithm 1 Fast Marching procedure

Require: initialization of the arrays *done* *far*, and the heap *close*

while *close* is not empty **do**

- find the element in *close* having the smallest value of $\left|\tilde{\phi}\right|$, let us call it *trial*
- delete the *trial* point from *close*
- mark the *trial* point as *accepted*
- compute the values $\tilde{\phi}$ of the *far* neighbors of *trial* and put them at the right place in the *close* heap
- update the values $\tilde{\phi}$ of the elements in *close* being neighbors of *trial*

end while

for finite element $\mathbb{P}^1$. Its direct extension to higher order would be difficult. We choose a different strategy which is presented hereafter.

The advantage of the fast marching is its robustness. Indeed, if the elements around the interface are correctly set when starting the fast marching algorithm, it always results to a nice distance function and there is not any parameter to be set by the user. This is an advantage compared to the reinitialization by solving the Hamilton Jacobi equation presented in section 1.3.3 where many parameters have to be set correctly and depend on the application. Moreover the method is efficient and fast. The main drawback is that at the time this document is written, the marching algorithm has not been made for parallel computation. Thus it works only in sequential. Even if it does not seem too difficult to make the algorithm work in parallel, the performances would not be increased since the algorithm described above is not meant to be parallel.

Fast marching at high order

There exists in Feel++ a method called operator Lagrange $\mathbb{P}^1$ allowing one to create a polynomial $\hat{\mathbb{P}}^1$ space from a $\mathbb{P}^n$ space, let us call it OpLag. We denote with a hat the space obtained by the transformation OpLag: $\hat{\mathbb{P}^1} = \text{OpLag}(\mathbb{P}^n)$. The mesh associated to the $\hat{\mathbb{P}}^1$ space is created from the mesh associated to the $\mathbb{P}^n$ space on which all the degrees of freedom have been replaced by nodes of a $\hat{\mathbb{P}}^1$ element. Such a transformation is represented on the figure 1.6 for $n = 2$. The two spaces have the exact same number of degrees of freedom. There also exists an operator $\pi^{\mathbb{P}^n \to \hat{\mathbb{P}}^1}$ to project a field living in the

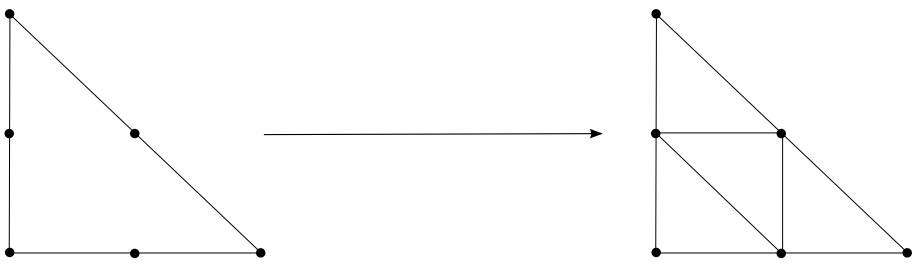


Figure 1.6: Action of operator Lagrange $\mathbb{P}^1$ on a $\mathbb{P}^2$ element in 2D. It transforms a $\mathbb{P}^2$ element into 4 $\mathbb{P}^1$ elements. The degrees of freedom are represented as black dots and the nodes of the elements are linked by black lines. The total number and the geometrical position of the degrees of freedom are not changed by this transformation.

$\mathbb{P}^n$ space onto the $\hat{\mathbb{P}}^1$ space. The opposite projector also exists $\pi^{\hat{\mathbb{P}}^1 \to \mathbb{P}^n}$. We emphasize the fact that an element created by the operator $\pi^{\mathbb{P}^n \to \hat{\mathbb{P}}^1}$ has the exact same number of degrees of freedom that the element in the $\mathbb{P}^n$ space used to create it. Moreover, both elements have the exact same values at degrees of freedom having the same geometrical coordinates. Of course the same remark holds for $\pi^{\hat{\mathbb{P}}^1 \to \mathbb{P}^n}$. We use this transformation to create a temporary level set function $\hat{\phi}$ being in a $\mathbb{P}^1$ space $\hat{\phi} = \pi^{\mathbb{P}^n \to \hat{\mathbb{P}}^1}(\phi)$. Then $\hat{\phi}$ is reinitialized using the fast marching method, let us call it $\hat{\phi}^*$ after reinitialization. Finally, the inverse transformation is used $\phi^* = \pi^{\hat{\mathbb{P}}^1 \to \mathbb{P}^n}(\hat{\phi}^*)$ and one obtains a function ϕ^* being a distance function in a $\mathbb{P}^n$ space. We summarized this transformation on the scheme 1.7. We present an example of reinitialization using the proposed high order fast marching

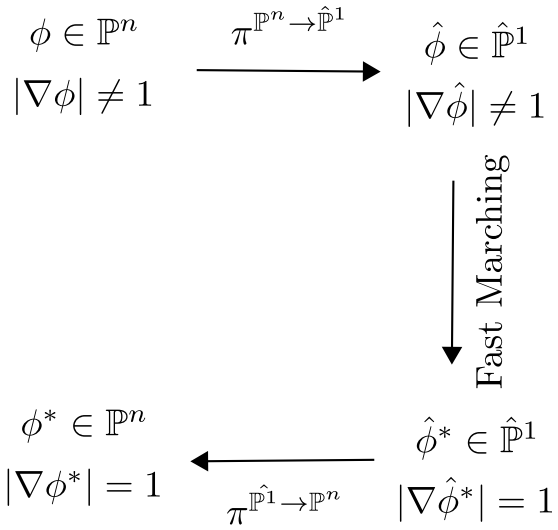
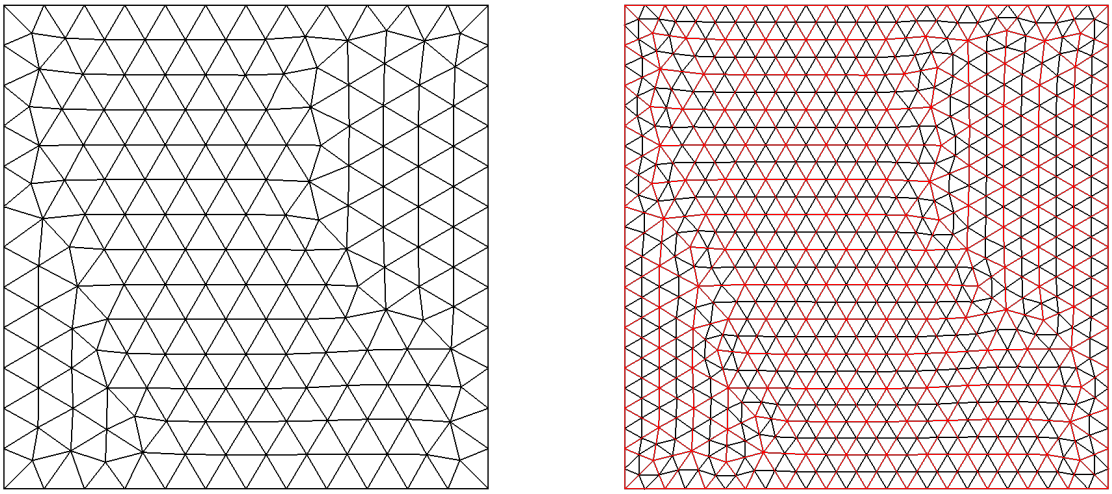


Figure 1.7: Scheme of the transformations used to do a reinitialization using the fast marching method for an element being in a space of polynomials of order greater than 1. The $\hat{\mathbb{P}}^1$ space is created from the $\mathbb{P}^n$ space: $\hat{\mathbb{P}}^1 = \text{OpLag}(\mathbb{P}^n)$.

method. The level set function is set on a unit square by the equation (1.32) for which we put $a = 0.3$ and $b = 0.1$. With this equation, we can show easily that $|\boldsymbol{\nabla}\phi| \neq 1$ thus ϕ is not a distance function. We use a space of polynomials of order 2 as space for ϕ and test the procedure below. Figure 1.8(a) shows the mesh associated to the $\mathbb{P}^2$ space and figure 1.8(b) shows the mesh associated to the $\mathbb{P}^1$ space created from the $\mathbb{P}^2$ mesh using OpLag. The faces which already existed in the initial mesh are reported in red so that it becomes clear which faces and points have been added.

Figures 1.9(a) and 1.9(b) represent the value of ϕ before and after the reinitialization procedure on a mesh thinner than the one used in figure 1.8. The iso 0 is represented in green and 30 iso lines are represented as black lines. The color is given by $|\boldsymbol{\nabla}\phi|$. We see clearly that $|\boldsymbol{\nabla}\phi|$ becomes close to 1 and that the iso lines are equally spaced.

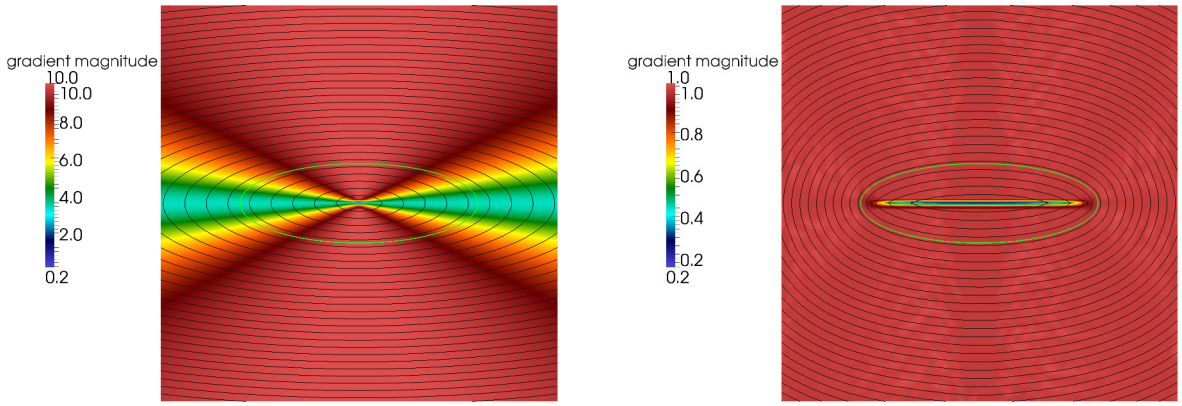
Finally, figure 1.10 shows the same reinitialization procedure but taking $\phi \in \mathbb{P}^5$. We have also plotted a close up of the intermediate mesh constructed with Operator Lagrange $\mathbb{P}^1$ on the mesh associated to the $\mathbb{P}^5$ space.



(a) Mesh associated to the $\mathbb{P}^2$ space.

(b) Mesh associated to the $\mathbb{P}^1$ space after using Operator Lagrange $\mathbb{P}^1$. The edges which were already present in the previous mesh are represented in red.

Figure 1.8: Action of Operator Lagrange $\mathbb{P}^1$ on a mesh associated to a $\mathbb{P}^2$ space.



(a) ϕ before reinitialization procedure.

(b) ϕ after reinitialization procedure.

Figure 1.9: Value of $\phi \in \mathbb{P}^2$ before and after the reinitialization procedure by fast marching method. The iso 0 is represented in green and 30 iso lines are represented as black lines. The color is given by $|\boldsymbol{\nabla}\phi|$.

The distance to the boundaries of a domain

The fast marching has been designed to reinitialize the level set field to a distance function, but it can also be used in another context: to get the distance to the closest boundary of a domain. Indeed, some applications need the information of the distance to the closest boundary as for example turbulence models where a boundary layer model is used. Although it is trivial to get the distance to the boundary when the geometry is a rectangle,

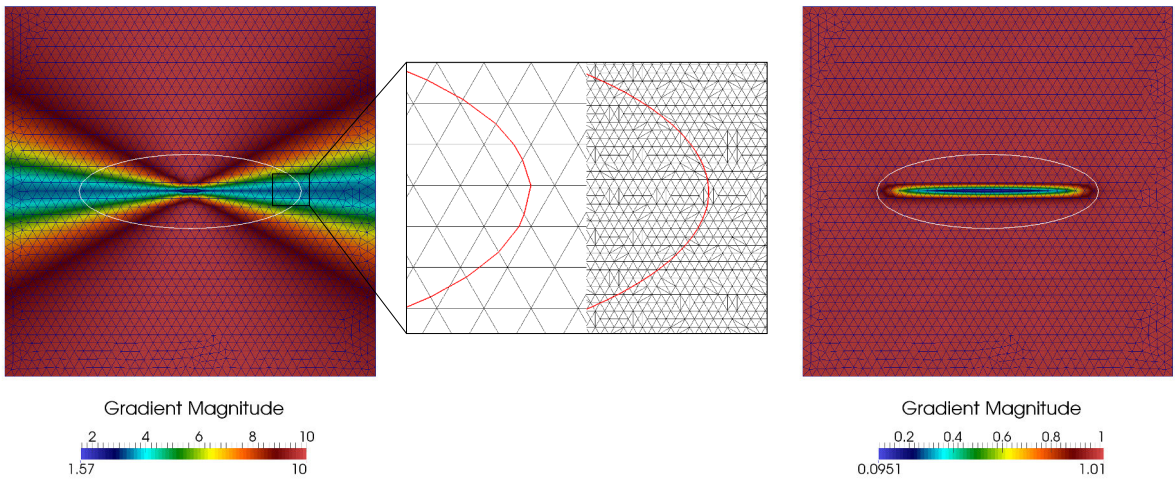


Figure 1.10: The fast marching procedure for $\phi \in \mathbb{P}^5$. The left and right hand side colored plot show respectively $|\boldsymbol{\nabla}\phi|$ before and after the reinitialization procedure. On the center, a close up is shown. It represents on the left, the mesh on which is defined ϕ and on the right the one after the use of Operator Lagrange $\mathbb{P}^1$.

it might be difficult in a complex geometry. A solution is to use the fast marching method.

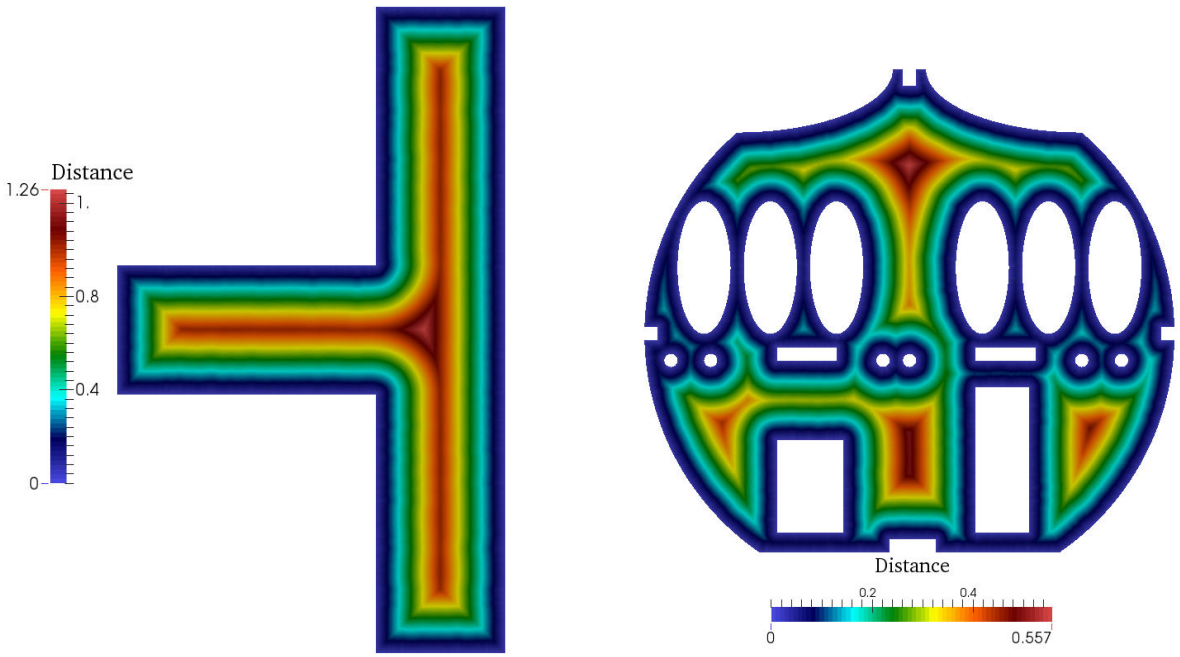
The trick is to give to the fast marching algorithm a starter field ϕ_0 so that the degrees of freedom touching the boundaries have a value very close to 0 but negative and the other degrees of freedom of the elements touching the boundaries have the value of the distance from it. This way, everything is like if a level set field was present with the iso 0 at the boundary and the first elements around the iso 0 have the value of a distance function.

The value given to the degrees of freedom not touching the boundary but being in an element touching it, is the local size of the mesh h. If the mesh is sufficiently regular, this is a good approximation of the distance of the degree of freedom to the wall (at least in a $\mathbb{P}^1$ approximation, but for higher order, the Operator Lagrange $\mathbb{P}^1$ can be used). Then the value of the degrees of freedom touching the wall has to be set to a negative value very small compared to h. In practice, one can set it to $\frac{-h}{100}$ for example. Finally, one can use the fast marching with this starter field and the result is the distance to the walls of the domain.

Figure 1.11(a) shows the distance function obtained by using this procedure on the bifurcation geometry of the chapter 7. The figure 1.11(b) shows the geometry for which we have actually designed this procedure. It represents the inside of an airplane cabin, some Feel++ users try to model the flow and heat transfer of the air through it. The flow is turbulent, and, as mentioned above, the distance to the boundary is needed to know where to activate the boundary layer model. The fast marching algorithm is used to provide this distance field.

The distance to a parametrized curve

For most of the simulations using the level set method, the initial value of the level set field should be a distance function or a field not too far from that. This way, a reinitializa-



(a) Bifurcation geometry. (b) Airplane cabin.

Figure 1.11: Distance to the boundaries of two geometries computed by the fast marching method.

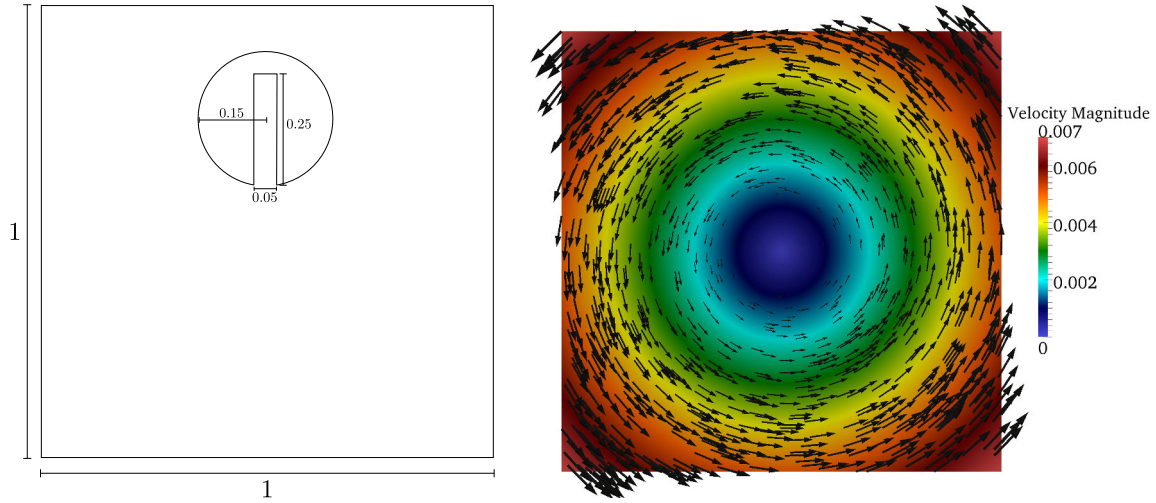
tion procedure can be applied and the simulation can start with a distance function field. Writing a distance function to a line or a circle is easy. Distance functions of shapes being a combination of lines and circles can be written with min and max functions of distance to lines and circles. Distance function to more complicated shapes are impossible to write analytically. To be able to start with more complicated shapes, we built a method to create a distance function to a parametrized curve.

Once again, the key is to create a distance function to the first elements touching the curve. Since this field is created, it can be given as a starter to the fast marching method which gives as a result a nice distance field. The procedure to create a distance function to a parametrized curve is given in appendix C.

1.4 Benchmark, solid rotation of a slotted disk

1.4.1 Presentation of the benchmark

This test has been initially proposed by Zalesac in [114]. It has become one of the most common tests for interface propagation. It has been used for example in [59, 12, 78, 29] to test numerical frameworks. The test consists of the rotation of a slotted disk in a square domain. The geometry of the domain is represented in figure 1.12(a). The domain Ω is the square $[0,1] \times [0,1]$. The center of the slotted disk is initially in $(0.5, 0.75)$. The imposed velocity is a solid rotation of center $(0.5, 0.5)$. The angular velocity is chosen so that the disks makes one round every $t_f = 628$. The velocity has the following form $\boldsymbol{u} = \left(\frac{\pi}{314}(50-y), \frac{\pi}{314}(x-50)\right)$, it is represented on the domain in the figure 1.12(b).



(a) Initial geometry (b) Amplitude and vector of the imposed velocity.

Figure 1.12: Zalesac test.

One imposes Neumann boundary conditions on the boundary of the domain. Every 628 time unit, the circle is supposed to come back at the initial place with the same shape. Thus, one can define errors by computing the difference between $\phi_0 = \phi(t = 0)$ and $\phi_f = \phi(t = t_f)$. One can define three different errors to measure the quality of the advection scheme. First, the *mass error*,

$$e_{\mathrm{m}} = \frac{|m_{\phi_f} - m_{\phi_0}|}{m_{\phi_0}} = \frac{\left|\int_\Omega \chi(\phi_f < 0) - \int_\Omega \chi(\phi_0 < 0)\right|}{\int_\Omega \chi(\phi_0 < 0)} \tag{1.33}$$

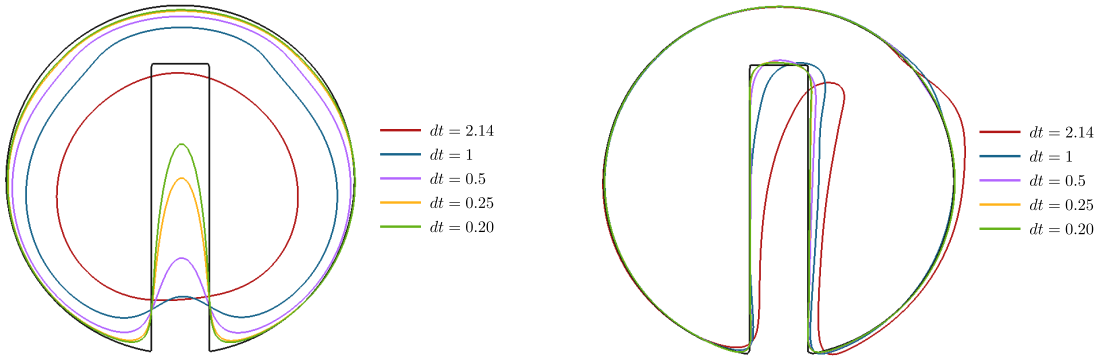
with χ the characteristic function defined by:

$$\chi(f(\phi)) = \begin{cases} 1 & \text{if} \quad f(\phi) \neq 0 \\ 0 & \text{if} \quad f(\phi) = 0. \end{cases} \tag{1.34}$$

This error measures the global mass loss. It is a tricky indicator, since in general, the level set method gains and loses some mass at the same time at different areas of the domain. Thus, one needs a better estimate of how the interface has moved during the simulation. Thus we introduce the *sign change error*:

$$e_{\mathrm{sc}} = \sqrt{\int_\Omega \left((1 - H_0) - (1 - H_f)\right)^2} \tag{1.35}$$

where $H_0 = H_\varepsilon(\phi_0)$ and $H_f = H_\varepsilon(\phi_f)$. This error is a way to compute the area in which $\phi_0\phi_f < 0$, in others words, where the interface has moved. The last error estimate that we define is the classical L^2-error. In this context, it might not be the best one to quantify the quality of the advection since it measures the difference between initial and final solution on all the space whereas our interest is only the interface transport quality. Indeed, there exists some inflow at the boundaries, and the value of ϕ in this inflow is not necessarily



(a) Euler (b) BDF2

Figure 1.13: Final shapes after one rotation with Euler and BDF2 scheme. The original shape is shown in black.

the same as the values of the initial ϕ. Thus, we modify this error quantification to make a local L^2-difference in the region near the initial interface. The L^2 error on the interface is given by:

$$e_{L^2} = \sqrt{\frac{1}{\int_\Omega \chi(\delta_\varepsilon(\phi_0) > 0)} \int_\Omega (\phi_0 - \phi_f)^2 \chi(\delta_\varepsilon(\phi_0) > 0)}. \tag{1.36}$$

1.4.2 Time convergence

The first test that we present is the convergence of the time schemes. To this goal, we set a fixed grid discretization $h = 0.004$ which leads to 72314 degrees of freedom (on a $\mathbb{P}^1$ space). Then we let the circle make one round and measure the errors. We set the following different time steps $dt = (2.14, 1, 0.5, 0.25, 0.20)$ and compare the errors obtained by an Euler or BDF2 discretization. The stabilization strategy used for this test is SUPG. This benchmark is run in sequential for the first tested time step and in parallel for the other one, going from 1 to 10 processors. The final shapes are represented in figure 1.13. The errors associated are reported in table 1.2 for Euler discretization and table 1.3 for BDF2 discretization. The shapes represented by figure 1.13(b) are asymmetrical, this is due to the fact that the BDF2 scheme has a strong asymmetry in time, the result depends on the two previous iterations. Since the circle rotates only in one way, a bias is created in one direction of space more than in the other. For Euler scheme at the opposite, only the previous iteration is needed to advance forward in time, the asymmetry does not exist and the shapes on figure 1.13(a) are symmetrical.

The convergence graph corresponding to the data of tables 1.2 and 1.3 are shown in figure 1.14(a), 1.14(b) and 1.14(c). As explained previously, the error on the sign change is better than the mass error to characterize the real quality of the scheme. The figure 1.14(b) shows the rate of convergence for this error. To our knowledge, there are no theoretical results giving the rate of convergence of such a test. However, a better rate of convergence is expected for the highest order of discretization. This is the result shown by the figures 1.14(a), 1.14(b) and 1.14(c). The error mass convergence is more a guide

dt	e_{L^2}	e_{sc}	e_{m}
2.14	0.0348851	0.202025	0.202025
1.00	0.0187567	0.147635	0.147635
0.50	0.0098661	0.10847	0.10847
0.25	0.008791	0.0782569	0.0782569
0.20	0.00803373	0.0670677	0.0670677

Table 1.2: Errors on the time convergence of Euler scheme for Zalesac benchmark.

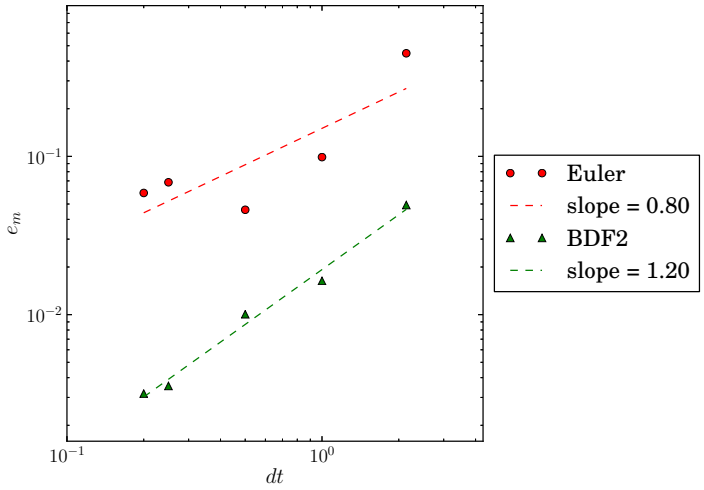
dt	e_{L^2}	e_{sc}	e_{m}
2.14	0.0118025	0.0906617	0.0492775
1.00	0.00436957	0.0445275	0.0163494
0.50	0.00173637	0.0216359	0.0100621
0.25	0.001003	0.0125971	0.00354644
0.20	0.000949343	0.0117449	0.00317368

Table 1.3: Errors on the time convergence of BDF2 scheme for Zalesac benchmark.

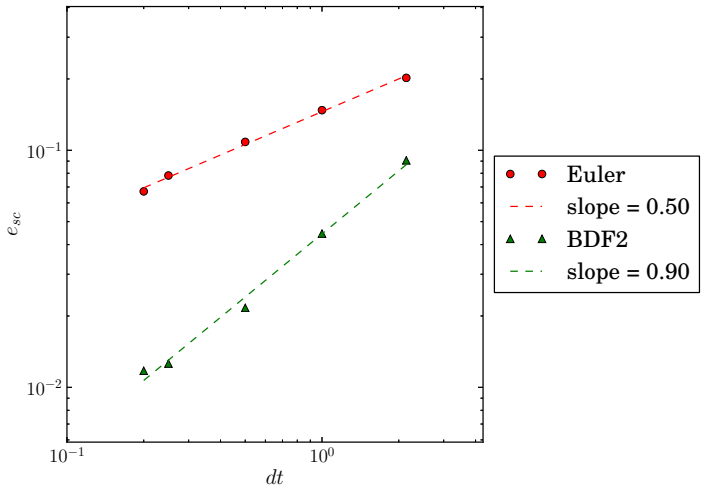
for the eyes since, as we already explained, the loss of mass in a region is sometimes compensated by a gain of mass in another region, making the mass error a poor error estimate. Thus this error might not be relevant in this case. The sign change error shows a gain of around half an order going from Euler discretization scheme to BDF2 whereas in the same case, the L^2 error shows a gain of one order.

1.4.3 Space convergence

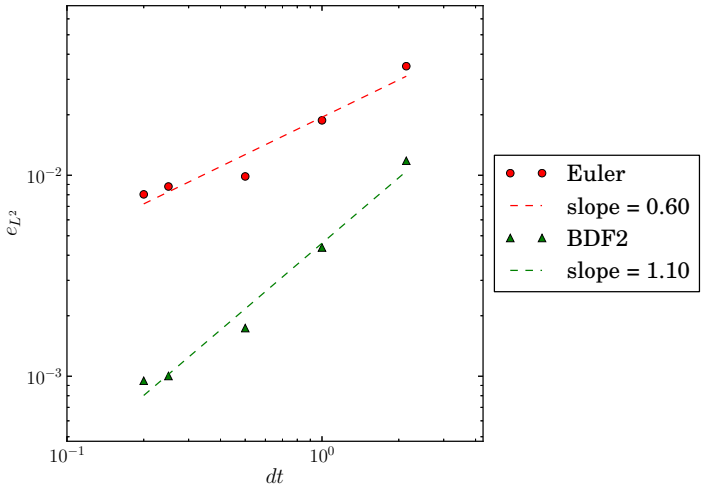
The goal of the second test is to verify the space convergence. We set a relation between the time step and the space discretization. The time step, is usually taken as $dt = C\,\dfrac{h}{U_{\max}}$, where C is a constant smaller than one and $U_{\max}$ the maximum velocity on the domain. In our case we set $C = 0.8$ and the maximum velocity is obtained for the points having the highest distance from the center. With the velocity imposed, we have $U_{\max} = 0.007$. Thus the time step is taken as $dt = 0.8\,\dfrac{h}{0.007}$. A BDF2 time scheme is chosen for the time discretization. Different mesh sizes are taken: $h = 0.32, 0.16, 0.08, 0.04$. The test is run in parallel from 2 processors, for $h = 0.32$ to 10 processors for $h = 0.04$. All the stabilization methods at our disposal are tested, namely, CIP, SUPG, GLS, SGS. The final shapes obtained are represented in figure 1.15. The numerical errors are represented in table 1.4. The sign change error is represented in figure 1.16(a). All the stabilization methods show a rate of convergence of 0.6 and all the errors are almost equivalent. The L^2 error convergence is represented in figure 1.16(b) and the convergence of this error is not as clear as the previous one. All the element stabilization show a rate of convergence of 0.9, whereas the CIP stabilization only shows a rate of convergence of 0.6.



(a) Mass error as a function of time step for Zalesac benchmark.



(b) Sign change error as a function of time step for Zalesac benchmark.



(c) L^2 local error as a function of time step for Zalesac benchmark.

Figure 1.14: Convergence of the errors measured in the benchmark for the time discretization methods Euler and BDF2.

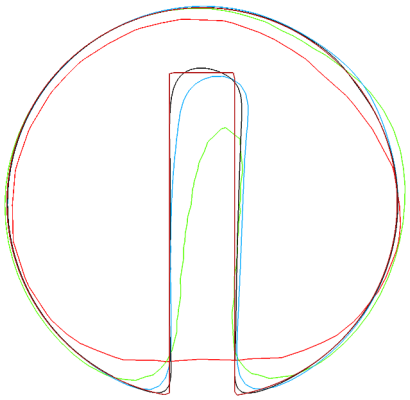
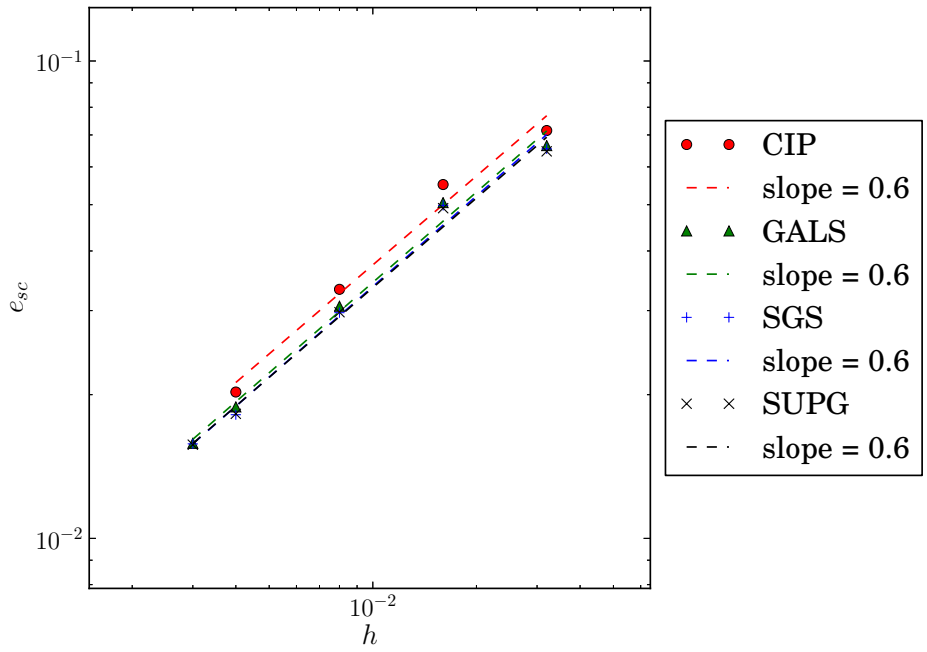


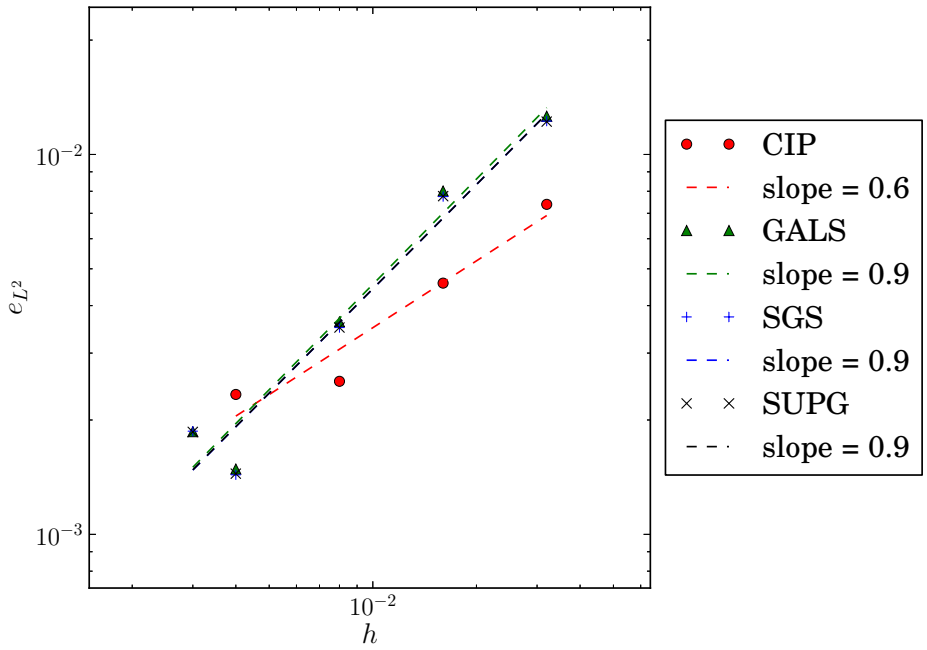
Figure 1.15: Final shapes of the Zalesac circle after a whole revolution for different mesh sizes. The initial shape is also depicted.

stab	h	e_{L^2}	e_{sc}	e_{m}
CIP	0.32	0.0074	0.072	0.00029
	0.16	0.0046	0.055	0.00202
	0.08	0.0025	0.033	0.00049
	0.04	0.0023	0.020	0.00110
SUPG	0.32	0.012	0.065	0.01632
	0.16	0.008	0.049	0.07052
	0.08	0.004	0.030	0.00073
	0.04	0.001	0.018	0.00831
GLS	0.32	0.013	0.066	0.02499
	0.16	0.008	0.051	0.05180
	0.08	0.004	0.031	0.00805
	0.04	0.001	0.019	0.00672
SGS	0.32	0.012	0.065	0.01103
	0.16	0.008	0.050	0.07570
	0.08	0.004	0.030	0.00084
	0.04	0.001	0.018	0.00850

Table 1.4: Errors for different mesh sizes and stabilization methods.



(a) Error and rate of convergence of the sign change error as a function of the space discretization.



(b) Error and rate of convergence of the L^2 error as a function of the space discretization.

Figure 1.16: Space convergence of the measured errors for the four stabilization methods tested.

1.5 An application : the active contours

The level set method can be used in different domains than multi-fluids simulations, for example it can be used in image analysis to find systematically the boundaries of objects. This technique has been used for the first time by Malladi [72]. Many improvements have been made, and a good review of those enhancements is given in [102]. This method is often used in image analysis. One can find it for example as a plugin for the free image analysis software ImageJ [2]. An image being organized in pixels, it is easy to represent it on a regular grid. Thus, the finite difference method is often used for this kind of application and most of the schemes presented in the cited references are written for the finite difference method. However, the method naturally extends to finite element method. We were interested to implement the active contours method not for a research purpose but to test the generality of the LevelSet class implemented in Feel++ . Indeed, the LevelSet class has to be totally decoupled from the fluid part and has to be usable for totally different applications than multi-fluid simulations. Moreover, if the class interface is correctly designed, a basic active contours application has to be easy and simple to implement. This part shows that, as for the level set method, the LevelSet class implemented can be used for other goals than multi-fluids simulations, and it has been a guide to design the LevelSet interface.

1.5.1 Principle of active contours method

The principle of the active contours method is to see an image as a scalar function of gray level (the method can be extended to colored images by using a vectorial function, but we will not go into the details in this description). Let us call $I(\boldsymbol{x})$ the function representing the gray level of an image. Finding a boundary in an image is equivalent to finding the part in the image where there exists a strong contrast. In other words, finding the boundary of an object in an image, is equivalent to find the set of $\boldsymbol{x}$ so that $|\boldsymbol{\nabla} I|$ is big. A possible solution could be to calculate the gradient of gray level everywhere in the image and define the boundary as being the set of $\boldsymbol{x}$ for which $|\boldsymbol{\nabla} I| > \mathrm{I}_{\max}$. The drawback of this method, is that if the image is noisy, all the noise will be included in the boundary. Malladi has then proposed to define a scalar level set field in the image and to initialize it as a small circle inside the area in which the boundary is searched. The idea is then to make evolve the iso 0 of the level set (called the *front*) by applying a velocity field $\boldsymbol{u}$. This velocity is normal to the front and depends on the function I. The goal is that at the end of the simulation, the front fits the contour of the object in the image. This front is called the *active contour*. The velocity $\boldsymbol{u}$ has to vanish when $|\boldsymbol{\nabla} I|$ is big enough. Such a velocity is given by :

$$\boldsymbol{u} = \frac{1}{1 + \alpha\,|\boldsymbol{\nabla} I|} \frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|} \tag{1.37}$$

where α is a parameter. Let us now imagine what happens in a noisy region. In such a region, a lot of small regions exists where $|\boldsymbol{\nabla} I|$ is big, making the front stop around them. These regions being small, the curvature of the front around it should very high. Thus, one can add to the velocity term, a component proportional to the curvature of the front.

[2] Module maintained by Erwin Frise http://fiji.sc/Level_Sets

This way, the velocity will not vanish anymore in these regions, and the front will bypass the noise. This is a solution to the problem explained previously. The velocity where the curvature term has been added is then given by :

$$\boldsymbol{u} = \frac{1 - \beta\kappa}{1 + \alpha\,|\boldsymbol{\nabla} I|}\boldsymbol{n} \tag{1.38}$$

with κ the curvature of the level set function, recalling $\kappa = \nabla \cdot \left(\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}\right) = \nabla \cdot \boldsymbol{n}$ and β a parameter. One generally adds a term attracting the front to the boundaries of the object. This term stabilizes the position of the active contour to the boundary of the object. It is an attractive force which derivates from a potential $P = -\,|\boldsymbol{\nabla} I|$. This term can be seen as a potential well in which falls the active boundary. The velocity is then :

$$\boldsymbol{u} = \frac{1 - \beta\kappa - \gamma\boldsymbol{\nabla} P \cdot \boldsymbol{n}}{1 + \alpha\,|\boldsymbol{\nabla} I|}\boldsymbol{n}. \tag{1.39}$$

The velocity $\boldsymbol{u}$ can contain other terms improving the precision or performances of the boundary detection. Once given an expression to $\boldsymbol{u}$, it goes to the user to choose the right parameter set (α, β, γ) adapted to his images and to the boundary being searching. The method is then quite flexible. Moreover, since the level set can handle topological changes, the user can start the detection with many circles of different radii at many places in the image. When different fronts meet, they naturally merge, this can enhance the quality of the detection or makes possible to find disjoint objects.

1.5.2 Implementation

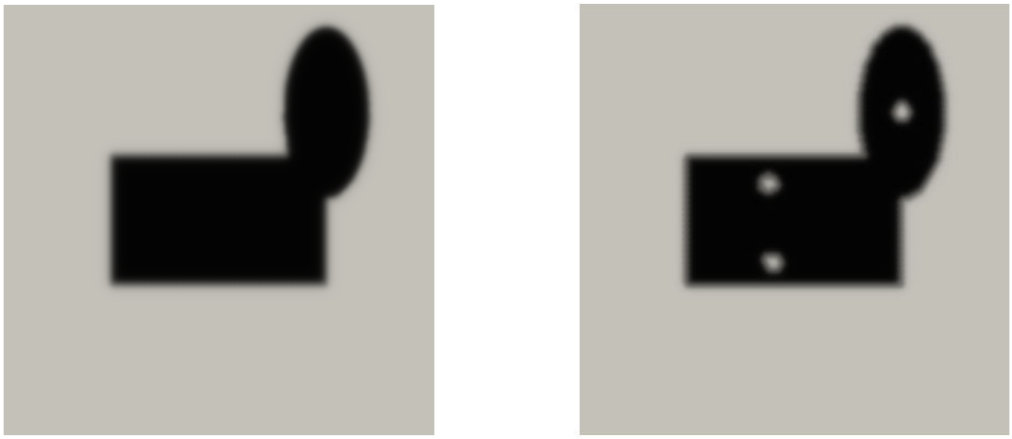
The image

Feel++ does not allow one yet to load an image as an element of a finite element space. Nevertheless, the mesh generation software [3]Gmsh allows creating a mesh associated to an image and to extract a field corresponding to the function I previously described. The boundary detection by level set method being for the moment a simple doability test, the interface allowing one to load Gmsh generated functions has not been written. However, it could be implemented for future applications. We wrote analytically some I functions representing different geometrical shapes. To this goal one only needs to know a formula giving the inside and outside boundary of a curve, and to give a different value to $I(\boldsymbol{x})$ according to the position of $\boldsymbol{x}$ with respect to the curve. We then smoothed this image thanks to the smoothing projection tool 4.1.2. In [72], the authors apply to the images a Gaussian filter called G. The image on which the boundary is actually searched is then $G \star I$ with $\star$ the convolution product. We defined two test images (fig 1.17(a) and 1.17(b)) representing a rectangle intersected by an ellipse. On the second image, some artificial noise has been introduced as small white disks.

The advection velocity

The major difference with a multi-fluid simulation system is that one cannot give on all the domain the previously described velocity $\boldsymbol{u}$. Indeed, this velocity is only written for

[3]http://geuz.org/gmsh



(a) Rectangle intersected by an ellipse. (b) Rectangle intersected by an ellipse with noise.

Figure 1.17: Images used for the detection.

the front (the 0 value of ϕ). Applying this velocity everywhere in the domain would lead to instabilities. A solution consists on calculating an extended velocity $\boldsymbol{u}_e$ being equal to the velocity $\boldsymbol{u}$ close to the interface and sufficiently regular everywhere else to avoid instabilities. Let us call Ω_{far} the space far away from the interface and Ω_{close} the space near the interface. In practice, one has to define what is *far* and *close*. In our case, we consider that a point is close if it is on an element crossed by the interface $\phi = 0$ as represented on the figure 1.18. One could also define close the same way we defined

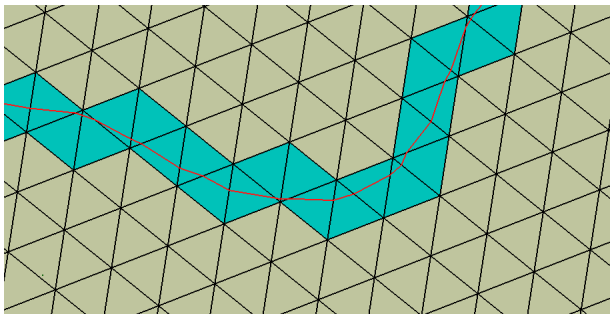


Figure 1.18: Marker on the element crossed by the interface.

the thickness of the Dirac function since the thickness is thin enough. The extensional velocity is defined by :

$$\boldsymbol{u}_e(\boldsymbol{x}) = \begin{cases} \boldsymbol{u}(\boldsymbol{x}) & \text{if} \quad \boldsymbol{x} \in \Omega_{\text{close}}, \\ \boldsymbol{u}(\boldsymbol{x}_{\text{mindist}}) & \text{if} \quad \boldsymbol{x} \in \Omega_{\text{far}} \end{cases} \tag{1.40}$$

with

$$\boldsymbol{x}_{\text{mindist}} \in \Omega_{\text{close}} | \, \text{dist}(\boldsymbol{x}_{\text{mindist}}, \boldsymbol{x}) = \min_{\boldsymbol{x}_2 \in \Omega_{\text{far}}}(\text{dist}(\boldsymbol{x}_2, \boldsymbol{x})) \tag{1.41}$$

in other words, each point in Ω_{far} has the same velocity as the closest point in Ω_{close}. At every iteration, one calculates the velocity at the interface and iterates on the degrees of

freedom far from the interface. One associates to each degree of freedom the velocity of the closest point in the interface. This should be done by iterating on the degrees of freedom in an optimized way, for example using a fast marching method as for the reinitialization step. Our goal for this application being the workable and not the performance, we only implemented a simple non optimized iterative process.

Stop criterion

The velocity never totally vanishes since $|\boldsymbol{\nabla} I|$ is never infinite. Thus, it is important to define a threshold below which the velocity is considered as 0. This minimum velocity $u_{\min}$ is another parameter the user has to monitor. The real velocity advecting the level set function is thus :

$$\boldsymbol{u}'_e(\boldsymbol{x}) = \boldsymbol{u}_e(\boldsymbol{x})\ \chi\left(|\boldsymbol{u}_e(\boldsymbol{x})| > u_{\min}\right) \tag{1.42}$$

In the following, we will intentionally forget the apostrophe and consider that $\boldsymbol{u}_e$ is the velocity on which the threshold has been applied. We can now write a stop criterion for the simulation. The program stops when the active contour does not move anymore, thus when

$$\int_{\Omega_{\text{close}}} \boldsymbol{u}_e(\boldsymbol{x}) = 0, \tag{1.43}$$

we integrate on Ω_{close} which is the real matter of interest, but since the extended velocity follows the velocity of the interface, it would be equivalent to integrate on all the domain.

1.5.3 Detection tests

The first and the simplest detection is done on the noiseless image. It is done without curvature term and with a very weak attractive force. This test being easy, a relatively big time step is used and few iterations are necessary to reach the final state. The results are presented in the figure 1.19. The second test (figure 1.20) is done on the noisy image but keeping the parameter $\beta = 0$. Thus, the noisy areas are detected as boundaries. The image resolution chosen being rough so as the time step, these boundaries are only approximately described. Finally the last test (figure 1.21) is done on the same noisy image but by choosing a β parameter big enough to overstep the noisy areas but sufficiently small in order not to modify the image boundary at the corners where the curvature is high.

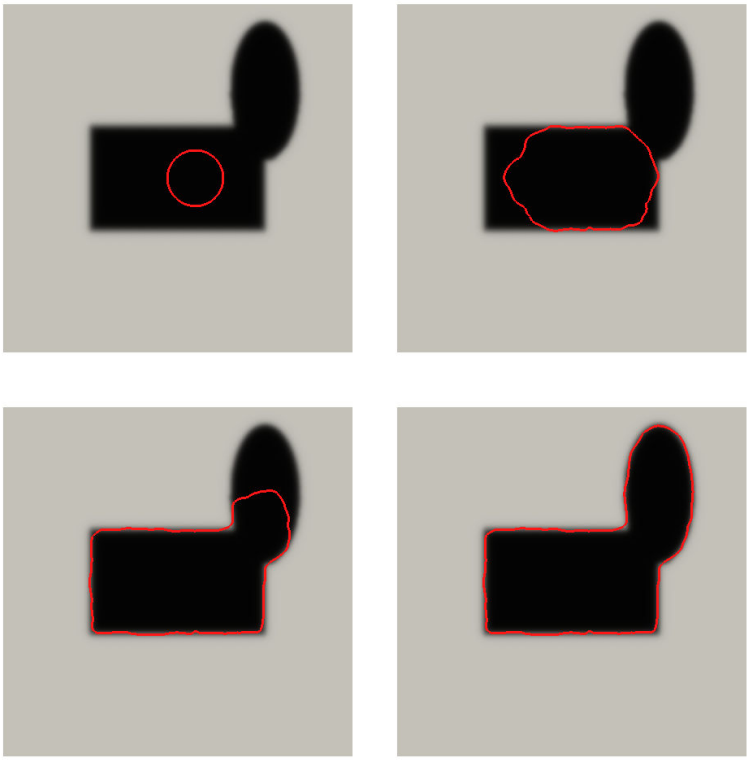


Figure 1.19: Active boundary detecting the boundary on a noiseless image.

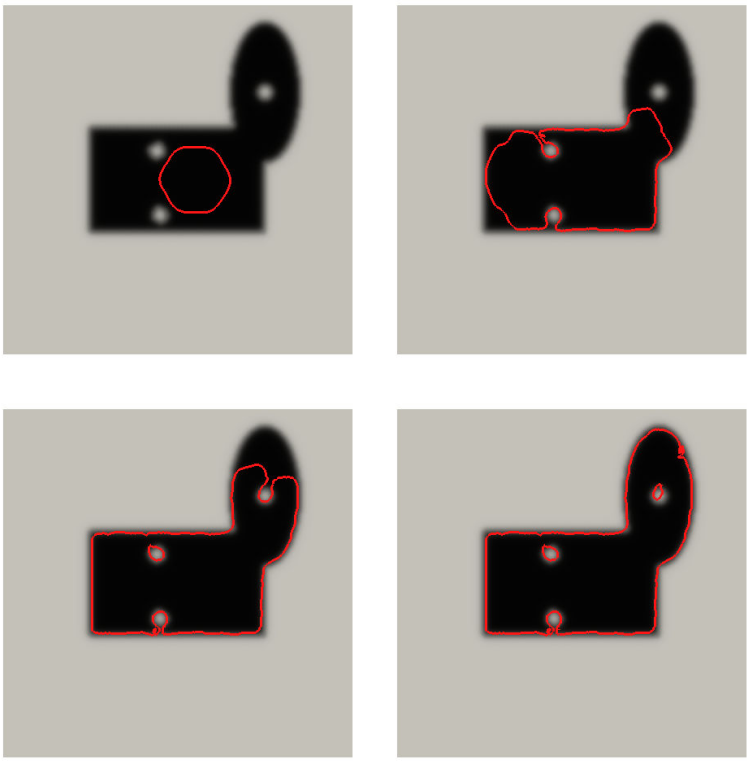


Figure 1.20: Active boundary moving on a noisy image with a velocity independent on its curvature.

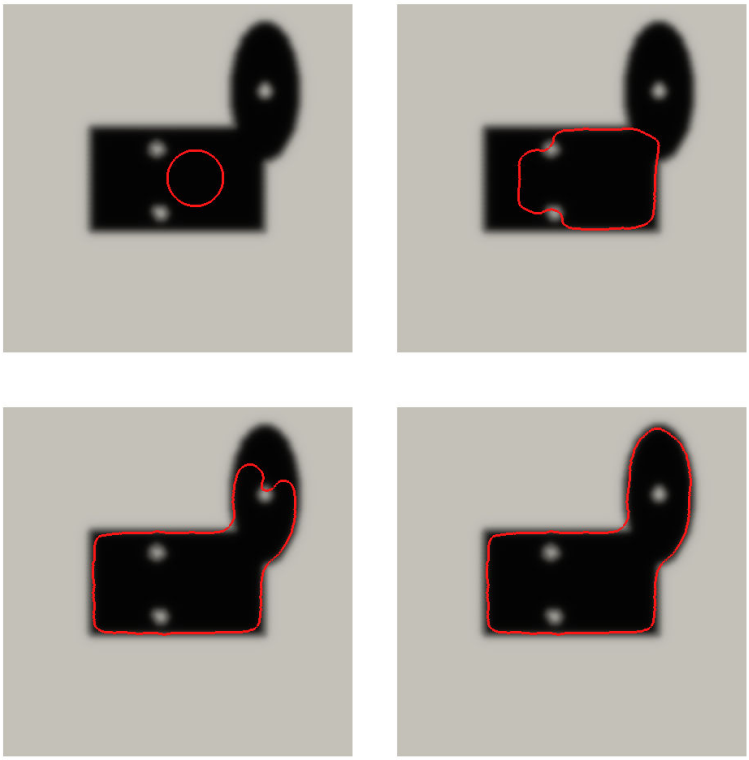


Figure 1.21: Active boundary moving on a noisy image with a velocity depending of its curvature to overstep the noise.

Chapter 2

Multi-fluid flows simulation

Contents

In this chapter, we will present the fluid equations, the strategy we used to solve them and their coupling with the level set method presented previously. We will start by recalling the Stokes and Navier-Stokes equations and show how they are solved in finite element context. Then we will show how we can couple the level set method and the fluid solver to have a two-fluid flow and extend this method to a multi-fluid flow. A verification on a precise benchmark on two-fluid flow will be done and a test of a 9 fluid flow will be presented. Finally, we will use this framework to answer a physical question proposed by experimental physicists: does the theoretical frequency of oscillations of a bubble in an infinite fluid still hold in a very confined environment?

2.1 Fluid equations

2.1.1 Stokes equations

Derivation of the Stokes equation

The Stokes equation is used to describe the behavior of a fluid when the viscosity dominates the flow. A way to determine the relative strength of the inertia and the viscosity is to compute the *Reynolds number*. This number gives a typical value of the ratio of the inertial and the viscous terms. It is given by:

$$R_e = \frac{\rho U L}{\mu} \tag{2.1}$$

with ρ the density of the fluid, μ its viscosity, L a typical length of the flow and U a typical velocity magnitude of the flow. When the effect of inertia is negligible compared to the viscous terms ($R_e << 1$), the momentum equation can be written as:

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} = -\boldsymbol{f}. \tag{2.2}$$

With $\boldsymbol{\sigma}$ the fluid stress and $\boldsymbol{f}$ the volume forces exerted to the fluid. The expression of $\boldsymbol{\sigma}$ can be complicated and the *rheology* is the science dedicated to the study of this object. Nevertheless, it is known for common fluids that the stress is proportional to the rate of strain being defined by:

$$\boldsymbol{D}(\boldsymbol{u}) = \frac{\boldsymbol{\nabla u} + {}^t(\boldsymbol{\nabla u})}{2}. \tag{2.3}$$

A fluid for which the strain rate and stress tensors are proportional is called a *Newtonian fluid*. For these fluids, the stress tensor can be expressed as:

$$\boldsymbol{\sigma} = -p\boldsymbol{I}_{\mathrm{d}} + 2\mu\boldsymbol{D}(\boldsymbol{u}) \tag{2.4}$$

with p the pressure, $\boldsymbol{I}_{\mathrm{d}}$ the identity matrix, $\boldsymbol{u}$ the fluid velocity and μ the viscosity of the fluid. When the velocities and forces involved are not too high and for a common liquid, one can usually admit that the liquid is incompressible. This results in:

$$\nabla \cdot \boldsymbol{u} = 0 \tag{2.5}$$

by taking the divergence of the equation (2.4) and using equation (2.5) one can re-writes (2.2) as:

$$-\mu\boldsymbol{\Delta u} + \boldsymbol{\nabla} p = \boldsymbol{f} \tag{2.6}$$

Equations (2.5) and (2.6) are the so-called Stokes equations governing an incompressible fluid at vanishing Reynolds number. Stokes equations are linear, and independent of time which makes them more simple than the general Navier-Stokes equations.

Discretization of the Stokes equations: a simple example

Let us see on a simple example how we solve the Stokes equations with finite element methods, from the physical equations to the algebraic representation. Let us write the Stokes equations with Dirichlet boundary conditions:

$$\begin{aligned} -\mu\boldsymbol{\Delta}\boldsymbol{u} + \boldsymbol{\nabla} p &= \boldsymbol{f} \quad \text{in } \Omega, \\ \nabla\cdot\boldsymbol{u} &= 0 \quad \text{in } \Omega, \\ \boldsymbol{u} &= \boldsymbol{u}_{\mathrm{D}} \quad \text{on } \partial\Omega. \end{aligned} \tag{2.7}$$

with $\boldsymbol{u}_D$ a known value of the velocity on the boundary $\partial\Omega$ of the domain Ω.
Let us introduce some functional spaces:

$$\begin{aligned} L^2(\Omega) &= \left\{ f : \Omega \to \mathbb{R}; \int_\Omega |f|^2 < +\infty \right\} \\ [L^2(\Omega)]^d &= \left\{ \boldsymbol{f} : \Omega^d \to \mathbb{R}^d; \int_\Omega |\boldsymbol{f}|^2 < +\infty \right\} \\ [L^2_0(\Omega)]^d &= \left\{ \boldsymbol{f} \in [L^2(\Omega)]^d; \int_\Omega \boldsymbol{f} = 0 \right\} \\ H^1(\Omega) &= \left\{ f \in L^2(\Omega); \boldsymbol{\nabla} f \in [L^2(\Omega)]^d \text{ if } \Omega \subset \mathbb{R}^d \right\} \\ [H^1(\Omega)]^d &= \left\{ \boldsymbol{f} \in [L^2(\Omega)]^d; \boldsymbol{\nabla}\boldsymbol{f} \in [L^2(\Omega)]^{d\times d} \text{ if } \Omega \subset \mathbb{R}^d \right\} \\ [H^1_0(\Omega)]^d &= \left\{ \boldsymbol{f} \in [H^1(\Omega)]^d; \boldsymbol{f} = 0 \text{ on } \partial\Omega \right\}. \end{aligned}$$

The problem (2.7) is not well posed since the pressure is only defined by its gradient. Thus there is not a unique solution. If $(\boldsymbol{u}, p)$ is a solution, then $(\boldsymbol{u}, p + K)$ is also a solution, with K a constant. A common solution to fix this constant is to impose that the pressure has a vanishing mean value on the domain

$$\int_\Omega p = 0,$$

thus, the solution $(\boldsymbol{u}, p)$ to the problem (2.7) is unique [90]. For this example, we will not enter into the details how to impose this constraint, we will assume $p \in L^2_0(\Omega)$. Later on, we will see two methods to impose this zero mean pressure. The variational formulation of the problem (2.7) is obtained by taking the scalar product of the first equation with a test function $\boldsymbol{v} \in [H^1_0(\Omega)]^d$ and multiplying the second one by a test function $q \in L^2_0(\Omega)$ and integrating over Ω. After integrating by parts, and recalling that $\boldsymbol{v} = 0$ on $\partial\Omega$, we have: for a given $\boldsymbol{f} \in [L^2(\Omega)]^d$, find $(\boldsymbol{u}, p) \in [H^1(\Omega)]^d \times L^2_0(\Omega)$ so that $\forall\, (\boldsymbol{v}, q) \in [H^1_0(\Omega)]^d \times L^2_0(\Omega)$:

$$\begin{aligned} \int_\Omega \mu\boldsymbol{\nabla}\boldsymbol{u} : \boldsymbol{\nabla}\boldsymbol{v} - \int_\Omega p\nabla\cdot\boldsymbol{v} &= \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v} \\ \int_\Omega q\nabla\cdot\boldsymbol{u} &= 0 \end{aligned} \tag{2.8}$$

with : the double contraction (see appendix G for details). Let us now introduce the finite element spaces:

$$\mathbb{U}_h^{n+1} \subset [H^1(\Omega)]^d,\ \mathbb{U}_h^{n+1} = \{\boldsymbol{w}_h | \boldsymbol{w}_h = \sum_{i=1}^{N_{\text{dof}}^{\mathbb{U}_h}} \boldsymbol{w}_i \boldsymbol{\Phi}_i\}$$

$$\mathbb{P}_h^n \subset L^2(\Omega),\ \mathbb{P}_h^n = \{r_h | r_h = \sum_{i=1}^{N_{\text{dof}}^{\mathbb{P}_h}} r_i \psi_i\}$$

with:

$N_{\text{dof}}^{\mathbb{U}_h}$ and $N_{\text{dof}}^{\mathbb{P}_h}$ the total number of degrees of freedom of respectively $\mathbb{U}_h^{n+1}$ and $\mathbb{P}_h^n$.

$\boldsymbol{\Phi}_i$ and ψ_i the finite element basis functions, Lagrange polynomials of order $n+1$ and n.

One can remark that the basis functions associated to the pressure are not constructed with the null mean constraint. In practice, this constraint is added by Lagrange multiplier or penalty method as we will see in the next section.

It can be proven theoretically that the Stokes equations solved with the formulation (2.9) are not stable if the same polynomial order is used for the velocity and for the pressure. These instabilities can be avoided by using the stabilization techniques described in section 1.1 and adapted to the Stokes problem. We choose another approach which consists of using the so-called Taylor-Hood finite element. They are defined by elements with polynomial approximations of order n for the pressure and $n+1$ for the fluid. This explains our space discretization choice $\mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$. In practice, Lagrange basis functions are used.

Let us introduce now $(\boldsymbol{u}_h, p_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$ and $(\boldsymbol{v}_h, q_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$ the discrete versions of $(\boldsymbol{u}, p)$ and $(\boldsymbol{v}, q)$. The problem (2.8) discretized reads now, for a given $\boldsymbol{f}_h$, find $(\boldsymbol{u}_h, p_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$ so that $\forall\, (\boldsymbol{v}_h, q_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$:

$$\begin{aligned} \int_\Omega \mu \boldsymbol{\nabla} \boldsymbol{u}_h : \boldsymbol{\nabla} \boldsymbol{v}_h - \int_\Omega p_h \nabla \cdot \boldsymbol{v}_h &= \int_\Omega \boldsymbol{f}_h \cdot \boldsymbol{v}_h \\ \int_\Omega q_h \nabla \cdot \boldsymbol{u}_h &= 0 \end{aligned} \tag{2.9}$$

As the problem (2.9) stands for all $(\boldsymbol{v}_h, q_h)$, it stands in particular for $\boldsymbol{v}_h = \sum_{i=1}^{N_{\text{dof}}^{\mathbb{U}_h}} \boldsymbol{\Phi}_i$ and $q_i = \sum_{i=1}^{N_{\text{dof}}^{\mathbb{P}_h}} \psi_i$ the basis functions. Thus we can re-write the problem (2.9) in an algebraic representation by introducing the matrices

$$A,\ A_{ij} = \int_\Omega \mu(\boldsymbol{\nabla} \boldsymbol{\Phi}_j : \boldsymbol{\nabla} \boldsymbol{\Phi}_i),$$

$$B,\ B_{ij} = \int_\Omega -\psi_j \nabla \cdot \boldsymbol{\Phi}_i,$$

and the vectors

$$\begin{aligned} &U,\ \boldsymbol{u}_j \\ &P,\ p_j \\ &F,\ \boldsymbol{f}_j. \end{aligned}$$

The problem (2.9) can be written as

$$\underbrace{\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix}}_{\boldsymbol{D}} \underbrace{\begin{pmatrix} U \\ P \end{pmatrix}}_{\boldsymbol{U}} = \underbrace{\begin{pmatrix} F \\ 0 \end{pmatrix}}_{\boldsymbol{F}} \tag{2.10}$$

Finally, a sparse problem $\boldsymbol{DU} = \boldsymbol{F}$ is obtained and can be solved numerically.

Impose vanishing mean pressure

The problem (2.7) is not posed correctly because when only Dirichlet boundary conditions are prescribed on $\partial\Omega$, the pressure is only defined by its gradient and thus exists with an infinity of possible solutions. The usual solution consists of imposing

$$\int_\Omega p = 0.$$

There are two ways to impose this constraint, and both of them are implemented in our framework. The first method consists of adding a Lagrange multiplier λ to impose the constraint. The Lagrange multiplier is a single constant since the pressure only needs one constant to fix its value. The pressure no longer belongs to $L^2_0(\Omega)$ but $L^2(\Omega)$ since the zero mean value constraint is now imposed by λ. The variational formulation reads then, for a given $\boldsymbol{f} \in [L^2(\Omega)]^d$, find $(\boldsymbol{u}, p, \lambda) \in [H^1(\Omega)]^d \times L^2(\Omega) \times \mathbb{R}$ so that $\forall (\boldsymbol{v}, q, \nu) \in [H^1_0(\Omega)]^d \times L^2(\Omega) \times \mathbb{R}$:

$$\begin{aligned} \int_\Omega \mu \boldsymbol{\nabla u} : \boldsymbol{\nabla v} - \int_\Omega p \nabla \cdot \boldsymbol{v} &= \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \\ \int_\Omega q \nabla \cdot \boldsymbol{u} + \int_\Omega \lambda\, q &= 0 \\ \int_\Omega p\, \nu &= 0 \end{aligned} \tag{2.11}$$

The discretizations of $(\boldsymbol{u}, p)$ are the same as in the previous formulation and give $(\boldsymbol{u}_h, p_h) \in \mathbb{U}^{n+1}_h \times \mathbb{P}^n_h$. The discretized Lagrange multiplier λ_h is taken in a space $\mathbb{L}^0_h$ for which the basis functions are continuous polynomials of degree 0. That is to say a single constant for all Ω, this constant is simply 1. Let us name the vector associated to the coupling between the pressure and the Lagrange multiplier, which is simply the integral of the basis functions associated to the pressure:

$$S_j = \int_\Omega \psi_j. \tag{2.12}$$

The algebraic representation of the discretized problem then reads:

$$\begin{pmatrix} A & B & 0 \\ B^T & 0 & S \\ 0 & S^T & 0 \end{pmatrix} \begin{pmatrix} U \\ P \\ \lambda_h \end{pmatrix} = \begin{pmatrix} F \\ 0 \\ 0 \end{pmatrix} \tag{2.13}$$

The main advantage of this method is that only one degree of freedom is added to the system and the constraint is fulfilled exactly. The drawback is that in Feel++ , the choice to construct the solution $\boldsymbol{U}$ either in a composite space $\boldsymbol{U} \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n \times \mathbb{L}_h^0$ or $\boldsymbol{U} \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$ has to be made at compilation time. Thus, for a very general application for which the boundary conditions have to be changed by the user without recompiling the code, two options remain: i) compile two versions of the code and choose at execution time the one to run according to the needs. ii) Use a penalty term to impose the zero mean pressure.

Indeed, the second method to impose the zero mean pressure is to add a penalty term to the second equation of (2.7) insuring that the mean value of p is very small. The weak problem reads, for a given $\boldsymbol{f} \in [L^2(\Omega)]^d$, find $(\boldsymbol{u}, p) \in [H^1(\Omega)]^d \times L^2(\Omega)$ so that $\forall (\boldsymbol{v}, q) \in [H_0^1(\Omega)]^d \times L^2(\Omega)$:

$$\begin{aligned} \int_\Omega \mu \boldsymbol{\nabla u} : \boldsymbol{\nabla v} - \int_\Omega p \nabla \cdot \boldsymbol{v} &= \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \\ \int_\Omega q \nabla \cdot \boldsymbol{u} + \int_\Omega \varepsilon\, p\, q &= 0 \end{aligned} \tag{2.14}$$

with ε a small scalar. The corresponding algebraic representation is given by:

$$\begin{pmatrix} A & B \\ B^T & \varepsilon \boldsymbol{I}_\mathrm{d} \end{pmatrix} \begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix} \tag{2.15}$$

The advantage of this method is that it does not change the composite space in which belongs the solution $\boldsymbol{U}$. Thus, the choice to impose or not the null mean pressure can be done automatically in the code without recompiling. As a drawback, the constraint is only imposed approximately whereas it was exact with a Lagrange multiplier. An added drawback is that the penalty term deteriorates the matrix conditioning making the system more difficult to solve. Finally, the tolerance of the iterative solver used to solve the algebraic system has to be tuned with respect to the penalty term, which can be tricky.

Boundary conditions

The boundary conditions have been purposefully left apart previously. Let us now describe how we treat them in our framework. First of all, let us re-write the problem (2.7) by adding both Neumann and Dirichlet boundary conditions. The boundary having Dirichlet conditions is called $\partial\Omega_\mathrm{D}$ and the one having Neumann $\partial\Omega_\mathrm{N}$. The Neumann boundary conditions for Stokes equations impose the normal stress at the boundary. Thus, we will re-write (2.7) in a form making appear $\boldsymbol{\sigma}$ and the strain $\boldsymbol{D}$. Of course this is not compulsory, and one can use the form (2.7) and get the same results. The problem reads:

$$\begin{aligned} -\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} &= \boldsymbol{f} && \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} &= 0 && \text{in } \Omega, \\ \boldsymbol{u} &= \boldsymbol{u}_\mathrm{D} && \text{on } \partial\Omega_\mathrm{D}, \\ \boldsymbol{\sigma} &= \boldsymbol{\sigma}_\mathrm{N} = -p_\mathrm{N} \boldsymbol{I}_\mathrm{d} + 2\mu \boldsymbol{D}(\boldsymbol{u}_\mathrm{N}) && \text{on } \partial\Omega_\mathrm{N}. \end{aligned} \tag{2.16}$$

With $\boldsymbol{u}_\mathrm{D}$ and $\boldsymbol{\sigma}_\mathrm{N}$ some known functions on the boundary. One multiplies (2.16) by test functions, integrate by parts and remark that $\boldsymbol{\nabla v}$ can be replaced by $\boldsymbol{D}(\boldsymbol{v})$ (see appendix

A, one obtains the problem, for a given $\boldsymbol{f} \in [L^2(\Omega)]^d$ find $(\boldsymbol{u}, p) \in [H^1(\Omega)]^d \times L^2(\Omega)$ so that $\forall (\boldsymbol{v}, q) \in [H^1(\Omega)]^d \times L^2(\Omega)$:

$$\begin{aligned} \int_\Omega 2\mu \boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{D}(\boldsymbol{v}) - \int_\Omega p \nabla \cdot \boldsymbol{v} &= \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} + \int_{\partial\Omega_\mathrm{N}} (\boldsymbol{\sigma}_\mathrm{N} \boldsymbol{n}) \cdot \boldsymbol{v} \\ \int_\Omega q \nabla \cdot \boldsymbol{u} &= 0. \end{aligned} \tag{2.17}$$

Thus, the Neumann boundary terms arise directly from the integration by parts of the Stokes equations since we do not take $\boldsymbol{v} \in [H_0^1(\Omega)]^d$ which makes the boundary terms vanish.
The Dirichlet boundary conditions can be treated in two ways. First they can be imposed strongly by modifying the entries in the matrix $\boldsymbol{A}$ and the right hand side $\boldsymbol{F}$ to put the exact values of $\boldsymbol{u}_\mathrm{N}$ on the entries where it is known. FEEL++ allows performance of this operation by a simple function `on` as shown on listing 2.1.

Listing 2.1: Imposing a simple shear flow on Top and Bottom boundaries of a domain with FEEL++

```
//shear
form2(Xh, Xh, A)+=
  +on(markedfaces(mesh, "Top"    ), u, F,  u_shear * oneX() )
  +on(markedfaces(mesh, "Bottom" ), u, F, -u_shear * oneX() );
```

The advantages of strongly imposing the Dirichlet boundary conditions are:

i) the boundary conditions are imposed exactly

ii) since the keyword `on` is already a part of FEEL++ language, the condition is very simple to add and no further changes in the variational formulation are needed

whereas the drawbacks are:

i) the matrix $\boldsymbol{A}$ and the vector $\boldsymbol{F}$ have to be modified together every time one of them is modified. Thus, in an application where only the external forces are changing in time, one has to re-impose the boundary conditions and modify the matrix $\boldsymbol{A}$ every iteration.

ii) for some matrices, finding the good entry to modify can be computationally costly

Thus another way to impose Dirichlet boundary conditions has been proposed by Nitsche (1971). The idea is to impose weakly the Dirichlet boundary conditions by adding a penalty term. This way, the boundary condition is imposed the same way as the Neumann boundary conditions, that is to say, included in the variational formulation. We multiply the problem (2.16) by test functions $(\boldsymbol{v}, q) \in [H^1(\Omega)]^d \times L^2(\Omega)$ and integrate by parts. We then add a penalty term with a factor $\frac{\gamma}{h}$ which insures consistently that the boundary condition is satisfied. The problem reads, for a given $\boldsymbol{f} \in [L^2(\Omega)]^d$ find $(\boldsymbol{u}, p) \in [H^1(\Omega)]^d \times L^2(\Omega)$ so that $\forall (\boldsymbol{v}, q) \in [H^1(\Omega)]^d \times L^2(\Omega)$:

$$\begin{aligned} \int_\Omega 2\mu \boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{D}(\boldsymbol{v}) - \int_{\partial\Omega_\mathrm{D}} (\boldsymbol{\sigma}\boldsymbol{n}) \cdot \boldsymbol{v} &- \int_\Omega p \nabla \cdot \boldsymbol{v} + \int_{\partial\Omega_\mathrm{D}} \frac{\gamma}{h} \boldsymbol{u} \cdot \boldsymbol{v} \\ &= \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} + \int_{\partial\Omega_\mathrm{D}} \frac{\gamma}{h} \boldsymbol{u}_\mathrm{D} \cdot \boldsymbol{v} \\ \int_\Omega q \nabla \cdot \boldsymbol{u} &= 0. \end{aligned} \tag{2.18}$$

The parameter γ has to be set strong enough to impose correctly the condition and low enough not to create instabilities. A typical value lies between 1 and 20. The advantages of this method are

i) in the common case discussed previously where only the vector $\boldsymbol{F}$ changes during successive iterations, the boundary conditions in the matrix $\boldsymbol{A}$ are set once for all the simulation

ii) every boundary conditions (Neumann and Dirichlet) can be treated the same way.

The drawbacks are

i) the parameter γ has to be fixed *by hand*. But the range of γ for which the boundary conditions are correctly imposed can be quite large and usually a value between 10 and 20 is satisfying

ii) the boundary conditions are not imposed exactly. However, the formulation is still consistent and optimal convergence rates are still obtained.

2.1.2 Navier-Stokes equations

The Navier-Stokes equation

The Navier-Stokes equations govern fluid motion when neither the viscosity neither the inertia is negligible. We will only deal with the incompressible Navier-Stokes equations when the density is constant; hence we deal with incompressible fluid flow. This equation is obtained by writing the momentum conservation:

$$\rho \frac{D\boldsymbol{u}}{Dt} = \boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \boldsymbol{f} \tag{2.19}$$

where ρ is the density of the fluid. By expressing the stress tensor for a Newtonian fluid, developing the divergence as done previously, and developing the Lagrangian derivative $\dfrac{D}{Dt}$, one obtains:

$$\rho(\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}) - \mu \boldsymbol{\Delta} \boldsymbol{u} + \boldsymbol{\nabla} p = \boldsymbol{f}, \tag{2.20}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{2.21}$$

which are the Navier-Stokes incompressible equations. The framework we used to solve these equations has been developed by Vincent Chabannes during his PhD [17]. In this section, we will describe briefly the method used. For more details, the interested reader should see [17].

Temporal discretization

The time is discretized by a time step Δt so that at iteration n, $t = n\Delta t$. The time derivative is discretized by a finite difference scheme of arbitrary high order. We use the backward differentiation formulation **BDF**. For a given order q, the $\mathbf{BDF}_q$ scheme is given by:

$$\partial_t \boldsymbol{u}^{n+1} \approx \frac{\beta_{-1}}{\Delta t} \boldsymbol{u}^{n+1} - \sum_{j=0}^{q-1} \frac{\beta_j}{\Delta t} \boldsymbol{u}^{n-j}. \tag{2.22}$$

The coefficients β_j for $\mathbf{BDF}_q$ schemes up to order $q = 4$ are given in the table 2.1.

q	β_{-1}	β_0	β_1	β_2	β_3
1	1	1			
2	3/2	2	-1/2		
3	11/6	3	-3/2	1/3	
4	25/12	4	-3	4/3	-1/4

Table 2.1: $\mathbf{BDF}_q$ schemes up to order $q = 4$

Spatial discretization

The variational formulation is given by find $(\boldsymbol{u}^{n+1}, p^{n+1}) \in [H^1(\Omega)]^d \times L^2(\Omega)$ so that $\forall(\boldsymbol{v}, q) \in [H^1(\Omega)]^d \times L^2(\Omega)$:

$$\begin{aligned} \int_\Omega \rho\partial_t \boldsymbol{u}^{n+1} \cdot \boldsymbol{v} + \int_\Omega \rho((\boldsymbol{u}^{n+1} \cdot \boldsymbol{\nabla})\boldsymbol{u}^{n+1}) \cdot \boldsymbol{v} - \int_\Omega \mu \boldsymbol{\nabla}\boldsymbol{u}^{n+1} : \boldsymbol{\nabla}\boldsymbol{v} \\ + \int_\Omega p^{n+1} \nabla \cdot \boldsymbol{v} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}, \\ \int_\Omega q \nabla \cdot \boldsymbol{u}^{n+1} = 0. \end{aligned} \tag{2.23}$$

One introduces then $(\boldsymbol{u}_h, p_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$ and $(\boldsymbol{v}_h, q_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$ the discrete versions of $(\boldsymbol{u}, p)$ and $(\boldsymbol{v}, q)$ has for equation (2.9). The problem finally reads: find $(\boldsymbol{u}_h, p_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$ so that $\forall(\boldsymbol{v}_h, q_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$:

$$\begin{aligned} \int_\Omega \rho\partial_t \boldsymbol{u}_h^{n+1} \cdot \boldsymbol{v}_h + \int_\Omega \rho((\boldsymbol{u}_h^{n+1} \cdot \boldsymbol{\nabla})\boldsymbol{u}_h^{n+1}) \cdot \boldsymbol{v}_h - \int_\Omega \mu \boldsymbol{\nabla}\boldsymbol{u}_h^{n+1} : \boldsymbol{\nabla}\boldsymbol{v}_h \\ + \int_\Omega p_h^{n+1} \nabla \cdot \boldsymbol{v}_h = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}_h, \\ \int_\Omega q_h \nabla \cdot \boldsymbol{u}_h^{n+1} = 0. \end{aligned} \tag{2.24}$$

One obtains an algebraic representation of (2.24). Once again, the problem (2.24) is solved as one problem, i.e. both equations are solved at the same time. This strategy is called *monolithic*. The advantage of a monolithic strategy is that the solution fulfills all the equations at the same time. The drawback is that the algebraic problem can be difficult to solve numerically. Other ways exist where one splits the equations and solves them separately, they are called *projection* methods [41]. The advantage of projection methods is that they are simple to implement and the algebraic problems obtained by the different equations are simpler to solve than the one given by the monolithic method. The drawback of projection methods is that at each projection, the solution does not fulfill all the equations at the same time. In our framework, Navier-Stokes equations are always solved using a monolithic strategy.

Non linear solver

The equation (2.24) is non-linear. A non-linear solver is used to find its solution. Two different non linear solvers are available in the Navier-Stokes framework. A fixed point method and a Newton method. In general, we use the Newton method which requires

computation of the Jacobian of the matrix associated to the algebraic problem but converges in few iterations.

2.2 Coupling fluid and level set equations

2.2.1 Two-fluid flow

In a two-fluid flow simulation, several quantities depend on the position of the interface between the two-fluids. These quantities are the density, the viscosity, and the forces. Let us call Ω_1 and Ω_2 the domains on which are set the fluids 1 and 2 with densities and viscosities being respectively $(\rho_1, \mu_1) \in (\mathbb{R} \times \mathbb{R})$ and $(\rho_2, \mu_2) \in (\mathbb{R} \times \mathbb{R})$. Let us also call $\Omega = \Omega_1 \cup \Omega_2$ the whole domain. The interface between Ω_1 and Ω_2 is called Γ. A level set function ϕ is defined as the signed distance function to the interface, positive in Ω_1, negative in Ω_2 and zero on Γ as described in section 1.1.

In this section, we will add a subscript ϕ on all the fluid problem related quantities which depend on the interface position, and thus, on the level set function ϕ. We can define the density and viscosity of the whole domain as $\rho_\phi \in L^2(\Omega)$, $\mu_\phi \in L^2(\Omega)$:

$$\rho_\phi = \rho_2 + (\rho_1 - \rho_2) H_\varepsilon(\phi), \tag{2.25}$$
$$\mu_\phi = \mu_2 + (\mu_1 - \mu_2) H_\varepsilon(\phi). \tag{2.26}$$

The forces are given by a function depending on the interface position. This function can depend on the normal $\boldsymbol{n}$ and curvature κ of the interface, thus $\dfrac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}$ and $\nabla \cdot (\dfrac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|})$. It can also depend on higher derivatives of the curvature as we will see for vesicles. Let us say that one knows the expression $\boldsymbol{g}(\phi, \boldsymbol{n}, \kappa)$ of the density of interfacial forces on all the space. The force is the projection of this expression, $\boldsymbol{f}_\phi \in [L^2(\Omega)]^d$:

$$\boldsymbol{f}_\phi = \boldsymbol{g}(\phi, \boldsymbol{n}, \kappa). \tag{2.27}$$

The forces can be volume or surface forces. In the case of a surface force, the expression for $\boldsymbol{g}$ is in general written for the entire domain and the force is projected on the interface thanks to the delta function. It reads then:

$$\boldsymbol{f}_\phi = \boldsymbol{g}(\phi, \boldsymbol{n}, \kappa) \delta_\varepsilon(\phi). \tag{2.28}$$

A typical two-fluid flow problem is then given by the set of equations:

$$\rho_\phi = \rho_2 + (\rho_1 - \rho_2) H_\varepsilon(\phi), \tag{2.29}$$
$$\mu_\phi = \mu_2 + (\mu_1 - \mu_2) H_\varepsilon(\phi), \tag{2.30}$$
$$\frac{D(\rho_\phi \boldsymbol{u})}{Dt} - \boldsymbol{\nabla} \cdot (2\mu_\phi \boldsymbol{D}(\boldsymbol{u})) + \boldsymbol{\nabla} p = \boldsymbol{f}_\phi, \tag{2.31}$$
$$\nabla \cdot \boldsymbol{u} = 0, \tag{2.32}$$
$$\partial_t \phi + \boldsymbol{u} \cdot \boldsymbol{\nabla} \phi = 0. \tag{2.33}$$

One also adds boundary conditions. The equations (2.31), (2.32) and (2.33) could be solved using a monolithic strategy, but the algebraic problem might be difficult to solve. Moreover, we chose not to use a monolithic strategy because of the flexibility we wanted

to keep. Indeed, this framework is meant to be very generic, and its goal is to keep independent as much as possible the level set from the fluid. This way, many fluid models could be coupled with the level set framework without much change in the code. Thus, the framework we present decouples the definition of density, viscosity, and forces, the solution of the Navier-Stokes problem, and the advection of the level set. The associated algorithm is given by algorithm 3. One can note that the **FluidSolver** could be also a

Algorithm 2 Coupling between Fluid solver and Level Set solver.

Require: ϕ_0
  $\phi^0 \leftarrow \phi_0$
  for $n = 0$ to N_{t_f} **do**
    $\rho_\phi^{n+1} =$ **UpdateDensity**(ϕ^n), eq (2.25)
    $\mu_\phi^{n+1} =$ **UpdateViscosity**(ϕ^n), eq (2.26)
    $\boldsymbol{f}_\phi^{n+1} =$ **UpdateForces**$(\phi^n, \boldsymbol{n}(\phi^n), \kappa(\phi^n))$, eq (2.27)
    $(\boldsymbol{u}^{n+1}, p^{n+1}) =$ **FluidSolver**$(\rho_\phi^{n+1}, \mu_\phi^{n+1}, \boldsymbol{f}_\phi^{n+1})$, eq (2.31) and (2.32)
    $\phi^{n+1} =$ **AdvectionSolver**$(\boldsymbol{u}^{n+1})$, eq (2.33)
  end for

Stokes solver or another fluid model. The forces can take many forms, volume forces or interfacial forces. Thus, this strategy is very generic and many models can be tested with it.

2.2.2 Multi-fluid flow

As we have shown previously, one level set field can handle two-fluids. The two-fluids can have many interfaces but there are still only two different fluids. Some applications may need more than only two fluids in the domain. For example, one wants to simulate the behavior of a mixture of immiscible fluids. For a blood flow purpose, it can be interesting to have two different kind of red blood cells, with some cells very rigid to simulate diseases like drepanocitosis. It may also be interesting for a low number of objects, to have a sort of *Lagrangian* information on each objects. Indeed, in an application where one looks at the interaction between few vesicles, it can be useful to have the independent information of the position, velocity, surface and volume of each object which is impossible if only one level set is involved. In these cases, it is necessary to define more than one level set field. Each field will define the interface between two fluids. All the level set fields have to be advected, thus the equation (2.33) has to be solved for each level set field which can be computationally costly. However, from an implementation point of view, we will see in section 5.4 that many features can be shared between all the level set fields to enhance the performance of the application, like the matrix associated to the discretized equation (2.33) or the reinitialization framework.
In our many-fluid flow, each level set represents the interface between one domain and its exterior called domain 0. The domain 0 does not need a proper level set since it will be defined by the area in which there is no level set having a negative value.
From a fluid application point of view, one can say that we need to define a level set for each embedded fluid. The solvent being simply defined by the area in which there is no other fluid. That is to say that for an N fluid flow, we need $N - 1$ level set fields.

Link fluids and level set fields

Let us call N_{LS} the number of level set fields in a multi-fluid application and $\phi_1, ..., \phi_{N_{\text{LS}}}$ the different level set fields. We start intentionally the numbering by 1 to keep the number 0 for the ambient fluid defined according to the others. We define ϕ_0 in eq (2.34) as being the min value of all the other level set fields. This way, ϕ_0 is a field being negative if one of the other ϕ is negative, and positive else. This function has also the information of the positions of all the interfaces and can be used to distribute common values on the interface or compute general information like the area of the total interfaces.

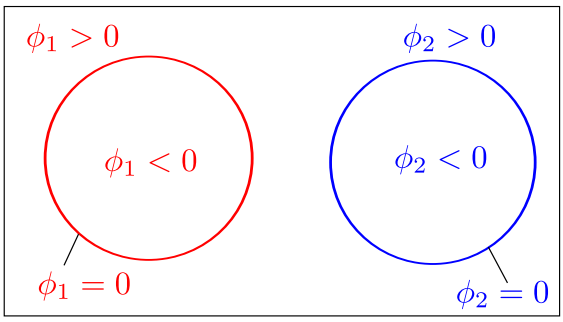
$$\phi_0 = \min(\phi_1, ..., \phi_{N_{\text{LS}}}) \tag{2.34}$$

The density and viscosity of such a multi-fluid system can be defined by using the fact that the function $H_\varepsilon(\phi_0)$ vanishes in the regions where one other fluid is supposed to be. Thus, the value ρ_0 and μ_0 has to be set in the area where $H_\varepsilon(\phi_0) = 1$. They are defined by:

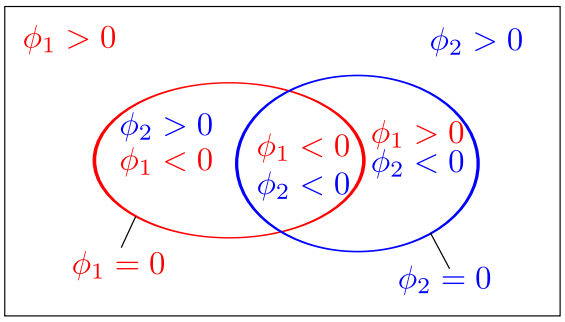
$$\rho_\phi = \rho_0 H_\varepsilon(\phi_0) + \sum_{i=1}^{N_{\text{LS}}} \rho_i (1 - H_\varepsilon(\phi_i)), \tag{2.35}$$

$$\mu_\phi = \mu_0 H_\varepsilon(\phi_0) + \sum_{i=1}^{N_{\text{LS}}} \mu_i (1 - H_\varepsilon(\phi_i)). \tag{2.36}$$

We emphasize the fact that the previous definitions hold for a mixture of immiscible fluids. It can be the case for example of drops of fluids immiscible or any object with a membrane (vesicle, red blood cell, white blood cell, capsule ...). More precisely, the definitions hold when the inner domains ($\phi_i < 0$) are not intersecting which is the case in figure 2.1(a). In the case of an intersection between the inner domains, like in figure 2.1(b), the definitions of the density and viscosity cannot be given by equations (2.35) and (2.36). The correct equations have to be given by the physics one wants to express. Anyway, it should not be too difficult since the intersected areas are easy to identify thanks to the sign of the different level set fields.



(a) No intersection between the inner domains of the level set fields.



(b) Intersection exists between the inner domains of the level set fields.

Figure 2.1: Intersection or not of the inner domains of the level set fields in a multi level set context.

Forces in multi-fluids context

The forces $\boldsymbol{f}_{i0}$ at the interface between one fluid $i > 0$ and the fluid 0 can be simply written as:

$$\boldsymbol{f}_{i0} = \boldsymbol{g}(\phi_i, \boldsymbol{n}_i, \kappa_i)\, \delta_\varepsilon(\phi_i) \tag{2.37}$$

where $\boldsymbol{g}$ is the known expression of the force possibly depending on the normal $\boldsymbol{n}_i$ and curvature κ_i of this interface. The forces $\boldsymbol{f}_{ij}$ at the interface between a fluid i and a fluid j can be distributed by multiplying the expression of the force $\boldsymbol{g}$ by the product of the two delta functions, which is non-zero only where the interfaces are touching. Such a force reads then:

$$\boldsymbol{f}_{ij} = \boldsymbol{g}(\phi_i, \boldsymbol{n}_i, \kappa_i)\, \delta_\varepsilon(\phi_i)\delta_\varepsilon(\phi_j). \tag{2.38}$$

Computing the normal and the curvature would require twice computing the derivative of ϕ_i for each interface if one uses $\boldsymbol{n}_i = \dfrac{\boldsymbol{\nabla}\phi_i}{|\boldsymbol{\nabla}\phi_i|}$ and $\kappa_i = \nabla \cdot \left(\dfrac{\boldsymbol{\nabla}\phi_i}{|\boldsymbol{\nabla}\phi_i|}\right)$. Fortunately, noticing that on the interface i, we have $\boldsymbol{n}_i = \boldsymbol{n}_0$ and $\kappa_i = \kappa_0$ since ϕ_0 has the information of the positions of all the interfaces, one can replace equation (2.38) by equation (2.39) and compute only once the normal and curvature for all the level set fields at the same time.

$$\boldsymbol{f}_{ij} = \boldsymbol{g}(\phi_i, \boldsymbol{n}_0, \kappa_0)\, \delta_\varepsilon(\phi_i)\delta_\varepsilon(\phi_j). \tag{2.39}$$

The same trick can be done for higher derivatives of the level set and in general for every geometrical parameter depending only on the position of the interface.

Solution of a multi-fluid flow problem

The generalization of the two-fluid flow problem to a multi-flow problem is simple. We have already seen how to define the density and the viscosity of all the fluids and how the different forces could be handled. The only difficulty is that the advection has to be done N_{LS} times, increasing the computational cost. A general algorithm to solve this problem is shown in algorithm 3.

Algorithm 3 Coupling between Fluid solver and multi level set.

Require: $\phi_1^0, ..., \phi_{N_{\text{LS}}}^0$
  for $n = 0$ to N_{t_f} **do**
    $\rho_\phi^{n+1} =$ **UpdateDensities**(ϕ_i^n), eq (2.35)
    $\mu_\phi^{n+1} =$ **UpdateViscosities**(ϕ_i^n), eq (2.36)
    $\boldsymbol{f}_\phi^{n+1} =$ **UpdateForces**$(\phi_i^n, \boldsymbol{n}(\phi_0^n), \kappa(\phi_0^n))$, eq (2.37) or (2.38)
    $(\boldsymbol{u}^{n+1}, p^{n+1}) =$ **FluidSolver**$(\rho_\phi^{n+1}, \mu_\phi^{n+1}, \boldsymbol{f}_\phi^{n+1})$, eq (2.31) and (2.32)
    for $i = 1$ to N_{LS} **do**
      $\phi_i^{n+1} =$ **AdvectionSolver**$(\boldsymbol{u}^{n+1})$, eq (2.33)
    end for
  end for

2.3 Benchmarks and tests

2.3.1 Rising of a bubble in a viscous fluid

We will present a benchmark which has been proposed in [45] and simulates the rising of a bubble of a viscous fluid in another viscous fluid. This benchmark is a good verification of the level set code, the Navier-Stokes solver and the coupling between both of them. The Navier-Stokes code that we used has been written and verified by Vincent Chabannes during his PhD [17]. We will show that we found a good agreement between the results of other groups and ours. We published these results in [26].

The benchmark objective is to simulate the rising of a 2D bubble in a Newtonian fluid. The idea is to put a fluid as a bubble in another fluid of a higher density. Both fluids are initially at rest. Because of the gravity, the bubble having the lower density rises. The equations solved are the incompressible Navier-Stokes equations for the fluid and the advection for the level set as described in section 2.2. To simulate the rising of a bubble, one adds a gravity force $\boldsymbol{f}_{\mathrm{g}}$ and a surface tension force $\boldsymbol{f}_{\mathrm{st}}$ defined by:

$$\boldsymbol{f}_{\mathrm{g}} = \rho_{\phi}\boldsymbol{g} \tag{2.40}$$

$$\boldsymbol{f}_{\mathrm{st}} = \int_{\Gamma} \sigma\kappa\boldsymbol{n} \tag{2.41}$$

with $\boldsymbol{g} =^{t} (0, 0.98)$ the gravity acceleration and σ the surface tension. We recall that $\boldsymbol{n} = \dfrac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}$ is the normal to the interface and $\kappa = \nabla \cdot (\dfrac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|})$ is its curvature.

The computational domain is $\Omega \times]0, T]$ where $\Omega = (0, 1) \times (0, 2)$ and $T = 3$. We denote Ω_1 the domain outside the bubble $\Omega_1 = \{\boldsymbol{x} | \phi(\boldsymbol{x}) > 0\}$, Ω_2 the domain inside the bubble $\Omega_2 = \{\boldsymbol{x} | \phi(\boldsymbol{x}) < 0\}$ and Γ the interface $\Gamma = \{\boldsymbol{x} | \phi(\boldsymbol{x}) = 0\}$. On the lateral walls, slip boundary conditions are imposed, *i.e.* $\boldsymbol{u} \cdot \boldsymbol{n} = 0$ and $\boldsymbol{t} \cdot (\nabla\boldsymbol{u} +^{t} \nabla\boldsymbol{u}) \cdot \boldsymbol{n} = 0$ where $\boldsymbol{n}$ is the unit normal to the interface and $\boldsymbol{t}$ the unit tangent. No slip boundary conditions are imposed on the horizontal walls *i.e.* $\boldsymbol{u} = \boldsymbol{0}$. The initial bubble is circular with a radius $r_0 = 0.25$ and centered on the point $(0.5, 0.5)$. The figure 2.2 taken from [45] shows the initial configuration of the simulation. We denote with indices 1 and 2 the quantities relative to the fluid in respectively in Ω_1 and Ω_2. The parameters of the benchmark are ρ_1, ρ_2, μ_1, μ_2 and σ and we define two dimensionless numbers. The first dimensionless number is the Reynolds number which is the ratio between inertial and viscous terms and is defined in this simulation by :

$$R_e = \frac{\rho_1 \sqrt{|\boldsymbol{g}|(2r_0)^3}}{\mu_1}. \tag{2.42}$$

The second one, is the Eötvös number which represents the ratio between the gravity force and the surface tension, it takes the form :

$$E_0 = \frac{4\rho_1 |\boldsymbol{g}| r_0^2}{\sigma}. \tag{2.43}$$

The table 2.2 reports the values of the parameters used for the two different test cases proposed in [45]. The quantities measured are $\boldsymbol{X}_{\mathrm{c}}$ the position of the center of mass of

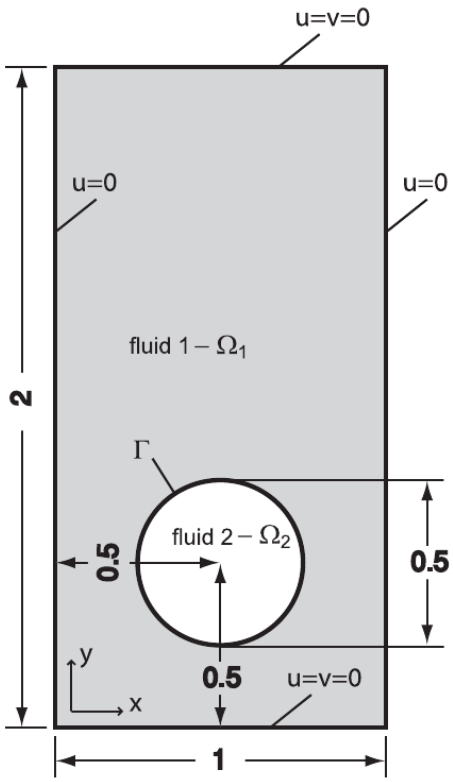


Figure 2.2: Initial configuration taken from [45]. In this notation $\boldsymbol{u} = (u, v)$.

Tests	ρ_1	ρ_2	μ_1	μ_2	σ	Re	E_0
Test 1 (ellipsoidal bubble)	1000	100	10	1	24.5	35	10
Test 2 (squirted bubble)	1000	1	10	0.1	1.96	35	125

Table 2.2: Numerical parameters taken for the benchmarks.

the bubble defined by

$$\boldsymbol{X}_{\mathrm{c}} = \frac{\int_{\Omega_2} \boldsymbol{x}}{\int_{\Omega_2} 1} = \frac{\int_{\Omega} \boldsymbol{x}(1 - H_\varepsilon(\phi))}{\int_{\Omega} (1 - H_\varepsilon(\phi))}, \tag{2.44}$$

the velocity of the center of mass $\boldsymbol{U}_{\mathrm{c}}$:

$$\boldsymbol{U}_{\mathrm{c}} = \frac{\int_{\Omega_2} \boldsymbol{u}}{\int_{\Omega_2} 1} = \frac{\int_{\Omega} \boldsymbol{u}(1 - H_\varepsilon(\phi))}{\int_{\Omega} (1 - H_\varepsilon(\phi))}, \tag{2.45}$$

and the circularity defined as the ratio between the perimeter of a circle which has the

same area and the perimeter of the bubble :

$$c = \frac{\left(4\pi \int_{\Omega_2} 1\right)^{\frac{1}{2}}}{\int_{\Gamma} 1} = \frac{\left(4\pi \int_{\Omega} (1 - H_\varepsilon(\phi))\right)^{\frac{1}{2}}}{\int_{\Omega} \delta_\varepsilon(\phi)}. \tag{2.46}$$

The results of both test cases are represented in figure 2.3 for test case 1 and in figure 2.4 for test case 2. One can see with the arrows and colors that the boundary conditions are correctly imposed, especially the slip boundary conditions on the left and right sides.

Three different groups have done the same benchmark and the results are presented in [45] and are available online [1]. The name of the codes, the institutions and the authors of the tests are reported in the table 2.3 taken from [45] in which we added our own code.

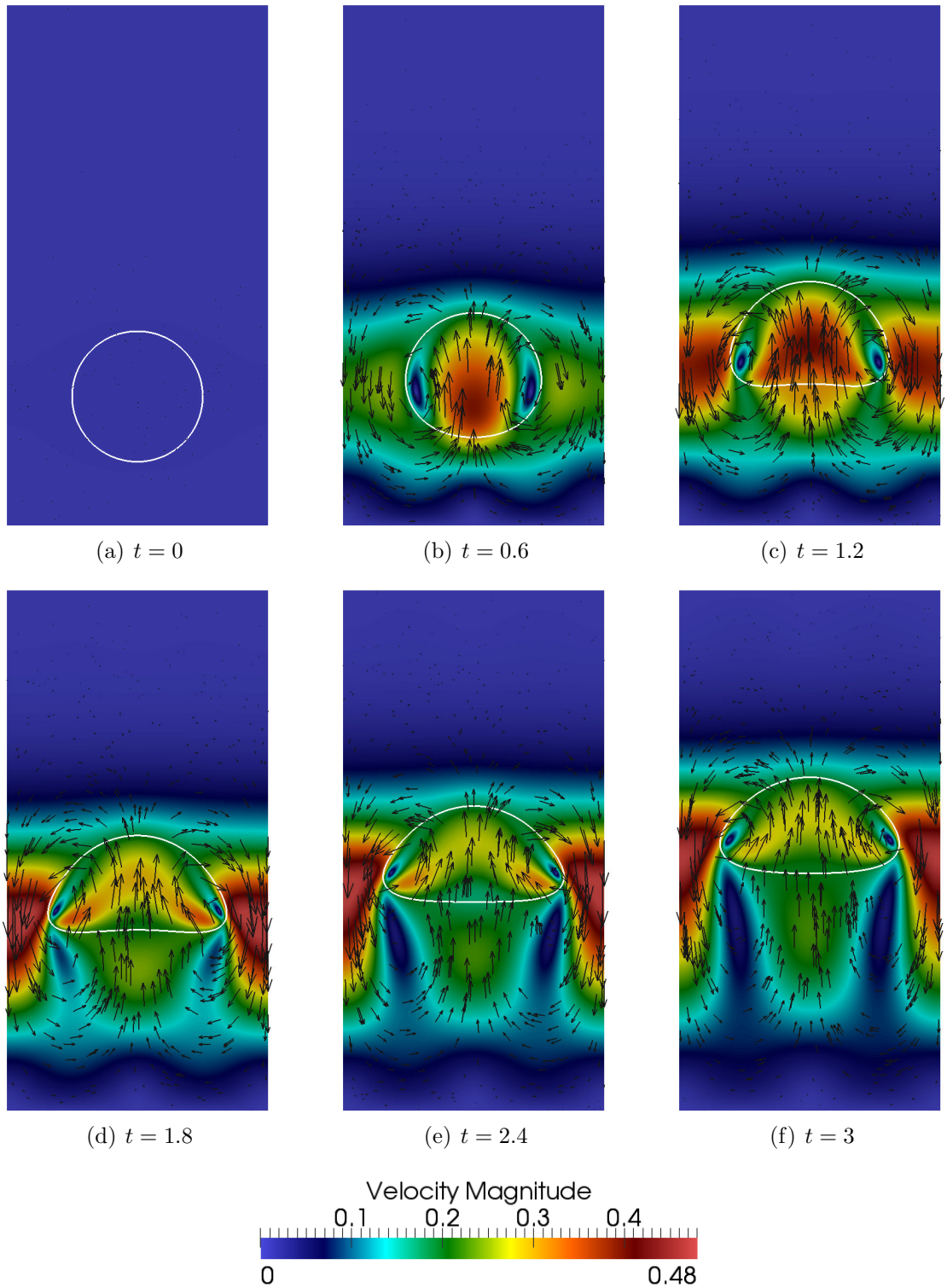
Group and affiliation	Code/Method
TU Dortmount, Inst. of Applied Math. *S. Turek, D. Kuzmin, S. Hysing*	**TP2D** FEM-Level Set
EPFL Lausanne, Inst. of Analysis ans Sci. Comp. *E.Burman, N.Parolini*	**FreeLIFE** FEM-Level Set
Uni. Magdeburg, inst. of Analysis and Num. Math. *L. Tobiska, S. Ganesan*	**MooNMD** FEM-ALE
Univ. Joseph Fourier, LIPhy. *V.Doyeux, Y.Guyot, V.Chabannes, C.Prud'homme, M.Ismail*	**Feel++** FEM-Level Set

Table 2.3: Summarize of the different groups having done the benchmark and the method used.

In the first test case, the energy cost to deform the bubble is strong enough to keep the shape of the bubble close to an ellipsoid. We will call this bubble the *ellipsoidal* bubble, whereas in the second test, the surface tension is weak and not sufficient to keep the bubble in one piece. The bottom part of the bubble lets appear some squirts, thus, this bubble will be called the *squirted* bubble. The quantities $\boldsymbol{X}_c$ and $\boldsymbol{U}_c$ are calculated directly from Feel++ using the discretized Heaviside function. Whereas the circularity c is calculated both by integrating the discretized Dirac function in Feel++ and by post processing using an interpolation tool of the software Paraview . Both circularity computation methods give the same results but we only show the post processed one. Indeed it shows almost no noise since thanks to Paraview we are able to reconstruct an iso 0 level set line by interpolation and integrate over Γ without the use of the delta function which always has some thickness. The final shapes of the bubbles are also monitored. We only plot the better approximation shape for the comparison. All the groups did not use the same approximations; the best approximation used for each group is reported in table 2.4.

The curves of the shape of the bubble, $\boldsymbol{X}_c$, $\boldsymbol{U}_c$ and c as functions of time are plotted on the same graph for different mesh sizes respectively in figures 2.5(a), 2.6(a), 2.7(a), 2.8(a) for the ellipsoidal bubble and 2.9(a), 2.10(a), 2.11(a), 2.12(a) for the squirted bubble.

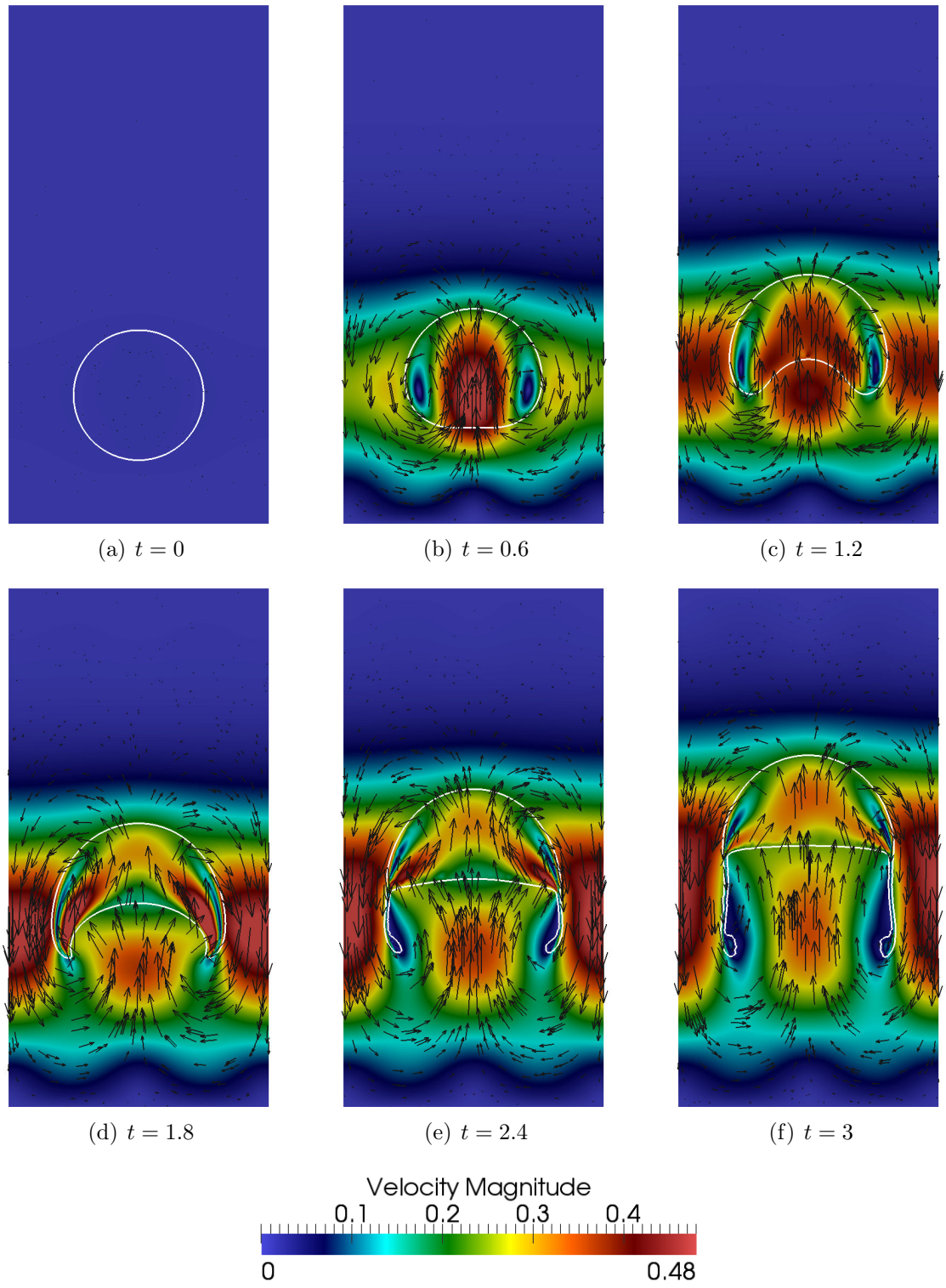
[1] http://www.featflow.de/en/benchmarks/cfdbenchmarking/bubble.html



(a) $t = 0$ (b) $t = 0.6$ (c) $t = 1.2$

(d) $t = 1.8$ (e) $t = 2.4$ (f) $t = 3$

Figure 2.3: Simulation of test case 1 at different times. The color is the magnitude of the velocity, the arrows represent the velocity vectors with a length proportional to the magnitude. The iso-0 line of the level set is represented by a white line.



(a) $t = 0$ (b) $t = 0.6$ (c) $t = 1.2$

(d) $t = 1.8$ (e) $t = 2.4$ (f) $t = 3$

Figure 2.4: Simulation of test case 2 at different times. The color is the magnitude of the velocity, the arrows represent the velocity vectors with a length proportional to the magnitude. The iso-0 line of the level set is represented by a white line.

Group	h	N_{elt}
FEEL++	0.00625	120174
TP2	0.003125	204800
FreeLIFE	0.00625	102400
MooNMD		8066

Table 2.4: Summary of the best spatial approximation for all the groups. h is the mesh size and N_{elt} is the number of elements. MooNMD has no typical mesh size. Since it used an adaptive mesh, the mean mesh size would not be a representative value.

The same parameters are plotted versus the 3 different group results in figures 2.5(b), 2.6(b), 2.7(b), 2.8(b) for the ellipsoidal bubble and 2.9(b), 2.10(b), 2.11(b), 2.12(b) for the squirted bubble. The curves can be compared by eye. Ones see that all the groups agree on the ellipsoidal bubble results whereas all groups have varying results for the squirted bubble. The difference comes from the squirts appearing on the bottom of the bubble which are difficult to resolve. In our case, the squirts are quite large and do not break from the bubble, this might be due to the reinitialization frequency set quite high (one reinitialization every 5 time step) which always make the bubble difficult to break. These part having a low velocity, the overall velocity of the bubble is lowered by the large amount of squirts in our simulation. This explains why our velocity curve in figure 2.11(b) is below the other curves. A more quantitative comparison uses few remarkable points that are quantitatively monitored. The quantitative points that are measured for the first test case are $c_{\min}$ the minimum of the circularity, $t_{c_{\min}}$ the time to attain this minimum, $u_{c_{\max}}$ the maximum velocity, $t_{u_{c_{\max}}}$ the time to reach it, and $y_c(t=3)$ the position of the bubble at final time ($t=3$).

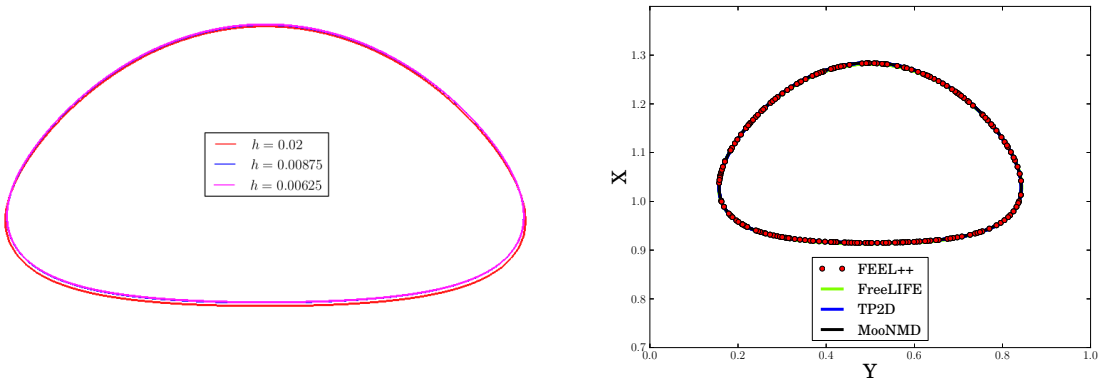
For the second test, we monitor the same quantities and add the second maximum velocity $u_{c_{\max_2}}$, and the time to reach it $t_{u_{c_{\max_2}}}$. The simulations are run for different mesh sizes and the benchmark quantities are reported in table 2.5 for the first case and table 2.6 for the second case with, for reference, the max and min values found by the groups in [45].

h	$c_{\min}$	$t_{c_{\min}}$	$u_{c_{\max}}$	$t_{u_{c_{\max}}}$	$y_c(t=3)$
lower bound	0.9011	1.8750	0.2417	0.9213	1.0799
upper bound	0.9013	1.9041	0.2421	0.9313	1.0817
0.02	0.8981	1.925	0.2400	0.9280	1.0787
0.01	0.8999	1.9	0.2410	0.9252	1.0812
0.00875	0.89998	1.9	0.2410	0.9259	1.0814
0.0075	0.9001	1.9	0.2412	0.9251	1.0812
0.00625	0.9001	1.9	0.2412	0.9248	1.0815

Table 2.5: Results comparison between benchmark values (lower and upper bounds) and ours for the ellipsoidal bubble.

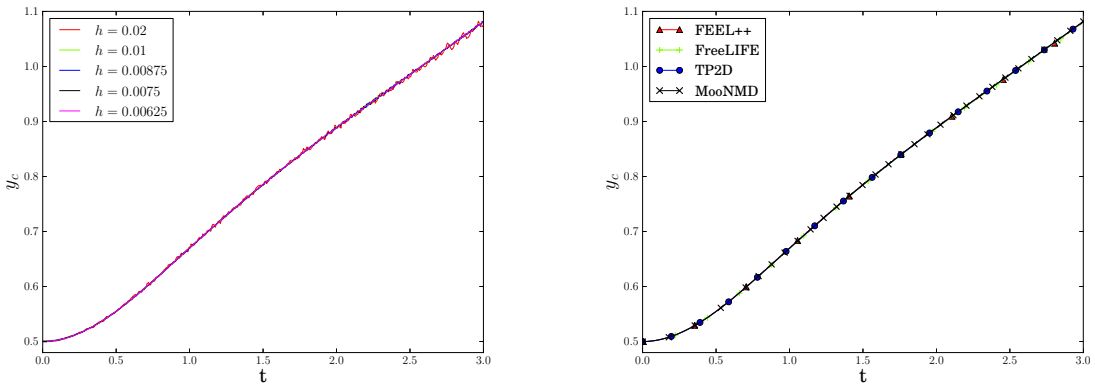
h	$c_{\min}$	$t_{c_{\min}}$	$u_{c_{\max_1}}$	$t_{u_{c_{\max_1}}}$	$u_{c_{\max_2}}$	$t_{u_{c_{\max_2}}}$	$y_c(t=3)$
lower bound	0.4647	2.4004	0.2502	0.7281	0.2393	1.9844	1.1249
upper bound	0.5869	3.0000	0.2524	0.7332	0.2440	2.0705	1.1380
0.02	0.4744	2.995	0.2464	0.7529	0.2207	1.8319	1.0810
0.01	0.4642	2.995	0.2493	0.7559	0.2315	1.8522	1.1012
0.00875	0.4629	2.995	0.2494	0.7565	0.2324	1.8622	1.1047
0.0075	0.4646	2.995	0.2495	0.7574	0.2333	1.8739	1.1111
0.00625	0.4616	2.995	0.2496	0.7574	0.2341	1.8828	1.1186

Table 2.6: Results comparison between benchmark values (lower and upper bounds) and ours for the squirted bubble



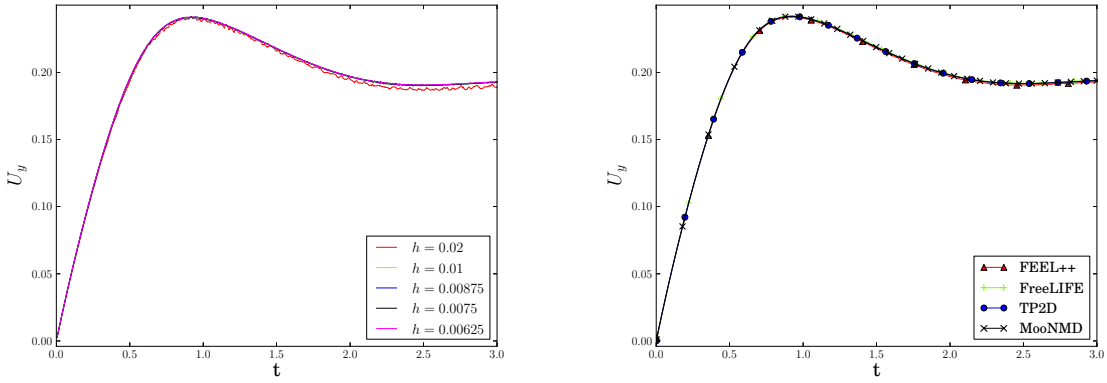
(a) Comparison of different mesh sizes. (b) Comparison with other codes.

Figure 2.5: Final shape ($t = 3$) of the ellipsoidal bubble.



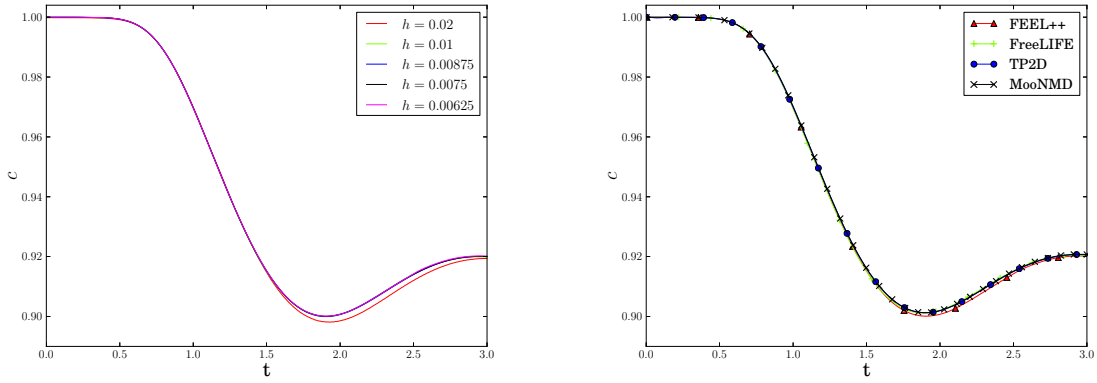
(a) Comparison of different mesh sizes. (b) Comparison with other codes.

Figure 2.6: y position of the ellipsoidal bubble versus time.



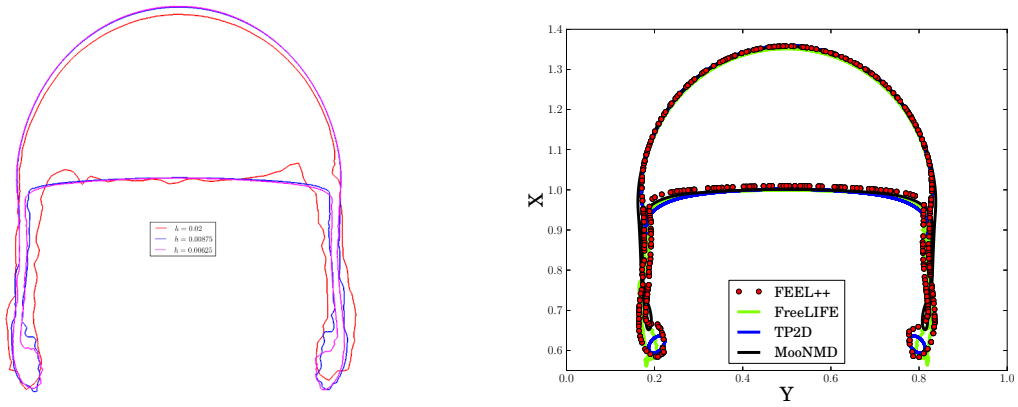
(a) Comparison of different mesh sizes. (b) Comparison with other codes.

Figure 2.7: Velocity magnitude the ellipsoidal bubble versus time



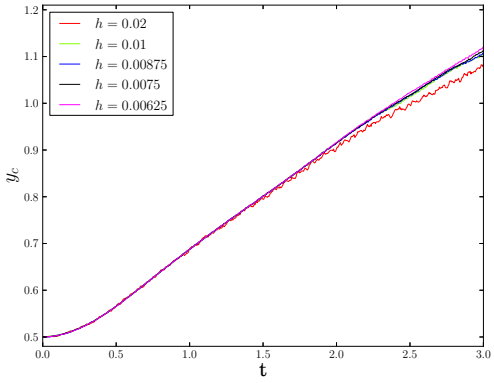
(a) Comparison of different mesh sizes. (b) Comparison with other codes.

Figure 2.8: Circularity of the ellipsoidal bubble versus time

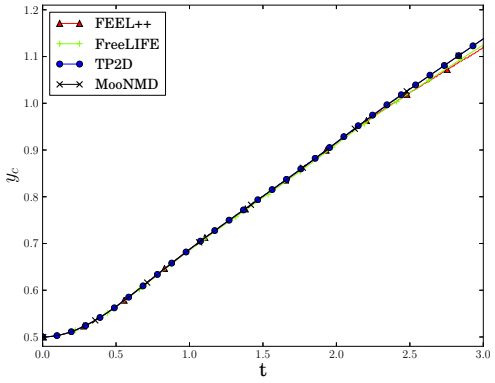


(a) Comparison of different mesh sizes. (b) Comparison with other codes.

Figure 2.9: Final shape ($t = 3$) of the squirted bubble.

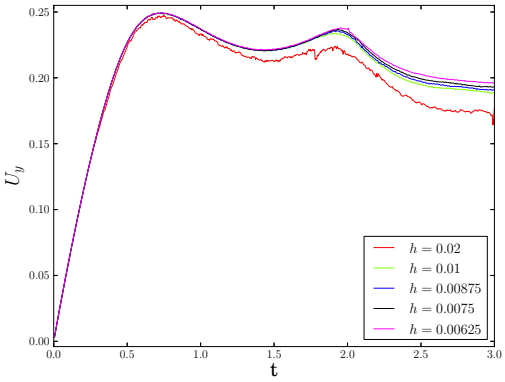


(a) Comparison of different mesh sizes.

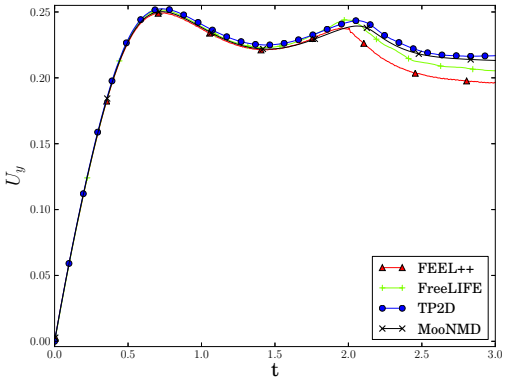


(b) Comparison with other codes.

Figure 2.10: y position of the squirted bubble versus time.

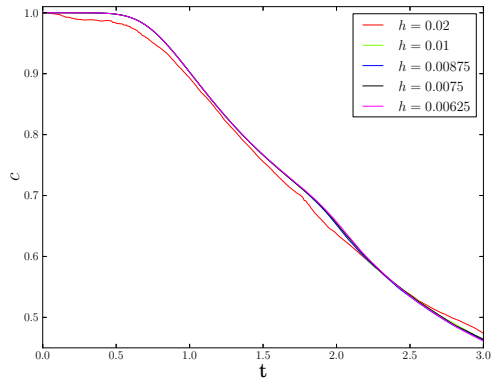


(a) Comparison of different mesh sizes.

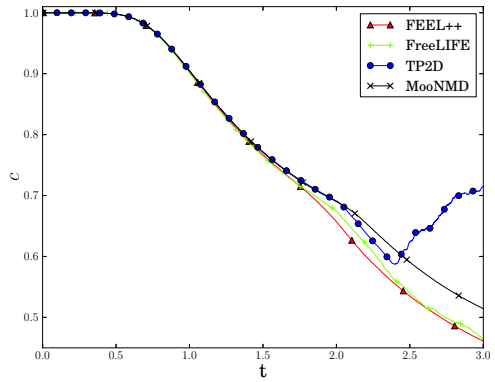


(b) Comparison with other codes.

Figure 2.11: Velocity magnitude the squirted bubble versus time



(a) Comparison of different mesh sizes.



(b) Comparison with other codes.

Figure 2.12: Circularity of the squirted bubble versus time

2.3.2 Rising of different fluid bubbles

This test is made to test the method described in section 2.2.2 in which one uses many level set fields to describe different fluids flowing in the domain. It consists of having 8 bubbles of fluids with different densities rising in another fluid. It is inspired by the benchmark presented in section 2.3.1, the bubbles are set in a fluid at rest and rise by gravity. We denote with a subscript number, the quantities related to the different fluids. The subscript 0 is kept for the surrounding fluid, which is not defined by a negative value of its own level set, but by the area where all the other level sets are not negative, as explained in section 2.2.2. The gravity force is given by :

$$\boldsymbol{f}_{\mathrm{g}} = \left(\rho_0 H_\varepsilon(\phi_0) + \sum_{i=1}^{8} \rho_i (1 - H_\varepsilon(\phi_i)) \right) \boldsymbol{g} \tag{2.47}$$

with $\boldsymbol{g} = {}^t(0, 0.98)$ the gravity acceleration.

The surface tension force is naturally given by the sum of the surface tension forces acting on all the interfaces :

$$\boldsymbol{f}_{\mathrm{st}} = \sum_{i=0}^{8} \sum_{j>i}^{8} \int_{\Gamma_{ij}} \sigma_{ij}\, \kappa_{ij}\, \boldsymbol{n}_{ij} \tag{2.48}$$

with Γ_{ij} the interface between fluid i and fluid j, κ_{ij} and $\boldsymbol{n}_{ij}$ the curvature and normal related to this interface. As already explained in section 2.2.2 with the equation (2.37), the interface between the ambient fluid and any other fluid can be handled by :

$$\int_{\Gamma_{0i}} 1 \approx \int_{\Omega} \delta_\varepsilon(\phi_i), \quad \text{with } \ i > 0 \tag{2.49}$$

and the interface between the other fluids is handled by the equation (2.38) that is to say

$$\int_{\Gamma_{ij}} 1 \approx \int_{\Omega} \delta_\varepsilon(\phi_i)\delta_\varepsilon(\phi_j) \quad \text{with } i > 0 \text{ and } j > i. \tag{2.50}$$

The surface tension σ_{ij} is an intrinsic property related to the interaction between two fluids (i and j). Consequently, the case of each fluid i interacting with any other fluid j by surface tension should be considered and all the σ_{ij} should be set. Of course $\sigma_{ij} = \sigma_{ji}$, this would make 36 different surface tensions to handle. There is no methodological problem to handle this, but for the sake of simplicity, in this particular application, we will only deal with the interaction with the surrounding fluid 0. Indeed, we allow a space between the bubbles large enough so that the bubbles do not touch each other. Thus, the only surface tensions to handle are σ_{0i} with $i = (1, ..., 8)$. Since only the interactions with interface 0 are considered, there is no ambiguity and the subscript 0 can be omitted. In what follows, we will denote Γ_i, κ_i, $\boldsymbol{n}_i$, σ_i the quantities Γ_{0i}, κ_{0i}, $\boldsymbol{n}_{0i}$, σ_{0i}. The surface tension reads then :

$$\boldsymbol{f}_{\mathrm{st}} = \sum_{i=1}^{8} \int_{\Omega} \sigma_i\, \kappa_i\, \boldsymbol{n}_i\, \delta_\varepsilon(\phi_i) = \sum_{i=1}^{8} \int_{\Omega} \sigma_i\, \kappa(\phi_i)\, \boldsymbol{n}(\phi_i)\, \delta_\varepsilon(\phi_i). \tag{2.51}$$

The quantities $\boldsymbol{n}_i$ and κ_i could be naturally calculated using $\boldsymbol{n}_i = \boldsymbol{n}(\phi_i)$ and $\kappa_i = \kappa(\phi_i)$. But an improvement is made recalling that $\phi_0 = \min(\phi_1, ..., \phi_8)$ contains the information

of the positions of all the interfaces at the same time. Thus $\boldsymbol{n}(\phi_0)$ has the same value as $\boldsymbol{n}(\phi_i)$ on the interface i. Hence the following equality stands :

$$\int_\Omega \boldsymbol{n}(\phi_i)\delta_\varepsilon(\phi_i) = \int_\Omega \boldsymbol{n}(\phi_0)\delta_\varepsilon(\phi_i) \quad \text{for all } 1 < i < 8. \tag{2.52}$$

The equivalent for κ_i is also true. The equality (2.52) is important when one recalls that one needs to differentiate the level set function to get the normal and differentiate one more time to get the curvature. Thus, for each time step, one can calculate $\boldsymbol{n}(\phi_0)$ and $\kappa(\phi_0)$ and economize the computing of 8 derivatives of the fields ϕ_i and 8 derivatives of $\boldsymbol{n}_i$. The expression of the force becomes :

$$\begin{aligned} \boldsymbol{f}_{\text{st}} &= \sum_{i=1}^{8} \int_\Omega \sigma_i \, \kappa(\phi_0) \, \boldsymbol{n}(\phi_0) \, \delta_\varepsilon(\phi_i) \\ \boldsymbol{f}_{\text{st}} &= \sum_{i=1}^{8} \int_\Omega \sigma_i \, \nabla \cdot \left(\frac{\boldsymbol{\nabla}\phi_0}{|\boldsymbol{\nabla}\phi_0|} \right) \frac{\boldsymbol{\nabla}\phi_0}{|\boldsymbol{\nabla}\phi_0|} \, \delta_\varepsilon(\phi_i) \end{aligned} \tag{2.53}$$

The equation (2.53) is the form of the surface tension force used in this test.

The computational domain is $\Omega \times]0, T[$ with $\Omega = (0, 8) \times (0, 2)$ and $T = 4.5$. On the lateral walls, slip boundary conditions are imposed, $\boldsymbol{u} \cdot \boldsymbol{n} = 0$ and $\boldsymbol{t} \cdot (\nabla \boldsymbol{u} + {}^t\nabla \boldsymbol{u}) \cdot \boldsymbol{n} = 0$ where $\boldsymbol{n}$ is the unit normal to the interface and $\boldsymbol{t}$ the unit tangent. No slip boundary conditions are imposed on the horizontal bottom wall, $\boldsymbol{u} = 0$. And Neumann homogeneous boundary conditions are imposed on the horizontal top wall, $\boldsymbol{\sigma} \cdot \boldsymbol{n} = 0$. The initial parameters given for each bubble are reported in table 2.7 and a scheme of the initial configuration is given in figure 2.13. The parameters of the fluids for the bubbles are

Bubble	1	2	3	4	5	6	7	8
r_0	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
x_0	0.5	1.5	2.5	3.5	4.5	5.5	6.5	7.5
y_0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
σ	50	50	50	50	50	50	50	50
μ	1	1	1	1	1	1	1	1
ρ	800	700	600	500	400	300	200	100

Table 2.7: Initial parameters given for the 8 bubbles, r_0 being the radius, (x_0, y_0) the initial position, σ the surface tension with the surrounding fluid, μ the viscosity, and ρ the density.

identical except the density. This makes the bubbles rise at different velocities. The local Reynolds number for a single bubble varies slightly since the densities and the velocities of each bubble are different, but all of them are of the same order. The quantities monitored are the same as for the benchmark presented in section 2.3.1, that is to say the center of mass $\boldsymbol{X}_{\text{c}}$ (and in particular its y component y_c), the mean velocity $\boldsymbol{U}_{\text{c}}$ and the circularity c given by respectively equations (2.44), (2.45) and (2.46). The simulation has been run on the super computer `SuperMUC` on 10 processors. The time step was set to $dt = 0.01$, the final time to 4.5 and the mesh size to $h = 0.03$. With this mesh size, the number of elements in the mesh was 41902, and all simulations lasted 50 minutes. This makes one

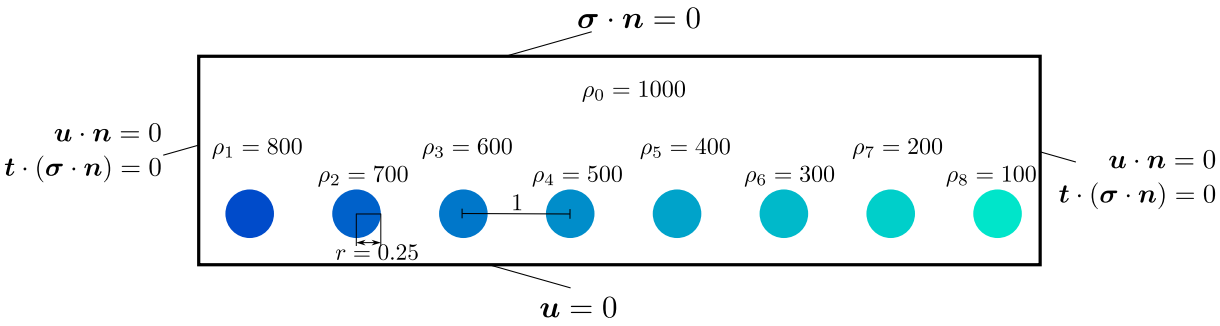
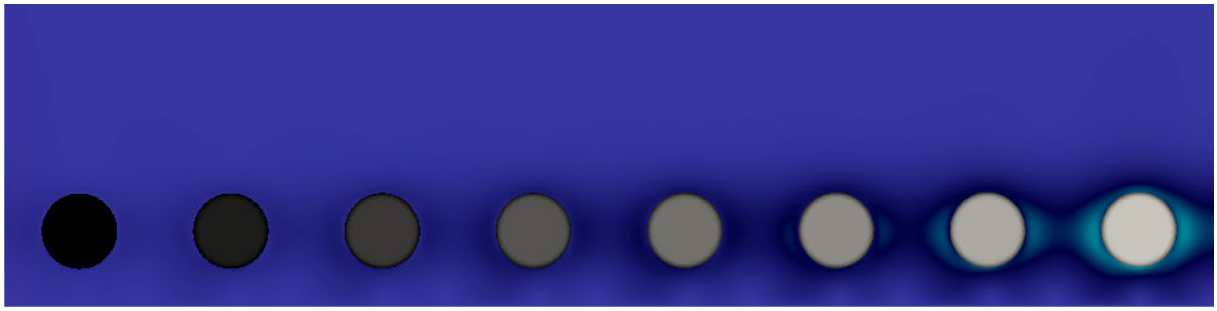
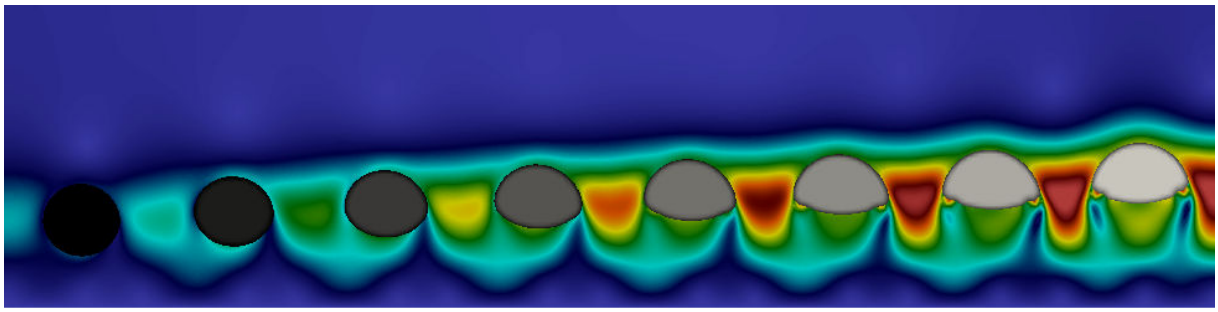


Figure 2.13: Initial geometry, boundary conditions and densities of the different bubbles.

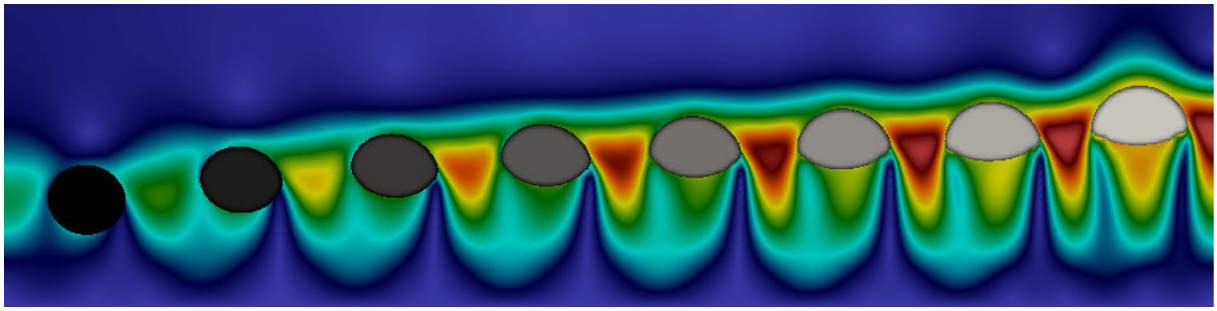
total iteration last for about 7 seconds. The figure 2.14 shows the colored results for 4 different times. The densities of the bubbles decrease from the left to the right. Thus the first bubble moves very slowly compared to the last one. The color on background represents the velocity magnitude of the fluid. One sees that as for the single bubble, the velocity is the highest on the sides of the bubbles where the fluid is pushed down. In figure 2.15 we have plotted the position of the center of mass of the bubbles as a function of time. As we can see, the shapes seem similar to the single bubble. We can see clearly a difference between all the bubbles, which attains a higher position for the lower density ones as expected. We can add that due to the other bubbles, the system is not symmetrical in the x direction at the opposite of the single bubble simulation, and this symmetry breaking induces a movement of the center of mass of the bubble in the x direction. The figure 2.16 shows the velocities of the center of mass of the bubbles. The fastest bubble is the bubble 8, its velocity attains a maximum, then, decreases to a local minimum and finally increases again. The bubbles 7, 6, 5 follow almost the same behavior than the single bubbles, being that the velocity increases quickly and attains a maximum then decreases and finally increases again to attain a second maximum. The bubbles 4, 3, 2, and 1 have a velocity increasing to attain a kind of plateau, the maximum velocity attained is smaller for the higher densities. Finally the figure 2.17 shows the circularities of the bubbles. The circularities show a noise which is approximately of the order of magnitude of the mesh. This is due to the fact that for this simulation, the circularity is computed directly by integrating the perimeter and the surface of the bubbles on the mesh without any post processing. Thus, the perimeter being very sensitive to the mesh size, the circularities show a bigger noise that the one post processed in the previous section. We recall that one advantage of using a multi-level set method is that it is possible to do such a computation directly in the simulation. Computing the circularity of many interfaces with only one level set is not possible in general since there is no way to differentiate the interfaces. We can see that the bubbles 8 to 4 show the same behavior as the single bubble with local variation stronger for the lower density bubbles. This can be explained by the fact that, this bubble going faster, the hydrodynamical forces acting on it are stronger and thus lead to more deformation. Finally, the higher density bubbles show weak deformations which can be directly seen on the graph since the circularity is roughly unchanged.



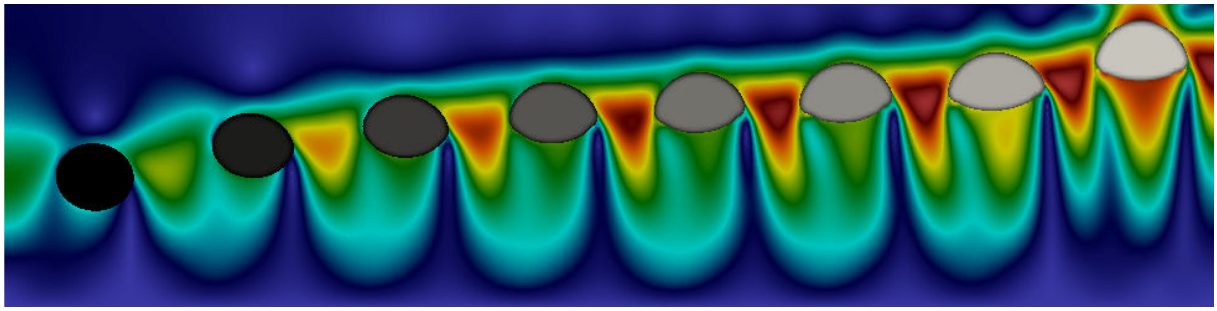
(a) $t = 0.2$



(b) $t = 1.6$



(c) $t = 3$



(d) $t = 4.5$

Figure 2.14: Rising of 8 bubbles of different fluids. Velocity magnitude $|\boldsymbol{u}|$ is represented in color outside the bubbles. The bubbles are colored by their density ρ.

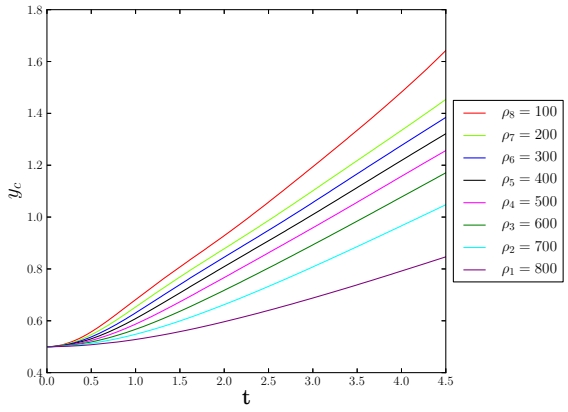


Figure 2.15: Position of each bubble as a function of time.

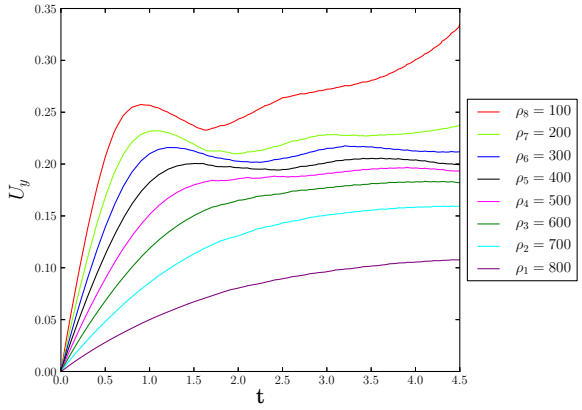


Figure 2.16: Velocity of each bubble as a function of time.

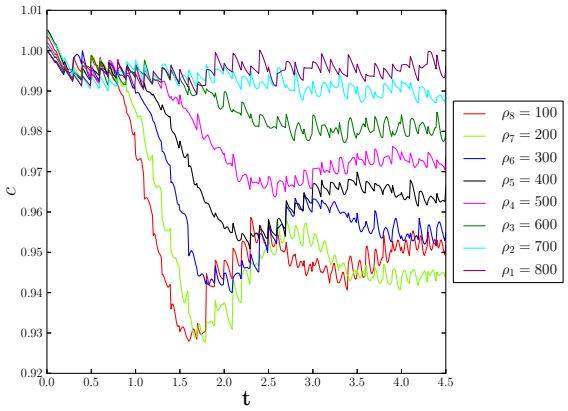


Figure 2.17: Circularity of each bubble as a function of time.

2.4 Oscillations of a bubble in a fluid at rest

2.4.1 Motivation

A validation of our code has been suggested to us by Philippe Marmottant and Olivier Vincent, experimental physicists in our laboratory. In the context of his PhD, Olivier Vincent [111] studied the cavitation of a bubble in a hydrogel. His experimental subject was composed of a hydrogel with many holes in it, these holes are called inclusions. These inclusions are filled with water. The hydrogel is then dried letting the water evaporate slowly toward the exterior of the inclusions. The water is then under tension, its pressure is strongly negative, that is to say that it pulls on the walls of the inclusions. At some pressure, a bubble forms very quickly, the pressure relaxes to a positive value. It is the cavitation phenomenon. The bubble formation is done in two different steps: i) the bubble size increases very quickly and compresses the fluid and the hydrogel. The Reynolds number of this step is relatively high ($R_e \sim 100$). The initially empty bubble is filled with vapor. ii) In the second step, the bubble is formed but its shape is out of equilibrium. It oscillates around its equilibrium shape. The oscillations come from the surface tension which exists between the vapor and the liquid water. The Reynolds number of this step is lower ($R_e \sim 5$), the fluid can be considered as incompressible. We were interested in reproducing this last phenomenon. The oscillation frequency of a bubble in a fluid is known theoretically since 1932 by Lamb [64]. It depends on the surface tension, the densities and the effective radius of the bubble. When Olivier Vincent compared his experimental data to the theoretical ones, he showed that an effective surface tension of about 30% lower than the one given for the water fitted well his data. Two main explanations were possible for that. The first one is that some chemical contamination could have happened during the polymerization of the hydrogel, which could have changed the surface tension of the solvent. The second one is that the confinement could have an impact on the oscillation frequency of the drops. It is this last point that we tried to figure out with our simulations. Indeed, the Lamb theory has been made with the assumption that the drop is in an infinite fluid. The goal of this study was to see if there exists a strong influence of the confinement on the oscillation frequency of the bubble. As we will see, our simulations have shown that the Lamb theory still holds even in a confined domain. Indeed, we have recovered the Lamb frequency with a good precision even with our strongest confinement. Consequently, as a first conclusion we can say that the influence of the confinement on the oscillation frequency of a bubble is relatively small. The second conclusion is that since we are able to recover the theoretical frequency of the oscillation, this test was a good validation for our model of two fluid flow. A similar test has already been done to validate a two fluid flow system with level set method [81].

2.4.2 Description of the simulation

Let us define a circular domain Ω, the fluid and the pressure are governed by the Navier-Stokes equations. No gravity force is added to the system. The only force added in the right hand side of the Navier Stokes equations is the surface tension force given by:

$$F_{\text{SurfTens}} = \int_{\Omega} \sigma \kappa \boldsymbol{n} \delta_{\varepsilon} \tag{2.54}$$

where σ is the surface tension between the two fluids. No-slip boundary conditions are added on the boundary of Ω. The bubble is initially at the center of the domain without any initial velocity. The initial shape given to the bubble is an ellipse. The non-dimensional characteristic numbers for this flow are the Reynolds number:

$$R_e = \frac{\rho U L}{\mu}$$

and the Weber number:

$$W_\mathrm{e} = \frac{\rho U^2 L}{\sigma} \tag{2.55}$$

which characterizes the relative importance of the inertia on the surface tension. Experimentally, the rigid inclusion size is of the order of 40 µm, the size of a bubble is about 10 µm. The densities of the liquid water and vapor at the experimental conditions are respectively $\rho_1 = 10^3$ kg/m^3 and $\rho_2 = 2 \times 10^{-2}$ kg/m^3, the viscosities are given by $\mu_1 = 10^{-3}$ Pa$\cdot$s and $\mu_2 = 2 \times 10^{-2}$ Pa$\cdot$s. The surface tension between the two water phases is $\sigma = 70 \times 10^{-3}$ N/m. For this experiment, the Reynolds and Weber numbers have been estimated as $R_e \sim 5$ and $W_\mathrm{e} \in [0.3, 2]$. The numerical parameters are simplified to ease the numerical simulation and to see a good number of oscillations in a short time. We choose $\rho_1 = 10$, $\rho_2 = 10^3$, $\mu_1 = \mu_2 = 1$. The surface tension is $\sigma = 12.5$. The area is taken as $A = 4.87$. The effective radius R_0 is defined as the radius of a circle having the same area as the bubble, $R_0 = \sqrt{\frac{A}{\pi}} = 0.775$. With these parameters, the Weber number is estimated to $W_\mathrm{e} = 1.4$ and the Reynolds number to $R_e = 100$. The Reynolds number is taken larger than the experimental one to be able to see many oscillations in a reasonable time. If it is taken too low, the oscillations are overly damped and the frequency is difficult to measure. The domain radius has been taken to the following values $R_\Omega = [2, 4, 6, 8]$, giving a confinement $C_\mathrm{n} = \frac{R_\Omega}{R_0}$ of respectively $\left[\frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}\right]$.

2.4.3 Results

The shapes that the bubbles take during their oscillations are represented in figure 2.18. The resolution of the experiments being done in our lab cannot give us the clear shape of the bubble. Nevertheless, Apfel et al. [2] did picture these shapes during a micro gravity experiment led in the Columbia space shuttle. In this experiment, the bubbles have been excited with an acoustic field, giving them a non equilibrium shape. After stopping the acoustic field, the bubbles did oscillate around their equilibrium shapes as in our simulations. In the same paper, some simulations by boundary integral method are compared to experiments. The results of these experiments and simulations are given in figure 2.19. The Reynolds number for these simulations is 600. One can see that we recover correctly the experimental shapes in our own simulations, even if the Reynolds number is 6 times lower. The only shape that we can not recover is the *8* shape at time 3.05 in figure 2.19 which needs a high Reynolds number.

The oscillation frequency of a bubble has been given by Lamb in 1932 [64] for a 3D bubble in an infinite fluid. Later, the oscillation frequency of an infinite cylinder in the z direction under surface tension has been given by Chandrasekhar [19]. The oscillation

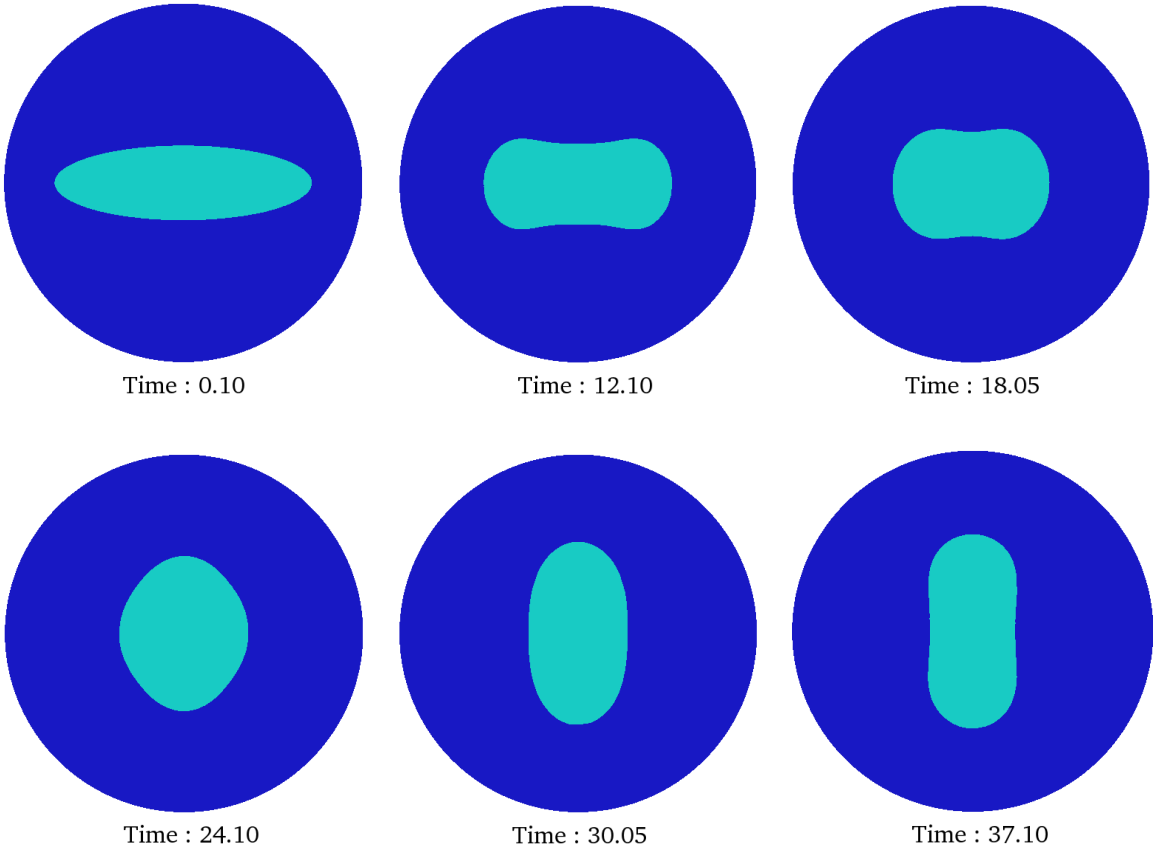


Figure 2.18: Our simulation of the shape oscillations of a bubble during time. The confinement is $\frac{1}{2}$ and we have taken $R_e = 50$, $W_e = 1$. Only half a period is represented since the following shapes are of the same type with different amplitudes.

frequency ω_n of a mode n is given by:

$$\omega_n^2 = n(n-1)(n+1)\frac{\sigma}{\rho_1 R^3} \tag{2.56}$$

where R is the radius of the bubble at equilibrium. The excited mode in our simulation is $n = 2$. The theoretical frequency that we obtain with the numerical parameters we used is $w_2 = 0.40$.
The simulations have been carried out on 6 processors. The mesh sizes have been taken to 0.135 away from the bubble and to 0.045 around and inside it. The time step is taken to 0.025. The reinitialization has been done by solving the Hamilton Jacobi equation presented in section 1.3.3 every 5 iterations. To measure the frequency of the oscillations, we choose to measure y_{length} the y long axis of the bubble. That is to say, the length of the bubble in the y direction at the center of the bubble. At equilibrium, this is equal to the diameter. Out of equilibrium, it oscillates around this value. We then do a fit of $y_{\text{length}}(t)$. For this fit, we use a model of damped oscillation for which we add a corrective term accounting for the loss of area due to the reinitialization of the level set. We modeled the loss of area by a linear loss of perimeter in time. At the end of the simulation, around 1% of the total mass has been lost, which is an acceptable result when taking into account how long last the simulation in time. The fits are done by the mean square method

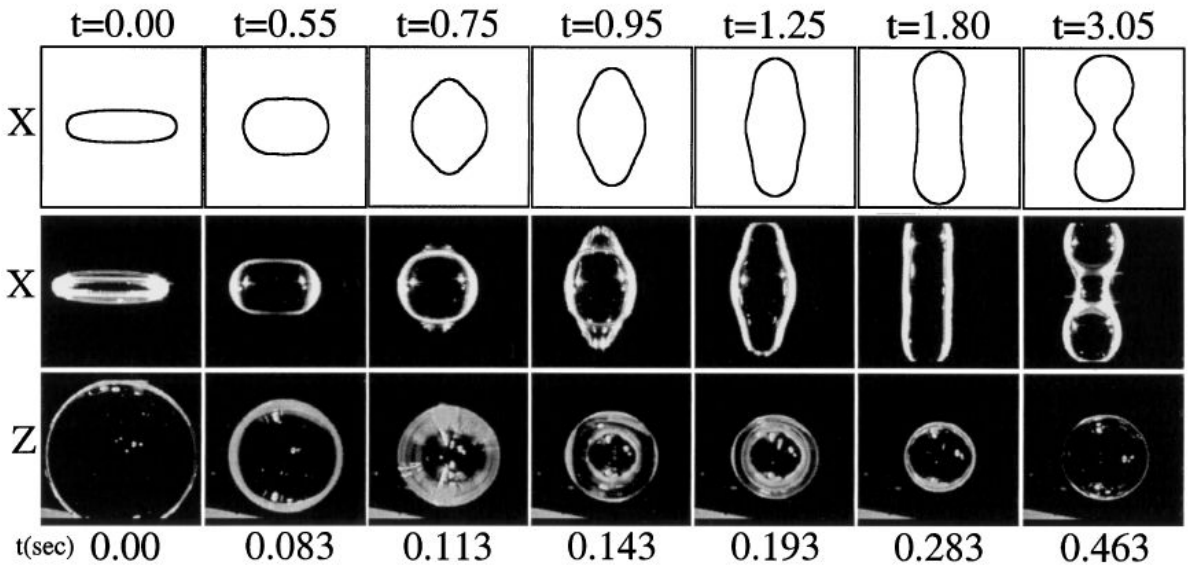


Figure 2.19: Figure taken from [2]. The first line represents the oscillation shape simulation of a bubble with an integral boundary method for a Reynolds number of 600. The two other lines are the views from two different axes of the oscillations of a bubble in the micro gravity experiment.

implemented in MINPACK [2] and interfaced by SCIPY [3] [51] that we used as a black box. The formula used has been given by:

$$y_{\text{length}} = a\sin(\omega t + \psi)e^{-\frac{t}{\tau}} - 2R_0 - bt \tag{2.57}$$

where a is the amplitude of the oscillations, ω the frequency we are seek, τ a characteristic time and b accounts for the loss of mass. The whole equation has subtracted the equilibrium value $2R_0$. The figure 2.20 represents the oscillations of y_{length} as a function of time. The curves with the parameters taken from the fit are represented as dashed lines. An offset is given to each curve so that we can plot them all on the same graph. Table 2.8 gives the values of the fit parameters for all the curves of figure 2.20. It turns out that the

C_{n}	a	ω	τ	b
1/2	0.86	0.40	54	0.0014
1/4	0.81	0.40	61	0.0013
1/6	0.79	0.40	62	0.0013
1/8	0.78	0.42	57	0.0018

Table 2.8: Parameters taken from the fit of the simulation curves with the model equation 2.57.

frequency is equal to the theoretical frequency up to the second decimal for the three first bubbles. The bubble with the lowest confinement has a slightly higher value, but it can be explained by the fact that the error for this simulation is also slightly higher than the other ones. Indeed, the value of the linear loss of area is much higher for this simulation

[2] http://www.netlib.org/minpack
[3] http://www.scipy.org

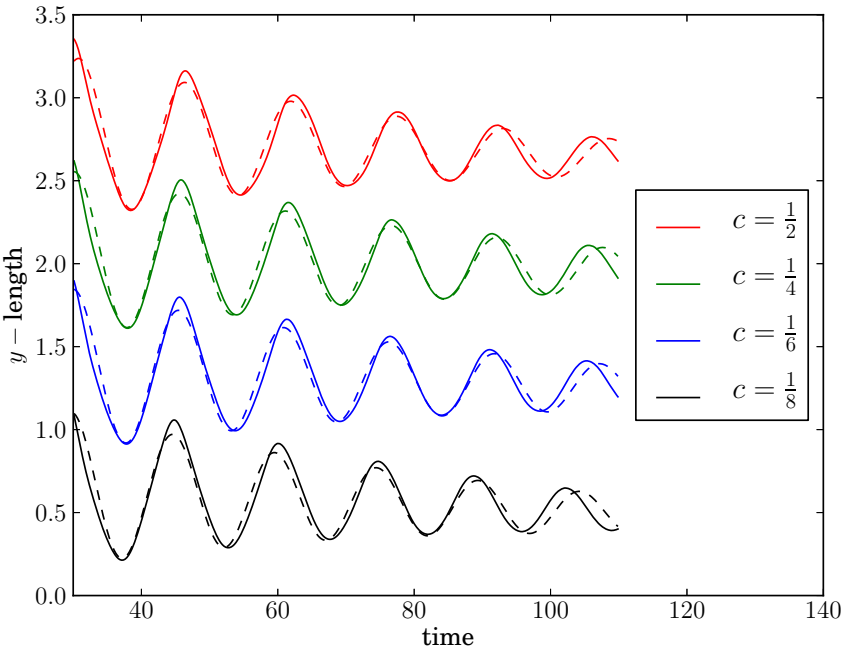


Figure 2.20: Results of the simulations of the oscillation of a drop around its equilibrium position for different confinements. The parameter being monitored, y_{length} is the length of the y axis of the bubble. An offset is added to each curve for the sake of visibility. The dashed lines are the curves of equation (2.57) with the parameters obtained from the fit of the simulation curves. The first period of the oscillations has been cut to see the actual oscillations without the initial excitation.

than the others. Anyway, in the experiments, a surface tension about 30% higher than the theoretical one is observed. If we compute how important should be the frequency gap with the theoretical one in our simulation to get the same percentage difference, we found that the frequency difference $\Delta\omega = \omega - \omega_{\text{th}}$ should be of the order of 0.1. Thus we can say that we do not see a significant effect of the confinement on the oscillation frequency and that the Lamb formula still holds at such confinements. Moreover, the theoretical frequency is obtained with a good accuracy and this simulation was a good validation of our two-fluid flow framework.

Chapter 3

Simulation of solid objects in flow

Contents

In this chapter we will address the problem of the simulation of solid objects in flow. Indeed, in general the understanding of the influence of the deformability of vesicles on a particular problem starts by the precise understanding of the same phenomenon with rigid particles and a comparison between both behaviors. Being able to simulate a rigid object in the same framework as the vesicle framework is then a significant advantage. We will show in this chapter that our framework is able to deal with solid objects and that it is actually close to two common methods used in the literature to address this kind of problem. Firstly we will give a brief review of some methods used for the simulation of solid objects in flow. Then we will show that the fluid particle dynamics and the penalty methods are two equivalent methods. Then we will show that at the fluid point of view, setting a huge viscosity inside the objects tracked by level set methods is equivalent to a penalty method in which the particles are followed with an Eulerian point of view. We will finally discuss the advantages and drawbacks of such a point of view.

3.1 Existing methods

The problem is to simulate the motion of several rigid bodies in a Newtonian fluid. Let us call $B = \bigcup_{i=1}^{N_{\text{part}}} B_i$ where B_i is the domain representing the particle i, and Ω the whole domain. Thus, the fluid domain is given by $\Omega_f = \Omega \setminus B$. We denote by $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$ the velocities of the center of mass of the particle i and $\boldsymbol{\omega}_i$ its angular velocity. The mass of the particle is called m_i and its moment of inertia I_i. The fluid is governed by the incompressible Navier-Stokes equations. There are no slip conditions between the particles and the fluid, thus the velocity of the fluid is equal to the velocity of the particle at the particle boundaries ∂B_i. The spatial and angular accelerations of the particles are the result of the total forces applied on it divided by the mass and angular momentum as expressed by the fundamental principle of dynamics. These conditions can be expressed as follows:

$$\rho \frac{D\boldsymbol{u}}{Dt} - \mu \boldsymbol{\Delta u} + \boldsymbol{\nabla} p = \boldsymbol{f} \qquad \text{in } \Omega_f, \tag{3.1}$$

$$\nabla \cdot \boldsymbol{u} = 0 \qquad \text{in } \Omega_f, \tag{3.2}$$

$$m_i \frac{d\boldsymbol{v}_i}{dt} = \boldsymbol{F}_i \qquad \forall i, \tag{3.3}$$

$$I_i \frac{d\boldsymbol{\omega}_i}{dt} = \boldsymbol{T}_i \qquad \forall i, \tag{3.4}$$

$$\boldsymbol{u} = \boldsymbol{v}_i + \boldsymbol{\omega}_i (\boldsymbol{x} - \boldsymbol{x}_i)^{\perp} \qquad \text{on } \partial B. \tag{3.5}$$

where the total forces $\boldsymbol{F}_i$ and torques $\boldsymbol{T}_i$ on the particles are the sum of the hydrodynamic forces and external additional forces:

$$\boldsymbol{F}_i = \boldsymbol{F}_{\text{ext}} + \int_{\partial B_i} \boldsymbol{\sigma n} \qquad \text{on } \partial B_i,\ \forall i \tag{3.6}$$

$$\boldsymbol{T}_i = \boldsymbol{T}_{\text{ext}} + \int_{\partial B_i} (\boldsymbol{x} - \boldsymbol{x}_i) \times \boldsymbol{\sigma n} \qquad \text{on } \partial B_i,\ \forall i. \tag{3.7}$$

We recall that $\boldsymbol{\sigma} = \mu(\boldsymbol{\nabla u} + {}^t\boldsymbol{\nabla u}) - p\boldsymbol{I}_{\text{d}}$. There exist several ways to solve these equations. Three good bibliographic works reviewing the different methods have been done in the

PhD theses [46, 66] and [32] (all in french). We will name briefly the main methods before comparing the penalty and fluid particle dynamic methods.

The conform methods

The simplest method to solve this problem is to create a conforming mesh to the domain Ω_f, in which the domains B_i are represented as *holes* in Ω_f. The velocities $\boldsymbol{v}_i$ and positions $\boldsymbol{x}_i$ are defined independently of the mesh. The equations (3.1) and (3.2) with the condition (3.5) are solved on the mesh by a PDE solver. Then the velocities and positions are updated solving the equations (3.3) and (3.4). The domain is finally re-meshed to conform to the new positions $\boldsymbol{x}_i$ and the loop is closed. A simple implementation of such a method is made with the software FreeFem++ in appendix D. This method suffers from several issues. Indeed, the re-meshing procedure can be computationally costly. Moreover, if one actually solves the Navier-Stokes equations, the solution to the previous iteration is needed at the current iteration. The problem is that the previous solution belongs to a different mesh and a projection procedure has to be employed which induces some error. An alternative to the systematic re-meshing is to move slightly the points of the mesh around the particles to their new positions. These methods are called **Arbitrary Lagrangian Eulerian** methods. This way, since the displacement of the mesh is not too high, the re-meshing procedure can be avoided. Such a method is used for example in [50] and an implementation of an ALE method for Feel++ has been done by Vincent Chabannes during his PhD [17].

Fictitious domain methods

The **fictitious domain** methods are a set of methods in which the fluid is defined on all the space Ω which is a domain larger than Ω_f but simpler. A rigid body constraint is applied to the fluid on the space which belongs to the particles. These methods have the advantage of not having to handle a changing mesh as done with the direct methods. There are two main ways to impose the no slip boundary condition (3.5). In a finite element context, one can impose the rigid body motion of the particles by adding a Lagrange multiplier to the equations as done in [40, 39]. The Lagrange multiplier is a vector which can be seen as the volume force needed to impose the rigid body constraint. The advantage of this method is to impose exactly the constraint and it does not change the conditioning of the matrix to invert. The drawback is that it can be difficult to implement. The other method, that we will develop in this section, consists of adding a penalty term to force the rigid body motion; as a shortcut name, we will call these methods **penalty** methods. This method has been introduced for the problem of rigid particles in a fluid in [49] and a precise analysis has been made in [73]. The method has also been studied during the PhD thesis of A.Lefebvre-Lepot [66]. It has the advantage of being simple to implement. However, the drawback is that the penalty term generally deteriorates the conditioning of the matrix, making the problem more difficult to solve numerically. Moreover, both methods induce a discontinuity in the normal derivative of the solution at the boundary of the particles. Some methods called **control** methods try to get a better regularity by changing the value of the forces inside the particle. Indeed, the value of the right hand side of the Stokes equation is usually extended by zero in the particle, but since this value has no physical meaning or natural value, it can be extended

to a value making the solution of the problem smoother. Such a method has been the subject of study of the PhD of B. Fabrèges [32].

The fat boundary method

The **fat boundary method (FBM)** has been introduced in [74] and study during the PhD thesis of M.Ismail [46]. It consists of writing the problem on two different meshes. A first mesh, called global mesh being defined on all the domain, having a relatively large discretization step. This mesh can possibly be structured in order to use fast solvers on it. A second mesh is defined around the particles, it is called the local mesh. It is thinner, and can move with the particles if they are motile. The problems written on the global and local meshes are coupled. The solution is obtained by an iterative procedure which, at convergence, imposes that the solution on both meshes match. A condition is added to impose the continuity of the normal derivative. Thus, the spatial error obtained with this method has a better rate of convergence than the previous one. An implementation of the FBM for Feel++ has been made in [17] and a mathematical analysis has been done in [6].

The immersed boundary method

The **immersed boundary method** has been introduced by C.Peskin [82]. A general description of this method is given in [83]. It has been initially written for a finite difference scheme but it can be extended to finite element. The idea is to write the structure boundary (here the particle boundary) as a collection of Lagrangian points. The velocity and pressure of the fluid are discretized on a fixed grid. The forces acting on the interface are calculated off the grid, since they usually depend only on the positions of the points of the interface (the normal, its curvature, or higher derivatives of the curvature). The forces obtained are defined only on the boundary, which do not belong to the grid points, thus it is projected on the grid thanks to a regularized delta-like function (with a form equivalent to the one used in a level set context). The thickness of the delta function is taken large enough so that few grid points are always intersected. Then the fluid equations are solved on the grid with the forces due to the boundary as right hand side. The solution is then interpolated to the Lagrangian points of the boundary which are advected at the fluid velocity. This method has been written for arbitrary interface force, but it has been extended to the rigid body motion in [113].

The fluid particle dynamics method

An other method comes from the *physics community* whereas all the methods presented above arise from the *applied mathematics* one. It has been introduced by H. Tanaka [105] for the simulation of colloids. It has been also used in [22, 84] in the context of rheology of rigid spheres and in the PhD of L. Jibuti [48] and [88] for the rheology of micro-swimmers. The idea is once again to have a Lagrangian approach for the particles. Then, the coupling with the fluid is made by fixing the fluid viscosity to a huge value where the particles are. The huge viscosity induces a rigid body motion of the fluid in this region. In [105], the flow field was solved by inverse Fourier transform. In [22, 84, 48] the method was written in a finite difference context, we will see that its extension to a

finite element context makes this method very similar to the fictitious domain where the rigid body constraint is imposed by a penalty term.

3.2 Penalty method vs Fluid Particle Dynamics method

3.2.1 A similar formulation for FPD and penalty method

We propose to show that the finite element version of fluid particle dynamics [105] is exactly equivalent to the penalty fictitious domain method [49]. Let us describe the methods following both points of view and show that at the end, the same equations are solved. We start by the penalty method, for the sake of simplicity. To make the problem unique, we impose the Dirichlet boundary conditions (3.8), but the problem can be extended to any type of boundary condition.

$$\boldsymbol{u} = 0 \text{ on } \partial\Omega. \tag{3.8}$$

Let us introduce the space:

$$K_B = \left\{ \boldsymbol{u} \in [H_0^1(\Omega)]^d, \boldsymbol{D}(\boldsymbol{u}) = 0 \text{ in } B \right\}. \tag{3.9}$$

As we have already shown in section 2.1.1, the problem (3.1) (3.2) (3.3) (3.4) (3.5) (3.8) can be written in a variational form as, for a given $\boldsymbol{f} \in [L^2(\Omega)]^d$, find $(\boldsymbol{u}, p) \in K_B \times L^2(\Omega)$ so that $\forall (\boldsymbol{v}, q) \in K_B \times L^2(\Omega)$:

$$\begin{array}{rcl} \displaystyle\int_\Omega \rho \frac{D\boldsymbol{u}}{Dt} \cdot \boldsymbol{v} + \int_\Omega 2\,\mu\, \boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{D}(\boldsymbol{v}) - \int_\Omega p \nabla \cdot \boldsymbol{v} & = & \displaystyle\int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \\ \displaystyle\int_\Omega q \nabla \cdot \boldsymbol{u} & = & 0, \end{array} \tag{3.10}$$

where the problem has been written on all the domain Ω instead of Ω_f by imposing the constraint that the solution belongs to K_B. This is the principle of the fictitious domain method. Now let us relax the constraint on the space by adding a penalty term, the formulation reads, for a given $\boldsymbol{f} \in [L^2(\Omega)]^d$ find $(\boldsymbol{u}, p) \in [H_0^1(\Omega)]^d \times L^2(\Omega)$ such that $\forall (\boldsymbol{v}, q) \in [H_0^1(\Omega)]^d \times L^2(\Omega)$:

$$\begin{array}{rcl} \displaystyle\int_\Omega \rho \frac{D\boldsymbol{u}}{Dt} \cdot \boldsymbol{v} + \int_\Omega 2\,\mu\, \boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{D}(\boldsymbol{v}) + \int_B \frac{2}{\varepsilon} \boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{D}(\boldsymbol{v}) - \int_\Omega p \nabla \cdot \boldsymbol{v} & = & \displaystyle\int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \\ \displaystyle\int_\Omega q \nabla \cdot \boldsymbol{u} & = & 0, \end{array} \tag{3.11}$$

with $\varepsilon << 1$ to impose correctly the constraint. We can notice at this point that $\dfrac{1}{\varepsilon}$ has the physical dimension of a viscosity $\left[\dfrac{1}{\varepsilon}\right] = [\mu]$. The space discretization of this equation is made as done previously, by introducing $(\boldsymbol{u}_h, p_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$ the discrete versions of $(\boldsymbol{u}, p)$ and their test functions $(\boldsymbol{v}_h, q_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$. A special treatment has to be done for the penalty integral. Indeed, it is defined on the space B which depends on the positions of the particles $B = B(\boldsymbol{x}_i(t))$. The simplest way to treat this integral is to define a characteristic function $\chi(\boldsymbol{x}_i)$:

$$\chi(\boldsymbol{x}) = \begin{cases} 1 & \text{if} \quad \boldsymbol{x} \in B, \\ 0 & \text{else.} \end{cases} \tag{3.12}$$

This characteristic function can be defined analytically for simple shapes like a sphere of radius r: $\chi(\boldsymbol{x}) = |\boldsymbol{x} - \boldsymbol{x}_i| < r$ or an ellipsoid of parameters (a, b, c): $\chi(x, y, z) = \frac{(x-x_i)^2}{a^2} + \frac{(y-y_i)^2}{b^2} + \frac{(z-z_i)^2}{c^2} < 1$, but it would be more difficult to handle a problem in which the rigid bodies have complicated shapes. Then one replaces the integral over B by an integral over Ω thanks to the characteristic function. The problem reads, find $(\boldsymbol{u}_h, p_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$ so that $\forall (\boldsymbol{v}_h, q_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$:

$$\begin{aligned} \int_\Omega \rho \frac{D\boldsymbol{u}_h}{Dt} \cdot \boldsymbol{v}_h + \int_\Omega 2\,\mu\, \boldsymbol{D}(\boldsymbol{u}_h) : \boldsymbol{D}(\boldsymbol{v}_h) + \int_\Omega \frac{2\,\chi(\boldsymbol{x}_i)}{\varepsilon}\, \boldsymbol{D}(\boldsymbol{u}_h) : \boldsymbol{D}(\boldsymbol{v}_h) - \int_\Omega p_h \nabla \cdot \boldsymbol{v}_h &= \int_\Omega \boldsymbol{f}_h \cdot \boldsymbol{v}_h \\ \int_\Omega q_h \nabla \cdot \boldsymbol{u}_h &= 0. \end{aligned} \tag{3.13}$$

Finally, the time discretization is performed if one considers a non zero Reynolds number. Our goal is now to show that a term similar to the third integral of equation 3.13 will be applied in the FPD method.
In [105] the authors propose to impose the rigid body constraint on the particle by solving the Navier-Stokes equations with a space dependent viscosity having a huge value inside the particle. Such a problem reads:

$$\begin{aligned} \rho \frac{D\boldsymbol{u}}{Dt} - \tilde{\mu}(\boldsymbol{x}_i)\boldsymbol{\Delta u} - \boldsymbol{\nabla} p &= \boldsymbol{f} \\ \nabla \cdot \boldsymbol{u} &= 0 \end{aligned} \tag{3.14}$$

with, $\mu_\mathrm{p} >> \mu$ if one call μ the viscosity of the fluid and μ_p the viscosity inside the particles. We also call $\Delta\mu = \mu_\mathrm{p} - \mu$ the viscosity difference. With these notations we define the viscosity $\tilde{\mu}$ as:

$$\tilde{\mu}(\boldsymbol{x}_i) = \mu + \Xi(\boldsymbol{x}_i)\,\Delta\mu, \tag{3.15}$$

$$\Xi(\boldsymbol{x}_i) = \sum_{i=1}^{N_\mathrm{part}} \frac{1}{2} \left\{ \tanh\left(\frac{r - |\boldsymbol{x} - \boldsymbol{x}_i|}{\zeta} \right) + 1 \right\} \tag{3.16}$$

where r is the radius of the rigid particle and ζ an interface thickness. The function Ξ imposes the shape of the particles to be spherical, and the viscosity varies continuously (but sharply) from μ to μ_p. If the problem is solved in a finite element context, the equations are put on a variational form and spatially discretized on finite element bases. The problem reads, find $(\boldsymbol{u}_h, p_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$ so that $\forall (\boldsymbol{v}_h, q_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n$:

$$\begin{aligned} \int_\Omega \rho \frac{D\boldsymbol{u}_h}{Dt} \cdot \boldsymbol{v}_h + \int_\Omega 2\,\tilde{\mu}(\boldsymbol{x}_i)\, \boldsymbol{D}(\boldsymbol{u}_h) : \boldsymbol{D}(\boldsymbol{v}_h) - \int_\Omega p_h \nabla \cdot \boldsymbol{v}_h &= \int_\Omega \boldsymbol{f}_h \cdot \boldsymbol{v}_h \\ \int_\Omega q_h \nabla \cdot \boldsymbol{u}_h &= 0, \end{aligned} \tag{3.17}$$

which reads if one replace $\tilde{\mu}$ by its expression (3.15):

$$\begin{aligned} \int_\Omega \rho \frac{D\boldsymbol{u}_h}{Dt} \cdot \boldsymbol{v}_h + \int_\Omega 2\,\mu\, \boldsymbol{D}(\boldsymbol{u}_h) : \boldsymbol{D}(\boldsymbol{v}_h) + \int_\Omega 2\,\Xi(\boldsymbol{x}_i)\,\Delta\mu\, \boldsymbol{D}(\boldsymbol{u}_h) : \boldsymbol{D}(\boldsymbol{v}_h) - \int_\Omega p_h \nabla \cdot \boldsymbol{v}_h &= \int_\Omega \boldsymbol{f}_h \cdot \boldsymbol{v}_h \\ \int_\Omega q_h \nabla \cdot \boldsymbol{u}_h &= 0. \end{aligned} \tag{3.18}$$

We see clearly with this expression that the two formulations are actually similar. They both introduce an additional term to the classical formulation defined as the integral over all the domain of $2\,\boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{D}(\boldsymbol{v})$ times a factor being the product of a term equivalent to a viscosity and having a huge value, and a term representing the shape and distribution of the particles. The only difference lies in the fact that one of the characteristic function is sharp and the other is continuous. However, $\Xi \to \chi$ when $\zeta \to 0$, . Thus we have shown that both methods are actually completely similar in a finite element context.

3.2.2 Particle motion in FPD/penalty methods

We have shown that FPD solved by finite element and penalty methods are the same methods for the fluid equations. The particle motion is actually also handled the same way in both methods. The particles are seen in a Lagrangian point of view. That is to say that one keeps the information of the position and velocity of each particle. We will see how the coupling is done between the fluid velocity and the particle velocities and positions.

First of all, one has to define the initial positions and velocities of the rigid inclusions. Then one solves the fluid equations with the rigid body constraint and possibly external forces depending on the positions of the particles (electrostatic forces for example). Then, from the solution $\boldsymbol{u}$, one needs to compute the velocity of each particle $\boldsymbol{v}_i$. To this end, the velocity $\boldsymbol{v}_i$ is simply taken as the mean velocity of $\boldsymbol{u}$ in B:

$$\boldsymbol{v}_i = \frac{1}{\text{volume}(B_i)} \int_{B_i} \boldsymbol{u}. \tag{3.19}$$

Finally, the velocities are integrated to get the new positions. A classical BDF integration scheme can be employed. For the example, let us take the simple Euler scheme:

$$\boldsymbol{x}_i^{n+1} = \boldsymbol{x}_i^n + \boldsymbol{v}_i^n \delta t. \tag{3.20}$$

An algorithm for FPD or penalty method is presented in algorithm 4.

Algorithm 4 FPD method or fictitious domain with penalty term method for the simulation of rigid inclusions in a fluid

Require: $\boldsymbol{x}_i^0, \boldsymbol{v}_i^0$
  for $n = 0$ to N_{t_f} **do**
    $\chi^n =$ **ComputeCharacteristicFunction**$(\boldsymbol{x}_i^n)$ eq (3.12) or (3.16)
    $(\boldsymbol{u}^n, p^n) =$ **SolveFluidProblem**$(\chi^n, \boldsymbol{x}_i^n)$ eq (3.13) or (3.18)
    $\boldsymbol{v}_i^n =$ **MeanVelocities**$(\boldsymbol{u}^n, \chi^n)$ eq (3.19)
    $\boldsymbol{x}_i^{n+1} =$ **UpdatePositions**$(\boldsymbol{x}_i^n, \boldsymbol{v}_i^n)$ eq (3.20)
  end for

This Lagrangian point of view has the advantage of being simple to implement. Moreover, in a context of a physical problem in which one needs the individual trajectories of the particles (as the bifurcation problem presented in section 7.1), the Lagrangian point of view is also the best. We will show that it is not the only option available and that in some cases, seeing the particles in an Eulerian point of view can also have some advantages.

3.3 FPD in Level set framework

In [27] (which we report in section 7.1), we used a penalty (FPD) method taken from [49], implemented on FreeFem++ . Another implementation of the same method has been made with Feel++ . The advantage was to be able to treat the problem in 2D and 3D with the same code. The implementation can be found in [87] and 2D and 3D simulations are shown respectively on figures 3.1 and 3.2. We can show easily that the

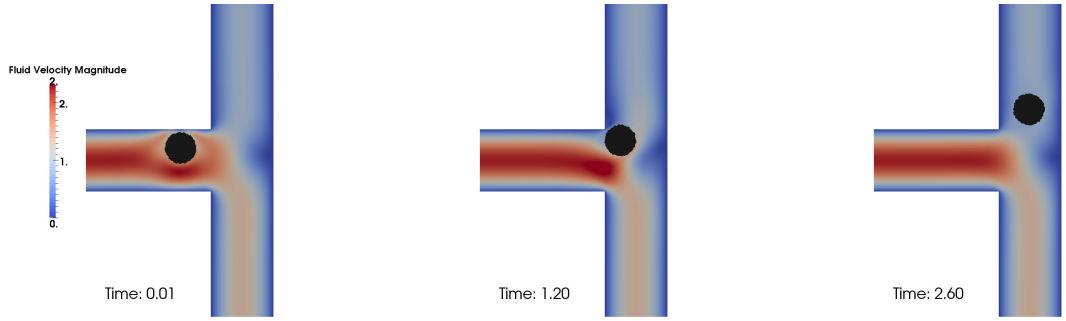


Figure 3.1: 2D simulation of the bifurcation problem presented in section 7.1. The color is the magnitude of the velocity vector $|\boldsymbol{u}|$. The area in which $\mu = \mu_2$ is represented in black. The particle enters in the low flow rate branch of the bifurcation.

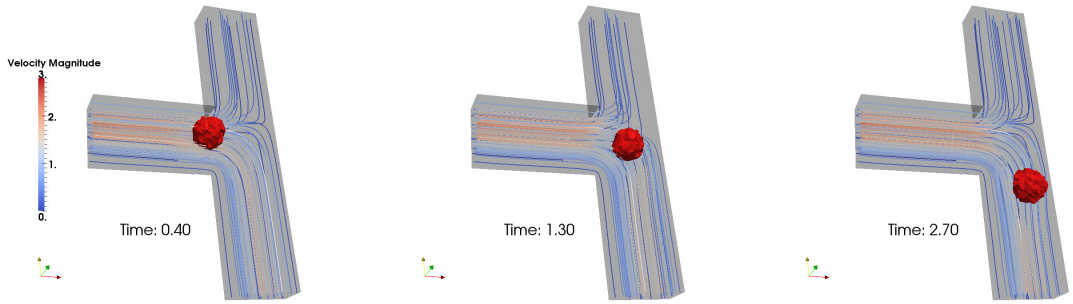


Figure 3.2: 3D simulation of the bifurcation problem. Some streamlines are represented on the figure and colored with the velocity magnitude $|\boldsymbol{u}|$. The area in which $\mu = \mu_2$ is represented in red. This simulation has been run on a single processor which explains the rough grain of the particle.

level set framework presented previously can be used as an implementation of an FPD method. Indeed, one only needs to set the internal viscosity of the fluid to a huge value compared to the fluid one, and the fluid formulation would be equivalent to (3.13) or (3.18). Such an algorithm is shown in algorithm 5. It is very simple and uses only tools already developed in the previous parts. Thus, the code need not to be changed and only the initial configuration and viscosities which can be monitored as a user, need to be modified to add rigid inclusions in the fluid. Of course, the level set field would play the role of the characteristic function making the integration on the domain B possible thanks to the Heaviside function.

The only difference from the FPD and penalty method is that in a level set context one does not follow the position of each rigid inclusion independently. Indeed, one would advect one level set field and lose the information of the position of each rigid inclusion. Moreover, this advection step takes generally more computational time than to do the integration of the positions of the particles (3.20). However, in a simulation where only one rigid inclusion is present (bifurcation in [27]) the position of the particle would not be lost by using a level set function instead of integrating the position $\boldsymbol{x}$ of the particle. On the other hand, for a very large number of rigid particles, one generally does not care about the exact position of each particle, and only mean values are monitored (viscosity, density of particles in a region) which are accessible by the level set field. In these cases, only the computational time could be a drawback for the use of level set instead of individual coordinates for the particles.
Nevertheless, two main advantages can be found for the use of level set field. The first one, is that the shape of the rigid particles has to be given once and for all at the beginning of the simulation. Then, there is no need to re-use any shape function and the initial shape will be advected correctly by the velocity of the fluid. The shape can be given by analytic distance functions (or close to it), as for spheres, lines or ellipses. It can also be given analytically by combination of min and max of these functions, as it has been done for the slotted disk in section 1.4. Or, for more complicated shapes, it can be given by the parametrization of a curve as explained in appendix C. Instead, for the FPD method, the shape function has to be easy to express since it is used at any time step to update the characteristic function.
Moreover, we only considered in this section spherical or disk shapes for which the inclination angle of the object has no influence. But to use ellipsoidal particles in an FPD context for example, one would have to add new sets of variables accounting for the angles of the objects $(\alpha_i, \beta_i, \gamma_i)$ (in 3d) and the rotational velocities $\boldsymbol{w}_i$ as it is done for example in [48]. At each time step, the positions and angles have to be updated which can start to be heavy in a 3D simulation where 3 angles are needed to describe the rotation. This is completely transparent in a level set context and only the initial shape has to be changed to go from spherical particles to other asymmetric particles.
The second advantage of the use of level set instead of classical explicit following of the particles resides in the fact that it can be easily coupled with deformable objects. Indeed, by using the multi level sets framework introduced in section 2.2.2, one can set a viscosity to a huge value for rigid inclusion and softer values to model vesicles, bubbles or other objects. In conclusion, we can say that using level set for the simulation of rigid inclusions might not be optimal for a simple simulation but it brings more flexibility like the possibility to handle easily the rotation and complex forms or to couple it with soft objects.

Algorithm 5 FPD method or fictitious domain with penalty term method using a level set field.

Require: $\phi_0, \mu_2 \gg \mu_1$
  for $n = 0$ to N_{t_f} **do**
    $(\boldsymbol{u}^n, p^n) =$ **SolveFluidProblem**$(H_\varepsilon(\phi^n))$ eq (2.31) and (2.32)
    $\phi^{n+1} =$ **SolveAdvection**$(\boldsymbol{u}^n)$ eq (2.33)
  end for

Chapter 4

Simulation of vesicles : comparison of two models

Contents

In this chapter we will describe our strategy for the simulation of vesicles in flow. First we will recall the expression of the bending force and its level set expression. This force requires the knowledge of the 4^{th} derivative of the level set field. Two strategies will be described to get this high order derivative, the increase of the polynomial approximation order of the level set field and the smoothing of the fields during each differentiation. Then we will present our implementation of two different methods used in the literature to impose the inextensibility of the vesicle membrane. The first one uses the information of the stretching of the interface recorded in the level set field to derive a force opposed to it (this is a penalty method). The other one uses a Lagrange multiplier to impose that the surface divergence is zero on the membrane. We finally present some validation tests on the behavior of a vesicle in a simple shear flow including its equilibrium shape, and show that the Lagrange multiplier method gives a better accuracy and is more stable than the penalty method.

4.1 Bending force in level set context

The energy associated to the bending of a vesicle membrane has been addressed by Canham and Helfrich, respectively, in [16, 43]. They have shown that this energy varies as:

$$E_{\text{b}} = \frac{k_{\text{B}}}{2}\int_{\Gamma}\kappa^2 \tag{4.1}$$

where k_{B} is the bending modulus of the vesicle (a typical value is 10^{-19} J), and κ is the curvature of the membrane. In the context of phase field simulation, the authors of [8] used the principle of virtual works to derive this energy and get a force. The same procedure has been done in the level set context in [71]. We used the force formulation given in [71]. A comparison between level set and phase field methods has been made in [70] where the authors have shown that equivalent results can be obtained with both methods. The curvature force given in [71] can be written as:

$$\boldsymbol{F}_{\text{b}} = \int_{\Omega} k_{\text{B}}\nabla\cdot\left(\frac{-\kappa^2}{2}\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}+\frac{1}{|\boldsymbol{\nabla}\phi|}\left(\boldsymbol{I}_{\text{d}}-\frac{\boldsymbol{\nabla}\phi\otimes\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|^2}\right)\boldsymbol{\nabla}\{|\boldsymbol{\nabla}\phi|\,\kappa\}\right)\delta_{\varepsilon}\boldsymbol{\nabla}\phi \tag{4.2}$$

where $\left(\boldsymbol{I}_{\text{d}}-\dfrac{\boldsymbol{\nabla}\phi\otimes\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|^2}\right)$ is the orthogonal projector operator, often called $\mathbb{P}_{\perp}$ and can be simplified in 2D by the expression $\mathbb{P}_{\perp}^{2D}(\boldsymbol{v}) = \boldsymbol{t}(\boldsymbol{t}\cdot\boldsymbol{v})$, with $\boldsymbol{t} = \boldsymbol{n}^{\perp}$. An index expression of the orthogonal projector operator is given in appendix G. One can see that the expression (4.2) leads to some numerical difficulties. Indeed, it consists of the divergence of a function of the gradient of the curvature, the curvature being itself the divergence of the gradient of the level set field ($\kappa = \nabla\cdot\boldsymbol{n} = \nabla\cdot\dfrac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}$). Thus the curvature force requires taking four times the derivative of the field ϕ. This problem is not specific to the level set or phase field methods as all the methods involving a force derived from the bending energy of the form (4.1) need the fourth-order derivative of the shape function of the membrane with a good accuracy. To overcome this issue different techniques have been used. In the context of spectral boundary integral methods [108, 33, 116, 117, 109] the shape is usually represented as a function of the spherical coordinates and time $\boldsymbol{r}(\theta_1,\theta_2,t)$ (in 3D) and is discretized using finite spherical harmonics. The fourth order derivative of the shape can

be obtained with a good accuracy by using high order spherical harmonics. In a finite element context, Laadhari during his PhD [62] used L^2 projection with quadrature order of 1 (mass lumping).

In this section, we will present two different strategies that we developed to be able to obtain the derivatives of the curvature. The first one is done by increasing the polynomial order of the discretized level set field. This method gives the better accuracy. It can be difficult to derive all the coefficients we talked about in this work for high order approximations. Indeed, all the parameters depending on the mesh size should, for a high order simulation, also depend on the polynomial approximation order used. These coefficients are: the interface thickness and the stabilization coefficients for the element stabilization (the CIP method already includes high order finite element). Moreover, to keep a reasonable computational time, one would need to decrease the mesh size, thus, the polynomial approximation for the fluid and pressure should also be increased in consequence to keep the overall approximation accurate. This is already included in our model and should work satisfactorily. Increasing the polynomial order of approximation for the vesicle application is thus possible and a few tests have been made in this way. However, we wanted also to derive a method working at low order (order 1). The goal was to be able to keep the classical level set parameters and simplify the method. Consequently, we also derived a projection method in which we smooth the projected field by adding a small Laplacian term to the left hand side of a projection. This method has a much worse accuracy but is enough to get an approximation of the curvature and its derivatives.

4.1.1 High order derivative by increasing polynomial approximation order

Let us define a function $u \in H^1(\Omega)$ for which we want to find the gradient $\boldsymbol{\nabla} u$. One way to get this gradient is to do a L^2 projection of $\boldsymbol{\nabla} u$. Such a projection reads, for a given $u \in H^1(\Omega)$, find $\boldsymbol{g} \in [L^2(\Omega)]^d$ such that $\forall \boldsymbol{v} \in [L^2(\Omega)]^d$:

$$\int_\Omega \boldsymbol{g} \cdot \boldsymbol{v} = \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{v}. \tag{4.3}$$

One discretizes this equation by introducing the continuous finite element spaces $\mathbb{U}_h^k$ and $\mathbb{G}_h^l$, respectively scalar and vectorial finite element spaces based on a mesh of typical size h and Lagrange polynomials of order k and l. Let us introduce the discrete version of u and $\boldsymbol{g}$, $u_h \in \mathbb{U}_h^k$ and $\boldsymbol{g}_h \in \mathbb{G}_h^l$. The discretized equation (4.3) reads then, for a given $u_h \in \mathbb{U}_h^k$, find $\boldsymbol{g}_h \in \mathbb{G}_h^l$ such that $\forall \boldsymbol{v}_h \in \mathbb{G}_h^l$:

$$\int_\Omega \boldsymbol{g}_h \cdot \boldsymbol{v}_h = \int_\Omega \boldsymbol{\nabla} u_h \cdot \boldsymbol{v}_h. \tag{4.4}$$

Let us write this projection in a more general form. We introduce the vectorial finite element space $\mathbb{A}_h^m \subset L^2(\Omega)$ and $\mathbb{B}_h^n \subset L^2(\Omega) \cap C^0$ based on a discretization mesh of size h and Lagrange polynomials of order respectively m and n. Let us introduce $\boldsymbol{a}_h \in \mathbb{A}_h^m$ and $\boldsymbol{b}_h \in \mathbb{B}_h^n$ elements of those spaces. We define the L^2 projection operator on space $\mathbb{B}_h^n$ as follows:

$$\begin{aligned} \pi_{L^2}^{\mathbb{B}_h^n} : \quad & \mathbb{A}_h^m \to \mathbb{B}_h^n \\ & \boldsymbol{a}_h \overset{(4.6)}{\mapsto} \boldsymbol{b}_h \end{aligned} \tag{4.5}$$

$$\begin{cases} \text{find } \boldsymbol{b}_h \in \mathbb{B}_h^n \text{ so that } \forall \boldsymbol{v}_h \in \mathbb{B}_h^n: \\ \displaystyle\int_\Omega \boldsymbol{b}_h \cdot \boldsymbol{v}_h = \int_\Omega \boldsymbol{a}_h \cdot \boldsymbol{v}_h \end{cases} \tag{4.6}$$

Thus, the projection (4.4) can be rewritten as:

$$\boldsymbol{g} = \pi_{L^2}^{\mathbb{G}_h^l}(\boldsymbol{\nabla} u_h) \tag{4.7}$$

Since u_h is discretized on polynomials of order k, its gradient $\boldsymbol{\nabla} u_h$ has only the information of polynomials of order $k-1$. If one wants to derive a quantity n times by this method (let us say one needs $\boldsymbol{\nabla}^n u$), this limitation requires taking u_h in a finite element space of order $k >= n$ to have at least $\boldsymbol{\nabla}^n u$ as polynomial of degree 0.

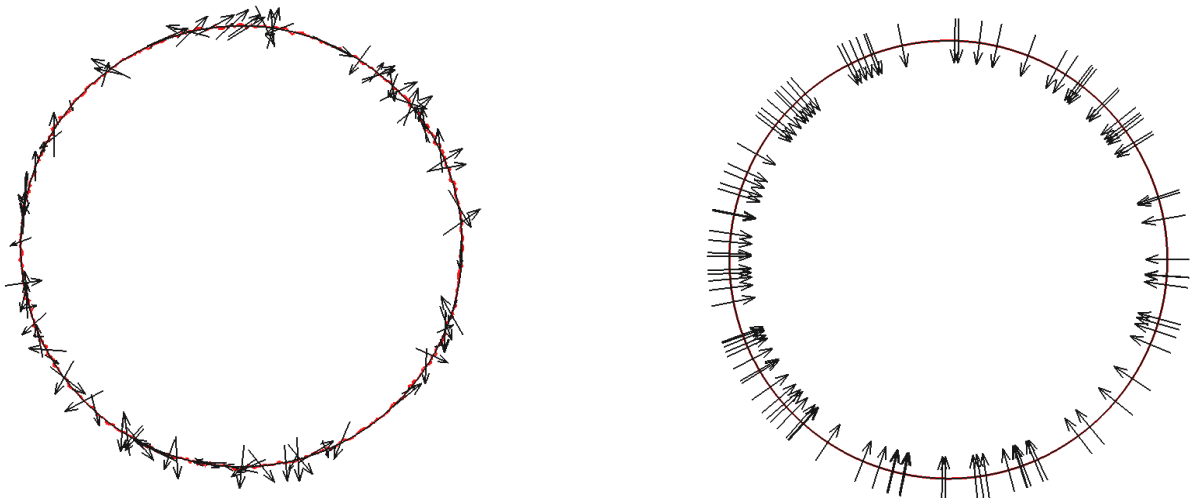
In the case where one projects the gradient of a $\mathbb{P}^1$ field, the field to project ($\boldsymbol{a}_h$ in the formulation (4.6)) is a discontinuous piecewise polynomials of order 0. Since the solution of the problem (4.6) is searched in a continuous space (we recall that $\mathbb{B}_h^n \subset L^2(\Omega) \cap C^0$), the solution is continuous; consequently, in this case, the projection is equivalent to project a discontinuous field on a continuous one.

FEEL++ allows use of a basis with arbitrarily high polynomial order. Consider a graphical example taken from the test presented in section 4.1.3 in which we define a level set field as a circle of radius $r = 0.5$. We want to compute the curvature of the level set field on the iso 0 of the field, and the gradient of the curvature (called $\boldsymbol{i}_h$). It requires then to differentiate 3 times the level set field ϕ_h. The quantities are computed this way, for a level set field $\phi_h \in \mathbb{R}_h^k$, we compute:

$$\begin{aligned} \boldsymbol{n}_h &= \pi_{L^2}^{(\mathbb{R}_h^k \times \mathbb{R}_h^k)}\left(\frac{\boldsymbol{\nabla}\phi_h}{|\boldsymbol{\nabla}\phi_h|}\right) \\ \kappa_h &= \pi_{L^2}^{\mathbb{R}_h^k}(\nabla \cdot \boldsymbol{n}_h) \\ \boldsymbol{i}_h &= \pi_{L^2}^{(\mathbb{R}_h^k \times \mathbb{R}_h^k)}(\boldsymbol{\nabla}\kappa_h) \end{aligned}$$

What is the expected result? The curvature field of the level set has the form of concentric circles and has the value 2 where the level set field is 0. The gradient of the curvature field is supposed to be a vector field for which each vector is pointing to the center of the domain. On this graphical example, we plot the iso value 2 of the curvature κ_h, and some vectors of $\boldsymbol{i}_h$ on this iso line. The figure 4.1(a) represents these values for a level set field taken in $\mathbb{R}_h^2$. The approximation is not sufficient to get a correct value of $\boldsymbol{i}_h$. One can see on the figure that κ_h which carries the information of a field of degree 0 has some oscillations. If one needs only the value of the curvature (as for a surface tension force for example), this approximation could be enough. But if one needs the gradient of κ_h, the result is completely wrong as we can see from the vectors plotted on the graph. On the figure 4.1(b) by contrast, we plotted the same values but by taking an initial approximation $\phi_h \in \mathbb{R}_h^3$. This way, the curvature field κ_h is differentiable and its gradient can be calculated correctly. This method of increasing the approximation order to get the proper derivative is very precise as we will see in the convergence test made in section 4.1.3, even though it suffers a few drawbacks.

No hypothesis has been made on the approximation order of the level set framework and consequently it can work at high order. Nevertheless, a special care has to be taken at the reinitialization step if the fast marching method is used as explained in section 1.3.4. The stabilization methods of the elements presented in section 1.2.2 has also been derived only



(a) $\phi_h \in \mathbb{R}^2_h$ (b) $\phi_h \in \mathbb{R}^3_h$

Figure 4.1: The iso 0 line of the level set field ϕ_h is represented in black. The iso value 2 of the curvature κ_h is represented as the red line, and its gradient $\boldsymbol{i}_h$ as black vectors randomly distributed on the curvature line.

for first order finite elements. These methods also work for high order finite elements but the stabilization factor has to be set *by hand* and to our knowledge no theoretical relation exists which includes the polynomial order (as for the CIP stabilization for example). Moreover, the thickness of the interface ε usually set as $1.5\,h$ might depend on the polynomial order. Indeed, since high order finite elements method are computationally costly, the mesh size is usually increased compared to low order finite elements. Consequently, if the thickness is kept to few mesh size, it can be too large, once again the parameter must be prescribed according to the needs of the simulation. An added drawback with the use of high order finite elements is that not all the visualization software are able to handle them. For example, Paraview that we used in this work can not plot such elements. It is then necessary to use the operator described in section 1.3.4 which creates a first order space out of a high order one. The visualization of a high order field can then be made, but this operation has a cost. Finally, not every finite elements software or library can provide an arbitrary high order finite element basis. Some numerical methods coupled with the level set framework added in the future could also work only at first order (for example some interface localization algorithm). For these reasons, we found necessary to develop a method able to get high order derivative of a low order field. This method is developed in section 4.1.2.

4.1.2 High order derivative by smooth projections

We saw in section 4.1.1 that one can differentiate a finite element field since the number of derivative is lower or equal to the polynomial order of the initial field. But what is happening if one takes a field in a low polynomial approximation set, let us say 1 and differentiate it twice by the method of derivation / projection explained previously? After the first derivation, the derivative is discretized by construction on a continuous polynomial set of order 1 even if it carries only the information of a piecewise discontinuous polynomial set of order 0. The missing information is filled by unpredictable values,

leading to oscillations (as seen in the curvature of figure 4.1(a)) making the differentiated field impossible to differentiate once again. The idea of the method presented here is to add artificial diffusion to the projection (4.3) to minimize the oscillations happening when deriving a variable discretized on a low order polynomial set. One adds a small term proportional to $\boldsymbol{\Delta g}$ which *smooths* the oscillations and makes the obtained derivative possible to derivate. The continuous formulation of such a *smoothed* projection, after integrating by parts is the following, for a given $u \in H^1(\Omega)$, find $\boldsymbol{g} \in [H^1(\Omega)]^d \cap [C^0]^d$ so that $\forall \boldsymbol{v} \in [H^1(\Omega)]^d$:

$$\int_\Omega \boldsymbol{g}\cdot\boldsymbol{v} + \int_\Omega \varepsilon_1 \boldsymbol{\nabla g} : \boldsymbol{\nabla v} - \int_{\partial\Omega} \varepsilon_1 (\boldsymbol{\nabla g}\, \boldsymbol{n})\cdot\boldsymbol{v} = \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{v} \tag{4.8}$$

with ε_1 a sufficiently small parameter. To be precise, $\boldsymbol{\nabla g}\, \boldsymbol{n}$ is the classical product of the matrix $\boldsymbol{\nabla g}$ and the vector $\boldsymbol{n}$, thus it is a vector. One remarks that for $\varepsilon_1 = 1$, we obtain a classical H^1 projection. One needs to impose boundary conditions to the equation (4.8). The third integral could be passed to the right hand side and $\boldsymbol{\nabla g}\, \boldsymbol{n}$ replaced by $\boldsymbol{\nabla}(\boldsymbol{\nabla} u)\boldsymbol{n}$ to impose Neumann boundary conditions but this would require that $u \in H^2(\Omega)$ and we do not want to impose such a strong regularity on u. Instead, we impose weakly Dirichlet boundary conditions to fix the value of $\boldsymbol{g}$ at the boundary using the Nitsche method. First we symmetrize the equation by adding the symmetric of the third integral on both sides of the equation. Then we impose the value of $\boldsymbol{g}$ on the boundary by adding a penalty term. Finally the projection reads, for a given $u \in H^1(\Omega)$, find $\boldsymbol{g} \in [H^1(\Omega)]^d \cap [C^0]^d$ so that $\forall \boldsymbol{v} \in [H^1(\Omega)]^d$:

$$\begin{aligned} &\int_\Omega \boldsymbol{g}\cdot\boldsymbol{v} + \int_\Omega \varepsilon_1 \boldsymbol{\nabla g} : \boldsymbol{\nabla v} - \int_{\partial\Omega} \varepsilon_1 (\boldsymbol{\nabla g}\, \boldsymbol{n})\cdot\boldsymbol{v} - \int_{\partial\Omega} \varepsilon_1 (\boldsymbol{\nabla v}\, \boldsymbol{n})\cdot\boldsymbol{g} \\ &+\gamma_1 \int_{\partial\Omega} \boldsymbol{g}\cdot\boldsymbol{v} = \int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{v} - \int_{\partial\Omega} \varepsilon_1 (\boldsymbol{\nabla v}\, \boldsymbol{n})\cdot\boldsymbol{\nabla} u + \gamma_1 \int_{\partial\Omega} \boldsymbol{\nabla} u \cdot \boldsymbol{v} \end{aligned} \tag{4.9}$$

with γ_1 a big enough parameter. The discrete version reads, for a given $\boldsymbol{\nabla} u_h \in \mathbb{U}_h^{k-1}$, find $\boldsymbol{g}_h \in \mathbb{G}_h^k$ so that $\forall \boldsymbol{v}_h \in \mathbb{G}_h^k$:

$$\begin{aligned} &\int_\Omega \boldsymbol{g}_h\cdot\boldsymbol{v}_h + \int_\Omega \varepsilon_2 \frac{h}{k} \boldsymbol{\nabla g}_h : \boldsymbol{\nabla v}_h - \int_{\partial\Omega} \varepsilon_2 \frac{h}{k} (\boldsymbol{\nabla g}_h \boldsymbol{n})\cdot\boldsymbol{v}_h - \int_{\partial\Omega} \varepsilon_2 \frac{h}{k} (\boldsymbol{\nabla v}_h \boldsymbol{n})\cdot\boldsymbol{g}_h \\ &+\frac{\gamma_2}{h} \int_{\partial\Omega} \boldsymbol{g}_h\cdot\boldsymbol{v}_h = \int_\Omega \boldsymbol{\nabla} u_h \cdot \boldsymbol{v}_h - \int_{\partial\Omega} \varepsilon_2 \frac{h}{k} (\boldsymbol{\nabla v}_h \boldsymbol{n})\cdot\boldsymbol{\nabla} u_h + \frac{\gamma_2}{h} \int_{\partial\Omega} \boldsymbol{\nabla} u_h \cdot \boldsymbol{v}_h \end{aligned} \tag{4.10}$$

with h the element mesh size as a factor of a small parameter ε_2 to insure consistency and γ_2 a penalty parameter strong enough to impose the boundary conditions. Once again we can write this projection in a more general form, it reads:

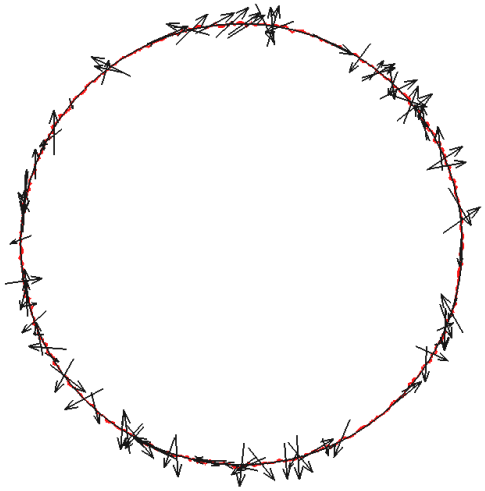
$$\begin{aligned} \pi_{\mathrm{sm}}^{\mathbb{G}_h^k} : \quad & \mathbb{U}_h^{k-1} \to \mathbb{G}_h^k \\ & \boldsymbol{\nabla} u_h \overset{(4.10)}{\mapsto} \boldsymbol{g}_h \end{aligned} \tag{4.11}$$

Of course, this projection also works to get the divergence of a vector by replacing the double contracted product (:) by a scalar product ($\cdot$), and the scalar products of vectors by simple products of scalars. The π_{sm} projection smooths the information, thus it might modify the initial field that one wants to project. Let us test graphically the method on

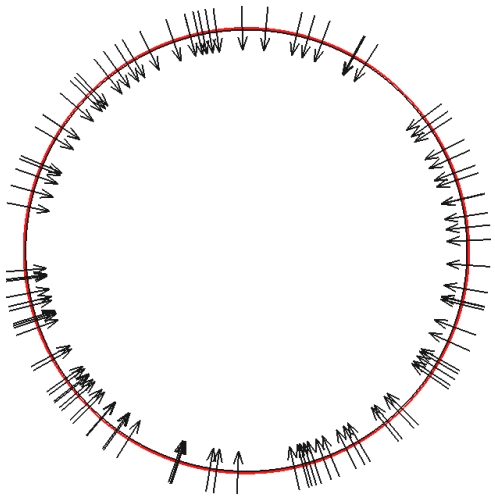
the same example as in section 4.1.1. The calculated quantities are the same except that this time, we use the following projections:

$$\begin{aligned}\boldsymbol{n}_h &= \pi_{\text{sm}}^{(\mathbb{R}_h^k \times \mathbb{R}_h^k)}\left(\frac{\boldsymbol{\nabla}\phi_h}{|\boldsymbol{\nabla}\phi_h|}\right)\\ \kappa_h &= \pi_{\text{sm}}^{\mathbb{R}_h^k}(\nabla \cdot \boldsymbol{n}_h)\\ \boldsymbol{i}_h &= \pi_{\text{sm}}^{(\mathbb{R}_h^k \times \mathbb{R}_h^k)}(\boldsymbol{\nabla}\kappa_h)\end{aligned}$$

The results are presented in figure 4.2. The figure 4.2(a) is the same than 4.1(a) that we reported here to compare more easily the methods. We recall that it uses L^2 projections and that the level set space is $\mathbb{R}_h^2$ making impossible to differentiate twice ϕ_h by this method, consequently, the gradient of the curvature vectors is wrong. We can see now on the figure 4.2(b) that with a smooth projection method, it is indeed possible to derive the level set field in $\mathbb{R}_h^2$. However, the interface moved slightly during the smoothing projections (the red curve representing the iso line 2 of the curvature is not exactly on the iso 0 of ϕ). Thus this method works to get high order curvature but the smoothing parameter ε_2 has to be tuned to get a value strong enough to smooth the field but small enough to keep the approximation correct.



(a) $\phi_h \in \mathbb{R}_h^2$, projectors used: $\pi_{L^2}^{\mathbb{R}_h^2}$ and $\pi_{L^2}^{(\mathbb{R}_h^2 \times \mathbb{R}_h^2)}$



(b) $\phi_h \in \mathbb{R}_h^2$, projectors used: $\pi_{\text{sm}}^{\mathbb{R}_h^2}$ and $\pi_{\text{sm}}^{(\mathbb{R}_h^2 \times \mathbb{R}_h^2)}$

Figure 4.2: The iso 0 line of the level set field ϕ_h is represented in black. The iso value 2 of the curvature κ_h is represented as the red line, and its gradient $\boldsymbol{i}_h$ as black vectors randomly distributed on the curvature line.

4.1.3 Test on the curvature of a circle

We choose to test the order of the curvature that we can get with this method. The test is simple consisting in taking the curvature of a function $\phi \in \mathbb{P}^n$ for which we know an exact value for the curvature and compute the error. The domain Ω is taken as a square of dimensions $(-1, 1) \times (-1, 1)$. The level set function ϕ is taken so that the interface Γ is a circle centered in the domain with a radius equal to 0.5, thus $\phi = \sqrt{x^2 + y^2} - 0.5$.

The level set lines of the function ϕ are concentric circles of radii $r = \sqrt{x^2+y^2}$. The curvature of a circle of radius r is equal to $\frac{1}{r}$. Thus the iso lines of the function curvature defined by $\kappa = \nabla \cdot (\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|})$ are concentric circles of radii $\frac{1}{\sqrt{x^2+y^2}}$. The goal of the test is to compute with the best accuracy possible the curvature of Γ. Theoretically, one has $\kappa_{\text{exact}} = 2$ for any point $(x, y) \in \Gamma$ but numerically, the integral over Γ is replaced by an integral over Ω by multiplying the expression by $\delta_\varepsilon(\phi)$. Since $\delta_\varepsilon(\phi)$ has a given thickness equal to 2ε, assuming that $\kappa_{\text{exact}} = 2$ on all Γ is only first order accurate. The exact value of the curvature for a point $(x, y) \in \Gamma$ is $\kappa_{\text{exact}} = \frac{1}{\sqrt{x^2+y^2}}$. Thus, for this test, the L^2 error will be defined as follow:

$$e_{L^2} = \left[\frac{\displaystyle\int_\Omega \left(\kappa(\phi) - \frac{1}{\sqrt{x^2+y^2}} \right)^2 \delta_\varepsilon(\phi)}{\displaystyle\int_\Omega \delta_\varepsilon(\phi)} \right]^{\frac{1}{2}} \tag{4.12}$$

The simulation is run on a single processor `Intel(R) Xeon(R) CPU E5-2670, 2.60GHz`. The initial mesh size is set to $h = 0.12$ and divided by 2 at each iteration. We have performed 6 iterations for polynomial approximation $\mathbb{P}^1$ and $\mathbb{P}^2$, 5 iterations for $\mathbb{P}^3$ and 4 iterations for $\mathbb{P}^4$ and $\mathbb{P}^5$ because of the increase in computational time.
Figure 4.3 shows the results obtained by the method in which we increase the polynomial order of the initial function ϕ and project each derivative on the same space without smoothing the solution. We can see clearly that the logarithmic slope of the error as a function of the mesh size is increased by one when the polynomial approximation in increased by one as well. Two major consequences can be given from this error analysis. Firstly, it confirms that it is impossible to get a good value of the curvature for a polynomial approximation of 1 by using this method. The error is bigger than unity and never decreases with the mesh size. Secondly, this method can give very high precision to compute the curvature for higher order. Indeed, at the best polynomial approximation that we tried, which is 5, the error decreases as h^4, and we were able to get an error down to $2 \cdot 10^{-8}$ on a single processor. Of course, the computational cost for such an approximation is very high. But for an application for which the approximation of the bending forces has to be really good (for steady shape study, for example), one might consider using high order polynomial approximation. The computational time could be decreased by increasing the number of processors on which running the application.

The figure 4.4 shows the same simulation for which we used the smooth projection as described in (4.10) by taking a factor $\varepsilon_2 = \frac{1}{3}$. We see that for any polynomial order approximation we get a rate of convergence of 1. This convergence at order one is reached when the number of degrees of freedom is high enough. Thus, the two first points of the $\mathbb{P}^1$ and $\mathbb{P}^2$ approximations do not respect this behavior. Thus, two main conclusions can be found from this simulation. Firstly, the absolute error at high order is big. Even at a polynomial approximation of order 5, the best error we get with the smallest mesh size is bigger than 10^{-2}. Secondly, the only configuration for which the multiple derivation by smooth projection is better than without smooth projection is for a polynomial approxi-

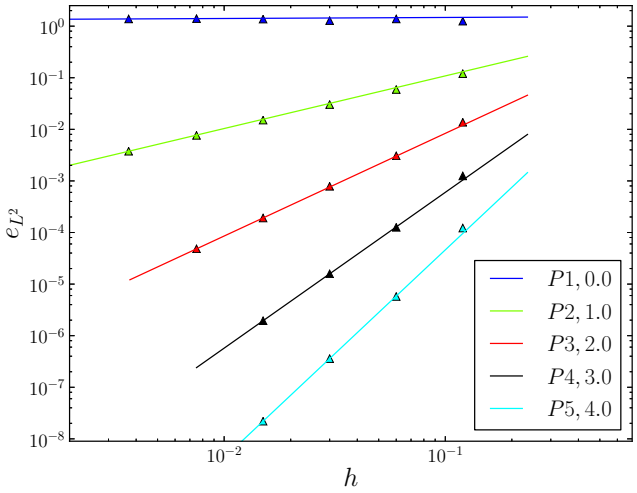


Figure 4.3: L^2 error as a function of mesh size for different polynomial approximation order ($\mathbb{P}^n$) using a L^2 projection. The legend gives the polynomial approximation order of the initial level set field and the slope associated to the error.

mation lower than the degree of differentiation needed. Indeed, in figure 4.5 we reported the errors for a $\mathbb{P}^1$ approximation for both methods. We see clearly that, even if the approximation is not very precise with smoothing projection, it is better than without. Consequently, for a simulation in which one can go to high order approximation, and if the precision required for the derivative is not too high (for a suspension of vesicles in which the major effect would be the volume fraction), the smoothing method is a good option.

4.2 Inextensibility by elastic force

4.2.1 Principle of the method

One method to model a membrane having its surface area conserved in time is to introduce a strong elastic force $\boldsymbol{F}_{\text{el}}$ on it. This force derives from an elastic energy. The energy depends on the stretching of the interface and on a scaling parameter being the energy cost to pay to stretch it. Thus, for a huge stretching energy cost, the membrane would be quasi-inextensible. It has been shown in [20, 21, 76] that the level set field can record the information of the variation of surface area (the stretching). Indeed, if there is no reinitialization between iteration 0 and iteration n, the quantity $\dfrac{|\boldsymbol{\nabla}\phi_n|}{|\boldsymbol{\nabla}\phi_0|}$ is proportional to the relative change in surface area. The mathematical proof has been given in [20], but it can be explained in a simple manner. Indeed, in figure 4.6, it is shown that when the interface is stretched, somehow, to fulfill the divergence free condition, the fluid has to pull in the direction perpendicular to the stretching direction. Thus, the level set lines get closer to each other and $|\boldsymbol{\nabla}\phi|$ increases. A force can be derived from this information, thus, the model of vesicle using such inextensibility force is such that one adds two external forces to the fluid solver depending only on the level set field. The big advantage of this

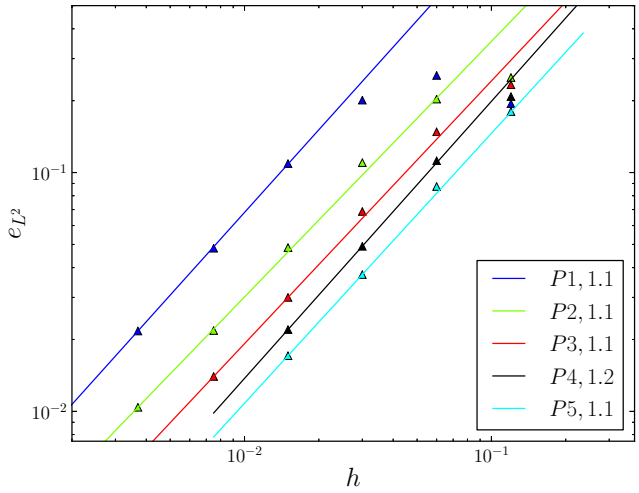


Figure 4.4: L^2 error as a function of mesh size for different polynomial approximation orders ($\mathbb{P}^n$) using a smooth projection method. The legend gives the polynomial approximation order of the initial level set field and the slope associated to the error.

method is that the fluid solver can be seen as a black box taking as entry an external force field, and giving as output the velocity and pressure solutions. This is an advantage if one wants to test different numerical techniques to solve a fluid equation or even test different fluid models. Another advantage is that the time cost to solve such a problem would be almost independant of the number of vesicles in the flow. As a matter of facts, the total number of elements in the final problem is always the same if one increases the number of vesicles. Only the number of non-zero entries in the force vector is increased. This is a big advantage compared to the method described in section 4.3 in which the Lagrange multiplier increases the number of unknowns to find. This method can be seen as a penalty method to restrict surface area change. It is not an exact method and suffers from some technical difficulties that we will describe later on. First, let us describe the method in detail. The elastic energy E_{el} is defined as:

$$E_{\text{el}} = \int_{\Omega} E(|\boldsymbol{\nabla}\phi|)\delta_{\varepsilon}, \tag{4.13}$$

where $E(|\boldsymbol{\nabla}\phi|)$ is a constitutive law for the membrane such that $E(1) = 0$ meaning that there is no initial stretching. Different models could be used for $E(|\boldsymbol{\nabla}\phi|)$, usually we prefer to describe $E'(|\boldsymbol{\nabla}\phi|)$ the derivative of $E(|\boldsymbol{\nabla}\phi|)$ which is directly used in the expression of the elastic force $\boldsymbol{F}_{\text{el}}$. A simple expression usually sufficient to get most of the effects is a linear law $E'(|\boldsymbol{\nabla}\phi|) = \Lambda(|\boldsymbol{\nabla}\phi| - 1)$ with Λ the energy cost per unit of surface to stretch the membrane. For this expression, the energy would be a quadratic law $E(|\boldsymbol{\nabla}\phi|) = \frac{\Lambda}{2}(|\boldsymbol{\nabla}\phi| - 1)^2$ making the membrane modeled as a *spring*, pulling when it is stretched and pushing back when it is compressed. A more realistic model is to cut the force when the membrane is compressed, making the membrane more like a *tensile* or a *balloon*, the derivative of the energy would be then $E'(|\boldsymbol{\nabla}\phi|) = \Lambda \min(|\boldsymbol{\nabla}\phi| - 1, 0)$. It has been shown in [20] that after having derived the energy (4.13), the elastic force takes

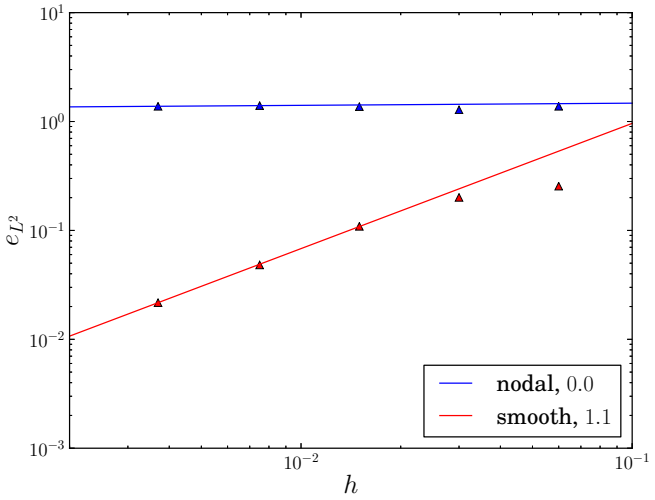


Figure 4.5: Comparison of the L^2 error on the curvature as a function of mesh size for a polynomial approximation of first order.

the form:

$$\boldsymbol{F}_{\text{el}} = \int_\Omega \left\{ \boldsymbol{\nabla} E'(|\boldsymbol{\nabla}\phi|) - \nabla \cdot \left[E'(|\boldsymbol{\nabla}\phi|) \frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|} \right] \frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|} \right\} \delta_\varepsilon. \tag{4.14}$$

If we re-write this equation by replacing $\dfrac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}$ by $\boldsymbol{n}$ and $E'(|\boldsymbol{\nabla}\phi|)$ by the spring like model, we obtain:

$$\boldsymbol{F}_{\text{el}} = \Lambda \int_\Omega \boldsymbol{\nabla}(|\boldsymbol{\nabla}\phi| - 1) - \nabla \cdot \left[(|\boldsymbol{\nabla}\phi| - 1)\boldsymbol{n} \right] \delta_\varepsilon \boldsymbol{n}. \tag{4.15}$$

4.2.2 Record the stretching information

Several difficulties arise when using this method. The first one has already been seen in the previous section: the high order derivative of the level set field. Indeed, the formulation (4.15) leads to a 2^{nd} derivative of the level set field. This problem has already been addressed in section 4.1. Another difficulty is that for a given simulation, one needs to keep the information $\dfrac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}$ during all the simulation time. Thanks to the elastic force $\boldsymbol{F}_{\text{el}}$, this quantity is supposed to stay close to 1 on the membrane. The problem is that it can reach really high values away from the membrane. For example, for a simulation of one vesicle at the center of a shear flow, the velocity induced by the wall forces the level set field to move in one direction since its value has to be 0 on the membrane. This results in an accumulation of level set lines on one side of the vesicle and a depletion in the other side. The gradient magnitude of the level set takes extremes values, leading to instabilities when solving the advection equation 1.11. For other level set simulations, the reinitialization resets $|\boldsymbol{\nabla}\phi|$ to values close to 1 and the problem is solved. But doing so in this context would lose the stretching information. A way to overcome this last problem

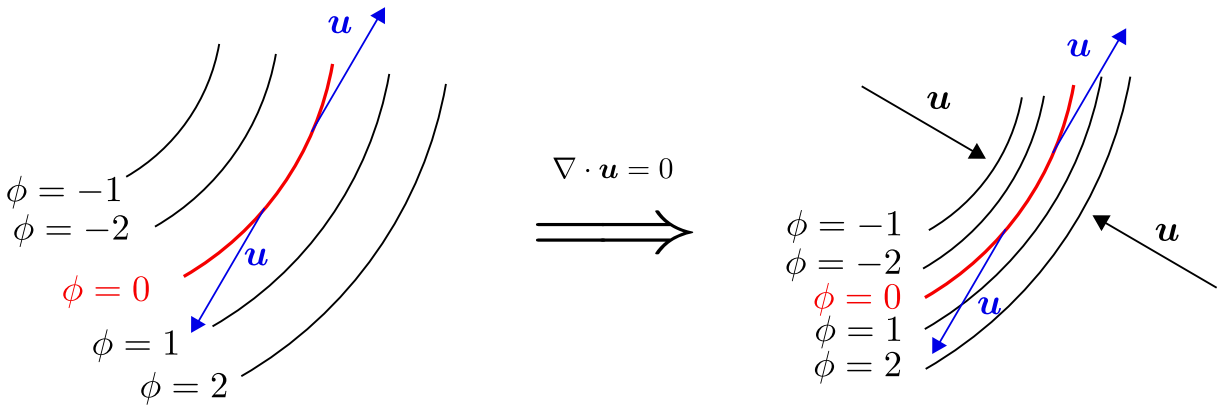


Figure 4.6: Scheme explaining how the divergence free velocity implies that $|\boldsymbol{\nabla}\phi|$ records the stretching of the interface. If the interface is stretched it means that the velocity field is *pulling* on it from two sides as shown on the left scheme. Since the velocity is divergence free, it has to push in the other direction to fulfill this condition. Thus, the level set lines are getting closer, consequently, the gradient magnitude of ϕ increases.

is to introduce $e = |\boldsymbol{\nabla}\phi|$, an alternative field being governed by the equation:

$$\partial_t e + \boldsymbol{u} \cdot \boldsymbol{\nabla} e = -e \frac{\boldsymbol{\nabla}\phi \otimes \boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|^2} : \boldsymbol{D}(\boldsymbol{u}). \tag{4.16}$$

This field is set initially to $e_0 = |\boldsymbol{\nabla}\phi_0|$ and then advected as an independent field. It can be solved numerically by the numerical framework presented in sections 1.2 and 5.2 as for the advection of the level set, by changing the right hand side $\boldsymbol{f}$. This method of taking the gradient magnitude of the level set field as an independent field is known as the *gradient augmented level set method* in [78], it has also been used in the vesicle simulation context in [92]. A first advantage of using this method is that it reduces the degree of derivation of equation (4.14) by one. Indeed, replacing $|\boldsymbol{\nabla}\phi|$ by e in equation (4.15) leads to only one differentiation of the field e. This field e is usually discretized on the same polynomial set as the level set field.
The problem of the extremes values of $|\boldsymbol{\nabla}\phi|$ is still here. The solution that we propose is to reinitialize the level set field and the field e, but to keep the information of the field e before reinitialization around the interface.

By doing so, after reinitializing, ϕ is equal to a distance function, e is almost equal to the unit everywhere and we reset e to its value before the reinitialization around the interface. This does not lead to a huge discontinuity of e since it is supposed to stay close to one thanks to the strong elastic force. Moreover, the discontinuity can be smeared out by using a Heaviside like function.

Let us call e^* and $\tilde{e}$ respectively the values of e before and after the reinitialization step ($\tilde{e}$ is almost equal to 1 everywhere). We call e the value of $\tilde{e}$ for which we reincorporated the information of the stretching before reinitialization. The value of e can be obtained by:

$$e = \tilde{e} + e^* R_{\varepsilon_2}(\phi) \tag{4.17}$$

with R_{ε_2} the smoothed rectangular function being equal to one in a thickness ε_2 around the interface. It can be defined by:

$$R_{\varepsilon_2} = H_\varepsilon(\phi + \varepsilon_2) - H_\varepsilon(\phi - \varepsilon_2). \tag{4.18}$$

The parameter ε_2 has to be taken larger than the interface thickness so that inside the interface one has $R_{\varepsilon_2} = 1$. One can also replace R_{ε_2} by a characteristic function if the discontinuities involved are not too large.

4.2.3 Equations and dimensionless numbers

A complete set of equations to solve a problem of vesicles flowing in a Newtonian fluid with the inextensibility ensured by a strong elastic force reads:

$$\begin{aligned}
\rho_\phi &= \rho_2 + (\rho_1 - \rho_2) H_\varepsilon(\phi) \\
\mu_\phi &= \mu_2 + (\mu_1 - \mu_2) H_\varepsilon(\phi) \\
\boldsymbol{F}_{\mathrm{b}} &= k_B \int_\Omega \boldsymbol{\nabla} \cdot \left[\frac{-\kappa^2}{2} \boldsymbol{n} + \frac{1}{|\boldsymbol{\nabla}\phi|} (\boldsymbol{I}_{\mathrm{d}} - \boldsymbol{n} \otimes \boldsymbol{n}) \boldsymbol{\nabla} \{ |\boldsymbol{\nabla}\phi| \kappa \} \right] \delta_\varepsilon(\phi) \\
\boldsymbol{F}_{\mathrm{el}} &= \Lambda \int_\Omega \boldsymbol{\nabla}(e-1) - \nabla \cdot \left[(e-1) \boldsymbol{n} \right] \boldsymbol{n} \, \delta_\varepsilon(\phi) \\
\rho_\phi(\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}) - \mu_\phi \boldsymbol{\Delta} \boldsymbol{u} + \boldsymbol{\nabla} p &= \boldsymbol{F}_{\mathrm{el}} + \boldsymbol{F}_{\mathrm{b}} \\
\nabla \cdot \boldsymbol{u} &= 0 \\
\partial_t \phi + \boldsymbol{u} \cdot \boldsymbol{\nabla} \phi &= 0 \\
\partial_t e + \boldsymbol{u} \cdot \boldsymbol{\nabla} e &= -e \, (\boldsymbol{n} \otimes \boldsymbol{n}) : \boldsymbol{D}(\boldsymbol{u}).
\end{aligned}$$

A typical simulation is solved by the algorithm given in algorithm 6. The condition **Reinitialize** is often set by a reinitialization frequency. It can also be monitored by the max of the gradient magnitude of the level set field.

Algorithm 6 Algorithm for the coupling of level set simulation of vesicles by using a stretching force to ensure inextensibility of the membrane and a fluid solver.

Require: ϕ^0
  for $n = 0$ to N_{t_f} **do**
    ρ_ϕ^{n+1} = **UpdateDensity**(ϕ^n), eq (2.25)
    μ_ϕ^{n+1} = **UpdateViscosity**(ϕ^n), eq (2.26)
    $\boldsymbol{F}_{\mathrm{b}}^{n+1}$, $\boldsymbol{F}_{\mathrm{el}}^{n+1}$ = **UpdateForces**($\phi^n, \boldsymbol{n}(\phi^n), \kappa(\phi^n)$)
    $(\boldsymbol{u}^{n+1}, p^{n+1})$ = **FluidSolver**($\rho_\phi^{n+1}, \mu_\phi^{n+1}, \boldsymbol{F}_{\mathrm{b}}^{n+1}, \boldsymbol{F}_{\mathrm{el}}^{n+1}$)
    $\tilde{\phi}^{n+1}, \tilde{e}^{n+1}$ = **AdvectionSolvers**($\boldsymbol{u}^{n+1}, \phi^n, e^n$), eq (2.33), eq (4.16)
    if Reinitialize then
      ϕ^{n+1} = **Reinitialization**($\tilde{\phi}^{n+1}$), Fast Marching or Hamilton Jacobi equation
      e^{n+1} = **SaveStretchingInformation**(e^n, $\tilde{e}^{n+1}$), eq (4.17)
    else
      $\phi^{n+1} = \tilde{\phi}^{n+1}$
      $e^{n+1} = \tilde{e}^{n+1}$
    end if
  end for

There are many parameters to set for this simulation. All the parameters are not independent and we propose now to see how we can reduce the physical parameters to a few dimensionless parameters controlling the behavior of the simulation. To do so, as usual in fluid mechanics, we need characteristic values. Let us call L the characteristic

length, U the characteristic velocity of a particular flow and τ a characteristic time. For a simulation of vesicles, the typical length L of the flow is usually taken as $R_0 = \frac{p_{\text{er}}}{2\pi}$, the radius of a circle having the same perimeter as the vesicle. A simulation given by the equations presented above is typically governed by the values of a few dimensionless numbers. The first one is the classical Reynolds number R_e, representing the ratio between the inertial and the viscous terms. We recall that its expression is:

$$R_e = \frac{\rho U L}{\mu}.$$

The second dimensionless number is the Weissenberg number. Usually, the Weissenberg number is used for a viscoelastic fluid and represents the ratio between the characteristic time of the fluid and the characteristic time of one specific event. In this case, the event is the relaxation time of the membrane under the elastic force. The Weissenberg number is given by:

$$W_{\text{e}} = \frac{\mu U}{\Lambda}.$$

The third dimensionless number is the capillary number associated to the curvature force. It represents the relative strength of the hydrodynamic forces and the curvature force. It reads:

$$C_{\text{a}} = \frac{\mu U L^2}{k_{\text{B}}}.$$

Of course the dynamics of the vesicle is also governed by the viscosity ratio $\frac{\mu_2}{\mu_1}$ and the density ratio $\frac{\rho_1}{\rho_2}$. All these parameters should in principle rule all the dynamics of the simulation from a physical point of view. Of course, there are still physical parameters to set, and the relative value of these parameters can not be taken randomly as well. Thus we shall see what range of dimensionless parameter values are acceptable. Firstly, for our simulations the Reynolds number is very low $O(10^{-4})$ and we solve the Stokes equations. The capillary number is taken between 10^{-1} and 10. The Weissenberg number has to be quite low to insure inextensibility. A better estimation of how this number should be set is to look at the ratio between the typical time scale of the curvature force ($\tau_{k_{\text{B}}} = \frac{\mu L^3}{k_{\text{B}}}$) and the one of the elastic force ($\tau_\Lambda = \frac{\mu L}{\Lambda}$). The typical time associated to the elastic force has to be much smaller than time associated to the curvature force so that the inextensibility force has a very fast response. The ratio between those two time scales is given by:

$$R_\tau = \frac{\tau_\Lambda}{\tau_{k_{\text{B}}}} = \frac{k_{\text{B}}}{\Lambda L^2}.$$

This ratio takes typical values between 10^{-3} and 10^{-5}. The numerical parameters have to be set as well according to the numerical values given to the physical parameters. In particular, the time discretization choice is critical in this simulation. Indeed, the elastic force being really strong for a good local surface conservation, the time step has to be small enough so that the force does not lead to instabilities. This parameter is difficult to choose. To start, a relation between the numerical ingredients should be:

$$dt < \frac{h}{\max(U_\Lambda, U_{k_{\text{B}}}, U)}$$

where U_Λ and U_{k_B} are the typical velocities induced by the elastic force and the curvature force, and U is the typical velocity of the flow. Replacing the velocities by their values leads to the relation:

$$dt < \frac{h}{\max\left(\frac{k_\mathrm{B}}{\mu L^2}, \frac{\Lambda}{\mu}, U\right)}.$$

This relation is a good starting point to check the range in which should be dt, however, usually one has to try different values to find the optimal one not leading to instabilities while keeping a reasonable computational time. In the previous formulation, usually the leading term is the velocity due to the inextensibility force. Indeed, if it is not the case, the membrane has some extensibility.

4.3 Inextensibility by Lagrange multiplier

4.3.1 Introduction of the Lagrange Multiplier

A more stable way than having a force to impose inextensibility of the membrane is to impose this constrain by adding a Lagrange multiplier to the equations. This method has been employed in [62] and we propose to see in this section its implementation within our framework. The idea is to insure that the velocity of the membrane is divergence free. Since in this model, the membrane is a part of the fluid, we can write this constraint as the fact that in the fluid, the surface divergence on the interface Γ has to vanish, that is to say:

$$\nabla_\mathrm{s} \cdot \boldsymbol{u} = 0 \quad \text{on} \quad \Gamma \tag{4.19}$$

where $\nabla_\mathrm{s} \cdot \boldsymbol{u}$ is the surface divergence, it can be written as: $\nabla_\mathrm{s} \cdot \boldsymbol{u} = (\boldsymbol{I}_\mathrm{d} - \boldsymbol{n} \otimes \boldsymbol{n}) : \boldsymbol{\nabla u} = \nabla \cdot \boldsymbol{u} - (\boldsymbol{\nabla u}\ \boldsymbol{n}) \cdot \boldsymbol{n}$. One can see appendix G for the indexes notation of the surface divergence. This constraint on the fluid is added to the fluid system of equation. For the sake of simplicity, we will only write the incompressible Stokes equations for this part but there is no problem to extend it to the Navier-Stokes equations. The system of equation reads now:

$$\begin{aligned} -2\mu \nabla \cdot \boldsymbol{D}(\boldsymbol{u}) + \boldsymbol{\nabla} p &= \boldsymbol{f} \quad \text{in} \quad \Omega \\ \nabla \cdot \boldsymbol{u} &= 0 \quad \text{in} \quad \Omega \\ \nabla_\mathrm{s} \cdot \boldsymbol{u} &= 0 \quad \text{on} \quad \Gamma. \end{aligned}$$

Boundary conditions are also applied depending on the type of flow needed, for the sake of simplicity, in this example we will consider that we impose Dirichlet boundary conditions on $\partial\Omega$. The variational formulation of the Stokes equations are classically obtained as for the problem 2.17 by using the strain tensor $\boldsymbol{D}(\boldsymbol{u})$. A Lagrange multiplier λ is also added to insure the constraint (4.19). The problem reads: for a given $\boldsymbol{f} \in [L^2(\Omega)]^d$, find

$(\boldsymbol{u}, p, \lambda) \in [H^1(\Omega)]^d \times L^2_0(\Omega) \times H^{1/2}(\Gamma)$ so that $\forall (\boldsymbol{v}, q, \nu) \in [H^1_0(\Omega)]^d \times L^2_0(\Omega) \times H^{1/2}(\Gamma)$:

$$\int_\Omega 2\mu \boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{D}(\boldsymbol{v}) - \int_\Omega p \, \nabla \cdot \boldsymbol{v} + \int_\Gamma \lambda \, \nabla_{\mathrm{s}} \cdot \boldsymbol{v} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \tag{4.20}$$

$$\int_\Omega q \, \nabla \cdot \boldsymbol{u} = 0 \tag{4.21}$$

$$\int_\Gamma \nu \nabla_{\mathrm{s}} \cdot \boldsymbol{u} = 0 \tag{4.22}$$

The third integral of equation 4.20 might theoretically not be always defined. It should actually be seen as a duality bracket: $< \lambda, \nabla_{\mathrm{s}} \cdot \boldsymbol{v} >$. In practice, the functions we are dealing with are regular enough so that the integral is always defined, and we will prefer the integral notation to the bracket one.

We can see what λ represents physically. As we can see in equation (4.20), the Lagrange multiplier plays a role very similar to the pressure. Indeed, the pressure p can be seen as a Lagrange multiplier imposing the constraint $\nabla \cdot \boldsymbol{u} = 0$ on all the domain, whereas λ imposes $\nabla_{\mathrm{s}} \cdot \boldsymbol{u} = 0$ on the interface. We can do a simple dimensional analysis on equation (4.20) to express what this parameter is physically. We recall that d is the topological dimension of the problem (2 or 3), F and L the dimensions of force and length and $[A]$ the dimension of a quantity A. Equation (4.20) shows that we have:

$$[\boldsymbol{f}] = \frac{F}{L^d}$$
$$[p] = \frac{F}{L^{(d-1)}}$$
$$[\lambda] = \frac{F}{L^{(d-2)}}$$

Thus, in 3D, $\boldsymbol{f}$ is a body force, p a surface force and λ a line force. Whereas in 2D, $\boldsymbol{f}$ is a surface force, p a line force and λ a point force. Thus λ is like a *tension*, that is to say a pressure of a topological dimension lower than the pressure p. Moreover, the other difference between p and λ is that p is defined on the all domain Ω whereas λ is restricted on the interface Γ.

4.3.2 Discretization

We discretize the equations (4.20), (4.21) and (4.22) by introducing the finite element spaces $\mathbb{U}_h^{n+1} \subset [H^1(\Omega)]^d$, $\mathbb{P}_h^n \subset L^2(\Omega)$ and $\mathbb{L}_h^n \subset L^2(\Omega)$. These spaces can be defined as:

$$\mathbb{U}_h^{n+1} = \{\boldsymbol{u}_h | \boldsymbol{u}_h \in [C^0]^d, \boldsymbol{u}_h = \sum_{i=1}^{N_{\mathrm{dof}}^{\mathbb{U}_h^{n+1}}} \boldsymbol{u}_i \boldsymbol{\Phi}_i\}$$

$$\mathbb{P}_h^n = \{p_h | p_h \in C^0, p_h = \sum_{i=1}^{N_{\mathrm{dof}}^{\mathbb{P}_h^n}} p_i \phi_i\}$$

$$\mathbb{L}_h^n = \{\lambda_h | \lambda_h \in C^0, \lambda_h = \sum_{i=1}^{N_{\mathrm{dof}}^{\mathbb{L}_h^n}} \lambda_i \xi_i\},$$

where $\boldsymbol{\Phi}_i$, ψ_i and ξ_i are the finite element basis functions being Lagrange polynomials of orders $n+1$, n and n respectively. We introduce $(\boldsymbol{u}_h, p_h, \lambda_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n \times \mathbb{L}_h^n$ the discrete versions of $(\boldsymbol{u}, p, \lambda)$. The discrete version of equations (4.20), (4.21) and (4.22) reads, for a given $\boldsymbol{f}_h$ find $(\boldsymbol{u}_h, p_h, \lambda_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n \times \mathbb{L}_h^n$ so that $\forall (\boldsymbol{v}_h, q_h, \nu_h) \in \mathbb{U}_h^{n+1} \times \mathbb{P}_h^n \times \mathbb{L}_h^n$:

$$\int_\Omega 2\mu_\phi \boldsymbol{D}(\boldsymbol{u}_h) : \boldsymbol{D}(\boldsymbol{v}_h) - \int_\Omega p_h \, \nabla \cdot \boldsymbol{v}_h + \int_\Omega \lambda_h \, \nabla_\mathrm{s} \cdot \boldsymbol{v}_h \, \delta_\varepsilon(\phi) = \int_\Omega \boldsymbol{f}_h \cdot \boldsymbol{v}_h \tag{4.23}$$

$$\int_\Omega q_h \, \nabla \cdot \boldsymbol{u}_h = 0 \tag{4.24}$$

$$\int_\Omega \nu_h \nabla_\mathrm{s} \cdot \boldsymbol{u}_h \, \delta_\varepsilon(\phi) = 0, \tag{4.25}$$

where we replaced the integral on Γ by an integral on Ω thanks to the delta function. This is important from the point of view of the dimensional analysis we performed previously. Indeed, as opposed to λ, λ_h is defined on the domain Ω even if most of its values are 0 on it. Thus it is defined on a space with the same topological dimension as the space of p. If we perform a dimensional analysis of equation (4.23), we can show that

$$[\lambda_h] = [p_h] = \frac{F}{L^{d-1}}.$$

Consequently, by discretizing the Lagrange multiplier and projecting it on a space of topological dimension superior to the one where it belongs, we changed its physical dimensions and transformed it to a pressure. It can be seen as a pressure acting on a very small portion of the fluid trying to keep the surface divergence free. Thus, the pressure p_h and the Lagrange multiplier λ_h are the same kind of objects. To put this problem in an algebraic form, we introduce the following matrices:

$$A, A_{ij} = \int_\Omega 2\mu \boldsymbol{D}(\boldsymbol{\Phi}_j) : \boldsymbol{D}(\boldsymbol{\Phi}_i)$$

$$B, B_{ij} = \int_\Omega -\psi_j \nabla \cdot \boldsymbol{\Phi}_i$$

$$C, C_{ij} = \int_\Omega \delta_\varepsilon \, \xi_j \nabla_\mathrm{s} \cdot \boldsymbol{\Phi}_i$$

and the following vectors:

$$U, \boldsymbol{u}_j$$
$$P, p_j$$
$$Q, \lambda_j$$
$$F, \boldsymbol{f}_j.$$

Then, the problem (4.23) (4.24) (4.25) can be written as:

$$\begin{pmatrix} A & B & C \\ B^T & 0 & 0 \\ C^T & 0 & 0 \end{pmatrix} \begin{pmatrix} U \\ P \\ Q \end{pmatrix} = \begin{pmatrix} F \\ 0 \\ 0 \end{pmatrix} \tag{4.26}$$

This algebraic system can be solved by a direct solver. The same method can be applied for the Navier-Stokes equation, for which a non-linear solver is used (Newton).

4.4 Validation

In this section we will try to validate our two different methods on several known behaviors of vesicles. The first one is the equilibrium shape of a vesicle in a fluid at rest. The other ones are the behaviors of vesicles under shear flow. In this system two remarkable motions are well known: the tank-treading and the tumbling motions. We will show that we can recover them easily with the Lagrange multiplier method and explain why it is difficult to do the same with the forces methods.

4.4.1 Equilibrium shape

The first validation of the model that we address is the equilibrium shapes of the vesicles. If one puts a vesicle in a fluid at rest in a shape which does not minimize the Helfrich energy (4.1), the bending force is not zero and leads to a movement of the membrane which causes a motion of the fluid as well. The force and the velocity vanish when the membrane attains its equilibrium shape. When the inner of two fluids is a drop, the surface energy is proportional to the curvature which leads to a minimization of the shape of a sphere. The case is different for a vesicle, indeed, the surface energy is proportional to the square of the curvature which leads to a different minimal energy shape. It has been shown [24, 95, 94, 1] that the equilibrium shape depends strongly on a geometrical parameter being given by the ratio between the volume and the area of the vesicle. A dimensionless parameter has been defined to quantify this ratio. It is called the *reduced volume*, it is defined as the ratio of the volume of the vesicle and the volume of a sphere having the same area as the vesicle. In two dimensions, the parameter is the *reduced area* α which is the ratio between the area of the vesicle and the area of a disk having the same perimeter. Its definition is

$$\alpha = \frac{4\pi A}{p_{\text{er}}^2}, \tag{4.27}$$

where A is the area of the vesicle and p_{er} its perimeter. This coefficient measures how much the vesicle is swollen or deflated. A reduced area of 1 means that the vesicle is circular, thus completely swollen. If one considers a vesicle for which $\alpha < 1$, it means that the vesicle is deflated. It has been shown that a deflated vesicle tends to take a bi-concave shape as shown in figure 4.7 taken from [95]. This test will mostly validate the expression of the bending force $\boldsymbol{F}_{\text{b}}$ since the equilibrium shape is totally dependent on it. The inextensibility is not too difficult to achieve in this simulation. Indeed, the initial shape that we set is not too far away from the equilibrium shape and the movement that the vesicle has to do is relatively small. Thus, the vesicle is not stretched too much. The difficulties of this test are, firstly to calculate precisely the bending force. Secondly, since this force vanishes close to the equilibrium, the time for which the equilibrium shape is reached can be long. Thus the numerical error, especially the loss of mass induced by the level set advection, has to be controlled. We chose to compare our results to those of Badr Kaoui obtained during his PhD [56] and published in [53].

In [53], Kaoui used a Lattice Boltzmann method to compute the equilibrium shapes of the vesicles that he compared to the one he obtained with the Boundary Integral Method. Both methods showed the same results. The numerical data being available to us, we chose to compare our results to it. Another possibility could have been to solve coupled EDO as shown in [62]. Since the inextensibility of the membrane is not the most critical

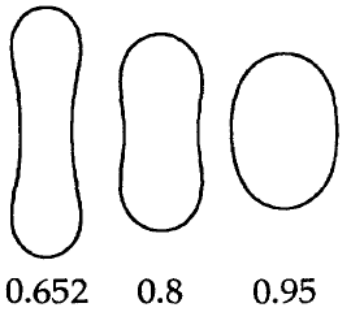


Figure 4.7: Scheme taken from [95] of the equilibrium shapes of a vesicle for different reduced volumes. The values of the reduced volumes are indicated at the bottom of each vesicle. One can see clearly the transition from a quasi circular vesicle to a bi-concave one, from right to left.

point of this simulation, it is insured by the inextensibility force seen in section 4.2 and we did not performed the same study with the Lagrange multiplier method which would have lead to a similar result. For this simulation, the domain has been taken as a square of 8×8, the mesh size was $h = 0.16$ away from the center and a box with a better mesh around the vesicle has been taken with $h = 0.064$. The fluid equations are the incompressible Navier-Stokes equations with Dirichlet boundary conditions. The zero mean pressure is imposed by a Lagrange multiplier. The viscosities inside and outside are $\mu_1 = \mu_2 = 1$ and the volumic mass $\rho_1 = \rho_2 = 10^{-5}$. The vesicles are taken initially as ellipses with major and minor axes a and b, since this shape is not the equilibrium shape, the vesicle's membrane will start to move. We have already seen in section 1.3.3 that we can put the following field as initial level set field:

$$\phi_0 = \sqrt{\left(\frac{x-x_0}{a}\right)^2 + \left(\frac{y-y_0}{b}\right)^2} - 1$$

which is not a signed distance function but has the property to be an ellipse on the iso-0 and positive outside and negative inside the ellipse. Thus we can start with this function, and the reinitialization procedure resets it to a distance function quickly. For this simulation, the fast marching method presented in section 1.3.4 has been used as reinitialization procedure. The chosen reduced areas go from 1 to 0.6. The perimeter p_{er} is fixed to 8, the equivalent radius is $R_0 = 1.27$. A simple numerical way to find a and b when α and p_{er} (or A) are fixed is presented in appendix B. The parameters for the forces are fixed to $k_B = 0.01$ and $\Lambda = 400$ giving a time ratio $R_\tau = 1.5 \times 10^{-5}$, thus the inextensibility of the membrane should be correctly maintained. With this parameter, if we consider the previous study of the relevant velocity, the time step should be really small. But in this particular case, the membrane will move smoothly and slowly and will not be too much stretched. Thus, the inextensibility force should stay small and the time step can rather be set according to the time scale of the bending force. Taking the previous scaling, the time step should follow: $dt < \dfrac{h\mu R_0^2}{k_B} = 0.64$. We chose a time step $dt = 0.01$, one order of magnitude smaller. The time discretization was BDF2 and the stabilization method GALS. The simulations have been run on a single processor. Figure 4.8 shows the result of the simulation for 4 different reduced areas that we already

presented in [28]. The results of our simulations are represented by blue dots and the one of Kaoui by red lines. One can see that there is a small difference in the results due to the error on the total surface conservation by the level set method. The relative variation of area is lower than 4%, this value is already large but enough to validate shape of the vesicles and thus the curvature force.

4.4.2 Dynamics of single vesicle under shear flow

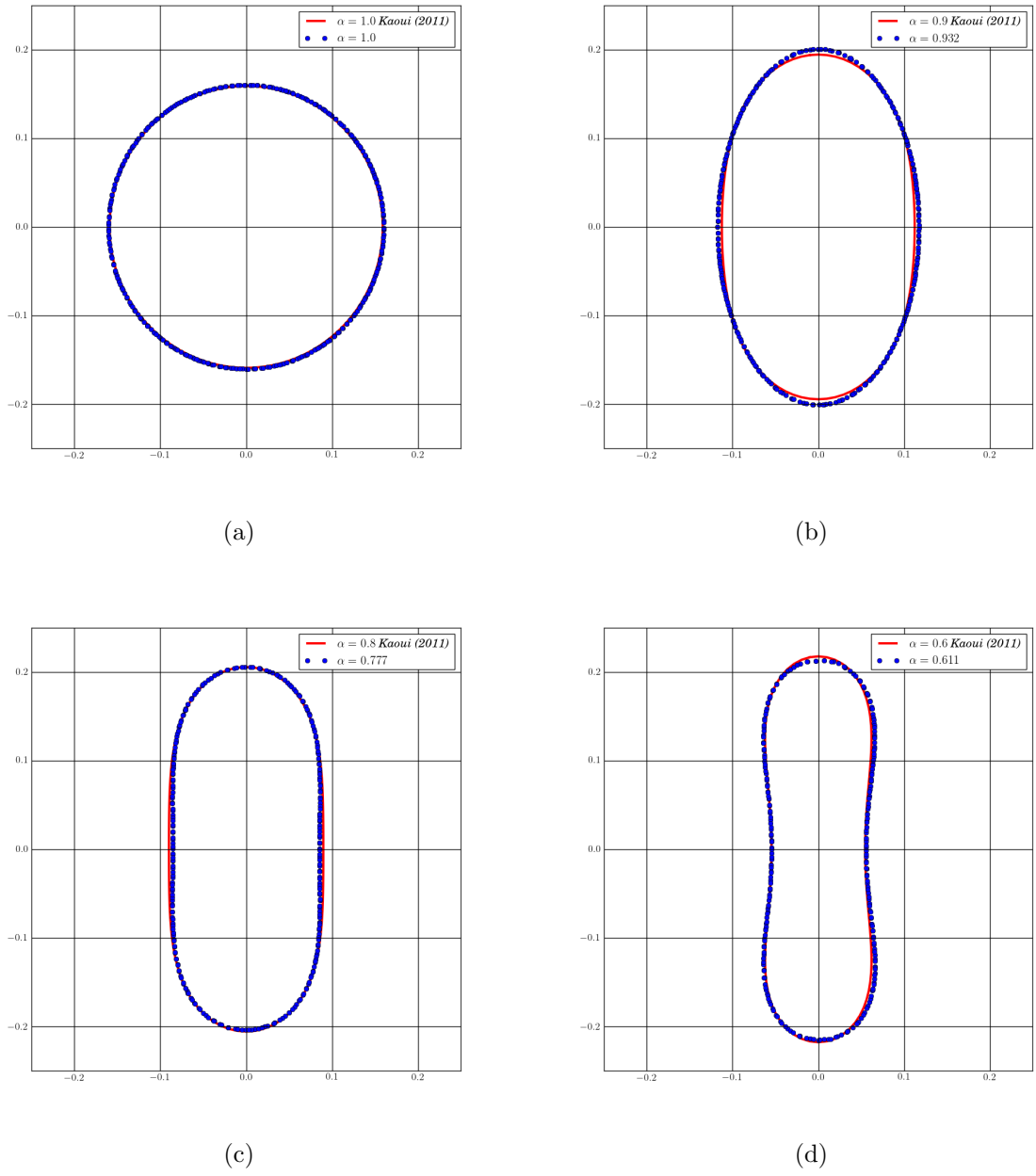
The behavior of a vesicle which has been the most studied and therefore the one which is the most understood is probably the dynamics of a single vesicle in a shear flow. A shear flow is a simple kind of flow in which the velocity has only the x component different from zero, and this component is a linear value of the y position. The coefficient linking the y position and the x velocity is called the *shear rate* (γ) and has the dimension of the inverse of a time. To summarize, in 2d:

$$\boldsymbol{u}_{\text{shear}} = \gamma \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \boldsymbol{x}. \tag{4.28}$$

It has been shown by the theory of Keller and Skalak [99] that when an ellipsoidal particle like a vesicle is placed at the center of a shear flow, it undergoes a kind of solid rotation if the viscosity of the internal fluid is high enough compared to the external one. This motion is called the *tumbling motion* (see figure 4.9). If the viscosity ratio decreases below a critical value λ_c, the vesicle takes a steady angle and the membrane of the vesicle rotates like a tank-tread. Thus, this behavior has been called the *tank-treading* motion (see figure 4.10). The authors have shown that the critical viscosity ratio depends on the reduced volume α. Later on, this transition between tank-treading and tumbling has been studied numerically in detail [7, 5]. Experimentally, the tank-treading motion has been obtained on vesicles by Kantsler et al. in [52]. The transition to tumbling has also been studied experimentally by Mader and Podgorski [69]. A state transition between the tank-treading and the tumbling has been found at the same time by Misbah et al. [77] and Kantsler et al. [52], it has been called the *vacillating breathing* or the *swinging* mode. In this motion, the vesicle's angle oscillates around its equilibrium value. The angle the a vesicle takes in a shear flow is defined as the angle between the horizontal line and the larger axis of the vesicle. Numerical investigations has been made to clarify the role of the different parameters and the associated regime in which the vesicle should be [58] and quantitative details about the phase diagram (represented schematically in figure 4.11) have been found. In [53], the effect of the confinement on the tank-treading regime has also been studied. The goal of the following tests will be to reproduce the tank-treading and tumbling motions to be sure that we are able to capture this physics with our framework.

Tank-Treading motion of a vesicle with inextensibility force

The first test that we have done is getting the tank-treading motion using the method presented in section 4.2. In this method, the inextensibility is obtained thanks to a force applied on the membrane and depending on the stretching of the level set field (recorded by a field e).



(a) (b)

(c) (d)

Figure 4.8: Equilibrium shapes of vesicles for different reduced areas. The blue points are our simulations, whereas the red lines are the simulation taken from [53] done by Lattice Boltzmann method. The exact reduced area is given as legend on the graphs. The difference between the two models are mainly due to the change of area in level set method.

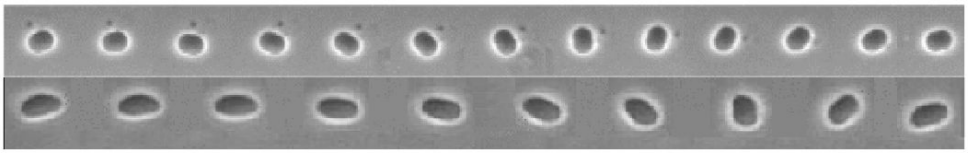


Figure 4.9: Experimental observation of the tumbling motion taken from [69]. The vesicle is doing a solid like rotation around its center.

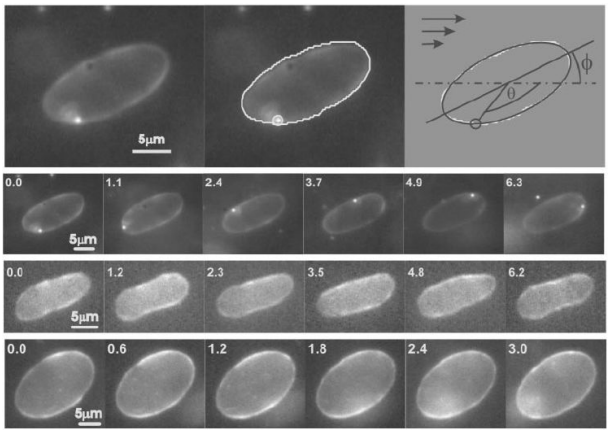


Figure 4.10: Experimental observation of the tank-treading motion taken from.

For this simulation we have set $\gamma = 1$ and the viscosity ratio $\lambda_v = 1$. A reduced area of $\alpha = 0.9$ is chosen. The force ratio is taken to $R_\tau = 5 \times 10^{-4}$ which is small enough to get a correct incompressibility. The time step is taken as $\Delta t = 3 \times 10^{-3}$ and the simulation is run on 10^4 elements. The finite element spaces in which are discretized the variables $(\boldsymbol{u}, p, \phi)$ are taken as $\mathbb{U}_h^k \times \mathbb{P}_h^l \times \mathbb{Q}_h^m$ where (k, l, m) are the polynomial orders of the discretization.

As we explained previously in section 2.1.1, for stability reasons we use the Taylor-Hood finite element, thus we always set $k = l + 1$. Moreover, we chose to discretize the stretching field e on the same space as the level set field, thus we have $e \in \mathbb{Q}_h^m$.

This simulation is run in sequential mode, thus the Fast Marching method was well suited to reset the level set to a distance function. The reinitialization procedure has been made every 5 iterations. Figure 4.12 represents the simulation at different times. In the figures 4.12(a) and 4.12(b), the vesicle is not yet at its equilibrium angle. Thus it undergoes a rotation centered on the vesicle's center. In figures 4.12(c) and 4.12(d), the vesicle has reached its equilibrium angle and the streamlines are taking the shape of the vesicle membrane, thus, the membrane is rotating as expected for a tank-treading.

The same simulation has been performed with several polynomial discretization sets to search for a good compromise between computational time and precision. The monitored property of a physical interest is the angle that the vesicle takes. Thus, this quantity is always measured and we see how the discretization influences it. The other property monitored is the perimeter of the vesicle. Indeed, the inextensibility force applied at the vesicle membrane is supposed to keep this value constant. Consequently, the choice of the approximation is made to have this constraint satisfied as well as possible. We monitor

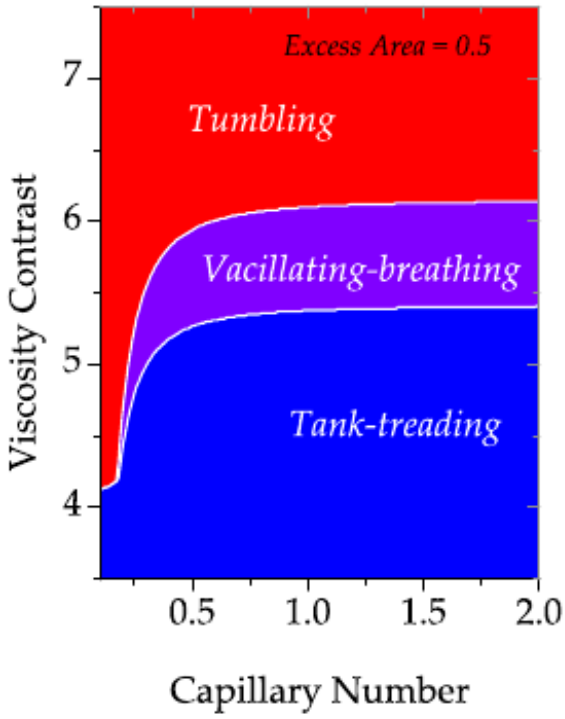


Figure 4.11: Scheme taken from [58] representing the phase diagram of the vesicle dynamics under shear flow. The y axis represents the viscosity ratio and the x axis the capillary number

the percentage of loss of perimeter defined as:

$$\Delta p_{\text{er}} = 100 \times \frac{p_{\text{er}} - p_{\text{er}0}}{p_{\text{er}0}} \tag{4.29}$$

where $p_{\text{er}0}$ is the perimeter at the initial time.

We first checked that changing the polynomial approximation of the fluid velocity and pressure does change neither the perimeter conservation nor the measure of the angle. Indeed, the perimeter conservation is mainly a consequence of the stretching force. By looking at the expression (4.15) we see that the force is the sum of two terms involving the first derivative of the level set field ($\boldsymbol{n} = \frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}$) and the first derivative of the *stretching* $(e-1)$.

When the polynomial order of discretization of the level set field and the stretching field is taken as $m = 1$, the derivatives of ϕ and e are discontinuous piecewise constants. Thus, even by projecting on a $\mathbb{P}^1$ space, the inextensibility force carries only this information. This is in fact the biggest limitation of this model for the inextensibility, and the approximation error on the velocity and the pressure are dominated by the error brought by the inextensibility force. Figure 4.13(a) shows the percentage of loss of perimeter as a function of time and figure 4.13(b) the angle dependency as a function of time for two different polynomial approximations for the fluid being $(k = 2, l = 1)$ and $(k = 4, l = 3)$.

We see clearly that the increase of the polynomial approximation has almost no effect on the loss of perimeter and on the evolution of the angle during time. By contrast, figure

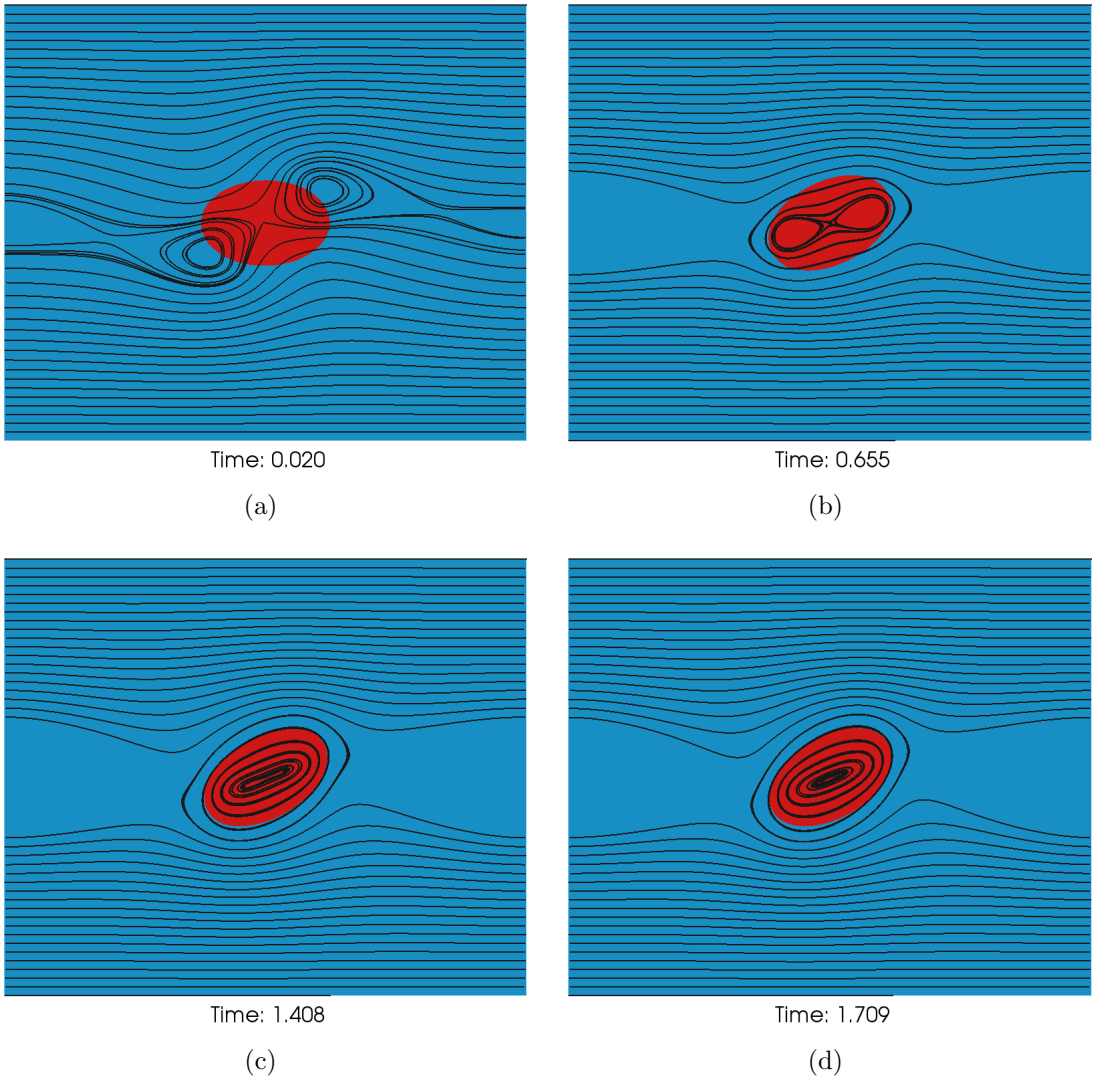
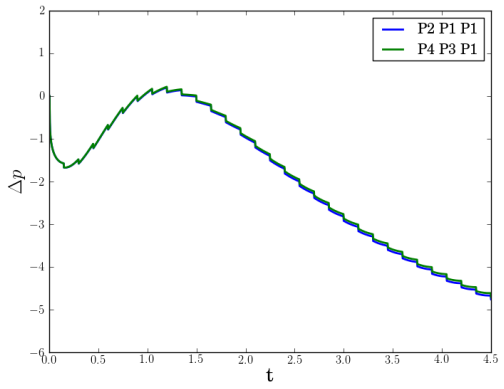
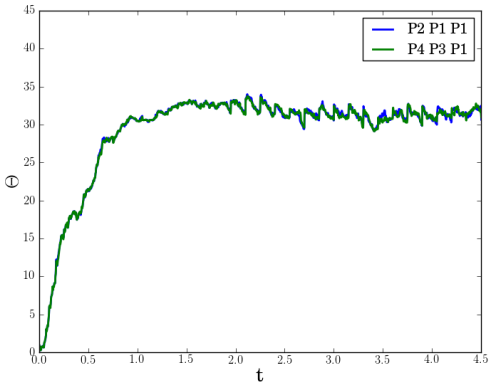


Figure 4.12: Vesicle in shear flow under tank-treading regime. The inside fluid of the vesicle (where $\phi < 0$) is represented in red, whereas the outside fluid ($\phi > 0$) is represented in blue. The streamlines are represented as black lines.



(a) Percentage of loss of perimeter. No effect is seen by increasing the fluid polynomial approximation.



(b) Angle evolution in time.

Figure 4.13: Loss of perimeter and angle as a function of time for different polynomial approximation for the fluid: $\mathbb{U}_h^2 \times \mathbb{P}_h^1$ and $\mathbb{U}_h^4 \times \mathbb{P}_h^3$.

4.14(a) and 4.14(b) show the same simulation in which only the level set (and stretching) polynomial approximation has been raised from 1 to 3. We see in figure 4.14(a) that the loss of perimeter is greatly improved by increasing the polynomial order of the level set from 1 to 2. Indeed, this makes the inextensibility force carry the information of a $\mathbb{P}^1$ continuous field which improves a lot the perimeter conservation (gain of almost 2%) for this particular simulation. Increasing again the approximation ameliorates slightly the perimeter conservation but the gain is small.

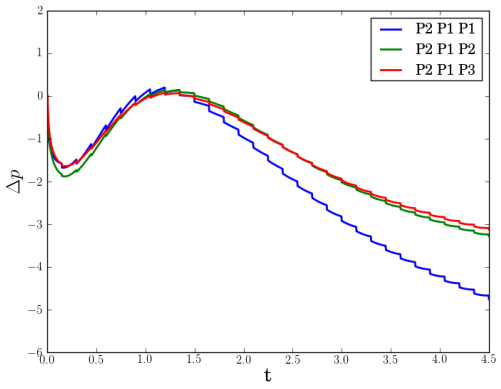
The angle dependency in figure 4.14(b) is not greatly affected by the change of polynomial approximation. This could be understood by the fact that the vesicle perimeter decreases almost symmetrically. Thus, the overall shape of the vesicle is almost the same and the angle with respect to the horizontal does not change much even if the global perimeter decreases. Of course, for a long time simulation this loss of perimeter would be critical.

Let us discuss the quality of this method. Even for the better approximation, the loss of perimeter is of the order of 3% for a simulation time which is not so long. This is a relatively bad result.

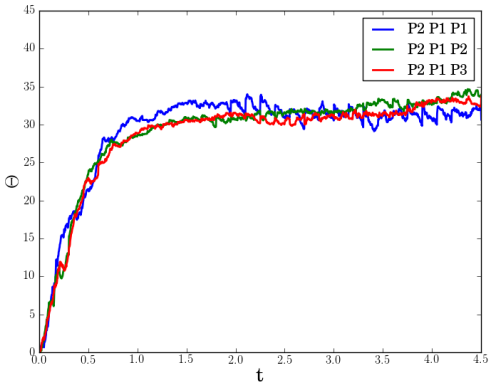
Moreover, we see some instabilities appearing around the vesicle if the time step is not small enough. This is because the inextensibility force has to be very strong to have good perimeter conservation. Thus, a relatively big local change in the stretching introduces a huge response of the inextensibility force. If the time step is not very small, this response introduces a local movement of the interface which possibly brings more stretching and makes the simulation completely unstable.

This stability issue makes this method unsuitable for difficult simulations of long time for now. Even if the advantage of having a vesicle completely defined only by adding a right hand side term in the fluid solver is really attractive, we did not go further than this test and the previous one with this method.

Moreover, keeping the stretching information might be difficult in the context of the simulation of a suspension of vesicles. When two particles get very close to each other, the level set has to be reinitialized often to avoid the merging of the two interfaces. In the



(a) Percentage of loss of perimeter.



(b) Angle of the vesicle in time. The effect on the angle dependency in time is not significant.

Figure 4.14: Loss of perimeter and angle as a function of time for different polynomial approximation of the level set and stretching fields: $\mathbb{Q}_h^1$, $\mathbb{Q}_h^2$ and $\mathbb{Q}_h^3$.

context of inextensibility forces many mesh elements between two vesicles are needed to do the computation of the stretching and the force, and to distribute the force on several elements.

In the future, this method might be improved to overcome those issues, maybe with a different formulation of the force or a mesh adaptation making the local approximation of the force very precise.

Tank-treading motion of a vesicle with inextensibility imposed by Lagrange multiplier

We will see that the Lagrange multiplier method gives better results and brings a better stability for this simulation. We take the model described in section 4.3 and put a vesicle in the conditions for a tank-treading motion.

The viscosity ratio is set to $\lambda_v = 1$. The vesicle is in the center of a box where the top and bottom walls are moving in opposite directions at speed $U = 1$. On the lateral walls, a linear velocity is imposed. The velocities of the walls are imposed by strong Dirichlet boundary conditions. The vesicles are chosen to have a fixed area $A = \pi^2$. The capillary number defined as $C_a = \dfrac{\mu U L^2}{k_B}$ is equal to π. The quantities $(\boldsymbol{u}, p, \lambda, \phi)$ are discretized on finite element spaces $\mathbb{U}_h^{k+1} \times \mathbb{P}_h^k \times \mathbb{L}_h^k \times \mathbb{Q}_h^k$, where we choose the Lagrange multiplier and the level set field to be discretized on a finite element basis of the same order as the pressure field. For this simulation, we have set $k = 1$.

We show a first simulation in which we set a box of $[30 \times 8]^2$, the mesh size is fixed at $h_{\min} = 0.06$ in a box centered on the vesicle of size $[1 \times 7]^2$ and $h_{\max} = 0.4$ elsewhere. This mesh has 36144 elements. The simulations are run in parallel on 5 processors.

A picture of the computational domain, the associated mesh and the mesh partitions is represented in figure 4.15. The time step is taken as $dt = 0.008$ and the final time is set to 40 which makes this simulation relatively long.

Since the simulation is run in parallel and since the Fast Marching method is not a

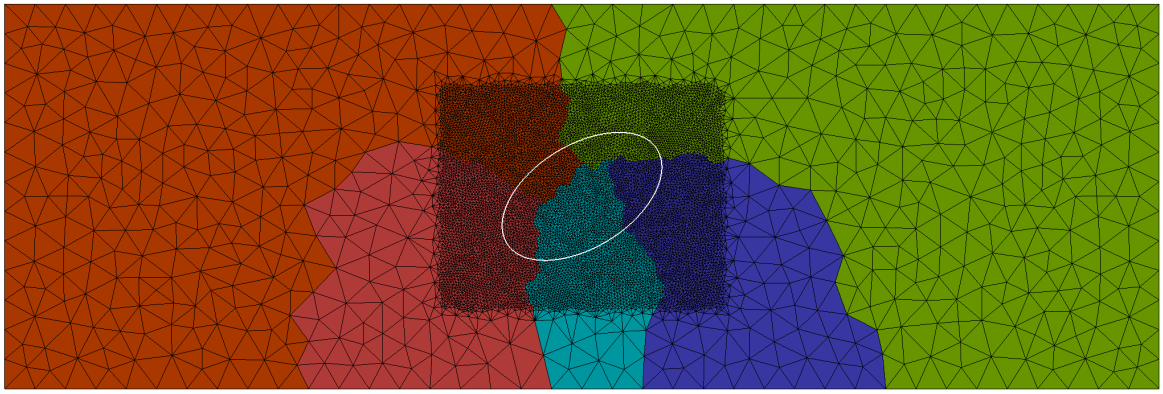


Figure 4.15: A vesicle in tank-treading simulation in the computational domain. The mesh has been added with its refinement around the vesicle. The sub domains associated to different processors are represented with different colors.

parallel algorithm, the reinitialization procedure has been made by solving the Hamilton Jacobi equation as explained in section 1.3.3. The reinitialization procedure is made every 5 iterations.

As expected, the vesicle rotates to attain a steady angle and a steady shape in which the curvature forces (shown in figure 4.16) and the hydrodynamic one are equilibrated. The tank-treading motion is obtained as we can see in figure 4.19(a) in which the stream-

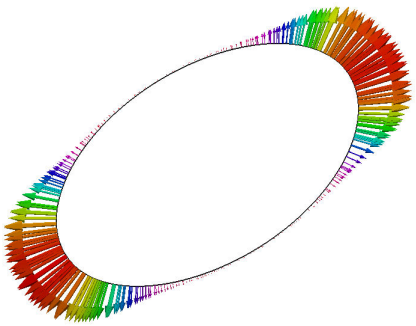


Figure 4.16: Curvature forces exerted on the vesicle at steady state.

lines are represented as black lines.

One can see that the streamlines are along the membrane, which implies that the membrane (as well as the internal fluid) is rotating. The velocity of the fluid is almost zero at the middle abscissa of the channel, but the continuity of the velocity imposes the streamlines to close.

On figure 4.19(b) we have plotted the magnitude of the perturbation field induced by the presence of the vesicle. This magnitude is defined by $|\boldsymbol{u} - \boldsymbol{u}_{\text{shear}}|$, where $\boldsymbol{u}_{\text{shear}}$ is the shear velocity as defined in equation (4.28).

We can see clearly on this figure that the disturbance of the velocity field is local and

that after two or three vesicle sizes on the lateral sides it has completely vanished. The disturbance is higher on the top right and bottom left sides of the vesicle. This can be understood by the fact that at these locations, the vesicle causes the fluid to go in the opposite direction that it would go in a pure shear flow.

At the top and bottom of the vesicle, one sees a blue spot where the perturbation flow seems to quickly vary. This is a numerical artifact due to the fact that the mesh becomes rough in this area.

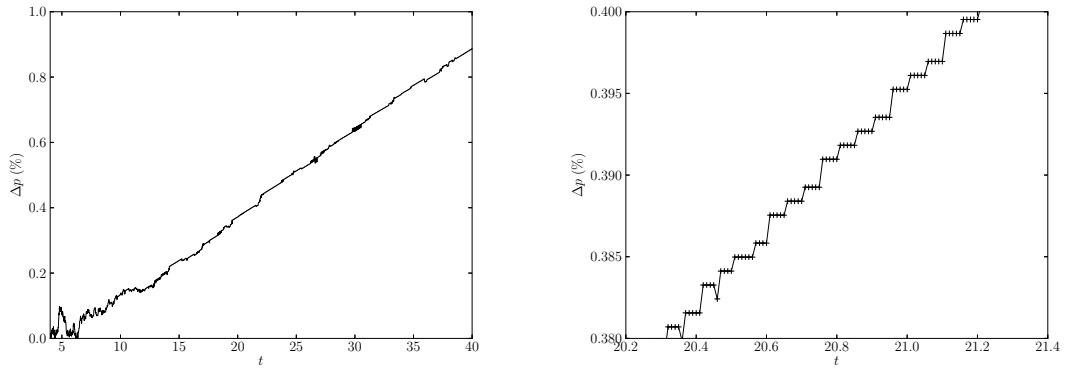
Figure 4.19(c) shows the pressure field around the vesicle. We recall that a zero mean pressure is imposed to fix the pressure constant which is defined only by its gradient in Stokes flow with Dirichlet boundary conditions.

We see that the vesicle is compressed on its lateral sides whereas it is elongated along its long axis. The pressure inside the vesicle is higher than outside and is almost constant.

An interesting point is to look at the high variations of the pressure which arise inside the membrane area. To emphasize this, we have drawn two black lines representing the area in which the delta function is higher than 0.1.

Inside this region, the Lagrange multiplier is defined and the curvature force is applied. These additional terms to the Stokes equation make the pressure vary significantly in that small region. For this particular case, we wanted to show explicitly the variations, thus an interface thickness of 6 mesh elements has been chosen. In general, only 3 or 4 mesh elements are sufficient to distribute correctly the forces and the Lagrange multiplier.

The perimeter conservation is obtained with a better accuracy than in the case of the simulations by inextensibility forces. The stability issue that we had with the forces does not exist since the Lagrange multiplier is distributed on a sufficient number of elements across the interface. Figures 4.17 and 4.18 show the loss of perimeter and surface as a function of time for this simulation.

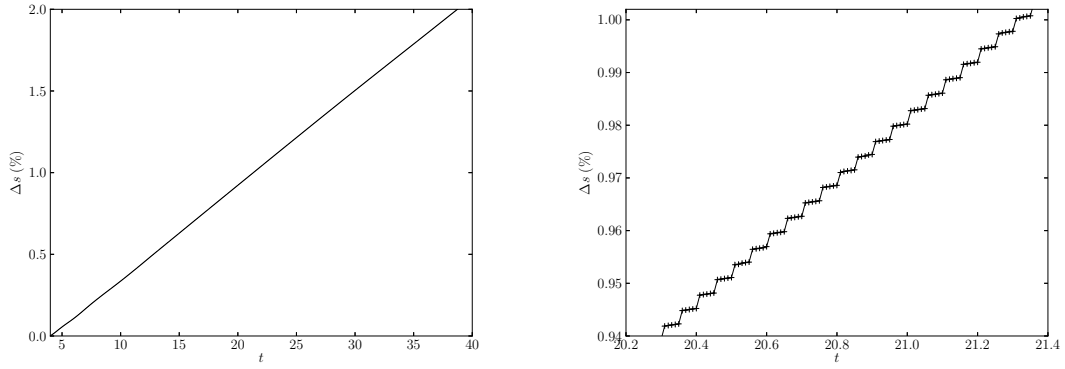


(a) Loss of perimeter (in percent) for the whole simulation.

(b) Close up of the loss of perimeter. The steps are due to the reinitialization procedure.

Figure 4.17: Perimeter conservation (in percent) as a function of time for the tank-treading simulation.

We recall that since the velocity chosen for this simulation is low $U = 1$, the shear rate is also weak $\gamma = 0.2$, and the time necessary to attain a steady state in tank-treading motion is long (4000 iterations for the whole simulation). This makes this simulation hard in the sense of perimeter and surface conservation. Thus, regarding the simulation time, the conservation of these quantities is good.



(a) Loss of surface (in percent) for the whole simulation.

(b) Close up of the loss of surface. The steps are due to the reinitialization procedure.

Figure 4.18: Surface conservation (in percent) as a function of time for the tank-treading simulation.

We also emphasize the fact that much of the loss of perimeter and surface is lost during the reinitialization procedure. Indeed, the figures 4.17(b) and 4.18(b) show some close up of the figures 4.17(a) and 4.18(a) in which we see that at each reinitialization step (every 5 iterations), the errors increase quickly compared to the error added by the simple advection without reinitialization.

The angle that the vesicle takes is measured between the long axis and the horizontal line. We call this angle Θ.

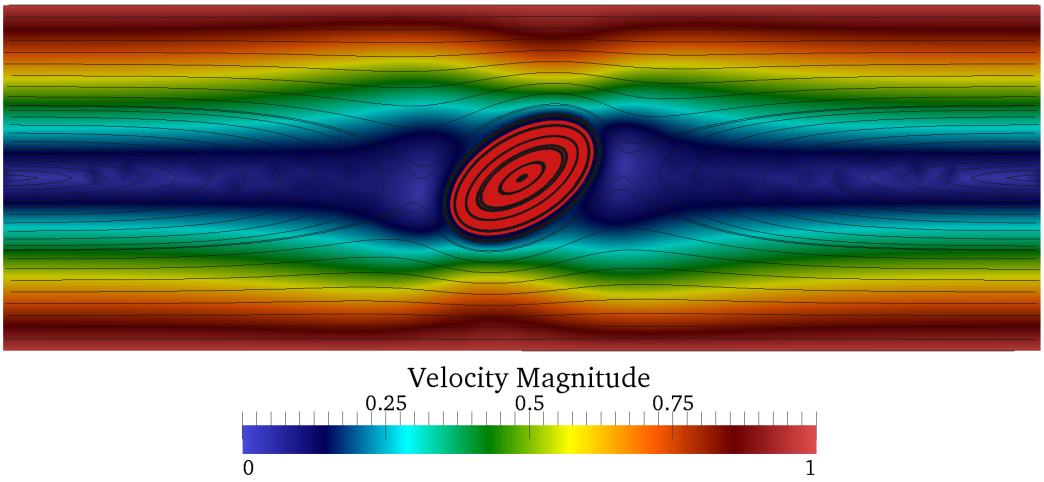
The variation of Θ as a function of time is represented in figure 4.20(a). On this graph, we have plotted the same information for different reduced areas. We can see that the steady angle decreases with the reduced area. In the limit case, where the vesicle is almost totally swollen (circular), the angle tends to the value 45° and of course, the steady state is obtained almost instantaneously.

The figure 4.20(b) shows the steady angle as a function of the reduced volume for two different confinements. We define the confinement as the ratio between twice the radius of a circle having the same area than the vesicle and the length of the channel:

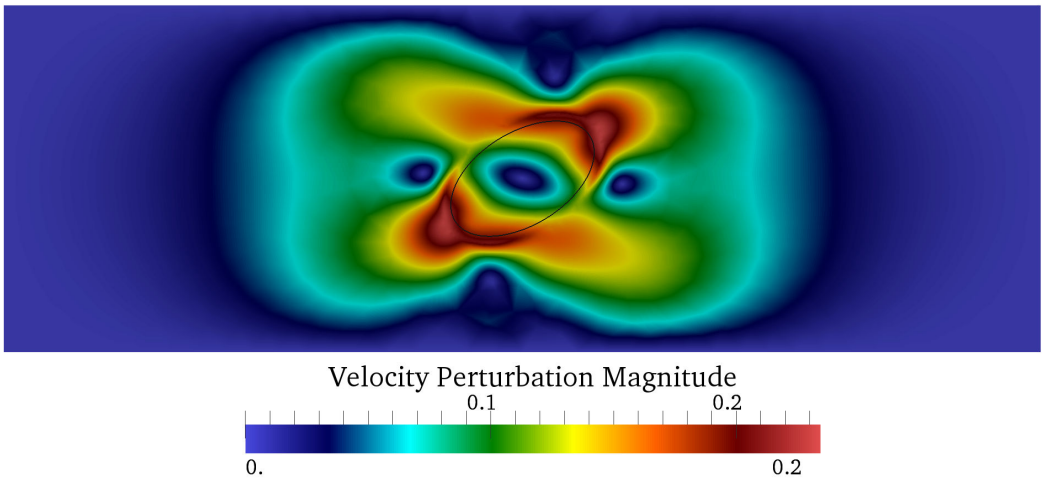
$$C_{\mathrm{n}} = \frac{2R_0}{L}. \tag{4.30}$$

In our case, we have $R_0 = \sqrt{\pi}$. We see that when the confinement increases, the steady angle decreases.

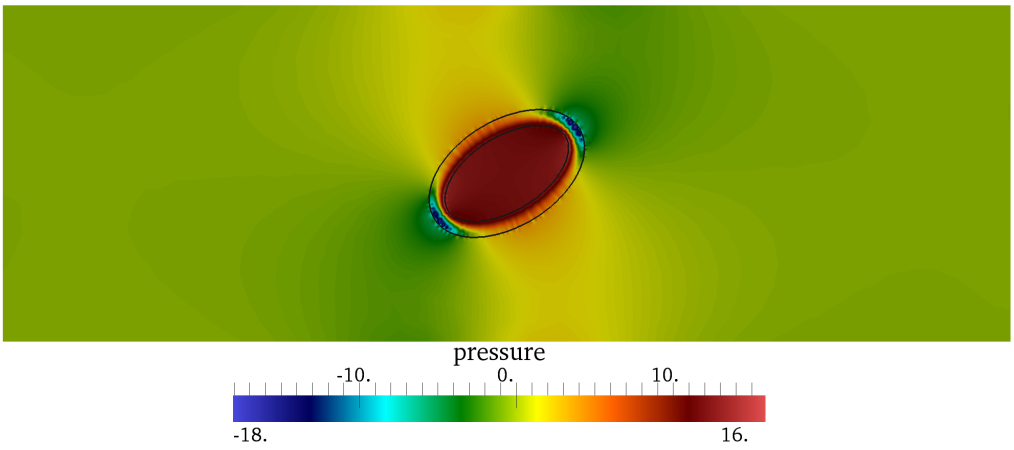
At the limit, one can imagine that when the length of the channel is really small, there is no space for the vesicle to rotate, and the angle has to go to 0. The behaviors shown in figure 4.20(a) and 4.20(b) are in good agreement with what was expected. The results are slightly different from those of [53] but it can be attributed to the difference of parameters chosen (capillary number and shear rate). However, the behavior is good.



(a) Velocity of the fluid for a vesicle in tank-treading regime. The color is the magnitude of the velocity. The streamlines are represented with black lines. The vesicle is at steady state, the streamlines are along the membrane which is rotating.

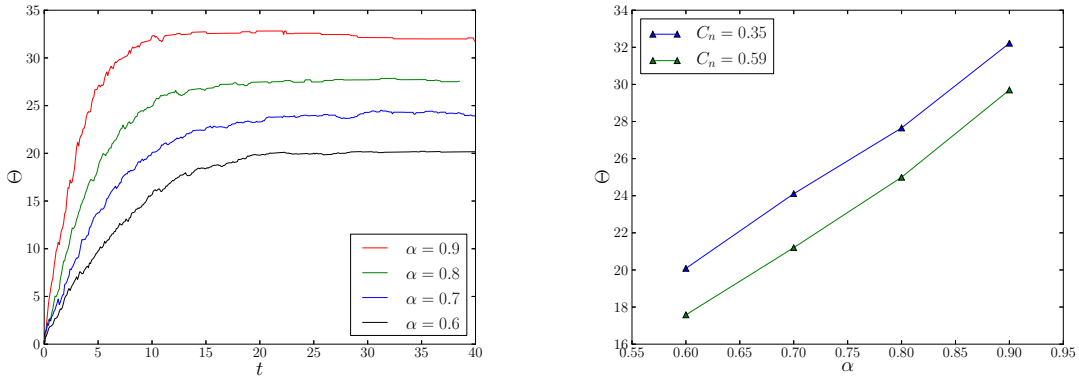


(b) Magnitude of the perturbation velocity $|\boldsymbol{u} - \boldsymbol{u}_{\text{shear}}|$. The perturbation field vanishes away from the vesicle. It is higher in the regions where the vesicle membrane forces the fluid to go in the opposite direction that the unperturbed one.



(c) Pressure field. We can see that the vesicle is elongated in the direction of its axis and compressed in the other direction. Because of the Lagrange multiplier, the pressure inside the membrane has values varying quickly. The membrane boundary are represented in black lines.

Figure 4.19: Results of the simulation of a tank-treading motion.



(a) Angle as a function of time for different reduced areas.

(b) Steady angle as a function of reduced area, for different confinements.

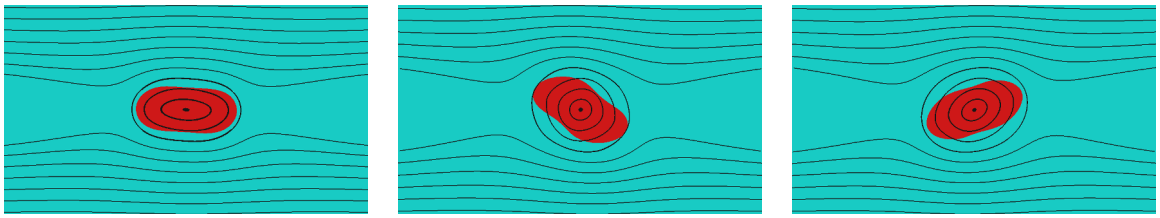
Figure 4.20: Angle behaviors with respect to different parameters.

Tumbling motion

We also wanted to check that we can obtain the tumbling motion. To reproduce this behavior, we set initially a vesicle as an ellipse in a box of size $[14, 5]^2$, discretized with 7218 elements. We chose a time step of $\delta t = 3 \times 10^{-2}$. These parameters are not as refined as in the previous section in order to get long time simulations in a reasonable computational time. The other parameters of the simulation were: $C_a = 6 \times 10^{-2}$, $C_\mathrm{n} = 0.25$ and $\alpha = 0.8$. Figure 4.21 shows 3 snapshots of a vesicle tumbling motion.

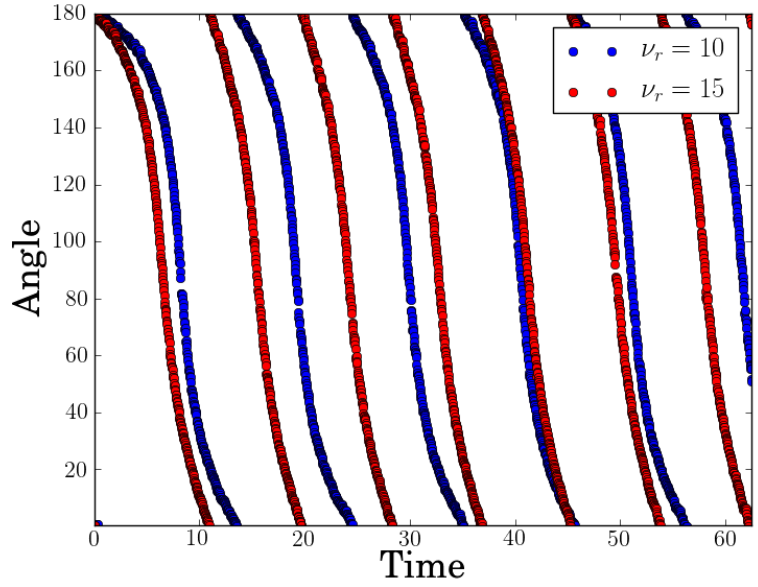
Increasing the viscosity ratio increases the rotation frequency of the tumbling as can be seen in figures 4.22(a) and 4.22(b). Moreover, as described in [37], when one increases the viscosity ratio, the rotation frequency of the vesicle reaches a steady value which corresponds to the steady rotation of a solid object in a shear flow. We can see this phenomenon in figure 4.22(b).

We can understand the fact that the rotation frequency decreases with decreasing viscosity as the *egg theorem*: take a raw egg and a cooked egg. Make them rotate with the same torque, the cooked egg rotates faster than the other one, because the different layers of fluid inside the raw egg are sheared and dissipate some energy.

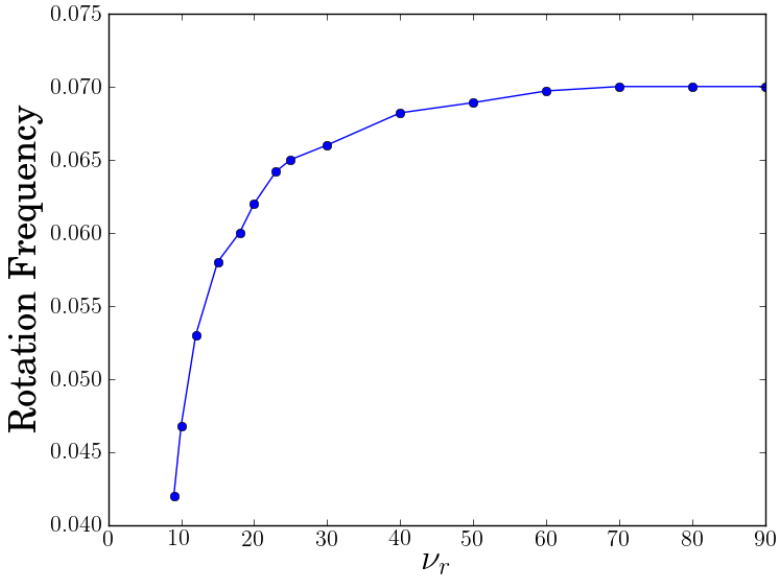


(a) $t = 1.0$ (b) $t = 1.1$ (c) $t = 6.0$

Figure 4.21: Tumbling of a vesicle



(a) Tumbling angle as a function of time for two different viscosity ratios. One can see that the higher viscosity ratio exibits a higher rotation frequency.



(b) Tumbling frequency for different viscosity ratios. The frequency increases with the viscosity ratio. It reaches the limit of a solid object rotation.

Figure 4.22: Tumbling angle and frequency.

Part II

Implementation

Chapter 5

Contributions to the library Feel++

Contents

In this chapter, we will present some development made for the vesicle application but which can be useful in other ones. Thus, a special care has to be taken to make these applications as generic and re-usable as possible. We first present a projection tool which has been of great use in writing the bending forces in the vesicle context but which can be used in many applications. Then, we present the framework for solving the advection equation which is used to transport the level set function and reinitialize it by solving the Hamilton Jacobi equation previously presented. We explain in this chapter how the C++ templates can be used to make a class solving this equation for any coefficient expression. Then the LevelSet class is presented and the MultiLevelSet class derived from it is also introduced.

5.1 Projection operator as a C++ class

5.1.1 Motivation

In Feel++ , a projection operator exists. It is a useful tool to do nodal projections. The function is a part of the language, its arguments are the following:

```
project(_space=<nodal function space in which the interpolant lives>
        _range=<domain range iterators>,
        _expr=<expression to be interpolated>, .... )
```

Note that this function uses the [1]boost parameter library which allows passing of arguments independently and in any order by name (prefixed with an underscore). This function calculates the expression given in _expr at every quadrature point in the range _range which is a part of the domain on which is defined the space _space. The resulting function is thus exact at quadrature points of _range. We have seen in section 4.1 that the vesicle application requires L^2 and *smooth* projections rather than nodal projections. These projections were already supported in Feel++ since they require solving a simple PDE problem. But each application requiring it had to create its own projector framework by providing the bilinear forms of the projection and solving the final problem. Since the application we were interested in required many of these projections, we created a C++ class which was able to handle all the types of projections ($L^2, H^1,$ smooth, CIP) needed for this application. Then, other types of projection needed in other applications have been added to complete this tool like $H^{\text{div}}, H^{\text{curl}}$ or the LIFT operator. The advantage to have a single model object for all these projections is that they are all handled the same way, the declarations of these objects are the same and only a parameter has to be given to differentiate the projector type. The `project` function itself takes exactly the same expression for any projector which brings more clarity in the different codes. Moreover, some optimization can be done (like keeping the bilinear form as it will be explained below) and the optimizations are completely hidden in the projector class, making the optimization systematic. Finally, the bilinear form projections are written and tested once and for all and there is no need to re-write them for each application.
The basic idea of this class is to construct `projector` objects. Each object can do a given projection. For the construction of such an object, one only need the type of projection, and the space in which the projection is done. Providing this information, one can create the matrix associated to the bilinear form (the left hand side of the projection equation)

[1] http://www.boost.org/doc/libs/

since the entries in the matrix depend only on the trial and test basis functions. Then a function `project`, for which the user provides the right hand side, solves the projection equation and returns the projected function. The matrix has only to be created once at the construction of the `projection` object and can be reused as many times as needed. All the matrices are completely hidden for the user of the class, and only the constructor and the `project` function are seen from the user point of view, making these projections almost as simple as the nodal projection.

5.1.2 Usage of the Projector class

At the construction of a new projector object, a bilinear form is created. Once the bilinear operator is created, its associated matrix is stored in memory and kept until the operator is destroyed. This form depends on the type of projection needed. The bilinear forms associated to each projection are reported in appendix E. As an example, the bilinear form associated to the L^2 projection is:

$$a_{L^2} = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{v}$$

where $\boldsymbol{u} \in [L^2(\Omega)^2]^d$ is the trial function and $\boldsymbol{v} \in [L^2(\Omega)^2]^d$ is the test function. This bilinear form takes the Feel++ expression:

```
a = integrate( elements( mesh ), trans( idt(u) ) * id(v) );
```

where idt represents the trial test functions and id the test functions.

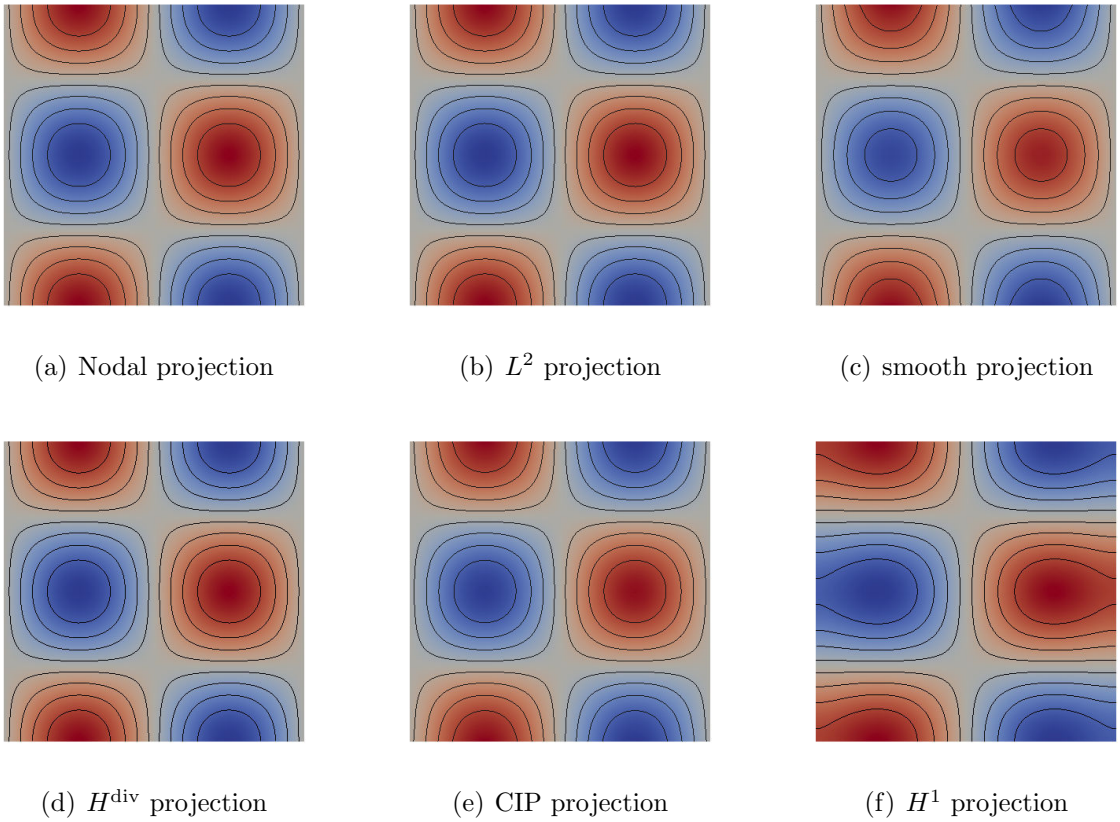
Then only the expression of the right hand side is needed and the system can be solved. Consequently, the use of the projector class is very simple. First, one has to declare one projector object for each projection wanted. One only needs to give the functional space on which is done the projection, a backend used to solve the problem, and if they are needed, the parameters for the projection. Second, the projection is done using simply the `project` function. We show in listing 5.1 an example of code creating two projectors and using them to project the function $f = \sin(2\pi x)\cos(2\pi y)$.

Listing 5.1: Example showing how to make L^2 and smooth projectors and use them to project a simple function.

```
// defines scalar polynomial space of order 1
auto Xh = Pch<1>(mesh);

auto proj_L2 = projector(Xh, Xh, backend(), L2);
auto proj_SM = projector(Xh, Xh, backend(), DIFF, epsilon, gamma);

auto f = sin(2*pi*Px()) * cos(2*pi*Py());

auto func_L2 = proj_L2->project( f );
auto func_SM = proj_SM->project( f );
```

The figure 5.1 shows the results of the projection of f for the following projections: nodal, L^2, smooth, div, CIP, and H^1. For the details of the formulation of the projectors, see appendix E. We can notice that, there is no difference between nodal, L^2, div and CIP projectors. The smooth projection tends to *diffuse* the field. It can be seen by looking at the iso value circles on the graph which are slightly enlarged. Finally, the H^1 projection is used without imposing the boundary conditions, thus implicitly, the Neumann boundary

condition $\boldsymbol{\nabla} f \cdot \boldsymbol{n} = 0$ is imposed. Consequently, there is no gradient of the function in the normal direction close to the boundaries, this makes the iso values perpendicular to the borders of the domain as we can see in figure 5.1(f).



(a) Nodal projection (b) L^2 projection (c) smooth projection

(d) H^{div} projection (e) CIP projection (f) H^1 projection

Figure 5.1: Results of the projection of f using different projectors. The color is the amplitude of the projected function f. The deepest red is 1, the deepest blue is -1. 10 iso values are reported in black.

5.1.3 Projector class used to compute the derivative of a field

Let us consider that we want $\boldsymbol{u} \in [L^2(\Omega)]^d$ as the projection of the gradient of a known scalar field $f \in H^1(\Omega)$. We can write this expression on the weak form (5.1) which is the classical L^2 projection of the gradient of f. We can also integrate by parts this equation, which leads to the formulation (5.2). This formulation requires that the test function $\boldsymbol{v} \in [H^1(\Omega)]^d$.

$$\int_\Omega \boldsymbol{u} \cdot \boldsymbol{v} = \int_\Omega \boldsymbol{\nabla} f \cdot \boldsymbol{v} \tag{5.1}$$

$$\int_\Omega \boldsymbol{u} \cdot \boldsymbol{v} = -\int_\Omega f \nabla \cdot \boldsymbol{v} + \int_{\partial\Omega} f \; \boldsymbol{v} \cdot \boldsymbol{n} \tag{5.2}$$

The formulation (5.2) has the advantage of never computing directly the gradient of the function f since the derivatives are on the test functions. We remark also that the left hand side is exactly the same as for a classical L^2 projection. Thus by changing only

the form of the right hand side term, we can provide an operator which computes the projection of the gradient of an expression. We called such a method `derivate`.

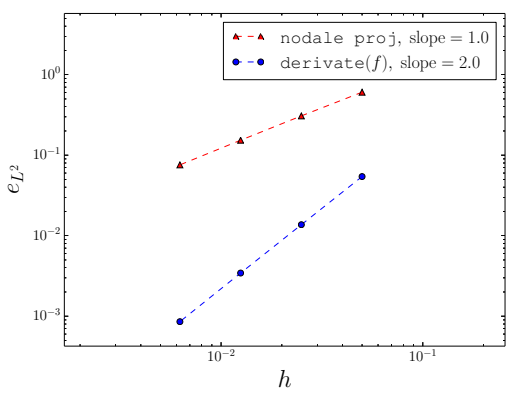
If we take the previous example, and we want the projection of $\boldsymbol{g} = \boldsymbol{\nabla} f$, we only need to create a projector in a vectorial space (since $\boldsymbol{g}$ is a vector), and use the `derivate` function on the expression f. The listing 5.2 gives an example of such a code. In this example, we also show an alternative to get the derivative of a field, by projecting the expression on the vectorial space, and doing a nodal projection of its gradient on the same space.

Listing 5.2: Create a projector and compute the projection of $\boldsymbol{\nabla} f$

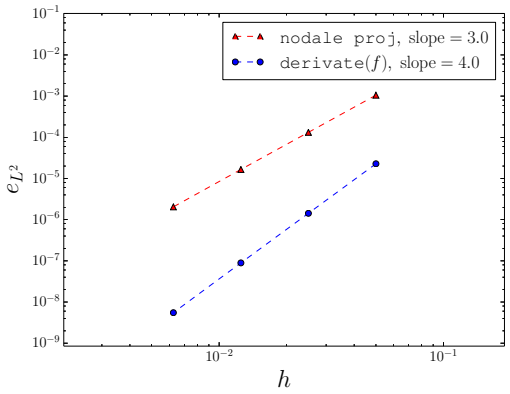
```
// Vectorial functionspace
const int order = 1;
auto Xhv = Pchv<order>(mesh);

// defines the expression to derive
auto f = sin(2*pi*Px()) * cos(2*pi*Py());

// compute the derivative using the projector class
auto proj_L2_vec = projector(Xhv, Xhv, backend(), L2);
auto g = proj_L2_vec->derivate( f );

// compute the derivative of a nodal projection of f
auto f_nodal = project(Xhv, elements(mesh), f );
auto g_nodal = project(Xhv, elements(mesh), gradv( f_nodal ) );
```

We made this test by taking a discretization of f as polynomials of order 1 and 3 and $\boldsymbol{u}$ has been taken in vectorial spaces of the same polynomial order.

The results of the nodal projection have been plotted in figure 5.3 and the result of the projection using the method `derivate` has been plotted in figure 5.4. Since the theoretical result is known, the L^2 error has been computed. We can see on figures 5.2(a) and 5.2(b) that we obtain one order of convergence higher using the `derivate` method compared to the nodal projection of $\boldsymbol{\nabla} f$ on the same space.

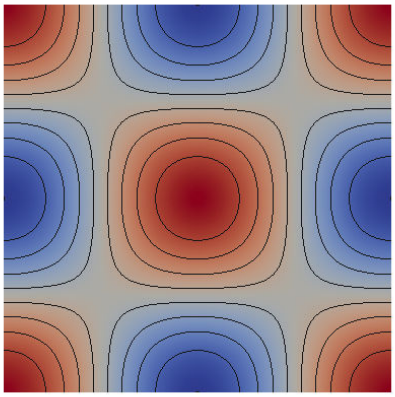


(a) f and $\boldsymbol{u}$ taken as polynomials of order 1

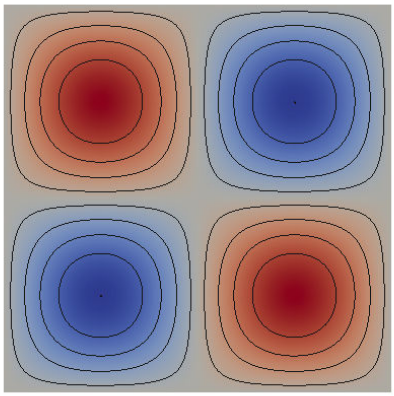


(b) f and $\boldsymbol{u}$ taken as polynomials of order 3

Figure 5.2: Comparison between the derivative using the `derivate` method and a nodal projection of the derivative.

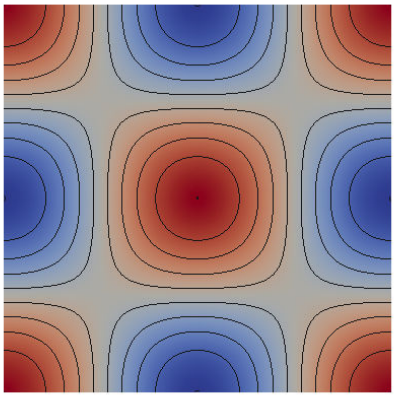


(a) X component of ∇f

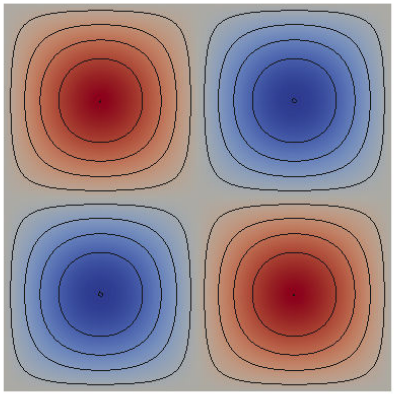


(b) Y component of ∇f

Figure 5.3: Gradient of f computed using a nodal projection.



(a) X component of $\boldsymbol{g}$



(b) Y component of $\boldsymbol{g}$

Figure 5.4: Gradient of f computed using 5.2.

5.2 The Advection class

Solving the advection equation (1.12) is a key point for this work. Indeed, it is used every time step for the evolution of the level set field that we have seen in section 1.2 and it is also used to reinitialize the level set front in the context of a reinitialization by solving a Hamilton-Jacobi equation as seen in section 1.3.3. Thus, the need of a general framework to solve this equation arises. A C++ class has been created during the PhD of Christophe Winkelmann [112]; we updated this to fit our needs and we added the stabilization methods presented in section 1.2.2. We will present in this section some aspects of the implementation of this class.

5.2.1 Description of the class

A simplified view of the Advection class is given in listing 5.3

Listing 5.3: Simplified view of the Advection class.

```
public:
    enum stabilizationMethod {NO, GALS, CIP, SUPG, SGS};

    Advection(space_type space, stabilizationMethod sMeth );

    // update the coefficients to solve the equation:
    // σφ + β·∇φ = f in Ω
    // φ = g on ∂Ω
    template <typename SigmaT, typename BetaT, typename FT, typename GT>
    void update(SigmaT sigma, BetaT beta, FT f, GT g);

    element_type phi();

private:
    sparse_matrix D;
    vector_ptrtype F;

    stabilizationMethod stabMethod;
```

The matrix D associated to the discretization of the equation and the right hand side F are created at the construction of the object once and for all. Of course, the matrix has to be reassembled at each resolution of the equation, but for a given problem, the test and trial functions of the problem do not change at each new resolution. Thus, creating the non null entries of the matrix once and for all is an optimization. From the user point of view, when solving the equation (1.12), one only has to give the values of $\sigma, \boldsymbol{\beta}, f$ and some Dirichlet boundary conditions g which are applied at the inflow boundaries. The method `update` is meant for that. The types of $\sigma, \boldsymbol{\beta}, f, g$ might be complicated. Indeed, we want to allow the possibility for the user to enter expressions for these values and not simply element of the space. The type of an expression in Feel++ depends on the expression. Thus, it is impossible to write explicitly the types of these parameters. It is not a problem since C++ handles meta-programming thanks to the template feature. Thus, the `update` function is a template function and the `SigmaT, BetaT, FT, GT` which are the types of respectively $\sigma, \boldsymbol{\beta}, f, g$ are computed automatically at compilation time depending on the expression that have been given to the function. In the `update` function, the matrix D and the vector F are assembled according to the variational formulation (1.14) as one can see on listing 5.4.

Listing 5.4: Assembly of the matrix D and the vector F in the `update` function

```
// trial function φ
auto phi = space->element();
// test function ψ
auto psi = space->element();

// ∫_Ω fψ
form1(space, F)=integrate(elements(mesh), f * id(psi)) ;

// ∫_Ω (σφ + β·∇φ)ψ
form2(space, space, D)=integrate(elements(mesh),
      (sigma * idt(phi) + gradt(phi) * beta) * id(psi));
```

As one can see on listing 5.4, there are obviously some restrictions on the types of the parameters given to this method. But the restrictions are more related to the mathematical objects that represent the parameters $\sigma, \boldsymbol{\beta}, f$ than to the C++ types. More precisely, the method can work since σ and f are of a type representing a scalar function or a scalar number, and $\boldsymbol{\beta}$ a line vector. If these conditions are not respected, an error will be thrown during the compilation.

The stabilization method is chosen by the user. It has to be picked as one of the enumeration `stabilizationMethod` elements. Then the algebraic representation of the corresponding terms of equations (1.20) or (1.23) is added to the matrix D and the vector F. Listing 5.5 gives an example of such terms for the SUPG stabilization.

Listing 5.5: SUPG stabilization added to the matrix D and vector F

```
// β·∇ψ
auto supg_term = grad(psi) * beta;

// τ^SUPG
auto tau_supg = h() / (2 * sqrt(trans(beta)*beta) );

// L(φ)
auto L_op = gradt(phi) * beta + sigma * idt(phi);
            form2(space, space, D)+=integrate(elements(mesh),
                  tau_supg * supg_term * L_op );

            form1(space, F)+=integrate(elements(mesh),
                  tau_supg * supg_term * f);
```

Finally the method `phi` solves the equation and returns the solution.

5.2.2 Example on Hamilton-Jacobi equation for reinitialization

As an example of the use of Advection class we could give the advection equation (1.11) used for the motion of the level set field, but we chose to present the Hamilton Jacobi equation used for the reinitialization as presented in section 1.3.3. This is a bit more complicated. We have shown that the signed distance function can be obtained from a field being not too far from a distance function by solving the equation (1.26) which is an advection equation of the form (1.12) for which the coefficients are:

$$\begin{aligned}\sigma &= \frac{1}{\Delta\tau},\\ \boldsymbol{\beta} &= \frac{S_{\text{gn}}(\phi^n)\boldsymbol{\nabla}\phi^n}{|\boldsymbol{\nabla}\phi^n|},\\ f &= S_{\text{gn}}(\phi^n) + \frac{\phi^n}{\Delta\tau},\end{aligned}$$

with the same notations that we used in section 1.3.3. Given the Advection class, the implementation of the reinitialization by solving the Hamilton Jacobi equation is simple. An example is given in listing 5.6.

Listing 5.6: Hamilton Jacobi equation for the reinitialization of the level set field solved by using the Advection class.

```
// max_iter and dtau are taken from user options
// create an Advection object
advection_hj = advection_type::New( space, SUPG );

for (int i=0; i<=max_iter; ++i)
{
  // update the Heaviside function according to (1.3)
  auto H = vf::project(space, elements(mesh),
  chi( phi< -epsilon ) * 0.
  + chi( phi >= -epsilon )
  * chi( phi <=  epsilon )
  * 1/2*(1 + phi / epsilon + sin( pi*phi/epsilon ) / pi )
  + chi( phi > epsilon ) * 1. );

  // expressions of S_gn(phi), sigma, beta, f
  auto sgn =  2 * ( idv(H) - 0.5 );
  auto sigma = 1 / dtau;
  auto beta = sgn * trans(gradv(phi))
  / sqrt(gradv(_phi_reinit)*trans(gradv(_phi_reinit)) );
  auto f = idv(_phi_reinit) / dtau + signe ;

  // update the matrices D and f
  advection_hj->update(sigma, beta, f, true);

  // solve the equation and save into phi
  phi = advection_hj->phi();
}
```

5.3 Design of the Level Set class

In this section, we will explain how we structured the Level Set class. This class has been made with the goal of a maximum generality and ease re-use.

5.3.1 The public part

Let us describe how we designed the LevelSet class in order to make it as generic as possible. The core of the level set class should provide methods to initialize a level set field and advect it with respect to the advection equation since the user provides a velocity. The advection equation should also contain the possibility to reinitialize the level set field if needed. The advection and the reinitialization should be the only two ways to modify the level set field and the C++ encapsulation is used to make sure it happens this way. The class should also provide some accessors to the Dirac and Heaviside fields which should be automatically updated after each modification of the level set fields. Thus, the basic components of the public level set class are the following:

Listing 5.7: Simplified part of the core of the public part of the LevelSet class.

```
public:

/* constructor and initialization*/
LevelSet( mesh_ptrtype mesh );
```

```
void initialize( element_ptrtype phio, bool doFirstReinit );

/* advection method */
template <typename Tu> bool advect( Tu& u );

/* getters */
const element_ptrtype phi();
const element_ptrtype H();
const element_ptrtype D();
```

where the suffix `ptrtype` represents the fact that the given type is actually a smart pointer type, the [2]`boost::shared_ptr` is used. The `initialize` method is used to set the initial level set ϕ^0 to the given `phio` field. If we are not able to provide an exact distance function for `phio`, the parameter `doFirstReinit` can be set to `true` and then a reinitialization is performed. The methods `phi()`, `H()` and `D()` are accessors to the values of respectively ϕ, δ_ε, and H_ε. They could be used directly in an expression, for example if one has to define the viscosity of a two fluids system, we have seen that the viscosity field has to be defined as: $\mu_\phi = \mu_2 + (\mu_1 - \mu_2)H_\varepsilon(\phi)$. If one has an instance `levelset` of the class LevelSet , providing the values of μ_1 and μ_2, the formulation simply takes the form:

```
mu2 + ( mu1 - mu2 ) * idv( levelset->H() );
```

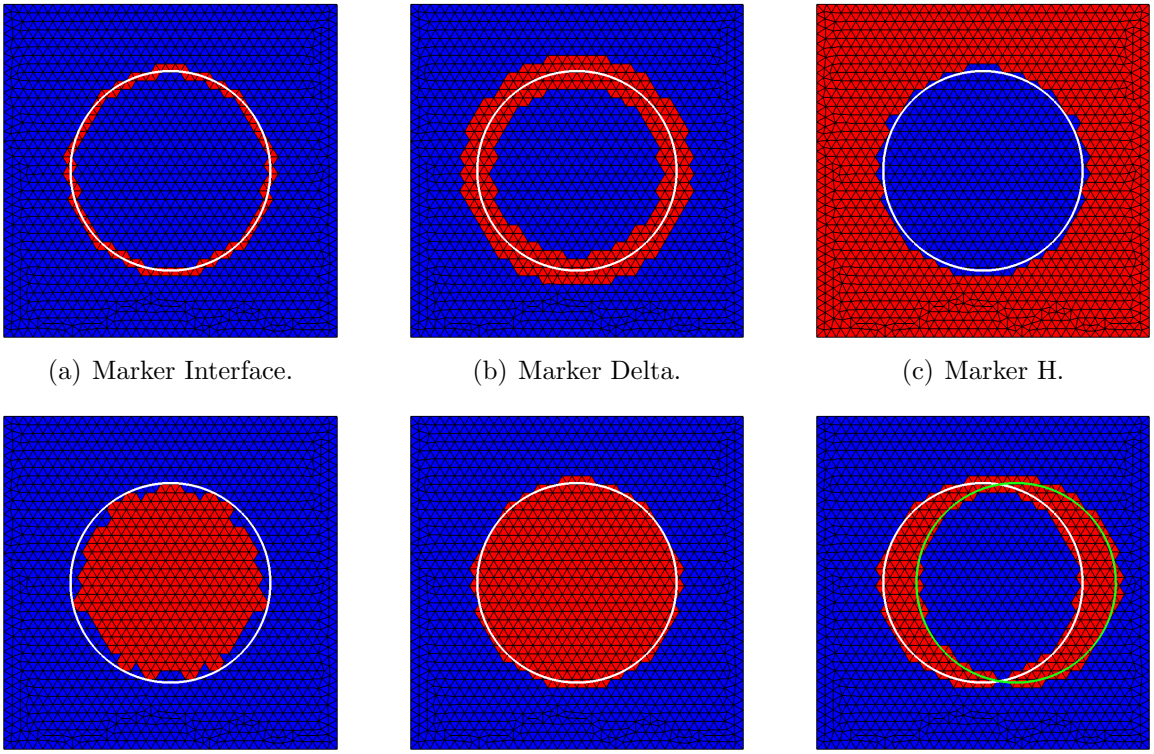
where `idv` is the Feel++ keyword to have access to the values of an element of a function space.
The `advect` method is used to advect the level set field with the given velocity `u`, check if the reinitialization is needed, do it if necessary, and update the quantities ϕ, H_ε and δ_ε. The `advect` method has to be a template method since the type of the velocity is not known a priori. The type will be deduced at the compilation time according to the type of the velocity given when the method is used. By introducing this template parameter, every velocity type for which the operator `*` is implemented can be used in this method. Indeed, only this operation is needed in the advection equation.
The type of an element of a functional space depends on its polynomial order of approximation. The `*` operator is implemented between elements of different polynomial orders. Thus, providing this template function `advect` allows coupling the level set with a velocity of an arbitrarily high order. Since in the level set framework the coupling with the fluid only appears at the level of the advection equation, making this method generic leads to a generic LevelSet class which will be easy to couple to any type of fluid velocity.

5.3.2 The markers

We also created other useful methods that we did not report in the listing 5.7, like several markers calculated from the level set field. A marker is a $\mathbb{P}^0$ element defined on all the space and having integer values at different regions of the domain. It is possible in Feel++ to do operations only on the area on which a marker has a specific value. We provide the following markers:

[2] http://www.boost.org/doc/libs/1_54_0/libs/smart_ptr/shared_ptr.htm

- Marker Interface is 1 in the region where $|\phi| < \varepsilon$, 0 elsewhere (fig 5.5(a)).
- Marker Delta is 1 in the region where $\delta_\varepsilon > 0$, and vanishes elsewhere (fig 5.5(b)).
- Marker H is 1 in the region where $H_\varepsilon > 0.999$, and vanishes elsewhere (fig 5.5(c)).
- Marker H inverted is 1 in the region where $H_\varepsilon < 0.001$, and vanishes elsewhere (fig 5.5(d)).
- Marker H In Out is 1 in the region where $H_\varepsilon < 0.5$, and vanishes elsewhere (fig 5.5(e)).
- Marker Crossed Elements is 1 only in the elements crossed by the interface between two iterations, thus, where $\phi^n \phi^{n-1} < 0$ (fig 5.5(f)).



(a) Marker Interface. (b) Marker Delta. (c) Marker H.

(d) Marker H inverted. (e) Marker H In Out. (f) Marker Crossed Elements. ϕ^{n-1} is represented in green.

Figure 5.5: Markers available in LevelSet . The elements having the value 1 are represented in red. The iso-0 line of the level set function is represented in white. The interface thickness is $2\varepsilon = 3h$.

These markers can be used as follows: let us say that we have a two-fluid flow application, and we want to create the surface tension force between the two different fluids. The surface tension needs the curvature and the normal of the interface. These quantities can be computed only in the region where the delta function is different from 0 since the force will be projected on the interface by multiplying by δ_ε. Thus, computing the normal and the curvature only on these elements could be a relevant optimization. An example of the use of markerDelta in this context is given in listing 5.8.

Listing 5.8: An example of the use of markerDelta to compute quantities only where the function δ_ε is different from 0.

```
// declare the elements on the right space
// the elements are initialized to 0
auto n = vectorialSpace->element();
auto k = scalarSpace->element();
auto F = vectorialSpace->element();

// update the marker2 of mesh with the markerDelta of levelset
mesh->updateMarker2( levelset->markerDelta() );

// make the projections only on the elements marked as 1 (where δ_ε > 0)
n += project(vectorialSpace, marked2elements(mesh, 1),
gradv(levelset->phi())
/ sqrt( gradv(levelset->phi()) * trans( gradv(levelset->phi()) )));

k += project(scalarSpace, marked2elements(mesh, 1), divv(n) );

F += project(vectorialSpace, marked2elements(mesh, 1),
sigma * idv(k) * idv(n) * idv(levelset->D()) );
```

5.3.3 The external options

Many choices of different numerical methods have to be made regarding the level set class, for example, the way to discretize and stabilize the advection equation, the choice of the reinitialization method, the reinitialization frequency and so on. Many numerical parameters have also to be set by the user. It can seem odd that these parameters never appear in the public part of the LevelSet class but the reason is simple. All these parameters are given through external options from the user to the program. The option parser used by Feel++ is the [3]boost::program_options library. It provides an easy way to handle a big number of options and recover it in the program. Thus, the LevelSet class provides a set of options which are used to monitor the behavior of each level set instance. This makes the code clear since the developer using the LevelSet class does not have to set the parameters at the time he writes the code and brings generality since one can, at execution time, try different parameters just by changing the options given to the program. The number of options can be high, but default parameters are set in order not to have to reset the parameters used almost always with the same value. Moreover, a configuration file can be written and given as options. As an example, let us say we have an application using the level set framework coupled with a fluid solver. This application is called `twoFluidFlow`. Let us say we want to use the Fast Marching method as reinitialization method every 10 iterations and we want a BDF2 time discretization for the advection equation stabilized with the SUPG method and an interface thickness of 0.01. An example set of options that one could give to this application is:

```
./twoFluidFlow --levelset.time-discr-scheme=BDF2 \
--levelset.thickness-interface=0.01 \
--levelset.advec-stab-method=SUPG \
--levelset.enable-reinit=true \
--levelset.reinitevery=10
--levelset.reinit-method=fm
```

[3] http://www.boost.org/doc/libs/1_54_0/doc/html/program_options.html

5.3.4 The protected part

The protected part of the level set function is by definition not accessible for the user of the class but its attribute are accessible from a class derived from it. These parameters and methods are important for the multi level set framework, thus, it is important to describe it. A simplified list of the protected part of the LevelSet class is given in listing 5.9

Listing 5.9: Simplified list of protected parameters and methods of the LevelSet class

```
protected:

element_ptrtype ptr_phi;
element_ptrtype ptr_h;
element_ptrtype ptr_d;

void updateHeaviside();
void updateDirac();
```

The parameters `ptr_phi`, `ptr_h` and `ptr_d` are pointers on the elements representing the levelset, Heaviside and Delta fields. The methods `updateHeaviside` and `updateDirac` are used just after the advection step to update the fields pointed by `ptr_h` and `ptr_d` according to the new value of the field pointed by `ptr_phi`.

5.4 The MultiLevelSet class

5.4.1 From LevelSet to MultiLevelSet

To handle several level set fields at the same time, one could think at first to create an array of LevelSet instances. This could work but would be dramatically costly in terms of computational time and memory. Indeed, this would copy all the internal ingredients of LevelSet such as for example the bilinear matrices for the advection. It would also copy the reinitialization framework which contains some features possibly heavy in memory. For example, the bilinear matrix for the reinitialization by solving Hamilton Jacobi equation or the table of the closest degree of freedom needed for the fast marching method.
Consequently, it is essential to think differently and try to share at maximum everything which can be constructed once and reused for all the different level set. For example, the non-stabilized part of the bilinear matrix associated to the advection equation only depends on the given velocity $\boldsymbol{u}$. Thus, in a context of multi-fluid flow, the velocity is the same for all the level sets and this matrix could be computed once and for all and reused to advect all the level set fields. The table of the first neighbors used for the fast marching reinitialization depends only on the mesh, this can also be used for all the level sets. Thus, we created a MultiLevelSet class which allows to handle an arbitrary high number of different level sets fields. The MultiLevelSet class derives from the LevelSet class. It defines in addition, a vector of elements `phis`, `hs`, and `ds` meant to contain several elements ϕ, H_ε and δ_ε. It also provides a method called `switchPhi(i)` which takes an integer `i` as argument. This method is essential. It makes the pointers `ptr_phi`, `ptr_h` and `ptr_d` that we described previously, point on the $\texttt{i}^{\text{th}}$ element of `phis`, `hs`, and `ds`. This way, the methods `advect`, `updateHeaviside` and `updateDirac` will act on the $\texttt{i}^{\text{th}}$

level set element since they act on the value pointed by `ptr_phi`, `ptr_h` and `ptr_d`. We emphasize the fact that `switchPhi(i)` only changes the value of a few pointers, thus it does not cost anything in term of performance. Thanks to this method, it is possible to do operations on a single chosen level set field without duplicating all the attributes of the LevelSet class since by inheritance, MultiLevelSet has these attributes once. A scheme representing the effect of `switchPhi(i)` is given on figure 5.6. The `advect`

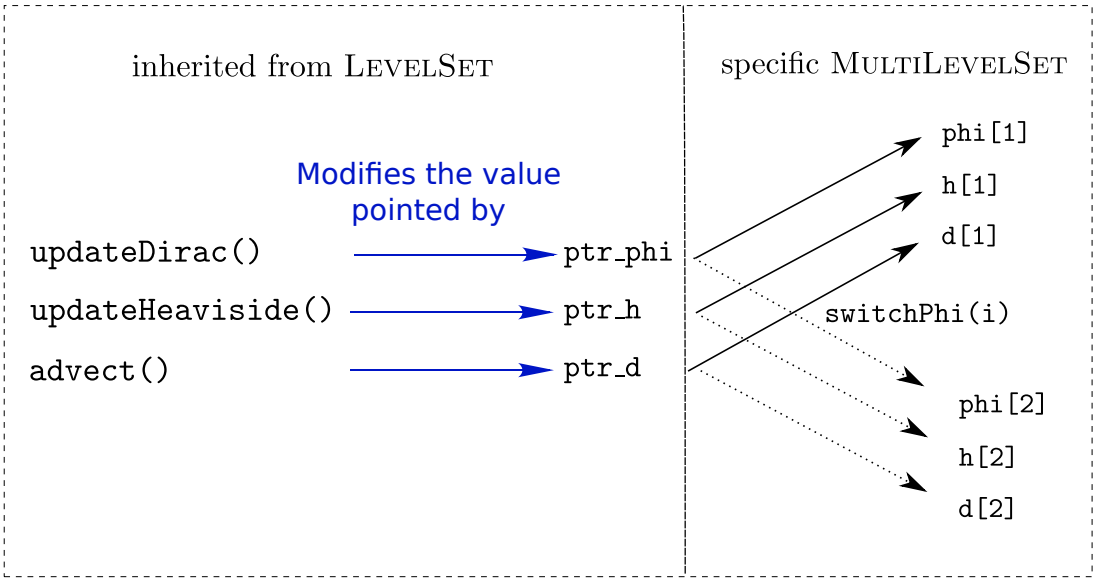


Figure 5.6: Action of `switchPhi()` on the pointers inherited from LevelSet .

method in a multi-level set context takes new arguments being the number of the level set to advect. Then one only needs to switch to the desired level set and advect it. The code is given in listing 5.10

Listing 5.10: The advect method in multilevelset context. `super` represents the mother class (here LevelSet)

```
template < typename Tu >
bool advect(Tu& u, int i)
{
  switchPhi(i);
  return super::advect(u);
}
```

5.4.2 Specific MultiLevelSet methods

The MultiLevelSet class provides accessors to individual ϕ, δ_ε or H_ε fields by the methods called `phi(i)`, `H(i)` and `D(i)` accepting one argument being the number of the field wanted. The class also provides interesting methods such as `minPhi()`, `globH()` and `globD()`. The first method returns a field corresponding to the minimum of all the level set fields (that we called ϕ_0 in section 2.2.2, equation (2.34)). Indeed, as explained in section 2.2.2, in the context of a multi-fluid application, this can be useful since the normal and curvature depend only on the geometry of the iso 0 and are not specific to each level set field. Thus, if the level sets are not interpenetrating, one could compute the geometrical fields (normal and curvature) once and finally use the delta function to

differentiate the areas of the different level sets.
The two other methods are simply the delta and Heaviside functions applied to the field ϕ_0. The `globH()` function can be used to describe the regions inside any interface or outside every interface. The `globD()` function can be used to distribute a quantity on all the interfaces regardless of which individual level set it is. We give an example to illustrate the use of these functions. In section 5.3, we have seen how to define the viscosity and the surface tension force in the context of a two-fluid flow. Let see how this works for a more than two-fluid flow. We already showed that the viscosity of such a system is given by the equation (2.36). The surrounding fluid is represented by the area in which all the level set fields are positive, it has a given viscosity `mup`. The other fluids have different viscosities stored in an array `mum`. Let us say we have an instance `multilevelset` of the class MultiLevelSet . The viscosity of such a system can be given by:

```
auto mu = project(scalarSpace, elements(mesh),
                  mup * idv(multilevelset->globH()) );

for (int i=0; i<multilevelset->getNbPhi(); i++)
{
  mu += project(scalarSpace, elements(mesh),
        mum[i] * (1 - idv(multilevelset->H(i))) );
}
```

If the number of different fluids is known at the compilation time, one could avoid the `for` loop and directly write the viscosity in one single projection. We wrote here the more general way to do it where the number of different fluids is not known at compilation time, but only at execution. Let us see now how to write the surface tension force between the different interfaces and the surrounding fluid. We will not describe here the general case where all the interfaces can be interacting with each other. Such a force is given by equation (2.37). As explained in section 2.2.2, we will use the fact than the min of all the level sets contains the geometrical information (normal and curvature) of all the interfaces at the same time. The different surface tension coefficients are given in an array called `sigma`. The surface tension force reads:

```
// expression of |∇φ|
auto modgradphi = sqrt( gradv(multilevelset->minPhi())
* trans(gradv(multilevelset->minPhi())));

// normal
auto n = project(vectorialSpace, elements(mesh),
         gradv(multilevelset->minPhi())
         / modgradphi );

// curvature
auto k = project(scalarSpace, elements(mesh), divv(n) );

// surface tension force
auto F = vectorialSpace()->element();
for (int i=0; i<multilevelset->getNbPhi(); i++)
{
  F += project(vectorialSpace, elements(mesh),
       - sigma[i] * idv(k) * idv(n) * idv(multilevelset->D(i)) );
}
```

Chapter 6

Specific development for the Vesicle application

Contents

We present in this chapter a few developments which have been made specially for the vesicle application. The first one is the problem of constructing the Lagrange multiplier which insures that the vesicle is unstretchable. This requires creating a subspace defined on a submesh around the vesicle. The Lagrange multiplier is defined only in this subspace which reduces substentially the number of total degrees of freedom in the final problem. Then we explain our strategy to make user-friendly simulations. Indeed, the vesicle code is made to be very generic and many options have to be set by the user. For real physicall simulation in which we only want to see the physically relevant parameters, we created a Python interface which allows us to concentrate on the physics of the problem, making transparent all the inter-dependent options.

6.1 Lagrange multiplier construction

6.1.1 A size problem

We have seen in section 4.3 that the constraint of the inextensibility of the membrane can be satisfied by adding a Lagrange multiplier to the variational formulation of the problem. This Lagrange multiplier is supposed to act on the membrane of the vesicle (i.e. in the area where $\delta_\varepsilon > 0$). If we look at the expression (4.23), the discretized Lagrange multiplier λ_h is defined on all the domain Ω and its action is restricted to the interface by multiplying by the function δ_ε. Thus, from the algebraic point of view, the matrix C, which corresponds to the contribution of the variables in the trial space $\mathbb{L}_h^k(\Omega)$ associated to the Lagrange multiplier and the test space $\mathbb{U}_h^{k+1}(\Omega)$ associated to the velocity, has its coefficients defined by

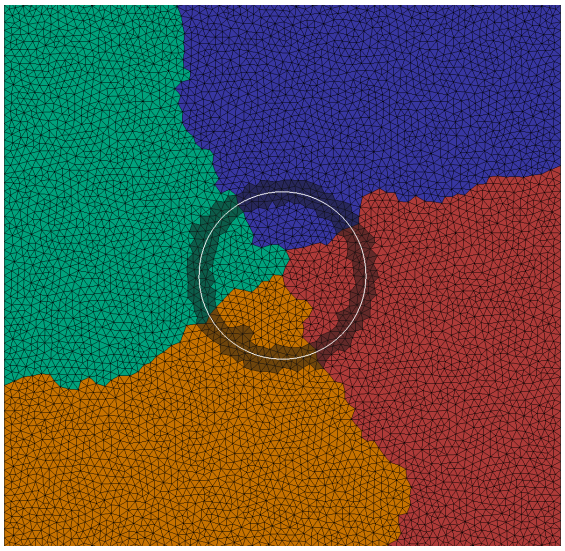
$$C_{ij} = \int_\Omega \delta_\varepsilon \xi_j \nabla_{\mathrm{s}} \cdot \boldsymbol{\Phi}_i = \int_\Omega \delta_\varepsilon \xi_j (\boldsymbol{I}_{\mathrm{d}} - \boldsymbol{n} \otimes \boldsymbol{n}) : \boldsymbol{\nabla} \boldsymbol{\Phi}_i.$$

The sparse matrix C has as many entries as there are non-vanishing pairs $(\xi_j, \nabla_{\mathrm{s}} \cdot \boldsymbol{\Phi}_i)$ in all the domain Ω. Of course, most of these entries are set to 0 thanks to the delta function, but they are still created. The size of this matrix is large. It makes the global matrix heavy to store in memory and the problem (4.26) is at best difficult, at worst impossible to solve.

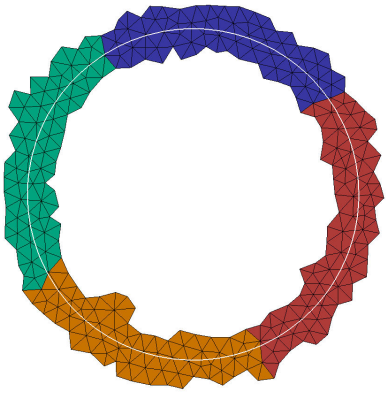
An optimization has been made which consists of adding in the matrix C only the entries which are not vanishing because of the delta function. This optimization is made by changing the space in which is defined the Lagrange multiplier $\mathbb{L}_h^n(\Omega)$ by $\mathbb{L}_h^n(\Omega_I)$. The domain Ω_I is a sub domain of Ω which corresponds to the elements of Ω for which $|\phi| < \varepsilon$. In other words, Ω_I corresponds to the domain of the interface, where $\delta_\varepsilon \neq 0$. Doing so requires extracting the submesh associated to these elements in an optimized manner. Then create the matrix C which counts a much lower number of entries than with the non optimized construction. Finally, one has to assemble the global matrix.

6.1.2 Extract the submesh

There is a method in Feel++ to extract a submesh from another mesh. The method makes sure that the daughter and the mother meshes keep the information of their relationship. It is always possible to project a variable which belongs to a space defined on a



(a) Initial mesh, the different partitions are represented in color. The area in which $\delta_\varepsilon > 0$ is represented in grey. The iso 0 of the level set is represented in white.



(b) Extracted submesh. The partitions are the same than the one of the original mesh. The level set position did not change during the extraction.

Figure 6.1: Extraction of a sub mesh.

mesh, on a space defined on another mesh. The degrees of freedom of the space owning the variable are localized on the mesh on which it is projected. This localization takes time. By keeping the information of the relationship between the meshes, we are able to avoid the localization since the two meshes perfectly match. This optimization has been made by the Feel++ developers in the context of the vesicle application but can be now used for many purpuses. Moreover, the submesh keeps the information of the partition of the mesh. Thus, there is no need to repartition it, to find again the ghost cells and so on. Extracting the submesh is then a relatively costless operation. The method used to create the submesh is called `createSubmesh`, which needs the original mesh and an iterator on the elements to be extracted. In the context of the Lagrange multiplier acting on the interface, we can use the marker described in section 5.3: `markerDelta` which marks the elements for which $\delta_\varepsilon > 0$. The figure 6.1(a) shows a mesh on which is defined a level set field for which the 0 represents a circle centered on the domain. The different partitions are represented in color. The area in which `markerDelta` is 1 is grey tone. The figure 6.1(b) represents the sub mesh extracted from the initial one. The colors are the different partitions. We can check that the partitions are exactly the same as the position of the iso 0 of the level set.

The code to create the associated submesh to the elements around the interface is really simple. We give an example doing so in listing 6.1

Listing 6.1: Extracting the submesh around the interface.

```
// mark the elements to extract
mesh->updateMarker2( *levelset->markerDelta() );
// extract the submesh and store it
auto submesh = createSubmesh(mesh, marked2elements(mesh, 1) );
```

6.1.3 Creating the Lagrange Multiplier contribution

For clarity reasons, in section 4.2 we expressed the matrices as if all the spaces associated to the velocity, the pressure and the Lagrange multiplier were completely separated. In fact, in the implementation that we did, the velocity and the pressure are actually embedded in a composite space being the product of the velocity space and the pressure space. Let us call this space $\mathbb{D}_h^k(\Omega) = \mathbb{U}_h^{k+1}(\Omega) \times \mathbb{P}_h^k(\Omega)$. This way, the matrix associated to the pure fluid problem is assembled in one block. We can call this matrix D, and it is simply defined as the contributions of the velocity and the pressure in the same matrix. With the notation of section 4.2, it reads:

$$D = \begin{pmatrix} A & B \\ B^t & 0 \end{pmatrix}.$$

We can remark by looking at the equations (4.23), (4.24) and (4.25) that since the equations solved are the Stokes equations, the only time dependent parameter is the position of the interface. Thus in these equations, only the viscosity μ_ϕ is time dependent. Consequently, if one makes a simulation with a viscosity ratio between the inner and outer fluid equal to 1, the whole matrix D can be created once and for all at the beginning of the simulation and reused every time step. If the viscosity is changing, we can optimize by changing only the elements which have been crossed by the interface (which are accessible by the marker `crossedelements`).

Then a matrix associated to the trial space $\mathbb{L}_h^k(\Omega_I)$ and the test space $\mathbb{D}_h^k(\Omega)$ is assembled. Let us call this matrix E. Of course, since there is no coupling between the pressure and the Lagrange multiplier, the corresponding entries are null. The matrix E is thus expressed as:

$$E = \begin{pmatrix} C \\ 0 \end{pmatrix}$$

At the opposite of D for which the construction might be optimized from an iteration to the other, the matrix E has to be recreated every time step. Moreover, this matrix does not have the same number of elements at each iteration since the size of Ω_I depends on how many elements are in the area where $\delta_\varepsilon > 0$. A typical example of the way to construct this matrix is given in listing 6.2. It assumes that we already provided a submesh associated to the elements of the interface. Then it creates the space $\mathbb{L}_h^k(\Omega_I)$ and assembles the associated matrix.

Listing 6.2: Creation and assembly of the matrix E.

```
// fluidSpace represents 𝔻_h^k(Ω)

// create 𝕃_h^k(Ω_I)
lagMultSpace = lagMultSpace_type::New(_mesh=submesh);

// get basis functions associated to λ and u
auto lam = lagMultSpace->element();
auto u = D->element<0>();

// identity matrix I_d
auto Id = Id<Dim, Dim>();
// n ⊗ n
auto nxn = idv(n)*trans(idv(n));
```

```
//surfacic divergence: (I_d - n ⊗ n) : ∇u
auto divs = trace( (Id-nxn) * trans(grad(u)) ) ;

E = M_backend->newMatrix(lagMultSpace, fluidSpace);
form2(_trial=lagMultSpace, _test=fluidSpace, _matrix=E)=
// ∫_{Ω_I} δ_ε λ∇_s · u
integrate( elements(submesh), idt(lam) * divs * idv(levelset->D()) );
E->close();
```

The transpose of E will also be needed in the final assembled matrix. It can simply be constructed by doing:

```
Et = M_backend->newMatrix(fluidSpace, lagMultSpace);
E->transpose(Et);
```

6.1.4 The global matrix assembly

At this stage, we dispose of the matrix associated to the fluid problem D, and the one associated to the Lagrange multiplier E and its transpose E^t. The final global matrix (let us call it G) is then the assembly of the different blocks:

$$G = \begin{pmatrix} D & E \\ E^t & Z \end{pmatrix}.$$

The last matrix Z corresponds to the coupling between the test and trial functions of the Lagrange multiplier space; since such terms do not exist in the variational formulation, all these entries are vanishing but the matrix still needs to be created.

During his PhD, Vincent Chabannes [17] created a tool to assemble some matrices block by block. This algorithm is fully parallel. We used this framework to assemble together the different contributions. In listing 6.3 we have put an example of how to create the matrix G.

Listing 6.3: Creation of the matrix G block by block.

```
// creation of the 0 matrix
  auto Z = backend()->newMatrix(_test=lagMultSpace_type,
                                _trial=lagMultSpace_type);
  Z->zero();

  // assemble the blocks
  BlocksBaseSparseMatrix<double> myblockMat(2,2);
  myblockMat(0,0) = D;  // fluid (velocity + pressure)
  myblockMat(0,1) = E;  // lagrange multiplier / velocity coupling
  myblockMat(1,0) = Et; // transposed of E
  myblockMat(1,1) = Z;  // the zero matrix

  // create the matrix G
  auto G = backend()->newBlockMatrix(_block=myblockMat, _copy_values=true);
```

6.1.5 Performance test

A small performance test has been made in order to show how important was this optimization. The setup is close to the one presented in section 4.4.2. A vesicle with a reduced area of 1 and a perimeter equal to 2π is centered in a box of (10×10). The

viscosity ratio is equal to 1 as well. The interface thickness is set to $\varepsilon = 2\,h$, where h is the mesh size. The polynomial order has been taken to $k = 1$. A shear flow is imposed. The goal of the test is simply to compute the size of the matrices associated to this problem by taking the non-optimized method $\mathbb{L}^1_h(\Omega)$ or the optimized one $\mathbb{L}^1_h(\Omega_I)$. We also performed one resolution of the final linear system and recorded the time to do it.

The table 6.1 shows the results for the non-optimized version of the test while the table 6.2 gives the results of the optimized one. We reported in the table the mesh size, the number of non-zero entries (nnz) of the different matrices of the problem and the time to solve the resulting linear problem. We emphasize the fact that nnz represents the number of entries which are non-zero by construction since the product of their basis functions does not vanish. All the entries which are in this case are taken into account in nnz, even the ones for which their value is set to 0 after the construction of the matrix (else the matrix Z would always have $\text{nnz}(Z) = 0$). Consequently, nnz gives how much information is stored in memory for each sparse matrix.

The simulations have been run separately on a single processor `Intel(R) Core(TM) i7-2820QM CPU @ 2.30GHz`. If the time to solve the final problem exceeded 6 minutes, the program was killed and a cross $\times$ is reported in table 6.1.

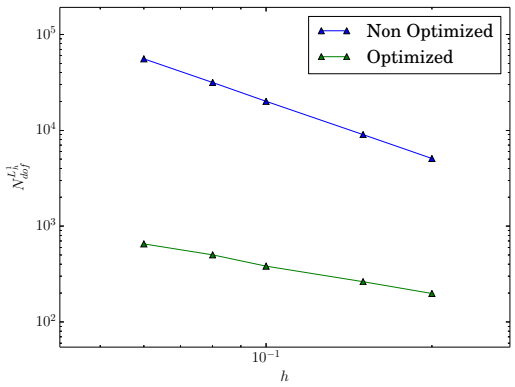
We report in the figure 6.2 some values taken in the tables which emphasize how much the optimization is essential. One can see in figure 6.2(a) that the slope of the logarithmic curve of the number of degrees of freedom in the space $\mathbb{L}^1_h$ as a function of the mesh size is better for the optimized version than for the non optimized one.
Moreover, the absolute value of the number of degrees of freedom is always one or two orders of magnitude lower for the optimized version. The same remark can be made for the number of non-zero entries in the matrices E and Z. Finally, the time cost increases dramatically when decreasing the mesh size in the non-optimized version. Even for this simple problem, the time was too long to solve the problem for the two best approximations. The optimized version, at the opposite, is solved in a reasonable time and shows a slope 3 order lower on the graph 6.2(d).

h	$N^{\mathbb{L}^1_h(\Omega)}_{\text{dof}}$	$N^{\mathbb{D}^1_h(\Omega)}_{\text{dof}}$	nnz(E)	nnz(D)	nnz(Z)	nnz(G)	time (s)
0.2	5076	45178	197072	$1.32853\,10^6$	35026	$1.62518\,10^6$	8.16
0.15	9008	80398	351488	$2.36947\,10^6$	62382	$2.8994\,10^6$	38.96
0.10	20046	179408	785418	$5.29902\,10^6$	139316	$6.48066\,10^6$	357.76
0.08	31500	282242	$1.23654\,10^6$	$8.34389\,10^6$	219242	$1.02041\,10^7$	$\times$
0.06	55626	498960	$2.18792\,10^6$	$1.47635\,10^7$	387708	$1.80572\,10^7$	$\times$

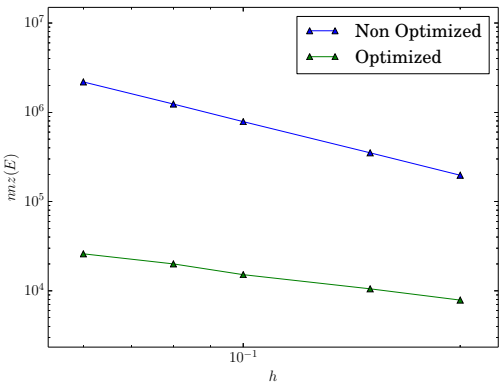
Table 6.1: Results of the test without the optimization. In this test, $\lambda \in \mathbb{L}^1_h(\Omega)$. We denote nnz($M$) the number of non zero entries of a matrix M.

h	$N_{\text{dof}}^{\mathbb{L}_h^1(\Omega_I)}$	$N_{\text{dof}}^{\mathbb{D}_h^1(\Omega)}$	nnz(E)	nnz(D)	nnz(Z)	nnz(G)	time (s)
0.2	198	45178	7869	$1.32853\,10^6$	1242	$1.33913\,10^6$	0.72
0.15	263	80398	10525	$2.36947\,10^6$	1651	$2.3843\,10^6$	1.51
0.10	382	179408	15166	$5.29902\,10^6$	2378	$5.31921\,10^6$	4.02
0.08	501	282242	20023	$8.34389\,10^6$	3133	$8.37059\,10^6$	7.49
0.06	652	498960	25966	$1.47635\,10^7$	4064	$1.47977\,10^7$	15.81

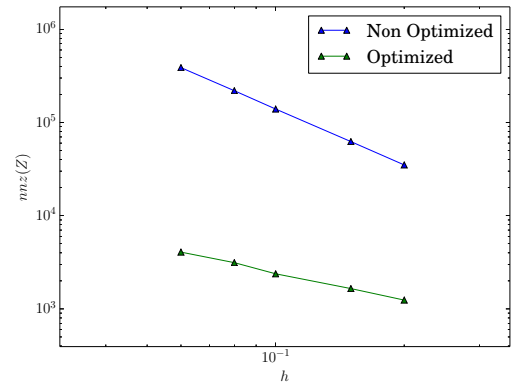
Table 6.2: Results of the test with the optimization. In this test, $\lambda \in \mathbb{L}_h^1(\Omega_I)$. We denote nnz($M$) the number of non zero entries of a matrix M.



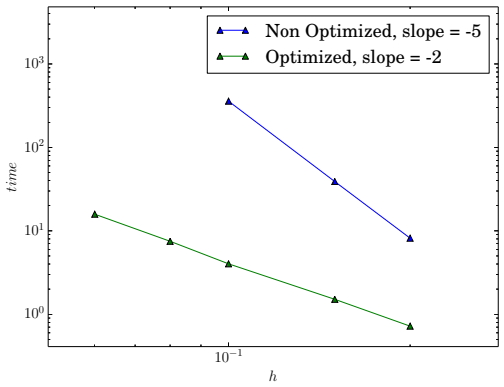
(a) $N_{\text{dof}}^{\mathbb{L}_h^1}(\Omega)$ and $N_{\text{dof}}^{\mathbb{L}_h^1}(\Omega_I)$ as a function of h.



(b) Number of entries in the sparse matrix E.



(c) Number of entries in the sparse matrix Z.



(d) Time to solve the linear system (in second) as a function of the mesh size. The slopes of the lines are reported in the legend.

Figure 6.2: Some results taken from tables 6.1 and 6.2 comparing the optimized and non optimized methods. We can see that the non zero entries in the matrices increases slower when the mesh size decreases with the optimization than without. Moreover, the absolute value of the number of non zero entries in the optimized version is one or two order of magnitude lower than the non optimized one.

6.2 A Python interface

6.2.1 Why an interface is needed?

A lot of parameters have been presented in this work which have to be set by the user according to the needs of the simulation. All the design of the different programs presented in this work has been made with the goal of a maximum of generality. Thus, all the parameters arising from the numerical models are never set in the code to a value, but left as an external option to the user. In section 5.3 we briefly presented the options automatically present when one creates a LevelSet object. To give a few examples of what are these options, let us just cite the thickness of the interface, the advection stabilization method, the coefficient for the stabilization, the reinitialization frequency, the reinitialization method; if the Hamilton Jacobi equation reinitialization is chosen, the pseudo time step, the maximum iteration, the tolerance, the stabilization for this advection equation and so on. With these options, come the Feel++ standard options, like the format in which the data are exported, the directory to export the data and many others. Other options come from other Feel++ classes, as for example the fluid solver which is coupled to the level set or the backend options. The backend options are the options which are given to Petsc [3], the algebra solver that Feel++ uses. Thus it controls by which method a particular algebraic system is solved, consequently, if one wants a control to every system solved in his program, one needs as many copy of the backend options that it exists different types of system to solve in the problem. All these added options make a huge number of options to set. Feel++ uses the [1]boost::program_options library as option parser. This library provides the possibility to set default options. Thus, when an option is frequently used to a common value, the default value can be set and there is, most of the time, no need to change it. This reduces the number of options to be set by the user. Another feature that boost::program_options provides is the possibility to handle configuration files. Thus, the options do not need to be given directly in the command line but can be set in a file.
The backend and test applications presented in section 2.3 have a precise set of parameters which are defined when the test is designed and almost never changed. Thus, for these applications, a configuration file have been set and no more options than the name of this file has to be given. Launching the application is then very simple, it looks like:
`application_name --config-file=a_config_file.cfg`.
For the vesicles or rigid disk applications, things are more complicated. Indeed, these applications are meant to explore some physical parameters which are not necessarily known when creating the application and the numerical parameters have to be adapted to the particular simulation. Moreover, the options are generally inter-dependent. As an example, the interface thickness depends on the mesh size, as well as the time step. The reinitialization frequency depends on the time step, the pseudo time for reinitialization depends on the mesh size and the time step as well... The physical parameters are also inter-dependent or might at least be defined thanks to dimensionless numbers (Reynolds, capillary) that a physicist user prefers to give instead of direct parameters (velocity, rigidity).
For all these reasons, we usually created an interface to our programs which handles the

[1] http://www.boost.org/doc/libs/1_54_0/doc/html/program_options.html

inter-dependency of the options. These interfaces are simple PYTHON programs. The choice of PYTHON has been made because of the easy readability of codes that it generates and the dynamic typing which makes the code easy to modify day to day. Moreover, it provides an efficient oriented object paradigm and a well furnished set of libraries (including scientific libraries like [2]SCIPY) [51] which allow more complicated operations if needed.
The idea is the following: for a particular simulation, we put all the options which are unlikely to be modified and which are not inter-dependent in a config file which is almost never modified. Then, we create the PYTHON code which handles the inter-dependencies, reducing the number of variables to set by the user to only a few of them. Finally, an argument parser for python is used and the user runs the python program with very few parameters of direct interest to the simulation being made.

6.2.2 An example: the vesicle in a shear flow application

As a simple example, let us show what the interface looks like for the application of a single vesicle in a shear flow that we presented in section 4.4.2. Many parameters have to be set, but the user, when interested in the physics, only wants to see some physical parameters such as the shear rate, the size of the box, the reduced area, and the viscosity ratio, as well as some numerical parameters such as the mesh size or the number of processors on which the application in run. First of all, a config file has to be made to set all the fixed parameters which do not have the same value as the defaults. An example of such a config file for this application is given in appendix F. Then, the PYTHON program sets a few options of direct interest for the simulation, thanks to the module [3]argparse:

Listing 6.4: Create the options of the vesicle in shear flow application

```
import argparse

parser = argparse.ArgumentParser()

# physical parameters
parser.add_argument("-height",        type=float, default=10 )
parser.add_argument("-large",         type=float, default=30 )
parser.add_argument("-reduced-area",  type=float, default=0.9 )
parser.add_argument("-visco-ratio",   type=float, default=1. )
parser.add_argument("-gamma",         type=float, default=1. )

# numerical parameters
parser.add_argument("-h-min",         type=float, default=0.1 )
parser.add_argument("-h-max",         type=float, default=0.1 )
parser.add_argument("-nb-proc",       type=int,   default=1 )
parser.add_argument("-export-name",   type=str,   default="data")

args = parser.parse_args()
```

Then we get the axes of the ellipse according to the reduced volume for a fixed area (that we fixed to π). For that, we use the method presented in appendix B and the same code which has been wrapped in a PYTHON function called `computeAxes`. It reads:

```
a_ell, b_ell = computeAxes( alpha = args.reduced_area, area = pi )
```

[2] http://scipy.org/
[3] http://docs.python.org/2/library/argparse.html

The mesh is also created from the interface. Actually, Feel++ provides interfaces to create several meshes which are really useful for testing applications (unit square, unit circle or sphere). But if we want a precise control of the mesh, like having a part of the mesh with a smaller typical size, once again, a Python interface can be useful. Thus, we made several functions which generate scripts for Gmsh and call it to create the mesh of the application. As examples, functions creating the following meshes are available:

- simple rectangular mesh
- rectangular mesh with one rectangular area in the middle having a different mesh size
- rectangular mesh having an arbitrary number of rectangular areas with different mesh sizes
- circular mesh having a circular area with different mesh size
- the bifurcation mesh used in section 7.1 as well as the 3D version [87]

We use for this application the rectangular mesh with a smaller mesh size at the middle:

```
mesh_name = makeMeshBox(large = L, height = l,
largeBand = 3*a_ell, heightBand = 3*b_ell,
hmin = args.h_min, hmax = args.h_max,
name="mesh", periodic=True, partitions=args.nb_proc)
```

We then create the command to run and compute at the same time the dependent parameters. The command is put in an array in which each argument is an element. And finally, the command is launched by the module [4]subprocess of Python .

```
l = args.height
L = args.large
command = [
# mpi command to run on multiple processors
"mpirun",
"-np",
str(args.nb_proc),
"-bind-to-core",

# -- application options --
# name of the executable
ExeName,
"--config-file="+Config,
"--exporter.directory="+args.export_name,
"--meshFile="+mesh_name,

#  -- numerical parameters --
"--bdf.time-step="+str(args.h_min / 4. ),
"--levelset.thickness-interface="+str(args.h_min * 1.5),
"--diffnum="+str(args.h_min/50.),

# -- physical parameters --
# mu_2 = lambda * mu_1
"--mum="+str( 10 * visco_ratio ),
"--a_ell="+str(a_ell),
```

[4] http://docs.python.org/2/library/subprocess.html

```
    "--b_ell="+str(b_ell),
    "--x0="+str(L/2.),
    "--y0="+str(1/2.),
    "--stokes.shearVelocity="+str(1 * gamma / 2.)]

    calc = subprocess.Popen(command)
    calc.wait()
```

The user can now concentrate on the physics and has only few arguments to set. As an example, we can now do a simulation of vesicle under shear flow by calling:
`vesInShearFlow.py -reduced-area=0.8 -gamma=2 -visco-ratio=3 -nb-proc=5`

Part III

Rheology and flows of solid disks

Chapter 7

Flow of disks at a microfluidic bifurcation

Contents

In this chapter, we present some results on the physical problem of the splitting of a suspension of particles in a microfluidic bifurcation. We first start by a simplified problem: a dilute suspension of hard disks. With this simplified model, we are able, in collaboration with experimenters, to explain the known phenomenon of the increasing concentration of particles in the higher flow rate branch. These explanations have been published in Journal Of Fluid Mechanics [27] and we reported the publication in this chapter. We finally show that it will be possible in the near future to do the same simulation with vesicles. We show that it is possible to measure the flux of vesicles in each branch and present a preliminary result. Finally we extend this simulation to the simulation of a suspension of two different kind of vesicles.

7.1 The splitting of a suspension at a bifurcation

7.1.1 The Zweifach-Fung effect

When a dilute suspension of particles reaches a geometrically symmetric bifurcation in which the branches do not have the same flow rate, the branch which has the higher flow rate always sees its concentration of particles increase. This phenomenon is called the Zweifach-Fung effect. It has often been interpreted as a consequence of the existence of an hydrodynamical force pushing particles toward the high flow rate branch. In other words, some particles in the inlet channel which have their center of mass in a region where the fluid goes to the low flow rate branch would actually go in the high flow rate branch. The goal of this work was to figure out if such a force exists. We have actually shown that there is no need for an hydrodynamical force to have an increase of the volume fraction of particles in the high flow rate branch. Indeed, this effect is a consequence of a geometrical effect of the distribution of particles in the inlet channel. Then, experiments in our group combined with our numerical simulations have shown that, at the opposite, there exists a force pushing particles toward the low flow rate branch. This force is not strong enough to overcome the geometrical distribution effect; this is why one sees an increase of the volume fraction of particles in the high flow rate branch. Let us explain in more detail this phenomenon.

7.1.2 The bifurcation without particles

We consider a fluid flow governed by the Stokes equations in a 2D bifurcation for which the inlet channel makes a 90° angle with the two daughter branches. A different flow rate exists in the two daughter branches. We call the branch having the lower flow rate *branch* 1 and its flow rate Q_1, the other one is *branch* 2 and its flow rate Q_2. Finally the mother branch is branch 0 and its flow rate $Q_0 = Q_1 + Q_2$. The velocity profile of the fluid in each branch away from the bifurcation is a parabolic profile. Thus we can calculate the position y_f in the inlet channel separating the fluid regions going to the branch 1 or 2 by simply writing the flow rate ratio:

$$\frac{Q_1}{Q_0} = \frac{\displaystyle\int_{y_f}^{L/2} \boldsymbol{u}_0 \cdot \boldsymbol{n}_0}{\displaystyle\int_{-L/2}^{L/2} \boldsymbol{u}_0 \cdot \boldsymbol{n}_0} \tag{7.1}$$

where $\boldsymbol{u}_0$ is the velocity profile in the inlet channel, $\boldsymbol{n}_0$ is the outward normal to the inlet 0 and L is the length of the channel and the ratio $\frac{Q_1}{Q_0}$ is known. Assuming that $\boldsymbol{u}_0$ is a parabolic profile (Poiseuille flow), one obtains the value of y_f. A scheme of such a bifurcation is represented in figure 7.1.

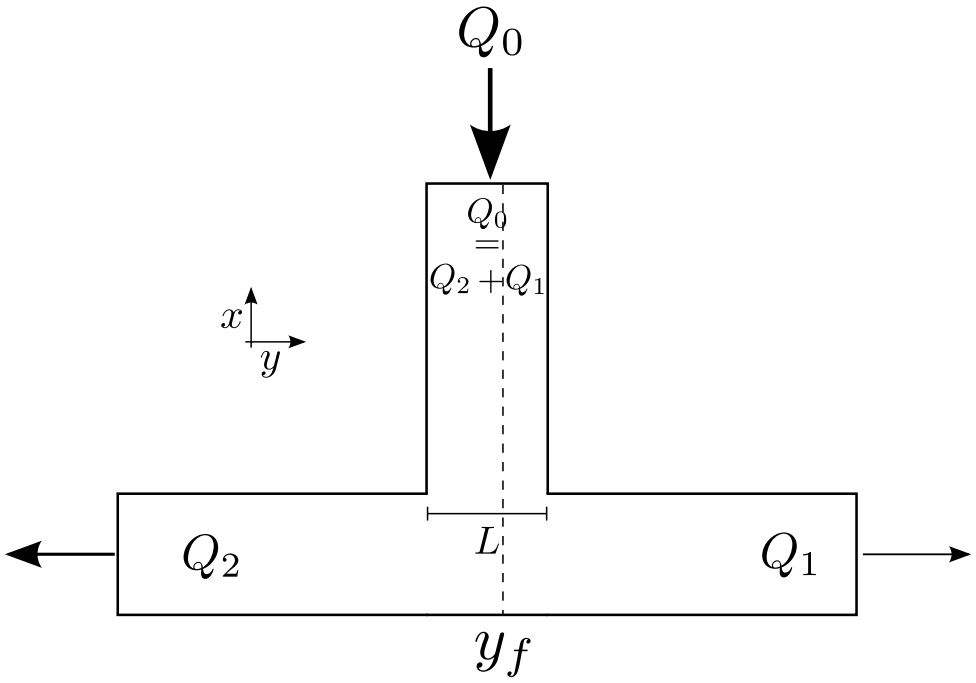


Figure 7.1: Bifurcation with a flow rate ratio $\frac{Q_1}{Q_0} = 0.37$. The separating line of fluid y_f is represented in dashed line. The fluid on the left of the separating line in the inlet channel goes in the left hand side branch and reciprocally for the right hand side.

7.1.3 Particle distribution with the simplest model

We will now start by the simplest possible approximation of the bifurcation and then add the following effects: influence of the free particle zone, influence of a hydrodynamic force. To start, we make the hypothesis that the center of mass of the particles can be anywhere in the inlet channel. Of course this is not true since it would mean that the particles can cross the wall. Nevertheless, this hypothesis could be relevant for very small particles for which the effect of the free particle zone is negligible. We also make the assumption that the dilute suspension is homogeneous, thus the probability to find the center of a particle is the same everywhere in the inlet branch. Such a configuration is shown in figure 7.2. In this figure, the positions of the particles can seem too close to be in the dilute regime in which there is no interaction between the particles, but only the position relative to the wall is important. Thus, we can say that we shrank a lot the distance between the particles for the scheme but that in reality it should be high enough so that the hydrodynamic interactions between particles are negligible. We also make the hypothesis that there is no hydrodynamic force pushing particles preferentially toward any branch. Thus, if the center of a particle is at the left of the separating line, it goes in the left hand side branch and respectively for the other one. The area in the inlet channel

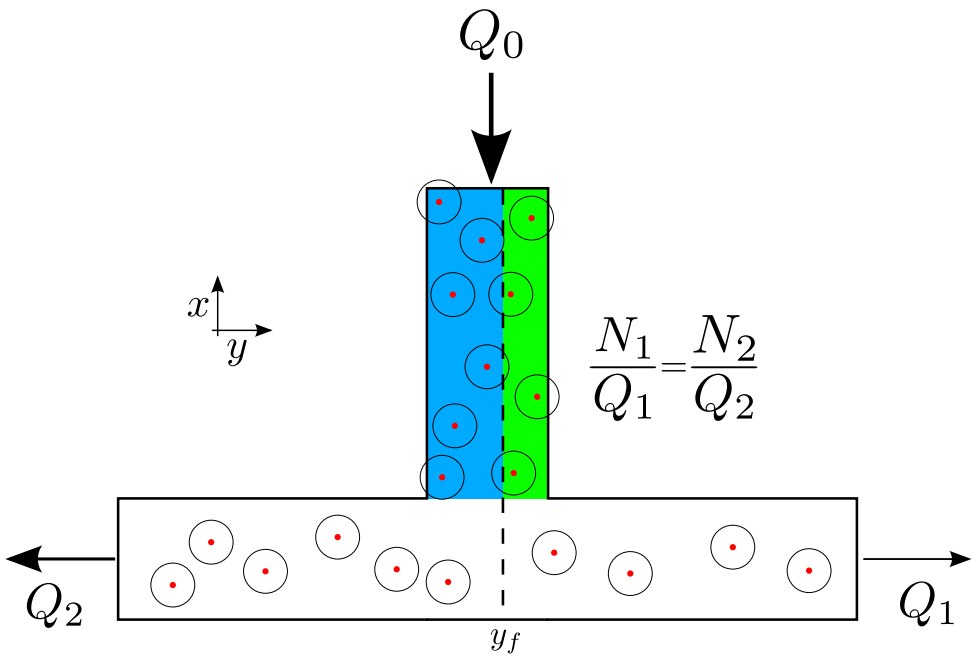


Figure 7.2: Distribution of particles in a bifurcation when the finite size of the particles is not taken into account and when no force pushed them to a particular branch. The center of mass of the particles are represented as red dots. The area in the inlet channel for which the particles present will go in the high flow rate branch is represented in blue. The area to go in the other branch is in green. In this configuration, there is no concentration increase in any branch.

in which particles will go in channel 1 is represented in green in figure 7.2 and the one in which the particles are going in branch 2 is represented in blue. The line separating the two areas is y_f, because of the no force assumption. We call N_1 and N_2 the particle flow rate in branches, respectively, 1 and 2.

With the previous hypotheses, we can say that the ratio between the number of particles going in each branch is the same as the flow rate ratio ($\frac{N_1}{N_2} = \frac{Q_1}{Q_2}$). Indeed the area in the inlet branch determining the destination branch is the same for the particles and the fluid. Thus, the volume fraction, which is the ratio between the number of particles going in a branch and the flow rate in the same branch $\frac{N}{Q}$, is the same for the two daughter branches ($\frac{N_1}{Q_1} = \frac{N_2}{Q_2}$). With this relation and recalling that $Q_0 = Q_1 + Q_2$ and $N_0 = N_1 + N_2$, we obtain that $\frac{N_0}{Q_0} = \frac{N_1}{Q_1} = \frac{N_2}{Q_2}$. Consequently, under the hypotheses that we formulated, the volume fraction is the same in all the branches. This is actually the case for very small particles for which our hypotheses are correct.

7.1.4 Effect of the free particle zone

Let us now consider that the particles are big enough so that the hypothesis saying that the center of mass of the particle can be everywhere in the inlet channel is not correct anymore. The particle's center cannot be closer to the wall than their radii. Thus, there

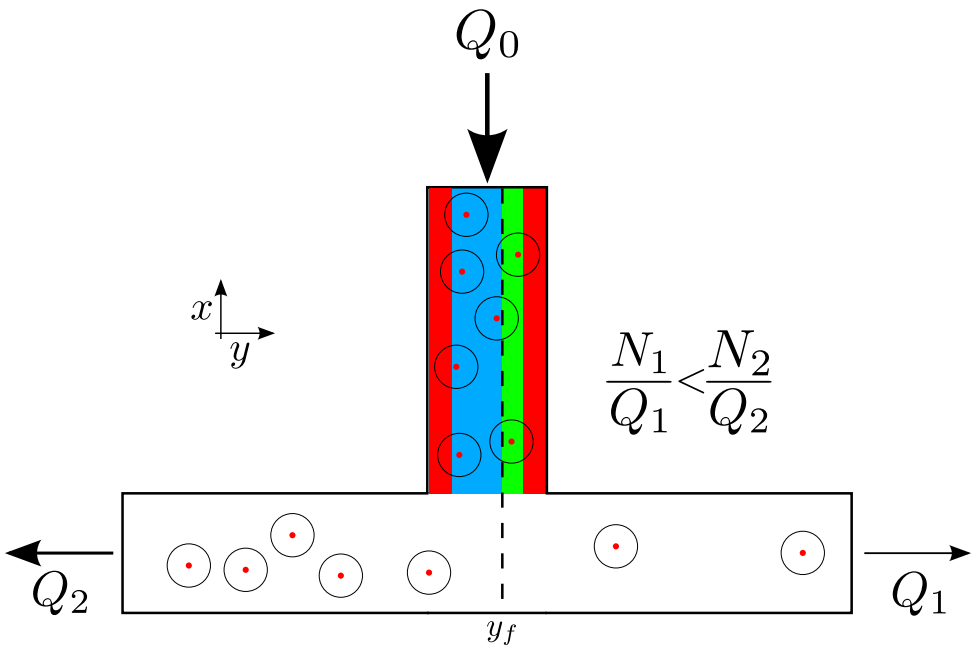


Figure 7.3: Distribution of particles in a bifurcation when the finite size of the particle is taken into account. The free particle layer in the inlet channel is marked in red. The number of particles entering in the low flow rate branch has been decreased more than the one entering in the other one. Consequently, there is an increase in the concentration of particles in the high flow rate branch.

exists a particle free zone in the inlet channel having a thickness equal to the radius of the particles in both sides of the channel. A scheme of the bifurcation with the free particle zone represented in red is shown in figure 7.3. A consequence of this free particle layer is that the area in the inlet channel in which the particles would enter the branch 1 is penalized compared to the other one. Indeed, an equal area (the free layer) is deleted from two non equal areas, thus relatively to their size, the smallest area is much more lowered than the other. In other words, N_1 decreases more than N_2. In a mean time, the fluid flow rate is not affected by the free layer zone, thus Q_1 and Q_2 stay the same than in figure 7.2. This breaks the previous equality and one has $\frac{N_2}{Q_2} > \frac{N_1}{Q_1}$, which means that there is an increase in the concentration of the particles in the high flow rate branch. Thus, only a geometrical consideration on the distribution of particles in the inlet channel is enough to explain the concentration increase in the high flow rate branch.

7.1.5 Hydrodynamic force

Even if we have shown that there is no need for an hydrodynamical force pushing particles toward the high flow rate branch to increase its concentration, it does not mean that such a force does not exist. Thus we compared the results found in the literature and the concentration of particles expected under the assumption that no hydrodynamical force such as described earlier exists. We have found that all the results in literature had actually a concentration of particles in the high flow rate branch lower than it should be if only the geometrical effect were present. This led us to think that there could at by

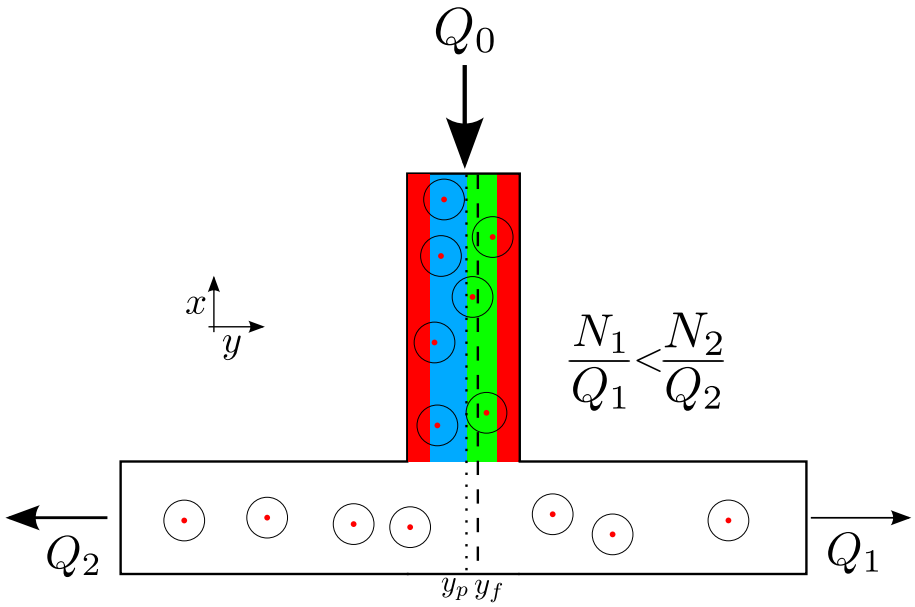


Figure 7.4: Scheme of the bifurcation when the finite size of the particle is taken into account so as a force pushing particles toward the low flow rate branch. The line separating the areas for which the particles go in each branch is represented as a black dotted line (at position y_p).

contrast exist a force pushing particles toward the low flow rate branch. We have shown by experiments and simulations that this force actually exists. The consequence of such a force is represented in figure 7.4. The line separating the areas where the fluid goes in each branch is no longer the same as the line separating the areas where the particles goes in each branch. The position of the separating line of particles is y_p. Subsequently, there exists a zone in the inlet channel, between y_p and y_f in which a solid particle would go to the low flow rate branch whereas a fluid particle would go to the higher flow rate one. The fact that particles are pushed toward the low flow rate branch implies that N_1 increases compare to the no force assumption. However, this force is never strong enough to invert the inequality $\frac{N_2}{Q_2} > \frac{N_1}{Q_1}$.

7.1.6 Finding the separating line of particles

The position of the separating line of particles (y_p) was the key to understand the whole phenomenon. Hence, our goal was to find this position. The experimenters Thomas Podgorski, Sarah Peponas and Gwennou Coupier designed an experiment in which they control with precision the position of an inlet particle in a bifurcation and explored the different trajectories until they found the separating line of particles. They did this experiment for several flow rate ratios and different sizes of particles. For the simulation part, we used the penalty method presented in section 3.1. The code has been written with the software FreeFem++ . Later, it has been re-written with Feel++ [87] to be able to handle 3D simulations. We searched for the separating line for different flow rate ratio, radii of particles, and sizes of channels. All the results, and their consequences on the Zweifach-Fung effect are published in [27] that we give hereafter.

J. Fluid Mech. (2011), *vol.* 674, *pp.* 359–388.
doi:10.1017/S0022112010006567

Spheres in the vicinity of a bifurcation: elucidating the Zweifach–Fung effect

V. DOYEUX, T. PODGORSKI, S. PEPONAS, M. ISMAIL AND G. COUPIER†

Université Grenoble 1 / CNRS, Laboratoire Interdisciplinaire de Physique / UMR 5588, Grenoble, F-38041, France

(Received 8 July 2010; revised 26 October 2010; accepted 21 December 2010; first published online 17 March 2011)

The problem of the splitting of a suspension in bifurcating channels divided into two branches of non-equal flow rates is addressed. As has long been observed, in particular in blood flow studies, the volume fraction of particles generally increases in the high-flow-rate branch and decreases in the low-flow-rate branch. In the literature, this phenomenon is sometimes interpreted as the result of some attraction of the particles towards this high-flow-rate branch. In this paper, we focus on the existence of such an attraction through microfluidic experiments and two-dimensional simulations and show clearly that such an attraction does not occur but is, on the contrary, directed towards the low-flow-rate branch. Arguments for this attraction are given and a discussion on the sometimes misleading arguments found in the literature is given. Finally, the enrichment of particles in the high-flow-rate branch is shown to be mainly a consequence of the initial distribution in the inlet branch, which shows necessarily some depletion near the walls.

Key words: blood flow, microfluidics, particle/fluid flows

1. Introduction

When a suspension of particles reaches an asymmetric bifurcation, it is well-known that the particle volume fractions in the two daughter branches are not equal; basically, for branches of comparable geometrical characteristics, but receiving different flow rates, the volume fraction of particles increases in the high-flow-rate branch. This phenomenon, sometimes called the Zweifach–Fung effect (see Svanes & Zweifach 1968; Fung 1973), has been observed for a long time in the blood circulation. Under standard physiological circumstances, a branch receiving typically one fourth of the blood inflow will see its haematocrit (volume fraction of red blood cells) drop down to zero, which will have obvious physiological consequences. The expression 'attraction towards the high-flow-rate branch' is sometimes used in the literature as a synonym for this phenomenon. Indeed, the partitioning not only depends on the interactions between the flow and the particles, which are quite complex in such a geometry, but also on the initial distribution of particles.

Apart from the huge number of *in vivo* studies on blood flow (see Pries, Secomb & Gaethgens 1996 for a review), many other papers have been devoted to this effect, either to understand it, or to use it in order to design sorting or purification devices.

† Email address for correspondence: gwennou.coupier@ujf-grenoble.fr

In the latter case, one can play at will with the different parameters characterizing the bifurcation (widths of the channels and relative angles of the branches), in order to reach a maximum efficiency. As proposed in many papers, focusing on rigid spheres has already given some keys to understand or control this phenomenon (see Bugliarello & Hsiao 1964; Chien *et al.* 1985; Audet & Olbricht 1987; Ditchfield & Olbricht 1996; Roberts & Olbricht 2003, 2006; Yang, Ündar & Zahn 2006; Barber *et al.* 2008). *In vitro* behaviour of red blood cells has also attracted some attention (see Dellimore, Dunlop & Canham 1983; Fenton, Carr & Cokelet 1985; Carr & Wickham 1990; Yang *et al.* 2006; Jäggi, Sandoz & Effenhauser 2007; Fan *et al.* 2008; Zheng, Liu & Tai 2008). The problem of particle flow through an array of obstacles, which can be considered as somewhat similar, has also been studied recently (see El-Kareh & Secomb 2000; Davis *et al.* 2006; Balvin *et al.* 2009; Frechette & Drazer 2009; Inglis 2009).

All the studies mentioned above have focused on the low-Reynolds-number limit, which is the relevant limit for applicative purposes and the biological systems of interest. Therefore, this limit is also considered throughout this paper.

In most studies, as well as in *in vivo* blood flow studies, which are for historical reasons the main sources of data, the main output is the particle volume fraction in the two daughter branches as a function of the flow rate ratio between them. Such data can be well described by empirical laws that still depend on some *ad hoc* parameters but allow some rough predictions (see Dellimore *et al.* 1983; Fenton *et al.* 1985; Pries *et al.* 1989), which have been exhaustively compared recently (see Guibert, Fonta & Plouraboue 2010).

On the other hand, measuring macroscopic data such as volume fraction does not allow identification of the relevant parameters and effects involved in this phenomenon of asymmetric partitioning.

For a given bifurcation geometry and flow rate ratio between the two outlet branches, the final distribution of the particles can be straightforwardly derived from two sets of data: first, their spatial distribution in the inlet and, second, their trajectories in the vicinity of the bifurcation, starting from all possible initial positions. If the particles follow their underlying unperturbed streamlines (as a sphere would do in a Stokes flow in a straight channel), their final distribution can be easily computed, although particles near the apex of the bifurcation require some specific treatment, since they cannot approach it as closely as their underlying streamline does.

The relevant physical question in this problem is thus to identify the hydrodynamic phenomenon at the bifurcation that would make flowing objects escape from their underlying streamlines and, as a consequence, a large particle would be driven towards one branch while a tiny fluid particle located at the same position would travel to the other branch.

In order to focus on this phenomenon, we need to identify more precisely the other parameters that influence the partitioning, for a given choice of flow rate ratio between the two branches.

(i) *The bifurcation geometry.* Audet & Olbricht (1987) and Roberts & Olbricht (2003) made it clear, for instance, that the partitioning in Y-shaped bifurcations depends strongly on the angles between the two branches (see figure 1*a*). For instance, while the velocity is mainly longitudinal, the effective available cross-section to enter a perpendicular branch is smaller than in the symmetric Y-shaped case. Even in the latter case, the position of the apex of the bifurcation relative to the separation line between the fluids travelling in the two branches might play a role, due to the finite size of the flowing objects.

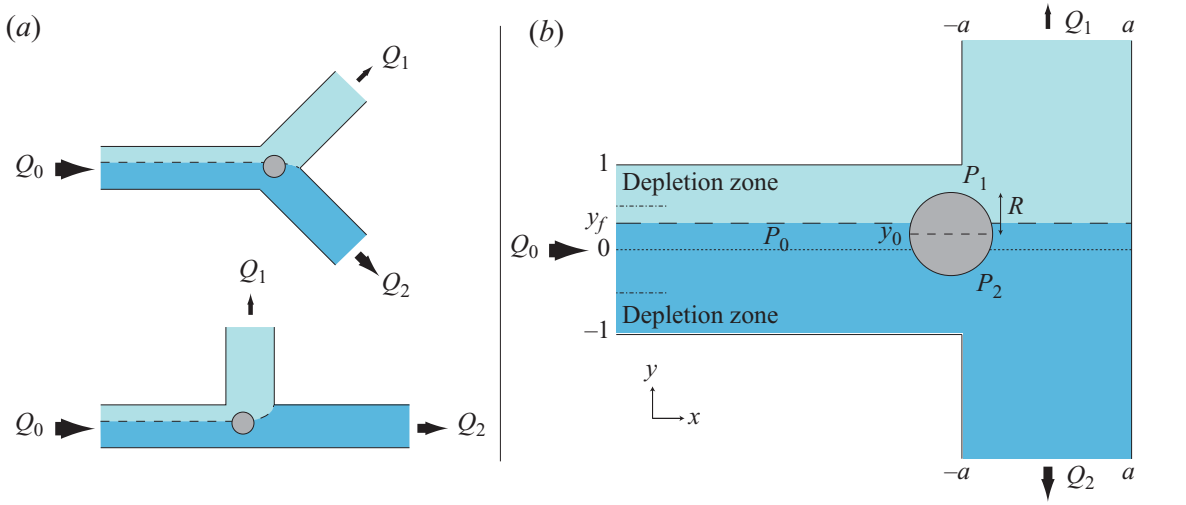


FIGURE 1. (Colour online) (*a*) The two Y-shaped geometries mainly studied in the literature. Here $Q_1 < Q_2$ and the dashed line stands for the separating streamline between the flows that will eventually enter branches 1 and 2 in the absence of particles. (*b*) The T-bifurcation that is studied in this paper and also in Chien *et al.* (1985) in order to remove geometrical effects as much as possible.

(ii) *Radial distribution in the inlet channel.* In an extreme case where all the particles are centred in the inlet channel and follow the underlying fluid streamline, they all enter the high-flow-rate branch; more generally, the existence of a particle free layer near the walls favours the high-flow-rate branch, since the depletion in particles it entails is relatively more important for the low-flow-rate branch, which receives fluid that occupied less space in the inlet branch. The existence of such a particle free layer near the wall has long been observed in blood circulation and is called plasma skimming. More generally, it can be due to lateral migration towards the centre, which can be of inertial origin (high-Reynolds-number regime) (see Schonberg & Hinch 1989; Asmolov 2002; Eloot, De Bisschop & Verdonck 2004; Kim & Yoo 2008; Yoo & Kim 2010) or viscous origin. In such a low-Reynolds-number flow case, while a sphere does not migrate transversally due to symmetry and linearity in the Stokes equation, deformable objects such as vesicles (closed lipid membranes) (see Coupier *et al.* 2008; Kaoui *et al.* 2009), red blood cells (see Bagchi 2007; Secomb, Styp-Rekowska & Pries 2007), which exhibit dynamics similar to vesicles (see Abkarian, Faivre & Viallat 2007; Vlahovska, Podgorski & Misbah 2009), drops (see Mortazavi & Tryggvason 2000; Griggs, Zinchenko & Davis 2007) or elastic capsules (see Risso, Collé-Paillot & Zagzoule 2006; Bagchi 2007; Secomb *et al.* 2007) might adopt a shape that allows lateral migration. This migration is due to the presence of walls (see Olla 1997; Abkarian, Lartigue & Viallat 2002; Callens *et al.* 2008) as well as the non-constant shear rate (see Kaoui *et al.* 2008; Danker, Vlahovska & Misbah 2009). Even in the case where no migration occurs, the initial distribution is still not homogeneous: since the barycentre of particles cannot be closer to the wall than their radius, there is always some particle free layer near the walls. This sole effect will favour the high-flow-rate branch.

(iii) *Interactions between objects.* As illustrated by Ditchfield & Olbricht (1996) or Chesnutt & Marshall (2009), interactions between objects tend to smooth the asymmetry of the distribution, in that the second particle of a couple will tend to travel to the other branch from the first one. A related issue is the study of trains of drops or bubbles at a bifurcation, which completely obstruct the channels and whose

passage in the bifurcation greatly modifies the pressure distribution in its vicinity, and thus influences the behaviour of the following element (see Engl *et al.* 2005; Jousse *et al.* 2006; Schindler & Ajdari 2008; Sessoms *et al.* 2009).

In spite of the large body of literature on this subject, but perhaps because of the applicative purpose of most studies, the relative importance of these different parameters is seldom discussed quantitatively, although most authors are fully aware of the different phenomena at stake.

Since we focus here on the question of cross-streamline migration in the vicinity of the bifurcation, we will consider rigid spheres, for which no transverse migration in the upstream channel is expected. These spheres are in the vanishing concentration limit and flow through symmetric bifurcations, that is the symmetric Y-shaped and T-shaped bifurcations shown in figure 1, where the two daughter branches have the same cross-section and are equally distributed relative to the inlet channel.

Indeed, the case of rigid spheres is still quite unclear in the literature. In the following, first we briefly review previous studies that consider a geometrically symmetric situation and thoroughly re-analyse their results in order to determine whether the Zweifach–Fung effect they see is due to initial distribution or due to some attraction in the vicinity of the bifurcation, which generally has not been done (§ 2).

Then, we present in §§ 3 and 4 our two-dimensional simulations and quasi-two-dimensional experiments (in the sense that the movement of three-dimensional objects is planar). We mainly focus on the T-shaped bifurcation, in order to avoid as much as possible the geometric constraint due to the presence of an apex.

Our main result is that there is some attraction towards the low-flow-rate branch (§ 4.1). This result is then analysed and explained through basic fluid mechanics arguments, which are compared with those previously discussed in the literature.

Secondly, we discuss consequences of this drift on the final distribution in the daughter branches. To do so, we focus on particle concentrations possible at the outlets in the simplest case, where particles are homogeneously distributed in the inlet channel, with the sole (and unavoidable) constraint that they cannot approach the walls closer than their radius (termed *depletion effect* below, see figure 1*b*). This has been done through simulations, which allow us to easily control the initial distribution in particles (§ 4.2). Consequences for the potential efficiency of sorting or purification devices are discussed. We finally recall, in § 4.3, some previous studies from the literature for quantitative comparisons to check the consistency between them and our results.

Before discussing the results from the literature and presenting our data, we introduce useful common notation (see figure 1*b*).

The half-width of the inlet branch is set as the length scale of the problem. The inlet channel is divided into two branches of width $2a$ (the case $a=1$ is mainly considered here by default, unless otherwise stated) and spheres of radius $R \leqslant 1$. The flow rate at the inlet is denoted by Q_0, and Q_1 and Q_2 are the flow rates at the upper and lower outlets ($Q_0 = Q_1 + Q_2$). In the absence of particles, all the fluid particles situated initially above the line $y = y_f$ eventually enter branch 1. This line is called the (unperturbed) fluid separating streamline. Here y_0 is the initial transverse position of the considered particle long before it reaches the bifurcation ($|y_0| \leqslant 1 - R$). N_1 and N_2 are the numbers of particles entering branches 1 and 2 in unit time, while $N_0 = N_1 + N_2$ have entered the inlet channel. The volume fractions in the branches are $\Phi_i = V N_i / Q_i$, where V is the volume of a particle.

With this notation, we can reformulate our question: if $y_0 = y_f$, does the particle experience a net force in the y-direction (e.g. a pressure difference) that would push

it towards one of the branches, while a fluid particle would remain on the separating streamline (by definition of y_f)? If so, for which position y_0^* does this force vanish, so that the particle follows the streamlines and eventually hits the opposite wall and reaches an (unstable) equilibrium position? If $Q_1 \leqslant Q_2$ and $y_0^* < y_f$, then one will talk about *attraction towards the low-flow-rate branch.*

Following this notation, we have

$$N_1 = \int_{y_0^*}^{1} n(y) u_x^*(y)\, \mathrm{d}y, \tag{1.1}$$

$$Q_1 = \int_{y_f}^{1} u_x(y)\, \mathrm{d}y, \tag{1.2}$$

where $n(y)$ is the mean density of particles at height y in the inlet branch, and u_x^* and u_x are respectively particle and flow longitudinal upstream velocities. Note that N_0 and Q_0 are given by the same formula with $y_0^* = y_f = -1$.

The Zweifach–Fung effect can then be written as follows: if $Q_1/Q_0 < 1/2$ (branch 1 receives less flow than branch 2), then $N_1/N_0 < Q_1/Q_0$ (branch 1 receives even less particles than fluid) or equivalently $\Phi_1 < \Phi_0$ (the particle concentration decreases in the low-flow-rate branch).

2. Previous results in the literature

In the literature, the most common symmetric case that is considered is the Y-shaped bifurcation with daughter branches leaving the bifurcation with a 45° angle relative to the inlet channel, and cross-sections identical as those of the inlet channel (figure 1*a*) (see Audet & Olbricht 1987; Ditchfield & Olbricht 1996; Roberts & Olbricht 2003, 2006; Yang *et al.* 2006; Barber *et al.* 2008). The T-shaped bifurcation (figure 1*b*) has attracted little attention (see Yen & Fung 1978; Chien *et al.* 1985). All studies but Yen & Fung (1978) showed results for rigid spherical particles, while some results for deformable particles are given by Yen & Fung (1978) and Barber *et al.* (2008). Explicit data on a possible attraction towards one branch are scarce and can only be found in a recent paper dealing with two-dimensional simulations (see Barber *et al.* 2008). In three other papers, dealing with two-dimensional simulations (see Audet & Olbricht 1987) or experiments in square cross-sectional channels (see Roberts & Olbricht 2006; Yang *et al.* 2006), the output data are the concentrations Φ_i at the outlets. In this section, we re-analyse their data in order to discuss the possibility of an attraction towards one branch. Experiments in circular cross-sectional channels were also performed (see Yen & Fung 1978; Chien *et al.* 1985; Ditchfield & Olbricht 1996; Roberts & Olbricht 2003), on which we comment later in the text.

In the two-dimensional simulations presented by Audet & Olbricht (1987), some trajectories around the bifurcation are shown; however, the authors focused on an asymmetric Y-shaped bifurcation. In addition, some data for N_1/N_0 in a symmetric Y-shaped bifurcation and $R = 0.5$ are presented. Yang *et al.* (2006) performed experiments with balls of similar size ($R = 0.46$) in a symmetric Y-shaped bifurcation with a square cross-section and showed data for N_1/N_0 as a function of Q_1/Q_0 (see Yang *et al.* 2006). Experiments with larger balls ($R = 0.8$) in square cross-sectional channels were carried out by Roberts & Olbricht (2006). Once again, the output data are the ratios N_1/N_0. In both experiments, the authors made the assumption that the initial ball distribution is homogeneous, as also considered in a paper on simulation by Audet & Olbricht (1987). In these three papers, although the authors

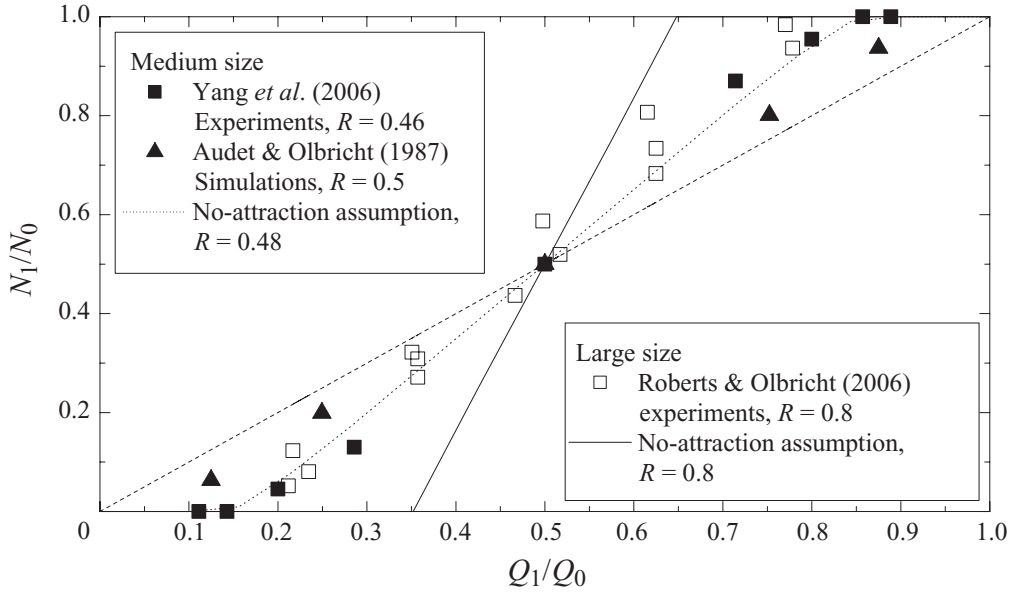


FIGURE 2. Comparison between data from the literature and theoretical distribution under the assumption of no attraction, which would indicate some previously unseen attraction towards the low-flow-rate branch. Rigid spheres distribution N_1/N_0 is shown as a function of the flow distribution Q_1/Q_0 in a symmetric Y-shaped bifurcation and a homogeneous distribution at the inlet (but the unavoidable *depletion effect*). Symbols: data extracted from previous papers: ■, Yang *et al.* (2006, figure 3), experiments, $R = 0.46$; ▲, Audet & Olbricht (1987, figure 8), two-dimensional simulations, $R = 0.5$; □, Roberts & Olbricht (2006, figure 5A), experiments, $R = 0.8$. Dotted and full lines, theoretical distribution for $R = 0.48$ and $R = 0.8$ in the case where the particles follow their underlying streamline ($y_0^* = y_f$: no-attraction assumption) and u_x^* given by our simulations; dashed line, fluid distribution ($N_1/N_0 = Q_1/Q_0$).

were sometimes conscious that the depletion and attraction effects might screen each other, the relative weight of each phenomenon has not been discussed. However, Yang *et al.* (2006) explicitly considered that there must be some *attraction towards the high-flow-rate branch* and gave some qualitative arguments for this. This opinion, initially introduced by Fung (see Fung 1973; Yen & Fung 1978; Fung 1993), is widespread in the literature (see El-Kareh & Secomb 2000; Jäggi *et al.* 2007; Kersaudy-Kerhoas *et al.* 2010). We shall come back to the underlying arguments in the following.

In figure 2, we present the data of N_1/N_0 as a function of Q_1/Q_0 taken from Audet & Olbricht (1987) for $R = 0.5$ (two-dimensional simulations), Yang *et al.* (2006) for $R = 0.46$ (experiments) and Roberts & Olbricht (2006) for $R = 0.8$ (experiments). It is instructive to compare these data with the corresponding values calculated with a very simple model based on the assumption that no particular effect occurs at the bifurcation, that is, the particles follow their underlying streamline (*no-attraction assumption*). To do so, we consider the two-dimensional case of flowing spheres and calculate the corresponding N_1 according to (1.1). The no-attraction assumption implies that $y_0^* = y_f$ and, as in the considered papers, the density $n(y)$ is considered constant for $|y| \leqslant 1 - R$. The particle velocity u_x^* is given by the simulations presented in § 4.2. Since we consider only flow ratios, this two-dimensional approach is a good enough approximation to discuss the results of the three-dimensional experiments, as the fluid separating plane is orthogonal to the plane where the channels lie; moreover, the position of this plane differs only by a few per cent from that of the separating line in two dimensions.

In all curves, it is seen that, if $Q_1/Q_0 < 1/2$, then $N_1/N_0 < Q_1/Q_0$, which is precisely the Zweifach–Fung effect. Note that this effect is present even under the no-attraction

assumption: as already discussed, the sole depletion effect is sufficient to favour the high-flow-rate branch.

Let us first consider spheres of medium size ($R \simeq 0.48$, Audet & Olbricht 1987 and Yang *et al.* 2006). If we compare the data from the literature with the theoretical curve found under the no-attraction assumption, we see that the enrichment of particles in the high-flow-rate branch is less pronounced than in the simulations by Audet & Olbricht (1987) and of the same order in the experiments by Yang *et al.* (2006). Therefore, we can assume that in the two-dimensional simulations by Audet & Olbricht (1987), there is an attraction towards the low-flow-rate branch, which lowers the enrichment of the high-flow-rate branch. The case of the experiments is less clear: it seems that no particular effect takes place.

The $R = 0.8$ case is even more striking: under the no-attraction assumption, we can see that for $Q_1/Q_0 < 0.35$, $N_1 = 0$ because $y_f > 1 - R$ and no sphere can enter the low-flow-rate branch. However, a non-negligible number of particles are found to enter branch 1 for $Q_1/Q_0 < 0.35$ by Roberts & Olbricht (2006) in their experiments (see figure 2). Thus, it is clear that there must be some attraction towards the low-flow-rate branch.

For channels with circular cross-sections, the data found in the literature do not all tell the same story, although spheres of similar sizes are considered. In Chien *et al.* (1985), $R = 0.79$ spheres are considered in a T-shaped bifurcation. The Y-shaped bifurcation was considered twice by the same research group, with very similar spheres: $R = 0.8$ (see Ditchfield & Olbricht 1996) and $R = 0.77$ (see Roberts & Olbricht 2003). In a circular cross-sectional channel, the plane orthogonal to the plane where the channels lie, parallel to the streamlines in the inlet channel and located at distance 0.78 from the inlet channel wall corresponds to the flow-separating plane for $Q_1/Q_0 = 0.32$. At low concentrations, very few spheres are observed in branch 1 for $Q_1/Q_0 < 0.32$ in Chien *et al.* (1985, figure 3D) and Ditchfield & Olbricht (1996, figure 3), in agreement with a no-attraction assumption. Chien *et al.* (1985) also showed that their data can be well described by the theoretical curve calculated by assuming that the particles follow their underlying streamlines. In marked contrast to these results, a considerable number of spheres are still observed in branch 1 in the same situation in Roberts & Olbricht (2003, figure 4). Similarly, in Ditchfield & Olbricht (1996, figure 4), many particles with $R = 0.6$ are found to enter the low-flow-rate branch 1 even when $Q_1/Q_0 < 0.19$, which would indicate some attraction towards the low-flow-rate branch. Thus, in a channel with circular cross-section, the results are contradictory. In the pioneering work of Yen & Fung (1978), a T-shaped bifurcation is also considered, with flexible disks mimicking red blood cells, but the deformability of these objects and the noise in the data do not allow us to make any reasonable comments.

More recently, Barber *et al.* (2008) have presented simulations of two-dimensional spheres with $R \leqslant 0.67$ and two-dimensional deformable objects mimicking red blood cells in a symmetric Y-shaped bifurcation. The values of y_0^* as a function of the flow rate ratios and the spheres radius are clearly discussed. For spheres, it is shown that $y_0^* < y_f$ if $Q_1 < Q_2$, that is, there is an *attraction towards the low-flow-rate branch*, which increases with R. Deformable particles are also considered. However, it is not possible to discuss from their data (or, probably, from any other data) whether the cross-streamline migration at the bifurcation is more important in this case or not: for deformable particles, transverse migration towards the centre occurs due to the presence of walls and non-homogeneous shear rates. This migration will probably screen the attraction effect, at least partly, and it seems difficult to quantify the relative

contribution of both effects. In particular, y_0^* depends on the (arbitrary) initial distance from the bifurcation. In Chesnutt & Marshall (2009), attraction towards the low-flow-rate branch is also quickly evoked, but considered as negligible since the focus was on large channels and interacting particles.

Finally, from our new analysis of previous results from the literature (and despite some discrepancies) it appears that there should be some attraction towards the low-flow-rate branch, although the final result is an enrichment of the high-flow-rate branch due to the depletion effect in the inlet channel. This effect was seen by Barber *et al.* (2008) in their simulations. On the other hand, if one considers the flow around an obstacle, as simulated in El-Kareh & Secomb (2000), it seems that spherical particles are attracted towards the high-flow-rate side.

From the above discussion, we conclude that the different effects occurring at the bifurcation level are neither well identified nor explained. Moreover, to date, no direct experimental proof of any attraction phenomenon exists. In § 4.1, we show experimentally that attraction towards the low-flow-rate branch takes place and confirm this through numerical simulations.

It is then necessary to discuss whether this attraction has important consequences on the final distributions in particles in the two daughter channels. This was not done explicitly in Barber *et al.* (2008); however, in § 4.2 through simulations we discuss the relative weight of the attraction towards the low-flow-rate branch and the depletion effect, which have opposite consequences.

3. Method

3.1. *Experimental set-up*

We studied the behaviour of hard balls as a first reference system. Since the potential migration across streamlines is linked to the way the fluid acts on the particles, we also studied spherical fluid vesicles. These are closed lipid membranes enclosing a Newtonian fluid. The lipids that we used are in liquid phase at room temperature, so that the membrane is a two-dimensional fluid. In particular, it is incompressible (so that spherical vesicles will remain spherical even under stress, unlike drops), but it is easily sheared: this means that a torque exerted by the fluid on the surface of the particle can imply a different response depending on whether it is a solid ball or a vesicle. Moreover, since vesicle suspensions are polydisperse, it is a convenient way to vary the radius R of the studied object.

The experimental set-up is a standard microfluidic chip made of polydimethylsiloxane bonded on a glass plate (figure 3). We wish to observe what happens to an object located around position y_f, that is, in which branch it goes at the bifurcation. In order to determine the corresponding y_0^*, we need to scan different initial positions around y_f. One solution would be to let a suspension flow and hope that some of the particles are close enough to the region of interest. In the meantime, as we shall see, the cross-streamline effect is weak and requires precise measurement, and noticeable effects appear only at high radius R, typically $R > 0.5$. Clogging is unavoidable with such objects, which would modify the flow rates ratio, and if a very dilute suspension is used, it is likely that the region of interest will only partly be scanned.

Therefore, we designed a microfluidic system that allowed us to use only one particle, which would go through the bifurcation with a controlled initial position y_0, would be taken back, its position y_0 modified, would flow again through the bifurcation, and so on. Moreover, we allowed continuous modification of the flow rate ratio between the two daughter branches. The core of the chip is the five branch

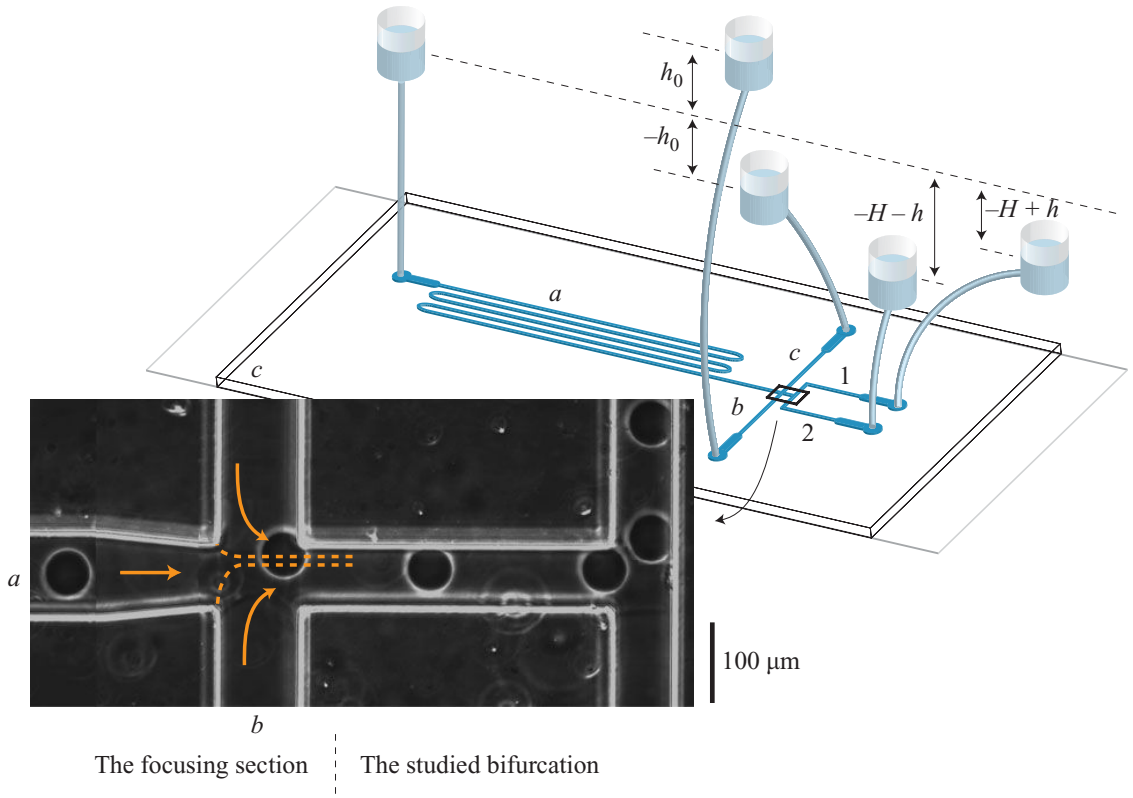


FIGURE 3. (Colour online) Scheme of the microfluidic device. The photograph shows the trajectory of a particle from branch a to branch 1 after having been focused on a given streamline thanks to flows from lateral branches b and c.

crossroad shown in the inset to figure 3. These five branches have different lengths and are linked to reservoirs placed at different heights, in order to induce flow by a hydrostatic pressure gradient. A focusing device (branches a–c) is placed before the bifurcation of interest (branches 1 and 2), in order to control the lateral position of the particle. Particles are initially located in the central branch a, where the flow is weak and the incoming particles are pinched between the two lateral flows. In order to modify the position y_0 of the particle, the relative heights of the reservoirs linked to the lateral branches are modified. The total flow rate and the flow rate ratios between the two daughter branches after the bifurcation are controlled by varying the heights of the two outlet reservoirs. Note that the flow rate ratio also depends on the heights of the reservoirs linked to inlet branches a, b and c. Since the latter two must be continuously modified to vary the position y_0 of the incoming particle in order to find y_0^* for a given flow rate ratio, it is convenient to place them on a pulley so that their mean height is always constant (resistances of branches b and c being equal). If the total flow rate is a relevant parameter (which is not the case here since we consider only Stokes flow of particles that do not deform), one can do the same with the two outlet reservoirs. In such a situation, if the reservoir of branch a is placed at height 0, reservoirs of branches b and c at heights $\pm h_0$, and reservoirs of branches 1 and 2 at heights $-H+h$ and $-H-h$, the flow rate ratio is governed by setting (h, H) and h_0 can be modified independently in order to control y_0. Once the particle has gone through the bifurcation, height H and the height of reservoir a are modified so that the particle comes back to branch a, and h_0 is modified in order to get closer and closer to the position y_0^*. Note that Q_1/Q_0 (or equivalently y_f) is a function of H, h, and the flow resistances of the five branches of rectangular cross-sections, which

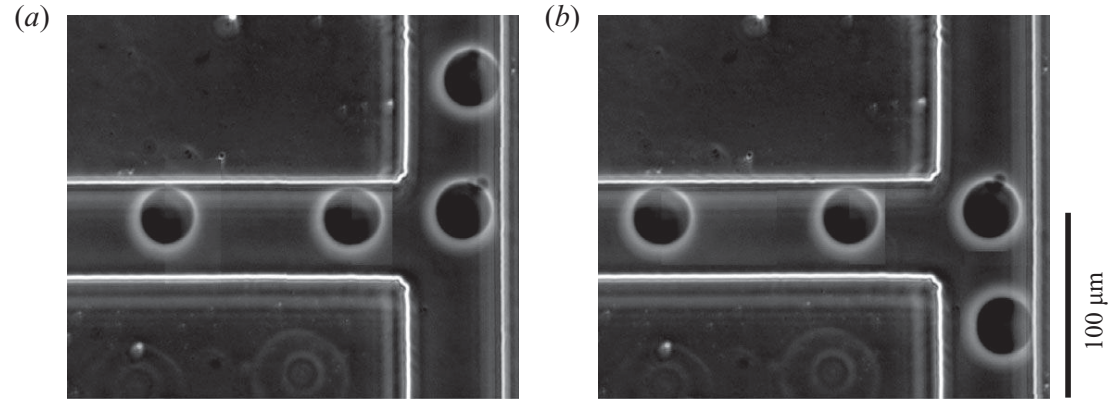


FIGURE 4. Photographs showing the different positions of a vesicle of radius $R = 0.60$ starting just above and just below its separating line. No clear difference between these two starting positions can be seen by eye, which illustrates the accuracy we get in the measurement of y_0^*. Here Q_1/Q_0 is set to 0.28.

are known functions of their lengths, widths and thicknesses (see White 1991). The accuracy of the calculation of this function was checked by measuring y_0^* for small particles, which must be equal to y_f.

Note that the length of the channel is much more important than the size of a single flowing particle, so that we can neglect the contribution of the latter in the resistance to the flow: hence, even though we control the pressures, we can consider that we work at fixed flow rates.

Finally, as can be seen in figure 4, our device allows us to scan very precisely the area of interest around the sought y_0^*, so that the uncertainty associated with it is very low.

At the bifurcation level, channels widths are all equal to $57 \pm 0.2\,\mu$m. Their thickness is $81 \pm 0.3\,\mu$m. We used polystyrene balls of maximum radius $40.5 \pm 0.3\,\mu$m in soapy water (therefore $R \leqslant 0.71$) and fluid vesicles of size $R \leqslant 0.60$. The vesicle membrane is a dioleoylphosphatidylcholine lipid bilayer enclosing an inner solution of sugar (sucrose or glucose) in water. Vesicles are produced following the standard electroformation method (see Angelova *et al.* 1992). Maximum flow velocity at the bifurcation level was around $1\,\text{mm s}^{-1}$, so that the Reynolds number $Re \simeq 10^{-1}$.

3.2. *The numerical model*

In the simulations, we focus on the two-dimensional problem (invariance along the z-axis). Our problem is a simple fluid/structure interaction and can be modelled by Navier–Stokes equations for the fluid flow and Newton–Euler equations for the sphere. These two problems can be coupled in a simple manner.

(i) The action of fluid on the sphere is modelled by the hydrodynamic force and torque acting on its surface. They are used as the right-hand sides of Newton–Euler equations.

(ii) The action of the sphere on fluid can be modelled by no-slip boundary conditions on the sphere (in the Navier–Stokes equations).

However, this explicit coupling can be numerically unstable and its resolution often requires very small time steps. In addition, as we have chosen to use the finite element method (FEM) (for accuracy) and since the position of the sphere evolves in time, we have to remesh the computational domain at each time step or in best cases every few time steps.

For these reasons, we chose another strategy to model our problem. Instead of using Newton–Euler equations for modelling the sphere motion and Navier–Stokes equations for the fluid flow, we use only the Stokes equations in the entire domain of the bifurcation (including the interior of the sphere). The use of Stokes equations is justified by the small Reynolds number in our case and the presence of the sphere is rendered by a second fluid with a 'huge' viscosity on which we impose a rigid body constraint. Such a strategy is widely used in the literature under different names, e.g. the so-called fluid particle dynamics (FPD) method (see Tanaka & Araki 2000; Peyla 2007), but we can group them generically as penalty-like methods. The method used here was mainly developed by Lefebvre *et al.* (see Janela, Lefebvre & Maury 2005; Lefebvre 2007) and we can find a mathematical analysis of such methods in Maury (2009).

In what follows, we briefly describe the basic ingredients of the FEM and the penalty technique applied to the problem.

The fluid flow is governed by Stokes equations written as follows:

$$-\nu\Delta\boldsymbol{u} + \nabla p = 0 \quad \text{in } \Omega_f, \tag{3.1}$$

$$\nabla\cdot\boldsymbol{u} = 0 \quad \text{in } \Omega_f, \tag{3.2}$$

$$\boldsymbol{u} = \boldsymbol{f} \quad \text{on } \partial\Omega_f, \tag{3.3}$$

where the variables

(i) ν, $\boldsymbol{u}$ and p are respectively the viscosity, velocity and pressure fields of the fluid;

(ii) Ω_f is the domain occupied by the fluid; typically $\Omega_f = \Omega\backslash\bar{B}$ if we denote by Ω the whole bifurcation and by B the rigid particle;

(iii) $\partial\Omega_f$ is the border of Ω_f;

(iv) $\boldsymbol{f}$ is some given function for the boundary conditions.

It is known that under some reasonable assumptions the problem (3.1)–(3.3) has a unique solution $(\boldsymbol{u}, p) \in H^1(\Omega_f)^2 \times L^2_0(\Omega_f)$ (see Girault & Raviart 1986). Below, we use the following functional spaces:

$$L^2(\Omega) = \left\{ f : \Omega \to \mathbb{R}; \int_\Omega |f|^2 < +\infty \right\}, \tag{3.4}$$

$$L^2_0(\Omega) = \left\{ f \in L^2(\Omega); \int_\Omega f = 0 \right\}, \tag{3.5}$$

$$H^1(\Omega) = \{ f \in L^2(\Omega); \nabla f \in L^2(\Omega) \}, \tag{3.6}$$

$$H^1_0(\Omega) = \{ f \in H^1(\Omega); f = 0 \text{ on } \partial\Omega \}. \tag{3.7}$$

As we use the FEM for the numerical resolution of the problem (3.1)–(3.3), we need to rewrite this in a variational form (an equivalent formulation of the initial problem). For the sake of simplicity, we start by writing it in a standard way (fluid without sphere), then we modify it using the penalty technique to take into account the presence of the particle. In what follows, we briefly describe these two methods, the standard variational formulation for the Stokes problem and the penalty technique.

3.2.1. *Variational formulation*

First recall the deformation tensor $\boldsymbol{\tau}$, which is useful in what follows

$$\boldsymbol{\tau}(\boldsymbol{u}) = \tfrac{1}{2}\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^t\right). \tag{3.8}$$

Thanks to the incompressibility constraint $\nabla \cdot \boldsymbol{u} = 0$, we have

$$\Delta \boldsymbol{u} = 2\nabla \cdot \boldsymbol{\tau}(\boldsymbol{u}). \tag{3.9}$$

Hence, the problem (3.1)–(3.3) can be rewritten as follows: find $(\boldsymbol{u}, p) \in H^1(\Omega_f)^2\, L_0^2(\Omega_f)$ such that

$$-2\nu\nabla \cdot \boldsymbol{\tau}(\boldsymbol{u}) + \nabla p = 0 \quad \text{in } \Omega_f, \tag{3.10}$$

$$\nabla \cdot \boldsymbol{u} = 0 \quad \text{in } \Omega_f, \tag{3.11}$$

$$\boldsymbol{u} = \boldsymbol{f} \quad \text{on } \partial\Omega_f. \tag{3.12}$$

By simple calculations (see the Appendix for details) we show that problem (3.10)–(3.12) is equivalent to the following: find $(\boldsymbol{u}, p) \in H^1(\Omega_f)^2 \times L_0^2(\Omega_f)$ such that

$$2\nu \int_{\Omega_f} \boldsymbol{\tau}(\boldsymbol{u}) : \boldsymbol{\tau}(\boldsymbol{v}) - \int_{\Omega_f} p\nabla \cdot \boldsymbol{v} = 0, \quad \forall \boldsymbol{v} \in H_0^1(\Omega_f)^2, \tag{3.13}$$

$$\int_{\Omega_f} q\nabla \cdot \boldsymbol{u} = 0, \quad \forall q \in L_0^2(\Omega_f), \tag{3.14}$$

$$\boldsymbol{u} = \boldsymbol{f} \quad \text{on } \partial\Omega_f, \tag{3.15}$$

where '$:$' denotes the double contraction.

3.2.2. *Penalty method*

We chose to use the penalty strategy in the framework of FEM, described here briefly (see Janela *et al.* 2005 and Lefebvre 2007 for more details).

The first step consists in rewriting the variational formulation (3.13)–(3.15) by replacing the integrals over the real domain occupied by the fluid ($\Omega_f = \Omega \setminus \bar{B}$) by those over the whole domain Ω (including the sphere B). This means that we extend the solution $(\boldsymbol{u}, p)$ to the whole domain Ω. More precisely, by the penalty method we replace the particle by an artificial fluid with huge viscosity. This has been made possible by imposing a rigid-body motion constraint on the fluid that replaces the sphere ($\boldsymbol{\tau}(\boldsymbol{u}) = 0$ in B). Obviously, the divergence-free constraint is also ensured in B.

The problem (3.13)–(3.15) is then modified as follows: find $(\boldsymbol{u}, p) \in H^1(\Omega)^2\, L_0^2(\Omega)$ such that

$$2\nu \int_{\Omega} \boldsymbol{\tau}(\boldsymbol{u}) : \boldsymbol{\tau}(\boldsymbol{v}) + \frac{2}{\varepsilon} \int_B \boldsymbol{\tau}(\boldsymbol{u}) : \boldsymbol{\tau}(\boldsymbol{v}) - \int_{\Omega} p\nabla \cdot \boldsymbol{v} = 0, \quad \forall \boldsymbol{v} \in H_0^1(\Omega)^2, \tag{3.16}$$

$$\int_{\Omega} q\nabla \cdot \boldsymbol{u} = 0, \quad \forall q \in L_0^2(\Omega), \tag{3.17}$$

$$\boldsymbol{u} = \boldsymbol{f} \quad \text{on } \partial\Omega, \tag{3.18}$$

where $\varepsilon \ll 1$ is a given penalty parameter.

Finally, if we denote the time discretization parameter by $t_n = n\delta t$, the velocity and the pressure at time t_n by $(\boldsymbol{u}_n, p_n)$, the velocity of the sphere at time t_n by $\boldsymbol{V}_n$ and its centre position by $\boldsymbol{X}_n$, we can write our algorithm as

$$\boldsymbol{V}_n = \frac{1}{\text{volume}(B)} \int_B \boldsymbol{u}_n, \tag{3.19}$$

$$\boldsymbol{X}_{n+1} = \boldsymbol{X}_n + \delta t \boldsymbol{V}_n, \tag{3.20}$$

where $(\boldsymbol{u}_{n+1}, p_{n+1})$ solves

$$2\nu \int_{\Omega} \tau(\boldsymbol{u}_{n+1}) : \tau(\boldsymbol{v}) + \frac{2}{\varepsilon} \int_{B} \tau(\boldsymbol{u}_{n+1}) : \tau(\boldsymbol{v}) - \int_{\Omega} p_{n+1} \nabla \cdot \boldsymbol{v} = 0, \quad \forall \boldsymbol{v} \in H_0^1(\Omega)^2, \tag{3.21}$$

$$\int_{\Omega} q \nabla \cdot \boldsymbol{u}_{n+1} = 0, \quad \forall q \in L_0^2(\Omega), \tag{3.22}$$

$$\boldsymbol{u}_{n+1} = \boldsymbol{f} \quad \text{on } \partial\Omega. \tag{3.23}$$

The implementation of algorithm (3.19)–(3.23) is done with a user-friendly finite element software Freefem++ (see Hecht & Pironneau 2010).

Finally, we consider the bifurcation geometry shown in figure 1(*b*) and impose no-slip boundary conditions on all walls and prescribe parabolic velocity profiles at the inlets and outlets such that, for a given choice of flow rate ratio, $Q_0 = Q_1 + Q_2$. For a given initial position y_0 of the sphere of radius R at the outlet, the full trajectory is calculated until it definitely enters one of the daughter branches. A dichotomy algorithm is used to determine the key position y_0^*. Spheres of radius R up to 0.8 are considered.

REMARK 1. *In practice, the penalty technique may deteriorate the pre-conditioning of our underlying linear system. To overcome this problem, one can regularize (3.22) by replacing it with the following:*

$$-\varepsilon_0 \int_{\Omega} p_{n+1} q + \int_{\Omega} q \nabla \cdot \boldsymbol{u}_{n+1} = 0, \quad \forall q \in L_0^2(\Omega), \tag{3.24}$$

where $\varepsilon_0 \ll 1$ *is a given parameter.*

4. Results and discussion

4.1. *The cross-streamline migration*

4.1.1. *The particle-separating streamlines*

In figure 5, we show the position of the particle-separating line y_0^* relative to the position of the fluid-separating line y_f when branch 1 receives less fluid than branch 2 (see figure 1*b*), which is the main result of this paper. For all particles considered, in the simulations or in the experiments, we find that the particle-separating line lies below the fluid-separating line, the upper branch being the low-flow-rate branch. These results clearly indicate an attraction towards the low-flow-rate branch: while a fluid element located below the fluid-separating streamline will enter into the high-flow-rate branch, a solid particle can cross this streamline and enter into the low-flow-rate branch, provided it is not too far away initially. It is also clear that the attraction increases with the sphere radius R.

In particular, in the experiments (figure 5*a*), particles of radius $R \lesssim 0.3$ behave like fluid particles. Note that $R = 0.52$ balls show a slight attraction towards the low-flow-rate branch, while the effect is more marked for big balls of radius $R = 0.71$. Vesicles show a comparable trend and it seems from our data that solid particles or vesicles with fluid membrane behave similarly in the vicinity of the bifurcation.

In the simulations (figure 5*b*), we clearly see that for a given R, the discrepancy between the fluid and particle behaviour increases when Q_1/Q_0 decreases. On the contrary, in the quasi-two-dimensional case of the experiments, the difference between

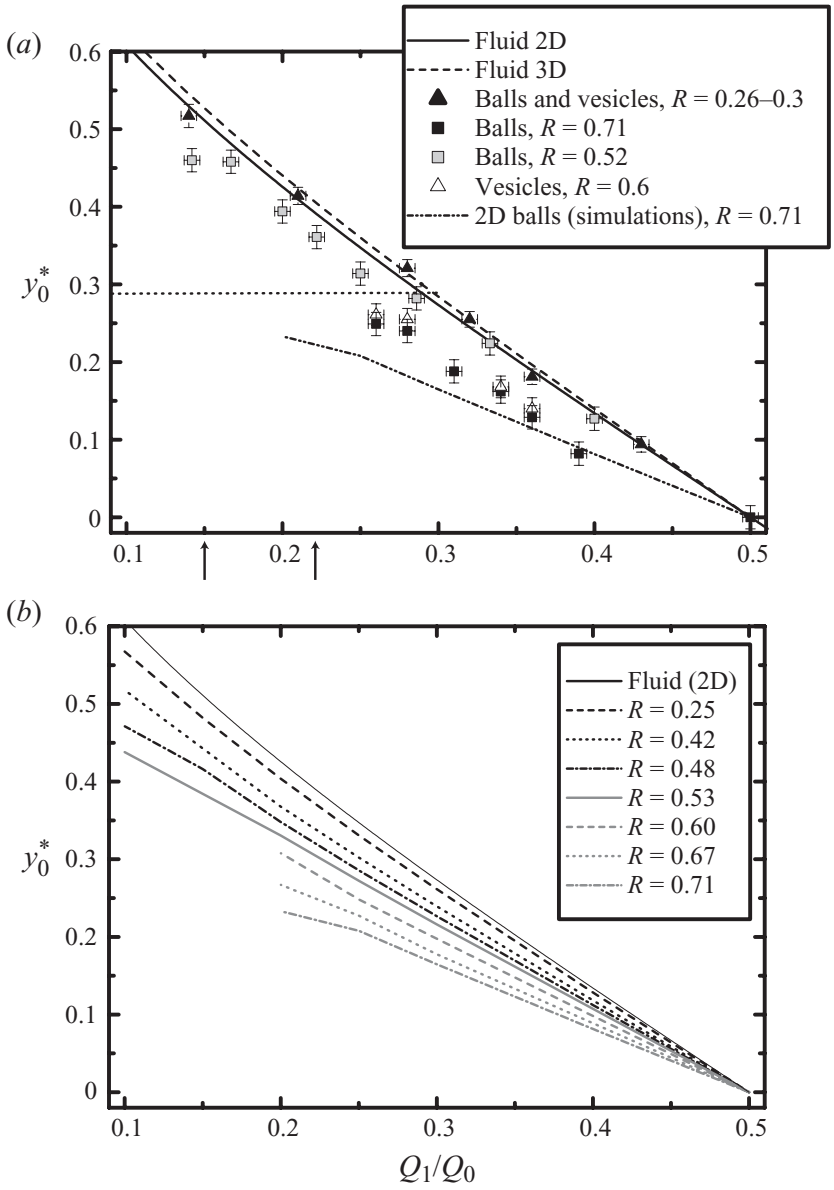


FIGURE 5. Position of the particle-separating line y_0^*. The T-bifurcation with branches of equal widths is considered. Branch 1 receives flow from high y values, so $y_0^* < y_f$ for $Q_1/Q_0 < 1/2$ indicates attraction towards the low-flow-rate branch (see also figure 1*b*). (*a*) Data from quasi-two-dimensional experiments and comparison with the two-dimensional case for one particle size. The two-dimensional and three-dimensional fluid-separating lines are shown to illustrate the low discrepancy between the two cases, as required to validate our new analysis of the literature in § 2. The horizontal dotted line shows the maximum position $y_0 = 1 - R$ for $R = 0.71$ spheres. Its intersection with the curve $y_0^*(Q_1/Q_0)$ yields the critical flow rate ratio Q_1/Q_0 below which no particle enters branch 1, the low-flow-rate branch. These expected critical flow rates for the two- and three-dimensional cases are shown by arrows. (*b*) Data from two-dimensional simulations.

the flow and the particle streamlines seems to be rather constant in a wide range of Q_1/Q_0 values. Finally, for small enough values of Q_1/Q_0, the attraction effect is more pronounced in the two-dimensional case than in the quasi-two-dimensional case, as shown in figure 5(*a*) for $R = 0.71$. This result was expected, since this effect has something to do with the non-zero size of the particle, and the real particle to channel size ratio is lower in the experiments for a given R, due to the third dimension. In all cases, below a given value of Q_1/Q_0, the critical position y_0^* would enter the depletion zone $y_0 > 1 - R$, so that no particle will eventually enter the low-flow-rate branch.

The corresponding critical Q_1/Q_0 is much lower in the two-dimensional case than in the experimental quasi-two-dimensional situation (see figure 5*a*).

4.1.2. *Discussion*

The first argument for some attraction towards one branch was initially given by Fung (see Fung 1973; Yen & Fung 1978; Fung 1993) and strengthened by recent simulations (see Yang & Zahn 2004): a sphere in the middle of the bifurcation is considered ($y_0 = 0$) and it is argued that it should go to the high-flow-rate branch since the pressure drop $P_0 - P_2$ is higher than $P_0 - P_1$ because $Q_2 > Q_1$ (see figure 1*b* for notation). This is true (we also found $y_0^* > 0$ when $Q_1 < Q_2$) but this is not the point to be discussed: if one wishes to discuss the increase in volume fraction in branch 2, therefore to compare the particles and fluid fluxes N_2 and Q_2, one needs to focus on particles in the vicinity of the fluid-separating streamline (to see whether or not they behave like the fluid) and not in the vicinity of the middle of the channel. On the other hand, this incorrectly formulated argument by Fung has led to the idea that there must be some attraction towards the high-flow-rate branch in the vicinity of the fluid-separating streamline (see Yang *et al.* 2006), which appears now in the literature as a well-established fact (see Jäggi *et al.* 2007; Kersaudy-Kerhoas *et al.* 2010).

In Barber *et al.* (2008), Fung's argument has been rejected, although it is not explained why. Arguments for attraction towards the low-flow-rate branch (that is, $P_2 > P_1$ in figure 1*b*) are given, considering particles in the vicinity of the fluid separating streamline. The authors' main idea was, first, that some pressure difference $P_0 - P_i$ builds up on each side of the particle because it travels more slowly than the fluid. Then, as the particle intercepts a relatively more important area in the low-flow-rate branch region ($y_f < y < 1$) than in the high-flow-rate region, they consider that the pressure drop is more important in the low-flow-rate region, so that $P_2 > P_1$. The authors called this effect 'daughter vessel obstruction'.

However, it is not clear in Barber *et al.* (2008) where the particles must be for this argument to be valid: they could be at the entrance of the bifurcation, in the middle of it or close to the opposite wall, since their arguments are used to explain what happens in the case of daughter branches of different widths. Indeed, we shall see that the effects can be quite different according to this position and, furthermore, the notion of 'relatively larger part intercepted' is not the key phenomenon to understand the final attraction towards the low-flow-rate branch, even though it clearly contributes to it.

To understand this, let us focus on the simulated trajectories starting around y_0^* shown in figure 6(*a*) ($R = 0.67$, $Q_1/Q_0 = 0.2$). These trajectories must be analysed in comparison with the unperturbed flow streamlines, in particular the fluid separating streamline, starting at $y = y_f$ and ending against the front wall at a stagnation point.

Particles starting around $y_0^* < y_f$ show a clear attraction towards the low-flow-rate branch (displacement along the y-axis) as they enter the bifurcation. More precisely, there are three types of motions: for low initial position y_0 (in particular $y_0 = 0$), particles travel directly into the high-flow-rate branch. Similarly, above y_0^*, the particles travel directly into the low-flow-rate branch. Between some $y_0^{**} > 0$ and y_0^*, the particles first move towards the low-flow-rate branch, but finally enter the high-flow-rate branch: the initial attraction towards the low-flow-rate branch becomes weaker and the particles eventually follow the streamlines entering the high-flow-rate branch. This non-monotonic variation of y_0 for a particle starting just below y_0^* is also seen in experiments, as shown in figure 4(*b*): the third position of the vesicle is

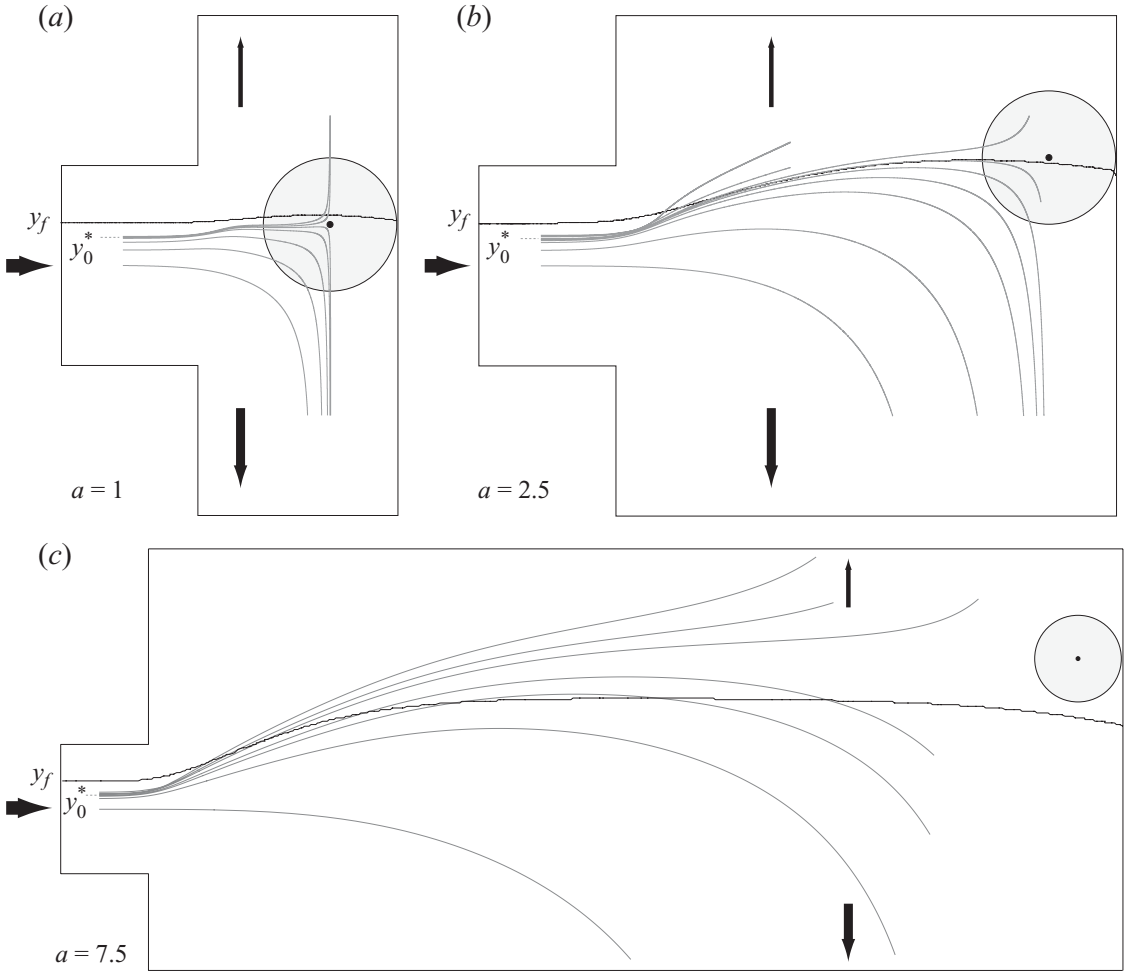


FIGURE 6. Comparison between fluid and particle trajectories in the vicinity of the bifurcation. Grey lines, some trajectories of an $R = 0.67$ particle when $Q_1/Q_0 = 0.2$ for (a) branches of equal widths, (b) daughter branches 2.5 times wider than the inlet branch and (c) daughter branches 7.5 times wider than the inlet branch. The unperturbed fluid separating streamline starting at $y = y_f$ is shown in black. The particle is shown approximately at its stagnation point.

characterized by a y_0 slightly higher than the initial one. Returning to the simulations, note that, at this level, there is still some net attraction towards the low-flow-rate branch: the particle stagnation point near the opposite wall is still below the fluid-separating streamline (that is, on the high-flow-rate side). This two-step effect is even more visible when the width $2a$ of the daughter branches is increased, so that the entrance of the bifurcation is far from the opposite wall, as shown in figure 6(b,c). The second attraction is, in such a situation, more dramatic: for $a = 7.5$, the particle stagnation point is even on the other side of the fluid-separating streamline, that is, there is some attraction towards the high-flow-rate branch. Thus, there are clearly two antagonistic effects along the trajectory. In the first case of branches of equal widths, where the opposite wall is close to the bifurcation entrance, the second attraction towards the high-flow-rate branch coexists with the attraction towards the low-flow-rate branch and finally only diminishes it.

These two effects occur in two very different situations. At the entrance of the channel, an attraction effect must be understood in terms of streamlines crossing: does a pressure difference build up orthogonally to the main flow direction? Near the opposite wall, the flow is directed towards the branches and being attracted means

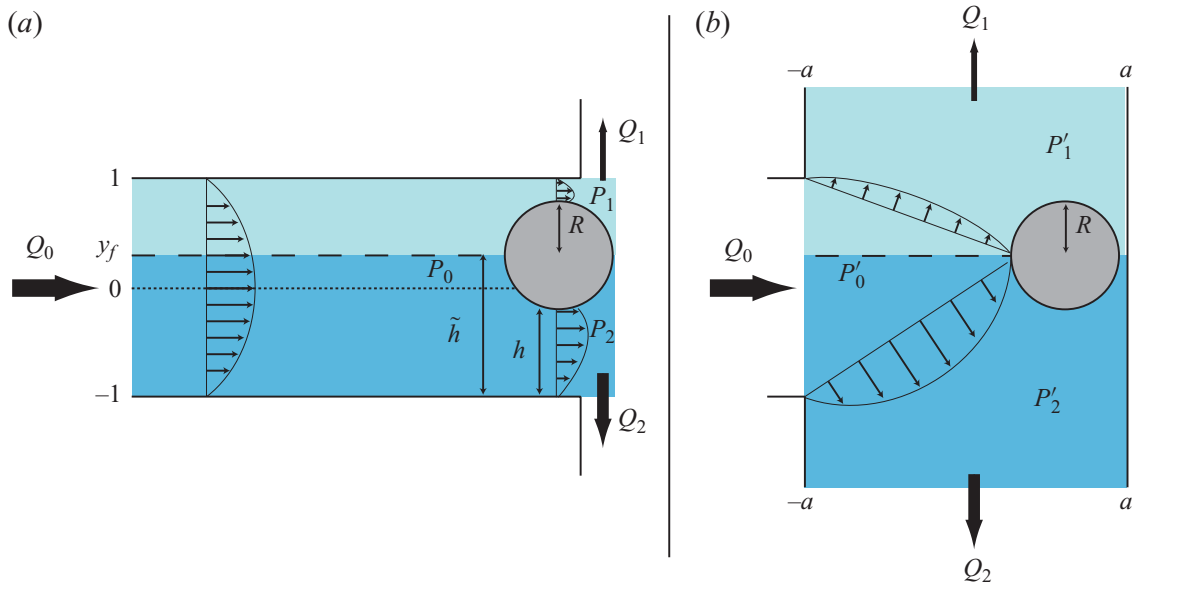


FIGURE 7. (Colour online) Schematic of the geometry considered for the two effects occurring in the bifurcation. (*a*) Entrance to the bifurcation showing attraction towards the low-flow-rate branch ($P_1 < P_2$). (*b*) Opposite wall showing attraction towards the high-flow-rate branch ($P_1' > P_2'$).

flowing upstream or downstream. In both cases, in order to discuss whether some pressure difference builds up or not, the main feature is that, in a two-dimensional Stokes flow between two parallel walls, the pressure difference between two points along the flow direction scales like $\Delta P \propto Q/h^3$, where Q is the flow rate and h is the distance between the two walls. This scaling is sufficient to discuss in a first-order approach the two effects at stake.

The second effect is the simplest one: the sphere is placed in a quasi-elongational, but asymmetric, flow. As shown in figure 7(*b*), around the flow stagnation point, the particle movement is basically controlled by the pressure difference $P_2' - P_1'$, which can be written $(P_0' - P_1') - (P_0' - P_2')$. Focusing on the y-component of the velocity field, which becomes all the more important as a is larger than 1, we have $P_0' - P_i' \propto Q_i/(a-R)^3$. Around the flow stagnation point, the pressure difference $P_2' - P_1'$ has then the same sign as $Q_1 - Q_2$ and is thus negative, which indicates attraction towards the high-flow-rate branch. For wide daughter branches, when this effect is not screened by the first one, this implies that the stagnation point for particles is above the fluid separating line, as seen in figure 6(*c*). The argument that we use here is similar to that introduced by Fung (see Fung 1993; Yang *et al.* 2006) but resolves only one part of the problem. Following these authors, it can also be pointed out that the shear stress on the sphere is non-zero: in a two-dimensional Poiseuille flow of width h, the shear rate near a wall scales as Q/h^2, so the net shear stress on the sphere is directed towards the high-flow-rate branch, making the sphere roll along the opposite wall towards this branch.

Finally, this situation is similar to that of a flow around an obstacle, which was considered by El-Kareh & Secomb (2000) as a model situation to understand what happens at the bifurcation. Indeed, the authors found that spheres are attracted towards the high-velocity side of the obstacle. However, we show here that this modelling is misleading, as it neglects the first effect, which eventually governs the net effect.

This first effect leads to an attraction towards the low-flow-rate branch. To understand this, let us consider a sphere located in the bifurcation with transverse position $y_0 = y_f$. The exact calculation of the flow around it is much too complicated, and simplifications are needed. Just as we considered the large a case to understand the second mechanism eventually leading to attraction towards the high-flow-rate branch, let us consider the small a limit to understand the first effect: as soon as the ball enters the bifurcation, it hits the front wall. On each side, we can write in a first approximation that the flow rate between the sphere and the wall scales as $Q \propto \Delta P h^3$, where $\Delta P = P_0 - P_i$ is the pressure difference between the back and the front of the sphere, and h is the distance between the sphere and the wall (see figure 7*a*).

Since the ball touches the front wall, the flow rate Q is either Q_1 or Q_2 and is, by definition of y_f, the integral of the unperturbed Poiseuille flow velocity between the wall and the $y = y_f$ line, so $Q \propto \tilde{h}^2 - \tilde{h}^3/3$, where $\tilde{h} = 1 \pm y_f$ (see figure 7*a* for notation).

We have then, on each side,

$$\Delta P \propto \frac{\tilde{h}^2 - \tilde{h}^3/3}{h^3}. \tag{4.1}$$

To make things clear, let us consider the extreme case of a flat particle: $h = \tilde{h}$. Then, $\Delta P \propto 1/\tilde{h} - 1/3$ is a decreasing function of $\tilde{h}$, that is, a decreasing function of Q. Therefore, the pressure drop is more important on the low-flow-rate side, and finally $P_1 < P_2$: there is an attraction towards the low-flow-rate branch. This is exactly the opposite result from the simple view claiming that there is some attraction towards the high-flow-rate branch since ΔP scales as Q/h^3 so as Q. Since one has to discuss what happens for a sphere in the vicinity of the separating line, Q and $\tilde{h}$ are not independent. This is the key argument. Note finally that there is no need for some obstruction arguments to build up a different pressure difference on each side. It only increases the effect since the function $\tilde{h} \mapsto (\tilde{h}^2 - \tilde{h}^3/3)/(\tilde{h} - R)^3$ decreases faster than the function $\tilde{h} \mapsto (\tilde{h}^2 - \tilde{h}^3/3)/\tilde{h}^3$. One can be even more precise and take into account the variations in the gap thickness as the fluid flows between the sphere and the wall to calculate the pressure drop by lubrication theory. Still, it is found that ΔP is a decreasing function of $\tilde{h}$.

In the more realistic case $a \simeq 1$, the flow repartition becomes more complex, and the particle velocity along the x-axis is not zero. Yet, as it reaches a low-velocity area (the velocity along the x-axis of the streamline starting at y_f drops to zero), its velocity is lower than its velocity at the same position in a straight channel. In addition, as the flow velocities between the sphere and the opposite wall are low, and since the fluid located e.g. between y_f and the top wall will eventually enter the top branch by definition, we can assume that it will mainly flow between the sphere and the top wall. Note that this is not true in a straight channel: there are no reasons for the fluid located between one wall and the $y = y_0$ line, where y_0 is the sphere lateral position, to enter completely, or to be the only fluid to enter, between the wall and the particle. Therefore, we can assume that the arguments proposed to explain the attraction towards the low-flow-rate branch remain valid, even though the net effect will be weaker.

Note, finally, that contrary to what we discussed for the second effect, the particle rotation probably plays a minor role here, as in this geometry the shear stress exerted by the fluid on the particle will mainly result in a force acting parallel to the x-axis.

Finally, this separation into two effects can be used to discuss a scenario for bifurcations with channels of different widths: if the inlet channel is broadened, the first effect becomes less strong while the second one is not modified, which results in a weaker attraction towards the low-flow-rate branch. If the outlet channels are broadened, as in figure 6(b,c), it becomes more subtle. Let us start again with the second effect (migration upstream or downstream) before the first effect (transverse migration). As seen in figure 6, the position of the particle stagnation point (relative to the flow-separating line) is an increasing function of a, so the second effect is favoured by the broadening of the outlets: for $a \to \infty$, we end up with the problem of flow around an obstacle, while for small a, one cannot write that the width of the gap between the ball and the wall is just $a - R$, therefore independent from Q_i, as it also depends on the y-position of the particle relative to y_0'. In other words, in such a situation, the second effect is screened by the first effect. On the other hand, as a increases, the distance available for transverse migration becomes larger, which could favour the first effect, although the slowdown of the particle at the entrance of the bifurcation becomes less pronounced.

Finally, it appears difficult to predict the consequences of an outlet broadening: for instance, in our two-dimensional simulations presented in figure 6 ($R = 0.67$, $Q_1/Q_0 = 0.2$), y_0^* varies from 0.27 when the outlet half-width a is equal to 1, to 0.31 when a is equal to 2.5 and drops down to 0.22 for $a = 7.5$. Note that the net effect is always an attraction towards the low-flow-rate branch ($y_0^* < y_f$).

For daughter branches of different widths, it was illustrated in Barber *et al.* (2008) that the narrower branch is favoured. This can be explained through the second effect (see figure 7b): the pressure drop $P_0' - P_i' \propto Q_i/(a - R)^3$ increases when the channel width decreases, which favours the narrower branch even in the case of equal flow rates between the branches.

4.2. *The consequences for the final distribution*

As there is some attraction towards the low-flow-rate branch, we could expect some enrichment of the low-flow-rate branch. However, as already discussed, even in the most uniform situation, the presence of a free layer near the walls will favour the high-flow-rate branch. We discuss now, through our simulations, the final distribution that results from these two antagonistic effects.

As in most previous papers in the literature, we focus on the case of uniform number density of particles in the inlet ($n(y) = 1$ in (1.1)). In order to compute the final splitting N_1/N_0 of the incoming particles as a function of flow rate ratio Q_1/Q_0, one needs to know, according to (1.1), the position y_0^* of the particle-separating line and the velocity u_x^* of the particles in the inlet channel. From figure 5, we see that y_0^* depends roughly linearly on (Q_1/Q_0–1/2), so we will consider a linear fit of the calculated data in order to get values for all Q_1/Q_0. The longitudinal velocity u_x^* was computed for all studied particles as a function of transverse position y_0. As shown in figure 8, the function $u_x^*(y_0)$ is well described by a quartic function $u_x^*(y_0) = \alpha y_0^4 + \beta y_0^2 + \gamma$, which is an approximation also used in Barber *et al.* (2008). Values for the fitting parameters for this velocity profile and for the linear relationship $y_0^* = \xi \times (Q_1/Q_0 - 1/2)$ are given in table 1.

The evolution of N_1/N_0 as a function of Q_1/Q_0 for two-dimensional rigid spheres is shown in figure 9 for two representative radii. By symmetry, considering $Q_1 < Q_2$ is sufficient. In order to discuss the enrichment of particles in the high-flow-rate branch (branch 2 then), it is also convenient to consider directly the volume fraction variation $\Phi_2/\Phi_0 = (N_2/Q_2)/(N_0/Q_0)$.

R	0	0.25	0.42	0.48	0.53	0.60	0.67	0.71	0.80
α	0	−0.96	−3.45	−4.77	−6.33	−9.52	−11.6	−12.6	–
β	−1	−0.85	−0.65	−0.70	−0.64	−0.61	−0.71	−0.73	–
γ	1	0.96	0.91	0.89	0.87	0.84	0.81	0.79	0.75
ξ	–	−1.35	−1.25	−1.17	−1.09	−1.01	−0.90	−0.81	–

TABLE 1. Values for the fitting parameters (α, β, γ) for the longitudinal velocity $u_x^*(y_0) = \alpha y_0^4 + \beta y_0^2 + \gamma$ of a two-dimensional sphere of radius R in a Poiseuille flow of imposed velocity at infinity $u_x(y) = 1 - y^2$; for $R = 0.80$, the velocity profile is too flat to be reasonably fitted by a three-parameter law, since all velocities are equal to 0.75 ± 0.005 in the explored interval $y_0 \in [-0.15; 0.15]$. We also give the values for the fitting parameter ξ of the linear relationship between the particle separating line position y_0^* and flow rate ratios: $y_0^* = \xi \times (Q_1/Q_0 - 1/2)$. For $R = 0.8$, the strong confinement leads to numerical problems as the sphere approaches the walls.

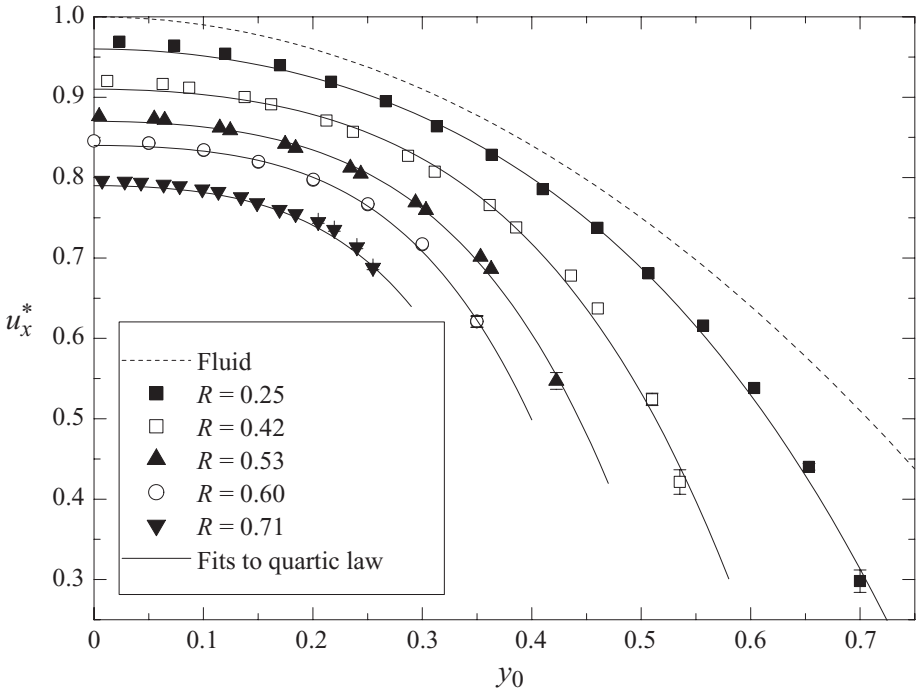


FIGURE 8. Some longitudinal velocity profiles $u_x^*(y_0)$ for two-dimensional spheres of different radii R in the inlet channel, where a Poiseuille flow of velocity $u_x(y) = 1 - y^2$ is imposed at infinity. The full lines show the fits by the quartic law $u_x^*(y_0) = \alpha y_0^4 + \beta y_0^2 + \gamma$.

When $Q_1/Q_0 = 1/2$, the particle flow splits equally into the two branches: $N_1 = N_0$ and $\Phi_2 = \Phi_0$. For all explored sizes of spheres, when the flow rate in branch 1 decreases, there is an enrichment of particles in branch 2, which is precisely the Zweifach–Fung effect: $N_1/N_0 < Q_1/Q_0$ or $\Phi_2/\Phi_0 > 1$. Then, even in the most homogeneous case, the attraction towards the low-flow-rate branch is not strong enough to counterbalance the depletion effect that favours the high-flow-rate branch. However, this attraction effect cannot be considered as negligible, in particular for large particles: while, when the particles follow their underlying fluid streamline, the maximum enrichment in the high-flow-rate branch would be around 40 % for $R = 0.71$, it drops down to less than 17 % in reality. Similarly, the critical flow rate ratio Q_1/Q_0 below which no particle enters into branch 1 is greatly shifted: from around 0.29 to around 0.15 for $R = 0.71$. For smaller spheres ($R = 0.42$), this asymmetry in the distribution between the two

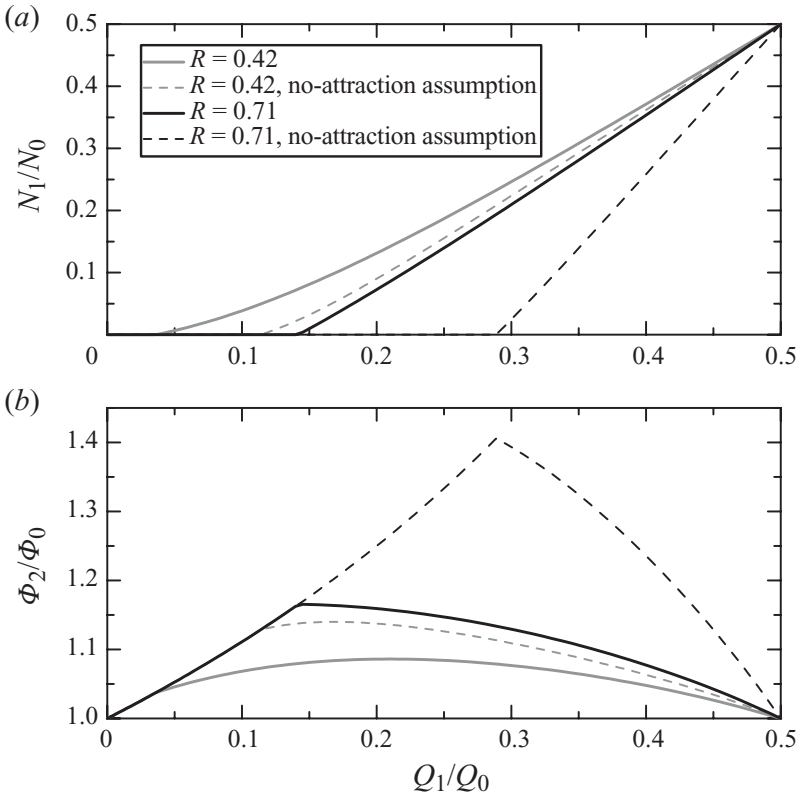


FIGURE 9. Final particle distributions. Full lines, spheres' relative distribution N_1/N_0 (a) and volume fraction Φ_2/Φ_0 (b) as a function of the flow rate distribution Q_1/Q_0 from our two-dimensional simulations for two representative radii $R = 0.42$ and $R = 0.71$. The curves are straightforwardly derived from (1.1) and (1.2) and computed values of y_0^* and u_x^* (table 1). The results are compared with the hypothesis where the particles would follow the streamlines ($y_0^* = y_f$) (dashed lines).

branches is weak: while the maximum enrichment in the high-flow-rate branch would be around 15 % in a no-attraction case, it drops to less than 8 % due to the attraction towards the low-flow-rate branch.

When the flow Q_1 is equal to zero or $Q_0/2$, Φ_2 is equal to Φ_0; thus, there is a maximal enrichment for some flow rate ratio between 0 and 1/2. Starting from $Q_1/Q_0 = 0.5$, the increase in Φ_2 with the decrease of Q_1 is mainly due to the decrease of the relative importance of the free layer near the wall on the side of branch 2. Two mechanisms are then responsible for the decrease of Φ_2 when Q_1 goes on decreasing: first, when no particle can enter the low-flow-rate branch because its hypothetical separation line y_0^* is above its maximum position $1 - R$, then the high-flow-rate branch receives only additional solvent when Q_1/Q_0 decreases and its particles are more diluted. Then, all curves fall onto the same curve $\Phi_2/\Phi_0 = 1/(1 - Q_1/Q_0)$ corresponding to $N_1 = 0$ (or $N_2 = N_0$), which results in a sharp variation as Q_1/Q_0 goes through the critical flow rate ratio. A smoother mechanism is also to be taken into account here, which is the one that finally determines the maximum for smaller R. As Q_1/Q_0 decreases, branch 2 recruits fluid and particles that are closer and closer to the opposite wall. As seen in figure 8, the discrepancy between the flow and particle velocities increases near the walls, so that N_2 increases less than Q_2: the resulting concentration in branch 2 finally decreases.

Finally, for applicative purposes, the consequences of the attraction towards the low-flow-rate branch are twofold: if one wishes to obtain a particle-free fluid (e.g. plasma without red blood cells), one has to set Q_1 low enough so that $N_1 = 0$. Because

of attraction towards the low-flow-rate branch, this critical flow rate is decreased and the efficiency of the process is lowered. If one prefers to concentrate particles, then one must find the maximum of the Φ_2/Φ_0 curve. This maximum is lowered and shifted by the attraction towards the low-flow-rate branch (see figure 9). Note that for small spheres (e.g. $R = 0.42$), the position of the maximum does not correspond to the point at which N_1 vanishes; in addition, the shift direction of the maximum position depends on the sphere size: while it shifts to lower Q_1/Q_0 values for $R = 0.71$, it shifts to higher values for $R = 0.42$.

The choice of the geometry, within our symmetric frame, can also greatly modify the efficiency of a device. Since the depletion effect eventually governs the final distribution, narrowing the inlet channel is the first requirement. On the other hand, it also increases the attraction towards the low-flow-rate branch, but one can try to diminish it. As discussed in the preceding section, this can be done by widening reasonably the daughter branches. For instance, if their half-width is not 1 but 2.5, as in figure 6(*b*), the slope ξ in the law $y_0^* = \xi \times (Q_1/Q_0 - 1/2)$ increases by around 15 % for $R = 0.67$. The critical Q_1/Q_0 below which no particle enters the low-flow-rate branch increases from 0.13 to 0.19, which is good for fluid–particle separation, and the maximum enrichment Φ_2/Φ_0 that can be reached is 22 % instead of 15 %. Alternatively, since the attraction is higher in two dimensions than in three, we can also infer that considering thicker channels, which does not modify the depletion effect, can greatly improve the final result. Note that this conclusion would have been completely different in the case of the high-flow-rate branch enrichment due to some attraction towards it, as claimed in some papers: in such a case, confining as much as possible would have been required, as it increases all kinds of cross-streamline drifts.

4.3. *Consistency with the literature*

We now come back to the previous studies already discussed in § 2 in order to check the consistency between them and our results.

The only paper that dealt with the position of the particle-separating streamline was by Barber *et al.* (2008), where a symmetric Y-shaped bifurcation was studied (branches leaving the bifurcation with a 45° angle relative to the inlet channel, see figure 1*a*). In figure 10(*a*), we compare their results with our simulations in a similar geometry. The agreement between the two simulations (based on two different methods) is very good, except for large particles ($R = 0.67$) and low Q_1/Q_0. Note that Barber *et al.* (2008) have chosen to consider branches whose widths follow the law $w_0^3 = w_1^3 + w_2^3$, where w_0 is the width of the inlet branch and w_1 and w_2 are the widths of the daughter branches. This law has been shown to describe approximately the relationship between vessel diameters in the arteriolar network (see Mayrovitz & Roy 1983). With our notation, they thus consider $a = \sqrt[3]{1/2} \simeq 0.79$, while we focused on $a = 1$ in order to compare with the T-shaped bifurcation. In addition, their apex has a radius 0.75 (for the $R = 0.67$ case) while ours is sharper (with radius of 0.1). These differences seem to impact only partly on the results, as discussed above. We can expect this slight discrepancy to be due to the treatment of the numerical singularities that appear when the particle is close to one wall. For $R = 0.67$, the maximum position y_0 is 0.33, which is close to the separating streamline position.

It is also interesting to compare our results in the Y-shaped bifurcation with the results in the T geometry, which was chosen to make the discussion easier. We can see that, for low enough Q_1/Q_0, the attraction towards the low-flow-rate branch is slightly higher. This can be understood by considering a particle with initial position y_0 slightly below the critical position y_0^* found in the T geometry: in the latter

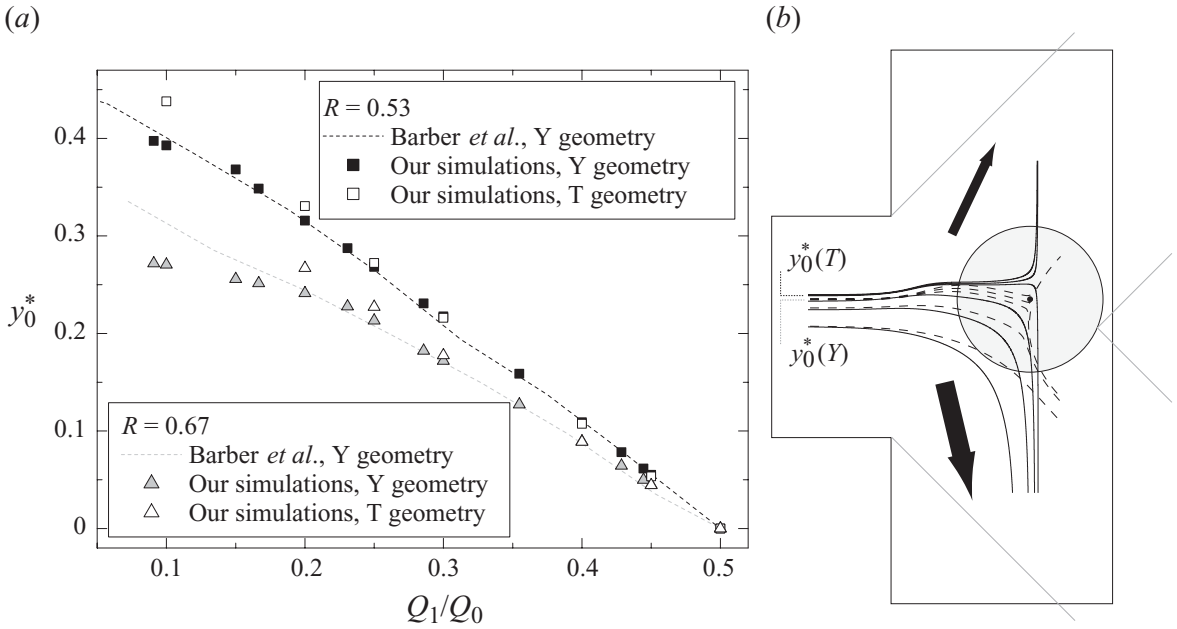


FIGURE 10. (*a*) Position of the particle separation line y_0^* in a symmetric Y-shaped bifurcation: according to Barber *et al.* (2008) (data extracted from their figure 4) and according to our simulations. The results for similar spheres in our T geometry are also shown. (*b*) Trajectories from our simulations in the T- and Y-shaped bifurcations, for similar sphere size ($R = 0.67$) and flow rate ratio ($Q_1/Q_0 = 0.2$). Full lines, T geometry; dashed lines, Y geometry. The corresponding separating streamline positions (respectively $y_0^*(T)$ and $y_0^*(Y)$) are also indicated. The sphere shown is located at its stagnation point y_0' (see figure 6) in the Y geometry.

geometry, it will eventually enter the high-flow-rate branch, by definition of y_0^*. As shown in figure 10(*b*), in the Y geometry, this movement is hindered by the apex since the final attraction towards the high-flow-rate branch occurs near the opposite wall (the second effect discussed in § 4.1.2). Finally, from this comparison we see that comparing results in T and symmetric Y geometry is relevant but for highly asymmetric flow distributions.

In § 2, the analysis of the two-dimensional simulations for $R = 0.5$ spheres in Audet & Olbricht (1987) showed that there should be some attraction towards the low-flow-rate branch. Our simulations for $R = 0.48$ showed that this effect is non-negligible (figure 5*b*) and modifies significantly the final distribution (figure 9). Finally, we can see in figure 11 that our simulations give results similar to those of Audet & Olbricht (1987).

As for the experiments presented in Yang *et al.* (2006) for $R = 0.46$, we showed that the final distribution was consistent with a no-attraction assumption. As we showed in figure 5(*a*), in a three-dimensional case, the attraction towards the low-flow-rate region is weak for spheres of radius $R \simeq 0.5$ or smaller, which is again consistent with the results of Yang *et al.* (2006). Note that, while their results were considered by the authors as a basis to discuss some attraction effect towards the high-flow-rate branch, we see that their final distributions are just reminiscent of the depletion effect in the inlet channel.

The other consistent set of studies in the literature deals with large balls in three-dimensional channels. We have studied balls of radius $R = 0.71$ that stop entering branch 1 when $Q_1/Q_0 \lesssim 0.22$ (figure 5*a*), while this critical flow rate would be around 0.29 if they followed the fluid streamlines. This critical flow rate is expected to be slightly higher for larger balls of radius $R \simeq 0.8$, but far lower than 0.35, which

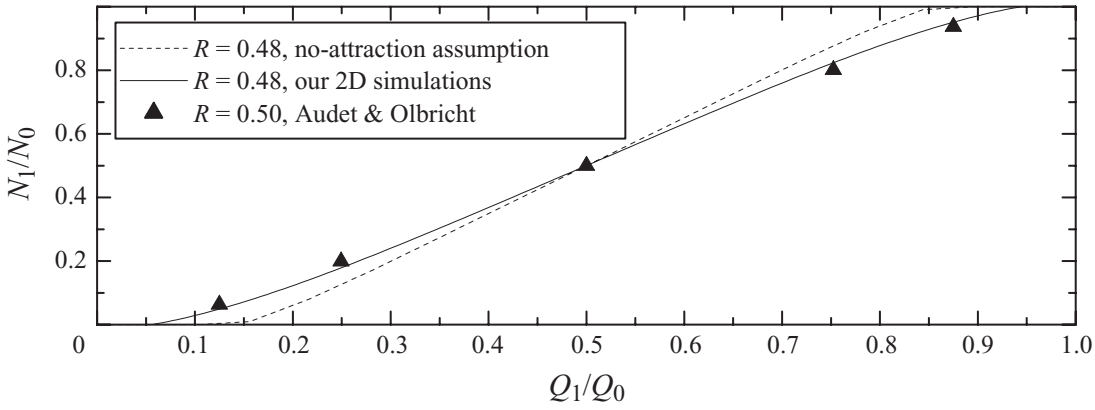


FIGURE 11. Particle distribution as a function of flow rate ratios for spheres of medium size, according to simulations of this paper and Audet & Olbricht (1987) (same data as plotted in figure 2).

would be the no-attraction case. In the experiments of Roberts & Olbricht (2006), some balls are still observed in branch 1 when $Q_1/Q_0 \simeq 0.22$ (figure 2), indicating a stronger attraction effect towards the low-flow-rate branch, which can be associated with the fact that the authors considered a square cross-sectional channel, while the confinement in the third direction is $0.5 < 0.71$ in our case. The experiments with circular cross-sectional channels lead to contradictory results: in Chien *et al.* (1985) and Ditchfield & Olbricht (1996), the results were consistent with a no-attraction assumption; therefore, they are in contradiction with our results. On the contrary, in Roberts & Olbricht (2003), the critical flow rate for $R = 0.77$ is around 0.2, which would show a stronger attraction than in our case. Note that all these apparently contradictory observations are to be considered keeping in mind that the data of N_1/N_0 as a function of Q_1/Q_0 are sometimes very noisy in the papers cited.

5. Conclusion

In this paper, we have focused explicitly on the existence and direction of some cross-streamline drift of particles in the vicinity of a bifurcation with different flow rates in the daughter branches. A new analysis of some previous unexploited results from the literature first gave us some indications on the possibility of an attraction towards the low-flow-rate branch.

Then, the first direct experimental proof of attraction towards the low-flow-rate branch was shown and arguments for this attraction were given with the help of two-dimensional simulations. In particular, we showed that this attraction is the result of two antagonistic effects: the first effect, which takes place at the entrance of the bifurcation, induces migration towards the low-flow-rate branch, while the second effect takes place near the stagnation point and induces migration towards the high-flow-rate branch but is not strong enough, in standard configurations of branches of comparable sizes, to counterbalance the first effect.

Only the second effect was previously considered in most papers in the literature, which has led to the misleading idea that the enrichment of particles in the high-flow-rate branch is due to some attraction towards it. On the contrary, it had been argued by Barber *et al.* (2008) that there should be some attraction towards the low-flow-rate branch. By distinguishing the two effects mentioned above, we have tried to clarify their statements.

In the second step, we have discussed the consequences of such an attraction on the final distribution of particles. It appears that the attraction is not strong enough, even in a two-dimensional system where it is stronger, to counterbalance the impact of the depletion effect. Even in the most homogeneous case where the particles are equally distributed across the channel but cannot approach the wall closer than their radius, the existence of a free layer near the walls favours the high-flow-rate branch, which eventually receives more particles than fluid.

However, these two antagonistic phenomena are of comparable importance, and neither can be neglected: the increase in the particle volume fraction in the high-flow-rate branch is typically divided by two because of the attraction effect. On the other hand, the initial distribution is a key parameter for the prediction of the final splitting. For deformable particles, initial lateral migration can induce a narrowing of their distribution, which will eventually favour the high-flow-rate branch. For instance, Barber *et al.* (2008) had to adjust the free layer width in their simulations in order to fit experimental data on blood flow. On the other hand, in a network of bifurcations, the initially centred particles will find themselves close to one wall after the first bifurcation, which can favour a low-flow-rate branch in a second bifurcation.

Note, finally, that as seen in Enden & Popel (1992), these effects become weaker when the confinement decreases. Typically, as soon as the sphere diameter is less than half the channel width, the variations of volume fraction do not exceed a few per cent.

For applicative purposes, the consequences of this attraction have been discussed and some prescriptions have been proposed. Of course, one can go further than our symmetric case and modify the angle between the branches, or consider many-branch bifurcations, and so on. However, the T-bifurcation case allowed us to distinguish between two goals: concentrating a population of particles or obtaining a particle-free fluid. The optimal configuration can be different according to the chosen goal. Similar considerations are also valid regarding sorting in polydisperse suspensions, which is an important activity (see Pamme 2007): getting an optimally concentrated suspension of large particles might not be compatible with getting a suspension of small particles free of large particles.

Now that the case of spherical particles in a symmetric bifurcation has been studied and the framework well established, we believe that quantitative discussions could be made in the future about the other parameters that we put aside here. In particular, discussing the effect of the deformability of the particles is a challenging task if one only considers the final distribution data, as the deformability modifies the initial distribution, but most probably also the attraction effect. In a network, the importance of these contributions will be different according, in particular, to the distance between two bifurcations, so they must be discussed separately.

Considering concentrated suspensions is of course the next challenging issue. Particles close to each other will obviously hydrodynamically interact, but so will distant particles, through the modification of the effective resistance to flow of the branches. In such a situation, considering pressure-driven or flow-rate-driven fluids will be different.

For concentrated suspensions of deformable particles in a network, like blood in the circulatory system, the relevance of a particle-based approach can be questioned. Historical models for the major blood flow phenomena are continuum models with some *ad hoc* parameters, which must be somehow related to the intrinsic mechanical properties of the blood cells (for a recent example, see Obrist *et al.* 2010). Building up a bottom-up approach in such a system is a long quest. For dilute suspensions,

some links between the microscopic dynamics of lipid vesicles and the rheology of a suspension have been recently established (see Danker & Misbah 2007; Vitkova *et al.* 2008; Ghigliotti, Biben & Misbah 2010). For red blood cells, which exhibit qualitatively similar dynamics (see Abkarian *et al.* 2007; Deschamps *et al.* 2009; Dupire, Abkarian & Viallat 2010; Farutin, Biben & Misbah 2010; Noguchi 2010), we can hope that such a link will soon be established, following Vitkova *et al.* (2008). For confined and concentrated suspensions, the distribution is known to be non-homogeneous, which has direct consequences on the rheology (the Fahraeus–Lindquist effect). Once again, while empirical macroscopic models are able to describe this reality, establishing the link between the viscosity of the suspension and the local dynamics is still a challenging issue. The final distribution of the flowing bodies is the product of a balance between migration towards the centre, as discussed in the Introduction, and interactions between them that can broaden the distribution (see Kantsler, Segre & Steinberg 2008; Podgorski *et al.* 2010). The presence of deformable boundaries also needs to be taken into account, as shown in Beaucourt *et al.* (2004). In the meantime, the development of simulation techniques for quantitative three-dimensional approaches is a crucial task, which is becoming more and more feasible (see McWhirter, Noguchi & Gompper 2009; Biben, Farutin & Misbah 2011).

The authors thank G. Ghigliotti for his final reading and acknowledge financial support from ANR MOSICOB and CNES.

Appendix. Derivation of the variational formulation

In this appendix, details of the derivation of (3.13)–(3.15) from (3.10)–(3.12) are given.

First, we introduce the scalar product in $L^2(\Omega_f)^2$ as follows:

$$\forall \boldsymbol{f}, \boldsymbol{g} \in L^2(\Omega_f)^2, \quad \langle \boldsymbol{f}, \boldsymbol{g} \rangle_{L^2(\Omega_f)^2} = \int_{\Omega_f} \boldsymbol{f} \cdot \boldsymbol{g}. \tag{A 1}$$

The variational formulation of the problem (3.10)–(3.12) is obtained by taking the scalar product of (3.10) in $L^2(\Omega_f)^2$ with a test function $\boldsymbol{v} \in H_0^1(\Omega_f)^2$ and we multiply (3.11) by a test function $q \in L_0^2(\Omega)$. It leads to the following problem: find $(\boldsymbol{u}, p) \in H^1(\Omega_f)^2 \times L_0^2(\Omega_f)$ such that

$$-2\nu \int_{\Omega_f} (\nabla \cdot \boldsymbol{\tau}(\boldsymbol{u})) \cdot \boldsymbol{v} + \int_{\Omega_f} \nabla p \cdot \boldsymbol{v} = 0, \quad \forall \boldsymbol{v} \in H_0^1(\Omega_f)^2, \tag{A 2}$$

$$\int_{\Omega_f} q \nabla \cdot \boldsymbol{u} = 0, \quad \forall q \in L_0^2(\Omega_f), \tag{A 3}$$

$$\boldsymbol{u} = \boldsymbol{f} \quad \text{on } \partial\Omega_f. \tag{A 4}$$

Applying Green's formula to (A 2), we obtain

$$2\nu \int_{\Omega_f} \boldsymbol{\tau}(\boldsymbol{u}) : \nabla \boldsymbol{v} - 2\nu \int_{\partial\Omega_f} \boldsymbol{\tau}(\boldsymbol{u}) \boldsymbol{n} \cdot \boldsymbol{v} - \int_{\Omega_f} p \nabla \cdot \boldsymbol{v} + \int_{\partial\Omega_f} p \boldsymbol{v} \cdot \boldsymbol{n} = 0, \tag{A 5}$$

where $\boldsymbol{n}$ denotes the outer unit normal on $\partial\Omega_f$. Taking into account the fact that $\boldsymbol{v}$ vanishes on $\partial\Omega_f$ (recall that we have chosen the test function $\boldsymbol{v} \in H_0^1(\Omega_f)^2$), the

problem (A 2)–(A 4) is now equivalent to this one: find $(\boldsymbol{u}, p) \in H^1(\Omega_f)^2 \times L_0^2(\Omega_f)$ such that

$$2\nu \int_{\Omega_f} \boldsymbol{\tau}(\boldsymbol{u}) : \nabla \boldsymbol{v} - \int_{\Omega_f} p \nabla \cdot \boldsymbol{v} = 0, \quad \forall \boldsymbol{v} \in H_0^1(\Omega_f)^2, \tag{A 6}$$

$$\int_{\Omega_f} q \nabla \cdot \boldsymbol{u} = 0, \quad \forall q \in L_0^2(\Omega_f), \tag{A 7}$$

$$\boldsymbol{u} = \boldsymbol{f} \quad \text{on } \partial\Omega_f. \tag{A 8}$$

Note that $\boldsymbol{\tau}(\boldsymbol{u})$ is symmetric ($\boldsymbol{\tau}(\boldsymbol{u}) : \nabla \boldsymbol{v} = \boldsymbol{\tau}(\boldsymbol{u}) : (\nabla \boldsymbol{v})^t$). So we can write $\boldsymbol{\tau}(\boldsymbol{u}) : \nabla \boldsymbol{v} = \boldsymbol{\tau}(\boldsymbol{u}) : \boldsymbol{\tau}(\boldsymbol{v})$. Finally, the variational formulation of our initial problem (3.1)–(3.3) is given by: find $(\boldsymbol{u}, p) \in H^1(\Omega_f)^2 \times L_0^2(\Omega_f)$ such that

$$2\nu \int_{\Omega_f} \boldsymbol{\tau}(\boldsymbol{u}) : \boldsymbol{\tau}(\boldsymbol{v}) - \int_{\Omega_f} p \nabla \cdot \boldsymbol{v} = 0, \quad \forall \boldsymbol{v} \in H_0^1(\Omega_f)^2, \tag{A 9}$$

$$\int_{\Omega_f} q \nabla \cdot \boldsymbol{u} = 0, \quad \forall q \in L_0^2(\Omega_f), \tag{A 10}$$

$$\boldsymbol{u} = \boldsymbol{f} \quad \text{on } \partial\Omega_f. \tag{A 11}$$

REMARK 2. *As we have*

$$\boldsymbol{\tau}(\boldsymbol{u}) : \boldsymbol{\tau}(\boldsymbol{v}) = \boldsymbol{\tau}(\boldsymbol{u}) : \nabla \boldsymbol{v} = \tfrac{1}{2} \nabla \boldsymbol{u} : \nabla \boldsymbol{v} + \tfrac{1}{2} (\nabla \boldsymbol{u})^t : \nabla \boldsymbol{v}, \tag{A 12}$$

the first integral in (A 9) can be rewritten thanks to this identity

$$\int_{\Omega_f} \boldsymbol{\tau}(\boldsymbol{u}) : \boldsymbol{\tau}(\boldsymbol{v}) = \frac{1}{2} \int_{\Omega_f} \nabla \boldsymbol{u} : \nabla \boldsymbol{v}. \tag{A 13}$$

Indeed, by integration by parts and using the incompressibility constraint $\nabla \cdot \boldsymbol{u} = 0$, *we have* $\int_{\Omega_f} (\nabla \boldsymbol{u})^t : \nabla \boldsymbol{v} = 0$. *Thus, we can retrieve the formulation of our problem as a minimization of a kind of energy. The velocity field* $\boldsymbol{u}$ *is then the solution of this problem*

$$\boldsymbol{J}(\boldsymbol{u}) = \inf_{\substack{\boldsymbol{v} \in H^1(\Omega_f)^2 \\ \nabla \cdot \boldsymbol{v} = 0, \boldsymbol{v}_{|\partial\Omega_f} = \boldsymbol{f}}} \boldsymbol{J}(\boldsymbol{v}), \tag{A 14}$$

where

$$\boldsymbol{J}(\boldsymbol{v}) = \frac{1}{2} \nu \int_{\Omega_f} \nabla \boldsymbol{v} : \nabla \boldsymbol{v} = \nu \int_{\Omega_f} \boldsymbol{\tau}(\boldsymbol{v}) : \boldsymbol{\tau}(\boldsymbol{v}). \tag{A 15}$$

REFERENCES

ABKARIAN, M., FAIVRE, M. & VIALLAT, A. 2007 Swinging of red blood cells under shear flow. *Phys. Rev. Lett.* **98**, 188302.

ABKARIAN, M., LARTIGUE, C. & VIALLAT, A. 2002 Tank treading and unbinding of deformable vesicles in shear flow: determination of the lift force. *Phys. Rev. Lett.* **88**, 068103.

ANGELOVA, M. I., SOLEAU, S., MELEARD, P., FAUCON, J. F. & BOTHOREL, P. 1992 Preparation of giant vesicles by external AC electric fields: kinetics and applications. *Prog. Colloid. Polym. Sci.* **89**, 127–131.

ASMOLOV, E. S. 2002 The inertial lift on a small particle in a weak-shear parabolic flow. *Phys. Fluids* **14**, 15–28.

AUDET, D. M. & OLBRICHT, W. L. 1987 The motion of model cells at capillary bifurcations. *Microvasc. Res.* **33**, 377–396.

Bagchi, P. 2007 Mesoscale simulation of blood flow in small vessels. *Biophys. J.* **92**, 1858–1877.

Balvin, M., Sohn, E., Iracki, T., Drazer, G. & Frechette, J. 2009 Directional locking and the role of irreversible interactions in deterministic hydrodynamics separations in microfluidic devices. *Phys. Rev. Lett.* **103**, 078301.

Barber, J. O., Alberding, J. P., Restrepo, J. M. & Secomb, T. W. 2008 Simulated two-dimensional red blood cell motion, deformation, and partitioning in microvessel bifurcations. *Ann. Biomech. Engng* **36**, 1690–1698.

Beaucourt, J., Rioual, F., Séon, T., Biben, T. & Misbah, C. 2004 Steady to unsteady dynamics of a vesicle in a flow. *Phys. Rev.* E **69**, 011906.

Biben, T., Farutin, A. & Misbah, C. 2011 Numerical study of 3D vesicles under flow: discovery of new peculiar behaviors. *Phys. Rev.* E (in press) arXiv:0912.4702.

Bugliarello, G. & Hsiao, G. C. C. 1964 Phase separation in suspensions flowing through bifurcations: a simplified hemodynamic model. *Science* **143**, 469–471.

Callens, N., Minetti, C., Coupier, G., Mader, M.-A., Dubois, F., Misbah, C. & Podgorski, T. 2008 Hydrodynamic lift of vesicles under shear flow in microgravity. *Europhys. Lett.* **83**, 24002.

Carr, R. T. & Wickham, L. L. 1990 Plasma skimming in serial microvascular bifurcations. *Microvasc. Res.* **40**, 179–190.

Chesnutt, J. K. W. & Marshall, J. S. 2009 Effect of particle collisions and aggregation on red blood cell passage through a bifurcation. *Microvasc. Res.* **78**, 301–313.

Chien, S., Tvetenstrand, C. D., Epstein, M. A. & Schmid-Schonbein, G. W. 1985 Model studies on distributions of blood cells at microvascular bifurcations. *Am. J. Physiol. Heart Circ. Physiol.* **248**, H568–H576.

Coupier, G., Kaoui, B., Podgorski, T. & Misbah, C. 2008 Noninertial lateral migration of vesicles in bounded Poiseuille flow. *Phys. Fluids* **20**, 111702.

Danker, G. & Misbah, C. 2007 Rheology of a dilute suspension of vesicles. *Phys. Rev. Lett.* **98**, 088104.

Danker, G., Vlahovska, P. M. & Misbah, C. 2009 Vesicles in Poiseuille flow. *Phys. Rev. Lett.* **102**, 148102.

Davis, J. A., Inglis, D. W., Morton, K. J., Lawrence, D. A., Huang, L. R., Chou, S. Y., Sturm, J. C. & Austin, R. H. 2006 Deterministic hydrodynamics: taking blood apart. *Proc. Natl Acad. Sci. USA* **103**, 14779–14784.

Dellimore, J. W., Dunlop, M. J. & Canham, P. B. 1983 Ratio of cells and plasma in blood flowing past branches in small plastic channels. *Am. J. Physiol. Heart Circ. Physiol.* **244**, H635–H643.

Deschamps, J., Kantsler, V., Segre, E. & Steinberg, V. 2009 Dynamics of a vesicle in general flow. *Proc. Natl Acad. Sci. USA* **106**, 11444.

Ditchfield, R. & Olbricht, W. L. 1996 Effects of particle concentration on the partitioning of suspensions at small divergent bifurcations. *J. Biomech. Engng* **118**, 287–294.

Dupire, J., Abkarian, M. & Viallat, A. 2010 Chaotic dynamics of red blood cells in a sinusoidal flow. *Phys. Rev. Lett.* **104**, 168101.

El-Kareh, A. W. & Secomb, T. W. 2000 A model for red blood cell motion in bifurcating microvessels. *Intl J. Multiphase Flow* **26**, 1545–1564.

Eloot, S., De Bisschop, F. & Verdonck, P. 2004 Experimental evaluation of the migration of spherical particles in three-dimensional Poiseuille flow. *Phys. Fluids* **16**, 2282–2293.

Enden, G. & Popel, A. S. 1992 A numerical study of the shape of the surface separating flow into branches in microvascular bifurcations. *J. Biomech. Engng* **114**, 398–405.

Engl, W., Roche, M., Colin, A., Panizza, P. & Ajdari, A. 2005 Droplet traffic at a simple junction at low capillary numbers. *Phys. Rev. Lett.* **95**, 208304.

Fan, R., Vermesh, O., Srivastava, A., Yen, B., Qin, L., Ahmad, H., Kwong, G. A., Liu, C.-C., Gould, J., Hood, L. & Heath, J. R. 2008 Integrated barcode chips for rapid, multiplexed analysis of proteins in microliter quantities of blood. *Nature Biotech.* **26**, 1373–1378.

Farutin, A., Biben, T. & Misbah, C. 2010 Analytical progress in the theory of vesicles under linear flow. *Phys. Rev.* E **81**, 061904.

Fenton, B. M., Carr, R. T. & Cokelet, G. R. 1985 Nonuniform red cell distribution in 20 to 100 μm bifurcations. *Microvasc. Res.* **29**, 103–126.

Frechette, J. & Drazer, G. 2009 Directional locking and deterministic separation in periodic arrays. *J. Fluid Mech.* **627**, 379–401.

Fung, Y. C. 1973 Stochastic flow in capillary blood vessels. *Microvasc. Res.* **5**, 34–48.

Fung, Y. C. 1993 *Biomechanics: Mechanical Properties of Living Tissues*. Springer.

Ghigliotti, G., Biben, T. & Misbah, C. 2010 Rheology of a dilute two-dimensional suspension of vesicles. *J. Fluid Mech.* **653**, 489–518.

Girault, V. & Raviart, P.A. 1986 *Finite Element Methods for Navier–Stokes Equations: Theory and Algorithms*. Springer.

Griggs, A. J., Zinchenko, A. Z. & Davis, R. H. 2007 Low-Reynolds-number motion of a deformable drop between two parallel plane walls. *Intl J. Multiphase Flow* **33**, 182–206.

Guibert, R., Fonta, C. & Plouraboue, F. 2010 A new approach to model confined suspensions flows in complex networks: application to blood flow. *Trans. Porous Med.* **83**, 171–194.

Hecht, F. & Pironneau, O. 2010 A finite element software for PDEs: freeFEM++. Available at: http://www.freefem.org/.

Inglis, D. W. 2009 Efficient microfluidic particle separation arrays. *Appl. Phys. Lett.* **94**, 013510.

Jäggi, R. D., Sandoz, R. & Effenhauser, C. S. 2007 Microfluidic depletion of red blood cells from whole blood in high-aspect-ratio microchannels. *Microfluid Nanofluid* **3**, 47–53.

Janela, J., Lefebvre, A. & Maury, B. 2005 A penalty method for the simulation of fluid-rigid body interaction. In *ESAIM: Proc* (ed. E. Cancès & J.-F. Gerbeau), vol. 14, pp. 115–123. ESAIM.

Jousse, F., Farr, R., Link, D. R., Fuerstman, M. J. & Garstecki, P. 2006 Bifurcation of droplet flows within capillaries. *Phys. Rev.* E **74**, 036311.

Kantsler, V., Segre, E. & Steinberg, V. 2008 Dynamics of interacting vesicles and rheology of vesicle suspension in shear flow. *Europhys. Lett.* **82**, 58005.

Kaoui, B., Coupier, G., Misbah, C. & Podgorski, T. 2009 Lateral migration of vesicles in microchannels: effects of walls and shear gradient. *Houille Blanche* **5**, 112–119.

Kaoui, B., Ristow, G., Cantat, I., Misbah, C. & Zimmermann, W. 2008 Lateral migration of a two-dimensional vesicle in unbounded Poiseuille flow. *Phys. Rev.* E **77**, 021903.

Kersaudy-Kerhoas, M., Dhariwal, R., Desmulliez, M. P. Y. & Jouvet, L. 2010 Hydrodynamic blood plasma separation in microfluidic channels. *Microfluid Nanofluid* **8**, 105–114.

Kim, Y. W. & Yoo, J. Y. 2008 The lateral migration of neutrally-buoyant spheres transported through square microchannels. *J. Micromech. Microengng* **18**, 065015.

Lefebvre, A. 2007 Fluid–particle simulations with FreeFem++. In *ESAIM: Proc.*, vol. 18, pp. 120–132. ESAIM.

Maury, B. 2009 Numerical analysis of a finite element/volume penalty method. *SIAM J. Numer. Anal.* **47** (2), 1126–1148.

Mayrovitz, H. N. & Roy, J. 1983 Microvascular blood flow: evidence indicating a cubic dependence on arteriolar diameter. *Am. J. Physiol.* **255**, H1031–H1038.

McWhirter, J. L., Noguchi, H. & Gompper, G. 2009 Flow-induced clustering and alignment of vesicles and red blood cells in microcapillaries. *Proc. Natl Acad. Sci. USA* **106**, 6039–6043.

Mortazavi, S. & Tryggvason, G. 2000 A numerical study of the motion of drops in Poiseuille flow. Part 1. Lateral migration of one drop. *J. Fluid. Mech.* **411**, 325–350.

Noguchi, H. 2010 Dynamic modes of red blood cells in oscillatory shear flow. *Phys. Rev.* E **81**, 061920.

Obrist, D., Weber, B., Buck, A. & Jenny, P. 2010 Red blood cell distribution in simplified capillary networks. *Phil. Trans. R. Soc. Lond.* A **368**, 2897–2918.

Olla, P. 1997 The lift on a tank-treading ellipsoidal cell in a shear flow. *J. Phys.* II *Paris* **7**, 1533–1540.

Pamme, N. 2007 Continuous flow separations in microfluidic devices. *Lab Chip* **7**, 1644–1659.

Peyla, P. 2007 A deformable object in a microfluidic configuration: a numerical study. *Europhys. Lett.* **80**, 34001.

Podgorski, T., Callens, N., Minetti, C., Coupier, G., Dubois, F. & Misbah, C. 2010 Dynamics of vesicle suspensions in shear flow between walls. *Microgravity Sci. Technol.* **23**, 263–270.

Pries, A. R., Ley, K., Claassen, M. & Gaethgens, P. 1989 Red cell distribution at microvascular bifurcations. *Microvasc. Res.* **38**, 81–101.

Pries, A. R., Secomb, T. W. & Gaethgens, P. 1996 Biophysical aspects of blood flow in the microvasculature. *Cardiovasc. Res.* **32**, 654–667.

Risso, F., Collé-Paillot, F. & Zagzoule, M. 2006 Experimental investigation of a bioartificial capsule flowing in a narrow tube. *J. Fluid. Mech.* **547**, 149–173.

Roberts, B. W. & Olbricht, W. L. 2003 Flow-induced particulate separations. *AIChE J.* **49**, 2842–2849.

Roberts, B. W. & Olbricht, W. L. 2006 The distribution of freely suspended particles at microfluidic bifurcations. *AIChE J.* **52**, 199–206.

Schindler, M. & Ajdari, A. 2008 Droplet traffic in microfluidic networks: a simple model for understanding and designing. *Phys. Rev. Lett.* **100**, 044501.

Schonberg, J. A. & Hinch, E. J. 1989 Inertial migration of a sphere in a Poiseuille flow. *J. Fluid Mech.* **203**, 517–524.

Secomb, T. W., Styp-Rekowska, B. & Pries, A. R. 2007 Two-dimensional simulation of red blood cell deformation and lateral migration in microvessels. *Ann. Biomed. Engng* **35**, 755–765.

Sessoms, D. A., Belloul, M., Engl, W., Roche, M., Courbin, L. & Panizza, P. 2009 Droplet motion in microfluidic networks: hydrodynamic interactions and pressure-drop measurements. *Phys. Rev.* E **80**, 016317.

Svanes, K. & Zweifach, B. W. 1968 Variations in small blood vessel hematocrits produced in hypothermic rats by micro-occlusion. *Microvasc. Res.* **1**, 210–220.

Tanaka, H. & Araki, T. 2000 Simulation method of colloidal suspensions with hydrodynamic interactions: fluid particle dynamics. *Phys. Rev. Lett.* **85** (6), 1338–1341.

Vitkova, V., Mader, M.-A., Polack, B., Misbah, C. & Podgorski, T. 2008 Micro–macro link in rheology of erythrocyte and vesicle suspensions. *Biophys. J.* **95**, 33–35.

Vlahovska, P. M., Podgorski, T. & Misbah, C. 2009 Vesicles and red blood cells in flow: from individual dynamics to rheology. *C. R. Physique* **10**, 775–789.

White, F. M. 1991 *Viscous Fluid Flow*. McGraw-Hill.

Yang, S., Ündar, A. & Zahn, J. D. 2006 A microfluidic device for continuous, real time blood plasma separation. *Lab Chip* **6**, 871–880.

Yang, S. & Zahn, J. D. 2004 Particle separation in microfluidic channels using flow rate control. In *Proc. ASME Conf. International Mechanical Engineering Exp Fluids Engineering (IMECE)*, Anaheim, CA.

Yen, R. T. & Fung, Y. C. 1978 Effect of velocity distribution on red cell distribution in capillary blood vessels. *Am. J. Physiol. Heart Circ. Physiol.* **235**, H251–H257.

Yoo, J. Y. & Kim, Y. W. 2010 Two-phase flow laden with spherical particles in a microcapillary. *Intl J. Multiphase Flow* **36**, 460–466.

Zheng, S., Liu, J.-Q. & Tai, Y.-C. 2008 Streamline-based microfluidic devices for erythrocytes and leukocytes separation. *J. Microelectromech. Syst.* **17**, 1029–1038.

7.2 Suspension of vesicle in a bifurcation

In the future we would like to continue the work previously done for dilute suspension of spheres and expand it to the simulation of soft objects and suspensions. In this section, we will show the ability of our framework to handle a complete suspension of vesicles in a bifurcation. We will explain some technical aspects and how to obtain the quantities of interest and show a few results. Then we will show that it is also possible to handle suspensions with two different kinds of vesicles.

7.2.1 Entry and exit of vesicles

The goal is to simulate the entry of a continuous flow of vesicle in a bifurcation and measure the out flux in each branch. As opposed to what has been done in the previous study for dilute suspension, the goal is not to follow the trajectory of an individual particle but to have a global information. First of all let us explain how it is possible to deal with the entry and exit of vesicles.

As all the vesicles are of the same kind (same viscosity and rigidity), a single level set field ϕ is defined and handles all the interfaces. The geometry of the bifurcation is the same as in the previous study, as are the fluid equations and boundary conditions. A first vesicle is set as an ellipse in the inlet boundary. Since a Poiseuille flow is imposed at the entrance of the channel, the vesicle moves forward. To have a continuous flow of vesicles it is necessary to add new vesicles in the inlet channel. One way to do it could be to add new vesicles periodically in time, but this might not be the way which most represents the reality. Indeed, if the vesicles at the bifurcation slow down the flow of vesicle, new vesicles would still be added and the concentration in the inlet branch would increase. Consequently, we choose to add a new vesicle only when the one before it has left an inlet region that we define. Indeed, we define a band of size L_{in} in the x direction and of the size of the inlet channel in the y direction, we call it Ω_{in}. We compute the amount of vesicles in this region A_{in} as:

$$A_{\text{in}} = \int_{\Omega_{\text{in}}} (1 - H_\varepsilon(\phi)). \tag{7.2}$$

When A_{in} is smaller than a certain value, a new vesicle is added in Ω_{in} or a bit upstream of it as shown on the scheme 7.2.1. The length L_{in}, the position of the new vesicle, as well as the threshold of A_{in} under which a new vesicle is added are the three parameters on which one can vary to change the concentration of particles in the inlet.
Let us explain now how we can add a new vesicle in the system. Since we are in a Stokes regime, the solution of Stokes equations does not need the previous one. Thus, no special care has to be taken at the fluid point of view and we can simply add a new object in the inlet, the Stokes equations will take it into account instantaneously. We only need to be careful to put the particle far enough from the center of interest of the physical study to avoid modifying the behavior we want to observe. In this case, it is sufficient to add the new particles at the very beginning of the inlet to be sure that the physics of the bifurcation point is not modified. Consequently, the procedure of adding a new vesicle simply consists of modifying the level set function ϕ to a new one, ϕ^* containing the new

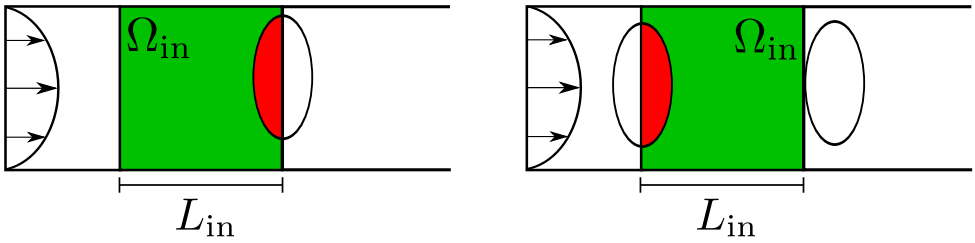


Figure 7.5: Scheme of the inlet of the bifurcation. The domain Ω_{in} is represented in green. The area where $(1 - H_\varepsilon(\phi)) \neq 0$ is represented in red. On the left hand side, the quantity A_{in} is too high for a new vesicle to be added. On the right hand side, the vesicle just exited the domain Ω_{in}, this makes A_{in} vanishing, thus a new vesicle is added.

vesicle. This is done by computing a field which represents the distance function to the new vesicles, ϕ_{ves} and set:

$$\phi^* = \min(\phi_{\text{ves}}, \phi). \tag{7.3}$$

This way, the interfaces already present in the bifurcation are not lost, and the new vesicle is added. Nevertheless, even if the fluid equations are independent of the time since we are concerned about the Stokes flow, the time evolution of the advection equation of the level set might by of order two. If so, we need to impose a first order scheme when a new particle is added in order to avoid the need of the previous value of ϕ which would induce an unwanted jump.

We discussed the choice of the x position of the new vesicle added in the channel, but the y position is also an important parameter. From the numerical point of view, there is no restriction for the vesicle to be set at any y position in the inlet channel, the choice is guided by the physics we want to represent. To represent a homogeneous suspension in the inlet branch, a random choice of the position y with the same probability in all the accessible y can be done. One can also take into account that the vesicles tend to migrate to the center and pick the new y position in a distribution taking into account this effect.

The exit of vesicles at the outlet channels is not difficult to handle. The only care which might have to be taken is because the curvature and its higher order derivatives might not be captured correctly when the vesicle is *cut* at the outlet. The vesicle is not a closed shape anymore and its normal, curvatures and higher derivatives have unpredictable values creating huge force points. This can be avoided simply by making vanish the forces few mesh sizes away from the exit. This way, from the fluid point of view, the vesicle outlet is a few mesh sizes before the geometric outlet, and at this point, the curvature is still defined.

7.2.2 Quantity of interest: the flux of vesicles

For a simulation of a semi-dilute or concentrated suspension, the quantities of interest are not anymore the trajectories of individual particles but the flux of particles in each branch. To measure these fluxes, we define three very small bands of about $2h$ size in length that we call Ω_{N_0}, Ω_{N_1} and Ω_{N_2}. The fluxes of particles are computed through these

domains as :

$$N_i = \frac{\displaystyle\int_{\Omega_{N_i}} (\boldsymbol{u}\cdot\boldsymbol{n}_i)(1-H_\varepsilon(\phi))}{\displaystyle\int_{\Omega_{N_i}} 1} \quad \text{with} \quad i=0,1,2 \tag{7.4}$$

where $\boldsymbol{u}$ is the fluid velocity and $\boldsymbol{n}_i$ is the normal to the domain Ω_{N_i}. More precisely, $\boldsymbol{n}_0 = (1,0)$, $\boldsymbol{n}_1 = (0,1)$, and $\boldsymbol{n}_2 = (0,-1)$. The different domains are disposed as in figure 7.6, not too far from the outlets.

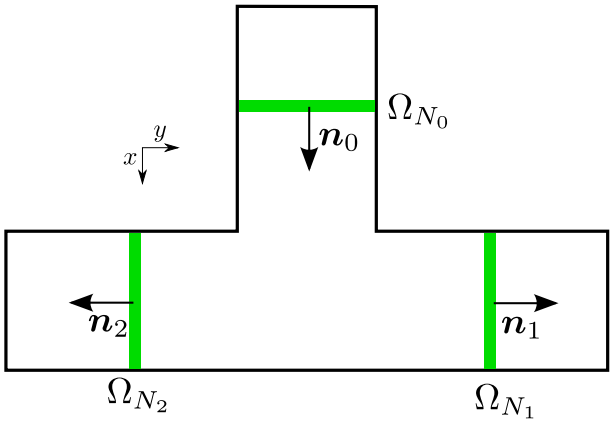


Figure 7.6: Scheme representing the positions of the domains in which the fluxes are measured.

The figure 7.8 shows a simulation made on 12 processors. The time step has been fixed to $2\cdot10^{-3}$ and the mesh size $h = 0.05$. The flow forces are assumed to be much stronger than the membrane forces, thus the parameter k_B has been set to a value close to 0. The flow rate ratio has been set to $\dfrac{Q_1}{Q_2} = \dfrac{2}{3}$. In this simulation we choose to pick the new y position of the vesicle in the inlet channel randomly, with an equal probability of each y to occur. The resulting fluxes are shown in figure 7.7(a). We also made another simulation for which all the vesicles were set at the center of the inlet channel. This could simulate the fact that a focusing device has been used upstream. The result of this simulation is shown in figure 7.7(b). We can see on the figure 7.7(b) that the inlet channel presents a nice regular periodicity. This comes from the fact that the vesicles are all at the center of the channel initially and thus pass through Ω_0 all the same way. Whereas, in figure 7.7(a), the vesicles which are closer to a wall see a local shear and incline which might result in a larger flux peak. Moreover, these particles are moving faster than the one at the center, thus, they also exhibit a slightly higher peak.

We can also guess only from the flux graphs which branch is the high or low flow rate one. Indeed, one sees clearly that the top branch sees a smaller flux of particles and moreover, the peaks are large which means that the vesicles spend a long time in the domain Ω_1 and the height is low which also means they are going slowly. It is the opposite in the branch 2. Furthermore, we see clearly that the first particles are entering in this branch.

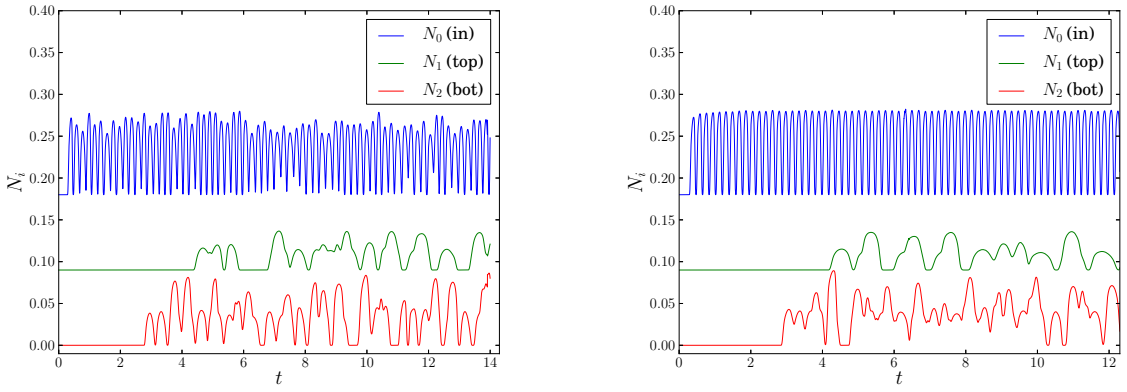
The kind of information that we would like to extract in the future is the influence of the many parameters that we can change on the mean fluxes in each branch. We define

the mean fluxes as:

$$< N_i >= \frac{1}{t_f - t_i} \int_{t_i}^{t_f} N_i \tag{7.5}$$

where t_i and t_f are respectively the initial and final time of the simulation. Thus, by integrating the fluxes in the branches we can compare the ratio of particle fluxes and the ratio of fluid fluxes. We obtained for this particular case: $\frac{< N_1 >}{< N_2 >} = 0.48$ for the simulation where the y inlet positions were random, and $\frac{< N_1 >}{< N_2 >} = 0.46$ for the other one. We can see that there is still a concentration increase in the high flow rate branch since the initial flow rate ratio were $\frac{Q_1}{Q_2} = \frac{2}{3}$.

This value should be studied by varying many parameters in this problem and we hope to be able to extract general laws out of such simulations on the general repartition of the cells in vivo and in vitro.



(a) Fluxes in each branch when the y position of the vesicle is randomly set.

(b) Fluxes in each branch when the y position of the vesicle is always at the center of the inlet channel.

Figure 7.7: Fluxes measured in each branch of a bifurcation. An offset of 0.9 have been used to separate all the curves.

7.2.3 Extension to two different kinds of vesicles

In the future we want to address the problem of the splitting of a suspension of different kinds of vesicles. It happen in some diseases like drepanocitosis, also called sickle cell disease, that some red blood cells become very rigid. This property makes their flow in microcapillaries very difficult and leads to dramatic effects on patients. A better understanding of the flow of suspensions with objects of different rigidity or viscosity ratio could also lead to a better knowledge of these diseases. During this work, we set up the simulation of such a problem and checked that we were able to measure the different fluxes.

The method to simulate vesicles is the one described in chapter 4 and the multi-level set framework presented in section 2.2.2 is used to deal with the two different kinds of vesicles. For this test, we call a kind of vesicles the *vesicles 0* and the other one the

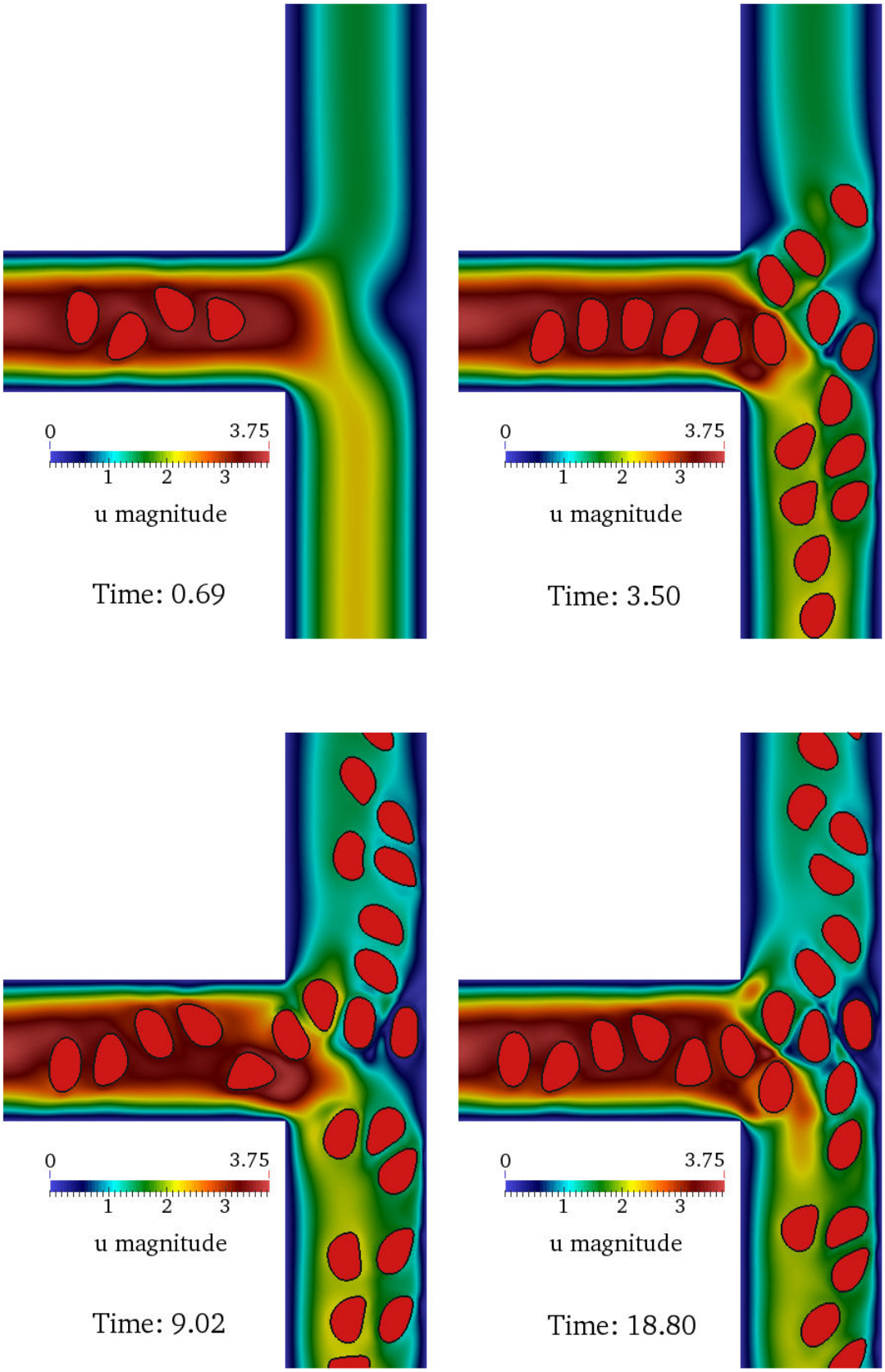


Figure 7.8: Suspension of vesicles in a bifurcation.

vesicles 1. We set the capillary number of the vesicles 0 to a huge value so that there is almost no curvature force. The viscosity ratio of these vesicles is set to 1. By contrast, the vesicles 1 have a viscosity ratio of 20 which makes them much less deformable. We set the inlet conditions so that one particle 1 enters for every two particles 0. Thus globally we have in the suspension $\frac{\Phi_1}{\Phi_0} = \frac{1}{2}$. The flux of each kind of vesicle is measured in every branch, the figure 7.10 shows this flux monitoring. We can see on this figure that the peaks associated to the vesicles 1 are more broad than those of vesicle 0 when they are in the daughter branches. This is because they are more deformable and a very elongated particle takes more time to pass through the line where the flux is defined. This effect is lowered for the first particles and for the particles inside the inlet branch because at these times of the simulation, the particles did not interact much with each other nor with the fluid and are not too deformed.
The time for which the fluxes are measured is too low to be able to say if there is a big difference between the fluxes going in each branch for the different kind of particles. The goal of this simulation was to show the ability of the code to handle such kind of simulation and show what kind of information it is possible to extract from it.

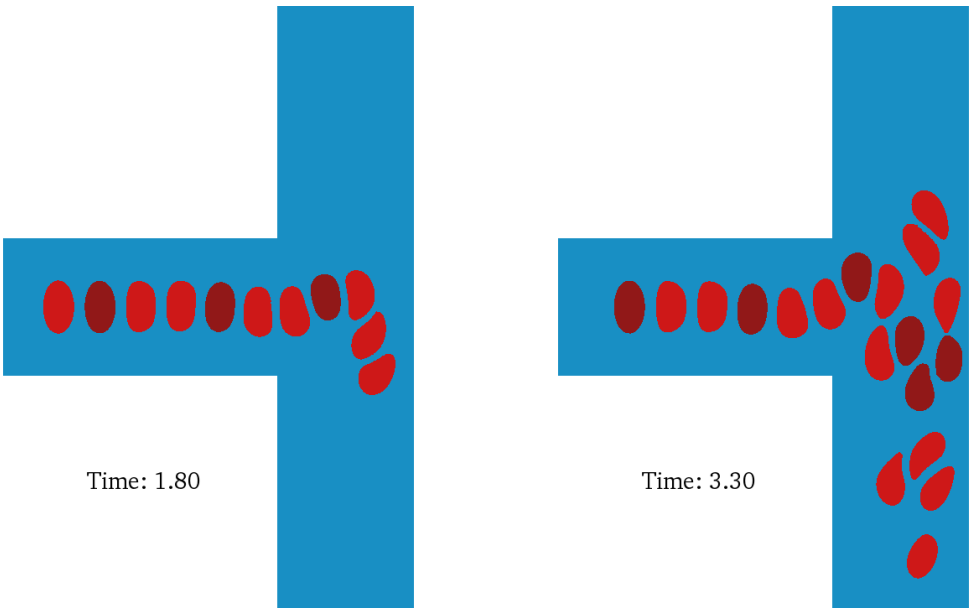


Figure 7.9: Simulation of a suspension of two kind of vesicles in a bifurcation. The dark red are 20 times more viscous than the light red. The curvature force is set to a very small value, allowing the vesicles to present very thin *tails*.

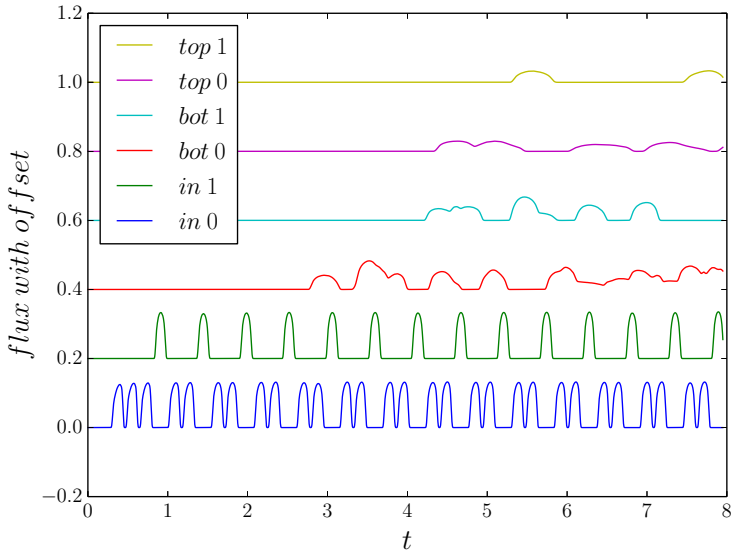


Figure 7.10: Fluxes of vesicle in each branch of the bifurcation. In the legend, in, top and bottom represent respectively the inlet, the top branch and the bottom branch. One can see that at the inlet, one vesicle 0 is added for every two vesicles 1. The vesicles 0 are highly deformable, which explains why the peaks when they pass through the branches are broad.

Chapter 8

Rheology of solid disks

Contents

The goal of this chapter is to show results that we obtained in preparation for the rheology study of semi dilute suspensions of vesicles within our framework. To be able to perform quantitative study of such a system, some work needs to be done to make sure that we are able to recover the basic known results of rheology. Moreover, a precise understanding of the rheology of deformable objects in flow requires the knowledge of the behavior of simple non-deformable objects. This is why we will start this chapter by showing the system used to measure the viscosity of a suspension. Then we will show that we are able to recover the Einstein viscosity of a dilute suspension of 2D rigid disks and control the accuracy of the measure of the viscosity. We will then measure the viscosity of a single particle in a confined shear flow, explore the possible parameters and compare our results with an analytical model. We extend the same method to the viscosity of two interacting particles. Finally, we will use these results to derive a method to measure the viscosity of a semi dilute suspension of particles for different configurations. We will show with this method that we can recover a behavior already known in 3D which is the decrease of the influence of the pair interactions at high confinement.

8.1 Numerical measurement of the viscosity of a suspension of particles

We compared three different methods to compute the effective viscosity of a suspension of rigid disks confined between two moving walls. The configuration of the problem is shown in figure 8.1. The fluid is governed by the Stokes equations. As in section 3.1, we denote Ω the whole computational domain, Ω_f the fluid domain and B_i the rigid disks. We recall the definition of the flow rate $\gamma = \dfrac{2U}{l}$.

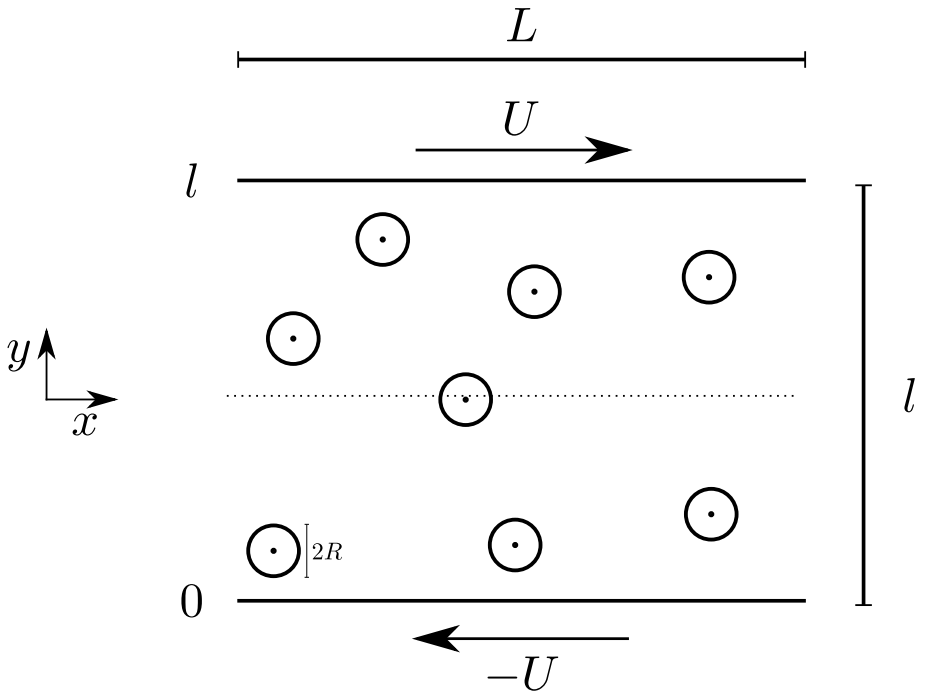


Figure 8.1: Suspension of rigid disks between two moving walls.

8.1.1 Some methods to compute the effective viscosity

Computing the forces on the walls

A possible way to compute the effective viscosity, is to compute the forces exerted by the fluids on the walls. It is equivalent to what is done experimentally with a rheometer. It is also the intuition that one has of viscosity: if you put a liquid between your fingers and move your fingers in opposite directions, you will say the liquid is more viscous if it causes a higher resistance.

Let us show that in a fluid where no particles are present, this is a way to get the viscosity. The force acting on the upper wall (called $\boldsymbol{f}_t$) is defined as :

$$\boldsymbol{f}_t = \int_{\text{Top\_Wall}} \boldsymbol{\sigma n} \tag{8.1}$$

where, in this case, $\boldsymbol{n} = (0, n_y)$ is the outward normal of the wall. Let us consider now only the x component of this force, that we will call f_t. We have, for a flow without particles:

$$f_t = \int_{\text{Top\_Wall}} \mu_0(\partial_x u_y + \partial_y u_x) n_y. \tag{8.2}$$

Which can be re-written for a pure shear flow:

$$f_t = \mu_0 \gamma L.$$

Thus, using the x component of the bottom force (that we will call f_b) defined the same way and having opposite sign, one has a relation between the force difference and the viscosity:

$$f_t - f_b = 2\mu_0 \gamma L. \tag{8.3}$$

By substitution, a definition of the effective viscosity of a suspension μ_{eff} can be given as:

$$\boxed{\mu_{\text{eff}} = \frac{f_t - f_b}{2L\gamma} = \frac{l\,(f_t - f_b)}{4LU}} \tag{8.4}$$

The advantage of this definition is that one has only to integrate the velocity field over the top and bottom walls to get the effective viscosity. Thus, it can be done really quickly. Moreover, this method is valid for vesicles as well as for rigid disks or other objects. This method is very close to the experimental one. Indeed, an experimental rheometer is measuring the force needed to impose a given velocity. The drawbacks of having to integrate only on few elements of the boundary of the domain is that the effective viscosity can be more sensitive to numerical noise than in other methods integrating in the whole volume.

Computing the mean of the stress tensor

Another way to compute the effective viscosity of a suspension of rigid disks is to integrate the off-diagonal part of the stress tensor. Let us call it σ_{xy}. Once again, we start by showing that we can recover the viscosity of a Newtonian fluid without particles with this

information and then extend the definition to the effective viscosity of a suspension. The mean of the off-diagonal part of the stress tensor reads:

$$< \sigma_{xy} >= \frac{1}{V} \int_{\Omega} \mu_0 (\partial_x u_y + \partial_y u_x)$$

with V the volume of Ω. It becomes for a pure shear flow of a Newtonian fluid:

$$< \sigma_{xy} >= \mu_0 \gamma$$

The viscosity is thus given by: $\mu_0 = \dfrac{< \sigma_{xy} >}{\gamma}$. Consequently, a way to define the effective viscosity is:

$$\boxed{\mu_{\text{eff}} = \frac{\mu_0}{\gamma V} \int_{\Omega} \partial_x u_y + \partial_y u_x.} \tag{8.5}$$

The advantage compared to the previous method is that we integrate over the whole domain, which tends to smooth more the numerical error. The method is valid for solid particles and can be extended to vesicles if one recalls that the stress to integrate is the total stress and not only the one induced by the fluid. Thus the stress coming from the membrane forces have also to be incorporated in the computation of the viscosity.

Computing the dissipation rate

It as been shown in [65] that the effective viscosity of a suspension can also be calculated by integrating the dissipation in the fluid. We have reported the ways to do this calculation in appendix H. The expression of the effective viscosity is:

$$\boxed{\mu_{\text{eff}} = \frac{\mu_0 \, l}{8LU^2} \int_{\Omega_f} |\boldsymbol{\nabla u} + {}^t\boldsymbol{\nabla u}|^2.} \tag{8.6}$$

This integral is defined on the fluid domain. This is an advantage when only the fluid domain is defined and the inclusions are seen as *holes* on the mesh, as in the conform method or the fat boundary method (see section 3.1 for more details). In our case, with the penalty method presented in algorithm 5, we can access the fluid domain thanks to the Heaviside like function. Since the integrand vanishes inside the particles, we can also integrate over the whole domain without changing the result.

8.1.2 Convergence to the Einstein's viscosity

We made a simple test in order to compare the three different ways to compute the viscosity. It consists in putting a single solid particle of radius $R = 1$ at the center of a rectangular domain Ω of size (L, l) and imposing a shear flow of shear rate $\gamma = 1$. The viscosity has been fixed to $\mu_0 = 10$. The method used to simulate the fluid and the particle is the one described in algorithm 5 of chapter 3, that is, a penalty method for which the position of the particle is tracked by level set method. The thickness of the interface is defined as $1.5\,h$ with h the discretization step of the domain around the particle. When the wanted mesh size becomes too small, the domain is meshed with a thin mesh size h in a vertical band of size $(5R, l)$ and with a bigger mesh size of $5\,h$ everywhere else. The

viscosity is computed using the three different methods exposed above. The theoretical viscosity of a dilute suspension of particles has been calculated by Einstein in 1905 [30]. We recall that the hypothesis of a dilute suspension means that each particle does not interact hydrodynamically with its neighbors. This theory has also been made for an infinite fluid. In our case, we only consider one particle, thus there is no possible mutual influence, and the walls are sufficiently far away for us to be able to neglect their influence on the particle. Thus, we normally are in the range where the Einstein's viscosity formula should work. The Einstein law for the viscosity of a dilute suspension is:

$$\mu_{\text{eff}} = \mu_0(1 + \alpha\Phi) \tag{8.7}$$

where μ_0 is the viscosity of the solvent, α is a coefficient, one has $\alpha = 2.5$ for 3D spheres and $\alpha = 2$ for 2D disks. Here Φ is the volume fraction of particles. Note that it is unrelated to the level set field ϕ which is denoted by a minor ϕ. The volume fraction Φ is defined as the volume (surface in 2D) occupied by the particles divided by the total volume of the suspension.

Computing the numerical volume fraction

In penalty / FPD methods, the following question arises: were is exactly the limit between the particle and the fluid? Since the particle is seen as part of the fluid having a huge viscosity, and since the transition from the viscosity of the fluid to the one of the particle is smooth, it is not obvious where the precise boundary of the particle should be defined. The simplest answer that one could give is that, in our case, the particle is defined thanks to the level set function by the region where $\phi < 0$. Doing this simple approximation leads to an underestimate of the real value of the volume fraction, the particle *seen* by the system is actually slightly bigger than that. This problem as been addressed by L.Jibuti [48] in his thesis for a 3D particle, where he did two numerical experiments for which the result is known to calibrate the real size of the particle. The first experiment was to compute the Stokes force on a sphere moving in a fluid at rest. This problem has no analytic solution in 2D (Stokes paradox) and was not accessible to us. The second one was the same simulation that we are concerned with here, *i.e* recovering the Einstein viscosity. We did similar tests and found that a correct behavior is obtained by setting the particle region as the region in which $\phi < \frac{\varepsilon}{2}$, that is to say for this simulation $\phi < 0.75\,h$. Thus we compute the numerical volume fraction as:

$$\Phi_h = \frac{1}{V}\int_\Omega \chi(\phi < 0.75\,h). \tag{8.8}$$

A consequence is that for a given radius, the size of the mesh has influence on the volume fraction of the suspension.

Results of the test

The test has been made for several mesh sizes. For $h < 0.1$, the refinement in the band around the particle explained previously has been used. The following error has been calculated:

$$e_{\text{visco}} = |\mu_{\text{eff}} - \mu_0(1 + 2\Phi_h)| \tag{8.9}$$

which is the absolute error of the effective viscosity compared to the theoretical viscosity at this numerical volume fraction. The length of the box in the x direction has been taken equal to $L = 50$ and two different channel widths have been tested: $l = 30$ and $l = 8$. To differentiate the methods, we will call the **force** method the method using expression (8.4) to compute the effective viscosity, the **stress** method the one using (8.5) and finally the **dissipation** method the one using equation (8.6).

We have reported in figure 8.2 the effective viscosity computed by each method for both channel widths. The theoretical value of the viscosity using the volume fraction of a particle with a fixed radius $R = 1$ has been represented as a red line, it is simply given in this problem by: $\mu_0\,(1 + 2\,\frac{\pi}{L\,l})$. The viscosity calculated using the numerical volume fraction is also represented as red squares. It is given by: $\mu_0\,(1 + 2\,\Phi_h)$. Of course, since $\Phi_h \to \Phi$ when $h \to 0$, the theoretical effective viscosity computed with Φ_h tends to the absolute theoretical one for small h.

We have also reported in figure 8.3 the values of e_{visco} for the different methods. The errors are plotted in logarithmic scale and the slope of the fit with a linear law has been given in the legend.

We see in figures 8.2(a) and 8.3(a) that for the problem with $l = 8$, the viscosities computed by the force method and the stress method are giving exactly the same results and we can almost not distinguish the different points on the graphs. This can be explained by the fact that the same components of the stress tensor is used to compute both of them whereas the dissipation method uses all the components of the stress tensor. Anyway, all the methods are giving comparable results and the rate of convergence to the theoretical value is 1 for all of them. The absolute value of the error can seem high compared with the system $l = 30$. This can be explained by the fact that the unconfined hypothesis starts to be false at this confinement. The theoretical value of the viscosity should be slightly higher than the one that we took, and then, the absolute value of the error should be of the same order as for the system $l = 30$.

The figures 8.2(b) and 8.3(b) show the results for the problem $l = 30$. We see that contrary to the previous result, the effective viscosity computed by the force method is completely wrong. It does not converge to the theoretical value. Whereas the other methods still have a rate of convergence of 1. This can be explained by the fact that the influence of the particle on the walls decreases with the distance. Theoretically, even the small influence that the particle exerts on the walls should be enough to measure the viscosity of the system. But in practice, at a certain distance, the disturbance induced by the particle on the values of the forces exerted on the walls are too small to be captured numerically. To emphasize this phenomenon, we have reported in figure 8.4 the following coefficient:

$$M_f = |\mu_0(\partial_x u_y^p + \partial_y u_x^p)|$$

where $\boldsymbol{u}^p = (u_x^p, u_y^p)$ is the perturbation velocity defined by $\boldsymbol{u}^p = \boldsymbol{u} - \boldsymbol{u}^{\text{shear}}$. Here, M_f has been chosen this way because at the boundary, it gives directly the magnitude of the perturbation on the force induced by the particle. We see clearly in figure 8.4(a) that for a box with $l = 8$, the boundaries see a non-negligible perturbation whereas in figure 8.4(b), for $l = 30$, the perturbation almost vanishes at the boundaries. The two other

methods, for which we integrate over the whole domain Ω or the whole fluid domain Ω_f do not suffer from this drawback.

Consequently, we can say that we validated our numerical method to measure the effective viscosity of a suspension by recovering a rate of convergence of 1 of the numerical error when compared to the Einstein viscosity of a suspension. We also showed a limitation of the method of the forces that we are able to explain. We can now do rheological studies of 2D disks with a controlled error in the effective viscosity.

8.2 A simulation method for confined semi-dilute suspension

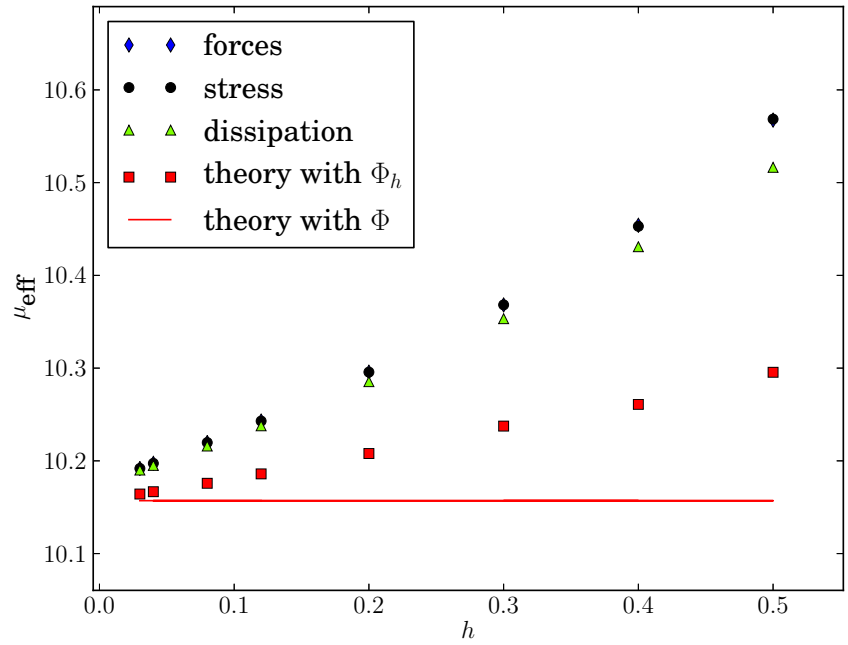
We have shown in section 8.1 that our code was able to recover the Einstein viscosity which has been derived under the hypotheses that the suspension is dilute *i.e.* the particles are not interacting with each other and the fluid is infinite that is to say that the effects of the boundaries of the domain are negligible. A long time after Einstein's theory, Batchelor and Green [4] extended the formula to a suspension of semi-dilute particles. In this work, they were considering pair interactions in an infinite fluid. They found that the effective viscosity of a semi-dilute suspension should obey the expression:

$$\mu_{\text{eff}} = \mu_0(1 + \alpha\Phi + \beta\Phi^2) \tag{8.10}$$

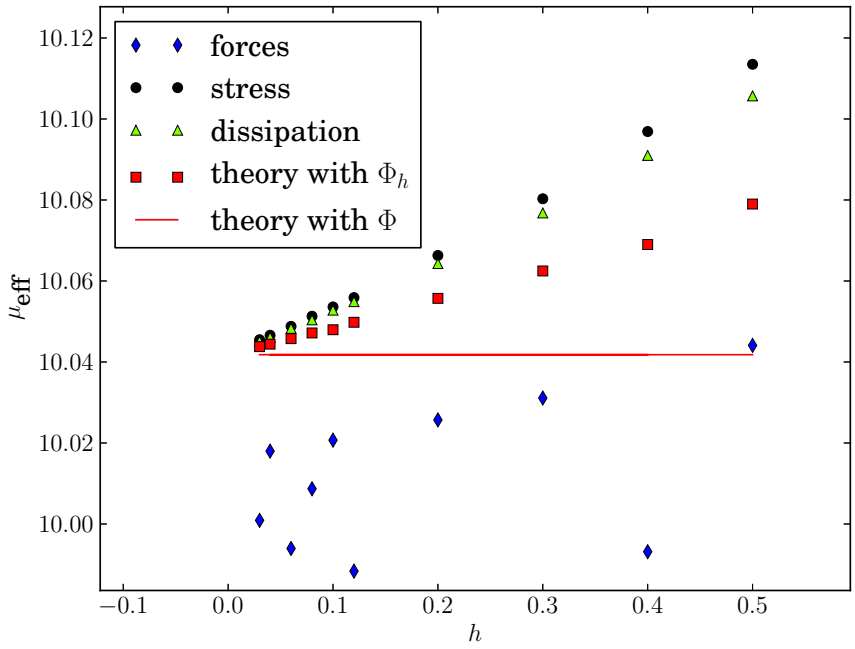
where β is a coefficient which is $\beta = 5.2$ for a homogeneous suspension of 3D, non Brownian, neutrally buoyancy particles under shear flow.
Finding the β coefficient is not a simple task, even numerically. Indeed, getting the viscosity of a suspension of hard disks in pure shear flow is not so difficult, but each configuration of the suspension corresponds to a different viscosity. Thus the β factor should represent the value for the more probable configuration and thus it is usually a mean on a great number of simulations with a random configuration.
To our knowledge, the value of the factor β has not been derived for a 2D suspension. Recently, some work has been done to incorporate the effects of the confinement on the coefficients α and β in 3D. Davit and Peyla [22] have shown that the α coefficient increases when the confinement $C_{\text{n}} = \dfrac{2R}{l}$ is greater than about $\dfrac{1}{6}$. They have also shown that the coefficient β decreases for a confinement greater than 0.2 and even becomes negative for confinements greater than 0.4. These results have been confirmed experimentally [84] and the decrease of the influence of the particle-particle interaction (the coefficient β) has been investigated in [93], still in 3D.

There are two main goals in this section. The first one is to show a numerical method for the fast simulation of the viscosity of a great number of different configurations. It consists of creating some maps of the mean stress tensor thanks to finite element solutions of a simplified system, and re-use these maps to obtain the viscosity of a suspension of particles. Indeed, since the Stokes equations are linear, the mean xy component of the stress tensor of a system of two particles interacting in a shear flow can be decomposed as:

$$< \sigma_{xy} > = < \sigma_{xy}^{\text{shear}} > + \Phi_1 < \sigma_{xy}^{\text{part 1}} > + \Phi_2 < \sigma_{xy}^{\text{part 2}} > + (\Phi_1\Phi_2) < \sigma_{xy}^{2\text{ part}} > \tag{8.11}$$

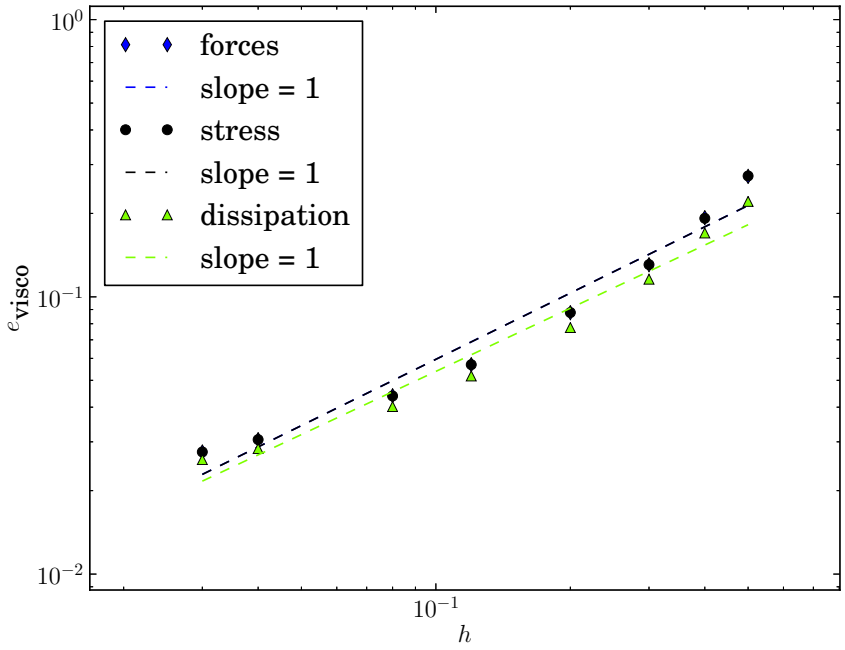


(a) $l = 8$

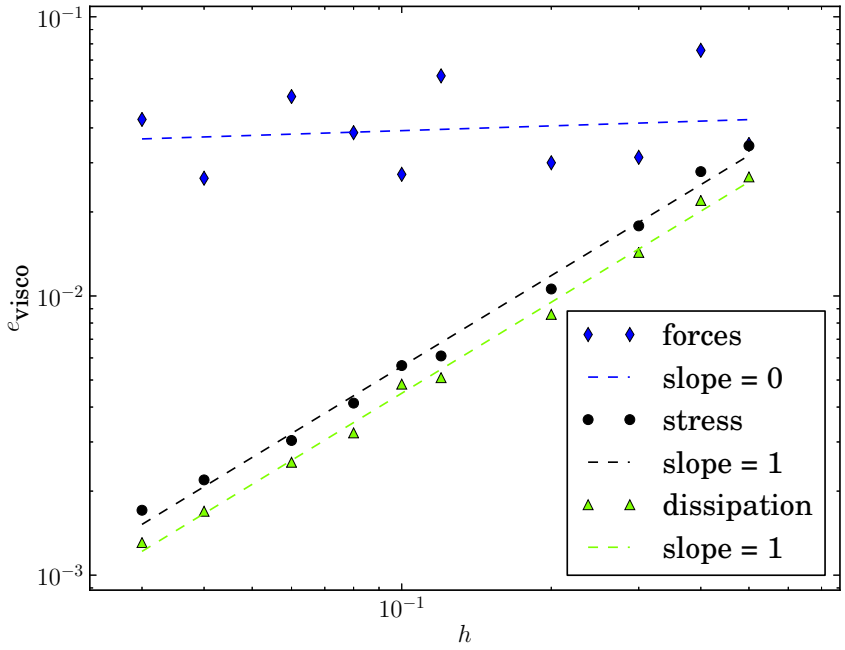


(b) $l = 30$

Figure 8.2: Effective viscosity computed by three different methods as a function of the mesh size. The theoretical value of the viscosity using the volume fraction of a particle with a fixed radius $R = 1$ is represented as a red line. The theoretical value computed using the numerical volume fraction is represented as red squares. This value naturally converges to the other theoretical value for small h.

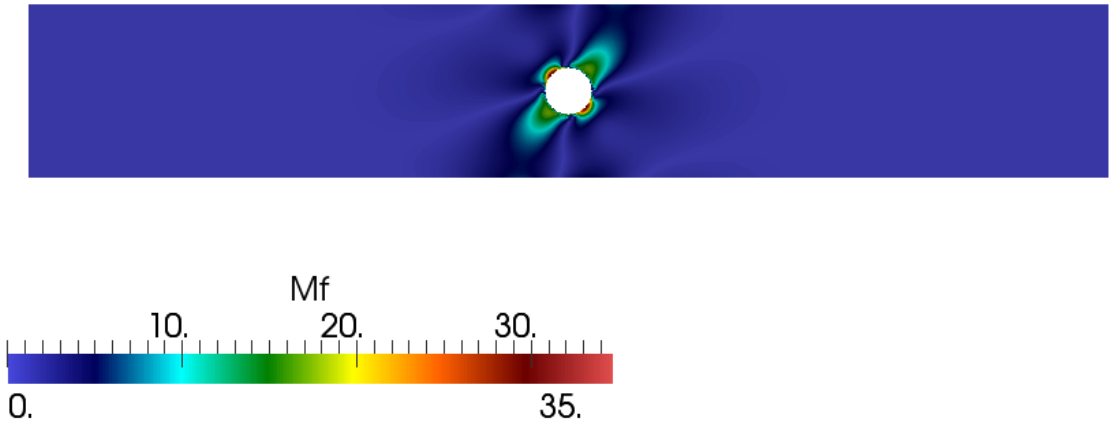


(a) $l = 8$

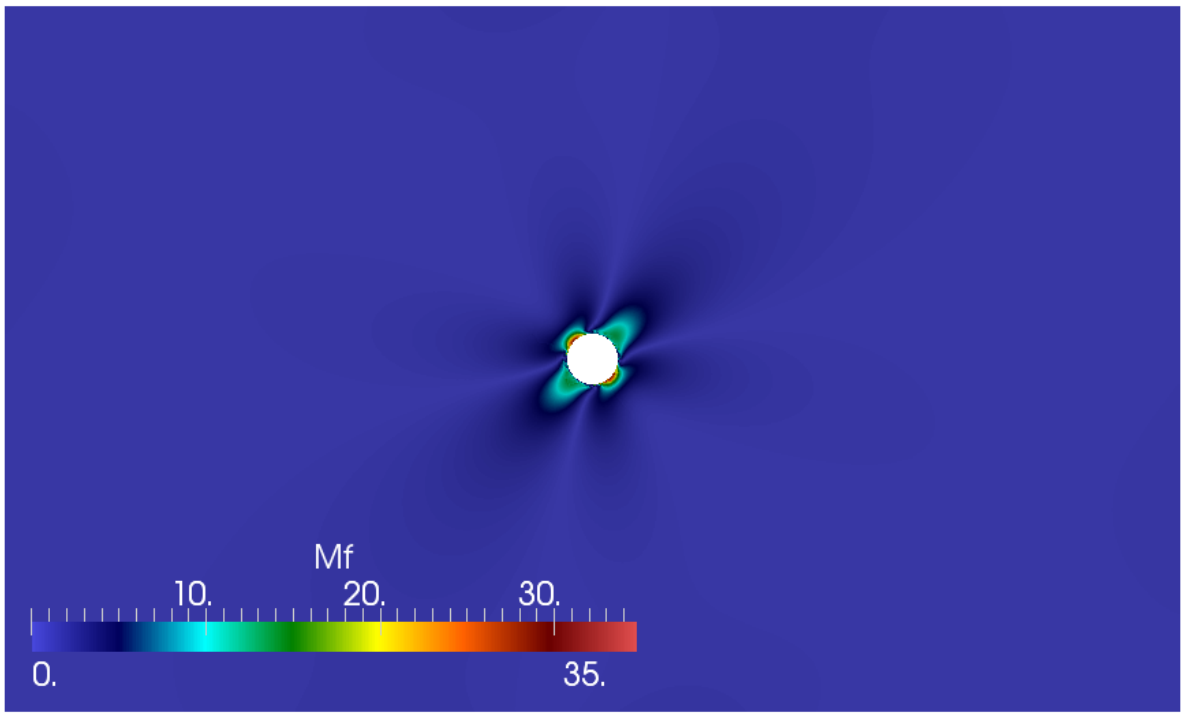


(b) $l = 30$

Figure 8.3: Error on the viscosity (e_{visco}) computed with different methods as a function of the mesh size in logarithmic scale. A fit of each line with a linear relation is shown as dashed lines and each slope is reported in the legend.



(a) $l = 8$



(b) $l = 30$

Figure 8.4: Representation of M_f on all the domain for $l = 8$ and $l = 30$. One can see that the walls of the larger domain do not experience a sensible perturbation due to the presence of the particle.

where $< \sigma_{xy}^{\text{shear}} >$ is the contribution of the pure shear flow ($< \sigma_{xy}^{\text{shear}} >= \mu_0\gamma$), $< \sigma_{xy}^{\text{part 1}} >$ and $< \sigma_{xy}^{\text{part 2}} >$ are respectively the contributions of the first and second particle and $< \sigma_{xy}^{2\text{ part}} >$ is the contribution of the interaction between particles 1 and 2 together. Φ_1 and Φ_2 are respectively the volume fractions of particles 1 and 2. For the decomposition, we extract from the mean tensor the volume fraction corresponding to each contribution. It is thus possible to create a map of the contribution of a single particle in a shear flow for every possible position in a channel. Then, we can use this map to extract the contribution of the interaction of two particles. Finally, for any random configuration of suspension, it is possible to search in the map the value of the contribution of each particle and each pair of particles. This way, a fast solution of the viscosity of a suspension of such a system can be obtained as long as the hypothesis of neglecting higher order interaction than pair interactions is enough. The method will be applied here for suspensions of solid disks but it can be extended to vesicles or drops easily.

The second objective of this section is to determine the behaviors of suspension of confined disks. Indeed, as we already explained, the understanding of the effect of the deformability of the vesicles on the viscosity of 2D suspensions starts by the knowledge of the behaviors of disks. Nothing indicated clearly that the effects found in [22, 84, 93] should still be true in 2D.

8.2.1 The viscosity of a confined dilute suspension of solid disks

The first step is to study and get a precise map of the effective viscosity of a rigid particle in a confined shear flow. We made the simulation represented in figure 8.5. The shear rate γ has been kept equal to 1 for all the simulations. The box was taken large enough ($L = 60$) so that no boundary effects were present in the x direction, and the wall distance l has been varied in order to study the effect of the confinement. The mesh size was taken equal to $h = 0.13$, the radius $R = 1$, the viscosity of the fluid $\mu_0 = 10$ and the thickness of the interface $\varepsilon = 1.5h$. Since the x direction is very long compared to the y one,

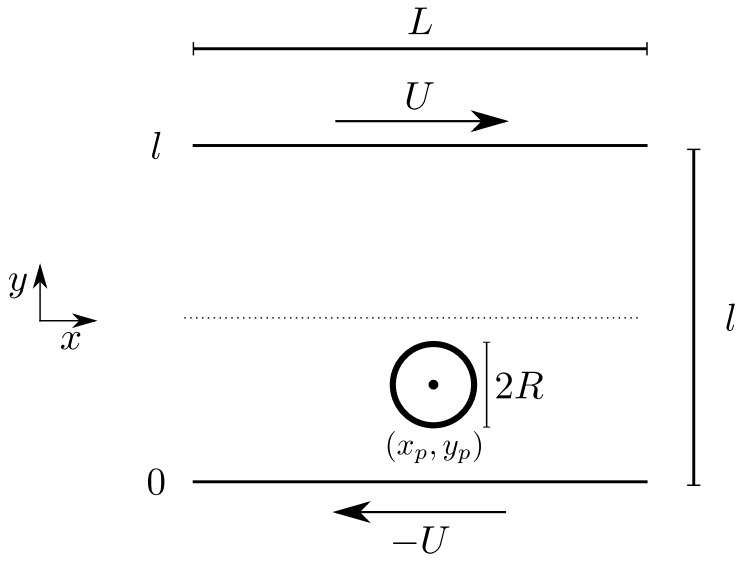


Figure 8.5: Scheme of the simulation of a single particle in a shear flow.

the viscosity of this system should depend only on the confinement C_n and the distance between the particle and the wall (that we called y_p). Following equation (8.11), the contribution of the particle to the viscosity is obtained by:

$$< \sigma_{xy}^{1\text{ part}} > (y_p, C_n) = \frac{< \sigma_{xy} > - < \sigma_{xy}^{\text{shear}} >}{\Phi^{1\text{ part}}} \tag{8.12}$$

$< \sigma_{xy} >$ is measured during the simulation, we have $< \sigma_{xy}^{\text{shear}} >= \gamma\mu_0 = \mu_0 = 10$ and $\Phi^{1\text{ part}} = \dfrac{\pi R^2}{(l\ L)} = \dfrac{\pi(1+0.75h)^2}{(l\ L)}$. The problem is symmetric with respect to the center of the channel. Thus, we can restrict our interest on the range: $\dfrac{l}{2} < y_p < l - R$.
The simulation consists in setting a particle at different y_p in the channel in the range given above and measuring of the viscosity. Since the Stokes equation does not depend on time, the measure is obtained instantaneously. The figure 8.6 shows the result of such a simulation for several confinements. The result is that when the particle approaches the wall, the viscosity of the system increases. The rate of increase is higher when the particle is closer to the wall. In the figure, we can see that for a small confinement, we recover the Einstein formula, which gives, for this simulation: $< \sigma_{xy}^{1\text{ part}} >= 2\mu_0 = 20$. For medium confinement, the viscosity is high near the wall and decreases to the Einstein viscosity when approaching the center of the channel. It is interesting to see that for strong confinements, the viscosity never decreases to the Einstein limit. This is due to the fact that at these confinements, the presence of the walls is still important at the distance $\dfrac{l}{2}$.

In [93] the authors used theory coupled with numerical results to show that the viscosity of a 3D particle in the center of the channel increases when the confinement increases. They have also shown that the viscosity of a uniform dilute suspension of particles increases with the confinement. The direct simulation of $< \sigma_{xy}^{1\text{ part}} > (y_p, C_\mathrm{n})$ was not done but the two limiting cases: *very confined* and *particle not too confined with the position near the center of the channel* were studied and gave results similar to the one obtained figure 8.6.

Alexander Farutin, a physicist in our group, made a development similar to the one done in [93] where he considered the problem of a single circular particle near a single wall. The fluid is then infinite in one direction. One can write the velocity in a complex form $(u_x + iu_y)$ and develop it as a series of integer powers of complex coordinate $(x - x_p) + i(y - y_p)$ and its complex conjugate. The reflection method is employed to satisfy the null boundary conditions at the wall (one considers that there is another particle on the other side of the wall at the same distance). He found the solution $\boldsymbol{u}$ and derived the mean stress; developped as series of integer powers of the distance from the center of the particle to the wall $(l - y_p)$. The result is, if one cuts the expansion at the 6^{th} order and normalizes by the viscosity of the fluid:

$$\frac{< \sigma_{xy}^{1\text{ part th}} > (y_p)}{\mu_0} = 2 + \frac{2}{(l-y_p)^2} - \frac{1}{4\,(l-y_p)^4} + \frac{15}{16\,(l-y_p)^6}. \tag{8.13}$$

We have plotted in figure 8.7 the same data as in figure 8.6 and added the values of equation (8.13) for the different confinements. An offset has been added to separate the curves and we have split the high and low confinements for more visibility.

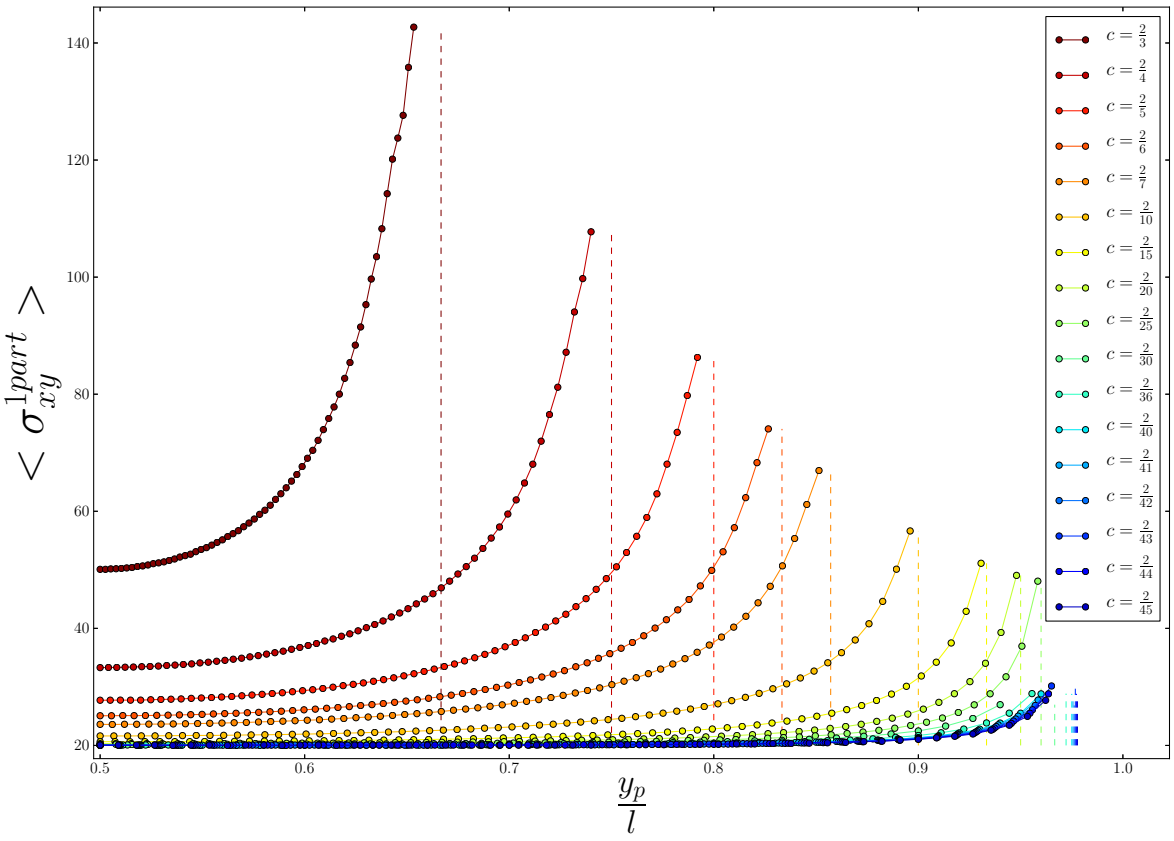
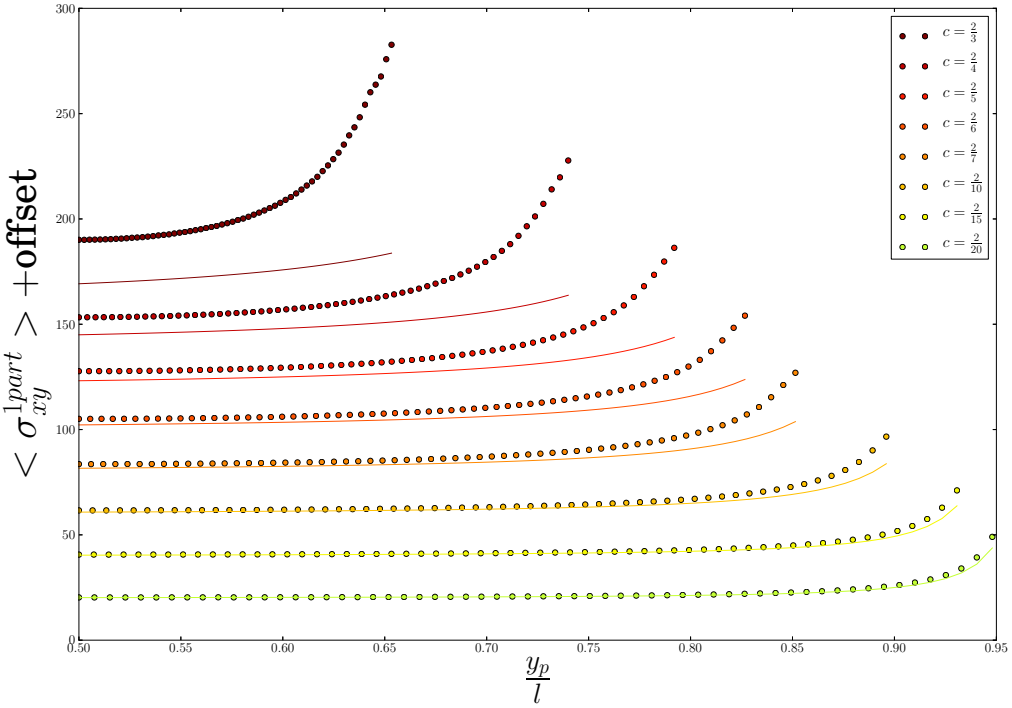


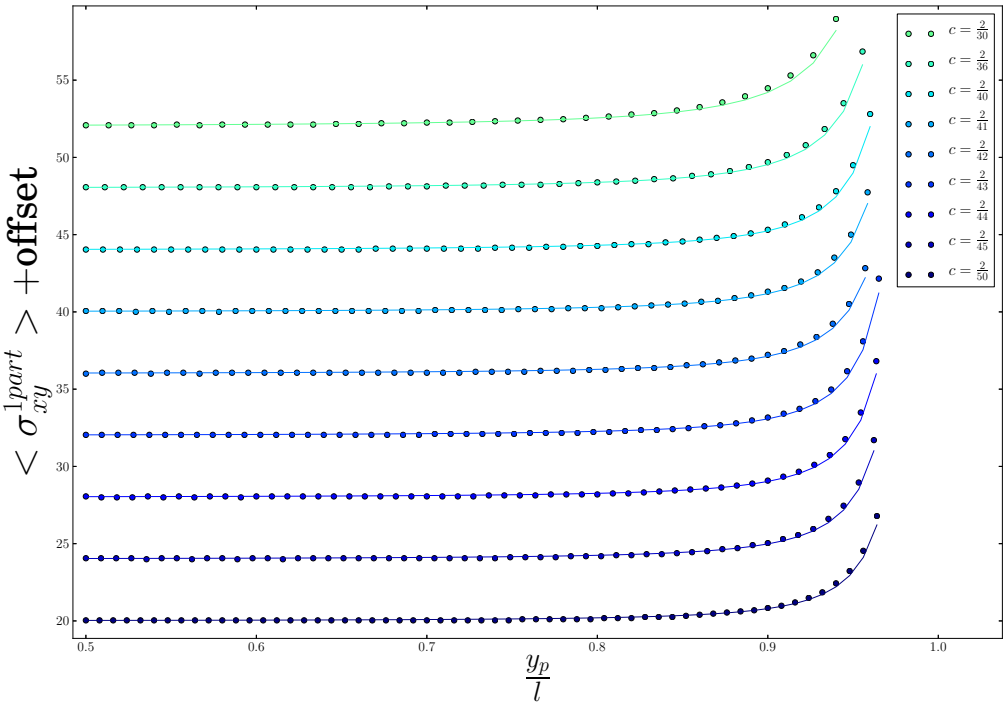
Figure 8.6: Contribution to the mean stress tensor of the particle. The position of the particle has been normalized by the length of the channel l for lisibility. The confinement of each curve $C_\mathrm{n} = \dfrac{2R}{l}$ is reported in the legend. The limit position that the particle can have without touching the wall $l - R$ is marked as a vertical dashed line, normalized also by l. Thus, the position 0.5 represents the center of the channel, and the dashed line, the top accessible. The dots are the results of the simulation, the line joining them is a guide for the eyes.

We can see in figure 8.7(b) that the equation (8.13) describes correctly our results except for the closest points to the wall. This might be due to the fact that we only took up to 6^{th} order terms in the expansion. The finite element solution might also not be precise very close to the wall because of the presence of the interface thickness of the particle. Moreover, for the larger confinements (figure 8.7(a)), we can see that the simulation points are not correctly represented by the model. Once again this was expected since the region of the large gap assumption is smaller for higher confinement and the interaction with the second wall is not negligible. The conclusion of this comparison is that the model (8.13) is valid for confinements smaller than about 0.13.

The conclusion of this section is that at the same time we have shown how we obtained the *maps* of the contribution to the viscosity of a single particle for many positions in the channel and several different confinements. In the next section we will show that it is possible to use linear interpolations of these curves to extract the two particles contribution.



(a) High confinements: $\frac{2}{20} \leq C_\text{n} \leq \frac{2}{3}$



(b) Low confinements: $\frac{2}{30} \leq C_\text{n} \leq \frac{2}{50}$

Figure 8.7: Contribution to the viscosity of a single particle displaced from the center of a channel. The points are representing the same data as in figure 8.6 except that we added an offset to separate the curves from each other. The lines represent the values of equation (8.13) with the same offset, which is the model of a particle near a single wall. For lisibility reasons, the high and low confinements have been split into two different graphs.

In the same time, we explored the behavior of the contribution to the viscosity of a solid circular particle in presence of walls. We can already extend this work on the viscosity of dilute suspensions. Indeed, for a dilute suspension of particles, the contribution of each particle is simply added. Thus, for a suspension of N_p similar particles of the same volume where the position of each one is known, one can express the viscosity of the whole suspension as:

$$< \sigma_{xy} > = < \sigma_{xy}^{\text{shear}} > + \Phi^{1 \text{ part}} \sum_{n=1}^{N_\mathrm{p}} < \sigma_{xy}^{1 \text{ part}} > (y_n) \tag{8.14}$$

where y_n is the vertical position of the particle n. For a large number of particles, one can drop the sum to integrate over the length l of the channel and multiply by the probability density $P(y)$ of finding a particle at the vertical position y. The equation 8.14 becomes:

$$< \sigma_{xy} > = < \sigma_{xy}^{\text{shear}} > + \Phi^{1 \text{ part}} \int_R^{l-R} < \sigma_{xy}^{1 \text{ part}} > (y) P(y) N_\mathrm{p} dy \tag{8.15}$$

$$= < \sigma_{xy}^{\text{shear}} > + \Phi^{\mathrm{N} \text{ part}} \int_R^{l-R} < \sigma_{xy}^{1 \text{ part}} > (y) P(y) dy \tag{8.16}$$

where $\Phi^{\mathrm{N} \text{ part}} = N_\mathrm{p} \Phi^{1 \text{ part}}$ is the volume fraction of all the particles, and we recall the property of a probability density : $\int_0^l P(y)dy = 1$. With the expression (8.16), we see clearly the different influences of the distribution of particles and the intrinsic function of contribution to the viscosity.

From this formula, we can also extract the minimum and maximum viscosity values that a confined dilute suspension of a given volume fraction can have. Indeed, the minimum effective viscosity is obtained by a distribution where all the particles are aligned on the center of the channel. The probability density of such configuration is given by $P(y) = \delta(y - \frac{l}{2})$ where δ is the Dirac function (the real one, not the smoothed δ_ε used in the level set framework of course). On the contrary, the maximum viscosity is obtained at the opposite when all the particles are almost touching the wall: $P(y) = \delta(y - R)$ or $P(y) = \delta(y - (l - R))$. A common configuration is the random distribution, in this case, there is the same probability to find a particle for any accessible position in the channel. Such a configuration is given by $P(y) = \frac{1}{l - 2R}$.

8.2.2 The contribution to the viscosity of the pair interaction

Since we are able to get the contribution of a single vesicle in a confined shear flow, it is also possible to extract the contribution to the viscosity of the pair interaction of two particles. It is given by:

$$< \sigma_{xy}^{2 \text{ part}} > = \frac{< \sigma_{xy} > - \Phi_1 < \sigma_{xy}^{1 \text{ part}} > (y_1) - \Phi_2 < \sigma_{xy}^{1 \text{ part}} > (y_2)}{\Phi_1 \Phi_2} \tag{8.17}$$

where y_1 and y_2 are the positions of the two particles. The contribution of one particle $< \sigma_{xy}^{1 \text{ part}} >$ can be either taken as the interpolation of the curves of the previous study or recomputed by solving the same system but taking into account only one particle. We made the following simulation: for a given confinement, we start a simulation with two

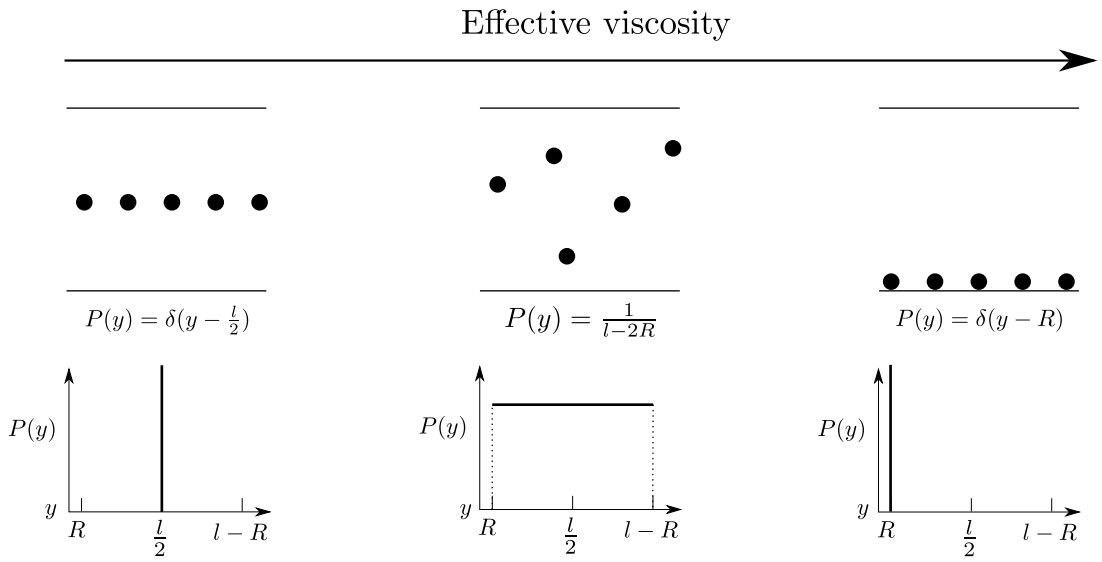


Figure 8.8: Three remarkable configurations at the points of view of the effective viscosity and their probability densities. The first and last one are respectively the configurations giving the lowest and highest effective viscosity. The middle one is the one often considered: a homogeneous suspension. In these schemes, the x direction has been shrunk for a better representation, but the particles should be separated of at least 5 radii to consider the suspension as dilute.

particles far away from each other in the x direction. Then, by keeping the y position of each particle the same, we measure the viscosity of the fluid when approaching the two particles. This simulation is repeated for many different y positions and different confinements. The two particles are located symmetrically to the center of the channel. Consequently, for a given confinement, the quantity $< \sigma_{xy}^{2\,\text{part}} >$ depends only on $\Delta x = |x_1 - x_2|$ and $\Delta y = |y_1 - y_2|$. The range in which Δy varies is $0 < \Delta y < l - 2R$. The maximum value given for Δx is $15R$, for Δx higher than this value, the contribution is small enough to be neglected. The minimum values that Δx and Δy can have is 0 (particles aligned) or higher if the particles are too close (contact is avoided). The scheme of such a simulation and the corresponding notations are shown in figure 8.9. The result of the simulation is shown in 2D for the different Δy in figure 8.11(a) and in 3D for all the $(\Delta x, \Delta y)$ in figure 8.11(b). We also reported in figure 8.10 the snapshots of the value of $\sigma_{xy}^{2\,\text{part}}$ for three different Δx showing characteristic behaviors. The 2D plot of the contribution is also given in figure 8.12 for a confinement twice greater than the one of figure 8.11(a).
We can see that, as expected, when the distance between the particles is large enough ($\sim 10R$), the contribution of the pair interaction vanishes. This can be seen graphically with the very low values of $\sigma_{xy}^{2\,\text{part}}$ in figure 8.10(a), the plateau on the 3D surface of figure 8.11(b) and the systematic 0-value of the curves in 8.11(a) for large Δx.

It is also interesting to see that on the other hand, when the two particles are on the top of each other ($\Delta x = 0$), the contribution of the pair to the viscosity is negative. This is illustrated by the figure 8.10(c), the negative values at ($\Delta x = 0$) in figure 8.11(a) and the negative gradient of the 3D surface.

Finally, we observe a maximum of the contribution to the viscosity for sufficiently

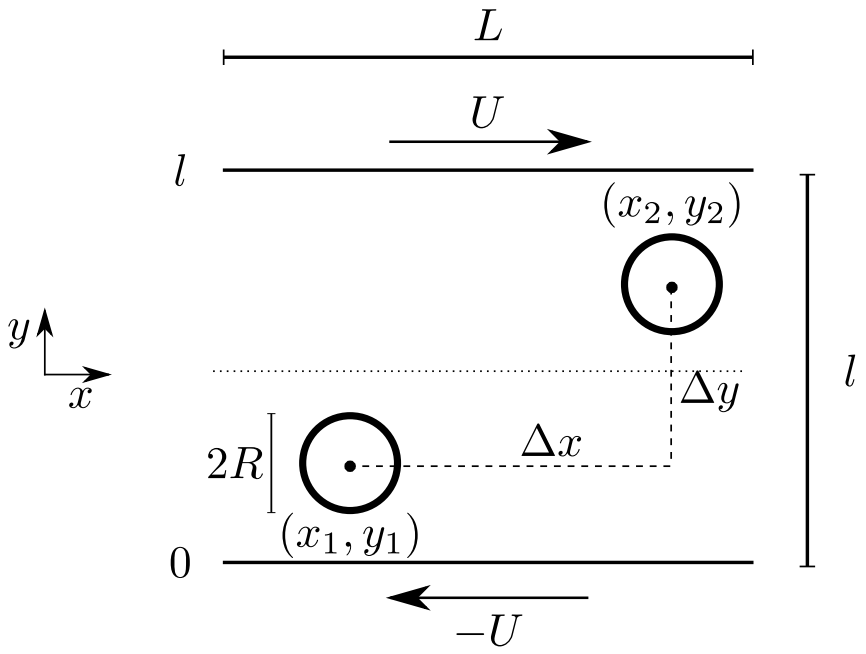


Figure 8.9: Scheme of the simulation of the two particles interaction in shear flow.

close particles with a certain disposition. This peak can be large and have a great value. It appears when the particles are aligned with an angle of around 45° with respect to the direction of the flow as we can see in figure 8.10(b).

It is interesting to see how evolve these behaviors for higher confinements. We have reported in figure 8.12 the contribution to the viscosity of a pair of particles in a channel having a confinement twice higher than the one of graph 8.11(a). The first obvious thing is that the high interaction occurring when the particles are close to each other is enhanced by the confinement. But this actually happens only for relatively high Δy, since, for lower Δy the particles would touch at the same Δx.

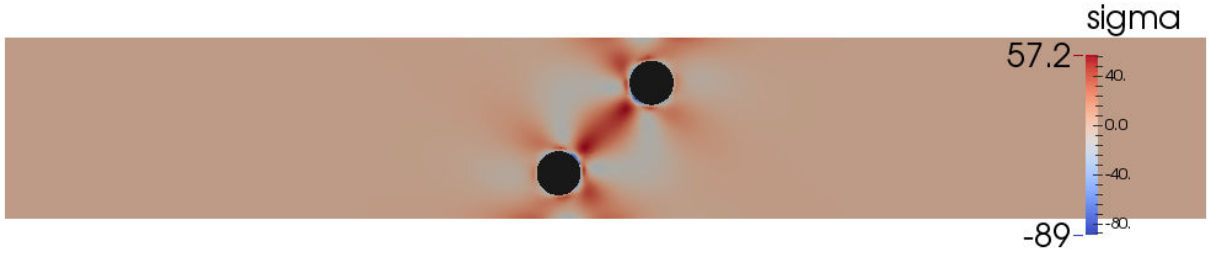
The other important remark to make is that from medium to low Δx at this confinement, almost all the curves are negative on a large domain. Thus, at high confinement, adding a pair of particles at a random place in the box would in general add a negative contribution to the overall viscosity.

The same simulation has been done for different confinement in the range $0.04 < C_{\mathrm{n}} < 0.33$. The *maps* generated are used to compute the viscosity of a semi-dilute suspension.

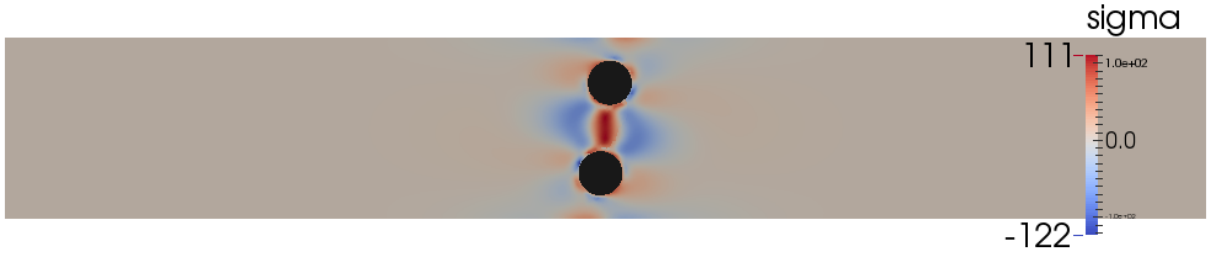
8.2.3 Extension to the simulation of a semi dilute suspension

We will show that the previous information can be used to compute the viscosity of a semi-dilute suspension of mono-disperse rigid disks. Indeed, for a given random configuration of positions, the contribution of each particle to the viscosity can be computed by interpolating the curves presented in section 8.2.1. Then, for each pair of particles, doing a 2D interpolation of the information presented in section 8.2.2 leads to the second order term of the effective viscosity. Two remarks have to be made about the accuracy of the method. The first one is that higher order interactions are neglected. These interactions can be dominant for high concentration, thus we limit our study to the semi-dilute case ($\phi < 0.2$).

(a) The particles are far away and almost do not interact.



(b) The pair interaction is at its maximum when the particles are aligned at around 45° with respect to the shear direction.



(c) The interaction is at its minimum when the particles are aligned along their y axis ($\Delta x = 0$).

Figure 8.10: The value of the field $\sigma_{xy}^{2\,\mathrm{part}}$ for a confinement $C_\mathrm{n} = 0.1$ and a given Δy at three different Δx.

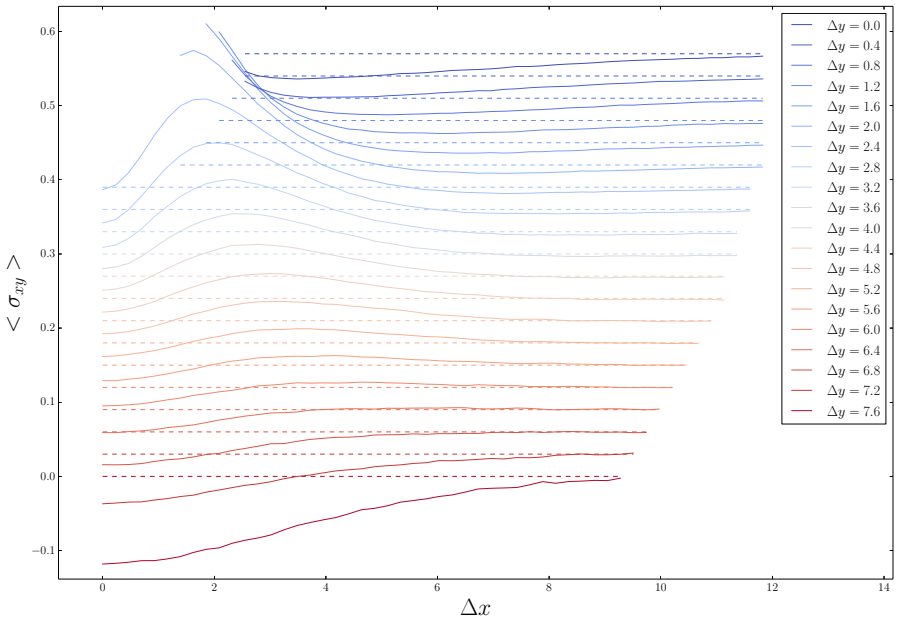
The second hypothesis made and which might be a source of error is that we will consider that for a given confinement, the pair interaction only depends on Δx and Δy. Thus we do not take into account the fact that the absolute distance of the pair of particles to the wall might change its viscosity contribution.

Effect of the confinement on the viscosity of a 2D semi dilute suspension

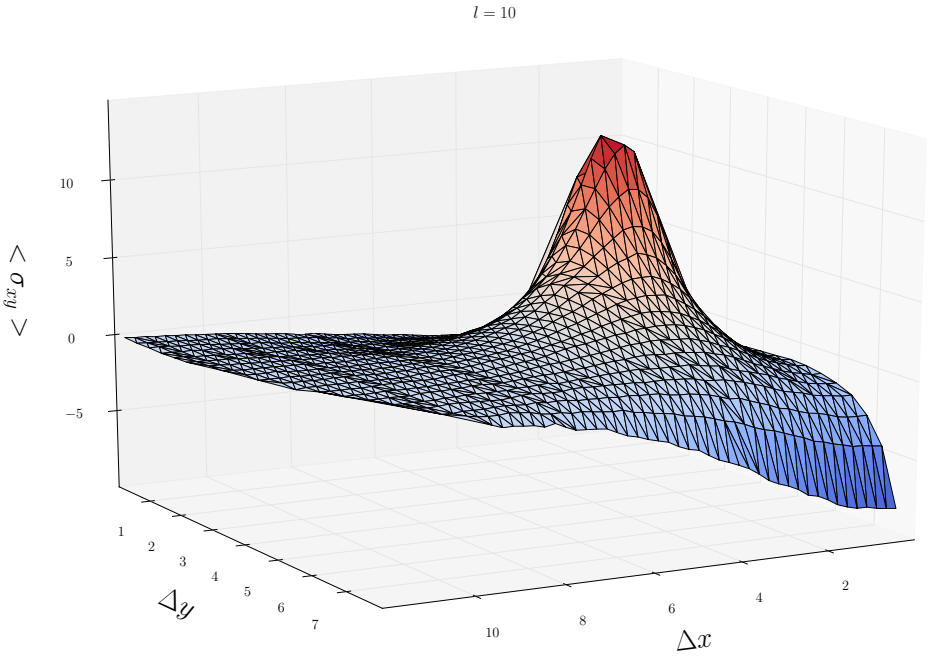
The processes to compute the viscosity of a suspension of a given confinement at a given volume fraction Φ is first to compute the number of particles needed to achieve such a volume fraction:

$$N_\mathrm{part} = \mathrm{int}\left(\frac{\Phi l L}{\pi R^2}\right) \tag{8.18}$$

where "int" represents the closest integer of the argument. Then N_part coordinates (x, y) are generated randomly in the range $([R, L-R], [R, l-R])$. The viscosity of this config-



(a) 2D representation of the simulation of the contribution to the viscosity of a pair of particles. For each different Δy, the quantity $< \sigma_{xy}^{2\,\text{part}} >$ is measured for many Δx. The curves are represented with an offset to avoid their overlapping. A dashed line represents the value of the offset for each curve. The color represents the value of Δy for each curve which are reported in the legend.



(b) The quantity $< \sigma_{xy}^{2\,\text{part}} >$ represented in elevation. The bottom plane represents the values of Δx and Δy. The color represent the value of $< \sigma_{xy}^{2\,\text{part}} >$. This quantity has been measured for every point represented on the 3D surface. The lines between the points are just a guide for the eyes.

Figure 8.11: The value of $< \sigma_{xy}^{2\,\text{part}} >$ represented in 2D and 3D for a confinement of $C_\text{n} = 0.2$.

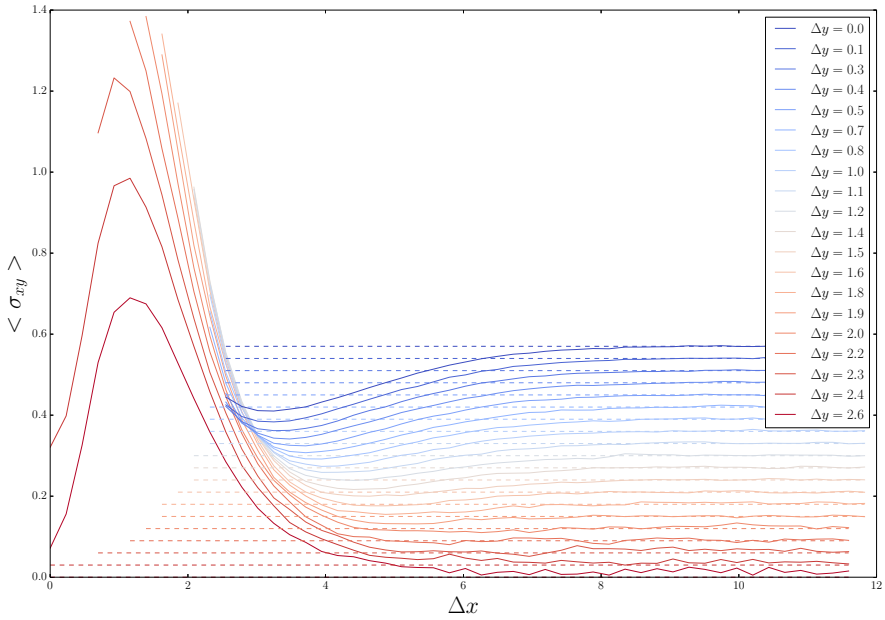


Figure 8.12: This figure represents the same graph as in figure 8.11(a), the contribution to the viscosity of a pair of particles, except that it is done for a higher confinement: $C_n = 0.4$.

uration is computed using the formula:

$$< \sigma_{xy} >=< \sigma_{xy}^{\text{sh}} > +\Phi_1 \sum_{i=1}^{N_{\text{part}}} < \sigma_{xy}^{1\text{ part}} > (y_i)+\Phi_1^2 \sum_{i=1}^{N_{\text{part}}-1} \sum_{j=i+1}^{N_{\text{part}}} < \sigma_{xy}^{2\text{ part}} > (|x_i-x_j|, |y_i-y_j|) \tag{8.19}$$

where the values of $< \sigma_{xy}^{1\text{ part}} > (y_i)$ and $< \sigma_{xy}^{2\text{ part}} > (|x_i - x_j|, |y_i - y_j|)$ are extracted from the data presented in sections 8.2.1 and 8.2.2 thanks to the interpolation tools of SCIPY [51].

Since each configuration has its own effective viscosity, this process is repeated on a great number of different configurations. Finally the mean value of all these viscosities is taken as the value for this particular volume fraction and the standard deviation is taken as the error bars.

The result of such simulation is shown in figure 8.13 where three different confinements have been plotted. They have been fitted with the expression (8.10). The coefficients of these fits and some other confinements are shown in figure 8.14.

The figure 8.14 shows a similar behavior as [22] which confirms that the same decrease of the interaction between the particles for high confinement also occurs in 2D.

Link with the two particle interaction effect

Thanks to the work done in the previous section, we can try to explain qualitatively what is happening. At low confinement, when a pair of particles is added, if the particles are not too far from each other, by addition to the linear single particle contributions, the interaction contribution exists. If we look at the figure 8.11(a), we see that there are large

areas where this contribution is positive. Moreover, the values where the contribution has a high negative value are obtained for high Δy. There are fewer possibility to obtain large Δy than small ones. For example, in the extreme case, to obtain a Δy of $l - 2R$ the only possibilities are to have $(y_1 = R, y_2 = l - R)$ and $(y_1 = l - R, y_2 = R)$.

Thus, we can say that globally, for this confinement, a pair of particles randomly disposed will add a positive contribution to the viscosity. Consequently, we expect the second order viscosity coefficient viscosity to be positive.

At the opposite, the figure 8.12 shows that for a strong confinement, a randomly chosen pair of particle will add a negative contribution to the linear viscosity. Indeed, there are large regions where the pair contribution is negative. Moreover these contributions are more probable since they exist at low Δy. At the contrary, the areas where the contribution is positive are happening for large Δy which have few chances to happen. Consequently, we can say that adding a particle in such a system would add a negative value to the second order viscosity, thus one can expect that the second order coefficient viscosity of such a confinement is negative.

Discussion about the method

The drawback of this method is the systematic errors that are added to the problem. Making the maps and extracting their data by interpolation are error prone and the coefficients obtained from the fits of the simulation of a large number of objects might not be very precise.

The whole simulation to obtain the figure 8.14 took only few minutes since we already had the maps of the single particle and pair contribution. This is one of the advantages of this method, since the maps are calculated, the rest of the simulation is done in a really short time.

The other advantage of not simulating directly a large number of particles as solution of the finite element problem, is that we are able to explain the behavior of a collection of objects thanks to their individual contributions.

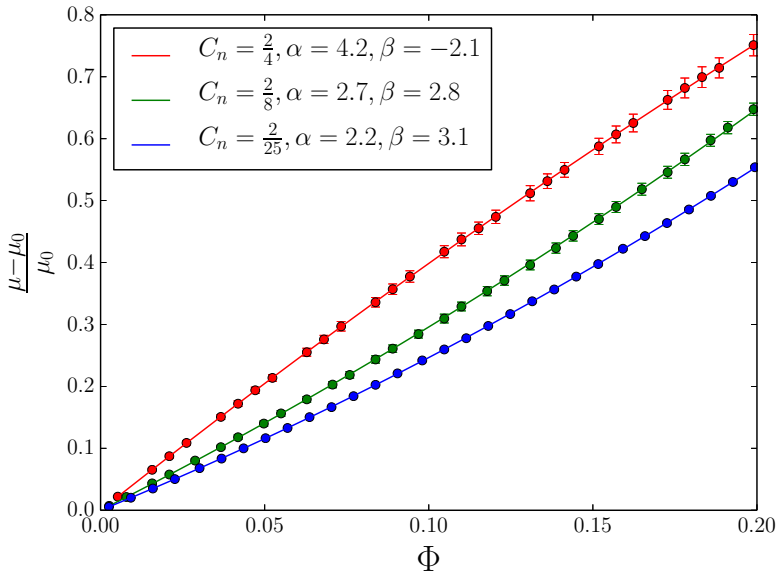


Figure 8.13: Reduced viscosity of a suspension for different volume fraction and different confinement. The points are the mean of 200 different configuration viscosities. The error bar are the standard deviation of the set of viscosities. The lines are the fits of these curves with a second order polynomial as given in equation (8.10). The coefficients obtained from the fit are given in the legend.

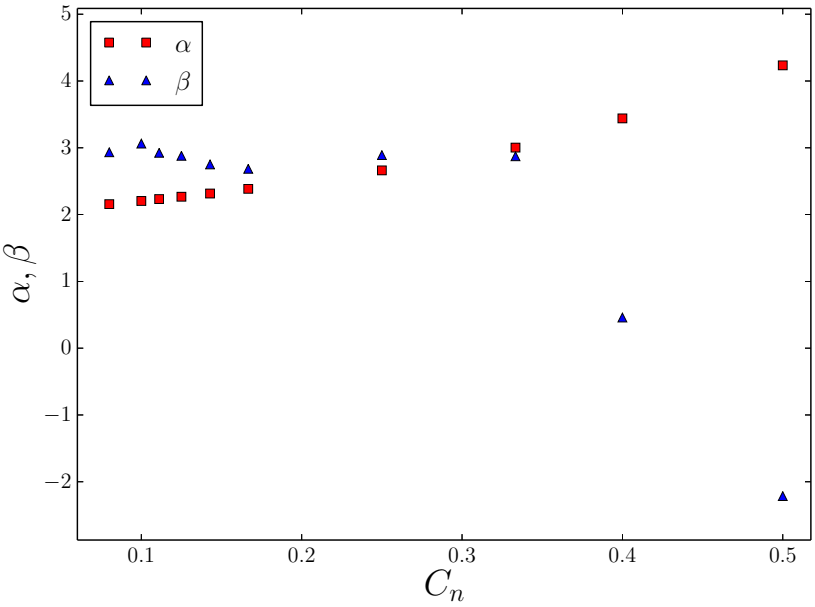


Figure 8.14: The coefficients of the fit of the viscosity curves as the one given in figure 8.13 for different confinement. The values of the α coefficient are represented as red squares whereas the β values are blue triangles. One can see that the α coefficient converges toward the Einstein's value for smaller confinement. The β factor decreases for stronger confinement and even becomes negative.

Chapter 9

Conclusion and perspectives

Français

Dans cette thèse, nous avons présenté un cadre méthodologique et numérique de simulation d'objets rigides et déformables sous écoulement. Notre approche est totalement Eulerienne et est basée sur la méthode de *level set*. Nous avons ensuite utilisé ce cadre pour explorer des problèmes ouverts de mécanique des fluides.

Première Partie

La première partie de cette thèse a été consacrée à l'exposé des méthodes mathématiques choisies et à leur validation et vérification. Nous avons commencé par exposer la méthode des *level set* et les choix que nous avons fait la concernant. En particulier, nous avons exposé les méthodes de stabilisations utilisées pour son transport, et les méthodes de réinitialisation implémentées. Pour chacune de ces méthodes, nous avons précisé si elles sont applicables pour des calculs parallèles, en deux comme en trois dimensions, et si un traitement spécial était nécessaire pour qu'elle soit aussi valable avec une discrétisation spatiale d'ordre élevée. Nous avons ensuite vérifié ces méthodes sur des cas tests.

Nous avons ensuite décrit notre stratégie pour la simulation d'un système multi-fluides, qui est la brique de base de la simulation d'une suspension de globules rouges. Le système classique bi-fluides a été décrit et vérifié sur un cas test. Puis nous avons présenté notre méthode pour passer d'un système bi-fluides à un système contenant un nombre arbitraire de fluides. Nous avons montré un exemple d'un tel système avec la simulations de la montée de huit bulles de fluides différents dans un neuvième fluide.

Nous avons aussi utilisé cette méthode de calcul pour montrer que les oscillations d'une bulle autour de sa forme d'équilibre ne dépendent pas du confinement auquel elle est soumise. Ce résultat a aussi une valeur de validation puisque nous sommes capable de retrouver la fréquence d'oscillations théorique d'une bulle avec une bonne précision.

Nous avons ensuite montré que notre méthode de simulation à deux fluides pouvait être utilisée pour simuler l'écoulement d'objets rigides dans un fluide. Nous avons fait le lien entre cette méthode de simulation et des méthodes déjà existantes dans la littérature. Nous avons enfin discuté de l'intérêt et des inconvénients d'utiliser une méthode de *level set* pour ce genre de simulation.

La fin de notre première partie est consacrée à la simulation de vésicules. Nous avons présenté deux méthodes différentes pour calculer la force de courbure appliquée aux vésicules qui dépend de la dérivée seconde de la courbure de la membrane. Les avantages et inconvénients de ces méthodes ont aussi été discutés.

Nous avons aussi testé deux méthodes tirées de la littérature pour appliquer la contrainte d'inextensibilité de la membrane des vésicules. Une méthode basée sur la pénalisation de l'étirement de la membrane par l'intermédiaire d'une force élastique, l'autre utilise un multiplicateur de Lagrange forçant la contrainte voulue. Nous avons ensuite montré une validation de ces méthodes sur des comportements connus de vésicules et avons conclus que la méthode utilisant le multiplicateur de Lagrange est bien plus stable que la méthode

de pénalisation. Bien qu'elle nécessite un développement numérique plus complexe, nous utiliserons la méthode du multiplicateur de Lagrange dans le futur pour nos simulations.

Futurs développements méthodologiques

Dans le futur, certains aspects méthodologiques pourraient être explorés en prenant comme base le travail effectué durant cette thèse. Il serait par exemple intéressant de développer un cadre de méthodes de stabilisation à l'ordre élevé. Une telle adaptation a déjà été faite pour la stabilisation des faces internes (CIP), néanmoins elle reste inconnue pour la stabilisation des éléments (SUPG, SGS, GLS).

Sauf indication contraire, toutes les méthodes que nous avons présenté sont utilisable en deux et trois dimensions. La librairie Feel++ a été conçue de telle sorte qu'un code soit aussi valable en deux et trois dimensions grâce à la méta-programmation. La plupart des tests présentés dans cette thèse sont donc déjà prêts à fonctionner en 3 dimensions. Néanmoins, les simulations 3D sont peu nombreuses dans ce travail. Bien que les codes soient valable en 3D, la simulation d'objets 3D nécessite de passer à un grand nombre de degrés de liberté. Elle nécessite aussi la parallélisation des codes de calculs ainsi que le rapatriement des données depuis le cluster de calcul sur lequel elles ont été obtenues et enfin le traitement du grand nombre de données ainsi générées. Tous ces éléments nécessitant un certain temps de développement, ils ne sont pas tous opérationnels aujourd'hui mais le seront dans un futur très proche.

Enfin, durant cette thèse, seul les cas de suspensions diluées ou semi-diluées ont été traités. Dans le futur, une extension à des suspensions concentrées pourrait être envisagée. Il se poserait alors le problème de la gestion des contacts entre les vésicules.

En théorie, les forces hydrodynamiques dues au fluide se trouvant entre deux vésicules devrait éviter leur contact. Cet effet a été suffisant pour éviter le contact entre les vésicules pour toutes les concentrations que nous avons testé jusqu'à présent (jusqu'à une fraction volumique de 25%). En pratique, cela nécessite de toujours avoir un certain nombre de points de discrétisation entre deux membranes. Il est possible que pour des suspensions de vésicules très concentrées, la force hydrodynamique ne soit plus suffisante pour éviter le contact faute d'un nombre suffisant de points de discrétisation entre les vésicules. Dans ce cas, plusieurs stratégies pourraient être imaginées.
Une méthode de raffinement de maillage pourrait être utilisée pour s'assurer qu'il y ait toujours assez d'éléments entre deux membranes. Un pas de temps adaptatif pourrait alors être mis en place pour garder une cohérence entre les discrétisations spatiale et temporelle.
Une force de contact dépendant de la distance à l'interface la plus proche pourrait aussi être développée. Une telle force est assez simple à implémenter dans le cas d'interfaces étant suivies par des champs de *level set* différents.
Néanmoins, il n'est pas trivial de trouver l'expression d'une force de contact pour des interfaces suivies par le même champ de *level set*. Ces questions pourraient faire le sujet de futurs travaux.

Deuxième partie

Les développements numériques apportés durant cette thèse s'inscrivent dans le projet *interactions fluide structure* de la librairie Feel++ dédiée à la résolution d'équations aux dérivées partielles. Ainsi certains codes développés dans le cadre de cette thèse pourraient être utiles dans d'autres contextes. Nous avons donc essayé de séparer un maximum les solutions apportées aux différents problèmes de façon à pouvoir les réutiliser et les faire évoluer dans le futur. La programmation orientée objet ayant été en grande partie inventée dans ce but, différentes classes ont été crées et seront incluses dans la partie publique de Feel++ dans le futur.

Ainsi, le début de notre seconde partie a été consacré à la description des classes pouvant être intégrées à Feel++ dans le futur et utilisées dans différent contextes de simulation.

La première de ces classes, propose un cadre pour créer différent opérateurs de projections nécessitant la solution d'une équation aux dérivées partielles (projection L^2, H^1, etc ...). Elle propose aussi la possibilité de dériver une expression mathématiques de façon efficace.

La deuxième classe, propose de résoudre l'équation d'advection utilisée par la méthodes des *level set* dans deux contextes différents (transport de la fonction et réinitialisation). Cette classe inclue toutes les méthodes de stabilisation décrites en première partie.

La troisième classe présentée est la classe LevelSet , qui contient tous les développements numériques nécessaires à la mise en place d'une méthode de *level set*.

Enfin, la dernière classe est une généralisation de la précédente, elle propose de mutualiser certains éléments de la classe LevelSet pour proposer une classe multi levelset capable de gérer efficacement plusieurs champs de *level set* à la fois.

La fin de notre seconde partie est consacrée à la présentation de développements créés pour les applications liées à cette thèse et qui ne sont pas directement applicable à d'autres contextes.

Nous avons ainsi présenté les détails de la construction du multiplicateur de Lagrange associé à la contrainte d'inextensibilité de la membrane. Puis nous avons décrit l'interface Python que nous avons développé pour lancer un grand nombre de simulation assez facilement.

Troisième Partie

La dernière partie de cette thèse est consacrée au comportement de disques rigides dans des géométries complexes et à leur rhéologie. Les problèmes étudiés dans cette partie s'inscrivent dans la thématique d'étude de l'équipe *Dynamique des fluides complexes* au sein de laquelle a été préparée cette thèse.

Le premier problème auquel cette partie est dédiée est l'étude du comportement d'une suspension diluée de particules rigides lorsqu'elle atteint une bifurcation micro-fluidique. L'hypothèse de faible concentration de particule nous a conduit à ne considérer qu'une particule à la fois dans le système étudié. Nous avons exploré un grand nombre de trajectoires possibles pour les particules en fonction de divers paramètres (rayons, rapport de débits dans les branches ...). Enfin, en comparant nos résultats avec des expériences réalisées dans notre laboratoire, nous avons pu expliquer l'effet connu d'accroissement de

la concentration de particule dans la branche recevant le plus grand débit. De plus, nous avons mis à jour l'existence d'une force hydrodynamique inconnue jusqu'alors poussant les particules vers la branche de bas débit et entrant en concurrence avec l'effet précédemment mentionné.

Enfin, nous avons étendu cette méthode à la simulation d'une suspension de vésicules dans une bifurcation. Nous avons aussi montré qu'il était possible d'effectuer la simulation d'une suspension de deux types de vésicules différent, c'est à dire ayant des rapports de viscosité ou des forces de courbure différents.

Dans le futur, un grand nombre de questions peuvent encore être posées à propos de ces systèmes. Des simulations de bifurcations dans lesquels deux particules sont présentes ont été menées depuis notre travail [80]. Il serait maintenant intéressant de voir comment la concentration en particules dans la branche de départ influe sur la répartition dans les branches d'arrivée.

L'influence de la déformabilité des particules sur la répartition dans les différentes branches est aussi une suite logique à ce travail. Finalement, des simulations de suspensions dans lesquels quelques vésicules se comportent comme de longues particules quasi-rigide pourraient être menée et représenteraient une avancée dans la compréhension de maladies telles que la drépanocytose dans laquelle un certain nombre de globules rouges se rigidifient et gênent la circulation sanguine lors du passage dans de petits vaisseaux.

La fin de cette troisième partie a été consacrée à la rhéologie d'une suspension de disques rigides dans un écoulement de cisaillement confiné. De précédents travaux ont montrés que lorsqu'une suspension de particules rigides est suffisamment confinée, l'effet de l'interaction entre les particules sur la viscosité effective de la suspension diminue grandement. Ces résultats on été obtenus en trois dimensions. Un des objectifs de ce travail était de vérifier si cet effet était un effet purement 3D ou si il était aussi présent en deux dimensions. Pour cela nous nous sommes d'abord intéressés à la viscosité effective d'une suspension diluée confinée, ce qui nous a permis de mettre en évidence l'influence de la position d'une particule par rapport au bord du domaine cisaillé.

Puis, nous avons exploré la viscosité d'un système de deux particules dans un écoulement de cisaillement confiné. En combinant les deux différentes études, nous avons pu montrer que l'effet connus à 3D était aussi présent en 2D bien que les phénomènes soient quantitativement différents.

Ce genre d'étude pourrait aussi être généralisée à une suspension de vésicules. Ainsi, l'influence de la déformabilité sur la viscosité effective d'une vésicule proche d'une parois pourrait être étudiée. Il est connu que les vésicules ont une tendance à migrer vers le centre du canal dans lequel elles évoluent. Connaître la dépendance de la viscosité effective par rapport à la position des vésicules dans le canal pourrait alors faire un lien entre les phénomènes de migration et la viscosité effective.

English

In this thesis, we have presented a numerical and methodological framework for the simulation of rigid and deformable objects in flow. Our approach is totally Eulerian and is based on the *level set* method. We then used this framework to explore open problems in fluid mechanics.

Part I

The first part of this thesis was dedicated to the overview of the mathematical methods chosen and their validation and verification. We explained the level set method and the choices that we made for it. Indeed, we have explained the stabilization methods used for its transport, and the implemented reinitialization methods. For each of these methods, we made precise if they are directly applicable to parallel computation, in 2 or 3 dimensions, and, if a special treatment is needed for high order polynomial approximation. We then verified these methods on test cases.

We then described our strategy concerning the simulation of a multi-fluid flow, which is the building block for the simulation of a red blood cell suspension. The classical two-fluid system has been described and verified on a test case. After what, we presented our method to adapt the two-fluid system to a system with an arbitrary number of fluids. We have shown an example of such a system with the simulation 8 rising bubbles of different fluids in a 9^{th} fluid.

We have used the two-fluid flow framework to show that the oscillations of a bubble around its equilibrium shape are not dependent on its confinement. This result has also a validation value since we are able to recover the theoretical oscillation frequency of the bubbles with a good accuracy.

Next, we have shown that our two-fluid flow simulation method can be used to simulate rigid object flows. We made a link between this simulation method and already existing ones. Finally, we discussed the advantages and drawbacks of using the level set method for this kind of simulations.

The end of our first part was dedicated to the simulation of vesicles. We have presented two different methods to compute the curvature force applied on the membrane of the vesicles. This force is complex since it requires the second derivative of the curvature of the membrane. We discussed the advantages and drawbacks of the two methods.

We also tested two methods taken from the literature to apply the inextensibility constraint on the membrane. One of them penalizes the stretching by using an elastic force, while the other one requires use of a Lagrange multiplier to enforce the constraint. We then showed a validation of these methods on known behaviors of vesicles and concluded that the method using the Lagrange multiplier is more stable than the penalty one. Although it necessitates a more complex numerical development, we will use the Lagrange multiplier method in the future for our vesicles simulations.

Future methodological developments

In the future, some methodological aspects could be explored taking this work as a basis. It would be interesting for example to develop a high order stabilization framework. Such adaptation has already been made for the internal faces stabilization (CIP), but it is still unknown for the elements stabilization (SGS, SUPG, GLS).

Unless we specifically mentioned it, all the methods that we presented are valid in 2 and 3 dimensions. The Feel++ library has been made so that the same code is valid in 2 and 3 dimensions thanks to the meta-programming concept. Most of the tests presented in this thesis are then ready to work in 3D. Nevertheless, 3D simulations are very rare in this work. Although the codes are valid in 3D, the simulation of 3D object necessitates a large number of degrees of freedom. It also necessitates making the code work in parallel as well as the downloading and treatment of the data from the cluster on which the simulations are made. All these features are time consuming to prepare and are not all operational to this date but they will be in a near future and the 3D versions of our simulations will be made.

Finally, during this thesis, only dilute or semi-dilute suspensions have been studied. In the future, an extension to more concentrated suspensions could be made. The problem of handling the contacts between the vesicles would arise.

Theoretically, the hydrodynamical forces due to the fluid being in between the vesicles should be sufficient to avoid their contact. This effect has been strong enough to avoid contact in every concentration we have tested (until a volume fraction of 25%). In practice, it necessitates always having several discretization points between two membranes. It is possible that for highly concentrated vesicles, the hydrodynamical force is not sufficient to avoid contact because of a lack of discretization points between two interfaces. In this case, several strategies could be used.
A mesh refinement method could be used to be sure that there are always enough elements between the membranes. An adaptive time step could also be set to keep coherent the spatial and temporal discretizations.
A contact force depending on the distance to the closest interface could also be developed. Such a force is simple to implement in the case of interfaces tracked by different level set fields.
Nevertheless, it is non-trivial to find an expression of this force for interfaces tracked by the same level set field. These questions could be the subject of future works.

Part II

The numerical developments made in this thesis take place in the Feel++ *fluid structure interaction* project of the library. Consequently, several codes developed for this thesis purpose could be used in other contexts. Thus, we tried to separate the solutions that we made to the different problems in order to re-use them and make them evolve in the future. The object oriented programming paradigm being created to this goal, different classes have been made and will be included in the public part of Feel++ in the future.

In this way, the beginning of our second part has been dedicated to the description of the classes which can be integrated in Feel++ in the future and used in other context.

The first of these classes proposes a framework to create different projection operator which necessitate to solve a partial differential equation (L^2 or H^1 projections ...).
The second one proposes solving the advection equation used by the level set method in two different contexts (transport and reinitialization). This class includes all the stabilization methods described in the first part.
The third class presented is the LevelSet class. It contains all the numerical developments necessary to setting up of a level set method.

Finally, the last class is a generalization of the previous one. It proposes to share several elements of the level set class to make a multi-level set class able to handle efficiently many level set fields at the same time.

The end of this second part, was dedicated to the developments made to create applications linked to this thesis by which are not directly applicable to other contexts.

We presented the details of the construction of the Lagrange multiplier associated to the inextensibility constraint. Then we described the Python interface that we developed to run easily a large number of simulations.

Part III

The last part of this thesis was dedicated to the behavior of rigid disks in complex geometries and their rheology. The problems studied in this part take place in the field of the studies of the *Dynamics of Complex Fluids* team in which this thesis has been done.

The first problem addressed in this part was the study of the behavior of a dilute suspension of particles when it reached a bifurcation. The low concentration hypothesis led us to only consider one particle in the system. We studied a large number of possible trajectories for the particles as a function of the different parameters (radius, flow rate ratio ...). Finally, by combining our results to experimental ones, we have explained the known effect of the increasing concentration of particles in the high flow rate branch. Moreover, we have discovered a hydrodynamic force pushing the particles toward the other branch balancing the first effect.

Finally, we have shown that our simulation framework was adapted to the simulation of a suspension of vesicles in a bifurcation. We also have shown that it is possible to use it for the simulation of a suspension having two types of different vesicles, that is to say having different viscosity ratio or curvature force intensities.

In the future, a large number of questions can be addressed. Simulations in which two particles are present have been made since our work [80]. It would be interesting to see how the concentration of the suspension in the inlet branch changes the repartition in the outlet branches.

The influence of the deformability of the particles on the concentration in the different branches could also be a relevant continuation of this work. Finally, simulations of suspension in which several vesicles behave as long rigid particles could be done. This would represent a step ahead in the understanding of diseases like drepanocitosis for which few red blood cells are getting rigid which hindrances the blood flow in circulatory system.

The end of this third part was dedicated to the rheology of a suspension of rigid disks in a confined shear flow. Previous works have shown that when a suspension of particles is sufficiently confined, the interaction effect between the particles is lowered. These results have been obtained in 3D. One of the objectives of this work was to check if this effect is purely 3D or if it is also present in 2D. To this goal, we first addressed the problem of the effective viscosity of a a dilute suspension, which allowed us to show the importance of the position of the particle relatively to the walls of the domain.

Then we explored the viscosity of a system of two particles in a confined shear flow. By combining the two studies, we were able to show that the known 3D effect was indeed present in 2D even if the results are quantitatively different.

This kind of study could also be generalized to a vesicle suspension. This way, the influence of the deformability on the effective viscosity of a vesicle near a wall could be studied. The vesicles having the property to lift toward the center of the channel, one understands the interest to study the influence of the distance to the wall on the viscosity.

Appendices

A Stokes variational formulation

Let us prove that:

$$\int_\Omega -(\boldsymbol{\nabla}\cdot\boldsymbol{\sigma})\cdot\boldsymbol{v} = \int_\Omega 2\mu\boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{D}(\boldsymbol{v}) - \int_\Omega p\nabla\cdot\boldsymbol{v} - \int_{\partial\Omega}(\boldsymbol{\sigma n})\cdot\boldsymbol{v} \tag{1}$$

First of all, let us integrate by part the first term and develop $\boldsymbol{\sigma}$ following the Newtonian fluid law:

$$\begin{aligned}
&\int_\Omega -(\boldsymbol{\nabla}\cdot\boldsymbol{\sigma})\cdot\boldsymbol{v}\\
&= \int_\Omega \boldsymbol{\sigma} : \boldsymbol{\nabla} v - \int_{\partial\Omega}(\boldsymbol{\sigma n})\cdot\boldsymbol{v}\\
&= \int_\Omega 2\mu\boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{\nabla} v - \int_\Omega p\nabla\cdot v - \int_{\partial\Omega}(\boldsymbol{\sigma n})\cdot\boldsymbol{v}
\end{aligned}$$

Now let us remark that $\boldsymbol{D}(\boldsymbol{u})$ is a symmetrical, thus when taking the contracted product with $\boldsymbol{\nabla v}$, the same results can be obtained if we invert the anti-symmetrical part of $\boldsymbol{\nabla v}$ (i.e taking ${}^t(\boldsymbol{\nabla v})$, we have then:

$$\boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{\nabla v} = \boldsymbol{D}(\boldsymbol{u}) :{}^t(\boldsymbol{\nabla v})$$

Thus we can write:

$$\begin{aligned}
&\boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{\nabla v}\\
&= \frac{\boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{\nabla v}}{2} + \frac{\boldsymbol{D}(\boldsymbol{u}) :{}^t(\boldsymbol{\nabla v})}{2}\\
&= \frac{\boldsymbol{D}(\boldsymbol{u}) : 2\boldsymbol{D}(\boldsymbol{v})}{2}\\
&= \boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{D}(\boldsymbol{v})
\end{aligned}$$

Thus, by replacing this expression we get:

$$\int_\Omega -\boldsymbol{\nabla}\cdot\boldsymbol{\sigma}\cdot\boldsymbol{v} = \int_\Omega 2\mu\boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{D}(\boldsymbol{v}) - \int_\Omega p\nabla\cdot\boldsymbol{v} - \int_{\partial\Omega}(\boldsymbol{\sigma n})\cdot\boldsymbol{v}$$

B Finding the axes of an ellipse

We will show in this appendix how to find easily the minor and major axes (a, b) of an ellipse for a given reduced area (α) and a fixed area A. The problem could be modified to find these parameters for a fixed perimeter P. Let us recall that the area of an ellipse is given by:

$$A = \pi\ a\ b.$$

The perimeter of an ellipse does not have a simple exact analytic expression but a good approximation is given by the Srinivasa Ramanujan formula:

$$P \approx \pi \left(3\ (a+b) - \sqrt{(3\ a+b)(a+3\ b)}\right).$$

We recall that the reduced area is given by the ratio between the area of the ellipse and the area of a circle having the same perimeter:

$$\alpha = \frac{4\pi A}{P^2}.$$

The problem is to find (a, b) for a given $A = A_0$ and $\alpha = \alpha_0$. Thus it can be re-written as, for a given (A_0, α_0), find (a, b) so that:

$$\pi\ a\ b - A_0 = 0$$
$$\pi \left(3\ (a+b) - \sqrt{(3\ a+b)(a+3\ b)}\right) - \sqrt{\frac{4\ \pi\ A_0}{\alpha_0}} = 0.$$

The problem is of the form $\boldsymbol{F}(\boldsymbol{U}) = 0$, with $\boldsymbol{U} = (a, b)$. It can be solved by the Newton Krylov non linear solver of SCIPY [51] that we use as a black box. We give as initial guess of the solution the radius of a circle having the area A_0 plus and minus an epsilon: $\boldsymbol{U}_0 = \left(\sqrt{\frac{A_0}{\pi}} + \varepsilon, \sqrt{\frac{A_0}{\pi}} - \varepsilon\right)$. With this guess, the non linear solver always converges in the range of α_0 and A_0 we are looking at. An example of such code, searching for the parameters (a, b) for a list of different α_0 is given in listing 1.

Listing 1: Finding minor and major axes of an ellipse given the area and reduced area

```
from pylab import *
from scipy.optimize import newton_krylov as nk

# define equations on perimeter P(a,b), area A(a,b) and the residual F(a,b)
P = lambda a,b : pi * (3*(a+b)-sqrt((3*a+b)*(a+3*b))) - sqrt(4*pi*A0/alpha)
A = lambda a,b : pi * a * b - A0
F = lambda x : ( P(x[0], x[1]), A(x[0], x[1]) )

# define the fixed values of A0 and a list of different alpha0
A0 = 1.
alpha_list = 1., 0.95, 0.9, 0.85, 0.8, 0.75, 0.6, 0.5, 0.4

# initial guess
X0 = sqrt(A0/pi)+0.2, sqrt(A0/pi)-0.2

for alpha in alpha_list :
    a, b = nk(F, X0)
    print "alpha = %f, a=%f, b=%f"%(alpha, a, b)
```

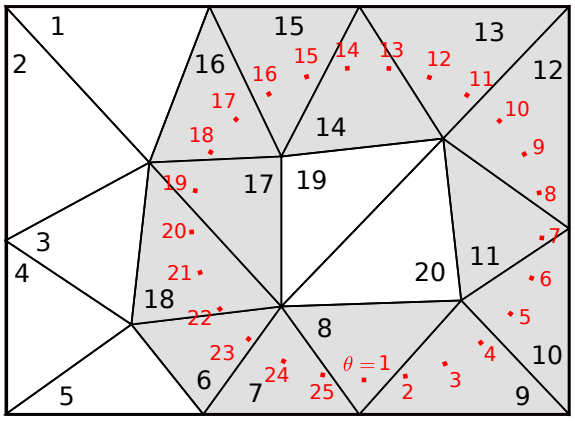
C Create a distance function to a parametrized curve

The level set field ϕ has to be initialized to a value on all the domain: ϕ_0. This value has to be a signed distance function or a function close to that. Indeed, if the reinitialization method used for the application is the reinitialization by solving a Hamilton Jacobi equation as presented in section 1.3.3, the method converges only if the initial function is not too far from the solution. In the case of a reinitialization by fast marching method as presented in section 1.3.4, ϕ_0 has to be a distance function in the first elements crossed by the iso 0 of ϕ_0. There are only few cases where the distance function to a curve is known analytically. For example, the distance function to a single line, or to a circle. A function close to a signed distance function to an ellipse is given by equation (1.32). Some shapes can be constructed by a combination of min and max function of the previous curves and thus, a distance function to these curves is known. It is the case for example of the slotted disk used in section 1.4 which is a combination of a signed distance to a circle and three signed distances to lines. For more complicated shapes, it is not possible to have a signed distance function. However, it might be useful to be able to describe more shapes than circles or lines and one might want to have directly the distance function to an ellipse and do not need to reinitialize it even before the first advection. Thus, we created a tool which provides a signed distance function to a parametrized curve. Parametrized curve is one of the better way to describe analytically a shape. We will present in this section, the method and the algorithm used to create such a distance function. Finally we will show how to use this method which is included in the level set framework and show some examples.

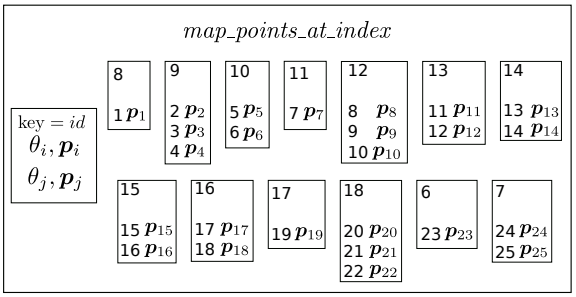
The method

The idea is to create a starter state for the fast marching method. That is to say, to have a distance function ϕ_0 to a parametrized curve only in the first elements crossed by the curve. Then, all the other elements are initialized by using the fast marching method presented in section 1.3.4. A brief explanation of the principle of the method is the following: we discretized the parametrized curve into a set of points. For each point, we locate the element to which it belongs, we store this information and we mark this element. Then, for all the degrees of freedom of all the marked elements (all the elements containing points), we compute the distance between the degree of freedom position and each point contained in this element. If the closest distance computed is smaller than the absolute value of ϕ_0 at this degree of freedom, ϕ_0 takes this value. The sign is determined thanks to the direction of the parametrized curve. Let us show into details how such a method is implemented.
To simplify, we will assume that we are in 2D even if the method can be extended to 3D. The curve is given by a parametrization $s(\theta) = (x(\theta), y(\theta))$. We assume that the user of the method provides the functions $x(\theta)$ and $y(\theta)$, the parametrized equations for the x and y positions of a point of the curve. The curve s has to be a closed curve. The user also provides the range in which is defined the parameter $\theta \in [\theta_{\min}, \theta_{\max}]$ and a discretization step $d\theta$ which will be used to discretize $s(\theta)$. Indeed, the first step of the method consists on creating a discretization of the curve $s(\theta)$. A set of points $\boldsymbol{p}_i$ is computed. It is a discretization of s defined as $\{\boldsymbol{p}_i = (x(\theta_i), y(\theta_i))\}$ for all $\theta_i \in [\theta_{\min}, \theta_{\max}]$ being multiple of $d\theta$. For each point $\boldsymbol{p}_i$, we locate the element on the mesh to which it belongs. The global

index of the element (called *id*) is a way to differentiate and access each element of the mesh. Thus, for each point $\boldsymbol{p}_i$, we search for the *id* of the element in which it is lying. We store all these information in a map for which the keys are the *ids* of the elements and the value are vectors containing all the $(\theta_i, \boldsymbol{p}_i)$ present in the element *id*. Let us call this map *map_points_at_index*. We also mark all the elements which contain at least one point in order to be able to iterate on it. A scheme of the mesh and the discretized curve is given in section 1(a) and the associated map is given in figure 1(b). Then, starts the most



(a) Scheme of the first step of the discretization of the parametrized curve. The indexes of the elements are represented in black. The parametrized curve is a circle and its discretization points are represented in red. The value of the parameter θ is given below the discretization points. We assume $d\theta = 1$. We represented in gray the marked elements, that is to say, the elements in which at least one discretization point is present.



(b) Representation of the map *map_points_at_index*. Each element of the map is composed of a key and a value. The key is the index (*id*) of the considered element. The value is a vector of pairs $(\theta_i, \boldsymbol{p}_i)$, the value of the parameter and the coordinates of the considered point. Thanks to this map, all the information concerning the discretization points which belong to an element are easily accessible.

Figure 1: First step of the creation of a distance function to a parametrized curve.

computationally costly part. For all the degrees of freedom of all the marked elements (all the elements crossed by the interface), we compute the distance between the position of the degree of freedom (let us call these points $\boldsymbol{p}_j^{\text{dof}}$) and all the points $\boldsymbol{p}_i$ contained in this element (they can be accessed easily via the map *map_points_at_index*. We search for the point $\boldsymbol{p}_i$ having the smallest distance to $\boldsymbol{p}_j^{\text{dof}}$, we call it $\boldsymbol{p}_{\min}$. If the distance to $\boldsymbol{p}_{\min}$ is smaller than the value of ϕ_0 at this degree of freedom (called ϕ_0^{dof}), then ϕ_0^{dof} takes this value. The sign of ϕ_0^{dof} is given by the vectorial product of the vector $\boldsymbol{v} = \boldsymbol{p}_{\min} - \boldsymbol{p}_j^{\text{dof}}$ and the normal to the curve $\boldsymbol{n}_{\text{s}}$ at the point $\boldsymbol{p}_{\min}$. That is to say, $S_{\text{gn}}(\phi_0^{\text{dof}}) = S_{\text{gn}}(\boldsymbol{v} \times \boldsymbol{n}_{\text{s}})$ The normal to the curve at the point $\boldsymbol{p}_{\min}$ is given by the difference between $\boldsymbol{p}_{\min}$ and the position of the next point on the curve. In other words, if $\boldsymbol{p}_{\min}$ corresponds to a value of the parameter $\theta = \theta_{\boldsymbol{p}_{\min}}$, that is to say $\boldsymbol{p}_{\min} = \boldsymbol{p}(\theta_{\boldsymbol{p}_{\min}})$ then the vector $\boldsymbol{n}_{\text{s}}$ is given by: $\boldsymbol{n}_{\text{s}} = \boldsymbol{p}(\theta_{\boldsymbol{p}_{\min}} + d\theta) - \boldsymbol{p}(\theta_{\boldsymbol{p}_{\min}})$. An example of these vectors is given in figure 2 for a given element. This whole algorithm gives a good approximation of a distance function to the parametrized curve s on the elements crossed by it. Then this function can be given to the fast marching algorithm which makes a distance function on all the domain from it. The method as we presented it is valid also at high order. But in the case of an high order approximation, making a loop on all the degrees of freedom of all the elements crossed by the curve can be computationally costly. We also recall that even at high order, the fast

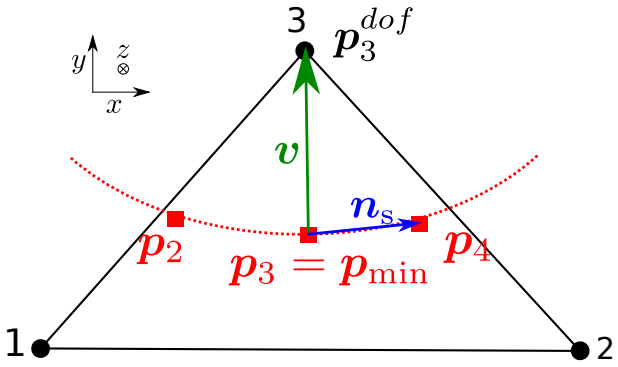


Figure 2: Representation of the second part of the algorithm. One element in represented and the considered degree of freedom is the number 3. Its coordinates are given by $\boldsymbol{p}_3^{\text{dof}}$. The curve is represented in red dotted line, and its discretization points in this element are represented as the three red squares. In this scheme, one has $|\boldsymbol{p}_3^{\text{dof}} - \boldsymbol{p}_3| < |\boldsymbol{p}_3^{\text{dof}} - \boldsymbol{p}_4| < |\boldsymbol{p}_3^{\text{dof}} - \boldsymbol{p}_2|$. Thus $\boldsymbol{p}_3^{\text{dof}} = \boldsymbol{p}_{\text{min}}$. The approximation of the normal $\boldsymbol{n}_{\text{s}} = \boldsymbol{p}_4 - \boldsymbol{p}_3$ is represented in blue. The vector $\boldsymbol{v} = \boldsymbol{p}_3^{\text{dof}} - \boldsymbol{p}_{\text{min}}$ is represented in green. One sees that the vector $\boldsymbol{v} \times \boldsymbol{n}_{\text{s}}$ is pointing toward the z positive.

marching method requires the field being reinitialized to be in a $\mathbb{P}^1$ space as explained in section 1.3.4. In practice, it is better to use the algorithm 7 directly in the same $\mathbb{P}^1$ space than the one used for the fast marching. It increases a bit the number of elements crossed by the curve, but decreases the number of degree of freedom to iterate in each element. It is usually benefit in term of computational cost to do it in this order.

Example of use

The previous algorithm has been embedded in the LevelSet class. To use it, one simply has to call the method `makeDistFieldFromParametrizedCurve` of the LevelSet class. We will show a simple example of use of this method. Let us say that we want a distance function to an [1]epitrochoid. The shape of the latter can be defined thanks to the following parametrization :

$$x(\theta) = (1+a)\cos(a\,\theta) - a\,b\cos((1+a)\,\theta) + x_0$$
$$y(\theta) = (1+a)\sin(a\,\theta) - a\,b\sin((1+a)\,\theta) + y_0$$

where θ is the parameter, a and b are parameters used to change the shape and the number of branches of the epitrochoid, finally x_0 and y_0 define the center of the shape. If we have defined a mesh and a level set object, we simply need to define the previous functions and start the algorithm. The function $x(\theta)$ and $y(\theta)$ are given as argument to `makeDistFieldFromParametrizedCurve`. Their types are [2]`std::function` which is a feature present in the standard C++11. The functions have to take one double argument as entry (θ) and returns one double. A very convenient way to define these functions is to use another C++11 feature, the [3]lambda functions which allow to create a function

[1] http://en.wikipedia.org/wiki/Epitrochoid
[2] http://en.cppreference.com/w/cpp/utility/functional/function
[3] http://en.cppreference.com/w/cpp/language/lambda

Algorithm 7 Algorithm making a distance function to a parametrized curve in the elements crossed by the curve.

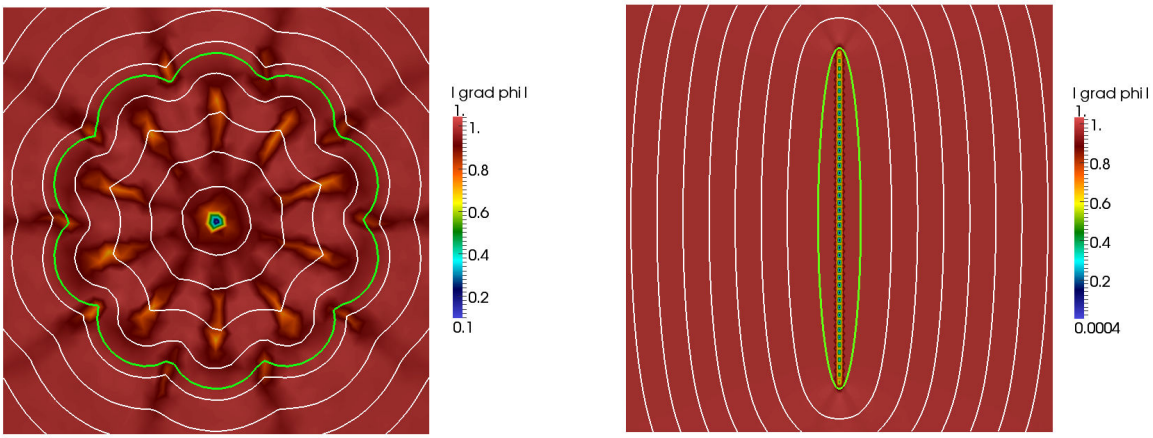
Require: $\theta_{\min}, \theta_{\max}, d\theta, x(\theta), y(\theta)$ given
Require: $\phi_0(\boldsymbol{x}) =$ HugeValue
  for $\theta_i = \theta_{\min}$ to $\theta_{\max}$ **do**
    id= **index of element containing point**$(\boldsymbol{p}_i = x(\theta_i), y(\theta_i))$
    if *map_points_at_index* contains key id **then**
      add $\boldsymbol{p}_i$ to *map_points_at_index* at key id
    else
      create key id in *map_points_at_index* and add the value $\boldsymbol{p}_i$
    end if
    create key θ_i in *map_point_param* with the value $\boldsymbol{p}_i$
    mark the element id
  end for
  for id_i in (ids of all marked elements) **do**
    for dof_j in **dof**(id_i) **do**
      $\boldsymbol{p}_j^{\text{dof}} =$ **position of** dof_j
      minDistToDof = HugeValue
      for $\boldsymbol{p}_k$ in (all points in *map_points_at_index* at index id_i) **do**
        distToDof = **compute distance**$(\boldsymbol{p}_k, \boldsymbol{p}_j^{\text{dof}})$
        if distToDof < minDistToDof **then**
          minDistToDof = distToDof
          $\boldsymbol{p}_{\min} = \boldsymbol{p}_k$
        end if
      end for
      if minDistToDof $< |\phi_0^{\text{dof}}|$ **then**
        $\boldsymbol{n}_{\text{s}} =$*map_point_param*$(\theta + d\theta) - \boldsymbol{p}_{\min}$
        $\boldsymbol{v} = \boldsymbol{p}_j^{\text{dof}} - \boldsymbol{p}_{\min}$
        $\phi_0^{\text{dof}} = S_{\text{gn}}(\boldsymbol{v} \times \boldsymbol{n}_{\text{s}})$ minDistToDof
      end if
    end for
  end for

embedded in another function in a very simple way. The code 2 shows on the example of the epitrochoid how to define these functions and use them as initial level set field.

Listing 2: Example showing how to define the functions $x(\theta), y(\theta)$ and create a distance function from a parametrized curve using the algorithm present in LevelSet .

```
double a=0.1, b=0.8;
double x0=5, y0=5;
double t_min=0, t_max=20*pi, dt=0.0001;

auto x = [&](double t) { return (1+a) * cos(a*t) - a*b * cos( (1+a) * t ) + x0; };
auto y = [&](double t) { return (1+a) * sin(a*t) - a*b * sin( (1+a) * t ) + y0; };

auto phio = levelset->makeDistFieldFromParametrizedCurve(x, y, t_min, t_max, dt);
```

The resulting field is shown in figure 3(a). We also show in figure 3(b) the result of the parametrization of an ellipse having a relatively big eccentricity, for which our usual approximation equation (1.32) is far from a distance function. One can see that in both cases, the gradient magnitude is around 1 almost every where and that the shape is well defined.



(a) Distance function from a epitrochoid with 10 branches.

(b) Distance function from an ellipse with a relatively big eccentricity.

Figure 3: Distance function from parametrized curves. The iso 0 is represented in green, some iso lines are represented in white and the color represents the gradient magnitude $|\nabla \phi_0|$.

D Simulation of rigid disk by direct method

We present in this appendix a *toy code* which simulates the flow of a rigid disk in a 2D Stokes fluid by a direct method. The equations solved are the Stokes equations, the particle is seen as a *hole* in the mesh as shown in figure 4. There are no slip boundary

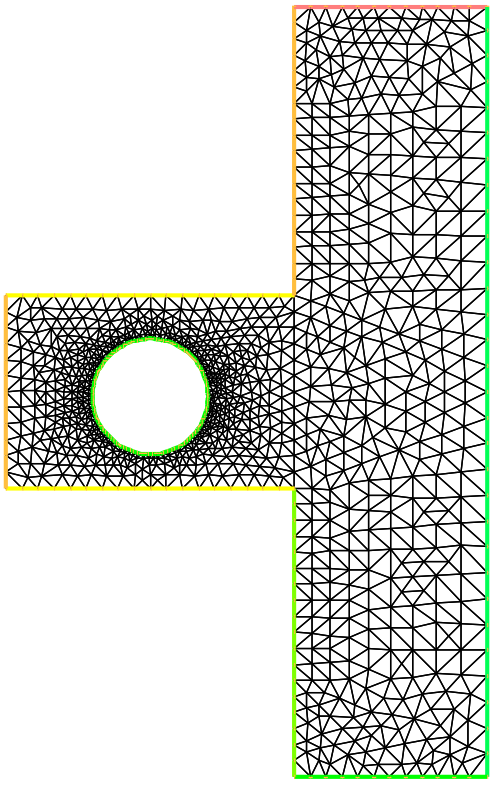


Figure 4: Mesh of a bifurcation in the case of a direct method for the simulation of a solid particle. One can see that the mesh is conform to the particle. The density of triangles around the particle can be increased to have a better accuracy when computing the hydrodynamical forces exerted on it.

conditions between the particle and the fluid and between the fluid and the walls. The geometry is the bifurcation studied in chapter 7. At each time step, the stokes equation is solved, the forces and torque are computed on the boundary of the particle. Then, the position and velocity of the particle are updated. Finally the domain is re-meshed according to the new position of the particle. Since the Stokes equations are independent of time, there is no need to project the solution of the previous step on the mesh of the current step. The Stokes equations are solved by finite element method using the variational formulation (2.14) where the null mean pressure is imposed by a penalty term. The finite element software chosen to solve this problem is FreeFem++ which is a Domain Specific Language dedicated for the solutions of PDE, that is to say that the software has its own language (with a syntax close to the one of C++). The use of FreeFem++ for this particular application is made because it integrates a mesh generator. Thus re-meshing the domain at each time step is very simple at the user point

of view. The update of the velocities and positions of the particle is made by a simple explicit Euler integration which is first order. This is one of the limitation of this method. A very small time step has to be taken to keep the system stable. And since the re meshing is time costly this method might not be adapted for long time simulations. For the case of the bifurcation problem, we finally chose a fictitious domain method. However, it is interesting to see how simple can be the implementation of such method if the PDE and meshing tools are provided.

Listing 3: Direct method for the simulation of a rigid disk with FreeFem++

```
// pressure penalty parameter
real epsilon=0.0000001;
// viscosity
real nu=666.67  ;
// radius of the particle
real R=1.2 ;
// mass of the particle
real M=10 ;
// intertial momentum of a disk
real I = M*0.5*R^2;

string FileName="Forces" ;

// time parameters
real dt=0.0003;
real finalTime = 3;

// mesh parameters
int n=3;
real m=2;

// geometrical parameters of the bifurcation
real H1=4;
real L1=6;
real H2=6;
real L2=4;

// flow rate imposed to the branches Q=Qh+Qb
real Q  = 150;
real Qratio=2./3.; // Qb/Qh
real Qh = Q/(1+Qratio);
real Qb = Qh*Qratio;

// Poiseuille flow velocities imposed with respect to the flow rates
func V  = -(6*Q /H1^3)*(y+H1/2)*(y-H1/2);     // V_inlet vx(y)
func Vh = -(6*Qh/L2^3)*(x-L1)*(x-(L1+L2));    // V_outlet top vy(x)
func Vb =  (6*Qb/L2^3)*(x-L1)*(x-(L1+L2));    // V_outlet bottom vy(x)

// create the borders of the bifurcation
border b1 (t=H1/2.,-H1/2.)                {x=0;      y=t;          label=1;};
border b2 (t=0,L1)                        {x=t;      y=-H1/2;      label=2;};
border b3 (t=-H1/2., -(H1/2.+H2))         {x=L1;     y=t;          label=3;};
border b4 (t=L1, L1+L2)                   {x=t;      y=-(H1/2+H2); label=4;};
border b5 (t=-(H1/2.+H2),(H1/2.+H2))      {x=L1+L2;  y=t;          label=5;};
border b6 (t=L1+L2, L1)                   {x=t;      y=H1/2+H2;    label=6;};
border b7 (t=H1/2.+H2, H1/2.)             {x=L1;     y=t;          label=7;};
border b8 (t=L1,0)                        {x=t;      y=H1/2;       label=8;};
```

```
ofstream ff(FileName);
ff<<"#time     x1     y1     teta      vx1       vy1       w1      fx       fy
C"<<endl;

// initial positions and velocities of the center of the particle
real x1=3, y1=-0.1, teta=0;
real w1=0;
real vx1=0;
real vy1=0;

func X=x-x1;
func Y=y-y1;

 for(real tt=0; tt<finalTime; tt+=dt)
   {

     // define the particle boundary
     border Part1 (t=0,2*pi){
       x=R*cos(t)+x1;
       y=R*sin(t)+y1;
       label=100;};

     // build the mesh
     mesh Th=buildmesh(b1(int(n * H1))
                       +b2(int(n * L1))
                       +b3(int(n * H2))
                       +b4(int(n * L2))
                       +b5(int(n * (H1+H2)))
                       +b6(int(n * L2))
                       +b7(int(n * H2))
                       +b8(int(n * L1))
                       +Part1(- int (5 * n * 2 * pi) ) );

     //-----------------------------------------------------
     // spaces for velocity and pressure
     fespace Esp(Th, P2);
     fespace Pre(Th, P1);
     Esp ux, uy, vx, vy, modU;
     Pre p, pp;
     //-----------------------------------------------------

     //-----------------------------------------------------
     // define the bilinear form and impose boundary conditions
     problem Stokes([ux,uy,p],[vx,vy,pp])=
       int2d(Th)( // bilinear form
                  // μ(∇u : ∇v)
         -nu*(dx(ux)*dx(vx)+dy(ux)*dy(vx)+dx(uy)*dx(vy)+dy(uy)*dy(vy))
         -dx(vx)*p  - dy(vy) * p        // p∇·v
         -dx(ux)*pp - dy(uy) * pp       // q∇·u
          +epsilon * p * pp )           // εpq
       // boundary conditions
       // no slip conditions on walls
       +on(2,3,5,7,8, ux=0, uy=0)
       // velocities imposed at outlet
       +on(1, ux=V, uy=0)
       +on(6, ux=0, uy=Vh)
       +on(4, ux=0, uy=Vb)
       // no slip condition on the particle
```

```
      +on(100,ux=vx1-w1*Y, uy=vy1+w1*X);
    //-----------------------------------------------------

    // solve Stokes problem
    Stokes;

    //-----------------------------------------------------
    // Compute the force exerted on the particle
    real F, fx,fy;
    fx=int1d(Th,100)( (-p+2*nu*dx(ux))*N.x + nu*(dx(uy)+dy(ux))*N.y)/M;
    fy=int1d(Th,100)( (-p+2*nu*dy(uy))*N.y + nu*(dx(uy)+dy(ux))*N.x)/M;
    //-----------------------------------------------------

    //-----------------------------------------------------
    // Compute the torque on the particle
    real C=-int1d(Th,100)( X*((-p+2*nu*dy(uy))*N.y + nu*(dx(uy)+dy(ux))*N.x)
                         - Y*((-p+2*nu*dx(ux))*N.x + nu*(dx(uy)+dy(ux))*N.y)
    //-----------------------------------------------------

    // ------------- save the results ---------
    ff<<tt<<"  "<<x1<<"  "<<y1<<"  "<<teta<<"  "<<vx1<<"  "<<vy1<<"
"<<w1<<"  "<<fx*M<<"  "<<fy*M<<"  "<<C*I<<endl;
    //-----------------------------------------------------

    //---- update the new positions and velocities by Euler step ---
    vx1 = vx1 + fx * dt;
    vy1 = vy1 + fy * dt;
    w1 = w1 + C * dt;
    x1 = x1 + vx1 * dt;
    y1 = y1 + vy1 * dt;
    teta = teta + w1 * dt;
    //-----------------------------------------------------

    modU=sqrt(ux*ux+uy*uy);
    plot(modU, fill=true,value=1, wait = false);

  }
```

E Bilinear forms of the different projection operators implemented in Projector

Here are the different expression of the bilinear forms implemented in the class PROJECTOR and the corresponding code. The FEEL++ expression are valid for $\boldsymbol{u}$ and $\boldsymbol{u}$ vectors or scalar.

- L^2 **projection** Bilinear form:

$$a_{L^2} = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{v}$$

FEEL++ expression:

```
a = integrate( elements( mesh ), trans( idt(u) ) * id(v) );
```

- H^1 **projection** Bilinear form:

$$a_{H^1} = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{v} + \int_\Omega \boldsymbol{\nabla u} : \boldsymbol{\nabla v}$$

FEEL++ expression:

```
a = integrate( elements( mesh ),
trans( idt(u) ) * id(v)
+ trace( gradt(u) * trans( grad(v) ) ) );
```

- $H^{\mathbf{div}}$ **projection** Bilinear form:

$$a_{H^{\text{div}}} = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{v} + \int_\Omega (\boldsymbol{\nabla} \cdot \boldsymbol{u})(\boldsymbol{\nabla} \cdot \boldsymbol{v})$$

FEEL++ expression:

```
a = integrate( elements( mesh ),
trans( idt(u) ) * id(v)
+ trans( divt(u) ) * div(v) );
```

- **Smooth projection** A special care has to be taken to impose the boundary conditions in a weak manner. The bilinear form reads:

$$a_{\text{smooth}} = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{v} + \int_\Omega \varepsilon \boldsymbol{\nabla u} : \boldsymbol{\nabla v} - \int_{\partial\Omega} \varepsilon (\boldsymbol{\nabla u}\, \boldsymbol{n}) \cdot \boldsymbol{v} - \int_{\partial\Omega} \varepsilon (\boldsymbol{\nabla v}\, \boldsymbol{n}) \cdot \boldsymbol{u} + \int_{\partial\Omega} \varepsilon \frac{\gamma}{h} \boldsymbol{u} \cdot \boldsymbol{v}$$

FEEL++ expression:

```
a = integrate( elements( mesh ),
trans( idt(u) ) * id(v)
+ epsilon * trace( gradt(u) * trans( grad(v) ) ) );

a += integrate( boundaryfaces( mesh ),
epsilon * ( - trans( id(v) ) * gradt(u) * N() )
+epsilon * ( - trans( idt(u)) * grad(v) * N() )
+epsilon * ( gamma * trans( idt(u) ) * id(v) / vf::hFace() ) );
```

- $H^{\mathbf{curl}}$ **projection** Bilinear form:

$$a_{H^{\text{curl}}} = \int_{\Omega} \boldsymbol{u} \cdot \boldsymbol{v} + \int_{\Omega} (\boldsymbol{\nabla} \times \boldsymbol{u}) \cdot (\boldsymbol{\nabla} \times \boldsymbol{v})$$

FEEL++ expression (in 2d):

```
a = integrate( elements( mesh ),
trans( idt(u) ) * id(v)
+ curlzt(u) * curlz(v) );
```

- **CIP projection** Bilinear form:

$$a_{\text{CIP}} = \int_{\Omega} \boldsymbol{u} \cdot \boldsymbol{v} + \int_{\Omega_I} \frac{\gamma}{h^2} [\![\boldsymbol{\nabla} \boldsymbol{u}]\!] [\![\boldsymbol{\nabla} \boldsymbol{v}]\!]$$

where Ω_I represents the internal faces of the domain Ω. FEEL++ expression:

```
a = integrate( elements( mesh ),
trans( idt(u) ) * id(v) );

a += integrate( internalfaces( mesh ),
gamma * hFace() * hFace()
* trans(jumpt(u)) * jump( grad(v) ) );
```

F A configuration file

In this appendix we present a configuration file. This file is used to give the static options of the simulation of a vesicle in a shear flow.

Listing 4: Configuration file for a simple simulation of vesicle in a shear flow. The options set in this files are the option usually not changed and do not depend on changing parameters. The other options are handled in a PYTHON interface.

```
# -- application options --
mup=10
k=10.
initial-state=0
export-plus=1
save-every=10

# -- levelset options --
[levelset]
time-discr-scheme=BDF2
advec-stab-method=GALS
enable-reinit=1
reinitevery=5
reinit-method=hj
hj-max-iter=5
hj-dtau=0.01
hj-tol=0.05

# -- fluid options --
[stokes]
cl_type=strong
flow_type=shear

# -- time options --
[bdf]
time-initial=0
time-final=40

# -- backend options --
# advection system
[ls-advec]
reuse-prec=0
ksp-atol=0.0000000001
pc-factor-mat-solver-package-type=mumps

# fluid system
[fluid]
pc-factor-mat-solver-package-type=mumps
reuse-prec=1
ksp-monitor=1

# projectors (L2, smooth)
[projectors]
ksp-monitor=1
reuse-prec=0
pc-factor-mat-solver-package-type=mumps
pc-type=lu
pc-gasm-overlap=1
```

G Indexes notations

In this work, we have choose to write all the equations using vectors and matrices. In other works, one can find the same equations written in component form. We will give in this appendix some equivalent forms between the vector and indexes notations. We assume the space has N_D dimensions. The position of a point is given by

$$\boldsymbol{x} = x_i,\ i = 1, ..., N_D.$$

The partial derivative of a scalar s in the direction x_i is given by, $\dfrac{\partial s}{\partial x_i}$ and will be written in our notations $\partial_{x_i} s$.

For each vector $\boldsymbol{v}$, one can write its components as

$$\boldsymbol{v} = v_i,\ i = 1, ..., N$$

with N the number of component of the vector. The same way, we can write the components of any matrix A as:

$$A = a_{ij},\ i = 1, ..., N \text{ and } j = 1, ..., M$$

with N the number of columns and M the number of lines of A.
The scalar product of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is given by:

$$\boldsymbol{u} \cdot \boldsymbol{v} = u_i v_i$$

with the Einstein summation convention, which stands that we sum over all the repeated indexes. In the following, we will always use the Einstein convention.
The double contraction product between a matrix A and a matrix B is a scalar given by:

$$A : B = a_{ij} b_{ij}.$$

The product of a matrix A and a vector $\boldsymbol{v}$ is a vector given by:

$$A\ \boldsymbol{v} = a_{ij} v_j,\ i = 1, ..., N.$$

The gradient of a scalar, let us say p is a vector given by:

$$\boldsymbol{\nabla} p = \partial_{x_i} p,\ i = 1, ..., N_D.$$

The gradient of a vector $\boldsymbol{v}$ is a matrix given by:

$$\boldsymbol{\nabla} \boldsymbol{v} = \partial_{x_i} v_j,\ i = 1, ..., N_D \text{ and } j = 1, ..., N.$$

The divergence of a vector $\boldsymbol{v}$ is a scalar given by:

$$\boldsymbol{\nabla} \cdot \boldsymbol{v} = \partial_{x_i} v_i$$

The laplacian of a vector $\boldsymbol{v}$ is a vector given by:

$$\boldsymbol{\Delta v} = \partial_{x_j}\partial_{x_j} v_i,\ i = 1, ..., N$$

The Navier-Stokes equation which we have written in the vectorial form:

$$\begin{aligned} \rho(\partial_t \boldsymbol{u} + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}) - \mu\boldsymbol{\Delta u} + \boldsymbol{\nabla} p &= \boldsymbol{f} \\ \nabla\cdot\boldsymbol{u} &= 0 \end{aligned}$$

$$\begin{aligned} \rho(\partial_t u_i + (u_j\partial_{x_j})u_i) - \mu\partial_{x_m}\partial_{x_m} u_i + \partial_{x_i} p &= f_i,\ i = 1, ..., N_D \\ \partial_{x_l} u_l &= 0. \end{aligned}$$

The surface divergence:

$$\nabla_{\mathrm{s}}\cdot\boldsymbol{u} = (\boldsymbol{I}_{\mathrm{d}} - \boldsymbol{n}\otimes\boldsymbol{n}) : \boldsymbol{\nabla u} = \nabla\cdot\boldsymbol{u} - (\boldsymbol{\nabla u}\ \boldsymbol{n})\cdot\boldsymbol{n} \tag{2}$$

$$\nabla_{\mathrm{s}}\cdot\boldsymbol{u} = (\delta_{ij} - n_j n_i)\partial_{x_j} u_i = \partial_{x_i} u_i - ((\partial_{x_i} u_j) n_j) n_i \tag{3}$$

with δ_{ij} the Kronecker symbol. We emphasize the fact that $(\boldsymbol{\nabla u}\ \boldsymbol{n})$ is the product of the matrix $\boldsymbol{\nabla u}$ by the vector $\boldsymbol{n}$, thus it is a vector. We can denote its components by $(\boldsymbol{\nabla u}\ \boldsymbol{n})_i$. Consequently $(\boldsymbol{\nabla u}\ \boldsymbol{n})_i\ n_i$ is a scalar which is expected.

The same way, we can re-write:

$$\boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{D}(\boldsymbol{v})$$

$$(\partial_{x_i} u_j + \partial_{x_j} u_i)(\partial_{x_i} v_j + \partial_{x_j} v_i).$$

H Derive the effective viscosity from energy dissipation

In this section, we will explicit the expression given in [65] where the viscosity of a suspension of rigid spheres or disks is given as a function of the energy dissipation $|\boldsymbol{\nabla u} +^t \boldsymbol{\nabla u}|^2$. In this appendix, all the notations are the same than in section 8.1. We will show that the energy dissipation in the fluid domain can be related to the forces applied on the top and bottom walls. Then we can use the definition of the effective viscosity given in equation (8.4). In this problem, we only consider a pure shear flow which is infinite in the x direction (or with periodic boundary conditions). Consequently, the boundaries $\partial\Omega_f$ are only the top and bottom moving walls.

The Stokes equation (2.6) without external forces can be re-written as:

$$-2\mu_0 \boldsymbol{\nabla} \cdot \boldsymbol{D}(\boldsymbol{u}) + \boldsymbol{\nabla} p = 0 \tag{4}$$

since $\nabla \cdot \boldsymbol{u} = 0$. We multiply the equation (4) by the velocity $\boldsymbol{u}$ and integrate over the fluid domain Ω_f. Integration by part gives:

$$-2\mu_0 \int_{\partial\Omega_f} [\boldsymbol{D}(\boldsymbol{u})\boldsymbol{n}] \cdot \boldsymbol{u} + 2\mu_0 \int_{\Omega_f} \boldsymbol{D}(\boldsymbol{u}) : \boldsymbol{\nabla u} + \int_{\partial\Omega_f} p(\boldsymbol{u}\cdot\boldsymbol{n}) - \int_{\Omega_f} p \nabla \cdot \boldsymbol{u} = 0$$

- Thanks to the symmetry of $2\boldsymbol{D}(\boldsymbol{u}) = (\boldsymbol{\nabla u} +^t \boldsymbol{\nabla u})$, the second term becomes:

$$\begin{aligned}
&\left(\boldsymbol{\nabla u} +^t \boldsymbol{\nabla u}\right) : \boldsymbol{\nabla u} \\
&= \frac{(\boldsymbol{\nabla u} +^t \boldsymbol{\nabla u}) : \boldsymbol{\nabla u} + (\boldsymbol{\nabla u} +^t \boldsymbol{\nabla u}) :^t \boldsymbol{\nabla u}}{2} \\
&= \frac{(\boldsymbol{\nabla u} +^t \boldsymbol{\nabla u}) : (\boldsymbol{\nabla u} +^t \boldsymbol{\nabla u})}{2} \\
&= \frac{|\boldsymbol{\nabla u} +^t \boldsymbol{\nabla u}|^2}{2} \\
&= 2\,|\boldsymbol{D}(\boldsymbol{u})|^2
\end{aligned}$$

- The third term can be re-written as $[(p\,\boldsymbol{I}_\mathrm{d})\boldsymbol{n}] \cdot \boldsymbol{u}$ so that we can factorize it with the fisrt term.
- The last term vanishes since $\nabla \cdot \boldsymbol{u} = 0$.

Factorization of the first and third term leads to:

$$2\mu_0 \int_{\Omega_f} |\boldsymbol{D}(\boldsymbol{u})|^2 - \int_{\partial\Omega_f} [(-p\,\boldsymbol{I}_\mathrm{d} + 2\mu_0 \boldsymbol{D}(\boldsymbol{u}))\boldsymbol{n}] \cdot \boldsymbol{u} = 0$$

$$2\mu_0 \int_{\Omega_f} |\boldsymbol{D}(\boldsymbol{u})|^2 - \int_{\partial\Omega_f} (\sigma\boldsymbol{n}) \cdot \boldsymbol{u} = 0$$

Finally, computing the last integral, which is done on the top and bottom boundaries gives:

$$2\mu_0 \int_{\Omega_f} |\boldsymbol{D}(\boldsymbol{u})|^2 = (f_t - f_b)\, U$$

The forces difference applied on the walls is related to the definition of the effective viscosity given in equation (8.4), it reads:

$$(f_t - f_b) = \frac{4\, L\, U\, \mu_{\text{eff}}}{l}$$

Replacing this term in the previous equation leads to the definition of the effective viscosity using the dissipation in the fluid domain:

$$\mu_{\text{eff}} = \frac{\mu_0\, l}{2LU^2} \int_{\Omega_f} |\boldsymbol{D}(\boldsymbol{u})|^2.$$

Bibliography

[1] D Andelman, T Kawakatsu, and K Kawasaki. Equilibrium shape of two-component unilamellar membranes and vesicles. *EPL (Europhysics Letters)*, 19(1):57, 1992.

[2] Robert E Apfel, Yuren Tian, Joseph Jankovsky, Tao Shi, X Chen, R Glynn Holt, Eugene Trinh, Arvid Croonquist, Kathyrn C Thornton, Albert Sacco, Jr, et al. Free oscillations and surfactant studies of superdeformed drops in microgravity. *Physical review letters*, 78(10):1912–1915, 1997.

[3] Satish Balay, Jed Brown, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2013. http://www.mcs.anl.gov/petsc.

[4] GK Batchelor and JT Green. The determination of the bulk stress in a suspension of spherical particles to order c2. *J. Fluid Mech*, 56(3):401–427, 1972.

[5] J. Beaucourt, F. Rioual, T. Seacuteon, T. Biben, and C. Misbah. Steady to unsteady dynamics of a vesicle in a flow. *Phys. Rev. E*, 69(1):011906–, January 2004.

[6] Silvia Bertoluzza, Mourad Ismail, and Bertrand Maury. Analysis of the fully discrete fat boundary method. *Numerische Mathematik*, 118(1):49–77, 2011.

[7] T. Biben and C. Misbah. Tumbling of vesicles under shear flow within an advected-field approach. *Phys. Rev. E*, 67(3):031908, Mar 2003.

[8] Thierry Biben, Klaus Kassner, and Chaouqi Misbah. Phase-field approach to three-dimensional vesicle dynamics. *Phys. Rev. E*, 72:041921, Oct 2005.

[9] R.P. Bonet Chaple. Numerical stabilization of vonvection-diffusion-reaction problems. Technical report, 2006.

[10] Alexander N Brooks and Thomas JR Hughes. 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):199–259, 1982.

[11] Cuc Bui, Vanessa Lleras, and Olivier Pantz. Dynamics of red blood cells in 2d. *ESAIM: Proceedings*, 28:182–194, 2009.

[12] Thi Thu Cuc Bui, P. Frey, and B. Maury. A coupling strategy based on anisotropic mesh adaptation for solving two-fluid flows. *International Journal for Numerical Methods in Fluids*, pages n/a–n/a, 2010.

[13] Erik Burman and Peter Hansbo. Edge stabilization for galerkin approximations of convection-diffusion-reaction problems. *Computer Methods in Applied Mechanics and Engineering*, 193(15-16):1437 – 1453, 2004.

[14] Erik Burman and Peter Hansbo. Edge stabilization for the generalized stokes problem: A continuous interior penalty method. *Computer Methods in Applied Mechanics and Engineering*, 195(19-22):2393–2410, April 2006.

[15] Céline Caldini Queiros, Vincent Chabannes, Mourad Ismail, Gonçalo Pena, Christophe Prud'Homme, Marcela Szopos, and Ranine Tarabay. Towards large-scale three-dimensional blood flow simulations in realistic geometries. pp. 17, Accepted in ESAIM: Proc, CEMRACS 2012, 2012.

[16] P.B. Canham. The minimum energy of bending as a possible explanation of the biconcave shape of the human red blood cell. *Journal of Theoretical Biology*, 26(1):61 – 81, 1970.

[17] Vincent Chabannes. *Vers la simulation des écoulements sanguins*. PhD thesis, Universite de Grenoble, 2013.

[18] Vincent Chabannes, Gonçalo Pena, and Christophe Prud'Homme. High-order fluid-structure interaction in 2D and 3D. Application to blood flow in arteries. *Journal of Computational and Applied Mathematics*, 246:1–9, July 2013. Fifth International Conference on Advanced COmputational Methods in ENgineering (ACOMEN 2011).

[19] S. Chandrasekhar. *Hydrodynamic and Hydromagnetic Stability*. Courier Dover Publications., 1981.

[20] Georges-Henri Cottet and Emmanuel Maitre. A level-set formulation of immersed boundary methods for fluid–structure interaction problems. *Comptes Rendus Mathematique*, 338(7):581 – 586, 2004.

[21] Georges-Henri Cottet and Emmanuel Maitre. A level set method for fluid-structure interactions with immersed surfaces. *Mathematical Models and Methods in Applied Sciences*, 2010.

[22] Y Davit and P Peyla. Intriguing viscosity effects in confined suspensions: A numerical study. *EPL (Europhysics Letters)*, 83(6):64001, 2008.

[23] KH De Haas, C Blom, D Van den Ende, MHG Duits, and J Mellema. Deformation of giant lipid bilayer vesicles in shear flow. *Physical Review E*, 56(6):7132, 1997.

[24] HJ Deuling and W Helfrich. The curvature elasticity of fluid membranes: a catalogue of vesicle shapes. *Journal de Physique*, 37(11):1335–1345, 1976.

[25] Jim Douglas and Todd Dupont. Interior penalty procedures for elliptic and parabolic galerkin methods. *Computing Methods in Applied Sciences*, 58:207–216, 1976.

[26] V. Doyeux, Y. Guyot, V. Chabannes, C. Prud'homme, and M. Ismail. Simulation of two-fluid flows using a finite element/level set method. application to bubbles and vesicle dynamics. *Journal of Computational and Applied Mathematics*, 246(0):251 – 259, 2013.

[27] V. Doyeux, T. Podgorski, S. Peponas, M. Ismail, and G. Coupier. Spheres in the vicinity of a bifurcation: elucidating the zweifach-fung effect. *Journal of Fluid Mechanics*, 674:359–388, 2011.

[28] Doyeux, Vincent, Chabannes, Vincent, Prud´homme, Christophe, and Ismail, Mourad. Simulation of vesicle using level set method solved by high order finite element. *ESAIM: Proc.*, 38:335–347, 2012.

[29] V. Ducrot and P. Frey. Anisotropic level set adaptation for accurate interface capturing. In RaoV. Garimella, editor, *Proceedings of the 17th International Meshing Roundtable*, pages 159–176. Springer Berlin Heidelberg, 2008.

[30] Albert Einstein. Eine neue bestimmung der moleküldimensionen. *Annalen der Physik*, 324(2):289–306, 1906.

[31] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, November 2002.

[32] Benoit Fabrèges. *Une méthode de prolongement régulier pour la simulation d'écoulements fluide / particules*. PhD thesis, Universite Paris XI, 2012.

[33] Alexander Farutin, Thierry Biben, and Chaouqi Misbah. Analytical progress in the theory of vesicles under linear flow. *Physical Review E*, 81(6):061904, 2010.

[34] Leopoldo P. Franca, Sergio L. Frey, and Thomas J.R. Hughes. Stabilized finite element methods: I. application to the advective-diffusive model. *Computer Methods in Applied Mechanics and Engineering*, 95(2):253–276, March 1992.

[35] Thomas Franke, Ronald HW Hoppe, Christopher Linsenmann, Lothar Schmid, Carina Willbold, and Achim Wixforth. Numerical simulation of the motion of red blood cells and vesicles in microfluidic flows. *Computing and visualization in science*, 14(4):167–180, 2011.

[36] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[37] Giovanni Ghigliotti, Thierry Biben, and Chaouqi Misbah. Rheology of a dilute two-dimensional suspension of vesicles. *Journal of Fluid Mechanics*, 653:489–518, 2010.

[38] Giovanni Ghigliotti, Abtin Rahimian, George Biros, and Chaouqi Misbah. Vesicle migration and spatial organization driven by flow line curvature. *Physical Review Letters*, 106(2):028101, 2011.

[39] R Glowinski, TW Pan, TI Hesla, DD Joseph, and J Periaux. A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow. *Journal of Computational Physics*, 169(2):363–426, 2001.

[40] Roland Glowinski, T-W Pan, Todd I Hesla, and Daniel D Joseph. A distributed lagrange multiplier/fictitious domain method for particulate flows. *International Journal of Multiphase Flow*, 25(5):755–794, 1999.

[41] J.L. Guermond, P. Minev, and Jie Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195(44-47):6011–6045, September 2006.

[42] G. Hauke. A simple subgrid scale stabilized method for the advection-diffusion-reaction equation. *Computer Methods in Applied Mechanics and Engineering*, 191(27-28):2925–2947, April 2002.

[43] W Helfrich. Elastic properties of lipid bilayers: theory and possible experiments. *Z Naturforsch C*, 28(11):693–703–, 1973.

[44] Thomas JR Hughes, Leopoldo P Franca, and Gregory M Hulbert. A new finite element formulation for computational fluid dynamics: Viii. the galerkin/least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, 73(2):173–189, 1989.

[45] S. Hysing, S. Turek, D. Kuzmin, N. Parolini, E. Burman, S. Ganesan, and L. Tobiska. Quantitative benchmark computations of two-dimensional bubble dynamics. *International Journal for Numerical Methods in Fluids*, 60(11):1259–1288, 2009.

[46] Mourad Ismail. *Méthode de la frontière élargie pour la résolution de problèmes elliptiques dans des domaines perforés. Application aux écoulements fluides tridimensionnels*. PhD thesis, Université Paris VI, 2004.

[47] Mourad Ismail and Aline Lefebvre-Lepot. A" necklace" model for vesicles simulations in 2d. *arXiv preprint arXiv:1202.3034*, 2012.

[48] Levan Jibuti. *Locomotion et écoulements dans les fluides complexes confinés*. PhD thesis, Université de Grenoble, 2011.

[49] João Janela, Aline Lefebvre, and Bertrand Maury. A penalty method for the simulation of fluid - rigid body interaction. *ESAIM: Proc.*, 14:115–123, 2005.

[50] AA Johnson and TE Tezduyar. Simulation of multiple spheres falling in a liquid-filled tube y. 1995.

[51] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[52] Vasiliy Kantsler and Victor Steinberg. Orientation and dynamics of a vesicle in tank-treading motion in shear flow. *Phys. Rev. Lett.*, 95:258101, Dec 2005.

[53] B. Kaoui, J. Harting, and C. Misbah. Two-dimensional vesicle dynamics under shear flow: Effect of confinement. *pre*, 83(6):066319, june 2011.

[54] B. Kaoui, G. H. Ristow, I. Cantat, C. Misbah, and W. Zimmermann. Lateral migration of a two-dimensional vesicle in unbounded Poiseuille flow. *pre*, 77(2):021903–+, February 2008.

[55] B. Kaoui, N. Tahiri, T. Biben, H. Ez-Zahraouy, A. Benyoussef, G. Biros, and C. Misbah. Complexity of vesicle microcirculation. *Phys. Rev. E*, 84:041906, Oct 2011.

[56] Badr Kaoui. *Modélisation de vésicules en géométrie étendue et dans des systèmes micro-fluidiques*. PhD thesis, Université Joseph-Fourier - Grenoble I, LSP - Laboratoire de Spectrométrie Physique, 2009.

[57] Badr Kaoui, George Biros, and Chaouqi Misbah. Why do red blood cells have asymmetric shapes even in a symmetric flow? *Phys. Rev. Lett.*, 103:188101, Oct 2009.

[58] Badr Kaoui, Alexander Farutin, and Chaouqi Misbah. Vesicles under simple shear flow: Elucidating the role of relevant control parameters. *Phys. Rev. E*, 80(6):061905, Dec 2009.

[59] C.E. Kees, I. Akkerman, M.W. Farthing, and Y. Bazilevs. A conservative level set method suitable for variable-order approximations and unstructured meshes. *Journal of Computational Physics*, 230(12):4536 – 4558, 2011.

[60] Yongsam Kim and Ming-Chih Lai. Simulating the dynamics of inextensible vesicles by the penalty immersed boundary method. *Journal of Computational Physics*, 229(12):4840–4853, 2010.

[61] M. Kraus, W. Wintz, U. Seifert, and R. Lipowsky. Fluid Vesicles in Shear Flow. *Physical Review Letters*, 77:3685–3688, October 1996.

[62] Aymen Laadhari. *Modelisation numérique de la dynamique des globules rouges par la méthode des fonctions de niveau*. PhD thesis, Universite de Grenoble, 2011.

[63] Aymen Laadhari, Pierre Saramito, and Chaouqi Misbah. Computing the dynamics of biomembranes by combining conservative level set and adaptive finite element methods. CNRS, June 2011.

[64] H LAMB. *Hydrodynamics.* Cambridge University Press., 1932.

[65] Aline Lefebvre and Bertrand Maury. Apparent viscosity of a mixture of a newtonian fluid and interacting particles. *Comptes Rendus Mécanique*, 333(12):923 – 933, 2005.

[66] Aline Lefebvre-Lepot. *Modélisation numérique d'écoulements fluide/particules*. PhD thesis, Université Paris XI, 2007.

[67] Randall J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhauser, 1992.

[68] Xu-Dong Liu, Stanley Osher, and Tony Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1):200 – 212, 1994.

[69] M. Mader, V. Vitkova, M. Abkarian, A. Viallat, and T. Podgorski. Dynamics of viscous vesicles in shear flow. *The European Physical Journal E: Soft Matter and Biological Physics*, 19(4):389–397, 2006-04-01.

[70] E. Maitre, C. Misbah, P. Peyla, and A. Raoult. Comparison between advected-field and level-set methods in the study of vesicle dynamics. *Physica D: Nonlinear Phenomena*, 241(13):1146 – 1157, 2012.

[71] Emmanuel Maitre, Thomas Milcent, Georges-Henri Cottet, Annie Raoult, and Yves Usson. Applications of level set methods in computational biophysics. *Mathematical and Computer Modelling*, 49(11-12):2161–2169, June 2009.

[72] Ravikanth Malladi, JamesA. Sethian, and BabaC. Vemuri. A fast level set based algorithm for topology-independent shape modeling. *Journal of Mathematical Imaging and Vision*, 6(2-3):269–289, 1996.

[73] B. Maury. Numerical analysis of a finite element/volume penalty method. *SIAM Journal on Numerical Analysis*, 47(2):1126–1148, 2009.

[74] Bertrand Maury. A fat boundary method for the poisson problem in a domain with holes. *Journal of Scientific Computing*, 16(3):319–339, 2001.

[75] S. Mendez, E. Gibaud, and F. Nicoud. An unstructured solver for simulations of deformable particles in flows at arbitrary reynolds numbers. *Journal of Computational Physics*, 2013.

[76] Thomas MILCENT. *Une approche eulérienne du couplage fluide-structure, analyse mathématique et applications en biomécanique.* PhD thesis, Université Joseph Fourier, 2009.

[77] Chaouqi Misbah. Vacillating breathing and tumbling of vesicles under shear flow. *Phys. Rev. Lett.*, 96(2):028104, Jan 2006.

[78] Jean-Christophe Nave, Rodolfo Ruben Rosales, and Benjamin Seibold. A gradient-augmented level set method with an optimally local, coherent advection scheme. *Journal of Computational Physics*, 229(10):3802 – 3827, 2010.

[79] Hiroshi Noguchi and Gerhard Gompper. Shape transitions of fluid vesicles and red blood cells in capillary flows. *Proceedings of the National Academy of Sciences of the United States of America*, 102(40):14159–14164, 2005.

[80] Santtu TT Ollila, Colin Denniston, and Tapio Ala-Nissila. One-and two-particle dynamics in microfluidic t-junctions. *Physical Review E*, 87(5):050302, 2013.

[81] Nicola Parolini. *Computational fluid dynamics for naval engineering problems.* PhD thesis, Politecnico di Milano, 2004.

[82] Charles S Peskin. Numerical analysis of blood flow in the heart. *Journal of Computational Physics*, 25(3):220 – 252, 1977.

[83] Charles S Peskin. The immersed boundary method. *Acta numerica*, 11(0):479–517, 2002.

[84] Philippe Peyla and Claude Verdier. New confinement effects on the viscosity of suspensions. *EPL (Europhysics Letters)*, 94(4):44001, 2011.

[85] C Pozrikidis. Effect of membrane bending stiffness on the deformation of capsules in simple shear flow. *Journal of Fluid Mechanics*, 440:269–291, 2001.

[86] C. Prud'homme, V. Chabannes, and G. Pena. Feel++: Finite Element Embedded Language in C++. Free Software available at http://www.feelpp.org. Contributions from A. Samake, V. Doyeux, M. Ismail and S. Veys.

[87] Prud´homme, Christophe, Chabannes, Vincent, Doyeux, Vincent, Ismail, Mourad, Samake, Abdoulaye, and Pena, Goncalo. Feel++ : A computational framework for galerkin methods and advanced numerical methods. *ESAIM: Proc.*, 38:429–455, 2012.

[88] Salima Rafaï, Levan Jibuti, and Philippe Peyla. Effective viscosity of microswimmer suspensions. *Phys. Rev. Lett.*, 104:098102, Mar 2010.

[89] Codina Ramon. Comparison of some finite element methods for solving the diffusion-convection-reaction equation. *Computer Methods in Applied Mechanics and Engineering*, 156(1-4):185–210, April 1998.

[90] PA Raviart and V Girault. Finite element approximation of the navier-stokes equations. *Lecture notes in mathematics, Springer-Verlag*, 1979.

[91] Elisabeth Rouy and Agnès Tourin. A viscosity solutions approach to shape-from-shading. *SIAM Journal on Numerical Analysis*, 29(3):867–884, 1992.

[92] D. Salac and M. Miksis. A level set projection model of lipid vesicles in general flows. *Journal of Computational Physics*, 230(22):8192–8215, September 2011.

[93] Ashok S. Sangani, Andreas Acrivos, and Philippe Peyla. Roles of particle-wall and particle-particle interactions in highly confined suspensions of spherical particles being sheared at low reynolds numbers. *Physics of Fluids*, 23(8):–, 2011.

[94] Udo Seifert. Configurations of fluid membranes and vesicles. *Advances in Physics*, 46(1):13–137, 1997.

[95] Udo Seifert, Karin Berndl, and Reinhard Lipowsky. Shape transformations of vesicles: Phase diagram for spontaneous- curvature and bilayer-coupling models. *Phys. Rev. A*, 44(2):1182–, July 1991.

[96] Hassib Selmi, Lassaad Elasmi, Giovanni Ghigliotti, and Chaouqi Misbah. Boundary integral and fast multipole method for two dimensional vesicle sets in poiseuille flow. *Discrete and Continuous Dynamical Systems-Series B (DCDS-B)*, 15(4):1065–1076, 2011.

[97] J.A. Sethian. *Analysis of flame propagation*. PhD thesis, Lawrence Berkeley Lab., CA (USA), 1982.

[98] J.A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1996.

[99] Richard Skalak. Motion of a tank-treading ellipsoidal particle in a shear flow. *Journal of Fluid Mechanics*, 120:27–47, 1982.

[100] Benjamin Stamm. *Stabilization strategies for discontinuous Galerkin methods*. PhD thesis, Ecole Polytechnique Federale de Lausanne, 2008.

[101] Ronald Fedkiw Stanley Osher. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, S.S. Antman, J.E. Marsden, L. Sirovich.

[102] Jasjit S. Suri, Sameer Singh, Swamy Laxminarayan, Xiaolan Zeng, Kecheng Liu, and Laura Reden. Shape recovery algorithms using level sets in 2-d/3-d medical imagery: A state-of-the-art review. *IEEE Transactions on information technology in biomedicine*, 2001.

[103] Mark Sussman, Ann S. Almgren, John B. Bell, Phillip Colella, Louis H. Howell, and Michael L. Welcome. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics*, 148(1):81–124, January 1999.

[104] Mark Sussman, Peter Smereka, and Stanley Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114(1):146 – 159, 1994.

[105] Hajime Tanaka and Takeaki Araki. Simulation method of colloidal suspensions with hydrodynamic interactions: Fluid particle dynamics. *Physical review letters*, 85(6):1338–1341, 2000.

[106] Anna-Karin Tornberg and Björn Engquist. A finite element based level-set method for multiphase flow applications. *Computing and Visualization in Science*, 3:93–101, 2000. 10.1007/s007910050056.

[107] Ken-ichi Tsubota and Shigeo Wada. Effect of the natural state of an elastic cellular membrane on tank-treading and tumbling motions of a single red blood cell. *Physical Review E*, 81(1):011910, 2010.

[108] Shravan K. Veerapaneni, Denis Gueyffier, Denis Zorin, and George Biros. A boundary integral method for simulating the dynamics of inextensible vesicles suspended in a viscous fluid in 2d. *Journal of Computational Physics*, 228(7):2334 – 2353, 2009.

[109] Shravan K Veerapaneni, Abtin Rahimian, George Biros, and Denis Zorin. A fast algorithm for simulating vesicle flows in three dimensions. *Journal of Computational Physics*, 230(14):5610–5634, 2011.

[110] Paul Vigneaux. *Méthodes Level Set pour des problèmes d'interface en microfluidique*. PhD thesis, Université Bordeaux I, 2007.

[111] Olivier Vincent. *Dynamique de bulles de cavitation dans de l'eau micro-confinée sous tension. Application à l'étude de l'embolie dans les arbres.* PhD thesis, Université de Grenoble, 2012.

[112] Christophe Winkelmann. *Interior penalty finite element approximation of Navier-Stokes equations and application to free surface flows*. PhD thesis, Ecole polytechnique federale de Lausanne EPFL, 2007.

[113] Sheng Xu. The immersed interface method for simulating prescribed motion of rigid objects in an incompressible viscous flow. *Journal of Computational Physics*, 227(10):5045–5071, 2008.

[114] Steven T Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics*, 31(3):335 – 362, 1979.

[115] Junfeng Zhang, Paul C Johnson, and Aleksander S Popel. An immersed boundary lattice boltzmann approach to simulate deformable liquid capsules and its application to microscopic blood flows. *Physical biology*, 4(4):285, 2007.

[116] Hong Zhao, Amir HG Isfahani, Luke N Olson, and Jonathan B Freund. A spectral boundary integral method for flowing blood cells. *Journal of Computational Physics*, 229(10):3726–3744, 2010.

[117] Hong Zhao and Eric SG Shaqfeh. The dynamics of a vesicle in simple shear flow. *Journal of Fluid Mechanics*, 674:578, 2011.