Simulation numérique d'écoulements multi-fluides sur grille de calcul

Olivier Basset

Ecole Doctorale 364 : Sciences Fondamentales et Appliquées

N° attribué par la bibliothèque

|_|_|_|_|_|_|_|_|_|_|_|_|

T H È S E

pour obtenir le grade de
Docteur de l'École des Mines de Paris
Spécialité «Mécanique Numérique»

présentée et soutenue publiquement par

M. Olivier BASSET

le 21 décembre 2006

SIMULATION NUMERIQUE D'ÉCOULEMENTS MULTI FLUIDES SUR GRILLE DE CALCUL

Directeur de thèse : Thierry COUPEZ
Encadrant de thèse : Hervé GUILLARD

Jury :

M.	François-Xavier ROUX	Rapporteur
M.	Alain DERVIEUX	Rapporteur
M.	Thierry COUPEZ	Examinateur
M.	Hervé GUILLARD	Examinateur
M.	Hugues DIGONNET	Examinateur

Table des Matières

INTRODUCTION GENERALE

Objectifs

L'objectif de cette thèse est de simuler des écoulements de fluides incompressibles à plusieurs phases avec capture d'interfaces en mouvement, dans un contexte de calcul parallèle et de grille de calcul. Cette étude utilise une approche éléments finis en mécanique des fluides bien adaptée à la modélisation directe incompressible, ou faiblement compressible.

Puisque aucune théorie rigoureuse ne semble en mesure de prédire les phénomènes complexes dans lesquels différents processus sont couplés (comme les effets de cisaillement, de la turbulence, et des interactions inter fluides), le développement de techniques de simulation numérique apparaît donc comme une voie prometteuse pour comprendre et prédire de tels écoulements. Un des buts de ce travail est ainsi de développer un code de simulation numérique capable de suivre le mouvement des interfaces dans un écoulement multiphasique complexe. Mais aussi, cela exige souvent l'utilisation de ressources de calcul importantes et performantes. Nous avons à notre disposition des outils de calcul parallèle diverses dont deux clusters (grappes de PC Pentium 3 et Pentium 4), et une grille de calcul.

C'est ainsi que les travaux de cette thèse s'inscrivent dans le cadre du projet MecaGrid qui vise à bâtir une grille de calcul régionale à partir de grappes de PC géographiquement dispersées pour des applications de calculs intensifs parallèles en mécanique des fluides hétérogènes.

Une bonne utilisation de ces outils passe non seulement par une connaissance approfondie de leurs performances théoriques et réelles, mais aussi par une parallélisation efficace des codes de calculs.

Introduction des méthodes numériques implémentées

Tout au long de cette thèse, nous implémentons, étudions, et comparons plusieurs types de méthodes numériques. L'idée est de garder une approche la plus simple possible, avec une approximation continue, qui facilite la mise en œuvre et la parallélisation.

Notre modélisation est basée sur le couplage entre deux problèmes distincts que représentent le calcul de l'écoulement et la capture des interfaces. Tous les écoulements sont considérés ici comme Newtonniens incompressibles, et peuvent donc être calculés grâce à la résolution du problème de *Navier-Stokes* incompressible. Cette hypothèse est tout à fait justifiée puisque notre étude ne comprend pas de très forte vitesse. En mécanique des milieux continus, le mouvement d'un fluide tel que l'eau, l'huile, l'air, le métal fondu ou les polymères, peut être traité par ce type de modèle. Ensuite, une équation de transport permet de faire évoluer les phases présentes dans l'écoulement, et donc l'interface qui les séparent, en se basant sur le champs de vitesse unique déterminé par le calcul de l'écoulement.

Présentation générale

Dans le chapitre 1 de cette thèse, nous présentons le projet MecaGrid dans son ensemble, depuis la conception de la grille de calcul [Foster, 1999], jusqu'à son utilisation. Nous étudions cet outil très particulier afin de mieux le comprendre et de faire en sorte de

bien s'en servir. Des techniques d'optimisation sont notamment nécessaires afin contourner ses inconvénients.

Dans le cadre de ces travaux, nous avons choisi des méthodes éléments finis sur maillage non structuré. Les chapitres 2 et 3 sont dédiés à l'étude de ces méthodes numériques : les solveurs de l'écoulement et du transport de l'interface sont étudiés et validés indépendamment, avant de les regrouper dans le chapitre 4.

Dans le chapitre 2, nous décrivons la méthode de résolution des équations de *Navier-Stokes* incompressible que nous avons implémenté pour calculer les écoulements de fluides Newtoniens. Après avoir correctement posé la formulation forte, une méthode de type éléments finis mixtes à interpolation P1+/P1 est appliquée en s'appuyant sur un maillage composé de tétraèdres. Cette technique, nommée *Mini-Elément* [Coupez, 1997], consiste à condenser une bulle pyramidale, et rappelle les approches du type *multiscale* [Hughes, 1998]. Ensuite, nous validons ce solveur en appliquant des cas tests reconnus.

Le chapitre 3 traite des méthodes numériques de capture d'interface, et présente le formalisme que nous avons choisi d'utiliser : le *Level Set*. Une équation de transport (équation de convection pure), stabilisée par *SUPG* [Brooks, 1982] ou *Residual-Free Bubbles* [Brezzi, 1997], est couplée avec une technique de réinitialisation [Sussman, 1994]. L'étude de ce solveur à interpolation continue P1 comprend une comparaison de performance et de résultat entre notre modèle et une approche Galerkin discontinu de la capture d'interface apparentée à une approche *Volume of Fluid*. Notamment, nous montrons que l'approche proposée règle les problèmes de diffusion observés lorsque l'on parle de capture d'interface avec *Volume of Fluid* [Coupez, 2000].

Enfin, dans le chapitre 4, nous cherchons à évaluer le couplage entre la résolution des équations de Navier-Stokes et le suivi d'interface pour la simulation d'écoulements. Une série de cas test nous permettent de valider qualitativement et quantitativement les simulations numériques, et différentes applications sont donc proposées et comparées à des résultats expérimentaux.

Publications

Les résultats de cette thèse ont fait l'objet des publications suivantes:

P. Laure, G. Beaume, O. Basset, L. Silva, T. Coupez
Numerical methods for solid particles in particulate flow simulations
Revue Européenne de Mécanique Numérique (2006)

O. Basset
MecaGRID, rapport interne CEMEF
Ecole des Mines de Paris (2006)

T. Coupez, O. Basset, L. Silva, H. Digonnet
Adaptive Capturing methods for moving surfaces and interfaces in material forming
7th World Congress on Computational Mechanics, Los Angeles, California (2006)

P. Laure, G. Beaume, O. Basset, L. Silva, T. Coupez
Les méthodes numériques pour les écoulements de fluides chargés
1er colloque du GDR interactions Fluide-Structures, Nice (2005)

O. Basset, H. Digonnet, H. Guillard, T. Coupez
Multiphase flow calculation with interface capture coupled solution
Int. Conf. on Computational Methods for Coupled Problems in Science and Engineering ECCOMAS, Santorin, Grèce (2005)

T. Coupez, O. Basset, H. Digonnet
Stabilisation for capturing moving surface and incompressible Navier Stokes solution
Conference on Material Forming ESAFORM, Roumanie (2005)

H. Digonnet, O. Basset, T. Coupez, L. Silva
Calcul parallèle appliqué aux écoulements de fluides complexes
Congrès Français de Mécanique, France (2005)
(A paraître dans la Revue Européenne de Mécanique Numérique)

T. Coupez, O. Basset, H. Digonnet, L. Silva
Capturing techniques for moving free surfaces and interfaces with application in material forming flows
Trends in Numerical and Physical Modeling for Industrial Multiphase Flows, Cargese, Corse (2005)

LE PROJET MECAGRID

1. Introduction

Les machines de calcul basées sur les réseaux sont récemment apparues comme une alternative pour permettre aux utilisateurs d'accéder à des ressources et des puissances de calcul considérables. Des applications parallèles comme SETI@home ont rendues populaire le concept des grilles. De plus, le développement et l'amélioration très rapide des réseaux rendent possible la considération pour des calculs massivement parallèles sur des machines autres que des "superordinateurs". Ces nouveaux outils sont appelés des grilles de calcul, qui ne sont d'autres que des clusters de PC interconnectés. Elles peuvent être vues comme des ordinateurs virtuels distribués et constitués d'un ensemble hétérogène de super nœuds, où chaque super nœud est homogène (ici, un cluster).

L'article [Foster, 1999] est reconnu comme la proposition de développement la plus intéressante pour la nouvelle technologie des grilles de calcul. Le succès de leur développement repose sur plusieurs facteurs : l'amélioration rapide de la puissance des processeurs et du débit des réseaux, la disponibilité des middlewares permettant l'interconnexion de ressources de calcul et de données, l'implémentation (ou l'amélioration) d'algorithmes parallèles hétérogènes, et enfin, la participation et le dévouement des utilisateurs. L'environnement des grilles diffère des machines parallèles standard pour plusieurs aspects :

- elles sont hétérogènes dans leur puissance de calcul (processeur et mémoire virtuelle),
- elles ont des débits de communication moins performants et plus hétérogènes,
- elles sont plus instables de part leur dépendance à des institutions autonomes, et
- elles peuvent intégrer des systèmes d'exploitation très différents.

Le développement des grilles de calcul commença dans le milieu des années 1990. Le but est de rendre disponible différentes ressources de calcul et de données sur une grille afin de réaliser des projets scientifiques qui seraient impossibles ou, surtout, trop coûteux dans un site esseulé. Toute une technologie se développe aujourd'hui, et depuis quelques années, sur ce sujet, et ce sont des organisation telles que iGrid [Brown, 1999] ou Globus Alliance [Foster, 1997] qui semblent être les mieux reconnues à l'heure actuelle dans le monde.

Ce chapitre présente différentes problématiques posées lors de la construction d'un tel outil dédié aux calculs en mécaniques des fluides, ainsi que lors de son utilisation avec la librairie MPI. Après avoir montré une vue d'ensemble sur ce qu'il se fait à l'heure actuelle sur les grilles de calcul, les thèmes abordés sont les suivants : le projet MecaGrid et l'ensemble des ressources disponibles pour créer une grille, les principaux intérêts portés sur la grille, les difficultés techniques rencontrées pour connecter des clusters avec adresses IP privées, les technologies employées pour les résoudre, les performances réelles obtenues et leurs conséquences. Puis, plusieurs techniques d'optimisation sont proposées en tenant compte des performances observées. Enfin, des applications sont étudiées : leurs résultats expérimentaux montrent combien il est possible d'améliorer les performances de la grille de calcul avec nos différentes techniques d'optimisation.

2. État de l'art des grilles de calcul

2.1. Technologie existante

Bien que chaque grille soit unique, il y a des problèmes de conception récurrents qui apparaissent. C'est justement parce qu'ils sont rencontrés souvent que des organisations développent des solutions les résoudre. Leurs principales préoccupations sont les suivantes :

- Sécuriser des données qui, pourtant, doivent être accessibles à un grand nombre de personnes et d'institutions.
- Donner une interface facile à utiliser tout en cachant une grande complexité, comme la diversité des ressources et des configurations.
- Créer un protocole de transfert de fichiers très rapide, puisqu'il est fort probable que le débit des réseaux soit moins performant qu'avec n'importante quelle autre application.
- Intégrer les différentes politiques des institutions.

Même si le concept des grilles de calcul est en développement depuis plus de dix ans, il reste difficile de construire de tels outils. Ainsi, les logiciels *plug-and-play* n'existent, à l'heure actuelle, que pour les *Desktop Computing* et les grilles d'entreprise qui sont simplifiées par le fait qu'elles profitent de leurs propres réseaux et administrateurs. Mais, ce type de logiciel n'existe pas encore pour les grilles de type *Virtual Organizations* (VO). On appelle VO un regroupement d'institutions totalement indépendantes les une des autres. Cependant, d'autres logiciels comme celui de Globus Alliance [Foster, 1997] (http://www.globus.org) permettent de créer des grilles VO dans un laps de temps raisonnable (de l'ordre de quelques mois).

2.2. Exemples d'applications en termes de simulations numériques

Une des plus grandes organisations s'occupant du développement des grilles est l'iGrid [Brown, 1999]. Elle regroupe des centres de recherche du monde entier : on y trouve des participants des USA, d'Australie, du Canada, d'Allemagne, du Japon, de Hollande, de Russie, de Suisse, de Singapour, et de Taiwan. Ils identifient quatre principales utilisations possibles des grilles :

- Applications distribuées : calculs parallèles qui sont exécutés sur des machines parallèles, qui font appelle à des librairies (*API*) pour l'allocation des ressources, l'autorisation, authentification et les communications (comme la librairie MPI et le middleware Globus). Comme exemples d'applications à grande échelle, on trouve des calculs en mécanique des fluides, ou des simulations de collisions de trous noirs.
- Applications sur demandes (*on demand*) : accès à distance d'instruments tels que des microscopes ou des accélérateurs.
- Applications de données intensives : recherche et visualisation basées sur des bases de données.
- Applications collaboratives : accès à des calculs ou des bases de données en temps réel, comme dans les diagnostiques médicaux ou la télé-immersion.

2.2.1. Analyse d'images satellites

Parmi les secteurs les plus demandeurs de la technologie des grilles, on trouve l'analyse d'images satellites [Lee, 1996] : des données apportées en très grand nombre par les satellites a généralement besoin d'importantes machines distribuées pour les stocker, les analyser, les visualiser, les partager…

2.2.2. Modélisation climatique

Dans le domaine de la modélisation climatique [Allcock, 2001] on retrouve les mêmes besoins. Des simulations de problèmes complexes ont une durée très grande, et une énorme quantité de données résultent de ces calculs sous formes de fichiers qui doivent être analysés. Ils sont donc très intéressés à la création d'une grille qui puissent analyser, transporter et distribuer des quantités importantes de données et gérer un très grand nombre d'utilisateurs. En effets, dans ce domaine, des équipes de chercheurs se trouvent très éloignés les uns des autres tout autour de la planète et doivent se communiquer des données sur les différents climats terrestres.

2.2.3. Étude des tremblements de terre

Dans [Pearlman, 2004], les auteurs mettent en évidence les progrès considérables que les gilles apportent à l'ingénierie des tremblements de terre. Les ingénieurs investiguent le comportement des structures terrestres avec à la fois des simulations numériques et des expériences physiques. Récemment, une nouvelle approche hybride demande une analyse distribuée des données expérimentales et calculées : ils font donc appel à la technologie des grilles pour construire NEESGrid (*Network for Earthquake Engineering Simulation*). NEESgrid permet de rassembler et accéder à des données géologiques récoltées dans différentes parties du monde par des ingénieurs dispersés, puis de les analyser avec des calculs intensifs et massivement parallèles. Le résultat est l'apparition de toutes nouvelles méthodes de travail bien plus efficaces.

2.2.4. Simulations numériques de la relativité

Un record de taille pour une simulation en relativité numérique a été réalisé sur grille de calcul par les auteurs de [Allen, 2001]. Le logiciel nommé Cactus [Allen, 1999] résout les équations d'Einstein pour simuler numériquement l'évolution d'une vague gravitationnelle. Leur modèle s'appuie sur une méthode de différence finie et une fonction discrétisée sur une grille régulière. En plus de la relativité numérique, Cactus est de plus en plus utilisé à des fins différentes comme en astrophysique, chimie, propagation de fissures, et modélisation des climats. Pour les calculs scientifiques parallèles, il se base sur les librairies MPI et PETSc, et le middleware Globus. Une application avec un maillage à environ 720 millions de nœuds a été exécutée avec succès sur les 1500 processeurs d'une grille de calcul.

Note : La puissance d'une machine parallèle se définie ainsi :

$$Flops/s = \frac{NbFlops_{total}}{Temps_{total}} \qquad (1)$$

où $NbFlops_{total}$ est le nombre d'opérations effectué sur tous les processeurs, et $Temps_{total}$ est la durée de l'exécution. Cette définition nous servira par la suite pour évaluer la puissance de nos outils de calcul.

Sur leur grille de calcul, ils obtiennent initialement une puissance de 42 Gigaflops par seconde ; puis, ils atteignent 249 Gigaflops par seconde en améliorant la répartition de données par partitionnement et en diminuant les communications par compression. Cet exemple impressionnant par sa taille montre à quel point les grilles de calculs peuvent fournir des calculateurs très puissants, à condition de bien les maîtriser.

2.2.5. Recherche de la Nasa

L'article [Barnard, 1999] décrit une expérience menée par la Nasa, dans laquelle une application en calculs de dynamique des fluides à grande échelle est adaptée pour être exécutée de façon performante sur un environnement distribué. Cette grille de calcul est créée grâce au middleware Globus [Foster, 1997]. Un logiciel de simulation numérique basé sur la librairie de MPI [Gropp, 1996] et appelé OVERFLOW [Buning, 1995] est utilisé pour expérimenter la grille de calcul.

La Nasa veut développer une nouvelle génération d'outils de conception (*next generation design tools*) pour améliorer la précision des simulations et diminuer de moitié les cycles de développement des avions. Afin de réussir un tel challenge, il est nécessaire d'apporter une amélioration conséquente dans la façon de créer, calculer, comprendre, stocker et communiquer les données. Il n'est cependant pas probable que tout ceci puisse passer par l'utilisation de machines parallèles conventionnelles, même très performantes. La Nasa est donc en train de construire une infrastructure à l'échelle des Etats-Unis appelée *Information Power Grid* (IPG). L'IPG a pour but de fournir un accès uniforme à travers une interface pratique, à des ressources très importantes de calcul, de communication, d'analyse et de stockage des données. Pour créer un environnement de calcul transparent avec une bonne scalabilité et adaptabilité, d'importantes ressources hétérogènes et géographiquement dispersées aux Etats-Unis sont reliées. L'utilisation de cette machine parallèle virtuelle unifiée passe par une interface qui ne présente pas de complications par rapport à celle d'une machine plus conventionnelle (comme un superordinateur ou un cluster).

Les deux principales utilisations de cet outil sont les suivantes : la première est la mise à disposition de technologies communes à des collaborateurs géographiquement dispersés et indépendants ; la seconde est de permettre de résoudre des applications extrêmement importantes et massivement parallèles, comme le font les clusters.

Cette application est très importante pour la Nasa qui bientôt, aura besoin de dépasser les ressources habituellement disponibles sur les machines parallèles de leurs sites ou celles de leurs partenaires. Une mauvaise répartition des données, ainsi que des temps de communication trop élevés, sont identifiés comme leurs principales sources de mauvais résultats. L'hétérogénéité de leur grille IPG et les distances qui séparent leurs différentes ressources de calculs sont donc à la base des problèmes de performance. Des techniques d'optimisation de répartition de la masse de calcul sont appliquées avec un certain succès, mais ils estiment que des efforts doivent encore être faits pour que IPG soit un environnement utile et accessible aux calculs de grandes tailles.

Avec les applications d'un véhicule de retour d'équipage et d'un hélicoptère de l'armée, l'utilisation de la grille de calcul est comparée à celle d'un cluster. Après avoir obtenu de mauvais résultats sur grille avec un maillage à 2.5 millions de nœuds, ils proposent des techniques d'optimisation telles qu'un schéma en temps décalé (*deferred scheme*) et une répartition adaptée de la masse de calcul et des communications. C'est avec le logiciel MeTiS [Karypis, 1995], plus quelques améliorations qu'un partitionnement optimisé est effectué. Les résultats qu'ils obtiennent sont assez encourageants, mais ils considèrent que beaucoup de travail et de réflexions doivent encore être menés pour pouvoir utiliser la grille dans des buts de production.

3. Le projet MecaGrid

3.1. Description du projet

Le projet MecaGrid est soutenu par l'ACI GRID – Action Concertée Incitative (ACI) Globalisation de Ressources Informatiques et des Données (GRID) – du ministère de la recherche depuis octobre 2002. L'ACI GRID a pour objectif de renforcer les recherches françaises menées autour des grilles de calcul ou de données. Le projet MecaGrid vise à bâtir une grille de calcul régionale (en PACA) à partir de grappes de PC (clusters) pour des applications de calculs intensifs parallèles en mécanique des fluides hétérogènes. Plus précisément, la grille de calcul sera dédiée à la simulation numérique multi matériaux, dont des écoulements multi fluides avec capture d'interfaces. Pour cela, des codes de calcul utilisant la librairie MPI doivent pouvoir être exécutés sur la grille sans pour autant avoir à en changer l'implémentation parallèle. Les clusters situés sur plusieurs sites différents dans la région PACA qui forment MecaGrid sont des clusters de production. C'est-à-dire qu'ils sont utilisés constamment et individuellement par les centres de recherche auxquels ils sont rattachés. Ceci est un aspect très important dans l'approche que l'on aura vis-à-vis de cet outil de calcul. Dans sa phase initiale, MecaGrid regroupe les clusters du CEMEF (Centre de Mise en Forme des Matériaux de l'Ecole des Mines de Paris à Sophia-Antipolis), de l'INRIA Sophia-Antipolis, et de l'IUSTI (Institut Universitaire des Systèmes Thermiques et Industriels) à Marseille.

Le but final du projet est de mettre en valeur le concept des grilles de calcul dans le cadre de simulation en mécanique des fluides hétérogènes. Il s'agit de montrer ce qu'il est possible de faire à l'heure actuelle avec les moyens et les ressources disponibles. De plus, cette étude permet de mieux comprendre le fonctionnement très particulier des grilles de calculs et d'identifier les configurations et les utilisations à améliorer dans l'avenir. Les méthodes présentées dans cette thèse ont pour but de permettre la réalisation de simulations numériques sur MecaGrid. C'est pourquoi, elles sont toutes choisies et développées de sorte à garder au maximum une simplicité de résolution et de stockage, dans l'optique de permettre des applications de grande taille.

3.2. Difficultés administratives

Les grilles de calcul de type *Virtual Organizations* sont particulièrement délicates à mettre en place pour plusieurs raisons. Les ressources de calcul sont situées à différents endroits géographiques, et les institutions indépendantes auxquelles elles appartiennent ont leurs propres priorités, administrateurs, gestionnaires de batch, systèmes de queues, procédures de sécurité, matériels, et utilisateurs. Tout ceci représente autant de conflits potentiels lorsque l'on parle de coordination : les règles locales peuvent contredire les règles communément décidées pour la globalité de la grille. De plus, les différents clusters sont des structures « vivantes », qui évoluent constamment et indépendamment des autres clusters de la grille (exemple : changement de configuration ou de version de librairie, évolution des machines…). Il apparaît donc évident qu'un administrateur « global » doit coordonner tous

les changements de configuration qui interviennent localement, et veiller au bon fonctionnement global de la grille de calcul.

3.3. Ressources disponibles

Les ressources de calcul qu'utilise MecaGrid sont des clusters situés sur les différents sites des membres du projet. Ces ressources sont assemblées pour donner naissance à un outil de calcul parallèle de plus grande taille que ceux qui sont disponibles localement.
Le site de l'INRIA Sophia a 2 clusters différents reliés à MecaGrid. Le premier (appelé *nina*) contient 16 biprocesseurs Xeon à 2 GHz. Ils sont reliés entre eux par un réseau Ethernet à 1 Gigabit/s (1 Gb/s). Le deuxième (appelé *pf*) contient 19 biprocesseurs Pentium III à 933 MHz. Ils sont eux reliés par un réseau Fast-Ethernet à 100 Mégabit/s (100 Mb/s). Le cluster de l'IUSTI à Marseille est constitué de 32 monoprocesseurs Pentium IV à 2 GHz avec un réseau Fast-Ethernet à 100 Mb/s. Enfin, le cluster situé sur le site du CEMEF contient 32 biprocesseurs Pentium III à 1 GHz reliés entre eux par un réseau Myrinet à 100 Mb/s.
Ce tableau résume toutes les caractéristiques de l'architecture théorique de MecaGrid :

Site	cluster	Adresse IP	CPUs	GHz	RAM/nœud	CPUs/nœud	LAN
INRIA Sophia	nina	publique	32	2.0	1 Go	2	1000 Mb/s
INRIA Sophia	pf	publique	38	1.0	512 Mo	2	100 Mb/s
IUSTI	iusti	privée	30	2.0	512 Mo	1	100 Mb/s
CEMEF	cemef	privée	64	1.0	512 Mo	2	100 Mb/s

Tableau 1 – Ressources de calculs disponibles

Chaque cluster a sa propre adresse IP, soit privée, soit publique, et il y a un total de 164 processeurs disponibles. Leur puissance de calcul est quantifiée en GHz, et on observe un facteur deux entre la fréquence des plus lents et des plus rapides. RAM/nœud montre la quantité de mémoire vive disponible sur chaque nœud (en Gigaoctet et Mégaoctet). CPUs/nœud est le nombre de processeurs que contient chaque nœud : certain sont des monoprocesseurs (un processeur par nœud), et d'autres sont des biprocesseurs (deux processeurs par nœuds). Enfin, le débit des réseaux locaux (LAN ou *Local Area Network*) est mesuré en Mégabit par seconde (Mb/s).

Une des informations les plus importantes à retenir du tableau *(1)* est la présence d'adresses IP publiques pour certains clusters et privées pour d'autres. Nous verrons par la suite les difficultés techniques que cela entraîne sur la construction de la grille, et les solutions que différentes technologies peuvent apporter pour les résoudre.

La colonne RAM/nœud du tableau *(1)* montre la quantité de mémoire virtuelle disponible sur chaque nœud. Or, les clusters à biprocesseurs (ceux de l'INRIA et celui du CEMEF) ont deux processeurs par nœud qui doivent se partager, non seulement la quantité de RAM contenue sur le nœud, mais aussi l'accès à cette mémoire. En effet, il n'existe qu'un seul bus d'accès à la mémoire pour deux processeurs d'un même nœud. Nous verrons par la suite que ce dernier point est loin d'être négligeable lorsque l'on parle du niveau de performance. Une distinction importante doit être dressée au sujet des deux utilisations possibles de ces biprocesseurs (emploi de un ou deux processeurs par nœud). Par exemple, si une simulation est lancée avec deux processus par nœud sur le cluster du CEMEF, la mémoire disponible pour chaque processeur n'est plus de 512 Mo, mais de 256 Mo, si l'on considère

que les données sont parfaitement distribuées. Une étude plus approfondie suivra au sujet des performances qu'offrent ces deux utilisations des biprocesseurs.

3.4. Intérêts d'une grille de calcul

L'intérêt que l'on porte sur les grilles de calcul à l'heure actuelle porte sur plusieurs points. Tout d'abord, avoir la possibilité d'effectuer des calculs plus important que sur un cluster isolé peut constituer un premier objectif. La quantité de RAM que contiennent les nœuds représente une limite dans l'utilisation des processeurs ; et de ce fait, plus la simulation est de taille importante, plus le nombre de processeurs nécessaires à l'exécution est grand. Il est donc appréciable d'avoir accès à des ressources de calcul supérieures à celles que l'on peut généralement avoir dans les centres de recherche ou de production à un coût qui reste raisonnable. Relier des clusters de production indépendants entres eux semble plus avantageux que d'acquérir une machine contenant la puissance réunie de cette manière.

Ensuite, le contexte des simulations de grandes tailles entraîne des problèmes qui interviennent lorsque l'on n'a accès qu'à un cluster local. Plus particulièrement, ce sont le nombre des processeurs réservés et les temps de calcul qui peuvent créer des obstacles sur un cluster de production isolé. En effet, l'exécution de gros calculs demande souvent la réservation d'un grand nombre de processeurs sur une longue durée, ce qui a pour conséquence d'entraîner deux inconvénients majeurs. Premièrement, de telles machines étant utilisées en permanence et par plusieurs personnes, il devient parfois difficile de trouver un nombre suffisant de processeurs disponibles afin d'effectuer la réservation et de commencer la simulation. De cette manière, une requête peut rester bloquée dans la file d'attente pendant une période relativement longue avant de pouvoir être exécutée. Deuxièmement, il est quasiment inenvisageable de réserver ne serait-ce que la moitié ou le trois-quarts des ressources de calcul d'un cluster pendant une période dépassant les quelques jours, ou la semaine. Ceci entraînerait une saturation, et de longues files d'attentes se formeraient pour les autres utilisateurs.

MecaGrid est conçue pour donner des solutions à ces différentes préoccupations. En reliant plusieurs clusters de production, les ressources de calculs sont considérablement augmentées, et des calculs de grande taille dépassant les possibilités d'un cluster isolé deviennent potentiellement réalisables. De plus, les processeurs disponibles étant largement plus abondants sur la grille, en réserver un grand nombre sur une longue période ne représente plus un problème, ni pour l'intéressé (temps d'attente diminué), ni pour les autres utilisateurs (pas de saturation de la file d'attente). Par exemple, il est clair que réserver 24 nina parmi les 32 disponibles (ce qui correspond à 75% des ressources) est bien plus problématique que réserver 24 processeurs sur les 164 que contient la grille (ce qui représente à peine 15% des ressources).

Pour conclure, MecaGrid n'a pas la prétention de fournir un outil de calcul plus performant qu'un cluster, mais de rendre disponible des ressources bien plus importantes, tout en optimisant au maximum ses performances. Reste à déterminer la technologie à utiliser, puis à étudier les caractéristiques et l'utilisation d'un tel outil de calcul.

3.5. Technologie de MecaGrid

3.5.1. Difficultés techniques

Les administrateurs des centres de recherche de l'INRIA, du CEMEF et de l'IUSTI ont participé activement à l'élaboration de cette grille de calcul sur laquelle des applications implémentées avec la librairie MPI doivent pouvoir être exécutées sans en changer le code. Ils ont collaboré ensemble pour mettre en place l'architecture et déterminer les technologies nécessaires pour y parvenir en tenant compte des différentes politiques (voir l'introduction). La principale difficulté technique réside dans la connexion de plusieurs clusters à adresses IP publiques et privées. En effet, la plupart des middlewares pour grilles n'acceptent pas les adresses privées.

Le fait qu'il y ait des clusters à adresse IP privée et d'autres à adresse publique pose un problème technique pour bâtir les liaisons. La communication entre deux nœuds arbitraires de sites différents, comme le nécessite la librairie MPI, suppose que les LAN de tous les clusters soit « visibles » ou accessibles par internet. Or, les paquets TCP ne peuvent pas circuler dans un réseau global si une des parties (source ou destination) a une adresse IP privée. Il faut donc trouver un moyen pour permettre la communication inter processeur dans la globalité de la grille.

3.5.2. Solutions envisageables

Une solution serait de concevoir une grille qu'à partir de cluster à adresse IP publique, mais pour des raisons administratives, et pour rassembler le maximum de ressources un autre moyen doit être employé.

La deuxième possibilité envisageable est de faire tourner MPICH avec des connexions distantes sécurisées de type SSH. MPICH est une implémentation spéciale de la librairie MPI [Gropp, 1996] v1.1 standard pour les grilles de calcul. Les problèmes engendrés par cette technique se situent au niveau de la communication inter clusters et sont liés à la façon dont elle est gérée par MPICH. Lors du lancement des processus MPI, la connexion directe sur chaque nœud, n'est pas permise. A l'exécution, et suivant les communications requises, le service CH_P4 de MPICH est amené à ouvrir des connexions supplémentaires entre les noeuds sur des ports dont nous ne connaissons pas à l'avance le numéro. Cela pose un gros problème de sécurité car il faudrait autoriser des connexions sur n'importe quel port, ce qui est incompatible avec la politique de sécurité des réseaux des différents sites. Une telle solution (MPICH/SSH) a donc vite été écartée à cause des modifications qu'elle oblige à opérer sur les architectures réseaux locales.

Une troisième solution pour relier des clusters à adresse privée consiste à créer un VPN (*Virtual Private Network*) qui, lui, est entièrement compatible avec MPI. Ce type de réseau fourni un outil totalement transparent pour gérer les envois de messages. Avec l'installation du VPN, il n'est pas nécessaire d'avoir une adresse IP publique par nœud, mais seulement une par site, qui est abritée par une machine « frontale ». Cette frontale n'est pas un nœud, mais une machine dédiée au réseau VPN. Bien que les clusters de l'INRIA ai des adresses publiques, ils sont considérés comme aillant des adresses privées pour qu'ils puissent intégrer ce VPN qui établi les communications au sein de MecaGrid. Plus précisément, le VPN crée des tunnels entre toutes les machines frontales qui elles ont des adresses IP

publiques. C'est ainsi que les communications entre les différentes ressources de calcul géographiquement dispersées vont s'opérer : chaque paire de frontale est connectée entre elles par un tunnel qui crypte les données, encapsule et compresse les paquets, puis les transmet. Le cryptage assure une certaine sécurité, et la compression permet d'améliorer le débit à travers les tunnels. Pour cela, on utilise CIPE, un logiciel simple qui gère les tunnels en transmettant les paquets TCP sous forme de paquets UDP encryptés sur des ports définis par l'utilisateur. Le VPN fait en sorte que chaque processeur peut communiquer avec n'importe quel autre processeur de la grille, et ce, qu'il fasse parti du même site ou pas. Les communications locales (entre deux processeurs du même site) se font par le réseau local (LAN), tandis que pour les autres, c'est la frontale qui est considérée comme le lieu de passage obligatoire. Ainsi, un message lancé depuis un nœud du site de l'INRIA jusqu'à autre nœud situé à l'IUSTI passe en premier dans la frontale de l'INRIA. Celle-ci a été configuré pour lancer le message dans le tunnel approprié, en direction de la frontale de l'IUSTI, qui elle, décrypte et décompresse les données avant de les rediriger vers le nœud de destination par le LAN. En fait, le VPN fonctionne comme si tous les processeurs faisaient partie d'un WAN (*Wide Area Network*).

3.5.3. Schéma conceptuel de MecaGrid

La figure *(1)* représente de façon schématique l'architecture théorique de MecaGrid. Les machines frontales par lesquelles passent les communications inter sites sont représentées par des carrés. Les nœuds reliés entre eux en forme d'étoile représentent les différents clusters. Enfin, les connexions inter sites qui transitent par internet sont en noir : ce sont les tunnels.

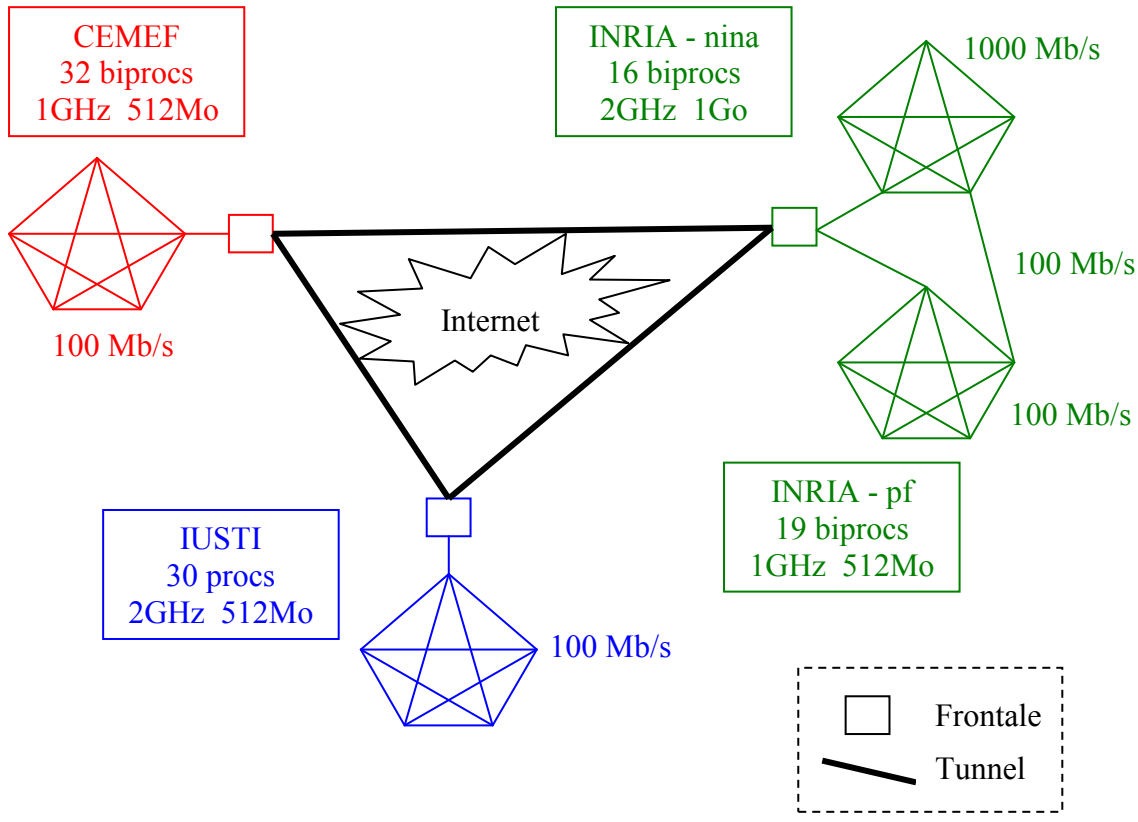


Figure 1 – Architecture théorique de MecaGrid

Ce VPN peut être mis en place avec le middleware de Globus Alliance [Foster, 1997]. Le prochain paragraphe détail l'utilisation des services inclus dans Globus2 qui nous apportent les technologies nécessaires à la construction du VPN.

3.5.4. Principes et fonctionnement de Globus2

Essentiellement pour des raisons de sécurité et d'administration, ce sont Globus 2.2.4 et MPICH-G2 [Foster, 1998] qui ont été choisis pour construire l'infrastructure de MecaGrid et atteindre l'objectif final : fédérer un ensemble de ressources et fournir à l'utilisateur une grille de calcul dont l'utilisation soit aussi simple qu'une machine parallèle classique.
MPICH-G2 est une implémentation spéciale de la librairie MPICH en utilisant exclusivement la version 2 de l'outil Globus.

Globus 2.2.4 est une boite à outils composée d'un ensemble de services dédiés au calcul sur grille. Ce middleware ne résout pas tous les problèmes que l'on peut rencontrer mais fournit plusieurs solutions intéressantes. Ses services interviennent dans les domaines de la sécurité (authentification, identification, confidentialité), de la gestion des ressources, de la communication (avec notamment un support concernant la librairie MPI), et de la gestion des requêtes. Pour l'interrogation et l'allocation des ressources, la communication entre ces services se fait grâce au langage RSL. Les principaux services dont nous avons besoin sont : GRAM, MDS et GridFTP. Voici brièvement le rôle de chacun d'eux :

- GRAM regroupe le *gatekeeper* et le *jobmanager*. Le *gatekeeper* sert à authentifier les requêtes RSL qui arrivent et détermine quelle est l'identité locale qui correspond au propriétaire de cette requête. Pour cela, il existe un fichier (grid-mapfile) associant un ensemble de certificats à un ensemble d'identités locales. La connexion au *gatekeeper* se fait sur le port 2119 en TCP et en provenance d'un port éphémère (supérieur à 1023) de la machine cliente. La requête est ensuite passée au *jobmanager* qui traduit la requête RSL dans le langage du gestionnaire de soumission local (LSF, PBS, ...).
- MDS constitue le service d'information de la grille et est basé sur OpenLDAP 2.0. Cet annuaire contient des informations sur tous les nœuds de la grille. MDS comprend le GRIS et le GIIS. Le GRIS fournit les informations sur une ressource spécifique (ex : un cluster). Le GIIS rassemble l'ensemble des informations mises à disposition par les GRIS et permet de sélectionner les machines de part leurs caractéristiques. Les connexions éventuelles se font à partir d'un port éphémère vers le port 2135 en TCP.
- GridFTP est l'équivalent sécurisé de FTP. Ce protocole fonctionne avec deux canaux : un canal de contrôle et un canal de transfert de données. La connexion de contrôle se fait à partir d'un port éphémère contrôlé par le client vers le port 2811 du serveur en TCP, et la connexion de transfert se fait d'un port éphémère du client vers un port éphémère du serveur.

L'ensemble des communications générées doit, évidemment, être authentifié et pour cela Globus utilise les certificats X.509 dont le principe est détaillé ici. Globus assure la sécurité de la grille grâce à GSI. Le GSI vérifie la provenance d'une requête et l'authenticité d'un service, à l'aide de certificats électroniques cryptés par un algorithme asymétrique.
Le principe du cryptage asymétrique est le suivant. Chaque utilisateur génère son couple de clé privée/publique qui lui servira à être authentifié par le module de sécurité de la grille. La clef privée est gardée par l'utilisateur alors que la clef publique est distribuée à tous. Si l'utilisateur crypte un message avec sa clef privée, tout le monde pourra le décrypter avec la clef publique et l'origine du message (utilisateur) sera garantie car seul l'utilisateur peut

crypter un message avec sa clef privée. Si une autre personne crypte un message avec la clef publique de l'utilisateur, seul celui-ci sera capable de le décrypter et donc de connaître le contenu. Un double cryptage (clef privé de A/clef publique de B) permet alors de certifier l'origine du message (venant de A) et que seul B peut le lire.
En résumé, dans Globus :

- l'utilisateur génère son couple de clefs
- garde sa clef privée dans un endroit sécurisé
- envoie sa clef publique à l'Autorité de Certification (AC)
- l'AC fabrique alors un certificat en cryptant avec sa propre clef privée :
    - le nom de l'AC
    - le sujet (nom de l'utilisateur, organisme, ...)
    - la clef publique de l'utilisateur
    - les dates de validité
- l'AC renvoie le certificat à l'utilisateur

A l'utilisation, le service de la grille (ex : *gatekeeper*) qui possède la clef publique de l'AC en question pourra décrypter le certificat de l'utilisateur et être sûr que la clef publique associée à l'utilisateur est la bonne. Le service demandera sous forme de challenge, à l'utilisateur de crypter un message pour voir si la clef privée détenue par l'utilisateur correspond à la clef publique certifiée par l'AC. Si tout se passe bien, l'utilisateur est considéré comme identifié de façon sûre. Les machines constituant la grille utilisent le même système de certificat pour passer les challenges d'authentification.

3.6. Conclusion

A ce stade, la grille de calcul est élaborée avec succès. Les différents problèmes rencontrés ont tous été résolus grâce aux technologies disponibles à l'heure actuelle. MecaGrid, qui relie les quatre clusters des centres de recherche participant au projet, est prête à être utilisée. Cependant, pour plusieurs raisons évoquées précédemment, il existe des moments où de nouveaux problèmes apparaissent, mais l'administrateur global est là pour restabiliser la grille. Les premières utilisations de MecaGrid sont consacrées aux tests de performance afin de mieux comprendre les caractéristiques de ce nouvel outil de calcul. Ensuite, une recherche importante sur l'optimisation d'utilisation sera menée pour contrebalancer les inconvénients de cette machine parallèle très particulière. Enfin, tout sera réuni pour permettre à l'utilisateur d'exécuter des applications à grande échelle dans le cadre de simulations numériques d'écoulement de fluides hétérogènes.

4. Performances réelles de la grille

4.1. Outil de mesure des performances

La partie précédente présente l'architecture théorique de MecaGrid, mais nous étudions ici la réalité, en contraste avec la théorie. Afin de connaître l'architecture réelle de la grille sous Globus, nous avons développé un outil, nommé Performance, qui mesure à la fois la vitesse des processeurs, et le débit de tous les réseaux que contient la grille de calcul. La durée d'exécution d'un test est d'environ 30 secondes. Voici comment s'opère la prise des mesures qui caractérisent la machine de calcul parallèle.

4.1.1. Vitesse des processeurs

Afin de mesurer la vitesse des processeurs, Performance mesure le temps que met chaque processeur pour calculer un produit scalaire de 100 000 réels codés en double précision. Ainsi, nous obtenons une mesure de la vitesse des processeurs en Mégaflops par seconde. Le flop correspond au nombre d'opérations effectuées (addition, soustraction, multiplication, division) sur des réels. Plus la mesure en Mégaflops par seconde (Mflops/s) est élevée, plus le processeur est puissant.

4.1.2. Débit des réseaux

L'idée est cette fois-ci de mesurer le temps nécessaire pour envoyer un message de données d'un processeur à un autre. Notre outil Performance mesure ces temps de communication en envoyant un paquet de 1000 réels en double précision (test de type « ping ») entre chaque pair de processeurs. Nous pouvons ainsi quantifier la vitesse des communications, ou le débit des réseaux, en Mégaoctets (Mo) par seconde.

4.2. Performances de la grille

Nous utilisons ici l'outil présenté précédemment pour évaluer les performances de MecaGrid. Pendant toute la durée du test, le programme est lancé plusieurs fois à intervalles de temps réguliers afin de mettre en évidence des éventuelles évolutions du débit des réseaux et de la vitesse des processeurs au sein de la grille de calcul.

Le premier enjeu est de mesurer les performances réelles de la grille. Notamment, c'est le débit du réseau reliant les trois sites qui n'est pas à priori connu avec précision, et qui a besoin d'être mesuré. Ainsi, nous allons pouvoir estimer le degré d'hétérogénéité de la grille, ce qui est très important lorsqu'il s'agit de calcul parallèle. En effet, la masse de calcul et les données doivent être réparties équitablement entre tous les processeurs en tenant compte de l'architecture de la machine parallèle. Dans le cas d'une machine parallèle homogène (comme un cluster), il s'agit simplement d'une division par le nombre de processeurs utilisés pour effectuer le calcul : c'est un partitionnement homogène (même masse de calcul et même quantité de donnée sur tous les processeurs). Comme le montre l'architecture théorique de MecaGrid, nous nous attendons à faire face à une machine de calcul hétérogène de part ses ressources. Les quantités mesurées dans cette partie ont pour but de déterminer l'hétérogénéité de la grille, pour ensuite pouvoir les étudier et optimiser l'utilisation d'un tel

outil. Suivant les résultats obtenus ici, il faudra notamment, ne plus partitionner de façon homogène, mais d'une manière à tenir compte de la puissance de chaque processeur et de la vitesse des différents réseaux.

Le deuxième enjeu consiste à savoir s'il est nécessaire de réajuster la répartition de la masse de calcul pendant la simulation numérique. La partition effectuée grâce aux caractéristiques mesurées vise à répartir au mieux la masse de calcul entre tous les processeurs disponibles. Cependant, si les caractéristiques de la grille changent au cours du temps, il est possible que la partition faite au début du calcul ne soit plus optimisée et ralentisse trop l'exécution du programme. On serait dans ce cas amené à repartitionner, puis continuer le calcul avec une nouvelle partition optimisée. Il faut donc être très attentif à la façon dont les caractéristiques de la grille évoluent au cours du temps.

4.2.1. Vitesse des processeurs

Le graphe suivant montre la vitesse des processeurs en Mflops/s en fonction du temps lors d'un test de performance avec un processeur sur chaque cluster : nina, pf, cemef et iusti. Un test de Performance a été lancé toutes les 30 min pendant 24h.

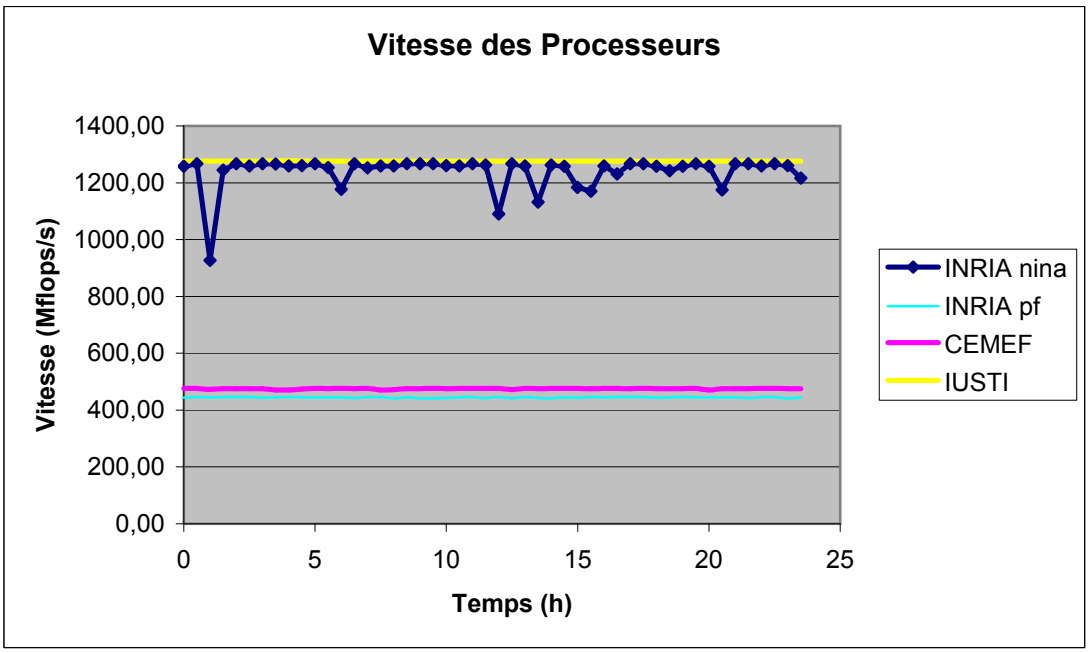


Figure 2 – Puissance des processeurs au cours du temps

D'autres tests ont été exécutés de la même façon en utilisant cette fois-ci 2 puis 4 processeurs sur chaque cluster. Aucune différence notable n'y apparaît en comparaison avec les résultats précédents. Ceci montre que les performances mesurées ici sont indépendantes du nombre de processeurs utilisés. Voici les moyennes des vitesses et des écarts types observées dans tous les tests effectués :

Processeurs	nina	pf	cemef	iusti
Mflops/s	125.71	44.45	47.59	127.52
Ecart Type %	3.03	0.33	0.11	0.19

Tableau 2 – Puissance moyenne des processeurs

Ces résultats sont conformes à l'architecture théorique de la grille. Par contre, la vitesse des processeurs nina de l'INRIA varie beaucoup plus que les autres : leur écart type moyen est de 3%, tandis que les autres vitesses ont un écart type moyen inférieur à 0.33%. Ces variations peuvent être dues au fait que ce sont des biprocesseurs. Mais, puisque les vitesses des autres biprocesseurs de la grille n'ont pas une telle variation, ceci ne représente pas une explication suffisante. Cependant, en observant la courbe de la puissance des nina, les variations aléatoires ne paraissent pas alarmantes, car, ce n'est qu'à certains moments bien précis et pendant une durée très courte que des bouleversements apparaissent. Le reste du temps, la vitesse parait constante.

4.2.2. Débit du réseau intra site (interne à chaque cluster)

La figure *(3)* montre le débit du réseau interne à chaque site en Mo/s en fonction du temps. Ces résultats ont été obtenus par des tests de Performance toutes les 30 min pendant 24h avec 2 processeurs sur chaque cluster.

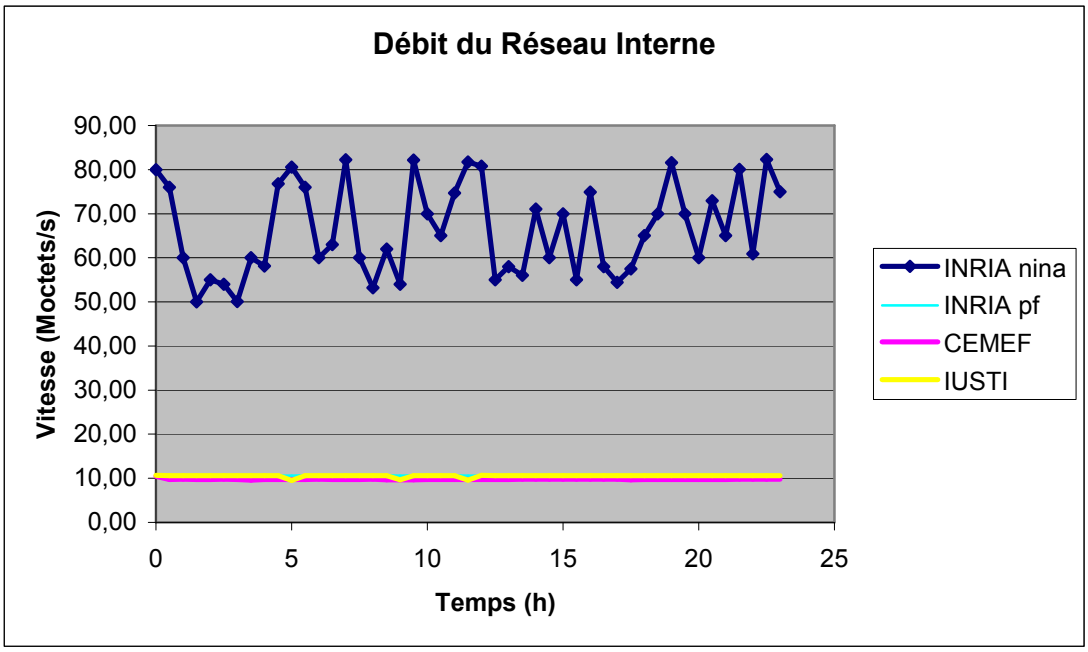


Figure 3 – débit des réseaux internes à chaque cluster en fonction du temps

D'autres tests ont été exécutés de la même façon en utilisant cette fois-ci 4 processeurs sur chaque cluster. Aucune différence notable n'y apparaît en comparaison avec les résultats précédents. Ceci montre que les débits des réseaux internes mesurés ici sont indépendants du nombre de processeurs utilisé. Voici les moyennes des débits (converties en Mégabits/s pour pouvoir les comparer au mesures théoriques) et des écarts types observées dans tous les tests effectués :

LAN	nina	pf	pf – nina	cemef	iusti
Mb/s Théorique	1000.00	100.00	100.00	100.00	100.00
Mb/s Réel	509.69	86.32	89.25	84.05	86.65
Ecart Type %	28.01	0.26	0.25	0.71	0.59

Tableau 3 – Débit moyen des réseaux internes

En comparant les débits théoriques et les débits réels observés, on s'aperçoit que le réseau interne au cluster INRIA-nina est assez loin de ce qu'il peut être (presque 50%), mais que les autres sont plus proches de leur valeur maximale (environ 90%). Nous remarquons aussi que le débit du réseau entre les processeurs nina varie beaucoup : son écart type moyen est de 28%, alors que les autres débits ont un écart type moyen de 0.5%.
Il est assez difficile d'expliquer les mauvais résultats du réseau entre les nina. Le fait que le cluster de l'INRIA ait une adresse IP publique pourrait éventuellement expliquer cela. Mais, si c'était le cas, le réseau entre les processeurs pf (qui ont eux aussi une adresse IP publique), serait alors dans le même cas. Cette explication ne parait donc pas satisfaisante. L'usage du réseau par les autres utilisateurs du cluster INRIA-nina semble alors la raison la plus probable qui puisse influencer de la sorte l'état du réseau interne.

4.2.3. Débit du réseau inter site (entre chaque site)

Sur la figure *(4)*, nous avons le débit du réseau externe entre chaque site en Mo/s en fonction du temps. Ces résultats ont été obtenus par des tests de Performance toutes les 30 min pendant 24h avec 1 processeur sur chaque site (3×1).

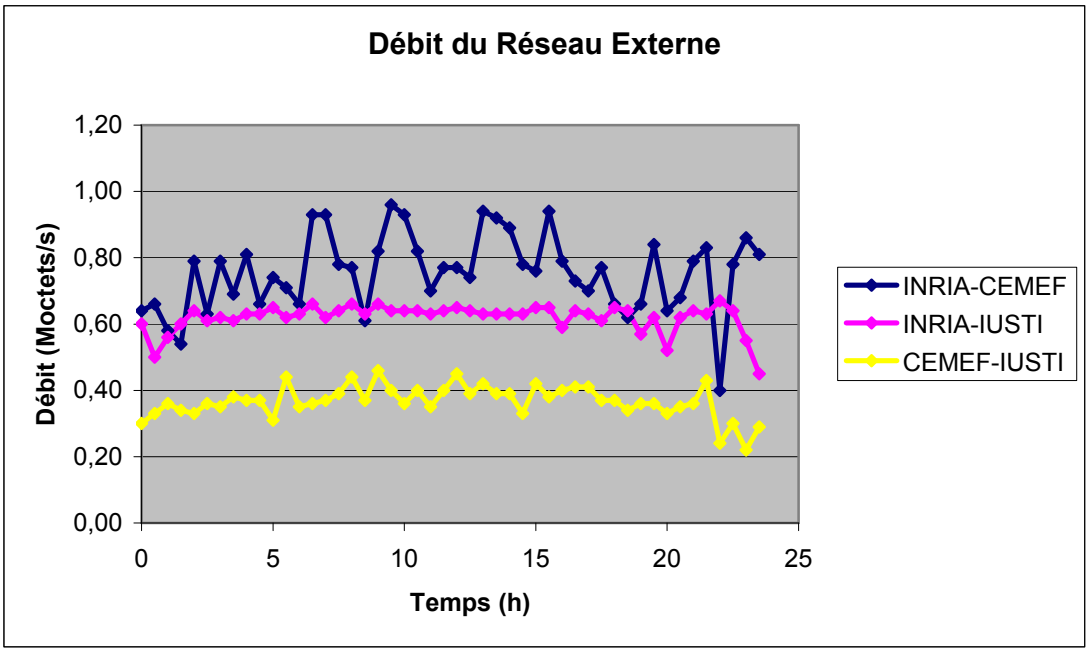


Figure 4 – Débit moyen des réseaux inter sites

Il nous faut maintenant vérifier si le fait d'augmenter le nombre de processeurs influe sur la vitesse du réseau inter site : avoir plus de processeurs augmente les communications et pourrait saturer les réseaux de la grille. Ainsi, nous avons fait d'autres mesures du débit inter site avec 2 puis 4 processeurs par site (3×2 et 3×4). Les résultats obtenus ne présentent pas de différence significative par rapport au cas 3×1. En moyenne, nous avons :

Réseau Externe	nina – cemef	nina – iusti	cemef – iusti
Mb/s	7.21	5.03	3.69
Ecart Type %	15.00	7.91	13.98

Tableau 4 – Débit moyen des réseaux inter sites

Les variations de débit sont assez importantes : leur écart type moyen peut aller jusqu'à 15%. Ce n'est pas étonnant, vu que les trois sites de la grille sont reliés par une connexion internet, dont le débit dépend forcément de nombreux paramètres aléatoires. En plus d'avoir des variations importantes, cette connexion à en moyenne une performance très faible par rapport aux débits constatés au sein des différents clusters. Ceci a pour conséquence d'avoir une grille de calcul très hétérogène dans ses performances de réseau, puisque le rapport entre le débit le plus rapide (509.69 Mb/s à l'intérieur du cluster INIRA-nina) et le plus faible (3.69 Mb/s entre le CEMEF et l'IUSTI) est d'environ 130.

4.2.4. Évaluation des fréquences de variation des performances

Dans tous les tests effectués, la vitesse des processeurs – autre que nina – ne varie que très peu (écart type de moins de 0.5%). Mais les nina varient un peu plus (environ 3%) : leurs puissances ont tendance à s'écrouler pendant une période très courte (inférieure à 5 minutes). Quant au débit des réseaux, leurs variations semblent complètement aléatoires. Du moins, si l'on devait leur donner une période caractéristique, elle serait de l'ordre de la dizaine de minutes.

4.2.5. Caractère hétérogène de la grille

Les résultats présentés sur la figure (*5*) montrent que la grille de calcul a des ressources de calcul très hétérogène : à la fois pour le débit de réseau et pour la vitesse des processeurs. Donc la répartition des tâches doit en tenir compte. Les réseaux sont séparés en 3 grands groupes :

- le réseau interne au cluster INRIA-nina à environ 510 Mb/s,
- les autres réseaux internes à environ 90 Mb/s, et
- les réseaux inter site à environ 5 Mb/s.

Les processeurs sont eux divisés en 2 groupes :

- INRIA-nina et IUSTI à plus de 125 Mflops/s, et
- INRIA-pf et CEMEF à plus de 44 Mflops/s.

La figure *(5)* résume l'architecture réelle de la grille. Puisque certaines des quantités mesurées ne sont pas tout à fait constantes au cours du temps, ce sont les valeurs moyennes qui sont affichées ici. Les frontales par lesquelles passent les communications inter sites sont représentées par des carrés. Les nœuds reliés entre eux en forme d'étoile représentent les différents clusters. Enfin, les connexions Internet inter sites sont en noir. Une comparaison peut être facilement établie avec le même schéma montrant l'architecture théorique de la grille (figure *1*).

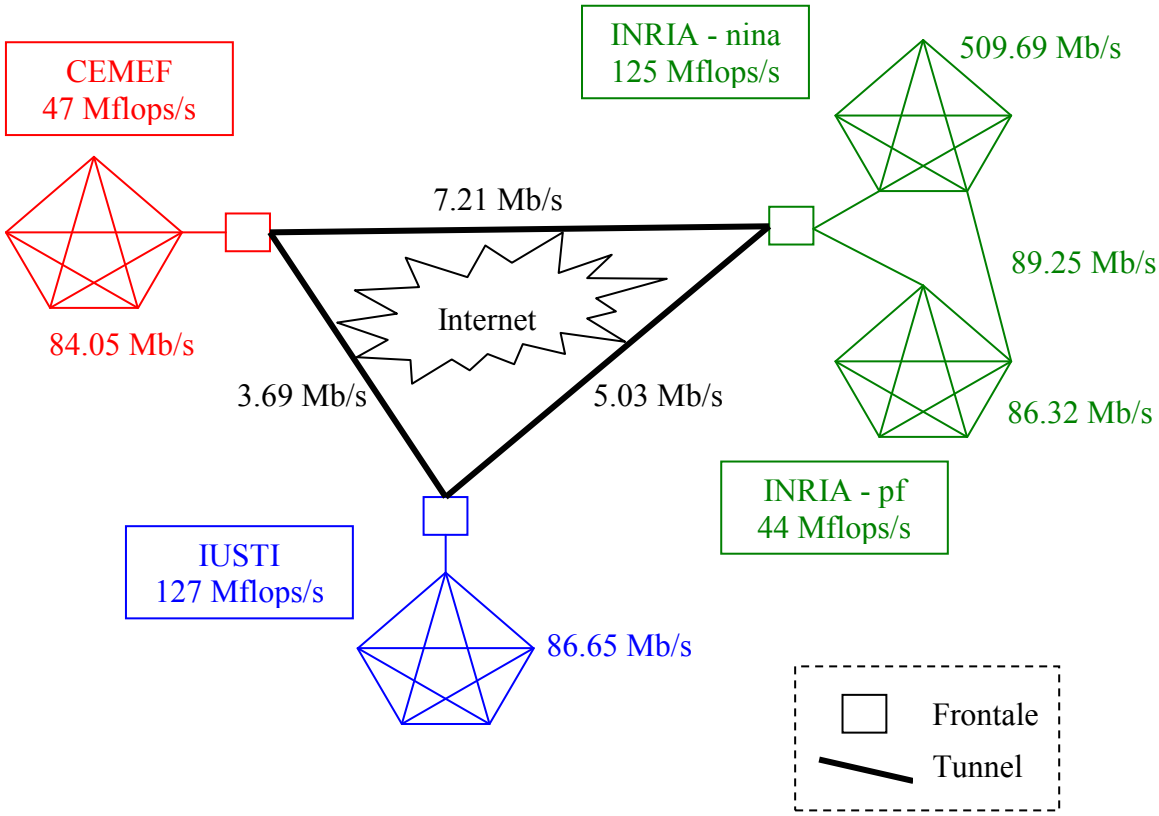


Figure 5 – Architecture réelle de MecaGrid

4.2.6. Globus vs Non-Globus

Il est légitime de se demander si l'ajout de middleware Globus joue un rôle dans les performances des codes de calculs. Afin de mettre en évidence une éventuelle influence, un calcul est exécuté plusieurs fois sur les mêmes ressources, avec, puis sans Globus. Pour cela, il faut non seulement utiliser un seul et même cluster, mais aussi avoir des conditions identiques, c'est-à-dire le même nombre de nœud et de processeurs par nœud. Nous avons choisi d'utiliser le cluster INRIA-nina et 16 nœuds (un processeur par nœud). Le programme doit être compilé deux fois pour créer un exécutable qui s'appuie sur Globus, et un autre qui ne l'utilise pas ; tout en faisant bien attention à appliquer les mêmes options de compilation. Dans la pratique, comme le montre le tableau *(5)*, les temps d'exécution des programmes avec et sans Globus ne montrent pas de différences notables.

1x3	Globus	Non Globus
Temps (s)	455.57	461.07

Tableau 5 – Temps d'exécution avec et sans Globus

Cette application est réalisée avec notre code de calcul et les méthodes discutées dans cette thèse. Pour cela, trois mêmes processeurs du cluster cemef ont été choisis pour les deux exécutions – avec Globus et MPICH-G2, puis sans Globus, mais avec MPI. Sans répéter ce test sur tous les différents clusters, cette hypothèse est supposée vraie pour MecaGrid dans sa globalité : Globus et MPICH-G2 ne détériorent pas les performances de calculs.

4.2.7. Importance des communications par rapport aux calculs

Un sujet sur lequel nous n'avons pas encore d'indication bien précise est l'importance que peuvent avoir les communications par passage de message par rapport au travail de calcul des processeurs. Malheureusement, notre logiciel ne contient pas à l'heure actuelle d'outil pour mesurer le temps passé dans les communications afin de le comparer à celui passé dans les calculs. Cela permettrait d'acquérir une meilleure compréhension sur la nature des calculs, et de mieux comprendre quelle est l'utilisation la plus propice de MecaGrid. Cependant, dans le cadre de sa recherche au sein de l'INRIA-Sophia, Steven Wornom trouve des résultats très intéressants avec un autre code de calcul. Ce logiciel, nommé AERO, simule des écoulements monophasiques et biphasiques d'aérodynamiques externe et interne avec des méthodes de type Volumes Finis. Il est développé par l'université du Colorado en collaboration avec l'INRIA [Guillard, 1994]. A titre indicatif, voici ses résultats sur le rapport de temps entre les communications et les calculs lors d'un exécution de AERO sur 8 processeurs.

	nina	pf	iusti	cemef
Calculs (s)	331.4	564.7	341.2	759.6
Communications (s)	10.6	25.7	16.7	47.9
Com/Calcul	0.03	0.05	0.05	0.06

Tableau 6 – Temps de calcul et de communication sur clusters isolés

Le tableau *(6)* montre les temps de calcul et de communication sur chacun des quatre clusters de MecaGrid individuellement. Pour des clusters isolés, les rapports entre les communications et les calculs sont très petits, ce qui montre que le travail des processeurs est largement supérieur aux communications inter processeurs.

	nina-pf	nina-cemef	nina-iust	iusti-cemef
Calculs (s)	401.8	550.8	640.7	970.6
Communications (s)	25.6	173.1	253.5	315.0
Com/Calcul	0.06	0.31	0.40	0.32

Tableau 7 – Temps de calcul et de communication sur MecaGrid

Après avoir vu le cas des clusters individuels, ce sont différentes paires de clusters qui sont étudiées sur le tableau *(7)*. L'hétérogénéité de MecaGrid fait que les résultats sont très différents de ceux du tableau *(6)* : les rapports communications/calculs sont à la fois plus importants, mais aussi plus diverses entre eux. Cela montre que, dans certains cas, les communications peuvent avoir une très grande importance vis-à-vis des calculs (jusqu'à 40%). Le meilleur rapport correspond à la paire nina-pf, ce qui est tout fait normal, vu qu'ils font partie tous les deux du site de l'INRIA Sophia. Ainsi, leurs communications passent par un réseau local (LAN) et non par le VPN et les frontales de MecaGrid. Le rapport le plus médiocre est pour nina-iusti bien qu'ils ne fassent pas partie des clusters les moins performants, bien au contraire : ce sont eux qui montrent les meilleurs performances individuelles. C'est justement parce que leurs processeurs sont puissants qu'ils passent plus de temps que les autres dans les communications par rapport aux calculs.

5. Techniques d'optimisations

Actuellement, une des principales déceptions survenues après les premières utilisations de la grille porte sur les performances obtenues par les applications parallèles. Ces résultats décevants proviennent de deux raisons : premièrement, les ressources de calcul étant très hétérogènes, la masse de calcul attribuée à chaque processeur peut être sévèrement déséquilibrée ; et deuxièmement, le débit des réseaux inter sites (qui forment les connections inter clusters) est beaucoup plus lent que celui des réseaux interne à chaque cluster (LAN). L'analyse des performances réelles de MecaGrid montre qu'il y a un facteur 2.87 entre les processeurs les plus puissants et les moins puissants, et un facteur de 138 entre les débits les plus lents et les plus rapides. Cette hétérogénéité extrême pose donc des problèmes de performance, et nous cherchons ici des techniques d'optimisation qui permettent de contre balancer ces inconvénients. De fait, cette problématique est beaucoup plus complexe que dans le cas d'une machine parallèle standard : pour obtenir des résultats satisfaisant sur une grille, il faut non seulement répartir au mieux la masse de calcul, mais aussi diminuer les communications sur les réseaux pénalisants au profit des meilleurs réseaux disponibles.

5.1. Instrumentation de PETSc

La librairie PETSc utilisée par la librairie CimLib sur la grille de calcul possède un outil d'instrumentation intéressant sur les performances parallèles. Quelques informations importantes fournies par cet outil *-log_summary* sont présentées ici. Par la suite, nous l'utiliserons pour évaluer une méthode pouvant améliorer les performances de la grille.

PETSc crée un fichier log qui présente un résumé des performances de temps (en seconde), de vitesse de calcul (en Flops/sec) et d'activité de communication entre processeurs (telle que le nombre et la taille des messages lancés). Nous pouvons comparer le travail fait par chaque processeur afin, par exemple, de vérifier la répartition de la masse de calcul, ou de quantifier les communications. Ensuite, des indications sur la mémoire utilisée, et sur la création / destruction des différents objets sont présentées. Enfin, le fichier log donne le temps moyen pour envoyer un message de taille zéro (appelé temps de latence), ainsi que le temps moyen passé dans MPI_Barrier(). Cette fonction MPI ordonne la synchronisation de tous les processeurs ; c'est-à-dire que plus la charge de calcul est bien répartie, plus ce temps est faible. Pour conclure, cette instrumentation offerte par PETSc donne plusieurs informations très importantes sur les performances parallèles du code, mais ne donne pas le temps réels passé dans les communications. Donc, le seul moyen que nous avons en ce moment pour mesurer les temps de communications reste l'outil Performance présenté précédemment.

5.2. Analyse des biprocesseurs

Les clusters de l'INRIA et du CEMEF sont constitués de biprocesseurs, c'est-à-dire que l'on a la possibilité de placer un ou deux processus sur chaque nœud lors de l'exécution. Ces deux utilisations sont très différentes, et le but est ici de déterminer laquelle est la plus judicieuse pour exécuter des codes éléments finis sur la grille de calcul.

5.2.1. Débit du réseau entre processeurs du même nœud

Tout d'abord, on remarque que le débit du réseau entre 2 processus d'un même nœud est bien supérieur au débit du réseau entre 2 nœuds distincts :

- INRIA (nina) :

nina07-nina10 : 509.69 Mb/s
nina10-nina10 : 2014.27 Mb/s

- INRIA (pf) :

pf12-pf10 : 86.32 Mb/s
pf10-pf10 : 616.65 Mb/s

- CEMEF :

node25-node26 : 84.05 Mb/s
node25-node25 : 744.32 Mb/s

La vitesse des communications constitue un point positif de l'utilisation des biprocesseurs.

5.2.2. Vitesse des processeurs d'un même nœud

Dans la pratique, on observe que la vitesse des processeurs est moins importante lorsque les nœuds sont partagés (c'est-à-dire lorsqu'il y a 2 processus par nœud). Cependant, l'outil Performance ne montre pas une telle différence de vitesse, mais c'est lors d'une exécution d'un code éléments finis que l'on peut l'observer. En effet, la résolution d'un système linéaire sur un nœud partagé se fait en 223.42 secondes, alors que cette même résolution se fait en 176.34 secondes sur un nœud libre. La vitesse des biprocesseurs serait donc plus lente lorsqu'ils sont partagés (environ 79% de leur vitesse normale). Ceci pourrait être expliqué par le fait que les 2 processeurs d'un nœud utilisent le même bus : l'utilisation intensive des 2 processeurs peut donc saturer leur bus commun d'accès à la mémoire.
Après avoir vu dans a) le côté positif des biprocesseurs, b) nous montre un désavantage important de cette utilisation. Mais alors, quelle est la meilleure solution ? Est-il plus avantageux d'utiliser les 2 processeurs d'un nœud en même temps ou pas ?

5.2.3. Conclusion sur l'utilisation des biprocesseurs

Afin de trancher entre les 2 utilisations possibles des biprocesseurs (1 processeur par nœud, ou 2 processeurs par nœud), nous exécutons un code éléments finis sur la grille des 2 manières différentes. Celle qui aura pris le moins de temps – en conciliant la vitesse des processeurs et la vitesse des communications – sera retenue.
En utilisant 2 processeurs sur chaque site contenant des biprocesseurs (2 nina, 2 pf et 2 CEMEF), nous notons les temps de résolution d'un système linéaire :

- Avec 1 processeur par nœud : 664.19 secondes
- Avec 2 processeurs par nœud : 715.82 secondes

La solution optimale est donc de ne pas utiliser les 2 processeurs d'un biprocesseur : il est plus avantageux de lancer l'exécution avec un seul processeur par nœud.

5.3. Études des préconditionneurs

Une idée pour améliorer les performances sur la grille de calcul consiste à faire varier le degré k du préconditionneur ILU(k) pour le système linéaire avec PETSc. Jusqu'à présent, nous utilisions le préconditionneur ILU(k) avec un degré k = 0. Le nombre permis d'entrées non nulles dans la matrice dépend directement de ce degré k : plus k est élevé, plus le nombre d'entrées non nulles est élevé. Avoir un k > 0 permet d'avoir une masse de calcul supérieure sur chaque processeur, tout en diminuant les communications entre les processeurs et le nombre itérations lors de la résolution du système linéaire. Compte tenu de l'architecture de la grille et du coût élevé des communications, cette méthode parait être adaptée à notre problème. Les résultats suivants ont été obtenus sur la grille de calcul 3×2 (avec 2 processeurs sur chaque site) avec un maillage à 65000 nœuds, puis sur un cluster isolé (avec 6 processeurs) :

Nombre d'itérations	ILU (0)	ILU (1)	ILU (2)	ILU (3)
MecaGrid	100	60	46	38
Cluster	100	61	44	37

Temps de résolution	ILU (0)	ILU (1)	ILU (2)	ILU (3)
MecaGrid	100.00	60.31	65.44	86.95
Cluster	100.00	97.21	126.27	203.53

Tableau 8 – Temps d'exécution relatifs en fonction du préconditionneur

Les figures suivantes montrent les résultats du tableau *(8)* sur des graphes pour bien se rendre compte que le nombre d'itérations nécessaire diminue quand k augmente, et que les temps de résolution évoluent différemment en fonction de k sur une grille de calcul et sur un cluster isolé. Ainsi, il est clair qu'utiliser ILU(1), voire même ILU(2) et ILU(3), fait gagner du temps sur la grille de calcul en diminuant les communications ; alors que sur un cluster seul, cela n'apporte pas d'améliorations notables. Nous concluons donc que les communications entres processeurs n'est que très peu pénalisante sur un cluster seul, comparé à MecaGrid.

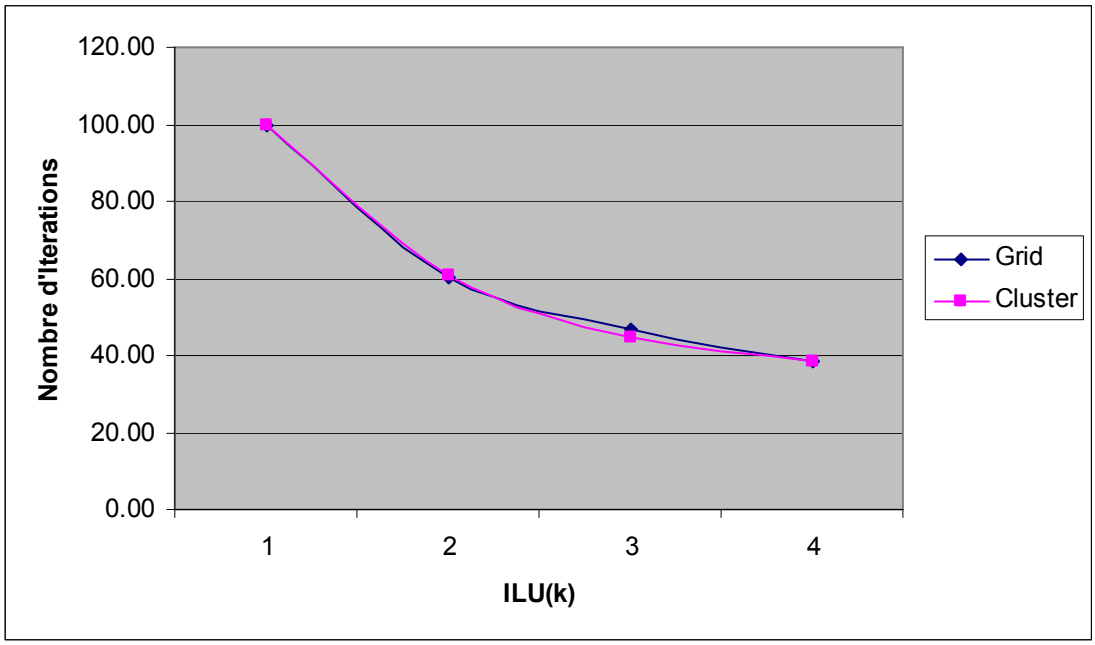


Figure 6 – Nombre d'itérations nécessaires à la convergence en fonction du préconditionneur

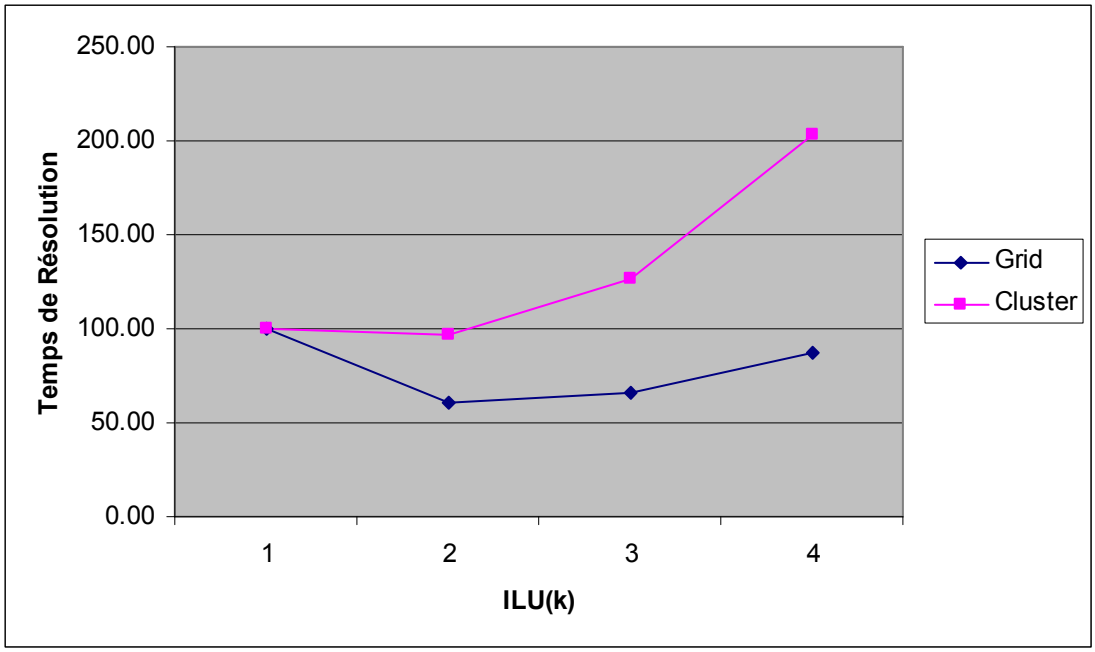


Figure 7 – Temps d'exécution en fonction du préconditionneur

Avec l'instrumentation de PETSc présentée précédemment, nous pouvons mesurer le nombre de messages MPI lancés, la taille maximale de ces messages MPI, le nombre de réductions globales MPI, et la mémoire utilisée par les processeurs :

ILU(k)	k = 0	k = 1	k = 2	k = 3
Nb MPI messages	1.936E+03	1.186E+03	9.000E+02	7.810E+02
Taille MPI messages	5.720E+07	3.940E+07	3.300E+07	2.978E+07
Nb MPI réductions	2.027E+02	1.277E+02	1.000E+02	8.717E+01
Mémoire (octets)	4.310E+07	7.639E+07	1.350E+08	2.119E+08

Tableau 9 – Instrumentation de PETSc en fonction du préconditionneur

Le nombre d'itérations lors de la résolution du système linéaire diminue fortement en fonction de k. Par contre, le temps de résolution est minimum pour k = 1, mais remonte quand k > 1. ILU(1) permet donc d'avoir un temps de résolution d'environ 60% du temps obtenus avec ILU(0), ce qui est loin d'être négligeable. Nous voyons d'ailleurs que, comme nous le souhaitions, les communications entre processeurs diminuent quand k augmente.

Cependant, cette méthode entraîne automatiquement une augmentation considérable de la place mémoire utilisée sur chaque processeur. En effet, entre ILU(0) et ILU(1), elle augmente d'un facteur 1.77. Ceci pourrait poser un problème important si le maillage contient beaucoup de nœuds, car la place mémoire utilisée pourrait alors devoir sortir de la mémoire cache, ce qui ralentit terriblement la vitesse de calcul du programme.

En résumé, entre ILU(0) et ILU(1), nous avons :

- un facteur 0.61 sur le nombre de messages MPI
- un facteur 0.69 sur la taille maximale des messages MPI
- un facteur 0.63 sur le nombre de réductions globales MPI
- un facteur 1.77 sur la quantité de mémoire utilisée
- et un facteur 0.60 sur le temps de résolution.

5.4. Particularités du processus maître

Lorsque l'on parle d'exécution parallèle avec la librairie MPI de passage de messages, l'idée de processeur maître est très importante. C'est lui qui lit et écrit les données sur le disque dur, qui initialise les communications globales, et qui réalise les tâches qui ne peuvent pas être parallélisées. De ce fait, il doit souvent accéder à une mémoire virtuelle plus importante que les autres. Un exemple simple est celui du partitionneur non parallélisé : le processeur maître (dit le processeur zéro) lit et stock la totalité maillage, affiche les informations à l'écran, communique avec les autres processeurs, puis échange des données pour créer une partition par processus. Dans ce cas, il a beaucoup plus de travail à faire, de messages à lancer et recevoir, et de données à stocker que les autres. C'est donc sans surprise que, dans la pratique, le choix du processeur maître montre une influence non négligeable sur les performances. Nous étudions ici les meilleures manières qui optimisent l'utilisation de la grille de calcul.

Puisque le processeur zéro est susceptible d'effectuer une quantité de travail supérieure, le but est de diminuer le temps pendant lequel il est le seul calculer, car pendant ce temps, les autres peuvent attendre qu'il finisse avant de pouvoir continuer. C'est pourquoi, il est judicieux de placer le processeur maître sur un nœud performant où la puissance du processeur est grande par rapport aux autres au sein de la grille. Ensuite, il faut que ce processeur maître ait un accès suffisant à la mémoire virtuelle car c'est lui qui comporte le grand risque de swap (dépassement de la RAM). Pour cela, les nœuds qui contiennent le plus de RAM sont privilégiés ; mais aussi, dans le cas de cluster à biprocesseurs, il est mieux d'avoir le processeur maître seul sur un nœud pour qu'il n'ait pas à partager la mémoire disponible avec un autre processus.

Sur le point de vue des communications, il y a encore plus de paramètres à considérer. Tout d'abord, les messages pouvant s'effectuer au sein d'un cluster (à travers un LAN, avec un nœud du même cluster) ou par internet (suivant un tunnel du VPN, avec un nœud d'un autre site), il faut non seulement choisir un cluster au réseau interne performant, mais aussi un site aux connexions internet rapide. Ensuite, dans le VPN mis en place, tous les messages sont pris en charge par les machines frontales : elles encapsulent les messages, les cryptent, les compressent, les envoient, les reçoivent, les décompressent, les décryptent. Il est donc important que le processeur maître soit placé sur un cluster dont le site contient une frontale performante (de part son processeur, sa mémoire, sa connexion…). En conséquence, les communications opérées par le nœud maître en grand nombre seront les moins pénalisantes possible.

Dans la pratique, des différences de performance considérables sont observables en fonction de la position du nœud maître. Ils sont montrés ici les temps obtenus lors de la résolution d'un système linéaire sur deux clusters et quatre processeurs par cluster. Les paires de cluster nina-pf, nina-cemef et iusti-cemef sont testés en changeant la localisation du processus zéro. Le temps d'exécution sur pf-nina (avec le nœud maître sur le cluster pf) est 140% celui obtenu sur nina-pf (avec le nœud maître sur le cluster pf). Dans ce cas, c'est la puissance du processeur maître et la vitesse de son LAN qui cause ce résultat puisque aucune frontale n'est utilisée. De la même manière, le temps d'exécution sur cemef-nina est 270% celui obtenu sur nina-cemef. Ici, tous les paramètres discutés précédemment influent, et les résultats sont spectaculaires. Enfin, le temps d'exécution sur cemef-iusti est 115% celui obtenu sur iusti-cemef.

5.5. Méthode de partitionnement optimisé

5.5.1. Enjeux

Les applications en calculs de mécanique des fluides s'appuient généralement sur des données de grande taille correspondant au maillage spatial de la cavité (domaine de calcul). Lors d'une exécution en parallèle, ce sont ces données qui doivent être partitionnées sur tous les processeurs pour répartir la masse de calcul. Pour chaque itération (ou incrément en temps), des opérations indépendantes sont effectuées en parallèle, et donc, des communications entre les données distribuées sur toutes les machines doivent s'opérer. La problématique de la répartition équitable de la masse de calcul et de la quantité des données est très importante dans le contexte du calcul parallèle, mais est primordial lorsque l'on parle de grille de calcul. C'est essentiellement le caractère hétérogène des grilles qui augmente l'importance que l'on attache à ce sujet. L'enjeu principal est de minimiser les temps d'attentes que peuvent avoir les processeurs les uns envers les autres, et les temps passés dans les communications inter processeurs tout au long de la simulation. Dans le cas des applications basées sur des maillages, c'est le partitionnement du maillage lui-même qui accomplit la répartition des données et de la masse de calcul.

La libraire CimLib que l'on utilise pour simuler des écoulements de fluides hétérogènes sur MecaGrid est basée sur des méthodes de type éléments finis qui s'appuient sur des maillages de tétraèdres. Le nombre d'inconnues à résoudre dans les systèmes linéaires est proportionnel au nombre de nœuds du maillage, puisqu'elles ont toute une interpolation continue P1. Plus précisément, et pour chaque incrément de temps, 4 inconnues par nœud sont résolues pour la mécanique (v_x, v_y, v_z, p), et 1 inconnue par nœud est résolue pour le transport de la matière (α). Cela montre bien que répartir la masse de calcul (et donc le nombre d'inconnues à résoudre) correspond parfaitement au nombre de nœuds que contient la partition (partie du maillage) envoyée sur chaque processeur. Chaque processeur calcule l'écoulement dans sa portion du maillage, tandis que les communications sont effectuées lors du produit matrice - vecteur est dépendent du nombre de nœuds à l'interface des partitions.

Le fait de vouloir un partitionnement très optimisé est d'autant plus important que les méthodes numériques choisies dans CimLib ont une approche Eulérienne : elles ne nécessitent pas de remaillage pendant la simulation. De ce fait, c'est le même maillage qui est utilisé tout au long du calcul. De plus, l'étude de performance réelle de la grille montre que celle-ci évolue d'une façon non négligeable en fonction du temps. Une question très importante se pose donc ici : faut-il repartitionner le maillage pendant la simulation pour tenir compte des évolutions des performances ? En effet, si une masse de calcul bien précise a été attribuée à un processeur en tenant compte de sa puissance calculée avant l'exécution du programme, mais que celle-ci se met à changer de façon importante, il est possible qu'il faille modifier la répartition des tâches initialement prévue. Et il en est de même pour le débit des réseaux. De cette façon, un partitionnement très bien optimisé au début de la simulation peut éventuellement se transformer en un partitionnement de mauvaise qualité qui ralentit les calculs, dès lors que les performances de la grille évoluent d'une trop grande manière. Afin de donner une réponse satisfaisante à cette question primordiale, il faut regarder de près la façon dont varient la puissance de calcul des processeurs et le débit des réseaux de MecaGrid. Les résultats observés précédemment à ce sujet montrent que la vitesse des processeurs ne varient pas d'une manière très importante : il y a un écart type d'environ 1% autour d'une valeur

moyenne. Par contre, les variations aléatoires qui se produisent dans le débit des réseaux (à la fois internes et externes aux clusters) peuvent avoir jusqu'à un écart type de 30% autour d'une valeur moyenne, ce qui n'est absolument pas négligeable. Seulement, il semble impossible de pouvoir attribuer une période d'oscillation, tellement les évolutions sembles aléatoires. Sinon, elle serait sûrement de l'ordre de quelques dizaines de minutes, ou de une heure. Maintenant, il faut comparer cette durée avec celle que prend un repartitionnement. Pour en avoir une idée, cette opération prend environ 10 minutes sur la grille avec 2 processeurs sur chaque site pour un maillage à 65000 nœuds ; et cela peut être bien plus long lorsque le maillage est de grande taille, et qu'un grand nombre de processeurs est utilisé. Ainsi, la durée du repartitionnement n'est pas être significativement inférieure aux périodes de variation de la puissance de calcul. Autrement dit, si on avait recours au repartitionnement, on serait amené à répéter cette opération très souvent, et le gain de temps sur les calculs serait bien inférieur au temps passé dans les recherches d'optimisation. En conclusion, il n'est pas nécessaire de repartitionner le maillage pendant la simulation. De plus, l'ensemble des quantités mesurées reste sensiblement dans la même zone autour de valeurs moyennes, qui elles, restent inchangées au cours du temps. Ces moyennes peuvent donc servir de référence à l'exécution du partitionnement, avant le début des calculs.

5.5.2 Méthode de partitionnement

La méthode de partitionnement de maillage présentée ici constitue une amélioration à celle établie lors du projet DRAMA [Basermann, 2000]. Plus précisément, elle apporte une généralisation au contexte hétérogène que l'on a sur une grille de calcul. L'algorithme d'optimisation des partitions est brièvement présenté ; puis, plusieurs techniques d'utilisation sont passées en revue, incluant des perspectives pouvant améliorer le lien entre le partitionneur et les solveurs numériques.

Pour partitionner un maillage destiné à une exécution parallèle sur une machine homogène, il suffit de diviser tout simplement le maillage de façon à ce que chaque processeur contienne le même nombre de nœuds. Il faut aussi faire en sorte que les communications inter processeurs – qui correspondent directement à l'interface entre les partitions du maillage – soient minimisées. Mais, dans le contexte des grilles de calcul, le partitionnement représente une problématique bien plus complexe. En effet, les grandes différences qui existent dans les vitesses des processeurs, les débits des réseaux et les quantités de mémoire disponibles, engendrent une importante hétérogénéité des ressources de calcul difficile à négocier. Pour obtenir des résultats satisfaisant sur une grille, il faut donner plus de travail à un processeur rapide qu'à un autre plus lent, mais aussi diminuer les communications sur réseaux pénalisants au profit des meilleurs réseaux disponibles.

Il faut faire attention à ce que le partitionnement ne crée pas de partitions trop grosses pour la mémoire des processeurs qui les utilisent. En effet, un dépassement de la mémoire entraîne un swap de la RAM, ce qui ralentit extrêmement les calculs avec des accès au disque trop fréquents. C'est une raison supplémentaire pour n'utiliser qu'un seul proc par nœud dans les clusters à biprocesseur et ainsi disposer d'une quantité de RAM supérieure. Par exemple, nous considérons que dans la pratique, la limite supérieure que peut avoir une partition de maillage serait autour de 50000 nœuds pour 256 Mo de RAM. Cette estimation est extrêmement dépendante des types de solveur appelés pendant la simulation. Dans notre cas, seuls des solveurs de type P1 (5 inconnues par nœuds du maillage au total et par incrément de temps) sont utilisés. Avec d'autres solveurs de type P0 (où les inconnues sont sur les

éléments), il y a bien plus d'inconnues à résoudre, et donc, des partitions de maillage plus petites doivent être produites.

5.5.3. Notations et définitions

Soit $G = G(V, E)$ un graphe non orienté avec des sommets *V* et des arrêtes *E*. Les arrêtes représentent des dépendances dans les données du maillage. Dans ce modèle, on considère que le graphe représente le maillage : les sommets représentent les nœuds et les éléments du maillage.
Ainsi, partitionner le maillage revient à partitionner son graphe correspondant. Dans la littérature, [Shirazi, 1995] traite du cas sur des machines parallèles à architecture homogène. Le problème du partitionnement homogène sur *n* processeurs est divisé en deux parties : définition des *n* sous graphes, et association de chaque sous graphe avec un processeur.
Sur grille de calcul, l'architecture est hétérogène pour ce qui est du réseau et des processeurs. En conséquence, il nous faut considérer ces paramètres pour définir les sous graphes, et la deuxième étapes doit être fusionnée avec la première pour associer correctement les sous graphes avec leur processeur.
Pour évaluer la qualité d'une partition π, il est nécessaire de définir deux modèles qui servent à estimer un temps caractéristique induit par une simulation parallèle. Pour tous les *n* processeurs, le temps de calcul est défini par :

$$t_p \equiv \frac{d_p}{s_p} \qquad (2)$$

où d_p est la masse de calcul du processeur *p* (exemple : nombre de nœuds du maillage associés au processeur *p*, car les inconnues à résoudre sont situées aux nœuds) et s_p est la vitesse mesurée du processeur *p*. Le temps passé dans les communications est définie par :

$$c_p \equiv \sum_{i \neq p} \frac{d_{pi}}{v_{pi}} \qquad (3)$$

où d_{pi} est le volume des communications entre les processeurs *p* et *i* (exemple : nombre de faces dans le maillage coupées par la partition) et v_{pi} est le débit mesuré du réseau reliant *p* et *i*.
Avec ces deux modèles, une fonction coût de la partition est définie par :

$$\Phi = \beta T + (1 - \beta) C \qquad (4)$$

où $T = (t_0, t_1, ..., t_{n-1})^T$ et $C = (c_0, c_1, ..., c_{n-1})^T$. Quant à β, c'est un paramètre appartenant à $[0,1]$ qui sert de poids entre *T* et *C*. Autrement dit, il détermine l'importance que l'on attache à la puissance de calcul des processeurs vis-à-vis du débit des communications inter processeurs. Aucune technique n'est aujourd'hui disponible pour choisir au mieux la valeur de β, mais c'est seulement dans la pratique, avec des résultats expérimentaux, qu'on y parvient. De cette manière, toute partition peut être évaluée en calculant cette fonction coût.
Maintenant, le problème du partitionnement revient à trouver la partition associée au graphe qui minimise la fonction coût Φ.

5.5.4. Méthode locale du repartitionnement parallèle

Dans cette partie, la stratégie de minimisation de la fonction coût est brièvement décrite. La méthode est itérative, locale et parallèle.
On introduit un nouveau graphe qui lui représente l'architecture de la machine de calcul appelé le graphe d'architecture. Ses sommets représentent les processeurs, et les arrêtes représentent les connexions réseaux entre les nœuds.
Dans le cadre de cette méthode d'optimisation de partition, il faut obligatoirement commencer par se munir d'une partition valide avec une qualité quelconque. C'est pourquoi, avant l'exécution de cette méthode, on crée une partition homogène sans tenir compte de l'architecture de la machine. En considérant donc une partition initiale π_0, la première étape consiste à déterminer les meilleures paires de processeurs qui vont optimiser la partition. Pour cela, nous développons un critère, fonction d'une partition π_i et des caractéristiques architecturales (s_p et v_{pi}), qui permet de comparer toutes les paires possibles. Une fois que toutes les paires séparées sont formées, la partition, limitée à chaque paire, est optimisée localement : les sommets (nœuds et éléments) sont transférés d'un processeur à un autre, tant que la partition limitée est meilleure. Un autre critère a été développé pour déterminer une priorité de transfert entre les sommets. Ainsi, les deux dernières étapes sont répétées tant que l'on a la possibilité d'améliorer de façon globale la partition.

5.6. Utilisations du partitionneur

A l'heure actuelle, il n'y a d'autre solution que de déterminer à l'avance – avant de commencer une simulation – les ressources sur lesquelles le calcul va être lancé. Cela correspond à choisir les clusters, puis le nombre de processeurs par cluster. Une fois ce choix fait, le partitionneur doit répartir le maillage en tenant compte des caractéristiques moyennes de ces ressources. Enfin, le calcul peut être exécuté en appliquant bien les correspondances entre les partitions et les processeurs. Cette technique « manuelle » a plusieurs inconvénients : elle implique au moins deux requêtes (une pour partitionner, une pour lancer la simulation), elle oblige à connaître les caractéristiques moyennes de toutes les ressources, à devoir choisir le nombre de processeurs par cluster au risque de ne plus en avoir l'accès au moment de l'exécution… Quelques perspectives peuvent donc être dressées à ce sujet.

La deuxième technique envisageable est de coupler l'outil de mesure de performance, le partitionneur, et le code de calcul (CimLib). Cela permettrait d'exécuter une seule requête, et sur des ressources quelconques. Après avoir fait un test de performance, le partitionneur pourrait répartir le maillage en considérant les performances des nœuds et des réseaux utilisés. Enfin, la simulation serait prête pour commencer directement après et sur les mêmes nœuds, sans qu'une nouvelle requête ne soit nécessaire. Cette méthode « automatique » ne demande pas une connaissance au préalable des performances des ressources, mais elle a le désavantage de présenter le choix entre mesurer rapidement des performances approximatives, ou rallonger la durée de ces mesures pour pouvoir connaître des performances plus précises (plus proches des valeurs moyennes réelles). Ce dernier point implique une quantité de calcul importante et superflue, dans le sens où elle ne participe pas à la simulation numérique. De plus, un outil de performance comme celui là peut ne pas être parfaitement adapté aux méthodes éléments finis, puisque la nature des calculs est différente,

et cela n'est pas optimal. C'est cette réflexion qui introduit une troisième technique qui serait idéale.

Cette dernière méthode, dite idéale, serait d'instrumenter le code de calcul pour que des données de performance puissent être enregistrées pendant les calculs d'éléments finis de la simulation. Ainsi, l'utilisation de l'outil de performance serait évitée au profit d'un outil bien plus adaptée (sur la nature des calculs), et puis surtout, les opérations effectuées pour les mesures de performance ne serait pas inutiles pour la simulation numérique. Pour cela, les noeuds à utiliser sont quelconque à priori, et la première étape représente la création d'une partition homogène, vu qu'à ce moment là, les performances réelles des ressources utilisées ne sont pas censées être connues. Ensuite, le calcul peut commencer tout en mesurant la puissance des processeurs et le débit des réseaux. Pour une simulation non stationnaire sur plusieurs centaines ou milliers d'incréments de temps, quelques incréments suffisent, pour que la durée des mesures soit assez importante pour avoir une moyenne convenable, mais qui reste négligeable par rapport à la durée totale de l'exécution. Ces calculs préliminaires sont peut être plus lents qu'avec une partition optimisée, mais ils servent en même temps à récolter les données utiles au partitionneur pour réaliser une nouvelle partition optimisée. Une fois cette deuxième étape accomplie, la simulation peut continuer, mais cette fois ci, avec une répartition des calculs et des données hétérogène pour s'adapter aux ressources. Cette méthode « automatique » demande quelques développements au sein du code de calcul (ici CimLib) pour enregistrer des mesures de performance pendant l'assemblage des matrices et la résolution des systèmes linéaires, mais représente sûrement une (évolution) qui améliore l'utilisation de la grille de calcul.

5.7. Conclusion

Les performances de MecaGrid varient considérablement au cours du temps ; mais l'idée de repartitionner pendant l'exécution du programme parait difficile compte tenu, à la fois, du temps nécessaire à repartitionner, et de la rapidité à laquelle changent les performances. Le repartitionnement aurait seulement été utile dans le cas où les variations des performances avaient une période significative bien plus importante (de l'ordre de la dizaine d'heures).
Dans le paragraphe suivant, diverses applications sont étudiées en utilisant les techniques d'optimisations présentées ici. Il est notamment montré l'influence que peuvent avoir les partitions sur la résolution des systèmes linéaires. Plus précisément, ce sont les méthodes de préconditionnement qui semblent dépendre de la façon dont le partitionnement a été effectué.

6. Exemples d'applications sur la grille

6.1. Code de calcul

Après avoir analysé les caractéristiques de l'architecture théorique puis réelle de la grille de calcul, le but est de l'utiliser pour effectuer des calculs à grande échelle qui serviront à la fois à tester nos techniques d'amélioration de performance sur cet outil, et à montrer des applications dans le cadre de la mécanique des fluides hétérogènes. C'est avec notre code de calcul implémenté en C++ que les simulations numériques sont exécutées. La librairie CimLib développée au CEMEF contient un ensemble de solveurs numériques qui s'appuient sur des méthodes de type éléments finis. Elle a été parallélisée en utilisant la librairie *MPI* (*Message Passing Library*) largement reconnue lorsque l'on parle de programmation parallèle par envois de messages. Les matrices locales qui sont calculées dans CimLib sont ensuite prisent en charge par PETSc. Cette librairie (*Portable and Extensible Toolkit for Scientific Computation*) permet de résoudre en parallèle et de façon itérative des systèmes d'équations linéaires et non linéaires impliquant des matrices creuses. Les matrices locales que PETSc reçoit, sont assemblées en une matrice globale. Plusieurs méthodes de préconditionnement sont à disposition pour permettre une meilleure résolution : selon la nature du système à résoudre, des préconditionneurs de type ILU(k) ou Jacobi par blocs sont souvent choisis. Enfin, le système linéaire global est résolu en utilisant une méthode du résidu conjugué, soit MINRES, soit GMRES, selon les propriétés du système.

Les méthodes d'éléments finis qu'utilisent les solveurs de CimLib s'appuient sur un maillage du domaine de calcul constitué de tétraèdres (ou des triangles en 2D). Pour les calculs parallèles, ce maillage est partitionné ; puis, chaque processeur calcule l'écoulement dans sa portion du maillage. Les communications inter processeurs effectuées lors du produit matrice - vecteur dépendent du nombre de nœuds à l'interface des partitions.

Conformément aux méthodes choisies et développées tout au long de cette thèse, les inconnues de tous nos systèmes à résoudre sont exclusivement situées sur les nœuds du maillage. Ainsi, comme nous le verrons dans le *chapitre II* qui traite des écoulements, le solveur de type P1+/P1 pour résoudre le problème de Navier Stokes a 4 inconnues par nœud en 3D (3 pour la vitesse, et 1 pour la pression) et 3 inconnues par noeud en 2D (2 pour la vitesse, et 1 pour la pression). Le solveur d'évolution des phases présenté dans le *chapitre III*, résulte en un système à 1 inconnue par nœud. En effet, la fonction de phase transportée est une fonction Level Set qui est à la fois scalaire et continue (P1). Ces deux systèmes sont résolus l'un après l'autre, et ce, à chaque incrément de temps, tout au long de la simulation. Par conséquent, il y 5 inconnues à trouver par incrément de temps. Le nombre d'incréments dépend de la durée totale de la simulation, et du pas de temps utilisé.

6.2. Cas d'extrusion à 65000 nœuds

Ce premier exemple est un cas d'extrusion en 3D avec un maillage à environ 65000 nœuds. Il correspond à la simulation stationnaire de l'écoulement d'un fluide newtonien très visqueux incompressible en négligeant les termes d'inertie. C'est donc avec les équations de Stokes qu'une telle simulation est calculée. Le maillage de la cavité 3D qui correspond au

domine de calcul est représenté dans la figure *(8)*. La partie cylindrique de la pièce fait 80 cm de diamètre. Les différentes couleurs indiquent le processeur qui contient chaque élément : il y a ici six partitions (et donc six couleurs) pour un calcul sur six processeurs.

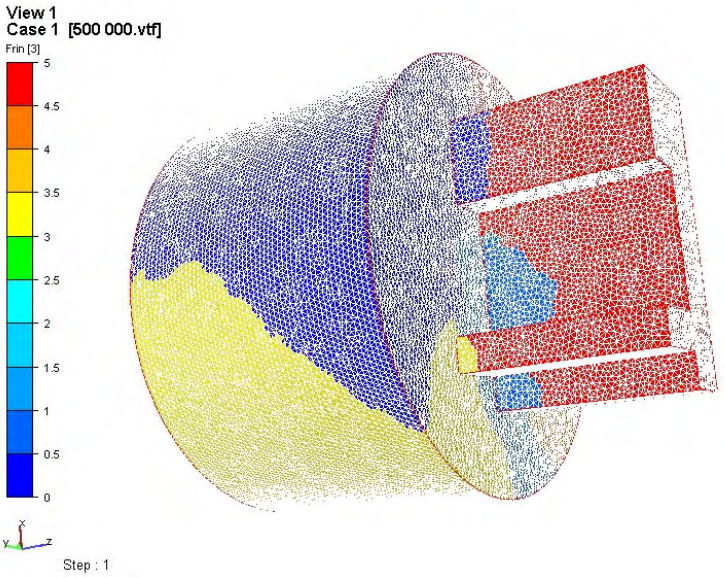


Figure 8 – Maillage et partitions de la cavité

Les conditions aux limites sont les suivantes pour faire passer le fluide dans la pièce de la gauche vers la droite. Une pression est imposée (condition limite de type Neumann) à 100000 Pa sur le plan d'entrée dans la pièce ($z = 0$, à gauche sur la figure), tant dis que sur le plan de sortie ($z = 85\,\text{cm}$, à droite sur la figure), et sur tous les autres nœuds surfaciques, la pression est laissée libre. Un contact collant (condition limite de type Dirichlet avec $v = 0$) est appliqué sur toute la surface de la cavité, excepté sur les plans d'entrée et de sortie où seule la composante en *x* est laissée libre ($v_x = v_y = 0$). Cela force le fluide à rentrée dans la pièce par la gauche avec une direction dans le sens de l'axe *z*, et d'en sortir par la droite, toujours dans le direction des *z*. Les figures *(9)* et *(10)* montrent les résultats de cette simulation stationnaire

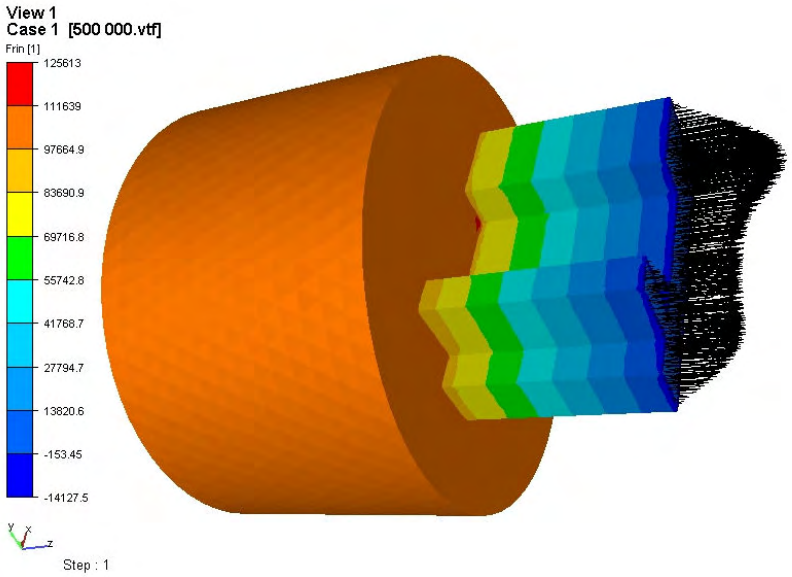


Figure 9 – Profil de pression et champ de vitesse

La pression dans le fluide newtonien contenu dans la cavité est visualisée avec les franges de couleurs. On observe un gradient de pression régulier dans la partie étroite, et elle s'annule quasiment sur le plan de sortie. Les vecteurs de couleur noirs représentent la vitesse du fluide lorsque celui-ci sort de la pièce sous l'effet de la pression imposée en entrée. Comme prévu par les conditions limites prédéfinies, cette vitesse est en direction de l'axe *z*, mais son intensité varie en fonction des coordonnées *x* et *y*. La figure *(10)* en montre la norme.

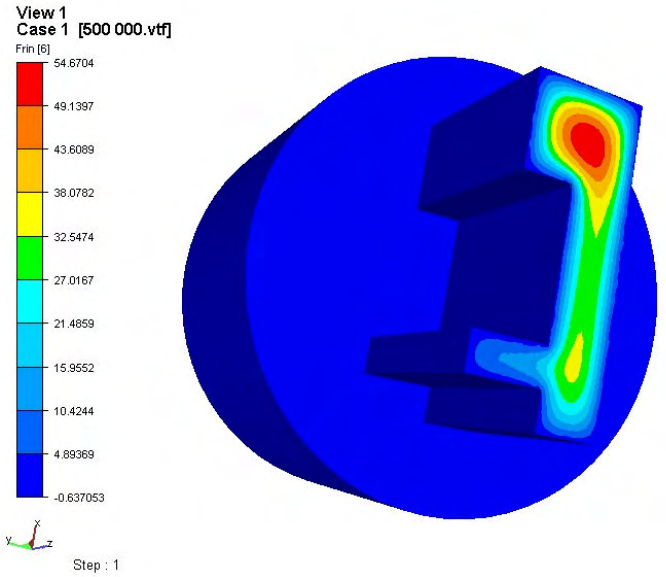


Figure 10 – Norme de la vitesse

La norme de la vitesse est nulle sur toute la surface de la pièce, sauf sur les plans d'entrée et de sortie où la plus forte valeur est de 55 cm/s (en rouge sur la figure *10*).

Voici maintenant les temps de résolution sur chaque site de MecaGrid séparément avec un (1×1) puis trois (1×3) processeurs. Les temps affichés ici englobent uniquement l'assemblage des matrices et la résolution d'un système linéaire et symétrique résultant des équations de Stokes. Après avoir appliqué un préconditionnement par ILU(0), PETSc résout le système linéaire par une méthode de résolution itérative de type résidu conjugué MINRES. Le nombre d'itérations nécessaire pour converger vers la solution avec une précision à 10^{-7} est indiqué pour que l'on puisse comparer la performance des différents clusters indépendamment du nombre d'itérations exécutées (avec le temps passé par itération).

1x1	INRIA-nina	CEMEF	IUSTI
Temps (s)	210.91	1038.46	207.01
Nb itérations	521	521	521
Temps/itér (s)	0.40	1.99	0.40

1x3	INRIA-nina	CEMEF	IUSTI
Temps (s)	87.56	455.57	173.66
Nb itérations	653	653	650
Temps/itér (s)	0.13	0.70	0.27

Tableau 10 – Temps d'exécution et nombre d'itérations sur un cluster

Ces résultats sont en accord avec l'architecture des clusters étudiés : le cluster de l'INRIA-nina a les meilleures performances, et le cluster du CEMEF a les moins bonnes. La différence qu'il existe dans les temps d'exécution montre bien la diversité des ressources de calcul et de l'hétérogénéité globale de MecaGrid. Ainsi, le même calcul est 5 fois plus lent en série (sur

un processeur) au CEMEF que sur INRIA-nina, et 5.4 fois plus lent en parallèle (sur trois processeurs). En série, INRIA-nina et IUSTI ont des performances similaires, mais en parallèle (sur trois processeurs), INRIA-nina se montre plus rapide (2 fois plus rapide). C'est sans doute la qualité du réseau interne (LAN) qui crée cette différence : il est de 1000 Mb/s sur INRIA-nina, et de 100 Mb/s sur IUSTI, soit 10 fois plus performant. En effet, en série, aucune communication n'est effectuée, donc le débit du LAN n'influe absolument pas sur les temps de calcul ; mais en parallèle, le réseau est utilisé pour échanger des messages entre les nœuds, et donc, les performances dépendent débit du LAN. C'est aussi sûrement ce qui explique qu'il y ai encore plus de différence entre le CEMEF et INRIA-nina en parallèle qu'en série. Enfin, les quelques itérations que l'IUSTI fait en moins viennent du fait que les versions des librairies et compilateurs ne sont pas tout à fait les mêmes sur tous les clusters de la grille.

Dans un deuxième temps, nous notons les temps de résolution sur la grille avec 1 processeur sur chaque site (3×1), 2 processeurs sur chaque site (3×2), puis 3 processeurs sur chaque site (3×3). Pour chaque cas, deux sortes de partitions ont été utilisées ici : une partition homogène, puis une partition optimisée en tenant compte des caractéristiques de la grille mesurées par notre outil de performance.

3x1	Homogène	Optimisée
Temps (s)	649.52	493.89
Nb itérations	651	634
Temps/itér (s)	1.00	0.78

3x2	Homogène	Optimisée
Temps (s)	458.86	436.05
Nb itérations	643	765
Temps/itér (s)	0.71	0.57

3x3	Homogène	Optimisée
Temps (s)	495.37	498.93
Nb itérations	657	780
Temps/itér (s)	0.75	0.64

Tableau 11 – Temps d'exécution et nombre d'itérations sur trois clusters

Sur trois processeurs, les temps d'exécution sont plus importants sur la grille (3×1) que sur chaque site séparément (1×3). Plusieurs raisons différentes viennent s'ajouter pour arriver à ce résultat décevant à première vue. Premièrement, tous les clusters étant concernés, les moins performants ralentissent les plus puissants. Il est donc normal qu'une résolution sur trois processeurs sur la grille soit inférieure à celle exécutée sur INRIA-nina. Cependant, les résultats ne sont pas catastrophiques car les temps du cas 3×1 sont bien inférieurs au cas 1×1 sur le CEMEF, et du même ordre que le cas 1×3 sur le CEMEF. Deuxièmement, les communications très lentes entre les sites importent une pénalisation supplémentaire qui n'existe pas au sein d'un cluster isolé. Et troisièmement, l'hétérogénéité des ressources de calculs entraîne une difficulté dans la répartition des tâches. Une amélioration importante de cette problématique est apportée par l'optimisation des partitions : le gain de temps est dans ce cas d'environ 20% par itération, quelque soit le nombre de processeurs.
Nous remarquons que dans certains cas (ici pour 3×2 et 3×3), le nombre d'itérations nécessaires à la convergence du système linéaire varie de façon importante entre les partitions homogènes et optimisées. Cela pourrait paraître surprenant, mais en fait, c'est une mise en évidence de l'influence que peut avoir le partitionnement sur la résolution. Plus

particulièrement, ce sont les méthodes de préconditionnement qui dépendent des partitions du maillage. Ce résultat est important car cela oblige à faire attention à ce que les partitions créées pour répartir la masse de calcul ne gênent pas la méthode de résolution pour converger vers la solution.

Enfin, la dernière observation que l'on peut tirer du tableau *(11)* est la baisse d'efficacité parallèle lorsque le nombre de processeurs augmente trop. La résolution d'un système linéaire sur six processeurs de la grille (3×2) est meilleur que sur trois (3×1) ; mais sur neuf processeurs (3×3), les temps de résolution sont supérieurs au cas 3×2. Si le cas 3×1 est considéré comme le temps de référence, l'efficacité parallèle est de 0.71 sur six processeurs, et de 0.44 sur neuf processeurs. Pour un « petit » maillage à 65000 nœuds comme celui utilisé ici, exécuter le programme sur neuf processeurs de MecaGrid est moins intéressant que sur six. Cela vient du fait que la masse de calcul répartie sur chaque processus devient trop petite, et le temps passé dans les communications prend le dessus pour ralentir les calculs. Plus le nombre de processeurs est grand, moins le nombre d'inconnues à résoudre par chaque processeur est petit, et plus il est nécessaire de communiquer entre les nœuds et les clusters. Ce phénomène est déjà observé sur des clusters isolés, mais est largement amplifié lorsqu'il s'agit de machine parallèle hétérogène et de grille de calcul.
Avec l'instrumentation de PETSc, il est possible de connaître la puissance en flops/s de la machine parallèle utilisée pour l'exécution. Cela correspond à la définition de la puissance *(1)*. Dans le cas 3×2 du tableau *(11)*, la puissance de MecaGrid sur 6 processeurs est de 89 Mflops/s pour une partition homogène, et de 93 Mflops/s pour une partition optimisée.

6.3. Écroulement du barrage

L'application de l'écroulement d'un barrage en 3D est un benchmark souvent utilisé pour tester des codes de simulation pour écoulement de fluides et de surfaces libres (voir [Gaston, 1997], [Anagnostopoulos, 1999] et [Maronnier, 2000]). Il s'agit de l'effondrement d'une colonne d'eau sous l'effet de la gravité dans une cavité en forme de parallélépipède rectangle dont les premiers expérimentaux sont présentés dans [Martin, 1952]. Notre code de calcul CimLib et les méthodes numériques développées dans cette thèse sont utilisés ici pour simuler cette application d'écoulement de fluide newtonien hétérogène et instationnaire. Brièvement, les équations de Navier Stokes qui tiennent compte des termes d'inertie sont généralement utilisées dans ce cas (voir *chapitre II*). L'évolution des deux phases en présence (l'eau et l'air) est assurée par le transport d'une fonction Level Set qui permet de capturer les interfaces (voir *chapitre III*).

Les résultats du paragraphe qui traite de l'optimisation de l'utilisation de MecaGrid montrent que la méthode de préconditionnement optimale pour résoudre les équations de Navier Stokes en 3 dimensions est ILU par bloc de 1. C'est donc celui-là qui est utilisé ici.

La cavité de dimensions 1.5 × 1.0 × 2.0 mètres est maillée avec 445 000 nœuds et 2 560 000 éléments. La taille de maille correspondante est de 2 centimètres. Le temps physique total de la simulation est 3 secondes, et le pas de temps utilisé est 0.005 seconde. 600 incréments en temps sont donc nécessaires. Sur la figure *(11)*, l'interface entre le fluide et l'air est montrée aux moments $t = 0s$, $t = 0.6s$, $t = 1.2s$, $t = 1.8s$, $t = 2.4s$, et $t = 3s$. Les paramètres utilisés sont les suivants : le fluide a une viscosité de 10 Pa.s (à peu près celle d'un

miel liquide) et une masse volumique de 1000 kg/m^3, l'air a une viscosité de 1 Pa.s et une masse volumique de 1 kg/m^3. Les conditions aux limites sont de type glissant, c'est-à-dire que, sur toutes les surfaces de la géométrie, seule la composante tangentielle de la vitesse est imposée à zéro.

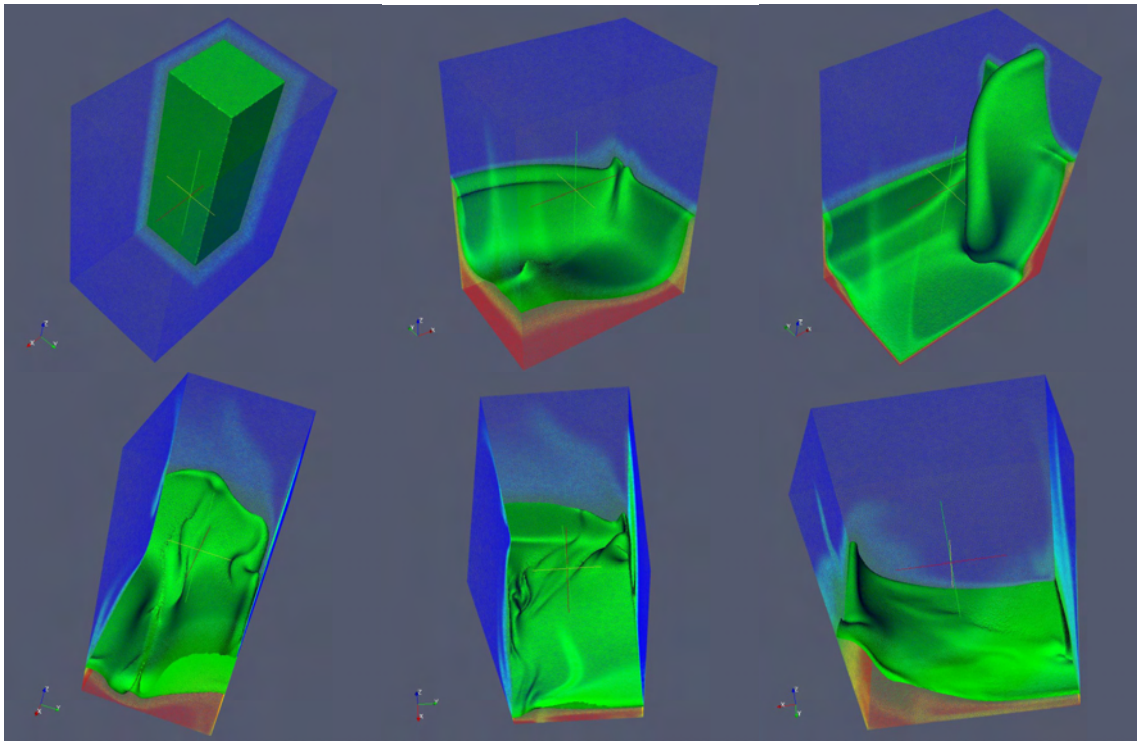


Figure 11 – Surface libre lors de l'écroulement d'un barrage

Un nombre de 12 processeurs est choisi pour exécuter cette application. Dans un premier temps, seul le cluster nina de l'INRIA est utilisé (1×12). Dans un deuxième temps, c'est avec 3 clusters différents de MecaGrid (nina, pf et iusti) que le cas est exécuté (3×4), avec un partitionnement homogène, puis, avec un partitionnement optimisé. Pour indication, chaque partition de maillage homogène contient environ le même nombre de nœuds, soit 35100. Par contre, ce n'est pas de tout le cas pour la partition optimisée : le nombre moyen de nœuds par processeur dans les clusters nina, pf et iusti sont respectivement de 45500, 18150, et 52600. Ces valeurs sont en accord avec la puissance des processeurs mesurée (respectivement 125, 44, et 127 Mégaflops par seconde).
Le tableau *(12)* montre les temps d'exécution de l'assemblage des matrices locales pour le solveur Navier Stokes (NS) et du transport des interfaces (Alpha), ainsi que les temps de résolution des systèmes.

Lieu	nina	3x4	3x4
Nb processeurs	12	12	12
Partition	Homogène	Homogène	Optimisée
NS assemblage	12 648	24 156	19 800
NS résolution	80 028	108 907	89 268
Alpha assemblage	11 472	28 841	23 640
Alpha résolution	564	996	816
Total (s)	104 712	162 899	133 524

Tableau 12 – Temps d'exécution sur un cluster et sur la grille

Premièrement, la partie qui prend le plus temps pendant la simulation est la résolution du solveur de Navier Stokes pour les écoulements (voir *chapitre II*). Le système non linéaire est linéarisé par la méthode de Newton qui consiste à résoudre plusieurs itérations de Newton, jusqu'à convergence du système (en moyenne, 4 à 5 itérations de Newton sont nécessaires). Ce solveur est donc le plus complexe, et consomme plus de temps que le solveur pour le transport des interfaces.

Deuxièmement, l'exécution sur le cluster nina isolé est la plus rapide (environ 29 heures), tandis que sur MecaGrid, la durée est bien plus longue (45 heures, soit 155% du temps d'exécution sur nina). Les raisons principales sont les suivantes : le cluster pf ralentit les calculs de part ses performances propres, et des communications inter sites très pénalisantes sont introduites.

Troisièmement, un partitionnement optimisé permet d'améliorer ce résultat : un gain de 18% est apporté au temps d'exécution total. Cela traduit une meilleure répartition de la masse de calcul : une quantité de travail supérieure est donnée aux processeurs les plus puissants (ici sur iusti) pour que ceux-ci terminent leur calcul approximativement en même temps. Cependant, la géométrie de la cavité ne permet pas de diminuer les communications pénalisantes de manière significative. Il est en effet très difficile de diviser un objet cubique en plusieurs parties dont les interfaces soient petites. Toutefois, la contribution d'une partition optimisée est loin d'être négligeable sur des simulations de grande taille.

La même application est présentée ici, avec, cette fois-ci, un maillage plus fin à 1 500 000 nœuds et 8 625 000 éléments. 11 processeurs sont utilisés sur les 3 mêmes clusters (nina, pf et iusti) pour avoir un total de 33 processeurs.

Lieu	3x11	3x11
Nb processeurs	33	33
Partition	Homogène	Optimisée
NS assemblage	60 847	48 792
NS résolution	351 407	281 784
Alpha assemblage	36 316	29 121
Alpha résolution	2 235	1 792
Total (s)	450 805	361 489

Tableau 13 – Temps d'exécution sur la grille avec des partitions homogène et hétérogène

Les durées totales d'exécution sont d'environ 5 jours et 6 heures avec des partitions optimisées, et de 4 jours et 5 heures avec des partitions optimisées. Le gain apporté par cette technique est donc de 20% : valeur similaire à celle obtenue avec un maillage 3 fois moins fin. On peut ainsi conclure que ce résultat ne dépend pas de la taille du problème, mais plutôt de la forme de la géométrie. Afin de vérifier cela, une autre application susceptible d'engendrer de meilleurs résultats est étudiée dans le paragraphe suivant.

6.4. Déverse d'un jerricane

Cette application montre l'écoulement d'un fluide par le goulot d'un jerricane. Au début de la simulation, le fluide se trouve totalement dans le jerricane renversé : son goulot ouvert est dirigé vers le bas. Le fluide commence alors à se déverser par l'ouverture, tandis

que l'air s'engouffre à son tour dans le jerricane sous forme de bulles. Ainsi, la cavité se vide peu à peu de son contenu initial et se remplit d'air, jusqu'à ce que le fluide soit totalement déversé. Le même code de calcul et les mêmes méthodes numériques que dans l'application précédente de l'écroulement du barrage sont utilisés ici.

La totalité du domaine est maillée avec 500 000 nœuds et 2 750 000 éléments tétraédriques. Les dimensions du goulot sont de (20 × 20 × 10) centimètres. Les cavités inférieures et supérieures sont de même taille, soit (1 × 1 × 0.40) mètres. Le solveur pour l'écoulement ne permet pas un pas de temps supérieure à 0.01 seconde. Un nombre très important d'incréments de temps est donc nécessaire (1000 incréments pour 10 secondes de simulation). Les paramètres utilisés sont les suivants : le fluide a une viscosité de 10 Pa.s (à peu près celle d'une huile) et une masse volumique de 1000 kg/m^3, l'air a une viscosité de 1 Pa.s et une masse volumique de 1 kg/m^3. Les conditions aux limites sont de type collant, c'est-à-dire qu'une vitesse nulle est imposée sur toutes les surfaces de la géométrie.

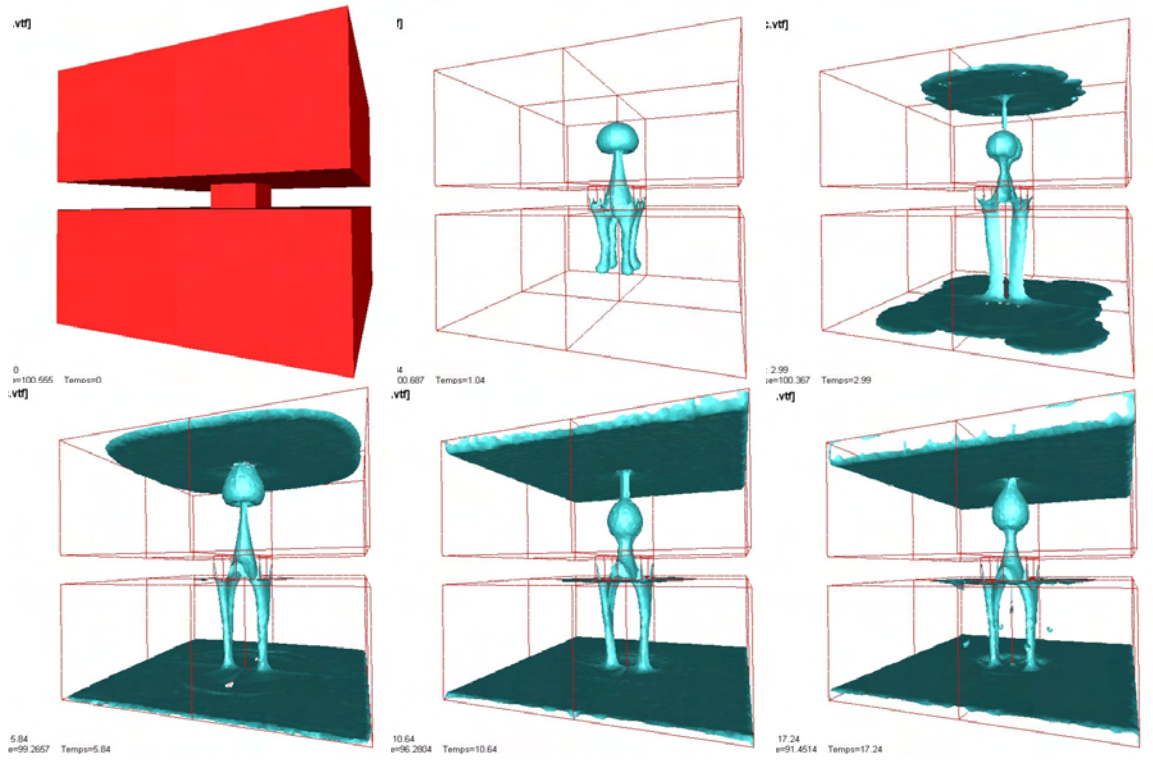


Figure 12 – Surface libre lors du déversement d'un jerricane

Sur la figure *(12)* est représentée l'interface entre le liquide et l'air aux temps $t = 0s$, $t = 1s$, $t = 3s$, $t = 6s$, $t = 11s$, et $t = 18s$. Alors que des bulles d'air montent dans le jerricane (la cavité du haut), le liquide coule dans une cuvette (la cavité du bas) à travers le goulot du jerricane. Le niveau du liquide baisse donc dans le jerricane et monte dans le réservoir.

Cette application est intéressante car, premièrement, elle induit des mouvements de surface libre complexes, et deuxièmement, sa géométrie facilite un partitionnement adapté à la grille de calcul. En effet, l'existence du goulot (partie étroite de la cavité) permet – si le partitionnement est bien réalisé – de diminuer considérablement les communications inter sites. Pour cela, il suffit d'attribuer la cavité du haut à un site, et celle du bas à un autre site. De cette manière, c'est seulement à l'endroit de la partie étroite que les communications inter sites s'opèrent, et elles deviennent donc bien moins importantes que les autres communications plus performantes.

Un total de 15 processeurs, et les 3 clusters nina, pf et iusti de MecaGrid sont choisis pour réaliser cette application. Il reste à déterminer combien de processeurs sont attribués sur chaque cluster. Chacun des 2 sites s'occupe d'une partie de la cavité afin de pouvoir placer l'interface entre les deux parties du maillage au niveau de la partie étroite : à l'INRIA, 4 nina et 5 pf ont la partie supérieure du maillage ; à l'IUSTI, 6 processeurs ont la partie inférieure. Ce choix se justifie par la puissance calculée des processeurs : pour les nina et les iusti, la puissance est d'environ 125 Mflops/s, et pour les pf, elle est de 44 Mflops/s, soit 2.5 fois inférieure. Sur le site de l'INRIA, 2 nina sont remplacés par 5 pf. Ainsi, avec 4 nina, 5 pf et 6 iusti, la puissance de calcul distribuée sur les deux sites sont quasiment les mêmes.

Les mesures de temps suivantes sont relevées après 100 incréments de temps, avec des partitions homogènes, puis optimisées. Dans le premier cas (tableau *14*), le préconditionnement ILU(0) est utilisé par le solveur des écoulements, mais dans le deuxième cas (tableau *15*), c'est avec ILU par blocs de 1 : ILU(1).

ILU(0)	4 5 6	4 5 6
Nb processeurs	15	15
Partition	Homogène	Optimisée
Nb itérations	10 702	10 847
NS assemblage	2 926	1 743
NS résolution	526 659	212 962
Alpha assemblage	1 846	1 808
Alpha résolution	231	62
Total (s)	531 662	216 575

Tableau 14 – Temps d'exécution sur la grille avec le préconditionneur ILU(0)

Avec ILU(0), le gain de temps apporté par le partitionnement optimisé est de 60%, ce qui est un excellent résultat. Le temps d'exécution est passé de 6 jours et 4 heures à 2 jours et 12 heures. Le progrès apporté par l'optimisation des partitions est spectaculaire. Mais puisque les résultats du paragraphe précédent avec l'écroulement du barrage sont obtenus avec le préconditionneur ILU(1), aucune comparaison peut être tenue pour l'instant.

ILU(1)	4 5 6	4 5 6
Nb processeurs	15	15
Partition	Homogène	Optimisée
Nb itérations	2 210	1 904
NS assemblage	3 680	2 455
NS résolution	235 504	107 899
Alpha assemblage	2 357	2 896
Alpha résolution	258	113
Total (s)	241 799	113 364

Tableau 15 – Temps d'exécution sur la grille avec le préconditionneur ILU(1)

La première observation est qu'avec le préconditionneur ILU(1), les résultats sont bien meilleurs que dans le tableau *(14)* : le temps d'exécution (avec un partitionnement homogène) correspond à 45% de celui obtenu avec ILU(0). Cette méthode fait diminuer le nombre d'itérations nécessaires à la convergence des systèmes linéaires de presque 5 fois, mais le coût de celles-ci augmente (106 secondes par itération contre 49 secondes). Au finale, utiliser ILU(1) est quand même bien plus avantageux que ILU(0).

Deuxièmement, avec ILU(1), le gain de temps apporté par le partitionnement optimisé est légèrement inférieure qu'avec ILU(0). Le temps d'exécution est passé de 2 jours et 19 heures à 1 jour et 8 heures, ce qui fait un gain de 53%, contre 60% avec ILU(0).

Dans l'application de l'écroulement du barrage, ILU(1) est utilisé et seulement 20% de temps est gagné par l'optimisation des partitions. Pourquoi donc, dans les mêmes conditions, cette technique fait-elle gagner 20% dans l'écroulement du barrage, et 53% dans la déverse d'un jerricane ? C'est la géométrie de la cavité et la façon dont on la partitionne qui est cruciale. Avec une géométrie purement cubique, l'apport d'une partition optimisée ne joue quasiment que sur une meilleure répartition de la masse de calcul, car les communications pénalisantes à l'interface des partitions ne peuvent pas être fortement diminuées. Or, en plus de répartir équitablement la masse de calcul, une géométrie telle que celle utilisée ici permet justement de diminuer considérablement les communications inter sites très pénalisantes en « coupant » le maillage dans un endroit où il y a peu de nœuds. Au total, l'addition des deux techniques d'optimisation (partitionnement et ILU(1)) fait gagner presque 80% sur les temps de calcul de MecaGrid.

L'instrumentation de PETSc monte les mesures de puissance suivantes pour ces exécutions sur 15 processeurs de MecaGrid : 915 Mflops/s avec un partitionnement homogène et ILU(0) ; et 1.96 Gflops/s avec un partitionnement optimisé et ILU(1).

6.5. Conclusion sur les applications numérique sur MecaGrid

Compte tenu des caractéristiques de la grille de calcul, il y a des applications qui offrent plus de possibilités d'optimisations que d'autres. Lorsque la cavité est de forme cubique, la seule optimisation possible sur les partitions se situe au niveau de la répartition des tâches. Mais, avec d'autres géométries plus adaptées, non seulement la masse de calcul est mieux répartie entre les processeurs suivant leur puissance, mais aussi, les communications pénalisantes sont minimisées au profit des réseaux les plus rapides.

7. Étude parallèle de la grille

7.1. Définition de l'efficacité parallèle

L'accélération (*speed up*) et l'efficacité parallèle sont des grandeurs habituellement utilisées pour évaluer les performances d'une application parallèle sur machine homogène. L'accélération correspond au gain de temps apporté par la parallélisation de l'exécution. C'est, en fait, le rapport entre les temps d'exécution en série (sur un seul processeur) et en parallèle (sur *p* processeurs) d'un même problème. Les temps d'exécutions sont notés ainsi :

$$t(n, p) \qquad (5)$$

où *n* est la taille du problème à résoudre et *p* est le nombre de processeurs utilisés.
L'évolution de l'efficacité en fonction du nombre de processeurs permet d'évaluer la scalabilité d'un code. Une efficacité de 1 représente une parallélisation parfaite, c'est-à-dire que le temps de résolution d'un problème est divisé exactement par le nombre de processeurs.
L'accélération et l'efficacité sont calculées comme suit :

$$Acc = \frac{t(n,1)}{t(n, p)} \qquad (6)$$

$$Eff = \frac{Acc}{p} \qquad (7)$$

Cependant, un problème se pose lorsque l'on parle de machine parallèle hétérogène et de grille de calcul. La définition de l'accélération *(6)* n'est plus consistante car une dépendance se crée pour le choix du processeur utilisé dans la mesure du temps d'exécution $t(n,1)$. Si, par exemple, on veut calculer l'accélération obtenue sur MecaGrid avec les clusters nina et cemef, le résultat sera très différent si l'on utilise un nina ou un cemef pour mesurer $t(n,1)$. Une solution intuitive pour adapter la définition *(6)* pour les grilles de calcul est de ne plus prendre $t(n,1)$ comme temps de référence, mais $t(n,r)$ qui correspond au temps d'exécution, non pas en série, mais sur un nombre *r* de processeurs spécifiques :

$$AccGrille = r \times \frac{t(n,r)}{t(n, p)} \qquad (8)$$

$$EffGrille = \frac{AccGrille}{p} \qquad (9)$$

Le choix des *r* processeurs dépend des ressources de calcul (ici des clusters) utilisées pour mesurer $t(n, p)$: ils doivent inclure un processeur par cluster employé. Si les *p* processeurs sont tous situés sur 2 clusters différents (par exemple : nina et cemef), les *r* processeurs seront d'un nombre de 2, avec un sur nina et un autre sur cemef. $t(n,r)$ peut ainsi être mesuré en concordance avec le choix au préalable de *p*. Pour que cela reste utile, il faut que *r* puisse être suffisamment petit par rapport à *p*. Entre autre, il faut que les ressources de calcul – parmi lesquelles les *p* processeurs sont choisis – puissent être regroupées entre elles en terme de performance. Sinon, le calcul de telles grandeurs risque de porter un intérêt minime. Même si

cette nouvelle définition pour l'accélération peut comporter quelques inconvénients, c'est celle-là que l'on utilisera par la suite pour étudier les codes de calculs sur MecaGrid.

7.2. Essais parallèles

Les tests effectués pour étudier l'accélération et l'efficacité parallèle de MecaGrid sont fait à partir d'une application réelle de simulation numérique. C'est le cas du déversement d'un jerricane similaire à celui présenté dans le paragraphe précèdent qui est choisi. Un maillage plus grossier à 120 000 nœuds a été créé afin de pouvoir utiliser un nombre minimal de processeurs, compte tenu de la quantité de mémoire maximale disponible. Avec un pas de temps fixé à 0.01 seconde, 100 incréments sont effectués pour une simulation numérique d'un temps physique de 1 seconde. Les modèles et les méthodes numériques sont identiques à celles présentées dans cette thèse et utilisées dans les applications ci-dessus. Les systèmes non linéaires résultant des équation de Navier Stokes sont préconditionnés par ILU(0), tandis que pour la capture des interfaces par Level Set, les systèmes linéaires sont préconditionnés par une méthode Jacobi par blocs. Toutes les résolutions sont faites par la méthode itérative du résidu général minimal GEMRES de PETSc.

Cette même application est lancée plusieurs fois sur MecaGrid avec des clusters prédéfinis : nina, pf et iusti. C'est donc en prenant 1 processeur sur chacun de ces 3 clusters (3×1) que le cas de référence correspondant à $t(n,r)$ est exécuté. Ensuite, c'est avec 6 (3×2), 12 (3×4), puis 24 (3×8) processeurs que l'exécution est répétée. Les partitions utilisées ici étant homogènes, le nombre de nœuds contenus dans chaque partition de maillage est sensiblement le même sur tous les processeurs, et dépend du nombre total de processeurs, le maillage restant le même.

Dans le tableau *(16)*, les temps d'exécution sont affichés dans chacun des cas. De plus, le nombre d'itérations total nécessaire pour faire converger les systèmes linéaires permet de calculer le temps passé en moyenne pour une itération.

Exécution	3x1	3x2	3x4	3x8
Nb processeurs	3	6	12	24
Nœuds/proc	39 843	19 921	9 961	4 980
Nb itération	365 000	447 900	533 600	543 600
Temps total	241 285	151 582	97 558	162 878
Temps/itér	0.66	0.34	0.18	0.30

Tableau 16 – Temps d'exécution sur la grille de calcul

Augmenter le nombre de processeurs fait baisser la durée d'exécution jusqu'à 12 processeurs. Au-delà de cette limite, elle se met au augmenter, comme le montre la figure *(13)*.

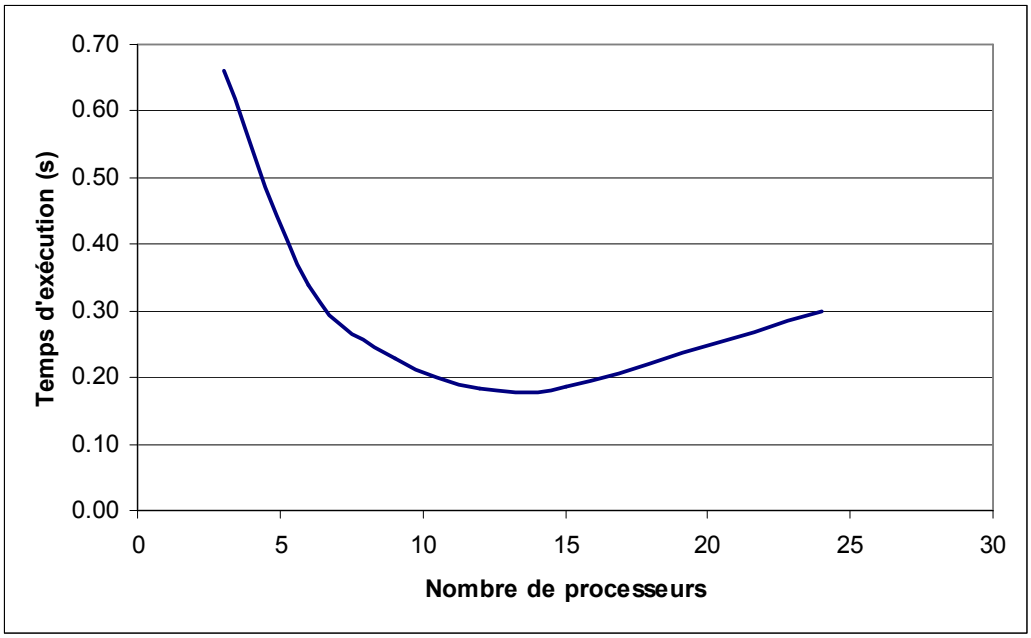


Figure 13 – Temps d'exécution en fonction du nombre de processeurs

Ce résultat pourrait paraître surprenant, mais, le fait de diviser la masse de calcul de façon trop importante entraîne une augmentation du rôle que jouent les communications – inter processeurs et inter clusters – par rapport à celui des calculs eux-mêmes. Ce premier constat amène à penser que la quantité de travail assignée aux processeurs doit être maximale pour obtenir de bonnes performances avec MecaGrid.

On remarque aussi que le nombre d'itérations nécessaire à la convergence des systèmes augmente avec le nombre de processeurs. La figure *(14)* montre l'évolution du nombre d'itérations total en fonction du nombre de processeurs. Cette augmentation est conséquente, et ne peut être négligée. Elle est due à l'influence qu'ont les partitions sur les méthodes de préconditionnement : plus le nombre de sous domaines est grand, moins les préconditionneurs sont efficaces.

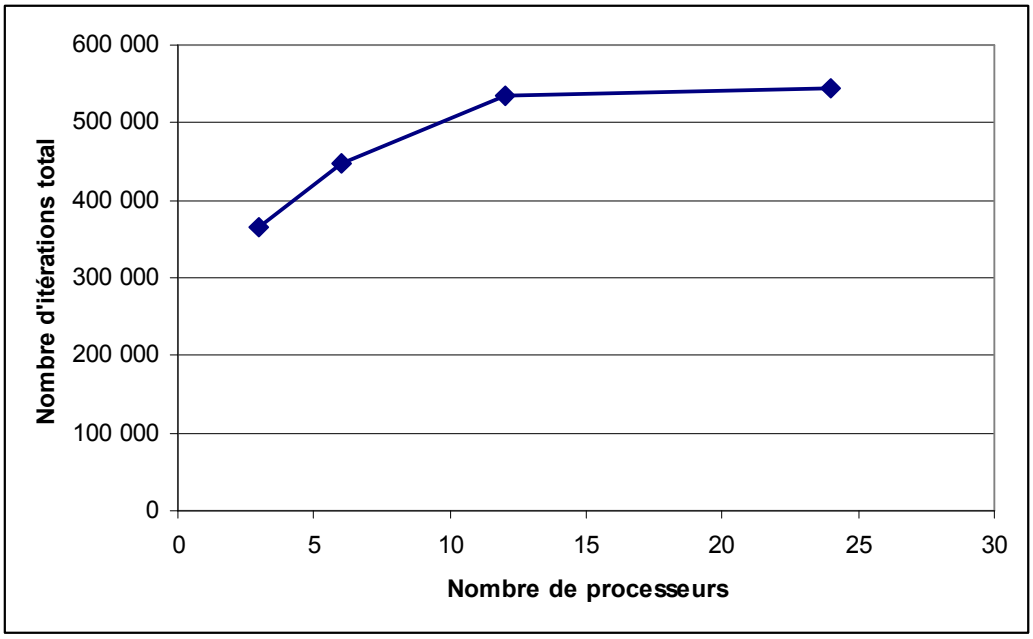


Figure 14 – Nombre d'itérations en fonction du nombre de processeurs

7.3. Efficacité et scalabilité sur MecaGrid

Avec les temps d'exécution du tableau *(16)*, l'accélération parallèle et l'efficacité peuvent être étudiées. Dans un premier, c'est avec les temps passés en moyenne pour une itération (temps total divisé par le nombre total d'itérations effectuées), et dans un deuxième temps avec les temps d'exécution totaux.

	3x2	3x4	3x8
Accélération	5.86	10.85	6.62
Efficacité	0.98	0.90	0.28

Tableau 17 – Efficacité parallèles (par itération)

Les résultats de ce premier tableau *(17)* sont indépendants du nombre d'itérations : le fait que celui-ci varie n'apparaît pas. Ce sont donc l'accélération et l'efficacité des calculs eux-mêmes, indépendamment de l'influence que peuvent avoir les partitions sur la résolution des systèmes. Ici, les efficacités parallèles pour 6 et 12 processeurs sont bonnes, car supérieure à 0.90 ; mais elle est médiocre pour 24 processeurs. Cela se traduit par une mauvaise scalabilité pour le code de calcul sur MecaGrid. La scalabilité d'un code est sa faculté de garder une bonne accélération, quelque soit le nombre de processeurs choisis pour le calcul parallèle. La figure *(15)* montre la scalabité lorsque l'on ne prend pas en compte le nombre d'itération. Elle est plutôt bonne lorsque peu de processeurs sont utilisés (et donc lorsque la masse de calcul sur chaque processeur est grande), mais devient médiocre dans le cas contraire.

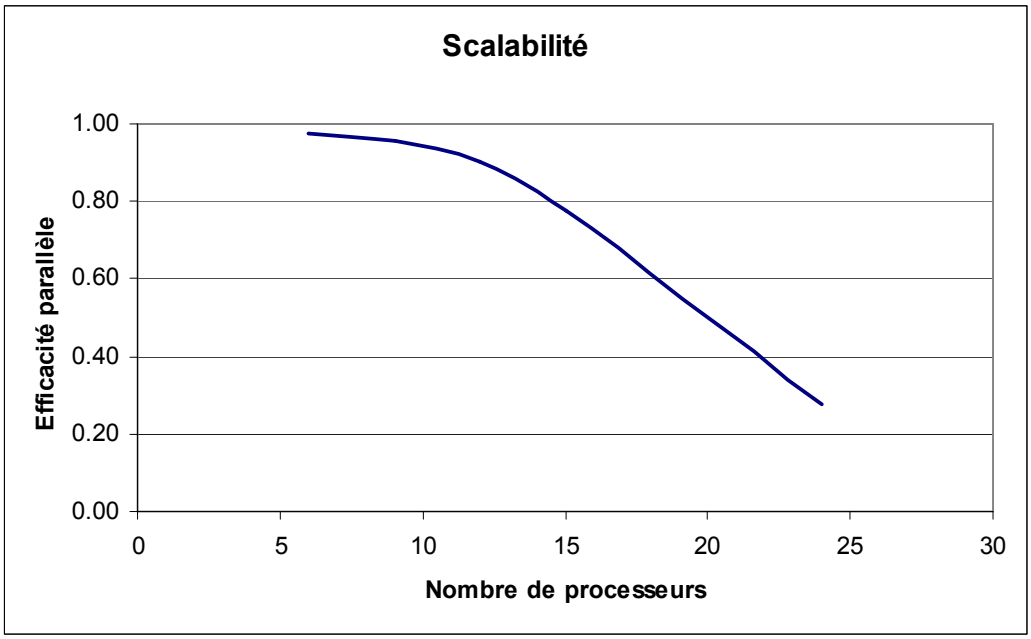


Figure 15 – Scalabilité (par itération)

Le tableau *(18)* est similaire au *(17)*, mais cette fois le nombre d'itérations est pris en compte. Puisque celui-ci augmente avec le nombre de processeurs, il est logique que l'efficacité soit moins bonne que dans le cas précédent (tableau *17*), surtout sur 12 et 24 processeurs.

	3x2	3x4	3x8
Accélération	4.78	7.42	4.44
Efficacité	0.80	0.62	0.19

Tableau 18 – Efficacité parallèles (en totalité)

En plus d'une mauvaise scalabilité naturelle sur grille de calcul, la convergence devient plus difficile lorsque le nombre sous domaines augmente. La figure *(16)* illustre ce cas, en comparaison avec la figure *(15)* précédente : la scalabilité totale est encore plus dégradée.

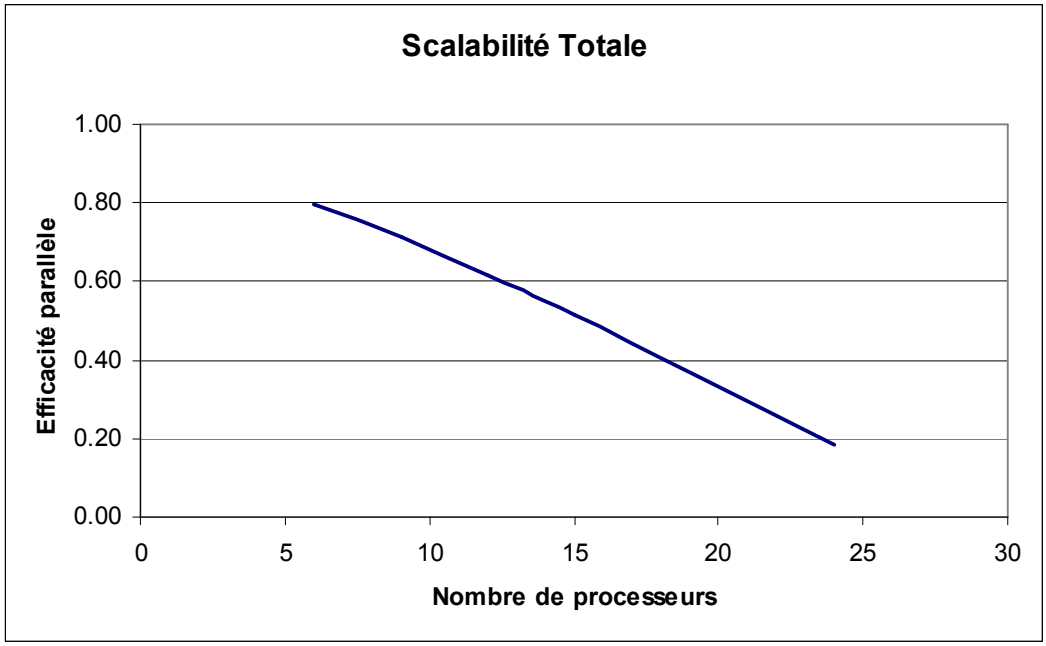


Figure 16 – Scalabilité (en totalité)

7.4. Conclusion de l'étude parallèle de MecaGrid

En général, les mauvaises communications inter cluster dues aux connexions internet font qu'il est préférable d'utiliser la grille pour effectuer des simulations à grandes échelles (et donc avec de gros maillages). Les petits maillages ne sont pas favorables à la grille car la masse de calcul des processeurs n'est pas assez importante vis-à-vis des communications pénalisantes. Et même, pendant une petite simulation, il est possible de passer plus de temps dans les communications que dans le calcul lui-même. Contrairement, avec des gros maillages, les tâches effectuées par les CPU prennent le dessus sur les échanges de messages (de données).

Ces derniers résultats montrent qu'une utilisation performante de MecaGrid passe par une répartition des tâches qui donne aux processeurs une masse de calcul conséquente pour que celle-ci garde le dessus sur les communications qui, nous l'avons vu, peuvent être médiocres dans MecaGrid. Dans l'idéal, il faudrait donc donner aux processeurs des partitions de maillage contenant le maximum de nœuds possible, tout en restant vigilant à la quantité de RAM disponible, car la dépasser amènerait des swaps de mémoire qui ralentissent les calculs. Ici, on observe pour avoir une efficacité proche de 1.0, il faut que la partition de maillage contienne un nombre de nœuds d'au moins 50 000. C'est dans la pratique, la valeur maximale permise par des processeurs aillant 256 Mo de RAM disponible.

8. Conclusion

Une grille de calcul a été élaborée avec succès en reliant quatre clusters de trois centres de recherche indépendants et géographiquement dispersés. Les nombreux problèmes rencontrés ont été résolus après avoir considéré les technologies disponibles à l'heure actuelle. Notamment, le middleware Globus apporte beaucoup lorsqu'il s'agit de coordonner différentes ressources indépendantes qui ont leurs propres politiques de sécurité. Le résultat est une grille de calcul très hétérogène dans sa conception, avec des processeurs et des réseaux très différents, et qui peut offrir une grande quantité de ressources de calculs à moindre coût. Les différentes études ont permis d'identifier plusieurs problématiques qui peuvent être améliorées avec des techniques d'optimisation adaptées. Ainsi, des utilisations performantes de la grille ont été définies avec succès. Enfin, une connaissance approfondie de cette infrastructure permet de la mettre en valeur avec des applications en mécanique des fluides hétérogènes. Tout est réuni pour réaliser des simulations à grande échelle. Et c'est dans cette optique que la suite de cette thèse consiste à rechercher des méthodes numériques adaptées aux gros calculs, dans le cadre de la simulation d'écoulements multi fluides. Cela veut dire que l'on privilégiera des techniques permettant des résolutions simples et des utilisations minimales de mémoire virtuelle. Enfin, des simulations à grande échelle de surfaces libres pourront être exécutées sur la grille de calcul.

Références

[Adalsteinsson, 1995] D. Adalsteinsson and J. A. Sethian, "A fast level set method for propagating interfaces", *J. Comput. Phys.* 118, 269 (1995).

[Allcock, 2001] B. Allcock, I. Foster, V. Nefedova, A. Chervenak, E. Deelman, C. Kesselman, J. Leigh, A. Sim, A. Shoshani, B. Drach, D. Williams, "High-Performance Remote Access to Climate Simulation Data: A Challenge Problem for Data Grid Technologies", *Proceedings of SC 2001* (2001).

[Allen, 1999] G. Allen, W. Benger, C. Hege, J. Mass´o, A. Merzky, T. Radke, E. Seidel, and J. Shalf, "Solving Einstein's equations on supercomputers", *IEEE Computer*, 32, 12 (1999).

[Allen, 2001] G. Allen, T. Dramlitsch, I. Foster, N. Karonis, M. Ripeanu, E. Seidel and B. Toonen, "Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus", *Proceedings of SC 2001*, 10 (2001).

[Anagnostopoulos, 1999] J. Anagnostopoulos, G.Bergeles. "Three-Dimensional Modeling of the Flow and the Interface Surface in a Continuous Casting Mold Model", *Metallurgical and materials transaction* N, 30B, 1095-1105, (1999).

[Balay, 2002] S. Balay, K. Buschelman,W. D. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, B. F. Smith and H. Zhang, "PETSc : Users Manual ANL-95/11 - Revision 2.1.5.", *Argonne National Laboratory* (2002).

[Barnard, 1999] S. Barnard, R. Biswas, S. Saini, R. Van der Wijngaart, M. Yarrow, L. Zechter, I. Foster and O. Larsson, "Large-Scale Distributed Computational Fluid Dynamics on the Information Power Grid using Globus", *Proc. of Frontiers '99* (1999).

[Basermann, 2000] A. Basermann, J. Clinckemaillie, T. Coupez, J. Fingberg, H. Digonnet, R. Ducloux, J. M. Gratien, U. Hartmann, G. Lonsdale, B. Maerten, D. Roose and C. Walshaw, "Dynamic load balancing of finite element applications with the DRAMA library", *Appl. Math. Modelling*, 25 (2), 83 (2000).

[Brown, 1999] M. D. Brown, T. DeFanti, M. A. McRobbie, A. Verlo, D. Plepys, D. F. McMullen, K. Adams, J. Leigh, A. E. Johnson, I. Foster, C. Kesselman, A. Schmidt, S. N. Goldstein, "The International Grid (iGrid): Empowering Global Research Community Networking Using High Performance International Internet Services", *iGrid* (1999).

[Buning, 1995] P. Buning,W. Chan, K. Renze, D. Sondak, I.-T. Chiu, J. Slotnick, R. Gomez, and D. Jespersen, "Overflow user's manual, version 1.6", *NASA Ames Research Center* (1995).

[Chen, 1991] Y. G. Chen, Y. Giga and S. Goto, "Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations", *J. Diff. Geom.* 33, 749 (1991).

[Evans, 1991] L. C. Evans and J. Spruck, "Motion of level set via mean curvature I", *J. Diff. Geom.* 33, 635 (1991).

[Foster, 1997] I. Foster and C. Kesselman "Globus : A metacomputing infrastructure toolkit", *The International Journal of Supercomputer Applications and High Performance Computing*, 11, 2 (1997).

[Foster, 1998] I. Foster, J. Geisler, W. Gropp, N. Karonis, E. Lusk, G. Thiruvathukal, and S. Tuecke, "Wide-area implementation of the Message Passing Interface", *Parallel Computing*, 24, 1735 (1998).

[Foster, 1999] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure", *Morgan Kaufmann* (1999).

[Gaston, 1997] L. Gaston. "Simulation numérique par éléments finis bidimensionnels du remplissage de moule de fonderie et étude expérimentale sur maquette hydraulique", *Thèse de doctorat*, Ecole Nationale Supérieure des Mines de Paris, (1997).

[Gropp, 1996] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A highperformance, portable implementation of the MPI Message Passing Interface Standard", *Parallel Computing*, 22, 789 (1996).

[Guillard, 1994] H. Guillard and B. N'Konda, "Godunov type method on non-structured meshes for three dimensional moving boundary problems", *Comput. Methods Appl. Mech. Eng.* 113, 183 (1994).

[Harabetian, 1996] E. Harabetian, S. Osher and C. W. Shu, "An Eulerian approach for vortex motion using a level set regularization procedure", *J. Comput. Phys.* 127, 15 (1996).

[Karypis, 1995] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs", *Department of Computer Science Tech. Rep. 95-035, University of Minnesota* (1995).

[Lee, 1996] C. Lee, C. Kesselman, S. Schwab, "Near-real-time Satellite Image Processing: Metacomputing in CC++", *Computer Graphics and Applications*, 16(4):79-84 (1996).

[Maronnier, 2000] V. Maronnier. "Simulation numérique d' écoulements de fluides incompressibles avec surface libre", *Thèse de doctorat*, Ecole polytechnique Fédérale de Lausanne (2000).

[Martin, 1952] J-C. Martin et M.J. Moyce. "Some gravity wave problems in the motion of perfect liquids", *Philos. Trans. Roy. Soc. London Ser*. A, 244, 231-334 (1952).

[Merriman, 1994] B. Merriman, J. Bence, and S. Osher, "Motion of multiple junctions: A Level Set approach", *J. Comput. Phys.* 112, 334 (1994).

[Osher, 1988] S. Osher and J. A. Sethian, "Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulation", *J. Comput. Phys.* 79, 12 (1988).

[Pearlman, 2004] L. Pearlman, C. Kesselman, S. Gullapalli, B.F. Spencer, Jr., J. Futrelle, K. Ricker, I. Foster, P. Hubbard and C. Severance, "Distributed Hybrid Earthquake Engineering Experiments: Experiences with a Ground-Shaking Grid Application", *Proceedings of the 13th IEEE Symposium on High Performance Distributed Computing* (2004).

[Peng, 1999] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang, "A PDE-Based Fast Local Level Set Method", *J. Comput. Phys.* 155, 410 (1999).

[Ripeanu, 2001] M. Ripeanu, A. Iamnitchi, and I. Foster, "Performance Predictions for a Numerical Relativity Package in Grid Environments", *International Journal of High-Performance Computing Applications*, 15, 4 (2001).

[Shirazi, 1995] B. A. Shirazi, A. R. Hurson and K. M. Kavi, "Scheduling and load balancing in parallel and distributed systems", *IEEE Computer Science Press* (1995).

[Sussman, 1996] M. Sussman, P. Smereka, and S. Osher, "A level set method for computing solutions to incompressible two-phase flow", *J. Comput. Phys.* 122, 179 (1996).

[Wissink, 1998] A. Wissink and R. Meakin, "Computational fluid dynamics with adaptive overset grids on parallel and distributed computer platforms", *In Intl. Conf. on Parallel and Distributed Processing Techniques and Applications*, 1628 (1998).

[Zhao, 1996] H. K. Zhao, T. Chan, B. Merriman and S. Osher, "A variational level set approach to multi-phase motion", *J. Comput. Phys.* 122, 179 (1996).

[Zhao, 1998] H. K. Zhao, B. Merriman, S. Osher and L. Wang, "Capturing the behaviour of bubbles and drops using the variational level set approach", *J. Comput. Phys.* 143, 495 (1998).

CALCUL D'ÉCOULEMENTS

PROBLEME DE NAVIER-STOKES

1. Introduction

La grille de calcul nommée MecaGrid est destinée aux applications en mécanique des fluides hétérogènes. Il est donc nécessaire de pouvoir simuler des écoulements multi-fluides instationnaires avec calculs d'interfaces ou de surfaces libres. Les travaux commencent à partir d'un solveur Stokes déjà implémenté dans la librairie CimLib qui calcule, avec une approche Eulérienne, des écoulements stationnaires de fluides newtoniens très visqueux. Cependant, afin de pouvoir considérer des fluides peu visqueux, il faut tenir compte des effets de la gravité et de l'inertie qui sont négligés dans les équations de Stokes. Les mouvements de fluides, dont la vitesse est petite devant celle du son ($v < 0.15c$), sont généralement considérés incompressibles. C'est donc avec les équations de Navier-Stokes incompressibles que l'on pourra décrire de tels écoulements. Elles peuvent être résolues avec une méthode numérique de type éléments finis mixtes en vitesse - pression, proche de celle utilisée dans le solveur Stokes. Une discrétisation spatiale linéaire et continue (de type P1+/P1 avec des inconnues situées sur les nœuds du maillage) avec le MINI-élément [Coupez, 1997] semble adaptée pour ce solveur mécanique afin de faciliter la réalisation de gros calculs. Enfin, une discrétisation temporelle est appliquée avec un schéma d'Euler explicite ou implicite.

L'objet de ce chapitre est de développer une méthode de résolution du problème de Navier-Stokes incompressible. Pour cela, une vue d'ensemble des techniques de stabilisation sera présentée, et notamment celle de la condensation d'une bulle qui rappelle les méthodes de type *multiscale*, largement reconnues dans la littérature. Le solveur ainsi implémenté sera étudié, puis validé, tout en gardant en mémoire qu'il doit permettre la réalisation de calculs de très grandes tailles en parallèle sur grille de calcul. Par la suite, c'est avec ce même solveur – mais avec un problème de Navier-Stokes non homogène – qu'une modélisation multi-fluide sera créée pour pouvoir simuler des écoulements hétérogènes avec capture d'interfaces.

Ce chapitre est consacré en premier à l'élaboration de la formulation du problème de Navier-Stokes. Puis, en parallèle avec les équations de Stokes, des techniques de stabilisation sont présentées en mettant en avant leurs intérêts apportés à la résolution du système. Ensuite, la librairie PETSc, qui propose des solutions intéressantes dans le domaine des calculs scientifiques parallèles, est choisie pour préconditionner et résoudre les systèmes, après linéarisation. Enfin, des cas tests issus de la littérature servent de validation pour notre solveur.

2. Formulations des équations de Navier-Stokes incompressibles

2.1. Formulation forte

Considérons l'écoulement d'un fluide newtonien dans une cavité Ω fermée et de dimension finie, dont la paroi s'écrit $\partial\Omega$. Cette cavité correspond au domaine de calcul dans lequel la vitesse et la pression du fluide évoluent avec le temps. Le vrai domaine de calcul est donc $\Omega \times [0,T]$, et les inconnues sont $v(x,t)$ et $p(x,t)$, $\forall (x,t) \in \Omega \times [0,T]$.

Un tel écoulement répond à loi fondamentale de la dynamique dont l'équation (nommée équation de l'équilibre dynamique) s'écrit de la façon suivante :

$$\nabla \cdot \sigma + F - I = 0$$

où σ est le tenseur des contraintes, F représente les forces extérieures agissant sur le fluide, et I est l'inertie du système. Généralement, on considère que seules les forces gravitationnelles jouent un rôle dans les écoulements de fluides, donc $F = \rho g$, où ρ est la masse volumique du fluide (en kg/m^3), et g est l'accélération gravitationnelle (-9.81 m/s^2). L'inertie, elle, peut être écrite sous la forme $I = \rho\gamma$, où γ est l'accélération du fluide.

$$-\nabla \cdot \sigma + \rho\gamma = \rho g \qquad (1)$$

Le tenseur des contraintes peut être décomposé en une partie déviatorique s et une partie sphérique pI :

$$\sigma = s - pI \qquad (2)$$

où p est la pression hydrostatique, et I est le tenseur identité. Dans le cas d'un fluide à comportement newtonien comme c'est le cas ici, la partie déviatorique des contraintes est connue pour être :

$$s = 2\eta\varepsilon(v) \qquad (3)$$

où η est la viscosité du fluide (en Pa.s), et $\varepsilon(v)$ est le tenseur des vitesses de déformation tel que :

$$\varepsilon(v) = \frac{1}{2}\left(\nabla v + \nabla v^T\right) \qquad (4)$$

Avec une formulation eulérienne pour l'accélération, nous avons :

$$-\nabla \cdot \left(2\eta\varepsilon(v) - pI\right) + \rho\frac{dv}{dt} = \rho g \qquad (5)$$

Puis :

$$-\nabla\cdot\left(2\eta\varepsilon(v)\right)+\nabla p+\rho\left(\frac{\partial v}{\partial t}+\nabla v.v\right)=\rho g \qquad (6)$$

L'équation de la conservation de la masse (aussi appelée équation de continuité) s'écrit :

$$\frac{\partial\rho}{\partial t}+\nabla\cdot\left(\rho v\right)=0 \qquad (7)$$

Pour un écoulement incompressible, l'évolution de la masse volumique en fonction de temps est négligeable par rapport au terme de divergence de la vitesse. On se limite donc à :

$$\nabla\cdot v=0 \qquad (8)$$

Afin de fermer ce problème, on doit lui ajouter des conditions initiales, ainsi que des conditions aux limites. En général, avec la considération d'espaces fonctionnels adéquats et de conditions limites assez régulières, il existe une solution unique à ce problème [Temam, 2000]. Les conditions appliquées aux limites sur $\partial\Omega$ peuvent être de type Dirichlet sur la vitesse $v=v_g$, ou de type Neumann avec une force surfacique $\sigma.n=f$.

Au final, la formulation forte du problème de Navier-Stokes incompressible, dans laquelle il faut trouver $v(x,t)$ et $p(x,t)$, $\forall(x,t)\in\Omega\times[0,T]$, est la suivante :

$$\begin{cases}\rho\dfrac{\partial v}{\partial t}+\rho\nabla v.v-\nabla\cdot\left(2\eta\varepsilon(v)\right)+\nabla p=\rho g\\ \nabla\cdot v=0\end{cases} \qquad (9)$$

La modélisation en mécanique des fluides par ces équations permet de simuler des écoulements visqueux plus ou moins turbulents. Le nombre de Reynolds est une valeur adimensionnelle qui quantifie cette turbulence : plus il est élevé, plus il y a la possibilité de turbulence. D'une manière générale, il est défini de cette façon :

$$\mathrm{Re}=\frac{\rho UL}{\eta} \qquad (10)$$

où *U* et *L* sont une vitesse et une longueur caractéristiques (en m/s et en m).

A noter que pour obtenir les équations de Stokes, il suffit de négliger les termes d'inertie et de généraliser les forces extérieures *f* comme ceci :

$$\begin{cases}-\nabla\cdot\left(2\eta\varepsilon(v)\right)+\nabla p=f\\ \nabla\cdot v=0\end{cases} \qquad (11)$$

2.2. Formulation variationnelle

2.2.1 Notations

Les espaces fonctionnels suivants sont nécessaires pour établir la formulation faible des équations de Navier-Stokes :

$$V = \left(H^1(\Omega)\right)^d$$
$$V^0 = \left(H_0^1(\Omega)\right)^d$$
$$P = L^2(\Omega)$$

où $L^2(\Omega)$ est l'espace de Lebesgue des fonctions carrées sommables sur un domaine Ω, et $H^1(\Omega)$ est l'espace de Sobolev inclus dans $L^2(\Omega)$:

$$L^2(\Omega) = \left\{ q, \int_\Omega q^2 dV < \infty \right\}$$
$$H^1(\Omega) = \left\{ q \in L^2(\Omega), \nabla q \in \left(L^2(\Omega)\right)^d \right\}$$
$$H_0^1(\Omega) = \left\{ q \in H^1(\Omega), q = 0 \text{ sur } \partial\Omega \right\}$$

Dans $L^2(\Omega)$, le produit scalaire est défini par :

$$(f_1, f_2) = \int_\Omega f_1 f_2 dV \qquad \forall f_1, f_2 \in L^2(\Omega) \tag{12}$$

2.2.2. Forme faible

Ecrivons le problème faible lié aux équations de Navier-Stokes incompressibles sous sa forme la plus générale des éléments finis, c'est-à-dire avec une méthode de Galerkin Standard. On recherche la vitesse *v* et la pression *p* dans les espaces fonctionnels *V* et *P*. Pour cela, on effectue le produit scalaire du problème au sens L^2 (voir définition *(12)*) par des fonctions tests *w* et *q* choisies dans V^0 et *P*. Avec la relation :

$$\frac{1}{2}\left(\nabla v + \nabla v^T\right) : \nabla w = \nabla v : \nabla w$$

et des intégrations par partie (formule de Green) qui se simplifient grâce au choix de la fonction test *w* ($w = 0$ sur $\partial\Omega$) :

$$\int_\Omega (\nabla p.w) = -\int_\Omega (p.\nabla \cdot w) + \int_{\partial\Omega} (p.w.n) = -\int_\Omega (p.\nabla \cdot w)$$

le problème variationnel consiste à trouver $(v, p) \in (V, P)$, quels que soient $(w, q) \in \left(V^0, P\right)$, tels que :

$$\begin{cases} \int_\Omega \rho \frac{\partial v}{\partial t}.w + \int_\Omega \rho.\nabla v.v.w + \int_\Omega 2\eta\varepsilon(v):\varepsilon(w) - \int_\Omega p.\nabla\cdot w = \int_\Omega \rho g.w \\ \int_\Omega q\nabla\cdot v = 0 \end{cases}$$

Ou alors, en appliquant la notation de la définition *(12)*, nous obtenons :

$$\begin{cases} \rho\left(\frac{\partial v}{\partial t}, w\right) + \rho(\nabla v.v, w) + (2\eta\varepsilon(v):\varepsilon(w)) - (p, \nabla\cdot w) = (\rho g, w) \\ (\nabla\cdot v, q) = 0 \end{cases} \qquad (13)$$

Des conditions aux limites (de type Dirichlet ou Neumann) doivent être imposées, ainsi que des conditions initiales : $v = v_0$ et $p = p_0$ pour $t = 0$.

2.3. Formulation discrète

Afin d'établir la formulation éléments finis, les espaces vectoriels V_h, P_h et V_h^0 de dimensions finies sont construits de façon à approcher V, P et V^0. Le domaine de calcul Ω est alors décomposé en tétraèdres K : c'est la triangulation $T_h(\Omega)$ (ou le maillage), où h est la taille de maille définie par le maximum des diamètres des éléments du maillage :

$$h = \max_{K\in T_h(\Omega)} diam(K)$$

Les éléments K de $T_h(\Omega)$ sont ici des d-simplexes, c'est-à-dire des triangles en 2D et des tétraèdres en 3D. Plus la taille de maille est petite, plus l'approximation des espaces fonctionnels est précise : $\lim_{h\to 0} V_h = V$, $\lim_{h\to 0} P_h = P$ et $\lim_{h\to 0} V_h^0 = V^0$. Avec une telle discrétisation, nous avons :

$$\Omega_h = \bigcup_{K\in T_h(\Omega)} K$$

La génération du maillage est réalisée par le logiciel nommé MTC [Coupez, 2000] qui, à partir d'une géométrie, crée une discrétisation spatiale à base de tétraèdres (ou triangles) non structurés. Il permet aussi une adaptation de la taille de maille de façon à raffiner les parties de la cavité les plus singulières.

Ainsi, une méthode élément fini mixte peut être appliquée en utilisant le MINI-élément (P1+/P1). Les champs de vitesse et de pression sont interpolés linéairement à partir des valeurs aux sommets des éléments (interpolation continue de type P1). Le champ de vitesse est enrichi par l'ajout d'un degré de liberté supplémentaire (appelé fonction bulle) au centre des éléments [Coupez, 1991]. Une sous-discrétisation est nécessaire pour que la fonction bulle soit linéaire et continue sur les sous-éléments et qu'elle s'annule sur les faces des éléments. La figure suivante montre les degrés de liberté du MINI-élément en 2 dimensions :

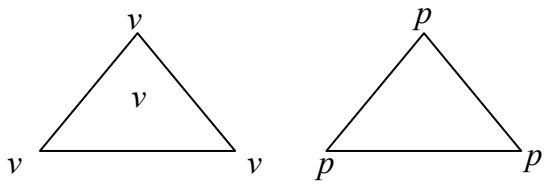
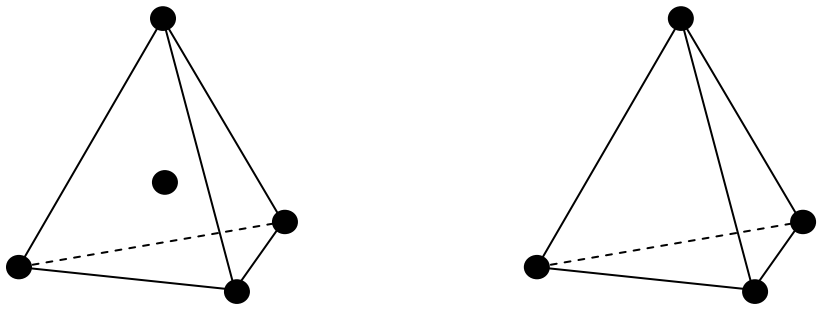


Figure 1 – MINI-élément en 2D

Sur un élément en 2D, il y a 4 degrés de liberté pour la vitesse (un à chaque nœud plus un au centre de l'élément), et 3 pour la pression (un à chaque nœud) ; tandis qu'en 3D, la figure suivante montre qu'il y en a 5 pour la vitesse et 4 pour la pression.



Interpolation en vitesse *Interpolation en pression*

Figure 2 – MINI-élément en 3D

Une telle discrétisation est choisie pour avoir des inconnues interpolées de façon linéaire et continue, tout en gardant un élément compatible à une méthode d'éléments finis mixtes, dans l'optique de générer des systèmes relativement simples à résoudre pour réaliser de gros calculs. Nous présenterons par la suite les détails de ces méthodes.

Les champs recherchés (v, p) sont approchés par les solutions $(v_h, p_h) \in (V_h, P_h)$ du problème discrétisé suivant pour tout $(w_h, q_h) \in (V_h^0, P_h)$:

$$\begin{cases} \rho\left(\frac{\partial v_h}{\partial t}, w_h\right) + \rho(\nabla v_h . v_h, w_h) + (2\eta\varepsilon(v_h) : \varepsilon(w_h)) - (p_h, \nabla \cdot w_h) = (\rho g, w_h) \\ (\nabla \cdot v_h, q_h) = 0 \end{cases} \qquad (14)$$

Les espaces d'approximation et le MINI-élément vérifient la condition de compatibilité de Brezzi-Babuška [Babuška, 1973] [Brezzi, 1974] qui assure l'existence et l'unicité de la solution de ce problème. Cette condition assure aussi la convergence de la solution (v_h, p_h) vers celle du problème fort (v, p) quand la taille de maille tend vers zéro. A noter que l'ordre de convergence du MINI-élément est O(h).

2.4. Linéarisation du problème

Avec le terme *(15)*, le problème de Navier-Stokes incompressible *(14)* est non linéaire. Il est donc nécessaire de trouver des procédés de linéarisation des équations, et notamment à partir de sa discrétisation temporelle.

$$\frac{\partial v}{\partial t} + \nabla v.v \qquad (15)$$

La discrétisation temporelle du problème de Navier-Stokes peut s'effectuer soit par une méthode de différences finies, soit par une méthode d'éléments finis [Magnin, 1994]. Ces dernières sont basées sur des formulations espace-temps des éléments finis et ne sont pas traitées ici. Cependant, ce type de méthode sera utilisé par la suite pour la résolution des équations de transport P0 par Galerkin Discontinu.

Dans la littérature, les schémas temporels par différences finies les plus utilisés pour les équations de Navier-Stokes sont ceux d'Euler et de Runge Kutta. C'est le premier que nous choisissons d'implémenter dans notre solveur mécanique.

2.4.1. Schémas temporels d'Euler

Pour décrire la discrétisation temporelle d'Euler, un problème non linéaire de Cauchy est introduit sous la forme suivante :

$$\frac{\partial v(t)}{\partial t} = f(v,t) \qquad (16)$$

où $t \in [0,T]$, $T > 0$ est la durée du phénomène, et les conditions initiales de l'inconnue $v(t)$ sont $v(t=0) = v_0$.

Tout d'abord, l'espace temps $[0,T]$ est partitionné en N intervalles réguliers de manière suivante : $[0,T] = \bigcup_n [t_{n-1}, t_n]$, Δt étant le pas de temps tel que $\Delta t = \frac{T}{N}$.

Le schéma temporel le plus simple est :

$$\frac{\partial v(t)}{\partial t} = \frac{v(t_n) - v(t_{n-1})}{\Delta t}$$

En approximant $v(t_n)$ par v^n, nous avons :

$$\frac{\partial v(t)}{\partial t} = \frac{v^n - v^{n-1}}{\Delta t}$$

Deux types de schéma d'Euler sont issus de cette approximation : Euler implicite et Euler explicite. Lorsque les conditions de convergence sont remplies, il est montré que ces deux schémas sont d'ordre 1. Cela veut dire que l'on peut majorer l'erreur due à l'approximation par rapport à la solution exacte par :

$\left| v(T) - v^n \right| \le C \frac{\Delta t}{T}$ quelque soit la valeur de la constante $C > 0$.

- Le schéma d'Euler explicite est :

$$\frac{v^n - v^{n-1}}{\Delta t} = f\left(v^{n-1}, t_{n-1}\right) \qquad \forall n \le N \tag{17}$$

La condition de stabilité de ce schéma est liée à la grandeur du pas de temps Δt, qui se trouve donc limité. Ce schéma implique clairement la linéarisation du problème de Cauchy. En l'appliquant au terme non linéaire *(15)* des équations de Navier-Stokes, nous obtenons $\frac{v - v^-}{\Delta t} = -\nabla v^- . v^-$, et :

$$\frac{v}{\Delta t} = \frac{v^-}{\Delta t} - \nabla v^- . v^- \tag{18}$$

où v^- est la vitesse calculée au pas de temps précédent.

Au final, la forme faible *(14)* du problème de Navier-Stokes linéarisé par un schéma d'Euler explicite s'écrit :

$$\begin{cases} \rho\left(\frac{v_h}{\Delta t}, w_h\right) + \left(2\eta\varepsilon\left(v_h\right) : \varepsilon\left(w_h\right)\right) - \left(p_h, \nabla \cdot w_h\right) = \left(\rho g, w_h\right) + \rho\left(\frac{v_h^-}{\Delta t}, w_h\right) - \rho\left(\nabla v_h^- . v_h^-, w_h\right) \\ \left(\nabla \cdot v_h, q_h\right) = 0 \end{cases} \tag{19}$$

où v_h^- est la solution trouvée pour approximer la vitesse au pas de temps précédent.

- Le schéma d'Euler implicite est :

$$\frac{v^n - v^{n-1}}{\Delta t} = f\left(v^n, t_n\right) \qquad \forall n \le N \tag{20}$$

L'avantage de ce schéma est qu'il est stable sous aucune condition. Le pas de temps Δt n'est donc pas limité. Cependant, un pas de temps trop grand peut causer des imprécisions importantes de part l'ordre du schéma (ordre 1). De plus, cette approche ne linéarise pas le problème par elle-même. Elle nécessite donc le traitement des termes $\nabla v.v$ qui apparaissent dans *(21)* :

$$\frac{v}{\Delta t} + \nabla v.v = \frac{v^-}{\Delta t} \tag{21}$$

où v^- est la vitesse calculée au pas de temps précédent.

2.4.2. Algorithme de Newton

Pour linéariser les termes $\nabla v.v$ non linéaires, nous appliquons une méthode de Newton qui consiste à négliger les termes du second ordre, et à faire itérer le système linéaire ainsi obtenu jusqu'à convergence. Pour cela, la vitesse v est substituée par $u+v-u$ où v est

l'inconnue à trouver à chaque itération de Newton, et u est la vitesse connue, calculée à l'itération précédente. Initialement, à la première itération de Newton, u a la valeur de la vitesse calculée à l'incrément de temps précédent. Pendant les itérations de Newton, lorsque v et u sont assez proches (c'est-à-dire égaux à une précision de 10^{-7}), on considère que le système a convergé, et que la solution du problème de Navier-Stokes est trouvée pour un temps t donné. Souvent, un nombre d'environ 5 itérations est nécessaire, mais cela dépend du problème, et du pas de temps choisi.

Voici le développement qui amène à la formule qu'il faut utiliser pour linéariser le terme $\nabla v.v$ par un algorithme de Newton :

$v = u + (v - u)$ où u est connu
$\nabla v.v = \nabla(u + (v - u)).(u + (v - u))$
$\nabla v.v = \nabla u.u + \nabla u.(v - u) + \nabla(v - u).u + \nabla(v - u).(v - u)$

Le terme du second ordre $\nabla(v - u).(v - u)$ est négligé. Il nous reste donc :
$\nabla v.v = \nabla(u + v - u).u + \nabla u.(v - u)$
$\nabla v.v = \nabla v.u + \nabla u.(v - u)$
$\nabla v.v = \nabla v.u + \nabla u.(v - u)$

Au final, nous avons :

$$\nabla v.v = \nabla v.u + \nabla u.v - \nabla u.u \qquad (22)$$

En substituant ce résultat dans la formule *(21)* du schéma d'Euler implicite, on obtient :

$$\frac{v}{\Delta t} + \nabla v.u + \nabla u.v = \frac{v^-}{\Delta t} + \nabla u.u \qquad (23)$$

Enfin, la forme faible du problème de Navier-Stokes linéarisé par un schéma d'Euler implicite et par un algorithme de Newton s'écrit :

$$\begin{cases} \rho\left(\frac{v_h}{\Delta t}, w_h\right) + \rho(\nabla v_h.u_h, w_h) + \rho(\nabla u_h.v_h, w_h) + (2\eta\varepsilon(v_h) : \varepsilon(w_h)) - (p_h, \nabla \cdot w_h) \\ \qquad = (\rho g, w_h) + \rho\left(\frac{v_h^-}{\Delta t}, w_h\right) + \rho(\nabla u_h.u_h, w_h) \\ (\nabla \cdot v_h, q_h) = 0 \end{cases} \qquad (24)$$

où v_h^- est la solution trouvée pour approximer la vitesse au pas de temps précédent, et u_h est l'approximation de la vitesse à l'itération de Newton précédent.
Un avantage important de cette formulation est qu'elle se montre plus robuste en terme de convergence et de pas de temps que celle avec un schéma d'Euler explicite. Le pas de temps dépend toujours de la nature du problème, mais il est moins limité.

2.4.3. Autres schémas temporels

Dans la littérature, on peut trouver d'autres schémas temporels comme ceux du type de Runge Kutta. Bien que nous n'implémentions pas de telles méthodes dans le cadre du solveur mécanique de Navier-Stokes, nous les présentons brièvement ici afin de terminer le tour d'horizon sur ce qu'il se fait couramment sur le sujet.

L'idée principale des ces schémas est la recherche de la solution de *(16)* en intégrant l'équation :

$$v(t_n) - v(t_{n-1}) = \int_{t_{n-1}}^{t_n} f(v(t),t)dt$$

Cette intégrale est, dans le cas générale, approchée par une formule faisant apparaître un paramètre θ : ce sont les θ-schémas.

$$\frac{v^n - v^{n-1}}{\Delta t} = \theta.f\left(v^n, t_n\right) + (1-\theta).f\left(v^{n-1}, t_{n-1}\right)$$

En choisissant $\theta = 1/2$, cela revient à la méthode des trapèzes d'ordre 2, aussi nommée schéma temporel semi implicite. Elle peut être vue comme la moyenne entre les schémas d'Euler implicite et explicite, eux-mêmes obtenues respectivement lorsque $\theta = 1$ et $\theta = 0$.

Pour linéariser ce type de problème, on peut par exemple construire un schéma appelé méthode de Heunn. Pour $\theta = 1/2$, il amène à résoudre le problème suivant :

$$\begin{cases} v^* = f\left(v^{n-1}, t_{n-1}\right) \\ v^{**} = f\left(v^{n-1} + \dfrac{\Delta t}{2} v^*, t_n\right) \\ v^n = v^{n-1} + \dfrac{\Delta t}{2}\left(v^* + v^{**}\right) \end{cases}$$

Les méthodes de Runge Kutta permettent aussi d'élaborer d'autres techniques qui atteignent des ordres plus élevés.

3. Stabilisations numériques

Bien que la forme continue *(13)* du problème de Navier-Stokes incompressible soit bien posée, la résolution de sa formulation discrète *(24)* peut ne pas converger vers la solution forte du problème, ou même présenter des oscillations. Une mauvaise gestion de la mixité du problème en vitesse-pression et une résolution non adaptée des termes d'advection sont les raisons principales des problèmes numériques rencontrés.

3.1. Stabilisation du problème mixte de Stokes

Commençons par étudier la difficulté apportée par la formulation de la méthode éléments finis mixtes. Dans ce cas, nous nous limitons dans un premier temps aux équations de Stokes (en négligeant les termes d'inertie) qui ne proposent pas de problématique supplémentaire, puisqu'elles sont stationnaires et linéaires. Rappelons ici les formulations fortes et discrètes du problème de Stokes :

$$\begin{cases} -\nabla\cdot\left(2\eta\varepsilon(v)\right)+\nabla p = f \\ \nabla\cdot v = 0 \end{cases} \qquad (25)$$

$$\begin{cases} \left(2\eta\varepsilon(v_h):\varepsilon(w_h)\right)-\left(p_h,\nabla\cdot w_h\right)=\left(f_h,w_h\right) \\ \left(\nabla\cdot v_h,q_h\right)=0 \end{cases} \qquad (26)$$

Pour assurer l'existence et l'unicité de la solution de ce problème, mais aussi que la solution du problème discret *(26)* converge vers celle du problème continu *(25)*, les espaces d'approximation doivent vérifier les conditions de Brezzi-Babuška [Babuška, 1973] et [Brezzi, 1974], aussi appelées conditions « inf-sup » :

$$\inf_{p_h\in P_h-\{0\}}\sup_{v_h\in V_h-\{0\}}\frac{\int_\Omega p_h\nabla\cdot v_h}{\|v_h\|_1\|p_h\|_0}\geq\gamma>0 \qquad (27)$$

La constante γ est indépendante de la taille de maille *h*, ce qui indique que le critère de convergence de la solution du problème discret ne dépend pas du maillage.

Cet aspect primordial pour les méthodes éléments finis permet de construire des « éléments compatibles », c'est-à-dire des interpolations et des espaces fonctionnels valides et appropriés au problème. Si on néglige le respect de cette condition, il est possible que la méthode ne converge pas, ou qu'elle converge vers une solution erronée [Aliaga, 2000].

Dans notre cas, la formulation mixte n'accepte pas d'interpolation linéaire simple pour à la fois la vitesse et la pression. En effet, l'élément P1/P1 ne vérifie pas les conditions inf-sup. Par contre, un moyen de construire des espaces mixtes stables est d'enrichir les espaces d'approximation de la vitesse par un espace de fonctions appelées « bulles ». Cela induit la création d'un élément P1+/P1 appelé le MINI-élément, et introduit dans [Arnold, 1984]. Ce

type de méthode a été implémenté au CEMEF dans [Coupez, 1996], [Coupez, 1997] et [Perchat, 2000] pour le problème de Stokes. Dans cette méthode, le champ de pression est linéaire et continu, tandis que la vitesse se décompose en une partie linéaire *v* et une partie bulle *b*. On défini l'espace d'interpolation des bulles de la manière suivante :

$$V_h^b = \left\{ b_h, b_{k|_K} \in P_k(K) \bigcap H_0^1(K) \quad \forall K \in \Omega_h \right\}$$

avec $P_k(K)$ l'espace des polynômes de degré *k* sur l'élément *K*.

En conséquence, le nouvel espace pour la vitesse est tel que :

$$\tilde{V}_h = V_h \oplus B_h$$

et, la vitesse est décomposée en deux parties distinctes :

$$\tilde{v}_h = v_h + b_h$$

Une des propriétés les plus importante de la bulle [Coupez, 1997] est son orthogonalité :

$$\forall K \in T_h(\Omega) \quad \int_K \varepsilon(v_h) : \varepsilon(b_h) d\Omega = 0 \quad \forall v_h \in P^1(K) \tag{28}$$

La fonction bulles vaut 1 au centre de l'élément et s'annule sur sa frontière. Une sous-discrétisation des éléments doit alors être créée :

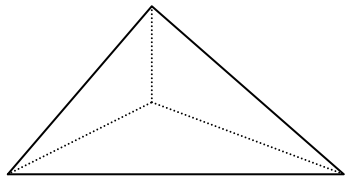


Figure 3 – Sous-discrétisation des éléments en 2D

Avec la décomposition de la vitesse, la forme faible du problème de Stokes devient :

$$\begin{cases} (2\eta\varepsilon(\tilde{v}_h) : \varepsilon(\tilde{w}_h)) - (p_h, \nabla \cdot \tilde{w}_h) = (f_h, \tilde{w}_h) \\ (\nabla \cdot \tilde{v}_h, q_h) = 0 \end{cases}$$

En substituant $\tilde{v}_h$ par $v_h + b_h$, nous avons :

$$\begin{cases} (2\eta\varepsilon(v_h) : \varepsilon(w_h)) + (2\eta\varepsilon(b_h) : \varepsilon(w_h)) - (p_h, \nabla \cdot w_h) = (f_h, w_h) \\ (2\eta\varepsilon(v_h) : \varepsilon(w_h^b)) + (2\eta\varepsilon(b_h) : \varepsilon(w_h^b)) - (p_h, \nabla \cdot w_h^b) = (f_h, w_h^b) \\ (\nabla \cdot v_h, q_h) + (\nabla \cdot b_h, q_h) = 0 \end{cases}$$

Puis, avec la propriété d'orthogonalité de la bulle *(28)*, on obtient le problème faible discret :

$$\begin{cases} \left(2\eta\varepsilon\left(v_h\right):\varepsilon\left(w_h\right)\right)-\left(p_h,\nabla\cdot w_h\right)=\left(f_h,w_h\right) \\ \left(2\eta\varepsilon\left(b_h\right):\varepsilon\left(w_h^b\right)\right)-\left(p_h,\nabla\cdot w_h^b\right)=\left(f_h,w_h^b\right) \\ \left(\nabla\cdot v_h,q_h\right)+\left(\nabla\cdot b_h,q_h\right)=0 \end{cases}$$

où $\left(v_h,b_h,p_h\right)\in\left(V_h,B_h,P_h\right)$ pour tout $\left(w_h,w_h^b,q_h\right)\in\left(V_h^0,B_h^0,P_h\right)$.

Ce qui amène localement au système algébrique suivant :

$$\begin{pmatrix} A_{vv} & 0 & A_{vp}^T \\ 0 & A_{bb} & A_{bp}^T \\ A_{vp} & A_{bp} & 0 \end{pmatrix}\begin{pmatrix} v_h \\ b_h \\ p_h \end{pmatrix}=\begin{pmatrix} b_v \\ b_b \\ b_p \end{pmatrix} \qquad (29)$$

Avec la sous-discrétisation des éléments (figure *3*), il est possible de calculer la bulle localement sur chaque élément :

$$b_h = A_{bb}^{-1}b_b - A_{bb}^{-1}A_{bp}^T p_h$$

Ensuite, avec une technique de condensation, elle peut être utilisée pour calculer v_h que l'on considère alors comme la vitesse calculable globalement sur le domaine discrétisé Ω_h. Ainsi, les degrés de liberté liés aux fonctions bulles sont éliminés, et seuls ceux placés sur les nœuds restent à être calculés :

$$\begin{pmatrix} A_{vv} & A_{vp}^T \\ A_{vp} & -C \end{pmatrix}\begin{pmatrix} v_h \\ p_h \end{pmatrix}=\begin{pmatrix} b_v \\ b_p' \end{pmatrix} \qquad (30)$$

où *C* est la matrice semi-définie positive, résultat de la condensation de la bulle telle que :

$$C = A_{bp}A_{bb}^{-1}A_{bp}^T \qquad (31)$$

Cette technique, appelé MINI-élément, est stable selon les conditions de Brezzi-Babuška. Elle est considérée comme une méthode mixte stabilisée, dans laquelle l'opérateur de stabilisation est obtenu par condensation des termes de bulles à l'intérieure de chaque élément. Les termes de nature elliptique rajoutés au problème préservent la consistance de celui-ci. Un avantage très important de cette approche est la possibilité d'utiliser un solveur itératif pour résoudre le système linéaire et symétrique *(30)*. Il est basé sur une méthode MINRES (*General Minimal Residual Method*) et se montre très efficace dans la plupart des cas. Enfin, elle se porte très bien aux applications massivement parallèles et à l'utilisation d'une grille de calcul.

3.2. Stabilisation des termes d'advection de Navier-Stokes

Les équations de Navier-Stokes apportent un problème supplémentaire à la résolution par rapport aux équations de Stokes. Ce problème est lié à la formulation discrète obtenue par une méthode éléments finis de type Galerkin standard. Dans sa forme variationnelle (*24)*, il

apparaît des termes d'advection $\nabla v.u$ qui se trouvent difficiles à traiter avec ce type de méthode. La raison principale vient du caractère hyperbolique non symétrique de ces termes. C'est ainsi que les méthodes éléments finis les plus générales de type Galerkin standard ne se portent pas bien aux problèmes dominés par un phénomène de convection. Un exemple tiré de la littérature est celui de la résolution des équations de convection-diffusion classiques. L'article [Codina, 1993] montre que l'utilisation du Galerkin standard crée des instabilités structurelles dès lors que les termes de convection prennent le dessus sur ceux de diffusion.

3.2.1. SUPG

Exactement le même problème sera rencontré lorsque l'on parlera dans le prochain chapitre de la résolution d'une équation de transport, et c'est dans cette partie que de plus amples informations à ce sujet seront présentées. Dans ce cas, une des techniques les plus répandues consiste à appliquer une stabilisation par SUPG (*Streamline upwind / Petrov-Galerkin*) qui donne un effet *upwind* à la discrétisation. Elle permet de traiter la non symétrie en décalant la fonction test en direction opposée à la vitesse de convection u, et la solution se retrouve beaucoup plus stable et ne laisse apparaître quasiment plus d'oscillations.

3.2.2. Introduction de la philosophie du *multiscale*

Une interprétation alternative au sujet de ces instabilités est de dire que les oscillations apparaissent à cause d'un phénomène particulier non résolu dans l'écoulement. Autrement dit, des variations à petites échelles – mais brutales – dans la solution ne peuvent pas être prises en compte par le maillage ; et, du fait de leur non résolution, elles polluent les grandeurs calculées. Cette réflexion introduit l'idée d'échelles « irrésolvables » numériquement en contraste avec les échelles visibles. Bien que l'on ne soit pas intéressé par leur résolution, de par leur petite taille, ces petites échelles ont un effet non négligeable sur les échelles, dites « résolvables », que l'on veut calculer. L'article [Brezzi, 1997] propose une technique appelée *multiscale* pour prendre en compte cet effet, et ne calculer que les grandes échelles au niveau globale.

3.3. Stabilisation du problème de Navier-Stokes

Comme nous venons de le souligner, la résolution du problème de Navier-Stokes incompressible propose deux difficultés majeures. La première est due à la mixité de la formulation en vitesse - pression et au respect des conditions de Brezzi-Babuška [Babuška, 1973] [Brezzi, 1974] ; et la deuxième vient du terme d'advection – introduit par la considération de l'inertie – qui n'est pas traité correctement par les méthodes éléments finis de type Galerkin standard.

3.3.1. SUPG

La stratégie la plus utilisée pour obtenir une formulation stable du problème de Navier-Stokes discret, est de séparer les deux difficultés : on utilise d'une part les méthodes de stabilisation du problème mixte créées pour la résolution du problème de Stokes, et d'autre

part les méthodes de stabilisation des problèmes d'advection diffusion. L'utilisation du MINI-élément – et de sa bulle – est largement reconnue pour stabiliser efficacement le problème mixte en vitesse - pression. Ensuite, il est possible d'utiliser une autre technique qui elle est destinée à stabiliser les oscillations apportées par les termes de convection. Par exemple, dans [Hétu, 1999] et dans [Codina, 1993], une méthode de type SUPG est appliquée pour traiter les termes d'advection.

3.3.2. Méthode de projection

Afin de mieux connaître les différentes solutions rencontrées dans la littérature au sujet de la stabilisation du problème Navier-Stokes incompressible, nous présentons brièvement une technique de projection souvent utilisée. L'article [Guermond, 1996] présente une méthode nommée Chorin-Temam qui sert à imposer l'incompressibilité. Elle est fréquemment utilisée dans le contexte des approximations par éléments finis. Elle consiste à décomposer la résolution en deux étapes distinctes : la première résulte en une équation de convection-diffusion, et la deuxième correspond à la projection sous la forme d'un problème de Darcy.

Rappelons la formulation forte du système de Navier-Stokes incompressible :

$$\begin{cases} \rho \dfrac{\partial v}{\partial t} + \rho \nabla v.v - \nabla \cdot \left(2\eta \varepsilon(v)\right) + \nabla p = \rho g \\ \nabla \cdot v = 0 \end{cases}$$

La première étape voit la résolution de ce problème de convection-diffusion :

$$\rho \frac{\hat{v} - v^{-}}{\Delta t} + \rho \nabla \hat{v}.v^{-} - \nabla \cdot \left(2\eta \varepsilon(\hat{v})\right) = \rho g$$

où v^{-} est le champ de vitesse connu calculé à l'incrément en temps précédent, et $\hat{v}$ est un champ auxiliaire qui demande à être projeté vers l'espace fonctionnel approprié. Des techniques comme celle de SUPG introduite précédemment suffisent pour stabiliser numériquement de problème.

Puis, la deuxième étape est la projection de $\hat{v}$ sous forme d'une équation de Darcy :

$$\begin{cases} \rho \dfrac{v - \hat{v}}{\Delta t} + \nabla p = 0 \\ \nabla \cdot v = 0 \end{cases}$$

Une technique de stabilisation performante est introduite dans [Masud, 2002]. Mais aussi, l'ajout d'une fonction bulles qui rappelle celle utilisée par le MINI-élément montre de bons résultats avec l'apparition d'un paramètre numérique équivalent à une viscosité de bulle.

3.3.3. Les bulles vues par le *multiscale* (*Residual-Free Bubbles*)

Ces dernières années, des études ont été menées pour trouver les liens théoriques entre toutes les méthodes standard de stabilisation. On peut retenir que l'équivalence entre les méthodes stabilisées pour le problème de Stokes et l'utilisation des fonctions bulles a été démontrée, tout comme celle entre les méthodes de stabilisation du problème de convection-diffusion avec les méthodes dites de *multiscale* qui traitent les sous échelles du problème, et finalement l'utilisation des fonctions bulles [Hughes, 1998].

Voyons ici comment les fonctions bulles utilisées pour stabiliser le problème mixte de Stokes peut jouer un rôle dans la stabilisation des équations de Navier-Stokes incompressible. Le principal avantage d'une telle méthode est qu'elle présente une solution unifiée autour de l'utilisation des fonctions bulles et qui allège donc l'implémentation du solveur dans l'optique de l'utiliser sur une grille de calcul pour réaliser des gros calculs.

C'est en faisant un rapprochement avec les méthodes de type *multiscale* que la bulle peut jouer un rôle direct dans la stabilisation des équations de Navier-Stokes. Présentons ici les grandes lignes des méthodes *multiscale* pour ensuite les comparer aux stabilisations par bulles.

La philosophie du *multiscale* s'appuie sur l'hypothèse que les instabilités dans la résolution du problème de Navier-Stokes peuvent être résolues en prenant en compte les effets des petites échelles sur les grandes [Brezzi, 1997]. Ainsi, pour la vitesse ($\tilde{v}_h$), les échelles que l'on peut résoudre (v_h) sont à distinguer des échelles que l'on ne résous pas (b_h) :

$$\tilde{v}_h = v_h + b_h$$

Pour cette raison, on introduit les espaces V_h et B_h tels que :

$$\tilde{V}_h = V_h \oplus B_h$$

Ainsi, $b_h \in B_h$ représente les perturbations des petites échelles, et $v_h \in V_h$ est la vitesse du fluide débarrassée de celles-ci. Par conséquence, les instabilités disparaissent dans la résolution de v_h. Les méthodes numériques qui partent du *multiscale* pour résoudre des problèmes à l'aide de fonctions bulles sont appelées *Residual-Free Bubbles*.

Pour présenter ces méthodes de type *Residual-Free Bubbles*, considérons le problème variationnel :

$$\begin{cases} \text{trouver } \tilde{v}_h \in \tilde{V}_h \text{ tel que :} \\ \left(L(\tilde{v}_h), \tilde{w}_h\right) = \left(f, \tilde{w}_h\right) \quad \forall \tilde{w}_h \in \tilde{V}_h^0 \end{cases}$$

Avec la décomposition des échelles, nous avons :

$$\begin{cases} \left(L(v_h), w_h\right) = \left(f, w_h\right) & \forall w_h \\ \left(L(b_h), w_h^b\right) = \left(f, w_h^b\right) & \forall w_h^b \end{cases}$$

Au niveau des petites échelles (*subgrid scales*), la solution est obtenue localement sur les éléments en résolvant l'équation :

$$\begin{cases} \left(L(b_h), w_h^b\right) = \left(f, w_h^b\right) - \left(L(v_h), w_h^b\right) & \forall w_h^b \\ b_h = 0 \quad \text{sur } \partial K \end{cases}$$

Le principe des méthodes *Residual-Free Bubbles* est de résoudre ce problème sur chaque élément, de sorte à « condenser » les termes de bulles :

$$b_h = M_b\left(L(v_h) - f\right)$$

où M_b est un opérateur intégral, inverse de la forme linéaire.
Il suffit maintenant de substituer ce dernier résultat dans l'équation des grandes échelles pour obtenir le système suivant où seule les échelles v_h sont inconnues :

$$\left(L(v_h), w_h\right) + \sum_K \left(M_b\left(L(v_h) - f\right), L^*(w_h)\right) = \left(f, w_h\right) \qquad (32)$$

Nous pouvons voir ces dernières étapes comme une condensation de bulle.

La brève présentation des méthodes *multiscale* [Hughes, 1998] montre que les bulles issues de la formulation du MINI-élément peuvent jouer le même rôle que les petites échelles prisent en charge par b_h. Dans les deux cas, la bulle et les petites échelles résultent de la décomposition de la vitesse de par l'enrichissement de son espace fonctionnel. Nous choisissons donc de stabiliser le problème de Navier-Stokes entièrement avec les fonctions bulles utilisées dans le MINI-élément pour les équations de Stokes, similairement au *multiscale*. Cependant, la généralisation de cette méthode nécessite le contrôle de la norme L^2 de la bulle pour être mieux adaptée à ce type de problème.

3.3.4. Correction de la bulle pour stabiliser le problème de Navier-Stokes

La conservation de la masse requière un contrôle de la norme L^2 de la bulle, en contraste avec la stabilisation classique pour les équations de Stokes.

$$\tilde{v}_h = v_h + b_h$$

Si on considère le problème discret *(24)* de Navier-Stokes incompressible avec seulement les termes de bulles qui stabilisent les équations de Stokes *(25)*, nous avons :

$$\begin{cases} \left(\rho \dfrac{dv_h}{dt}, w_h\right) + \left(2\eta\varepsilon(v_h) : \varepsilon(w_h)\right) - \left(p_h, \nabla \cdot w_h\right) = \left(\rho g, w_h\right) \\ \left(2\eta\varepsilon(b_h) : \varepsilon(b_h^*)\right) - \left(p_h, \nabla \cdot b_h^*\right) = \left(\rho g, b_h^*\right) \\ \left(\nabla \cdot v_h, q_h\right) + \left(\nabla \cdot b_h, q_h\right) = 0 \end{cases} \qquad (33)$$

Le problème avec cette formulation est qu'elle suppose que $(\nabla \cdot v_h)$ et $(-\nabla \cdot b_h)$ sont égaux. Seulement, dans le cas d'écoulements turbulents – et donc de nombre de Reynolds élevés –le terme $(-\nabla \cdot b_h)$ pour la bulle ne s'annule pas, et donc, force le terme $(\nabla \cdot v_h)$ pour la vitesse à être différent de zéro. En conséquence, le terme d'incompressibilité $(\nabla \cdot v_h = 0)$ n'est plus vérifiée pour l'échelle de la vitesse que l'on calcul. Et, plus le nombre de Reynolds est élevé, moins l'incompressibilité est respectée.

Après avoir corrigé la stabilisation pour les équations de Navier-Stokes, nous avons à résoudre le système suivant :

$$\begin{cases} \left(\rho \dfrac{dv_h}{dt}, w_h\right) + \left(2\eta\varepsilon(v_h) : \varepsilon(w_h)\right) - \left(p_h, \nabla \cdot w_h\right) = \left(\rho g, w_h\right) \\ \left(\rho \dfrac{db_h}{dt}, b_h^*\right) + \left(2\eta\varepsilon(b_h) : \varepsilon(b_h^*)\right) - \left(p_h, \nabla \cdot b_h^*\right) = \left(\rho g, b_h^*\right) \\ \left(\nabla \cdot v_h, q_h\right) + \left(\nabla \cdot b_h, q_h\right) = 0 \end{cases} \qquad (34)$$

Le terme de bulles à rajouter ici est difficile à calculer rigoureusement, et nous choisissons donc de l'approximer de façon suivante :

$$\left(\rho \frac{db_h}{dt}, b_h^*\right) \approx \frac{\rho}{\Delta t} \qquad (35)$$

La correction apportée par l'ajout de ce dernier terme à la bulle suffit pour obtenir une méthode stabilisée performante pour le problème incompressible de Navier-Stokes à partir de la fonction bulles généralement utilisée pour stabiliser le problème de Stokes avec le MINI-élément. Dans la pratique, cette formulation donne de bien meilleurs résultats en ce qui concerne la conservation de la masse.

3.4. Conclusion

L'intérêt des méthodes de stabilisation avec des fonctions bulles présentées, est de traiter les problèmes de stabilité liés au traitement des termes d'advection en s'appuyant sur des approximations compatibles pour le problème mixte. L'idée de départ vient des méthodes dites *multiscale* qui distinguent les échelles perturbatrices aux « grandes » échelles que l'on est intéressé de calculer au niveau global.

A partir des liens qui existent entre les différentes méthodes, on vérifie qu'il est possible, sous réserve de traitements supplémentaires, de ne pas utiliser deux ingrédients différents pour traiter les deux problèmes de stabilité évoqués dans ce paragraphe. Les estimateurs d'erreur de ces méthodes ont notamment l'avantage d'être généraux. On trouve, dans la littérature, des auteurs, qui résolvent le problème de Navier-Stokes discret en faisant « d'une pierre deux coups ». Par exemple, dans [Nechaev, 2002], pour une application en océanographie, les auteurs génèrent une fonction bulle optimale par la méthode de *Residual-Free Bubbles*, qui

leur permet d'utiliser des éléments compatibles et de résoudre en même temps le problème de convection-diffusion. Un autre exemple est donné par l'utilisation d'une méthode de sous-échelles (*subgrid scale*) [Codina, 2002] appliquée à des cas tests standards (par exemple la cavité entraînée qui est un cas que nous étudierons dans la suite de ce chapitre).

En plus de la description des problèmes de stabilité liés à la résolution en espace du problème de Navier-Stokes, la description des principaux schémas en temps nous a permis de rendre compte de l'ensemble des types de problèmes à traiter dans sa mise en œuvre numérique. Nous avons également décrit plusieurs méthodes de linéarisation de ce problème.

Notre objectif, au cours de cette étude, est d'obtenir un outil utilisable sur grille de calcul, c'est-à-dire, un solveur entièrement parallélisé, qui n'est pas trop gourmant en place mémoire, et qui simplifie au maximum la résolution de problèmes complexes d'écoulement de fluides visqueux. Bien que les techniques les plus souvent utilisées sont différentes pour traiter le problème mixte et les termes de convection, nous choisissons une méthode unifiée qui traite ces deux problématiques ensembles avec des fonctions bulles qui rappellent le *multiscale*. Ainsi, nous adaptons l'élément P1+/P1 compatible – généralement utilisé pour stabiliser les équations de Stokes – au problème de Navier-Stokes incompressible. La discrétisation temporelle, quant à elle, est basée sur un schéma d'Euler implicite linéarisé avec un algorithme de Newton qui offre une bonne robustesse et une meilleure liberté sur le pas de temps.

4. Formulation matricielle et résolution du problème de Navier-Stokes

Rappelons que, conformément à la mise en place des formulations du paragraphe précédent, une méthode éléments finis de type P1+/P1 (appelé MINI-élément) est utilisée pour approcher les champs de vitesse et de pression $(v,p)\in(V,P)$ par les inconnues $(v_h,p_h)\in(V_h,P_h)$. Cette méthode s'appuie sur une discrétisation spatiale de la cavité Ω sous forme de maillage de tétraèdres $\Omega_h = \bigcup_{K\in T_h(\Omega)} K$. Dans ce domaine de calcul, la pression et la pression sont interpolées linéairement, et une bulle est ajoutée pour enrichir l'espace fonctionnelle des vitesses par V_h^b. La formulation mixte du MINI-élément remplie les conditions de Brezzi-Babuška qui assurent la stabilité.

Dans le cadre d'une résolution non stationnaire des équations de Navier-Stokes incompressibles, le temps est discrétisée avec un pas de temps Δt. Ainsi, une résolution est nécessaire par incrément de temps. Les deux techniques de linéarisation présentées dans les paragraphes précédents se résolvent de façons très différentes de par leurs caractéristiques. A partir de leur formulation matricielle, nous présentons des solutions performantes pour les résoudre.

4.1. Traitement explicite de la convection

Dans le schéma temporel explicite d'Euler, les termes d'advection, qui sont difficiles à résoudre, apparaissent explicitement dans le second membre. Deux conséquences très importantes résultent de ce fait sur le système à résoudre : il est à fois linéaire et symétrique. Il s'apparente donc à un problème de Stokes, et les techniques de résolution peuvent être les mêmes. Rappelons la formulation discrète du problème de Navier-Stokes quasi-explicite, qui contient un schéma temporel explicite sur la convection :

$$\begin{cases} \rho\left(\frac{v_h}{\Delta t}, w_h\right) + \left(2\eta\varepsilon(v_h) : \varepsilon(w_h)\right) - \left(p_h, \nabla\cdot w_h\right) = \left(\rho g, w_h\right) + \rho\left(\frac{v_h^-}{\Delta t}, w_h\right) - \rho\left(\nabla v_h^- . v_h^-, w_h\right) \\ \left(\nabla\cdot v_h, q_h\right) = 0 \end{cases} \qquad (36)$$

Par rapport au problème de Stokes, le terme $\rho\left(\frac{v_h}{\Delta t}, w_h\right)$ est rajouté à la matrice locale A_{vv}, et les termes $\rho\left(\frac{v_h^-}{\Delta t}, w_h\right) - \rho\left(\nabla v_h^- . v_h^-, w_h\right)$ sont rajoutés au second membre B_v. Pour les termes de bulles servant à la stabilisation $\frac{\rho}{\Delta t}$ est ajouté à A_{bb} (se référer au paragraphe traitant de la formulation du MINI-élément) :

$$\begin{pmatrix} A_{vv} & 0 & A_{vp}^T \\ 0 & A_{bb} & A_{bp}^T \\ A_{vp} & A_{bp} & 0 \end{pmatrix} \begin{pmatrix} v_h \\ b_h \\ p_h \end{pmatrix} = \begin{pmatrix} B_v \\ B_b \\ B_p \end{pmatrix} \qquad (37)$$

où :

$$A_{vv} = \rho\left(\frac{v_h}{\Delta t}, w_h\right) + \left(2\eta\varepsilon(v_h) : \varepsilon(w_h)\right)$$

$$A_{vp} = \left(\nabla \cdot v_h, q_h\right)$$

$$A_{bb} = \left(\frac{\rho}{\Delta t}, b_h^*\right) + \left(2\eta\varepsilon(b_h) : \varepsilon(b_h^*)\right)$$

$$A_{bp} = \left(q_h, \nabla \cdot b_h\right)$$

$$B_v = \left(\rho g, w_h\right) + \rho\left(\frac{v_h^-}{\Delta t}, w_h\right) - \rho\left(\nabla \cdot v_h^- . v_h^-, w_h\right)$$

$$B_p = 0$$

$$B_b = \left(\rho g, b_h^*\right)$$

La condensation de la bulle se fait de la même manière que dans le paragraphe précédent qui traite du MINI-élément. Et, finalement, ce système linéaire et symétrique doit être résolu :

$$\begin{pmatrix} A_{vv} & A_{vp}^T \\ A_{vp} & -C \end{pmatrix} \begin{pmatrix} v_h \\ p_h \end{pmatrix} = \begin{pmatrix} B_v \\ B'_p \end{pmatrix} \qquad (38)$$

où la condensation de la bulle donne $C = A_{bp} A_{bb}^{-1} A_{bp}^T$.

Cette matrice est symétrique mais non définie positive puisque la matrice locale -*C* est semi-définie négative. La résolution d'un tel problème est courante en mécanique des fluides, et plusieurs méthodes efficaces, comme celle du gradient conjugué ou du résidu minimal, sont disponibles [Perchat, 2000], accompagnées par des techniques de préconditionnement qui permettent une meilleure résolution.

Le code de calcul CimLib, dans lequel le solveur Navier-Stokes est implémenté, fait appel à PETSc (*Portable, Extensible Toolkit for Scientific Computation*) [PETSc, 2004]. C'est une bibliothèque composée d'un ensemble de procédures permettant de résoudre aussi bien en séquentiel qu'en parallèle des équations aux dérivées partielles et des problèmes d'algèbre linéaire par des méthodes numériques itératives. Elle propose donc des solutions intéressantes dans le domaine des calculs scientifiques parallèles. Plus précisément, des méthodes entièrement parallélisées sont disponibles dans le cadre de la résolution de systèmes linéaires, incluant des préconditionneurs et des méthodes de résolution de type Krylov. La création des matrices locales est réalisée par CimLib, sous forme de solveur local. Ensuite, elles sont prisent en charge par PETSc pour le préconditionnement et la résolution dont quelques méthodes sont brièvement présentées ici.

La méthode itérative du résidu minimal, appelée MINRES, est la méthode de Krylov équivalente au gradient conjugué pour des problèmes symétriques non définis positifs. Nous

pouvons donc l'utiliser à travers la librairie PETSc pour résoudre le système linéaire *(38)* qui résulte du problème de Navier-Stokes incompressible. Plusieurs types de préconditionnement sont applicables ici. En général, les préconditionneurs diagonaux et bloc diagonaux sont intéressant de part leur simplicité et leur facilité de parallélisation. De plus, ils sont peu coûteux à assembler et à inverser. La méthode de Jacobi par bloc se montre efficace lorsque le problème à résoudre est proche de celui de Stokes.

En conclusion, ce problème de Navier-Stokes quasi-explicite peut être relativement simple à résoudre car il résulte en un système linéaire symétrique. La méthode de résolution MINRES proposée par PETSc, associée à un préconditionneur bloc diagonal, est choisie et montre de bons résultats. Néanmoins, le schéma quasi-explicite utilisé ici ne se montre pas très robuste, surtout dans la définition du pas de temps. Dès que l'écoulement simulé se montre assez turbulent, le pas de temps se trouve très limité, ce qui implique un nombre très important d'incréments en temps, et donc un grand temps de calcul. Et même, dans certain cas, la convergence du système n'est plus possible, quelque soit la taille du pas de temps.

4.2. Méthode implicite de Newton

Précédemment, nous voyons que la formulation discrète du problème de Navier-Stokes linéarisé par un schéma temporel implicite et un algorithme de Newton est :

$$\begin{cases} \rho\left(\frac{v_h}{\Delta t}, w_h\right) + \rho\left(\nabla v_h . u_h, w_h\right) + \rho\left(\nabla u_h . v_h, w_h\right) + \left(2\eta\varepsilon\left(v_h\right) : \varepsilon\left(w_h\right)\right) - \left(p_h, \nabla \cdot w_h\right) \\ \qquad = \left(\rho g, w_h\right) + \rho\left(\frac{v_h^-}{\Delta t}, w_h\right) + \rho\left(\nabla u_h . u_h, w_h\right) \\ \left(\nabla \cdot v_h, q_h\right) = 0 \end{cases} \qquad (39)$$

La formulation matricielle équivalente est :

$$\begin{pmatrix} A_{vv} & 0 & A_{vp}^T \\ 0 & A_{bb} & A_{bp}^T \\ A_{vp} & A_{bp} & 0 \end{pmatrix} \begin{pmatrix} v_h \\ b_h \\ p_h \end{pmatrix} = \begin{pmatrix} B_v \\ B_b \\ B_p \end{pmatrix} \qquad (40)$$

où :

$$A_{vv} = \rho\left(\frac{v_h}{\Delta t}, w_h\right) + \rho\left(\nabla v_h . u_h, w_h\right) + \rho\left(\nabla u_h . v_h, w_h\right) + \left(2\eta\varepsilon\left(v_h\right) : \varepsilon\left(w_h\right)\right)$$

$$A_{vp} = \left(\nabla \cdot v_h, q_h\right)$$

$$A_{bb} = \left(\frac{\rho}{\Delta t}, b_h^*\right) + \left(2\eta\varepsilon\left(b_h\right) : \varepsilon\left(b_h^*\right)\right)$$

$$A_{bp} = \left(q_h, \nabla \cdot b_h\right)$$

$$B_v = (\rho g, w_h) + \rho\left(\frac{v_h^-}{\Delta t}, w_h\right) + \rho(\nabla u_h . u_h, w_h)$$
$$B_p = 0$$
$$B_b = (\rho g, b_h^*)$$

Et finalement, après condensation de la bulle, ce système linéaire et symétrique doit être résolu :

$$\begin{pmatrix} A_{vv} & A_{vp}^T \\ A_{vp} & -C \end{pmatrix} \begin{pmatrix} v_h \\ p_h \end{pmatrix} = \begin{pmatrix} B_v \\ B'_p \end{pmatrix} \qquad (41)$$

où C vient de la condensation de la bulle avec $C = A_{bp} A_{bb}^{-1} A_{bp}^T$. Les détails de la stabilisation par bulle sont développés précédemment.

Ce problème implicite propose une résolution différente de celle vue précédemment pour plusieurs raisons. Premièrement, l'algorithme de linéarisation de Newton consiste à effectuer plusieurs itérations (appelés itérations de Newton) pour faire converger le système *(41)* ; et deuxièmement, la non symétrique du problème apportée par A_{vv} interdit l'usage de la méthode de résolution MINRES.

Rappelons que la solution du système *(39)* est atteinte lorsque les champs *u* et *v* sont suffisamment proches (égaux à une précision de 10^{-7}). Ainsi, pour un incrément de temps, plusieurs itérations de Newton sont nécessaires (généralement, quelques unes suffisent). Chacune de ces itérations représente la résolution d'un système linéaire. Les méthodes de résolution de systèmes linéaires disponibles dans PETSc peuvent donc encore être utilisées ici.

Par contre, MINRES n'est plus valide dans ce cas. En effet, les termes de convection $\rho(\nabla v_h . u_h, w_h) + \rho(\nabla u_h . v_h, w_h)$ entraînent la non symétrie du système. Une autre méthode de résolution telle que le résidu minimal généralisé doit donc être utilisée. Cette technique nommée GMRES [Saad, 1986] se montre très performante dans beaucoup d'applications, mais elle est forcément plus lourde que MINRES, l'hypothèse de symétrie des matrices n'étant plus valide. Les préconditionneurs évoqués précédemment sont toujours applicables. Mais, d'autres techniques par factorisation incomplète sont aussi utilisables lorsqu'il s'agit de simulation d'écoulement plus turbulent, et notamment la méthode de Cholesky incomplet dite ILU(*k*). Cette dernière se montre très robuste et stable, même lorsque la nature du problème à résoudre entraîne un mauvais conditionnement de la matrice. Elle a, notamment, la particularité de permettre une adaptation avec le paramètre *k* qui défini le niveau de remplissage de la matrice creuse. L'influence de ce préconditionneur est un aspect important de nos travaux, surtout dans le cadre de l'utilisation d'une grille de calcul. En effet, élever le niveau *k* aide à la convergence de la résolution itérative des systèmes linéaires : plus *k* est grand, moins d'itérations sont nécessaires pour converger vers la solution. Cependant, cela implique une demande plus importante de mémoire vive. A l'aide plusieurs applications, une étude est menée dans cette thèse sur l'utilisation du préconditionneur ILU(*k*). Il est montré qu'en 2D, ILU(4) se montre le plus rapide en parallèle sur un cluster ; tandis qu'en 3D, c'est ILU(1) qui semble le plus adapté.

Bien que la résolution du problème de Navier-Stokes implicite semble plus lourde que dans le cas explicite à cause de la non linéarité et de la non symétrie, cette technique montre une bien meilleure robustesse. Le pas de temps est largement moins limité, et des écoulements de fluides plus complexes et plus turbulents peuvent être simulés. Au final, c'est celle-là que nous adapterons. Les cas tests du paragraphe suivant montrent à quel point il est plus intéressant d'utiliser cette formulation implicite, et surtout lorsque l'on parle d'écoulements à hauts Reynolds.

4.3. Conclusion

Les méthodes décrites jusqu'à présent sont implémentées en C++ dans le logiciel CimLib qui vise à offrir un ensemble complet de solveurs sous forme de librairie servant à la simulation numérique dans le cadre de mise en forme des matériaux. Afin de pouvoir simuler des grosses applications, ce solveur Navier-Stokes utilise des méthodes numériques simples avec des interpolations linéaires pour les degrés de liberté. Il s'appuie sur la librairie MPI pour la parallélisation du code, et sur l'outil PETSc pour résoudre les systèmes linéaires. L'implémentation permet la simulation d'applications aussi bien en 2 dimensions qu'en 3 dimensions, sans changer pour autant les méthodes numériques.

La partie suivante présente une validation de ce code de calcul, ainsi qu'une étude numérique, à partir de cas tests (*benchmarks*) couramment utilisés dans la littérature pour évaluer les méthodes de résolution du problème de Navier-Stokes.

5. Applications

5.1. Validations

5.1.1. Présentation du cas test de la Cavité Entraînée

Le cas test que nous allons étudier dans ce paragraphe est très couramment utilisé dans la littérature pour valider et évaluer les méthodes de résolution du problème de Navier-Stokes ([Ghia, 1982], et [Erturk, 2005]). Il n'existe pas de solution analytique à ce problème, et la solution de référence utilisée, est une solution obtenue par une méthode multigrille implicite basée sur une formulation en fonction de courant et vortex [Ghia, 1982]. Cette référence fournit des solutions pour des nombres de Reynolds allant jusqu' à 10000 sur des grilles (129x129) et (257x257).

On considère une cavité carrée en 2 dimensions, remplie de fluide newtonien. On applique une vitesse constante sur le plan supérieur de la cavité, en cisaillement, dans la direction *Ox*. En condition limite, une vitesse nulle est imposée sur les autres parois de la cavité (*figure 4*). Avec notre solveur instationnaire de Navier-Stokes incompressible, on simule les mouvements du fluide entraîné dans la cavité jusqu'à ce qu'il se stabilise. Ensuite, on étudie le champ de vitesse après qu'il ait atteint un état stationnaire. On considère généralement qu'il est atteint lorsque les variations de la vitesse sont inférieures à 10^{-6} m.s^{-1} :

$$\left| v - v^{-} \right| < 10^{-6} \quad \text{m.s}^{-1}$$

Puisque notre logiciel CimLib est applicable aussi bien pour des calculs 2D que 3D, aucune difficulté ne s'oppose à la réalisation de ce cas test avec un maillage 2D de triangles.

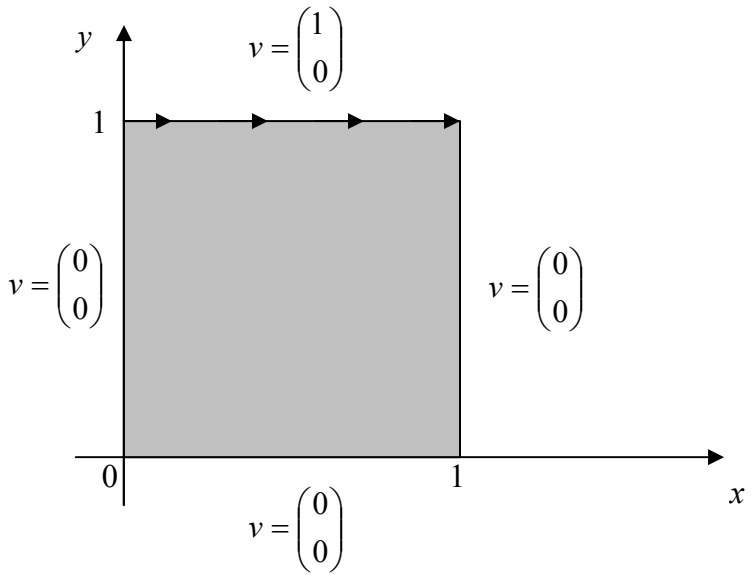


Figure 4 – Cas test de la cavité entraînée

Les résultats que l'on trouve dans la littérature dépendent du nombre de Reynolds associé à l'écoulement. Rappelons que ce nombre adimensionnel caractérise l'influence des termes d'inertie sur un écoulement visqueux :

$$\mathrm{Re} = \frac{\rho U L}{\eta}$$

où U et L sont la vitesse et la longueur caractéristiques, et ρ et η sont la masse volumique et la viscosité du fluide. Plus le nombre de Reynolds est élevé, plus il y a de turbulence.

Dans notre cas, les paramètres sont fixés à :
$U = 1 \ \mathrm{m.s^{-1}}$
$L = 1 \ \mathrm{m}$
$\rho = 1 \ \mathrm{kg.m^{-3}}$

Ainsi, il suffit de faire varier la viscosité η du fluide pour atteindre les valeurs du nombre de Reynolds souhaitées.

5.1.2. Solutions de référence

Dans la littérature, les tests effectués avec la cavité en traînée sont généralement pour des écoulements visqueux très turbulents, avec des nombre de Reynolds allant de 100 jusqu'à 20000. Au-delà de cette valeur, il est signalé que l'écoulement devient périodique, et par conséquent, aucun état stationnaire ne peut être atteint, même après une très longue simulation. L'article de référence lorsque l'on parle du cas test de la cavité entraînée est [Ghia, 1982]. Il montre les résultats que l'on est censé avoir avec des nombres Reynolds allant jusqu'à 10000 ; et notamment, il donne les valeurs des deux composantes de la vitesse sur des points fixés sur les deux axes médians de la cavité. Pour visualiser les solutions, les lignes de courant, qui représentent la trajectoire du fluide, sont tracées. De cette manière, les recirculations qui caractérisent ce type d'écoulement turbulent sont mises en évidence. On obtient les lignes de courant d'un écoulement avec les isovaleurs de la fonction courant ψ définie à partir du champ v des vitesses :

$$\begin{cases} \dfrac{\partial \psi}{\partial x} = v_y \\ \dfrac{\partial \psi}{\partial y} = -v_x \end{cases}$$

Cette définition montre qu'en tout point, la tangente à ψ est colinéaire au vecteur vitesse.

La publication [Erturk, 2005] plus récente obtient de très bons résultats en comparaison avec le premier, mais, cette fois-ci, avec des nombres de Reynolds allant jusqu'à 20000. De plus, celui-ci met à disposition des résultats plus complets, et montre avec précision l'emplacement où l'on doit trouver des recirculations dans l'écoulement.

Nous nous servons des informations tirées de la littérature pour évaluer nos résultats obtenus par notre de code de calcul CimLib, et pour valider nos méthodes numériques. Autrement dit, les résultats présentés dans [Ghia, 1982] et [Erturk, 2005] nous servent de solutions de référence que l'on veut reproduire.

5.1.3. Résultats numériques

Afin de réaliser une étude complète, nous choisissons les viscosités de fluides suivantes : 10^{-3}, 2.10^{-4}, 10^{-4}, et 5.10^{-5} Pa.s pour obtenir des nombres de Reynolds variés : 1000, 5000, 10000, et 20000. Dans chacun de ces cinq cas, un maillage assez fin à 50000 nœuds est utilisé (ce qui correspond à une taille de maille de 0.005 mètre) pour que les phénomènes physiques complexes soient bien captés pendant la simulation.

Les résultats de notre code de calcul montrés ici sont obtenus après stabilisation de l'écoulement. Avec notre solveur de Navier-Stokes instationnaire, plusieurs incréments en temps sont donc nécessaires. Nous considérons que l'état stationnaire est atteint lorsque le champ des vitesses a convergé avec une précision de 10^{-6}. Seulement, plus la viscosité est faible, plus l'écoulement est turbulent et prend du temps pour se stabiliser. Avec le même pas de temps de 0.1 seconde, le nombre d'incréments en temps varie donc en fonction du nombre de Reynolds. Ceci n'est pas le seul aspect variable entre les différents étudiés. En effet, plus l'écoulement est turbulent, plus le problème devient complexe à résoudre. Une conséquence directe est la nécessité d'un nombre plus important d'itérations de Newton pour résoudre le problème de Navier-Stokes à chaque incrément de temps. Par exemple, pour Re=1000, 4 itérations de Newton sont effectués en moyenne, tandis que pour Re=10000, ce sont 6 itérations de Newton qui sont nécessaires en moyenne. Et même, il devient parfois impossible de converger vers la solution sans baisser le pas de temps. C'est pourquoi, pour Re=33333, il devient obligatoire de baisser le pas de temps à 0.05 secondes. Le tableau suivant résume les différences de pas de temps et de durée de simulation pour chacun des nombres de Reynolds étudiés pour que l'état stationnaire soit atteint :

Reynolds	1 000	5 000	10 000	20 000	33 333
Nombre d'incréments	333	940	1205	1621	∞
Pas de temps (s)	0.1	0.1	0.1	0.1	0.05

Tableau 1 – Nombre d'incréments nécessaires pour atteindre l'état stationnaire

Les figures suivantes comparent les lignes de courant entre nos résultats et celles des solutions de référence de [Ghia, 1982] et [Erturk, 2005] pour les différents nombres de Reynolds choisis.

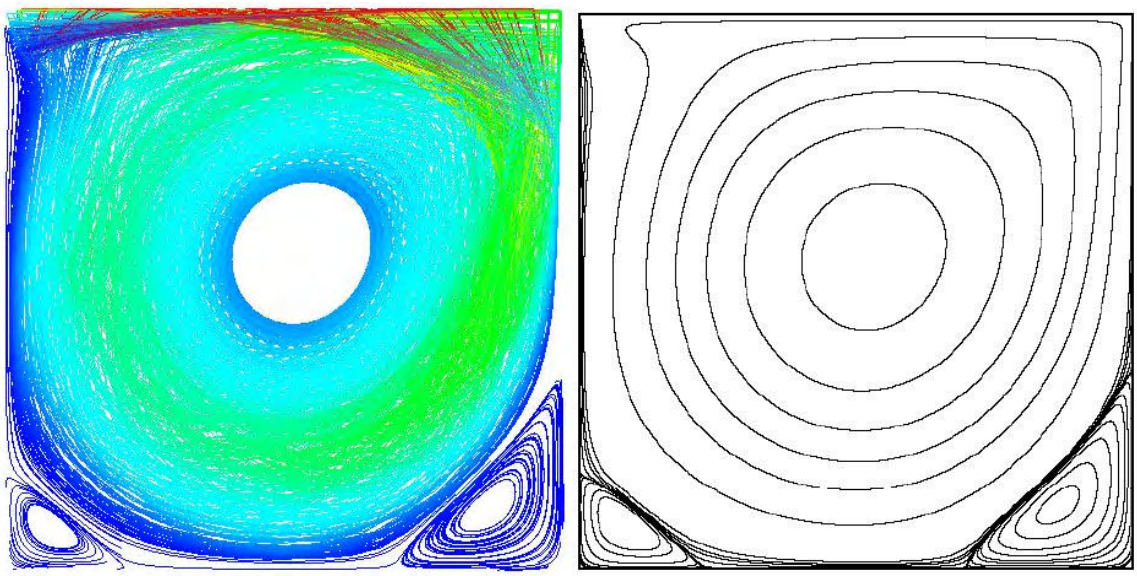


Figure 5 – Lignes de courant pour Re = 1000

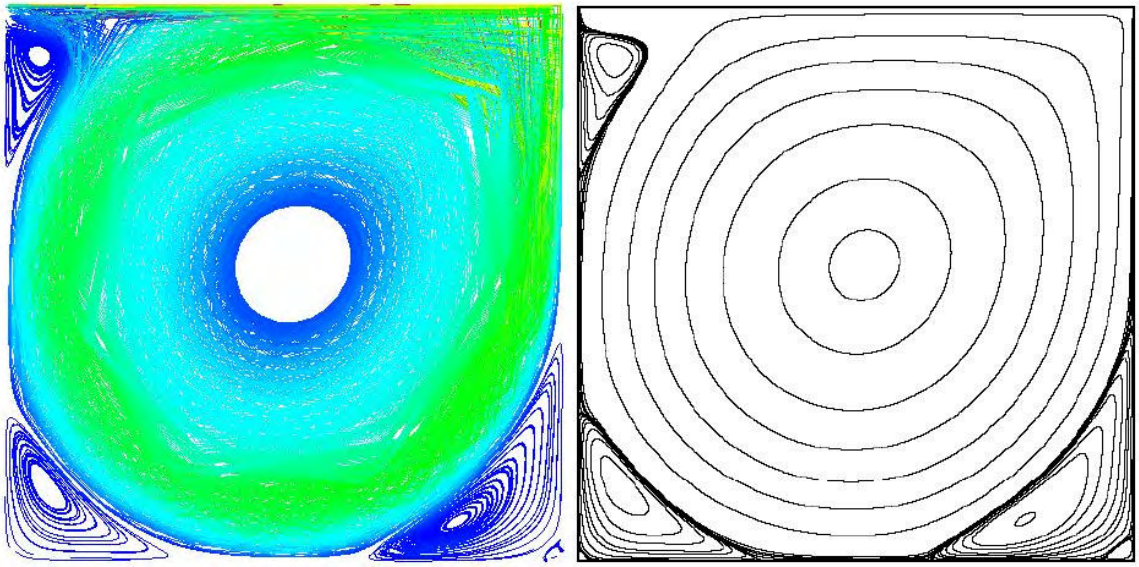


Figure 6 – Lignes de courant pour Re = 5000

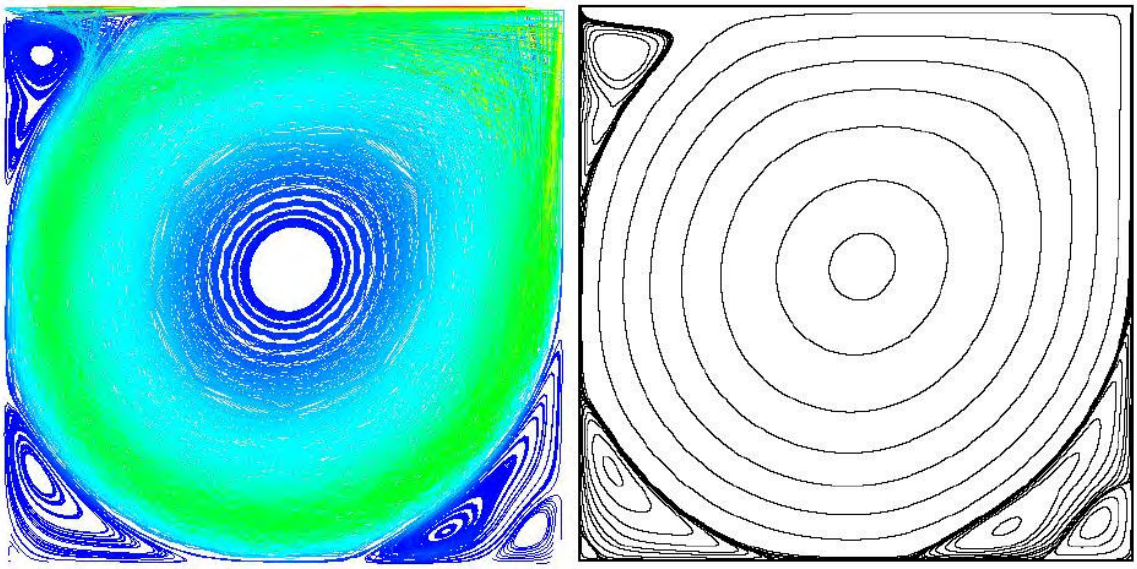


Figure 7 – Lignes de courant pour Re = 10000

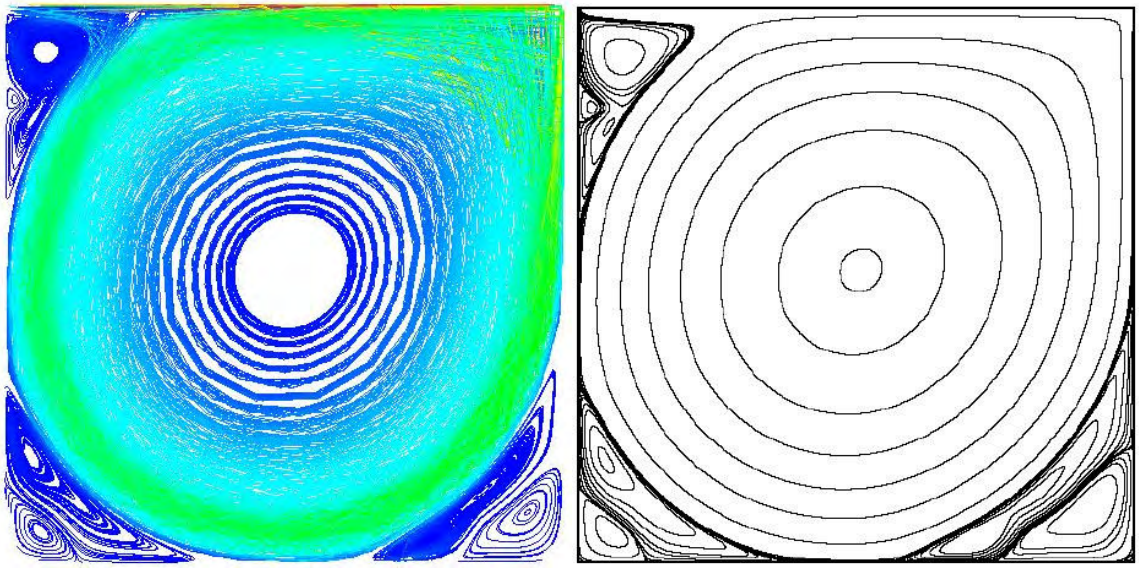


Figure 8 – Lignes de courant pour Re = 20000

Notre première remarque est que, pour chacun des cas, l'allure des lignes de courant obtenue est très proche du cas de référence (figures de droite). On peut ainsi observer les effets dus à la prise en compte de l'inertie, tels que la position décalée vers la droite de la recirculation principale, et la formation de recirculations dans les angles de la cavité. Plus précisément, ces recirculations semblent avoir des formes, positions et tailles correctes, en comparaison avec les solutions de références.

De façon générale, lorsque le nombre de Reynolds augmente, le centre de la recirculation principale se déplace vers le centre de la cavité, et des recirculations secondaires apparaissent de plus en plus nombreuses, tout en s'écrasant et en se resserrant sur la recirculation centrale.

Puisque ces résultats sont qualitativement corrects, nous cherchons maintenant un moyen pour quantifier leur précision par rapport aux solutions de référence. Les articles [Ghia, 1982] et [Erturk, 2005] mettent à disposition des mesures précises des composantes v_x et v_y de la vitesse sur des points placés sur les axes $x = 0.5$ et $y = 0.5\,\text{m}$. Nous reproduisons donc la mesure de ces grandeurs afin de pouvoir calculer les différences entre nos résultats et ceux que l'on trouve dans la littérature. Les figures suivantes montrent les courbes de v_x et v_y en fonction des coordonnées *y* et *x* des capteurs, et les points ronds correspondent aux solutions de références :

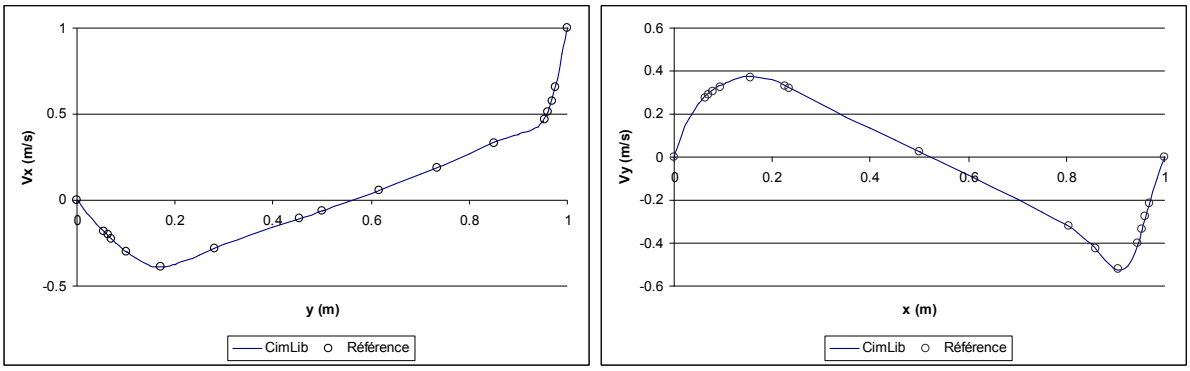


Figure 9 – Vitesses v_x sur $x = 0.5$ et v_y sur $y = 0.5$ pour Re = 1000

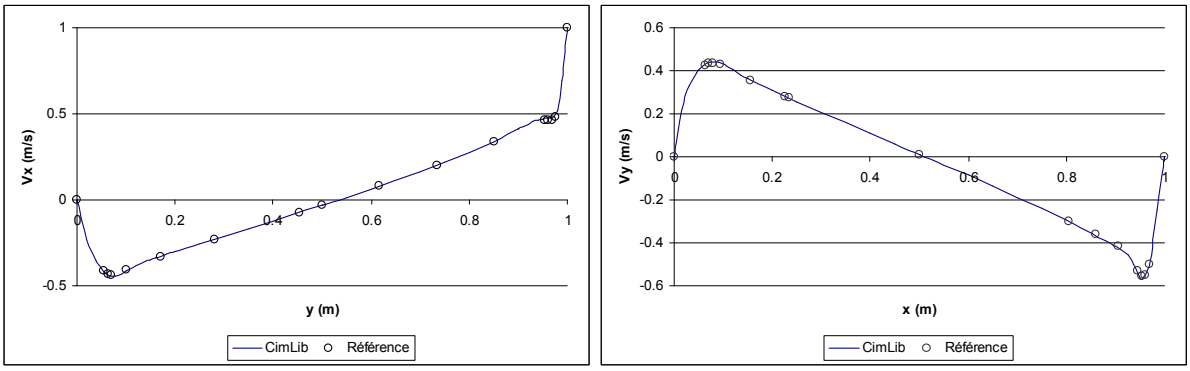


Figure 10 – Vitesses v_x sur $x = 0.5$ et v_y sur $y = 0.5$ pour Re = 5000

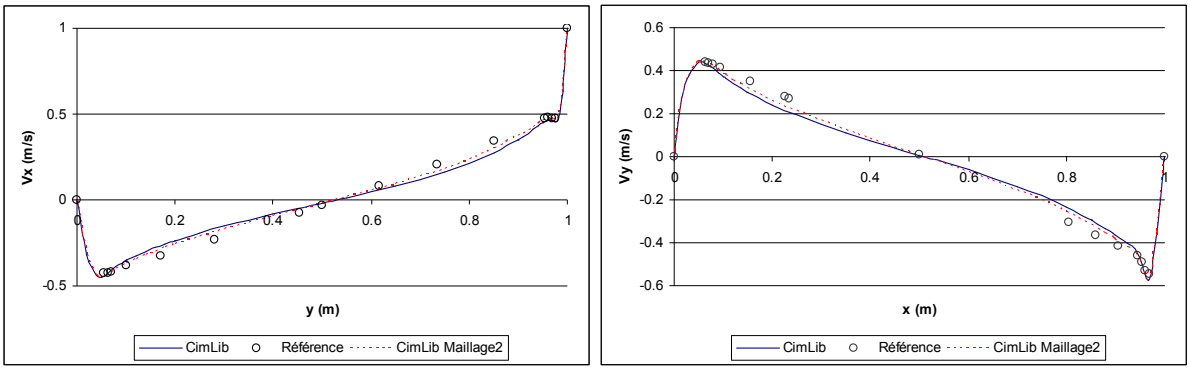


Figure 11 – Vitesses v_x sur $x = 0.5$ et v_y sur $y = 0.5$ pour Re = 10000

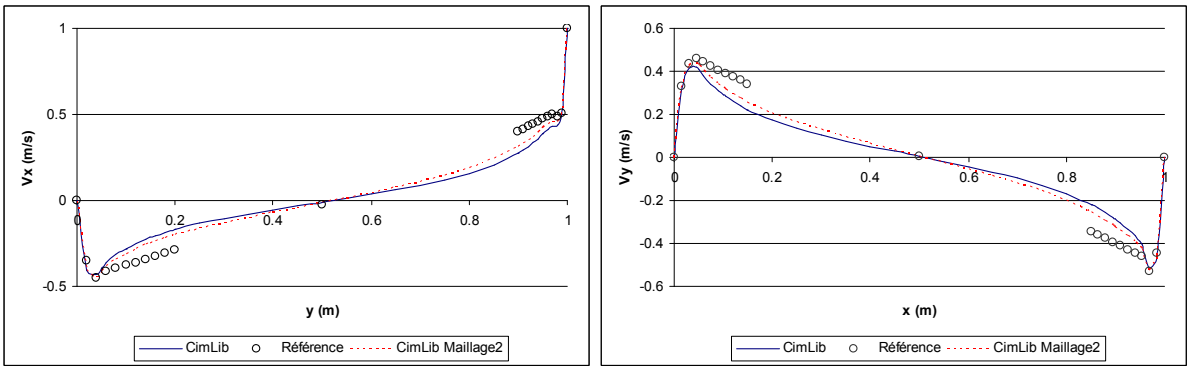


Figure 12 – Vitesses v_x sur $x = 0.5$ et v_y sur $y = 0.5$ pour Re = 20000

On constate qu'au fur et à mesure que le nombre de Reynolds augmente, l'écart entre les courbes (nos résultats) et les points (solutions de référence) augmente. Ceci traduit le fait que notre calcul converge vers la solution de référence, mais aussi que la taille de maille choisie est de moins en moins adaptée à l'écoulement étudié : plus le nombre de Reynolds est élevé, plus le maillage doit être fin pour obtenir de bons résultats. Le maillage à 50000 nœuds utilisé ici est assez fin pour que le comportement global du fluide corresponde à celui observé dans la littérature, mais les courbes de la vitesse montrent quand même une diminution de la précision lorsque le nombre de Reynolds dépasse 5000. C'est au niveau du milieu de la cavité (illustré par les *figures 13* et *14*) que notre calcul semble ne pas donner de gradients de vitesse assez réguliers lorsque la taille de maille n'est pas assez petite. Pour vérifier que cette imprécision vient bien d'un manque de raffinement du maillage, les cas avec des Reynolds de 10000 et 20000 sont réexécutés avec un autre maillage plus fin à 100000 nœuds (avec une taille de maille de 0.0035m). Ce test montre en effet de meilleurs résultats : les courbes en pointillés rouge sont plus proches de la référence que celles obtenues avec le maillage à 50000 nœuds. Même si il existe encore une différence, la forme de la courbe que nous obtenons est plus satisfaisante, et l'on peut légitimement penser que l'utilisation d'un maillage encore plus fin nous permettrait d'obtenir des résultats beaucoup plus proches du cas de référence. Rappelons d'ailleurs que pour ce dernier cas, la solution de référence a été calculée avec un maillage régulier et encore plus fin à 361201 nœuds (601x601).

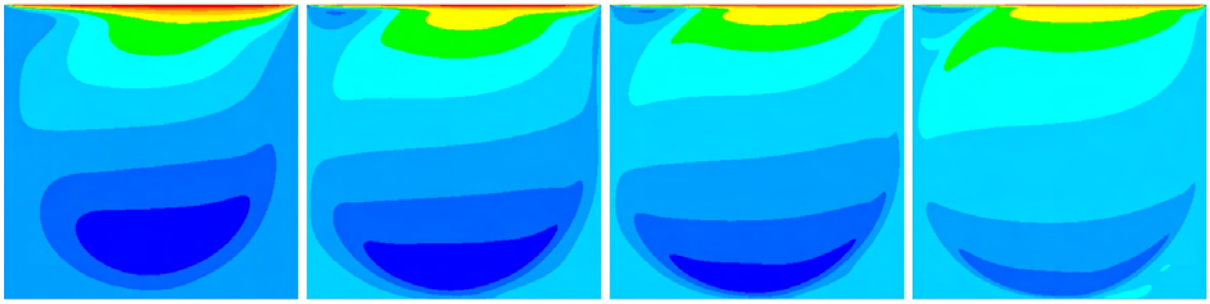


Figure 13 – Isovaleurs de la composante v_x pour Re = 1000, 5000, 10000, 20000

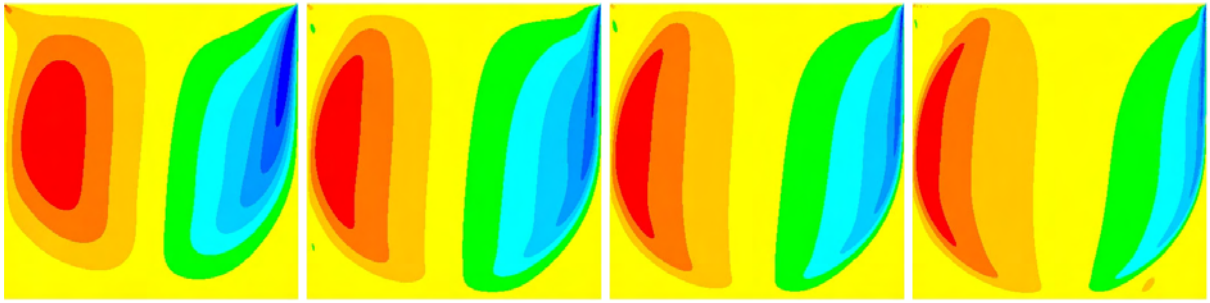


Figure 14 – Isovaleurs de la composante v_y pour Re = 1000, 5000, 10000, 20000

Pour finir sur l'évaluation de nos résultats, nous noterons que pour un Reynolds de 10000, l'article [Erturk, 2005] indique la position des recirculations dans la cavité. Nous les comparons donc à nos résultats :

	Référence		CimLib	
Recirculation	x	y	x	y
1	0.5117	0.5300	.513	.531
2	0.7767	0.0600	.786	.058
3	0.9350	0.0667	.947	.056
4	0.0583	0.1633	.058	.165
5	0.0717	0.9117	.067	.912

Tableau 2 – Positions des recirculations pour Re = 10000

Voici à quoi correspondent les numéros des recirculations : 1 est la recirculation principale centrale, 2 est la grande recirculation en bas à droite, 3 est la petite recirculation en bas à droite, 4 est la grande recirculation en bas à gauche, et 5 est la petite recirculation en haut à gauche. Bien que la mesure de ces grandeurs soit peu précise, nous voyons encore ici une bonne correspondance entre nos résultats et la solution de référence pour un nombre de Reynolds élevé.

5.2. Robustesse du code de calcul

Afin d'évaluer la robustesse de notre de code de calcul, nous testons le cas de la cavité entraînée avec des nombres de Reynolds très élevés. Bien qu'il soit indiqué dans plusieurs publications qu'au delà de 21000, le problème devient périodique et donc ne converge jamais

vers une solution stationnaire, il est intéressant d'apprécier le comportement de nos méthodes numériques avec des écoulements extrêmement turbulents.

Nous choisissons donc une viscosité de 3.10^{-5} pour le fluide afin d'atteindre un nombre de Reynolds de 33333. Dans ce cas, le pas de temps doit être baissé à 0.05 seconde pour que la résolution se passe bien. Après 8000 incréments de temps, soit une simulation de 400 secondes, l'écoulement n'est toujours pas stabilisé. La figure suivante montre l'état de la simulation à $t = 400$ secondes :

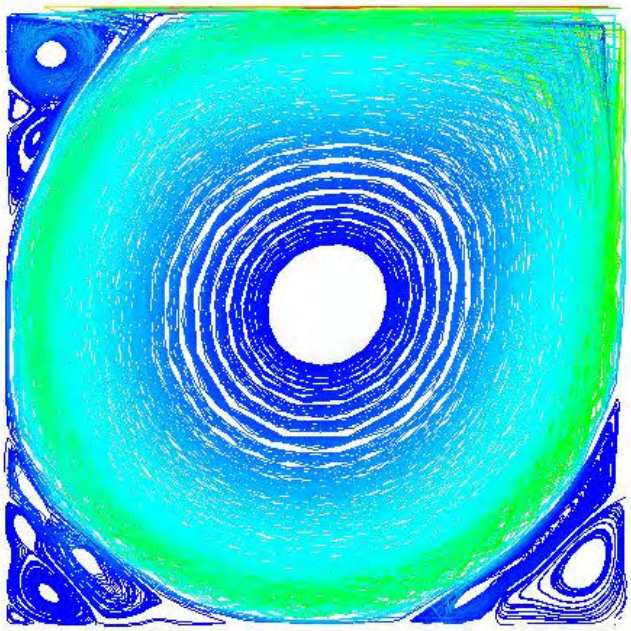


Figure 15 – Lignes de courant non stabilisées pour Re = 33333

Bien qu'aucun état stationnaire ne soit atteint, la simulation se déroule correctement pour un nombre de Reynolds très élevé. Cela montre la robustesse de notre solveur, mais les résultats ne peuvent pas être évalués puisque aucune solution de référence n'existe pour ce cas là.

5.3. Étude de la convergence

Nous avons vu précédemment que pour un Reynolds de 20000, les résultats sont bien meilleurs avec un maillage à 100000 nœuds qu'avec celui de 50000 nœuds. Afin de mieux comprendre ce phénomène, nous étudions ici la convergence de notre code calcul en fonction de la taille de maille.

On étudie l'écoulement ayant un nombre de Reynolds de 5000 en faisant varier la taille de maille des maillages (*figure 16*) avec $h = 0.1$, $h = 0.05$, $h = 0.025$, $h = 0.0125$, et $h = 0.00625$ mètre. Cela correspond à des maillages ayant respectivement 143, 529, 2035, 8200, et 34015 nœuds.

Figure 16 – Les différents maillages utilisés

Sur les figures *(17)* et *(18)* sont tracées les composantes v_x et v_y de la vitesse sur les axes $x = 0.5$ et $y = 0.5\,\text{m}$. Nous remarquons que plus le maillage est fin, plus la courbe est proche de la solution de référence publiée dans [Erturk, 2005]. Cela montre premièrement que la solution devient de plus en plus précise lorsque la taille de maille diminue, mais aussi que nos méthodes permettent une bonne prise en compte des termes d'inertie pour des nombres de Reynolds élevés.

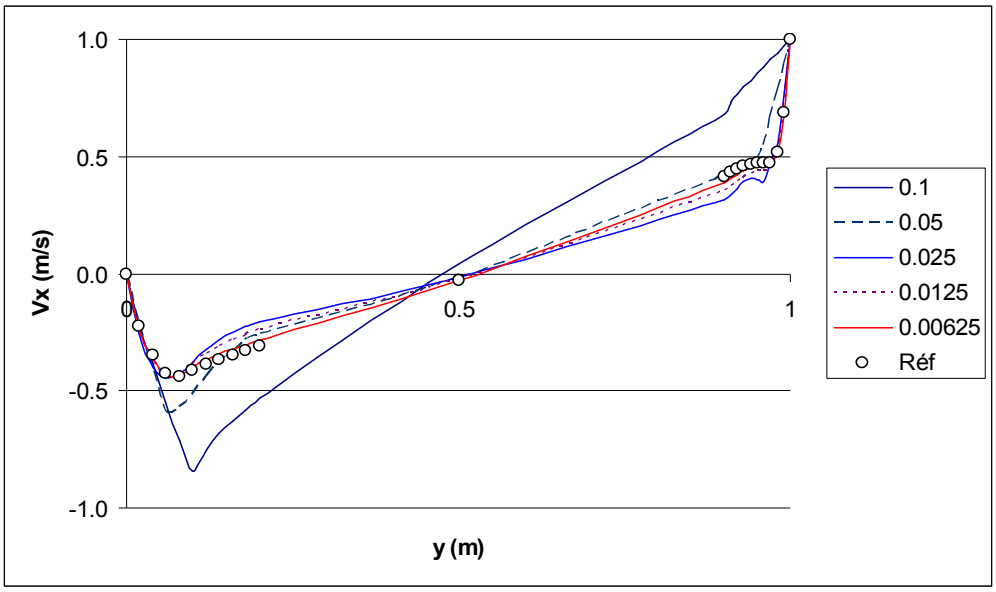


Figure 17 – Composante x de la vitesse sur l'axe x=0.5m pour différentes tailles de maille

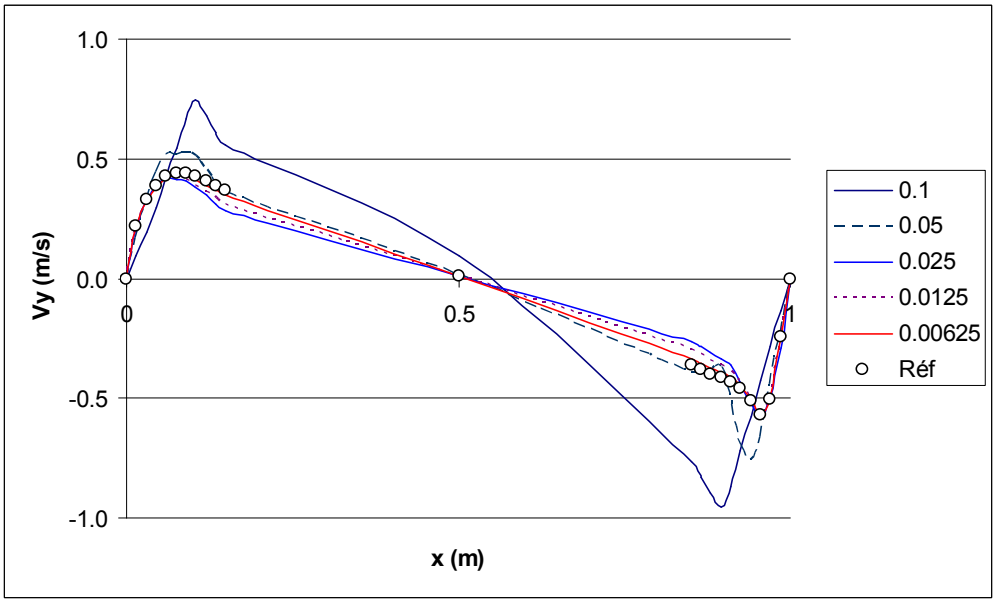


Figure 18 – Composante y de la vitesse sur l'axe y=0.5m pour différentes tailles de maille

Les différences que l'on obtient avec les différents maillages illustre l'influence qu'à celui-ci sur les résultats. De cette manière, nous pouvons observer la vitesse de convergence vers la solution de référence.

Pour donner une meilleure idée de l'influence du maillage sur les résultats obtenus, nous calculons la différence entre les vitesses calculées sur les axes $x = 0.5$ et $y = 0.5$ et celles de la solution de référence de [Erturk, 2005], et ce, avec les cinq maillages de la figure *(16)*. La différence entre ces deux solutions est définie comme suit, et elle est considérée comme l'erreur de nos résultats :

$$Erreur(h) = \sqrt{\sum_{x,y} \left(v_i^{ref} - v_i^{cal} \right)^2}$$

où v_i^{cal} et v_i^{ref} sont, respectivement, les vitesses calculées par notre code de simulation et les vitesses de références tirées de [Erturk, 2005] sur les axes médians de la cavité.

Ensuite, nous pouvons tracer sur la figure *(19)* l'erreur obtenue en fonction de la taille de maille *h* utilisée. Il s'agit ici d'étudier la convergence de notre schéma :

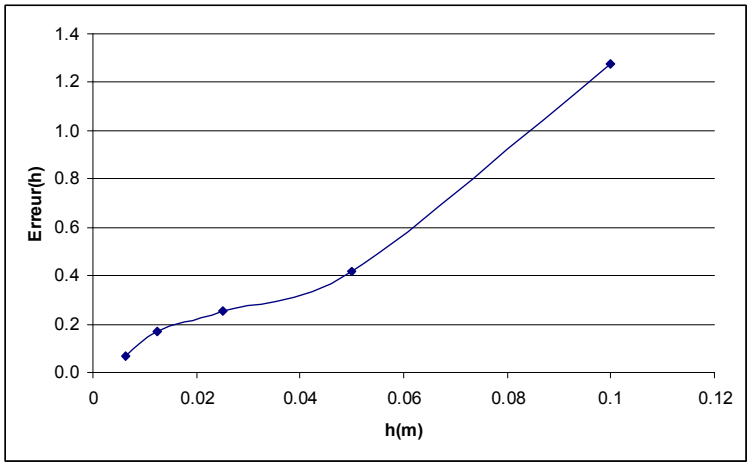


Figure 19 – Evolution de l'erreur L^2 en fonction de la taille de maille

Cette courbe retranscrit une amélioration très nette des résultats lorsque la taille de maille du maillage diminue. Cependant, la spécificité de ce cas test ne permet pas d'obtenir un profil de convergence propre. Lorsque l'on veut simuler des écoulements turbulents, les résultats ont une grande dépendance au maillage car, sous l'effet de l'inertie, les recirculations qui se forment peuvent être plus ou moins bien captés par le maillage. Par exemple, avec une taille de maille de 0.1 mètre, il n'est pas possible de capter une recirculation de la même taille. Deux maillages différents peuvent donc donner des résultats complètement opposés. Par conséquent, même si le résultat que nous donnons ici n'est pas exactement l'ordre de la méthode, il permet de vérifier que celui-ci n'est pas dégradé de manière notable par le traitement des termes d'inertie par rapport au solveur de Stokes sur lequel s'appuie notre méthode pour résoudre le problème de Navier-Stokes.

5.4. Comparaison avec le schéma temporel d'Euler explicite

Afin de comparer les schémas temporaires d'Euler explicite et d'Euler implicite linéarisé par un algorithme de Newton, nous exécutons le cas test de la cavité entraînée avec un Reynolds de 1000 avec chacun des deux modèles. L'étude de la convergence menée précédemment indique qu'une taille de maille de 0.00625 mètre suffit pour obtenir une bonne précision. Nous choisissons donc un maillage à 34015 nœuds qui est censé pouvoir capter correctement les phénomènes qui se déroule dans la cavité.

Rappelons que l'utilisation d'un schéma temporel explicite sur la convection linéarise par lui-même le problème de Navier-Stokes incompressible, et qu'en plus, il permet de garder le caractère symétrique du système linéaire afin de le résoudre avec MINRES. Par contre, avec un schéma temporel implicite, il est nécessaire de rajouter une autre technique pour linéariser le système. Un algorithme de Newton permet de réaliser cela, mais il demande la résolution de plusieurs itérations de Newton pour calculer un incrément de temps. De plus, les systèmes linéaires résultant de cette méthode ne sont plus symétrique, et doivent donc être résolu avec GMRES.

A première vue, il est moins lourd d'appliquer un schéma temporel explicite, mais il reste beaucoup moins robuste qu'un schéma implicite, et surtout lorsqu'il s'agit du pas de temps. Ainsi, alors que l'on peut choisir un pas de temps de 0.1 seconde dans le cas implicite, il n'est pas possible de prendre un pas de temps supérieur à 0.005 secondes dans le cas explicite. Par conséquent, avec cette méthode, le coût de la résolution d'un incrément en temps est inférieur, mais il en faut beaucoup plus à cause du petit pas de temps. Sur le tableau suivant, nous pouvons voir les coûts de résolution par incrément dans les deux cas, puis le temps total de calcul pour une simulation de 1 seconde :

Schéma temporel	Explicite	Implicite
Pas de temps (s)	0.005	0.1
Temps par incrément (s)	10.15	56.20
Nombre d'incréments	200	10
Résolution totale (s)	2029	562

Tableau 3 – Temps de calcul total et par incrément

Les résultats numériques obtenus par ces deux méthodes sont sensiblement identiques, mais leurs utilisations sont très différentes.

En conclusion, le cas de la cavité entraînée avec un nombre de Reynolds de 1000 met en évidence l'avantage porté par un schéma temporel implicite linéarisé par un algorithme de Newton par rapport à un autre schéma explicite. Le coût de calcul pour un incrément de temps y est 5.5 fois supérieur, mais 20 fois moins d'incréments en temps sont nécessaires. Au final, utiliser la méthode de Newton est 3.6 fois plus rapide qu'une méthode explicite. Et même, des tests similaires à celui-ci pour des nombres Reynolds encore supérieures montrent que même en baissant le pas de temps le méthode explicite ne converge plus. Elle n'est donc pas assez robuste pour simuler des écoulements complexes, et c'est le schéma d'Euler implicite accompagné de l'algorithme de Newton que nous adoptons.

6. Conclusion

Au cours de ce chapitre, nous avons présenté, puis validé, une méthode de résolution du problème de Navier-Stokes incompressible basée sur les éléments finis mixtes stabilisés avec des fonctions bulle qui rappellent les techniques de type *multiscale* et de *Residual-Free Bubbles*. Un schéma temporel d'Euler implicite associé à un algorithme de Newton pour linéariser le problème s'avère performant, même dans le cas de simulations très complexes.

Le solveur instationnaire que nous avons implémenté en C++ dans CimLib est entièrement parallélisée avec la librairie MPI. Il fait appelle aux algorithmes optimisés et parallèles de l'outil de calcul PETSc pour préconditionner et résoudre les systèmes linéaires.

Les validations numériques réalisées au travers du cas test de la cavité entraînée tirée de la littérature montrent que nos méthodes numériques permettent de simuler efficacement des écoulements complexes de fluides newtoniens aux nombres de Reynolds allant au-delà de 20000. Cela montre bien que, pour les applications que l'on projette de réaliser, il n'est pas nécessaire d'ajouter un modèle de turbulence généralement utilisé pour simuler des écoulements à hauts Reynolds.

Enfin, la robustesse de ces techniques ainsi que leur simplicité (avec notamment des interpolations spatiales linéaires) font que des applications à grande échelle sont envisageables en parallèle et sur grille de calcul.

Références

[Aliaga, 2000] C. Aliaga, "Simulation numérique par éléments finis en 3D du comportement thermomécanique au cours du traitement thermique d'aciers : application à la trempe de pièces forgées ou coulées", *Thèse de doctorat*, Ecole Nationale supérieure des Mines de Paris (2000).

[Arnold, 1984] D.N. Arnold, F. Brezzi, et M.Fortin. "A stable finite element for stokes equations. Calcolo", 21, 337-344 (1984).

[Babuška, 1973] I. Babuška, "The finite element method with penalty" *Math. Comp.*, 27, 221-228 (1973).

[Batkam, 2002] S. Batkam. "Thermique multidomaine en simulation numérique du remplissage 3D", *Thèse de doctorat*, Ecole Nationale supérieure des Mines de Paris (2002).

[Brezzi, 1974] F. Brezzi, "On the existence, uniqueness and approximation of saddle point problems arising from Lagragian multipliers", *RAIRO Modélisation Mathématique Analyse Numérique*, 8, 129-151 (1974).

[Brezzi, 1997] F.Brezzi, L.P. Franca, T.J.R. Hugues, A.Russo. "$b = \int g$", *Computer methods in applied mechanics and engineering*, 145, 329-339 (1997).

[Codina, 1993] R. Codina. "A finite Element Formulation for the Numerical Solution of Convection-Diffusion Equation", *CIMNE Monograph* (1993).

[Codina, 2002] R. Codina. "Stabilized finite element approximation of transient incompressible flows using orthogonal subscales". *Computer methods in applied mechanics and engineering,* 191, 4295-4321 (2002).

[Coupez, 1991] T. Coupez, "Grandes déformations et remaillage automatique", *Thèse de doctorat*, École Normale Supérieure des Mines de Paris (1991).

[Coupez, 1996] T. Coupez, "Stable-stabilized finite element for 3D forming calculation. CEMEF", rapport interne (1996)

[Coupez, 1997] T. Coupez and S. Marie, "From a direct solver to a parallel iterative solver in 3D forming simulation". *International Journal of Supercomputer and Applications*, 11:205 (1997).

[Coupez, 2000] T. Coupez, "Génération et adaptation de maillage par optimisation locale". *La revus européenne des éléments finis*, 9(4), 403-423 (2000).

[Erturk, 2005] E. Erturk, T.C. Corke and C. Gokcol, "Numerical Solutions of 2-D Steady Incompressible Driven Cavity Flow at High Reynolds Numbers", *International Journal for Numerical Methods in Fluids* (2005)

[Ghia, 1982] U. Ghia, K. N. Ghia, T. Shin. "High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and Multigrid Method". *Journal of computational Physics*, 48, 387-411 (1982).

[Gruau, 2003] C. Gruau, T. Coupez, "Anisotropic and multidomain mesh automatic generation for viscous flow finite element method.", *International Conference on Adaptative and Modeling Simulation* (ADMOS 2003), Barcelone (2003)

[Guermond, 1996] J.-L. Guermond. "Some implementations of projection methods for Navier-Stokes equations". *Mathematical Modelling and Numerical Analysis*, 30(5):637-667 (1996).

[Hughes, 1998] T. J.R. Hughes, G. R. Feijoo, L. Mazzei, J.-N. Quincy. "The variational multiscale method – a paradigm for computational mechanics". *Computer methods in applied mechanics and engineering*, 166, 3-24 (1998).

[Hétu, 1999] J.-F. Hetu. F. Ilinca. "A finite element method for casting simulations. Numerical Heat Transfer", Part A, 36, 657-679 (1999).

[Magnin, 1994] B. Magnin. « Modélisation du remplissage des moules d'injection pour les polymères thermoplastiques par une méthode eulérienne-lagrangienne arbitraire », *Thèse de doctorat*, Ecole Nationale supérieure des Mines de Paris (1994).

[Masud, 2002] A. Masud and T.J.R. Hughes, "A stabilized mixed finite element method for Darcy flow", *Comput. Methods Appl. Mech. Engrg*. 191, 4341 (2002)

[Nechaev, 2002] D. Nechaev, J. Schröter, M. Yaremchuk. "A diagnostic stabilized finite element ocean circulation model. Ocean Modelling", 5, 37-63 (2003).

[Perchat, 2000] E. Perchat "MINI-élément et factorisations incomplètes pour la parallélisationd'un solveur de Stokes 2D. Application au forgeage", *Thèse de doctorat*, Ecole Nationale supérieure des Mines de Paris (2000).

[PETSc, 2004] http://www-unix.mcs.anl.gov/petsc/petsc-as/

[Saad, 1986] Youcef Saad and Martin H. Schultz. "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems". *SIAM J. Sci. Stat. Comput.*, 7:856–869 (1986).

[Temam, 2000] R. Temam, "Some developments on Navier-Stokes equations in the second half of the 20th century", *Development of Mathematics 1950-2000*, J-P. Pier (ed.) (2000).

CALCUL D'INTERFACES & METHODE LEVEL SET

1. Introduction

Après avoir implémenté un solveur capable de simuler des écoulements de fluides newtoniens incompressibles, rappelons que notre but est de résoudre des problèmes en mécanique des fluides hétérogènes sur grille de calcul. Pour avoir la possibilité de considérer simultanément plusieurs fluides de différentes natures, nous voyons dans ce chapitre comment modéliser le solveur de Navier-Stokes présenté dans le chapitre précédent pour le rendre non homogène.

Les calculs d'interfaces et de surfaces libres sont cruciaux lorsque l'on parle de problèmes d'écoulements multi fluides. Ils permettent l'observation de l'évolution que peuvent avoir les différentes phases, les unes par rapport aux autres, tout au long d'une simulation. Le solveur d'écoulement présenté dans le chapitre précédent à partir des équations de Navier-Stokes incompressibles n'inclut pas explicitement le mouvement des interfaces. Ainsi, il apparaît naturellement le besoin d'introduire une méthode de couplage entre ces équations et un autre type de modèle. La prédiction numérique du mouvement d'une interface dans un milieu continu pose de nombreux problèmes, auxquels des solutions très diverses ont été apportées, aussi bien dans un contexte lagrangien [Rider, 1995] [Magnaudet, 1995] qu'eulérien [Coupez, 2000]. Et même, plusieurs techniques intermédiaires visent à compenser les défauts propres à ces deux approches [Enright, 2005] [Vincent, 1998] [Bigot, 2001]. Parmi les différentes méthodes qui existent aujourd'hui, on trouve les méthodes de suivi d'interface (*interface tracking*) [Rider, 1995] et *Marker and Cell*, ou d'autres de capture d'interfaces (*interface capturing*) comme *Volume Of Fluid* (*VOF*) [Coupez, 2000] [Batkam, 2002] [Bruchon, 2003] ou *Level Set* [Adalsteinsson, 1995] [Zhao, 1998]. Enfin d'assembler les points forts de certaines techniques, et donc de compenser leurs défauts respectifs, des méthodes couplées sont apparues comme [Sussman, 2000] qui fusionne le *Level Set* et le *Volume of Fluid*.

Les applications dans ce domaine sont diverses et incluent les problèmes de solidification (solide/liquide), extrusion de polymères (fluide/fluide), remplissage de moules (liquide/air) [Batkam, 2002] [Bruchon, 2004], etc. De nombreux critères doivent être étudiés afin d'évaluer la qualité d'une méthode de calcul d'interface. Parmi ces critères, on peut citer les propriétés purement géométriques, à savoir, la capacité à décrire correctement le mouvement d'une interface dans un champ de vitesse donné. Il est important que la méthode soit robuste et qu'elle converge vers la solution physique. De plus, il est intéressant que le formalisme soit suffisamment général pour que l'extension en 3D à partir d'un cas 2D soit abordable d'un point de vue formel. Ainsi, pour résumer, les méthodes de calcul d'interface doivent allier des propriétés de haute résolution, de souplesse d'utilisation, et de robustesse, pour supporter des simulations très complexes dans la mécanique des fluides.

Dans ce chapitre, on se focalise sur les techniques de capture d'interfaces, avec, notamment, la mise en place d'une formulation Level Set, et sa comparaison avec la méthode *VOF* déjà disponible dans le centre du CEMEF [Coupez, 2000]. Entre autre, nous verrons que dans les deux cas, l'interface est transportée avec une équation de convection pure, mais la différence principale réside dans la nature de la quantité à transporter, et cela n'est pas sans répercussion sur la résolution. Alors qu'une adaptation de maillage est nécessaire pour faire du *VOF*, le Level Set requiert une stabilisation appropriée de type SUPG ou *Residual-Free Bubbles*.

Les différentes parties de ce chapitre développent les aspects suivants : la modélisation multi-phasique ; l'implémentation d'une technique Level Set performante par le biais d'une

équation de transport continue et une technique de réinitialisation ; et enfin, l'analyse et l'amélioration d'une telle méthode, et sa comparaison avec celle du *Volume Of Fluid*, incluant ses points forts et ses désavantages.

2. Modélisation hétérogène et calcul d'interfaces

Les techniques de calcul d'interfaces peuvent être très différentes en fonction du problème à résoudre, et se divisent généralement en deux grandes familles. La première famille, basée sur une approche Lagrangienne, inclue la méthode appelée *interface tracking* qui découle naturellement du traitement des problèmes de surfaces libres [Rider, 1995]. Cependant, nous ne choisissons pas une telle approche, et seule une brève introduction est présentée dans ce paragraphe. La deuxième famille, basée sur une approche Eulérienne, est souvent appelée *interface capturing*. Elle offre d'autres particularités au traitement des problèmes d'écoulements multi fluides, et peut aussi servir aux simulations de surfaces libres, à condition de pouvoir modéliser un gaz (l'air par exemple) par un fluide incompressible peu visqueux. C'est dans cette dernière famille que se trouvent les méthodes *VOF* (*Volume Of Fluid*) [Batkam, 2002] et Level Set (Lignes de Niveaux) [Peng, 1999] sur lesquelles nous allons nous attarder.

2.1. Approche Lagrangienne

2.1.1. Définition d'une surface libre

La surface libre d'un domaine est une partie de sa frontière sur laquelle ne s'exerce aucune contrainte. Il est donc naturel de faire coïncider les surfaces libres avec les bords du domaine de calcul. Seulement, dans le cas instationnaire, il s'agit d'un problème d'évolution pour lequel on ne connaît que la géométrie initiale. Afin de garder cette même philosophie, il est nécessaire de faire évoluer la frontière du domaine de calcul suivant les mouvements de l'écoulement, ce qui revient à traiter le problème instationnaire de façon Lagrangienne.

2.1.2. Interface Tracking

Les calculs de surfaces libres effectués par *interface tracking* sont des méthodes à approche Lagrangienne [Rider, 1995] [Magnaudet, 1995]. Le suivi d'un front est réalisé grâce à une description explicite de l'interface, au sens où l'interface est décrite par un ensemble d'objets qui seront transportés dans le domaine de calcul à la vitesse locale. Autrement dit, les surfaces font partie intégrale des bords du domaine de calcul, et donc du maillage ; et la déformation dynamique des contours du maillage constitue ainsi exactement l'évolution de l'interface. La reconnection d'un ensemble d'objets permettra de reconstruire la géométrie globale, incluant l'interface. Cette approche est naturelle pour les problèmes de surface libre dans le sens où la condition limite appropriée (une contrainte nulle imposée) peut être appliquée facilement sur les bords connus du maillage. Cependant, ce dernier doit évoluer de la même manière que le fluide, et la distorsion des éléments ou le changement de topologie de surface demandent une procédure très complexe.

L'avantage principale de ces méthodes réside dans leur faculté à tenir compte très précisément des conditions de surface libre (ces conditions sont en fait des conditions limites entre deux domaines séparés par le maillage-interface). En revanche, elles manquent de

flexibilité lorsque l'on souhaite suivre des mouvements d'interface complexes, puisqu'il faut reconstruire le maillage à chaque pas de temps.

2.1.3. Marker And Cell

D'autres méthodes Lagrangiennes ont été développées : il s'agit par exemple des chaînes de marqueurs ou MAC (*Marker And Cell*) [Enright, 2002]. L'interface est représentée par un ensemble de points connectés entre eux se déplaçant dans le domaine de calcul. Même si elles possèdent souvent une résolution très supérieure aux méthodes eulériennes et qu'elles permettent une localisation très précise de l'interface, elles souffrent de plusieurs inconvénients liés à leur nature Lagrangienne. Les marqueurs doivent être régulièrement redistribués dans le domaine de calcul pour éviter qu'ils ne s'accumulent dans certaines zones et que la résolution de l'interface ne soit altérée dans d'autres zones de par une trop grande distance entre eux. De plus, les changements topologiques (recouvrement ou séparation de deux interfaces) ne s'effectuent pas naturellement. Pour les rendre possible, un critère arbitraire doit être imposé pour définir l'instant de la jonction ou de la séparation des interfaces, et c'est certainement là le principal inconvénient de ces méthodes. De plus, tous ces inconvénients sont largement amplifiés sur des simulations tridimensionnelles.

2.2. Approche Eulérienne

Bien que les méthodes Lagrangiennes soient très souvent utilisées en mécanique de solide, dans les problèmes de grandes déformations, nous préférons une autre approche plus flexible, comme celle de l'interface diffuse dans un contexte Eulérien [Coupez, 2000], mieux adaptée à la mécanique des fluides.

2.2.1. Capture d'interface

Décrire la surface et la paramétrer est facile lorsqu'elle est une partie du bord du maillage. Mais on peut aussi modéliser les surfaces libres sans pour autant devoir les localiser précisément à l'aide de la frontière d'un maillage. Il est possible d'aborder les problèmes de mouvements de surfaces libres et d'interfaces de manière Eulérienne [Harabetian, 1996]. Pour cela, on considère un champ scalaire qui évolue en fonction du temps grâce à une équation de transport et un champ de vitesse. Ces méthodes sont appelées « capture d'interface ». Contrairement aux méthodes d'*interface tracking*, la capture d'interface a une approche globale qui considère de façon Eulérienne tous les sous domaines présents dans un écoulement en même temps. Pour traiter le calcul des surfaces libres et des interfaces de façon Eulérienne, le domaine de calcul Ω est supposé fixe en fonction de temps *t*. Par contre, il est composé de plusieurs sous domaines qui eux ne sont pas fixes et qui dépendent du temps $\Omega_i = \Omega_i(t)$ tels que :

$$\Omega = \bigcup \Omega_i(t) \quad \text{et} \quad \bigcap \Omega_i(t) = 0 \qquad (1)$$

Nous considérons les interfaces $\Gamma_i(t)$ qui sont représentées par les bords de chaque sous domaines Ω_i et qui se déplacent aussi en fonction du temps. Alors que dans les méthodes de

tracking, l'interface est suivie explicitement par les bords du maillage, celle-ci passe à travers les éléments du maillage dans les méthodes de capture.

Même si cette approche est plus naturelle pour capturer l'interface entre plusieurs fluides, on peut aussi l'utiliser pour modéliser des surfaces libres si l'on considère que la cavité Ω contient un sous domaine Ω_f pour le fluide et un autre Ω_a pour l'air :

$$\Omega = \Omega_f(t) \cup \Omega_a(t) \qquad (2)$$

Chaque sous domaine correspond donc à une phase en présence dans l'écoulement, et la surface libre de la phase liquide est définie par l'interface $\Gamma(t)$ entre ces deux sous domaines.

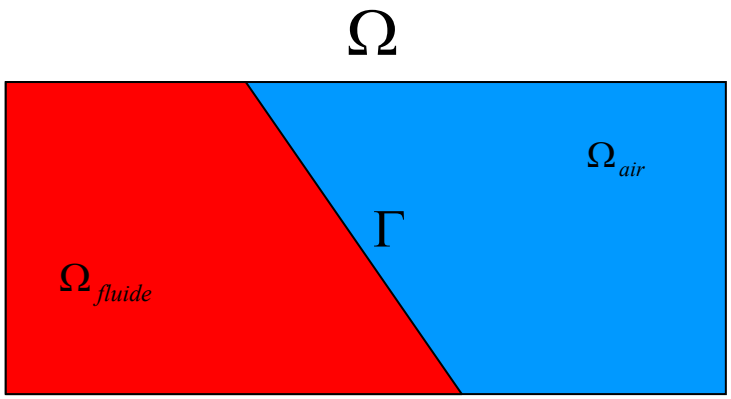


Figure 1 – Les sous domaines d'une cavité Ω et l'interface Γ qui les sépare

Il reste maintenant à introduire une fonction servant à différencier les sous domaines et à capturer l'interface. Deux différentes techniques permettent cela : le *Volume Of Fluid* et le Level Set.

2.2.2. Volume of Fluid

Les méthodes de *Volume Of Fluid*, spécialement conçues pour la simulation d'écoulements diphasiques, sont basées sur le principe de conservation de la masse [Batkam, 2002]. La distribution de fraction volumique de liquide (ou de gaz) est répartie dans le domaine de calcul sur un maillage fixe, puis elle est transportée par le champ de vitesse. Les caractéristiques de l'interface sont ensuite déduites du champ de fraction volumique. Les méthodes *VOF* sont robustes lors des changements topologiques, qui s'effectuent de façon implicite. La conservation du volume est garantie lorsque l'on transporte la fraction volumique [Pichelin, 1998].

Le moyen le plus naturel pour définir les sous domaines est d'introduire une fonction de présence, appelée fonction caractéristique, associée à chacun d'entre eux. Ainsi, la fonction correspondante au sous domaine Ω_f peut avoir une valeur de 1 lorsqu'on se situe dans celui-ci, et une valeur nulle dans le cas contraire. Plus précisément, cette variable sert de fraction volumique dans le sens où, en la définissant constante par élément, elle représente le taux de remplissage de chaque élément du maillage. Lorsqu'elle vaut 1, l'élément est entièrement rempli par le fluide, mais elle vaut 0 quand l'élément ne contient pas de fluide du tout. Les valeurs intermédiaires indiquent que l'élément n'est rempli que partiellement. Cette technique

permet de facilement visualiser les différents domaines ainsi que leur évolution. Par contre, nous savons à priori que les interfaces ne peuvent pas être très bien définies : avec une approximation spatiale *P0*, la précision est limitée à la taille des éléments du maillage.

Comme nous le montrons par la suite, c'est par la résolution d'une équation de convection pure que la fonction caractéristique peut être transportée. Puisque la variable, solution de cette équation, est de type *P0*, elle peut être résolue avec une méthode de Galerkin discontinu. Plus précisément, ce problème peut être réécrit en introduisant une dimension temporelle pour créer une méthode éléments finis 4D en espace-temps avec une approximation *P0* en espace, mais *P1* (ou *Pn*) en temps [Coupez, 2000]. Cette technique est bien appropriée à ce type de problème et se montre très robuste : elle converge inconditionnellement (c'est-à-dire indépendamment du pas de temps). Aussi, la mauvaise définition de l'interface causée par le bas degré de l'interpolation spatiale peut être compensée par une adaptation dynamique du maillage qui raffine la taille de maille aux alentours de cette interface [Bigot, 2000].

2.2.3. Level Set

Enfin la méthode Level Set est une méthode récente de capture d'interface, et a été développée dans [Osher, 1988], puis améliorée dans [Chang, 1996] [Peng, 1999] [Sussman, 1999]. Son principe est de définir une fonction interface dans le domaine de calcul dont la courbe de niveau zéro est l'interface que l'on cherche à décrire. Une fonction de phase plus régulière que dans la fonction caractéristique de *VOF* est alors introduite. Cette fonction Level Set associée à un sous domaine est la distance signée à son interface. Elle est donc positive à l'intérieure du sous domaine, et négative partout ailleurs dans la cavité. L'interface est implicitement représentée par son iso valeur zéro. Sa nature est très différente de celle présentée précédemment car celle-ci est linéaire et peut être interpolée sur les nœuds du maillage. Non seulement elle est continue, mais en plus, elle est régulière. Avec son interpolation spatiale de type *P1*, un ordre de précision est gagné sur la description de l'interface. De cette manière, une géométrie linéaire dans la cavité peut être représentée exactement par une fonction Level Set. Les changements topologiques sont gérés naturellement, et les caractéristiques géométriques du problème sont calculées grâce à cette fonction interface. Sa mise en œuvre est aussi directe en 2D qu'en 3D. De plus, nous verrons par la suite que la procédure d'adaptation de maillage, indispensable dans la méthode *VOF*, n'est plus essentielle avec du Level Set.

Ici aussi, les mouvements de l'interface demandent la résolution d'une équation de convection pure pour transporter la fonction de phase et prédire les mouvements de l'interface dans un champ de vitesse donné. Cependant, bien que cette équation soit strictement la même que dans la méthode *VOF*, sa résolution est complètement différente. Les méthodes éléments finis de type Galerkin standard, généralement utilisées dans ce cas, se comportent mal avec les problèmes dominés par un phénomène de convection. Le transport de la fonction Level Set *P1* nécessite donc une stabilisation adaptée [Franca, 2000].

L'application de cette méthode à la simulation d'écoulements diphasiques n'a pas été immédiate car un certain nombre de défauts sont apparus. Entre autres, la présence de zones de cisaillement ou d'étirement dans le champ de vitesse va fortement étaler ou resserrer les lignes de niveau, altérant ainsi ses propriétés algébriques remarquables [Rider, 1995]. Cependant, depuis les premiers pas de cette méthode des progrès majeurs ont été réalisés, tant

dans l'amélioration de la résolution de l'équation de transport, que dans l'amélioration de ses propriétés algébriques. Par la suite, on se consacrera au formalisme Level Set, et à ces développements récents, qui en font une méthode précise et élégante [Sussman, 1999].

3. La méthode *Volume of Fluid*

Après avoir présenté brièvement des techniques de calcul d'interfaces, nous étudions plus précisément les deux méthodes de capture d'interface que sont le *Volume of Fluid* [Bruchon, 2003] et le Level Set.

3.1. Présentation de la méthode

Dans les simulations d'écoulements de fluides avec surface libre, la masse totale du fluide contenu dans le domaine Ω_f doit rester constant en fonction du temps :

$$\frac{d}{dt}\int_{\Omega_f}\rho_f = 0 \qquad (3)$$

où ρ_f est la masse volumique du fluide. Pour pouvoir étendre cette propriété au domaine Ω dans son ensemble, on peut introduire une fonction de présence (ou fonction caractéristique) du domaine du fluide 1_{Ω_f} :

$$1_{\Omega_f}(x) = \begin{cases} 1 & \text{si } x \in \Omega_f \\ 0 & \text{si } x \in \Omega - \Omega_f \end{cases} \qquad (4)$$

Cette fonction discontinue vaut 1 dans le domaine du fluide, et 0 dans le reste de la cavité qui ne contient pas de fluide. Ainsi, *(3)* devient :

$$\frac{d}{dt}\int_{\Omega} 1_{\Omega_f}\rho_f = 0 \qquad (5)$$

Et donc :

$$\int_{\Omega}\left(\frac{d1_{\Omega_f}}{dt}\rho_f + 1_{\Omega_f}\left(\frac{d\rho_f}{dt} + \rho_f \nabla\cdot v\right)\right) = 0 \qquad (6)$$

Deux aspects sont tirés de cette dernière formule : premièrement, l'équation de la conservation de la masse du fluide peut être résolue globalement sur tout le domaine Ω lorsque l'on introduit la fonction caractéristique 1_{Ω_f} ; et deuxièmement, une équation de transport doit être résolue pour assurer l'évolution de cette fonction :

$$\frac{d}{dt}1_{\Omega_f} = 0 \quad \text{dans } \Omega \qquad (7)$$

Avec une description Eulérienne, on obtient :

$$\frac{d}{dt}1_{\Omega_f} = \frac{\partial}{\partial t}1_{\Omega_f} + v.\nabla 1_{\Omega_f} \qquad (8)$$

où v est le champ de vitesse matériel. Cette équation de transport peut alors être résolue avec un schéma approprié pour faire évoluer le domaine du fluide en fonction du temps, ainsi que sa frontière qui représente la surface.

Avec une approximation spatiale de type *P0* (interpolation constante sur les éléments) associée à une méthode de Galerkin discontinue, on s'aperçoit que l'approximation de la valeur de la fonction caractéristique est précisément la fraction de volume contenu dans chaque élément du maillage. De ce fait, $1_{\Omega_f}\Big|_K = 1$ veut dire que l'élément K est entièrement rempli de fluide, et, lorsque $1_{\Omega_f}\Big|_K = 0$, K est entièrement rempli d'air. Les éléments qui ont une valeur intermédiaire (entre 0 et 1), contiennent à la fois du fluide et de l'air. Au final, l'approximation *P0* de la fonction caractéristique sur un élément K s'écrit :

$$1^h_{\Omega_f}\Big|_K = 1_K = \frac{\left|K \cap \Omega_f\right|}{\left|K\right|} \qquad (9)$$

3.2. Une loi de mélange pour une modélisation hétérogène

L'introduction de cette fonction de présence permet de formuler un système de Navier-Stokes non homogène. A l'aide d'une loi de mélange appropriée, un seul et même problème de Navier-Stokes peut servir à simuler un écoulement hétérogène contenant plusieurs types de fluides. Pour cela, il suffit de faire dépendre la masse volumique ρ et la viscosité η de la fonction de caractéristique et des paramètres propres à chacun des sous domaines. Par exemple, dans le cas d'une simulation de surface libre, l'idée est de donner les paramètres du liquide dans le sous domaine Ω_l, et d'autres, proches de ceux de l'air, ailleurs dans la cavité $(\Omega - \Omega_l)$. Cela sous-entend que le comportement de l'air est modélisé par les équations de Navier-Stokes habituellement utilisées pour des fluides newtoniens incompressibles. Par la suite, nous nous contenterons donc de considérer l'air comme un fluide peu visqueux (environ 10^{-5} Pa.s) et incompressible, avec une masse volumique de 1.3 kg/m^3. Cette hypothèse d'incompressibilité est en général acceptable lorsque la vitesse des écoulements est relativement petite, ce qui est le cas dans l'ensemble des applications que l'on vise à simuler.

Rappelons la formulation forte du problème de Navier-Stokes incompressible :

$$\begin{cases} \rho\dfrac{\partial v}{\partial t} + \rho\nabla v.v - \nabla\cdot\left(2\eta\varepsilon(v)\right) + \nabla p = \rho g & \text{dans } \Omega \\ \nabla\cdot v = 0 \end{cases} \qquad (10)$$

Pour rendre ce système non homogène, nous appliquons la loi de mélange suivante pour la masse volumique et la viscosité :

$$\begin{aligned} \rho &= 1_{\Omega_l}\rho_l + \left(1 - 1_{\Omega_l}\right)\rho_a \\ \eta &= 1_{\Omega_l}\eta_l + \left(1 - 1_{\Omega_l}\right)\eta_a \end{aligned} \qquad (11)$$

où :

ρ_l est la masse volumique du liquide

η_l est la viscosité du liquide

ρ_a est la masse volumique de l'air

η_a est la viscosité de l'air

Finalement, c'est ce système qu'il faut résoudre pour simuler des écoulements hétérogènes :

$$\begin{cases} \rho\dfrac{\partial v}{\partial t} + \rho\nabla v.v - \nabla\cdot\left(2\eta\varepsilon(v)\right) + \nabla p = \rho g \\ \nabla\cdot v = 0 \\ \rho = 1_{\Omega_l}\rho_l + \left(1 - 1_{\Omega_l}\right)\rho_a \\ \eta = 1_{\Omega_l}\eta_l + \left(1 - 1_{\Omega_l}\right)\eta_a \end{cases} \quad \text{dans } \Omega \qquad (12)$$

De cette manière, sur les éléments où $1_K = 1$, le comportement de l'écoulement est celui du liquide, tandis qu'avec $1_K = 0$, c'est le comportement de l'air qui est considéré. Pour des valeurs intermédiaires de la fonction de présence, ce sont des viscosités et masses volumiques mélangées qui donnent le comportement de l'écoulement.

Ainsi, il n'existe qu'un seul champ de vitesse et de pression sur tout le domaine de calcul, et c'est cette vitesse qui va servir à transporter les différentes fonctions de présence des sous domaines. Ce modèle peut aisément être généralisé à des écoulements hétérogènes de 2 fluides ou plus : il suffit d'avoir une fonction caractéristique pour chaque sous domaine, et de les transporter séparément avec le champ de vitesse solution du solveur de l'écoulement *(12)*.

3.3. Résolution de l'équation de transport discontinue

3.3.1. Formulations

Comme nous l'avons vu précédemment, à un instant *t* donné, un sous domaine Ω_i de la cavité Ω est défini par une fonction caractéristique $1_{\Omega_i}(x,t)$. L'évolution de ce sous domaine au cours du temps est calculée avec l'équation de transport *(7)* où il faut trouver $1_{\Omega_i}(x,t)$ tel que, $\forall(x,t) \in \Omega \times [0,T]$. Avec une approche Eulérienne, l'équation *(8)* fait apparaître une vitesse de convection $v(x,t)$, généralement choisie comme la vitesse de la matière. Les détails sur le principe de traitement de cette équation de transport sont dans [Bruchon, 2003], et nous nous contentons ici de présenter les grandes lignes de la résolution.

Avec une approximation discontinue de la solution du problème, la méthode éléments finis développée pour résoudre l'équation *(8)* fait partie de la classe des méthodes de Galerkin discontinu. Le fait qu'elles privilégient l'information amont de la solution est un grand avantage pour ce type de problème.

Afin d'obtenir une méthode inconditionnellement stable (indépendamment du pas de temps), le temps est traité implicitement. Ainsi, seul l'erreur souhaitée sur la solution détermine la valeur du pas de temps. Pour cela, une formulation en espace-temps peut être considérer : elle consiste à rajouter une dimension temporelle aux dimensions spatiales.

La formulation de type éléments finis 4 dimensions en espace-temps [Coupez, 2000] s'appuie sur une discrétisation structurée en temps. On se place dans un domaine spatio-temporel, noté : $\widetilde{\Omega} = \Omega \times]0,T[$ où T représente la durée du processus. On appelle alors $\widetilde{\Omega}_h$ la discrétisation de ce domaine, telle que $\widetilde{\Omega}_h = \bigcup \widetilde{K}$ où $\widetilde{K}$ sont les éléments 4D de la méthode éléments finis.

Rappelons que la cavité Ω est décomposée en tétraèdres (ou *d*-simplexes) à travers un maillage Eulérien $T_h(\Omega)$ pour discrétiser le domaine spatial : $\Omega_h = \bigcup_{K \in T_h(\Omega)} K$. Le domaine temporel, lui, est discrétisé avec N intervalles de temps réguliers : $[0,T] = \bigcup_{i=1}^{N} I_i$ où $I_i = [t_i, t_{i+1}]$, $t_0 = 0$, et $t_N = T$.

Le problème de l'évolution d'un sous domaine spatial Ω_i associée à la fonction de présence $1_{\Omega_i}(x,t)$ au cours du temps est le suivant :
Trouver $1_{\Omega_i}(x,t)$ tel que, $\forall (x,t) \in \widetilde{\Omega}$:

$$\begin{cases} \dfrac{d1_{\Omega_i}(x,t)}{dt} = \dfrac{\partial}{\partial t} 1_{\Omega_i}(x,t) + v(x,t).\nabla 1_{\Omega_i}(x,t) = 0 \\ 1_{\Omega_i}(x,0) = 1^0 \end{cases}$$

Ensuite, on redéfinit l'opérateur gradient que l'on notera $\widetilde{\nabla}$, et la vitesse, notée $\tilde{v}$ dans $\widetilde{\Omega}$ pour se placer en 4 dimensions :

$$\tilde{v} = \begin{pmatrix} v \\ 1 \end{pmatrix} \text{ et } \widetilde{\nabla} = \begin{pmatrix} \nabla \\ \dfrac{\partial}{\partial t} \end{pmatrix} \qquad (13)$$

L'équation à résoudre se réécrit donc ainsi :

$$\frac{d1_{\Omega_i}(\tilde{x})}{dt} = \tilde{v}.\widetilde{\nabla} 1_{\Omega_i} = 0 \qquad (14)$$

3.3.2. Galerkin Discontinu

La fonction caractéristique est approchée par une fonction discrète $1^h_{\Omega_i}(x,t)$ – notée 1^h – constante sur chaque élément spatial du maillage.

Soit l'espace d'approximation :

$$P^{0,q}(\widetilde{\Omega}) = \left\{ f \in L^2(\Omega), f_{\widetilde{K}} \in P^0(K) \times P^q(I_i), \forall \widetilde{K} = K \times I_i \right\} \qquad (15)$$

où $P^0(K)$ est l'ensemble des polynômes de degré zéro sur chaque élément K de Ω_h, et $P^q(I)$ est l'ensemble des polynômes de degré q sur I, s'annulant au temps t_i, de base : $\psi_j(t) = (t - t_i)^j$ avec $j \in [0,q]$ et $\forall t \in I_i$.

On construit $1^h \in P^{0,q}$, polynomiale par morceaux : polynôme constant en espace, et de degré q en temps. Sur chaque élément $\widetilde{K}$ de $\widetilde{\Omega}_h$ on a : $1^h_{\widetilde{K}} = \sum_{p=0}^{q} 1^p_{\widetilde{K}} (t - t_i)^p$.

Les $1^h_{\widetilde{K}}$ sont les $q+1$ inconnues sur les éléments espace-temps $\widetilde{K}$.

En supposant que la vitesse v est continue sur $\widetilde{\Omega}_h$, la forme faible du problème *(14)* est donnée au sens des distributions par la formule suivante :

$$\left\langle v.\widetilde{\nabla} 1^h, \varphi \right\rangle_{\widetilde{\Omega}} = -\sum_K \left(\sum_{F \subset \partial \widetilde{K}} \int_F \left[1^h \right]^{\widetilde{F}}_{\widetilde{K}} \varphi \left(v.n^{\widetilde{F}}_{\widetilde{K}} \right)^- d\Gamma + \sum_{\widetilde{K} \in \widetilde{\Omega}(\widetilde{F})} \int_{\widetilde{K}} \widetilde{\nabla} 1^h . v . \varphi . d\Omega \right) = 0 \qquad (16)$$

où $\widetilde{\Omega}(\widetilde{F})$ est l'ensemble des éléments qui se partagent la face $\widetilde{F}$, et $n^{\widetilde{F}}_{\widetilde{K}}$ est la normale sortante de $\widetilde{F}$. $\left(v.n^{\widetilde{F}}_{\widetilde{k}} \right)^-$ résulte du décentrement qui assure la stabilité de la méthode : le flux de la matière $v.n^{\widetilde{F}}_{\widetilde{k}}$ est décomposé en sa partie positive $\left(v.n^{\widetilde{F}}_{\widetilde{k}} \right)^+$ et sa partie négative $\left(v.n^{\widetilde{F}}_{\widetilde{k}} \right)^-$.

Rappelons que la fonction de présence 1^h est constante par élément, et donc, le saut de 1^h à travers une face $\widetilde{F}$ peut être exprimé de la façon suivante :

$$\left[1^h \right]^{\widetilde{F}}_{\widetilde{K}} = \sum_{\widetilde{K} \in \widetilde{\Omega}(\widetilde{F})} 1^h_{\widetilde{K}} . n^{\widetilde{F}}_{\widetilde{K}} . n^{\widetilde{F}}_{\widetilde{K}} \qquad (17)$$

Enfin, il est montré dans [Coupez, 2000] qu'un ordre d'interpolation *P1* pour l'espace temporel est suffisant. Donc, en général, on se place dans l'espace $P^{0,1}(\widetilde{\Omega})$, c'est-à-dire que l'on cherche une solution constante en espace et linéaire en temps. Cela nous amène à résoudre le système linéaire de dimension 2 sur $\widetilde{K}$:

Trouver $1^0_{\tilde{K}}$ et $1^1_{\tilde{K}}$, $\forall r \in [0,1]$ tel que :

$$\delta_{0r} 1^0_{\tilde{K}} + \frac{1}{1+r} 1^1_{\tilde{K}} |I|^{1+r} - \frac{1}{|K|} \sum_{p=0}^{1} \frac{|I|^{p+r+1}}{p+r+1} \sum_{F \in \partial\Omega} \left(1^p_{\tilde{K}} - 1^p_{\tilde{K}(F)} \right) \int_F \left(v.n^F_{\tilde{K}} \right)^- d\Gamma - \delta_{0r} \sum_{p=0}^{1} 1^p_{\tilde{K}^-} |I_{i-1}|^p = 0 \qquad (18)$$

où :

δ_{0r} est le symbole usuel de Kronecker ($\delta_{0r} = 1$ si $r = 0$, et $\delta_{0r} = 0$ sinon)

I_{i-1} est l'intervalle de temps précédent

$|I|$ désigne la taille de l'intervalle de temps I

$|K|$ est le volume de l'élément spatial K.

3.4. Adaptation de maillage

Nous avons déjà souligné l'importance d'une bonne description des interfaces entre les différents sous domaines présents dans l'écoulement. Or, cette stratégie numérique est basée sur une interpolation spatiale *P0* avec une approche Eulérienne, ce qui limite la définition géométrique des interfaces. De plus, une diffusion numérique assez importante résulte du calcul d'évolution : tout au long d'une simulation, la fonction caractéristique contient de plus en plus de valeurs intermédiaires (entre 0 et 1).

Afin de limiter cette diffusion, une technique de r-adaptation de maillage [Bigot, 2000] [Bigot, 2001] peut être utilisée. Cette technique consiste à déplacer les noeuds d'un maillage, tout en maintenant sa topologie inchangée. Contrairement aux approches lagrangiennes ou ALE (Arbitrairement Lagrangien Eulérien), le maillage ne se déforme pas suivant le mouvement de la matière : ses éléments proches d'une interface sont contractés, puis reprennent leur taille initiale lorsque celle-ci s'éloigne.

Ce déplacement des noeuds du maillage se traduit par une vitesse de maillage, que nous appellerons vitesse d'adaptation, notée $v_{adapt}(x,t)$. Cette vitesse d'adaptation est prise en compte dans l'équation de transport *(8)* en substituant :

$$v(x,t) = v_{fluide}(x,t) - v_{adapt}(x,t) \qquad (19)$$

3.5. Validations

Afin de valider notre solveur de l'équation de transport *(8)* basé sur une méthode éléments finis de Galerkin discontinu en espace-temps, nous étudions un cas test présenté dans [Bruchon, 2004] et inspiré par [Vincent, 1998]. Il s'agit d'effectuer la rotation complète d'un cercle autour d'un axe à partir d'un champ de vitesse fixe dans une cavité carré en 2 dimensions. Le solveur implémenté dans CimLib se porte aussi bien en 2D qu'en 3D, et c'est donc une formulation espace-temps en 3 dimensions qui est utilisée ici (2 dimensions

spatiales et 1 dimension temporelle). La figure *(2)* montre le champ de vitesse fixé pour faire transporter par convection le cercle dans le carré de dimension (1x1) avec les valeurs suivantes en fonction des coordonnées (*x,y*) des nœuds du maillage :

$$v = \begin{pmatrix} -\frac{\pi}{2}\left(y - \frac{1}{2}\right) \\ \frac{\pi}{2}\left(x - \frac{1}{2}\right) \end{pmatrix} \qquad (20)$$

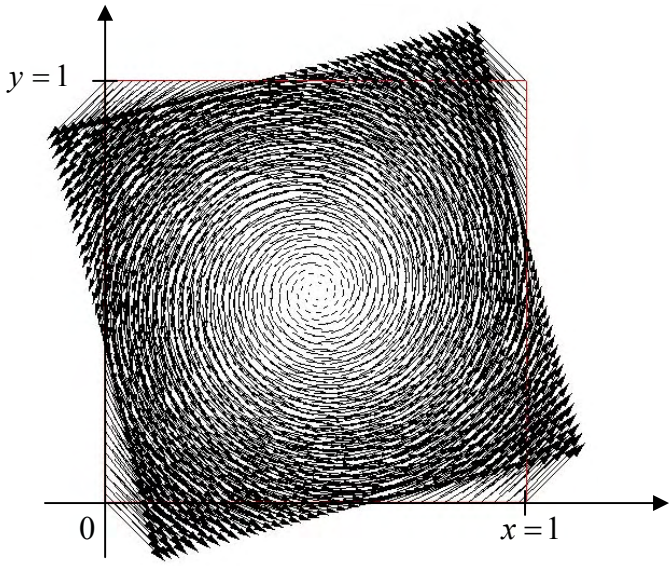


Figure 2 – Champ vitesse

Initialement, nous définissons une géométrie circulaire avec un rayon de *0.1* centré sur le point (*0.7, 0.5*). L'intérieur de ce cercle défini un sous domaine qui va être transporté tout autour du point central de la cavité (*0.5, 0.5*) avec le champ de vitesse *(20)*, jusqu'à ce qu'il revienne à se place initiale (*0.7, 0.5*).

Comme il est indiqué dans [Bruchon, 2004], nous prenons un pas de temps $\Delta t = 0.01$. Aussi, nous avons des interpolations *P0* en espace et *P1* en temps, conformément aux détails sur la méthode présentés précédemment. Avec la vitesse définie avec *(20)*, il faut 400 incréments de temps pour que le cercle effectue un tour complet : analytiquement, le sous domaine circulaire doit retrouver sa position initiale centrée sur le point (*0.7, 0.5*) au temps $t = 4$.

Les résultats de la simulation de ce cas test sont montrés dans la figure *(3)* : on peut y voir les valeurs de la fonction de présence du sous domaine qui nous intéresse. Initialement, à $t = 0$, elle vaut *1* dans le cercle, *0* à l'extérieur, et entre les deux sur les éléments traversés par l'interface. Celle-ci est donc assez bien représentée, mais dépend de la taille des éléments : plus la taille de maille est petite, plus l'interface est bien représentée. Ensuite, cette même fonction caractéristique est représentée aux temps $t = 1$, $t = 2$, $t = 3$, et $t = 4$ de la rotation.

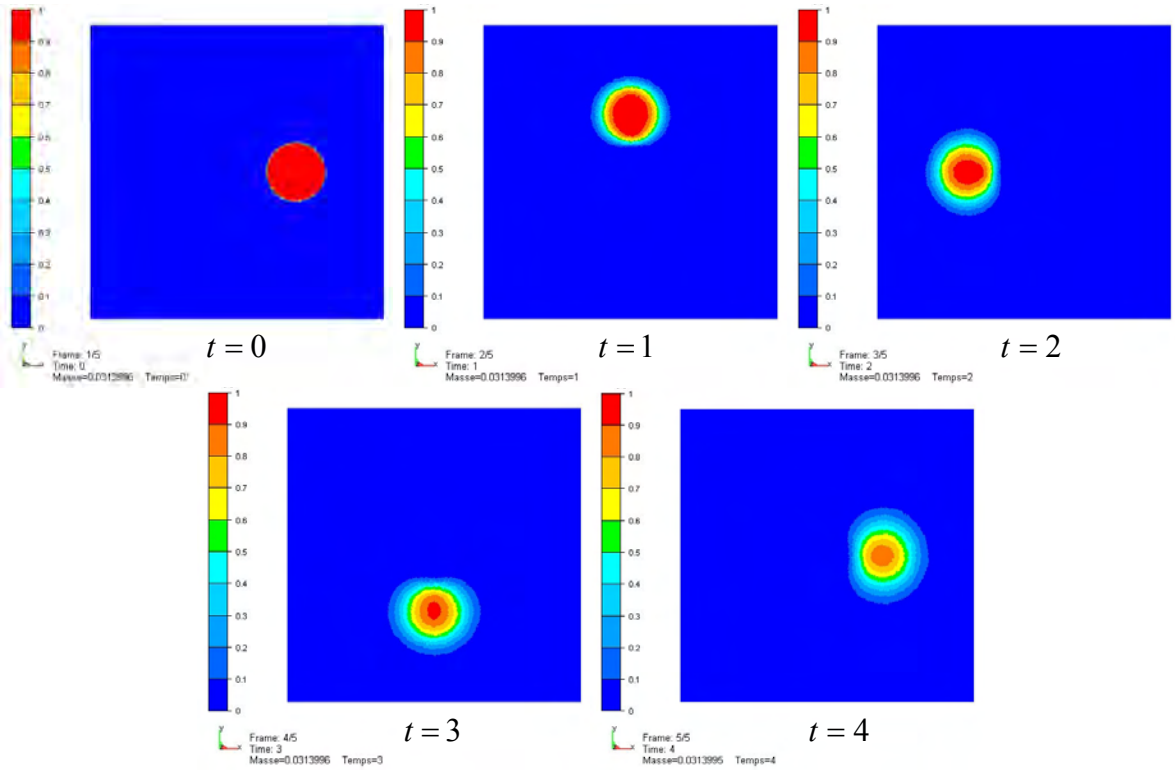


Figure 3 – Rotation sans adaptation de maillage

Nous constatons que le transport du cercle est correctement réalisé car, non seulement il revient exactement à sa place initiale à $t = 4$, mais aussi, son volume est parfaitement conservé. Il occupe un volume de *3.14%* du volume total de la cavité tout du long de la simulation. Par contre, une importante diffusion de l'interface est observée : les éléments dont la fonction caractéristique a une valeur intermédiaire sont de plus en plus nombreux. Par conséquent, la définition de l'interface se voit fortement altérée, et la forme initiale parfaitement circulaire du sous domaine n'est plus vraiment reconnaissable.

Dans cette simulation, 400 incréments de temps ont été réalisés. Ils consistent tous à la résolution d'un système linéaire non symétrique contenant 138756 inconnues (2 inconnues pour les 69378 éléments du maillage). Après avoir préconditionné les matrices avec une méthode de type Jacobi par blocs, PETSc résous les systèmes avec GMRES (résidu minimal généralisé). A la fin de la simulation, le solveur a passé 171 secondes dans l'assemblage des matrices locales, et 392 secondes dans la résolution des systèmes linéaires.

La méthode de r-adaptation de maillage brièvement décrite précédemment permet de limiter le phénomène de diffusion. Dans [Bruchon, 2004], on peut observer l'amélioration importante qu'elle apporte. Le raffinement de maillage aux alentours de l'interface diminue fortement le phénomène de diffusion non souhaitable pour garder une bonne précision numérique.

3.6. Conclusion : forces et faiblesses de *VOF*

La modélisation hétérogène présentée dans ce paragraphe a une approche Eulérienne et s'appuie sur une fonction de présence à interpolation spatiale *P0* (constante sur les éléments, et discontinue aux bords de ceux-ci). Elle sert à distinguer les différents sous domaines présents dans l'écoulement de fluides. L'évolution des interfaces est assurée par une équation de convection pure qui transporte la fonction caractéristique associée à chaque sous domaine. De par la nature de la quantité à transporter, une méthode éléments finis de type Galerkin discontinu avec une formulation en espace-temps est choisie pour résoudre cette équation.

Les forces de cette technique résident à la fois dans sa robustesse et sa conservation. Son traitement implicite du temps la rend inconditionnellement stable, c'est-à-dire que la grandeur du pas de temps ne joue aucun rôle sur la stabilité de la méthode. La liberté sur le choix du pas de temps est donc uniquement limitée par la précision numérique que l'on veut atteindre. Ensuite, comme le montre le cas test du cercle, la conservation de la masse est parfaitement respectée. La méthode de Galerkin discontinu prend très bien en considération les données amonts par rapport à la vitesse de convection et stabilise efficacement l'équation de transport.

Cependant, cette modélisation présente plusieurs désavantages non négligeables. Premièrement, elle provoque une importante diffusion numérique qui n'est pas souhaitable lorsque l'on souhaite capturer précisément les interfaces. Un bon moyen existe pour contrebalancer ce problème : la r-adaptation dynamique du maillage. Elle apporte une très forte amélioration aux résultats, mais nécessite un traitement supplémentaire qui alourdi les calculs. Deuxièmement, le bas degré d'interpolation spatiale *P0* de la fonction caractéristique limite la représentation des interfaces à la taille de maille des éléments du maillage. En effet, la valeur associée à un élément traversé par l'interface se trouve entre *0* et *1*, mais cela n'indique rien sur l'endroit exact où passe l'interface à l'intérieure de l'élément. Elle donne seulement la fraction volumique, c'est-à-dire la quantité relative de la matière contenue dans l'élément. Et finalement, la nature de la quantité à transporter implique un nombre important d'inconnues à résoudre : lorsque l'on choisi une interpolation temporelle *P1*, il y a deux fois plus d'inconnues à résoudre que d'éléments dans le maillage. En sachant qu'en moyenne, un maillage contient 2 fois plus d'éléments que de nœuds en 2D, et 5.5 fois plus en 3D, il y a au final une grande différence dans le système linéaire à résoudre entre ce solveur et un autre basé sur une interpolation linéaire en espace, comme l'est, par exemple, notre solveur de Navier-Stokes.

Nous présenterons donc par la suite une méthode de capture d'interface qui, de part sa nature, sera moins lourde, mais sans pour autant porter atteinte à la qualité des simulations numérique.

4. La méthode Level Set

Une première technique de capture d'interface appelée *Volume of Fluid* est présentée et analysée dans le paragraphe précédent. Elle a été développée au sein du centre du CEMEF [Coupez, 2000] et utilisée depuis plusieurs années. Nous allons maintenant nous intéresser aux méthodes de type Level Set qui représentent une alternative à la capture d'interface lorsqu'il s'agit d'écoulements de fluides hétérogènes. Après avoir introduit, implémenté, puis étudié cette technique, elle sera comparée au *Volume of Fluid* en identifiant les aspects les plus importants pour les applications que l'on vise à simuler.

4.1. Présentation et historique de la méthode

Les bases du formalisme de la méthode Level Set ont été posées par [Osher, 1988]. Cette méthode, inspirée des travaux précédents de [Sethian, 1984] sur la propagation de front de flamme, a été motivée par la difficulté numérique de représenter une structure de forme arbitraire de dimension N, dans un domaine de dimension $N+1$. Cette première ébauche appelée *PSC* (*Propagation of Surfaces under Curvature*) permettait de suivre le mouvement d'une interface transportée par un champ de vitesse quelconque, ou dépendant de la courbure locale du front. Partant du constat que les méthodes de marqueurs étaient déficientes, lors des changements topologiques, et que les méthodes *VOF* avaient des propriétés algébriques limitées, les auteurs proposèrent un formalisme eulérien plus général, en associant au mouvement de l'interface une équation de transport pour une fonction linéaire.

Depuis sa conception dans [Osher, 1988], la méthode Level Set est utilisée pour capturer des interfaces (*interface capturing*), plutôt que de les suivre (*interface tracking*). Les avantages de cette technique sont maintenant largement reconnus. La méthode est stable, les équations ne sont pas lourdes à résoudre, et les problèmes en trois dimensions ne posent aucune difficulté. Des améliorations récentes ont été apportées sur les écoulements multiphasiques dans [Zhao, 1996] et [Zhao, 1998], puis sur les fronts instables dans [Harabetian, 1996]. Les premiers travaux effectués au sujet du Level Set, comme dans [Adalsteinsson, 1995], diffèrent de notre approche dans le sens où seulement les valeurs d'une fonction Level Set sont utilisées, et non pas la localisation explicite des points dans le domaine de calcul. Aussi, dans la littérature, les méthodes Level Set sont souvent basées sur des schémas de type différences finies, comme dans [Sussman, 1998], avec des restrictions importantes (géométries et maillages réguliers). Or, nous avons ici une approche exclusivement basée sur les éléments finis qui ne sous-entend aucune restriction de la sorte, et qui se veut donc générale. Par contre, la formulation standard des éléments finis (Galerkin standard) se comporte mal sur ce type de problème, et il est nécessaire d'ajouter des techniques de stabilisation et de réinitialisation.

La nature continue du Level Set offre des avantages non négligeables, mais elle nécessite l'implémentation d'une technique de réinitialisation pour limiter la détérioration des données tout au long d'une simulation. Des équations de Hamilton-Jacobi sont généralement utilisées dans ce cas. Elles s'apparentent à des équations de transport, et se résolvent de la même manière. Au final, notre implémentation est directe, facile à utiliser, en entièrement basée sur les éléments finis.

En plus d'appliquer ce genre de méthode bien connue, nous introduisons plusieurs techniques visant à alléger les calculs. Certaines propriétés des équations de Hamilton-Jacobi sont utilisées pour créer une méthode Level Set locale qui évite d'avoir à étendre la propagation sur tout le domaine. De plus, des critères de déclanchement de réinitialisation sont conçus pour détecter au mieux les moments au cours de la simulation où cette opération est absolument nécessaire, sans que des calculs superflus ne soient effectués. La méthode Level Set locale avec critère automatique présentée ici est très facile à implémenter et contient des avantages importants par rapport à une méthode Level Set globale; et ce, dans n'importe quelle dimension on se place. Finalement, une technique innovante de couplage fort entre les étapes de transport et de réinitialisation est brièvement présentée.

4.2. Modélisation hétérogène

4.2.1. La fonction Level Set

Commençons par présenter la méthode Level Set standard, telle qu'elle fut développée à l'origine dans [Osher, 1988], ainsi que ses conventions, qui seront employées par la suite. Considérons une « fonction Level Set » : fonction distance signée à l'interface qui sert à distinguer les différentes phases présentes dans un écoulement. L'ensemble de la cavité constitue un domaine fixe Ω qui correspond au domaine de calcul dans son intégralité. Ω est divisé en plusieurs sous domaines qui eux ne sont pas fixes et qui dépendent du temps t :

$$\Omega = \bigcup \Omega_i(t) \qquad (21)$$

Un sous domaine correspond à une phase présente dans l'écoulement simulé. Elle peut être gazeuse, liquide ou solide. De plus, les sous domaines servent à différencier les différents gaz, fluides et solides s'il y en a plusieurs. Ils ne se chevauchent pas, mais peuvent être non connexes :

$$\bigcap \Omega_i(t) = 0 \qquad (22)$$

Nous considérons l'interface $\Gamma_i(t)$ du sous domaine $\Omega_i(t)$ et qui se déplace aussi en fonction du temps. La méthode Level Set consiste à introduire une fonction continue $\alpha_i(x,t)$ associée à chaque sous domaine $\Omega_i(t)$, et dont la courbe de niveau zéro représente l'interface, c'est-à-dire :

$$\Gamma_i(t) = \{x, \alpha_i(x,t) = 0\} \qquad (23)$$

La fonction Level Set $\alpha_i(x,t)$ est définie et continue sur tout le domaine de calcul Ω telle que toutes ses valeurs situées d'un même côté de l'interface $\Gamma_i(t)$ soient du même signe. Celles situées de l'autre côté sont de signe opposé.

Comme nous le verrons par la suite, la fonction Level Set doit être la plus régulière possible afin de garder une bonne précision numérique. En fait, il est bien qu'elle se rapproche au maximum d'une fonction distance signée à l'interface qui a la propriété :

$$\left|\nabla \alpha_i\right| = 1 \qquad (24)$$

en plus d'être nulle sur $\Gamma_i(t)$.

Il est donc possible et judicieux de définir cette fonction comme une fonction distance signée à l'interface :

$$\begin{cases} \alpha_i(x,t) = dist(x,\Gamma_i(t)) & \text{dans } \Omega_i(t) \\ \alpha_i(x,t) = 0 & \text{sur } \Gamma_i(t) \\ \alpha_i(x,t) = -dist(x,\Gamma_i(t)) & \text{dans } \Omega - \Omega_i(t) \end{cases} \qquad (25)$$

où $x \in R^d$, $t \in R^+$ et d est la dimension spatiale. L'initialisation d'une telle fonction distance peut être facilement construite dans la plupart des applications.

Similairement à la méthode *VOF*, une équation de convection pure transporte la fonction scalaire pour assurer l'évolution de l'interface. Pour le montrer, on considère une particule $x(t)$ placée sur l'interface $\Gamma(t)$. Son mouvement est alors décrit par :

$$\left.\frac{d\alpha(x,t)}{dt}\right|_{x(t)\in\Gamma} = 0 \qquad (26)$$

Puis, après extension, on généralise cette relation à l'ensemble des lignes de niveau, et on obtient :

$$\frac{d\alpha(x,t)}{dt} = 0 \qquad \forall x \in \Omega \qquad (27)$$

Cela constitue donc une équation de transport avec laquelle la fonction Level Set est mise à jour. Ensuite, il suffit de trouver le niveau zéro de la solution pour décrire l'interface après évolution. A noter que seul le transport du niveau zéro de α a un sens physique : la généralisation aux autres lignes de niveau est simplement une extension pratique.

Néanmoins, l'extension *(27)* ne gardant pas la propriété de fonction distance, de forts gradients peuvent apparaître, ce qui conduit à des termes instables de convection. De plus, l'importance des mouvements des phases en présence dans l'écoulement accentue ce phénomène. Une technique possible pour y remédier est présentée ici sous le nom réinitialisation. Une méthode de localisation permet juste de réinitialiser la fonction Level Set dans une région proche de l'interface, car ailleurs dans la cavité, il n'est pas nécessaire d'avoir une fonction distance. Puis, des critères automatiques de réinitialisation sont présentés pour que celle-ci ne soit déclanchée seulement aux moments nécessaires au cours de la simulation numérique.

Dans le cadre d'un formalisme Eulérien, équation scalaire *(27)* de la propagation de l'interface à caractère hyperbolique revient à résoudre ce problème fort :

$$\begin{cases} \dfrac{\partial \alpha(x,t)}{\partial t} + \nabla \alpha(x,t).v = 0 & \text{dans } \Omega \\ \alpha(x,0) = \alpha_0 & \text{dans } \Omega \\ \alpha(x,t) = g & \text{sur } \partial\Omega^- \end{cases} \qquad (28)$$

où $\partial\Omega^- = \{x \in \partial\Omega, v(x).n(x) < 0\}$ est le bord amont de la cavité.

A partir de cette modélisation, nous pouvons décrire quelques caractéristiques appréciables de la formulation Level Set : certaines quantités géométriques ont une représentation très simple en fonction de α :

$$\hat{n} = \frac{\nabla \alpha}{|\nabla \alpha|} \qquad (29)$$

$$\kappa = \nabla \cdot \frac{\nabla \alpha}{|\nabla \alpha|} \qquad (30)$$

où $\hat{n}$ est le vecteur normal unitaire, et κ est la courbure de l'interface. Puis, on constate également que la méthode peut être généralisée à *N* dimensions sans contrainte particulière.

4.2.2. Loi de mélange

La modélisation d'un écoulement de fluide hétérogène se fait de la même façon que dans la méthode *VOF* : une loi de mélange dépend de la fonction de phase (ici, la fonction Level Set) et des caractéristiques de chaque fluide.

Pour rendre non homogène le problème de Navier-Stokes, nous appliquons la loi de mélange suivante pour la masse volumique et la viscosité :

$$\begin{aligned} \rho &= f(\alpha)\rho_f + (1 - f(\alpha))\rho_a \\ \eta &= f(\alpha)\eta_f + (1 - f(\alpha))\eta_a \\ f(\alpha) &\in [0,1] \end{aligned} \qquad (33)$$

où :

ρ_f est la masse volumique du fluide

η_f est la viscosité du fluide

ρ_a est la masse volumique de l'air

η_a est la viscosité de l'air

$f(\alpha)$ est une fonction de α comprise entre 0 et 1.

La formulation non homogène du solveur d'écoulement est donc :

$$\begin{cases} \rho\dfrac{\partial v}{\partial t} + \rho\nabla v.v - \nabla\cdot\left(2\eta\varepsilon(v)\right) + \nabla p = \rho g \\ \nabla\cdot v = 0 \\ \rho = f(\alpha)\rho_f + (1-f(\alpha))\rho_a \\ \eta = f(\alpha)\eta_f + (1-f(\alpha))\eta_a \end{cases} \quad \text{dans } \Omega \qquad (34)$$

Avec la loi de mélange *(34)*, le comportement du fluide guide l'écoulement dans le sous domaine du fluide : $\rho = \rho_f$ et $\eta = \eta_f$ dans Ω_f ; et c'est celui de l'air qui est considéré dans le teste de la cavité : $\rho = \rho_a$ et $\eta = \eta_a$ dans $\Omega - \Omega_f$. Par contre, une zone de mélange existe lorsque $f(\alpha)$ a des valeurs intermédiaires (entre 0 et 1). Le choix de la fonction $f(\alpha)$ est donc très important, car, entre autre, c'est elle qui détermine la zone de mélange des différents fluides, qui se traduit par une diffusion de l'interface.

4.2.3. Choix de la fonction de mélange $f(\alpha)$

La formulation Level Set offre une certaine liberté dans la modélisation hétérogène : le choix de la fonction de mélange $f(\alpha)$ détermine le profil du mélange des différents sous domaines, et donc, influe sur l'interface. Plus elle est abrupte, moins la zone de diffusion de l'interface est grande. Dans la littérature [Sussman, 1998] et [Sussman, 1999], des fonctions lissées de type *heaviside* sont souvent utilisées afin de limiter la zone de diffusion de l'interface :

$$f(\alpha) = \begin{cases} 0 & \text{si } \alpha < -\varepsilon \\ \dfrac{1}{2}\left(1 + \dfrac{\alpha}{\varepsilon} + \dfrac{1}{\pi}\sin(\pi\alpha/\varepsilon)\right) & \text{si } |\alpha| \le \varepsilon \\ 1 & \text{si } \alpha > \varepsilon \end{cases} \qquad (35)$$

où ε est une zone définie autour de l'interface.

Dans notre solveur d'écoulement, les grandeurs de viscosité et de masse volumique sont prises en comptent constantes sur les éléments. La fonction $f(\alpha)$ doit donc, elle aussi, avoir cette interpolation *P0*, bien que le scalaire α (la fonction Level Set) soit interpolé linéairement sur les nœuds du maillage (*P1*). Autrement dit, cette fonction joue exactement le rôle de la fonction de présence 1_{Ω_f} dans le *VOF* : elle représente la fraction volumique de fluide présent dans chaque élément.

$$f_K(\alpha) = \frac{\alpha_K^+}{|\alpha|_K} \qquad (36)$$

où

$f_K(\alpha)$ est définie constante sur l'élément *K* en fonction des valeurs aux nœuds de α

α_K^+ est la somme des α positifs aux nœuds de l'élément *K*

$|\alpha|_K$ est la somme des valeurs absolue des α aux nœuds de l'élément *K*

Pour mieux comprendre la définition *(36)*, expliquons brièvement chacun des trois cas possibles :

- $f_K(\alpha)=1$ sur l'élément K : tous ses nœuds sont positifs, et donc tous à l'intérieur de Ω_f.
- $f_K(\alpha)=0$ sur l'élément K : tous ses nœuds sont négatifs, et donc tous à l'extérieur de Ω_f.
- $0 < f_K(\alpha) < 1$ sur l'élément K : certains de ses nœuds sont positifs, et d'autres sont négatifs. Dans ce cas, il est clair que K est traversée par l'interface Γ.

Pour conclure, la définition *(36)* fait disparaître toute diffusion, puisque seuls les éléments traversés par l'interface ont des valeurs $f_K(\alpha)$ intermédiaires entre 0 et 1. Aussi, il est clair que $f(\alpha)$ joue un rôle de fonction de présence, similairement à la fonction caractéristique de la méthode *VOF*.

4.3. Résolution de l'équation de transport continue

L'un des principaux attraits de la méthode Level Set réside dans sa formulation mathématique sous forme d'équations aux dérivées partielles. La qualité des calculs effectués dépend alors fortement des discrétisations spatiale et temporelle qui ont été développées. Dans ce contexte, les travaux réalisés sur la formulation éléments finis des équations de Navier-Stokes incompressibles semblent adaptés au formalisme Level Set. La simulation d'écoulements multi-fluide nécessite des méthodes numériques très robustes, capables de décrire des surfaces de discontinuité de certaines variables (comme la viscosité et la masse volumique). En contrepartie de leur grande robustesse, ces schémas peuvent introduire des erreurs numériques dissipatives qui ont tendance à lisser les solutions obtenues, notamment dans les zones de fort gradient (on parle de dissipation numérique).

4.3.1. Formulation forte

Considérons un champ de vitesse v solution du solveur d'écoulement de Navier-Stokes non homogène dans une cavité fixe Ω fermée et de dimension finie. Nous avons vu précédemment qu'une fonction de type de Level Set α doit être transportée grâce à une équation de convection pure telle que :

$$\frac{d\alpha(x,t)}{dt}=0 \qquad (37)$$

Afin de fermer ce problème, on doit lui ajouter des conditions initiales et des conditions limites, le plus souvent de type de Dirichlet :

$$\begin{cases} \dfrac{\partial \alpha(x,t)}{\partial t} + \nabla \alpha(x,t) v = 0 & \text{dans } \Omega \\ \alpha(x,0) = \alpha_0 & \text{dans } \Omega \\ \alpha(x,t) = g & \text{sur } \partial\Omega^- \end{cases} \qquad (38)$$

4.3.2. Formulation variationnelle

4.3.2.1. Notations

Les espaces fonctionnels A et A^0 sont nécessaires pour établir la formulation faible du problème tels que :

$$A = H^1(\Omega) = \left\{ q \in L^2(\Omega), \nabla q \in L^2(\Omega) \right\}$$
$$A^0 = \left\{ q \in H^1(\Omega), q = 0 \text{ sur } \partial\Omega^- \right\}$$

où $L^2(\Omega)$ est l'espace de Lebesgue des fonctions carrées sommables sur un domaine Ω, et $H^1(\Omega)$ est l'espace de Sobolev inclus dans $L^2(\Omega)$:

$$L^2(\Omega) = \left\{ q, \int_\Omega q^2 dV < \infty \right\}$$
$$H^1(\Omega) = \left\{ q \in L^2(\Omega), \nabla q \in L^2(\Omega) \right\}$$

Dans $L^2(\Omega)$, le produit scalaire est défini par :

$$(f_1, f_2) = \int_\Omega f_1 f_2 dV \qquad \forall f_1, f_2 \in L^2(\Omega) \qquad (39)$$

4.3.2.2. Forme faible

Ecrivons dans un premier temps le problème variationnel correspondant aux équations de convection pure sous sa forme la plus générale des éléments finis (Galerkin standard). Pour cela, on effectue le produit scalaire au sens L^2 par une fonction test α^* choisie dans l'espace A^0.

La forme faible du problème consiste à trouver $\alpha \in A$ tel que :

$$\int_\Omega \frac{\partial \alpha}{\partial t} . \alpha^* + \int_\Omega \nabla \alpha . v . \alpha^* = 0 \qquad \forall \alpha^* \in A^0 \qquad (40)$$

Soit, en appliquant la notation *(39)* :

$$\left(\frac{\partial \alpha}{\partial t}, \alpha^*\right) + \left(\nabla \alpha . v, \alpha^*\right) = 0 \qquad \forall \alpha^* \in A^0 \tag{41}$$

4.3.3. Formulation discrète

Afin d'établir la formulation éléments finis, les espaces vectoriels A_h et A_h^0 de dimensions finies sont construits de façon à approcher A et A^0. Le domaine de calcul Ω est alors décomposé de la même façon que dans le solveur des écoulements de Navier-Stokes avec un maillage Eulérien $T_h(\Omega)$ de tétraèdres *K*, où *h* est la taille de maille. Plus cette taille de maille est petite, plus l'approximation des espaces fonctionnels est précise : $\lim_{h\to 0} A_h = A$ et $\lim_{h\to 0} A_h^0 = A^0$. Nous avons ainsi une discrétisation spatiale basée sur le maillage :

$$\Omega_h = \bigcup_{K \in T_h(\Omega)} K$$

La fonction Level Set α étant continue, sont interpolation spatiale est linéaire, et ses degrés de liberté sont situés sur les nœuds du maillage, comme le montre la figure *(4)* :

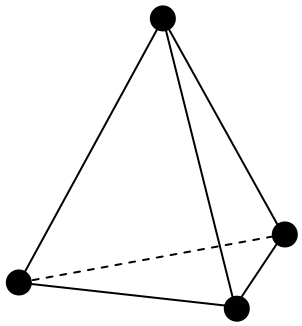


Figure 4 – Interpolation linéaire de α en 3D

A l'aide de cette discrétisation, α peut être approchée par la solution $\alpha_h \in A_h$ de ce problème discret :

$$\left(\frac{\partial \alpha_h}{\partial t}, \alpha_h^*\right) + \left(\nabla \alpha_h . v, \alpha_h^*\right) = 0 \qquad \forall \alpha_h^* \in A_h^0 \tag{42}$$

4.3.4. Schémas temporels

Nous choisissons de discrétiser temporellement le problème de convection à l'aide d'une méthode de différences finies de type de Euler. Soit $T > 0$ la durée du phénomène, et $t \in [0, T]$. Tout d'abord, l'espace temps $[0, T]$ est partitionné en *N* intervalles réguliers de manière suivante :

$$[0, T] = \bigcup_n [t_{n-1}, t_n]$$

où le pas de temps est défini par :

$$\Delta t = \frac{T}{N}$$

La discrétisation temporelle la plus simple consiste à écrire :

$$\frac{\partial \alpha}{\partial t} = \frac{\alpha(t_n) - \alpha(t_{n-1})}{\Delta t}$$

En approximant $\alpha(t_n)$ par α_h et $\alpha(t_{n-1})$ par α_h^-, nous avons :

$$\frac{\partial \alpha_h}{\partial t} = \frac{\alpha_h - \alpha_h^-}{\Delta t}$$

Plusieurs types de schéma temporel sont issus de cette approximation, et nous les appliquons à l'équation de transport *(42)*.

- Schéma implicite

$$\frac{\alpha_h - \alpha_h^-}{\Delta t} + \nabla \alpha_h v = 0 \qquad (43)$$

où α_h est l'inconnue de l'incrément de temps présent, et α_h^- est la solution de l'incrément de temps précédent.

L'avantage de ce schéma d'ordre 1 est qu'il est stable sous aucune condition. Le pas de temps Δt n'est donc pas limité. Cependant, un pas de temps trop grand peut causer des imprécisions importantes de part l'ordre du schéma.

La forme faible du problème de convection avec un schéma implicite est la suivante :

$$\left(\frac{\alpha_h}{\Delta t}, \alpha_h^*\right) + \left(\nabla \alpha_h . v, \alpha_h^*\right) = \left(\frac{\alpha_h^-}{\Delta t}, \alpha_h^*\right) \qquad \forall \alpha_h^* \in A_h^0 \qquad (44)$$

- Schéma semi implicite

$$\frac{\alpha_h - \alpha_h^-}{\Delta t} + \frac{1}{2}\left(\nabla \alpha_h v + \nabla \alpha_h^- v\right) = 0 \qquad (45)$$

Ce schéma d'ordre 2 est issu d'un modèle de Runge Kutta appelé θ-schémas :

$$\frac{\alpha_h - \alpha_h^-}{\Delta t} + (\theta)\nabla \alpha_h v + (1-\theta)\nabla \alpha_h^- v = 0 \qquad (46)$$

où nous fixons $\theta = 1/2$ pour obtenir la formule semi implicite *(45)*.

Lors de la validation de ce solveur de transport continu, une comparaison numérique sera effectuée entre ces deux types de schéma temporel. Nous verrons notamment que le schéma semi implicite montre de bien meilleurs résultats, aussi bien pour le transport lui-même, que pour la conservation.

La forme faible du problème de convection avec un schéma semi implicite est la suivante :

$$\left(\frac{\alpha_h}{\Delta t}, \alpha_h^*\right) + \frac{1}{2}\left(\nabla \alpha_h . v, \alpha_h^*\right) = \left(\frac{\alpha_h^-}{\Delta t}, \alpha_h^*\right) - \frac{1}{2}\left(\nabla \alpha_h^- . v, \alpha_h^*\right) \qquad \forall \alpha_h^* \in A_h^0 \qquad (47)$$

4.3.5. Stabilisations numériques

4.3.5.1. Nature des instabilités

Les méthodes numériques de type éléments finis – et leur formulation classique par Galerkin standard – se comportent généralement bien lorsqu'il s'agit de modéliser des écoulements de fluides. Cependant, elles sont peu appropriées à la résolution de problèmes dominés par un phénomène de convection. La raison principale vient de la propriété de différence centrée des méthodes de Galerkin standard : elles font apparaître des oscillations erronées dans la solution quand le problème a un caractère hyperbolique dominant. Il est montré que l'utilisation du Galerkin standard crée des instabilités structurelles dès lors que les termes de convection prennent le dessus sur ceux de diffusion. Il faut donc trouver des techniques pour stabiliser la résolution des problèmes de convection.

Premièrement, on observe que le Galerkin standard donne de bien meilleurs résultats lorsque la quantité à transporter est régulière, comme c'est le cas avec une fonction Level Set. Deuxièmement, des méthodes de stabilisations sont très fréquemment utilisées, comme SUPG ou *Residual-Free Bubbles* (RFB) [Brezzi, 1994], inspirée de la philosophie du multiscale.

4.3.5.2. Apport d'une fonction régulière

Les problèmes linéaires hyperboliques sont généralement mal pris en charge par les méthodes de Galerkin standard qui provoquent d'importantes oscillations dans la solution. Cependant, on remarque que, dans certains cas, la solution peut se montrer de bien meilleure qualité, sans pour autant changer la méthode de résolution, ni rajouter une technique de stabilisation. Dans notre cas, c'est le profil de la fonction α à transporter qui détermine l'efficacité de la résolution lorsque seulement du Galerkin standard est utilisé.

Nous montrons ici l'influence que peut avoir la forme de la fonction de phase sur le transport de l'interface. Dans une cavité 2D en forme de marche, un champ de vitesse est fixé à l'aide d'un solveur de Stokes qui simule des écoulements stationnaires de fluides Newtoniens incompressibles (voir figure *5*). Dans cette simulation, on s'appuis sur un maillage plutôt grossier contenant 5000 nœuds et 15000 éléments.

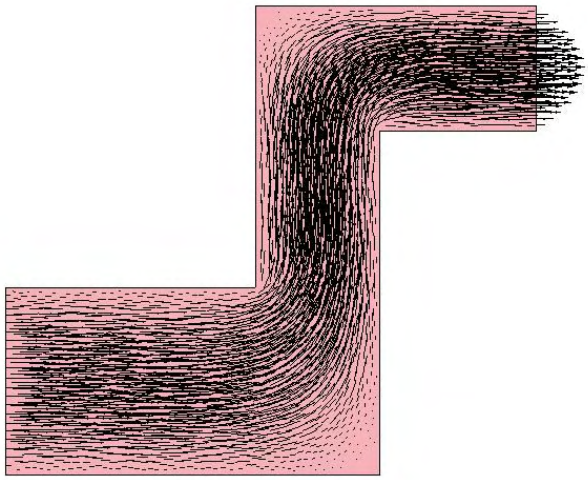


Figure 5 – Champ fixe des vitesses

En gardant cette vitesse constante au cours du temps, une fonction de phase est convectée sur plusieurs incréments de temps avec l'équation de transport *(38)*. Une méthode éléments finis de type Galerkin standard est simplement utilisée pour la résolution. Avec un pas de temps fixé à 0.5 seconde, 54 incréments de temps sont réalisés pour transporter la fonction pendant 27 secondes. Dans un premier temps, la fonction linéaire est initialisée avec une pente très abrupte autour de son niveau zéro. La figure *(6)* montre son profil en 1D :

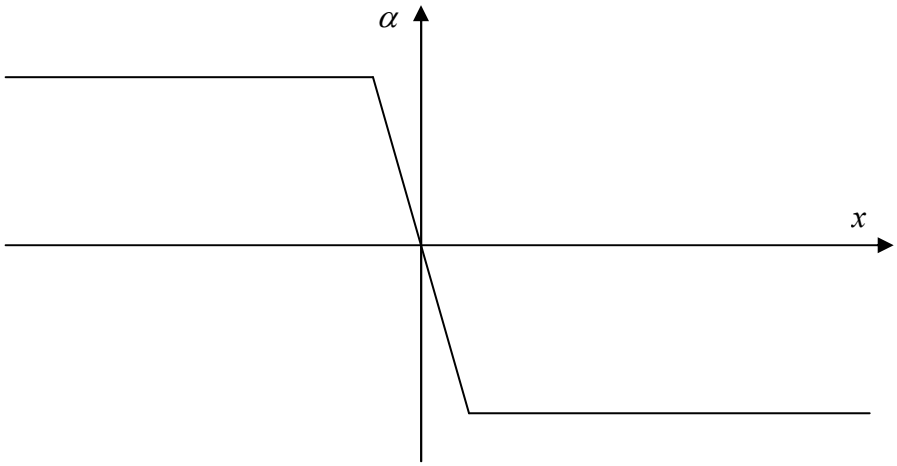


Figure 6 – Graphe d'une fonction abrupte en 1D

La figure *(7)* montre cette fonction dans son état initial (à $t = 0s$), puis à la fin de la simulation (à $t = 27s$), après avoir été transportée avec le champ de vitesse de la figure *(5)* :

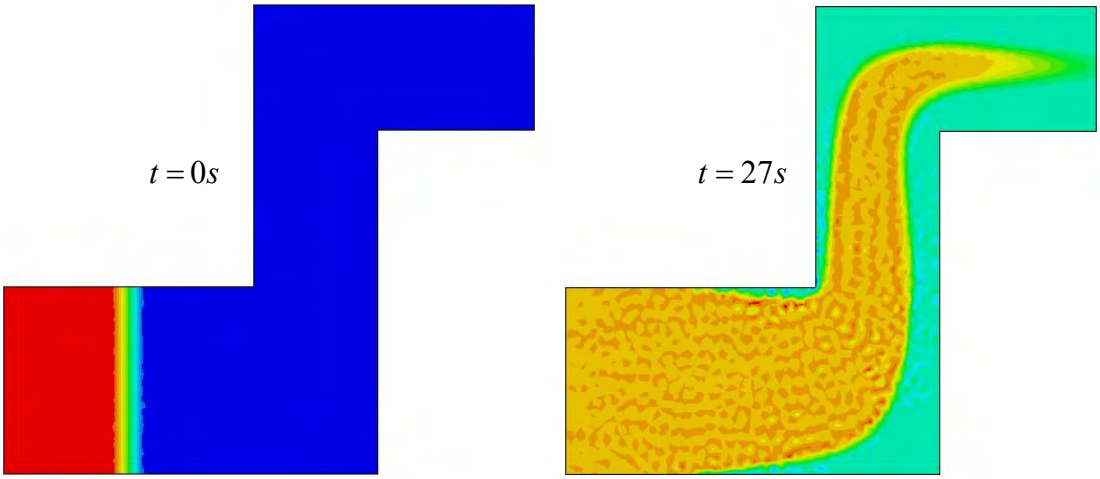


Figure 7 – Transport d'une fonction continue et abrupte

On remarque sur ces résultats que le transport de la fonction abrupte n'est pas efficace. La solution de l'équation *(38)* comporte d'importantes oscillations, non seulement dans la zone proche du niveau zéro (zone initialement abrupte), mais aussi partout ailleurs, dans le reste de la cavité. Il y a donc des instabilités numériques considérables qui empêchent une bonne résolution.

Maintenant, on considère une fonction linéaire beaucoup plus régulière que la précédente. On l'initialise avec une pente constante, qui en fait, la rend similaire à une fonction Level Set. En effet, elle peut être vue comme une fonction distance signée. Pour la comparer à la fonction abrupte précédente, voici son profil en 1D :

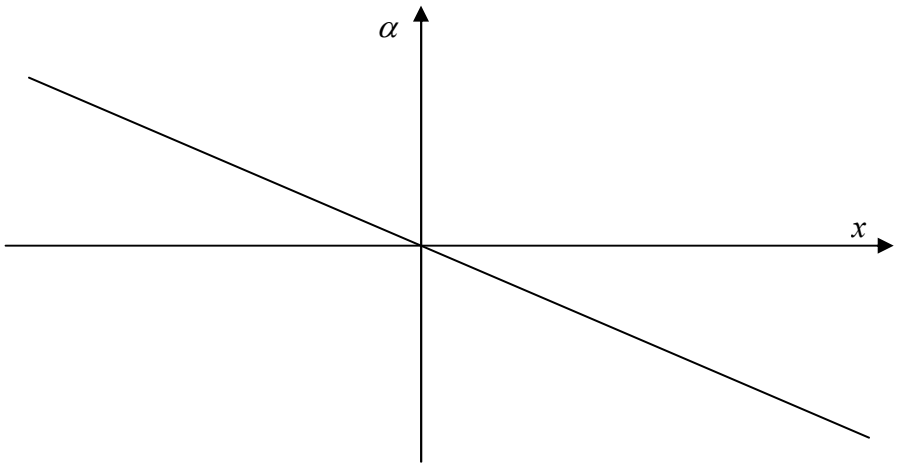


Figure 8 – Graphe d'une fonction régulière en 1D

La même simulation que précédemment est réalisée (avec un pas de temps de 0.5 seconde, sur 54 incréments de temps, etc.). Les résultats obtenus sont illustrés par la figure *(9)* :

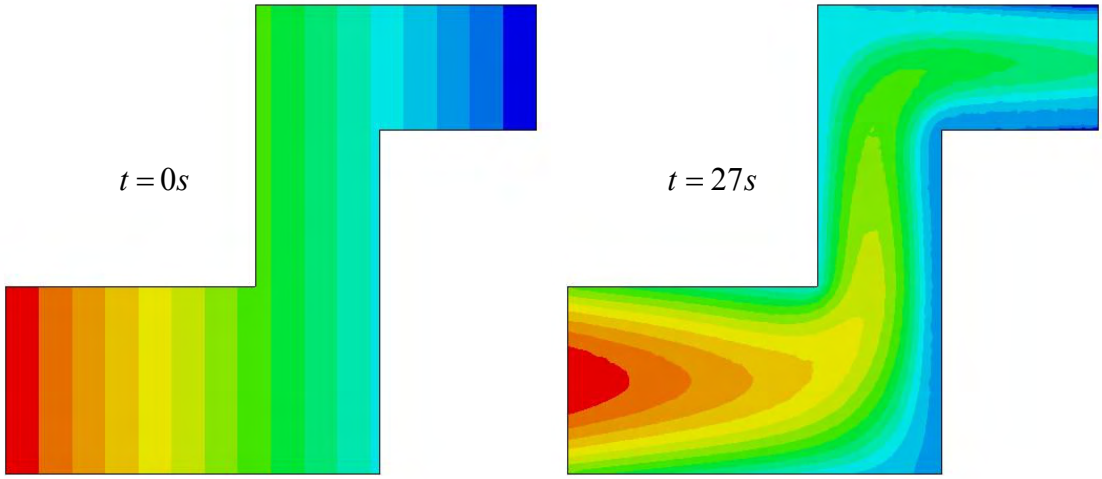


Figure 9 – Transport d'une fonction continue et régulière

Ces résultats sont très différents de ceux présentés dans le cas d'une fonction abrupte. On voit très clairement que les oscillations ont totalement disparut. Ce cas test met donc en évidence la bonne portabilité d'une fonction régulière à l'équation de transport linéaire *(38)*. Les méthodes de Galerkin standard, pourtant mal adaptées aux problèmes de convection, réagissent beaucoup mieux aux fonctions Level Set qu'aux autres fonctions moins régulières. Cela représente donc le premier intérêt que l'on peut porter aux méthodes de Level Set : la résolution des équations de transport est facilitée.

4.3.5.3. SUPG

Il existe plusieurs techniques de stabilisation qu, appliquées aux formulations éléments finis, permettent de résoudre des problèmes hyperboliques linéaires. Afin de les décrire de façon générale, on considère le problème de convection-diffusion suivant :

$$\left(L\left(v_h\right), w_h\right) = \left(f, w_h\right) \qquad (48)$$

où L est la somme des termes de diffusion et de convection.

Les méthodes stabilisées sont construites de manière à conserver la consistance du problème : la solution du problème continu converge vers la solution du problème discret. D'une manière générale, elles consistent à rajouter un terme de diffusion le long des lignes de courant dans la formulation Galerkin standard. Ce terme peut s'écrire comme le produit, sur chaque élément, du résidu par un opérateur appliqué aux fonctions tests que l'on notera L_{op} et par un paramètre τ_K constant sur chaque élément K. La forme finale du problème est alors :

$$\left(L\left(v_h\right), w_h\right) + \sum_K \tau_K \left(r, L_{op}\left(w_h\right)\right)_K = \left(f, w_h\right) \qquad (49)$$

où $r = L\left(v_h\right) - f$ est le résidu de l'équation.

Autrement dit, la stabilisation des problèmes dominés par la convection passe par l'ajout de la formule :

$$stab(v_h, w_h) = \sum_K \tau_K \left(r, L_{op}(w_h)\right)_K \qquad (50)$$

Ce terme de stabilisation rajouté s'annule avec le résidu du problème *(42)*, c'est-à-dire lorsque la solution est atteinte. Il permet de traiter le caractère hyperbolique non symétrique des termes de convection en décalant la fonction test en direction opposée à la vitesse de convection. Ainsi, la solution se retrouve beaucoup plus stable et ne laisse apparaître quasiment plus d'oscillations.

La formule *(49)* représente la forme générale des techniques de stabilisation pour les problèmes de convection-diffusion. En effet, nous verrons par la suite que plusieurs méthodes venant de différentes philosophies, se retrouvent dans cette formule en jouant sur le paramètre τ_K. Nous présentons ici les techniques SUPG et RFB, ainsi que les liens qui les unissent. La première méthode *Streamline Upwind / Petrov-Galerkin* (SUPG) est largement reconnue dans la littérature depuis de longues années. Elle est basée sur la création d'un effet *upwind* à la discrétisation spatiale. La deuxième technique appelée *Residual-Free Bubbles* (RFB) [Brezzi, 1994] est inspirée par la philosophie des méthodes de type *multiscale* qui considèrent plusieurs échelles de résolution avec l'utilisation d'espaces fonctionnels enrichis par des fonctions de bulle. Ce référer à [Batkam, 2002] et [Magnin, 1994] pour avoir des détails sur le traitement des problèmes d'advection tels que SUPG, ST/SGS (*Space Time / SubGrid Scale*), CG (Caractéristiques), et TG (Taylor Galerkin).

Une des techniques les plus populaires est celle introduite dans [Brooks, 1982], appelée SUPG (*Streamline Upwind/Petrov-Galerkin*). Cette méthode, devenue classique, fait disparaître la plupart des oscillations mises en évidence par la figure *(7)*. Elle fait partie des méthodes de type Petrov Galerkin qui consistent à formuler un problème faible discret à l'aide d'une fonction test qui n'est pas choisie dans le même espace que celui dans lequel on cherche la solution. On construit donc un nouvel espace de discrétisation $\widetilde{A}_h$ de même dimension que A_h, et auquel appartient la fonction test $\widetilde{\alpha}_h^*$. Enfin, le problème faible discret suivant est pose :

Trouver $\alpha_h \in A_h$, $\forall \widetilde{\alpha}_h^* \in \widetilde{A}_h$ tel que :

$$\left(\frac{\partial \alpha_h}{\partial t}, \widetilde{\alpha}_h^*\right) + \left(\nabla \alpha_h . v, \widetilde{\alpha}_h^*\right) = 0 \qquad (51)$$

Cela revient à substituer la fonction test α_h^*, de la forme faible discrète *(42)* du problème de convection pure, par une autre fonction $\widetilde{\alpha}_h^*$.

Remarque : Une autre méthode, nommée SU, consiste à modifier la fonction test uniquement pour le terme de convection $\nabla \alpha . v$:

$$\left(\frac{\partial \alpha_h}{\partial t}, \alpha_h^*\right) + \left(\nabla \alpha_h . v, \widetilde{\alpha}_h^*\right) = 0 \qquad (52)$$

Par conséquent, ce problème ne fait pas parti des méthodes de Petrov Galerkin. Cette formulation variationnelle n'est pas consistante, c'est-à-dire que la solution α du problème

fort ne vérifie pas tout à fait ces équations. Toutefois, la solution α_h du problème modifié *(52)* tend quand même vers celle du problème fort α lorsque *h* tend vers zéro. Mais cette technique apporte généralement une diffusion numérique excessive. C'est pourquoi, SUPG est préférée pour stabiliser les problèmes dominés par la convection.

Dans la méthode SUPG, on construit les fonctions test de la façon suivante :

$$\tilde{\alpha}_h^* = \alpha_h^* + \tau_K^{SUPG}.\nabla\alpha_h^*.v \qquad (53)$$

Cette substitution agit comme l'ajout d'une diffusion numérique, mais seulement dans la direction *upwind*, c'est-à-dire dans l'axe donné par la vitesse de convection *v*. Le paramètre de stabilisation τ_K^{SUPG} (constant par élément) dépend de la taille de maille et de la norme de la vitesse. Généralement [Brooks, 1982], on lui donne la valeur suivante :

$$\tau_K^{SUPG} = \frac{1}{2}\frac{h_K}{\left|v_{|K}\right|} \qquad (54)$$

où $v_{|K}$ est un champ de vitesse constant sur l'élément *K*. Pour obtenir sa valeur, il suffit de prendre la moyenne des vitesses aux nœuds de *K*.

Si on développe les produits de la formule *(51)*, on obtient :

$$\left(\frac{\partial\alpha_h}{\partial t}, \alpha_h^* + \tau_K^{SUPG}.\nabla\alpha_h^*.v\right) + \left(\nabla\alpha_h.v, \alpha_h^* + \tau_K^{SUPG}.\nabla\alpha_h^*.v\right) = 0 \qquad (55)$$

Puis :

$$\overbrace{\left(\frac{\partial\alpha_h}{\partial t}, \alpha_h^*\right) + \left(\nabla\alpha_h.v, \alpha_h^*\right)}^{(1)} + \overbrace{\tau_K^{SUPG}\left(\frac{\partial\alpha_h}{\partial t}, \nabla\alpha_h^*.v\right)}^{(2)} + \overbrace{\tau_K^{SUPG}\left(\nabla\alpha_h.v, \nabla\alpha_h^*.v\right)}^{(3)} = 0 \qquad (56)$$

On reconnaît alors clairement la forme de Galerkin standard (partie 1), à laquelle sont ajoutés deux termes supplémentaires (parties 2 et 3). Le premier (partie 2) contient quasiment le symétrique du terme convectif de la forme de Galerkin standard ; et le deuxième (partie 3) correspond au terme symétrique à caractère elliptique qui crée la diffusion numérique recherchée le long des lignes de courant (pour donner l'effet *upwind*).

Rappelons que le transport d'une fonction raide (non régulière) avec simplement du Galerkin standard fait apparaître des oscillations qui polluent considérablement la solution (voir figure *7*). Maintenant, on réexécute ce même cas test, mais cette fois-ci, en appliquant une technique de stabilisation de type SUPG. La figure *(10)* suivante montre que les résultats sont bien meilleurs, et que les oscillations ont pratiquement disparut. Par conséquent, l'ajout de la technique SUPG stabilise efficacement la résolution de l'équation de transport *(38)*.

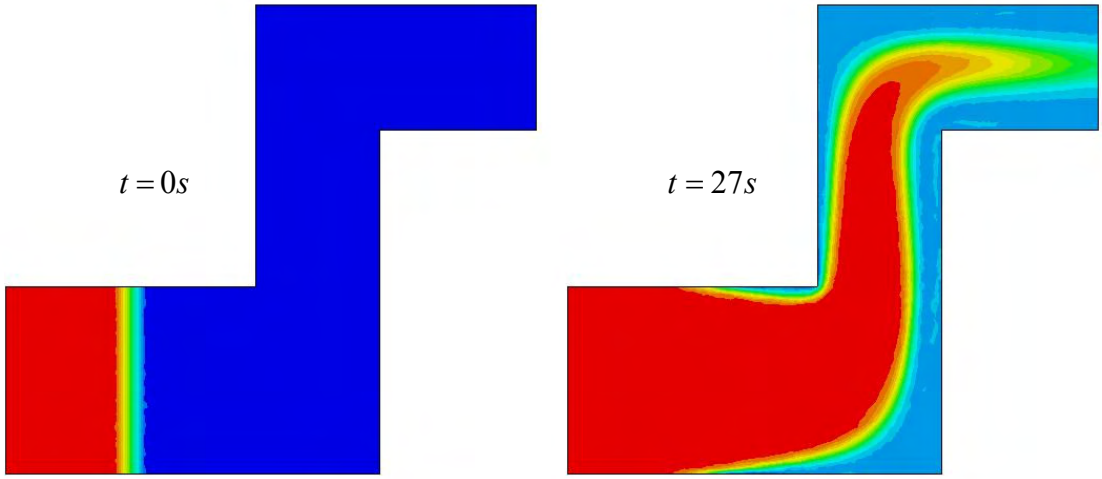


Figure 10 – Transport d'une fonction continue et abrupte avec SUPG

4.3.5.4. *Residual-Free Bubbles*

La philosophie de cette méthode RFB s'appuie sur l'hypothèse que les instabilités dans la résolution du problème d'advection peuvent être résolues en prenant en compte les effets des petites échelles sur les grandes [Brezzi, 1997].

Rappelons que la formulation faible du problème de convection considéré consiste à trouver $\alpha \in A$ tel que :

$$\left(\frac{\partial \alpha}{\partial t}, \alpha^*\right) + \left(\nabla \alpha . v, \alpha^*\right) = 0 \quad \forall \alpha^* \in A^0 \tag{57}$$

où la notation $(\cdot,\cdot)$ représente le produit scalaire dans l'espace $L^2(\Omega)$. Nous considérons toujours que le domaine de calcul spatial Ω est discrétisé par un maillage $T_h(\Omega)$ de tétraèdres (triangles en 2D), où *h* représente la taille de maille. Rappelons que les espaces discrétisés de dimensions finies A_h et A_h^0 approchent A et A^0 pour établir la formulation faible discrète du problème *(57)* qui consiste à trouver $\alpha_h \in A_h$ tel que :

$$\left(\frac{\partial \alpha_h}{\partial t}, \alpha_h^*\right) + \left(\nabla \alpha_h . v, \alpha_h^*\right) = 0 \quad \forall \alpha_h^* \in A_h^0 \tag{58}$$

On construit alors un nouvel espace de *Residual-Free Bubbles* A_{RFB} qui enrichie A_h avec un espace de bulle B_h tel que :

$$A_{RFB} = A_h \oplus B_h \tag{59}$$

L'espace de bulle est défini comme cela :

$$B_h = \bigoplus_{K \in T_h(\Omega)} span(b_K) \tag{60}$$

où la bulle b_K est choisie dans $H_0^1(K)$ telle que $\int_K b_K \neq 0$. D'ailleurs, la bulle généralement utilisée dans le MINI élément peut être appliquée ici. Puis, sur chaque élément *K*, on introduit dans [Brezzi, 1997] la fonction test $b_K^* \in B_h^*$ de la bulle $b_K \in B_h$ qui est définie comme la solution de :

$$\begin{cases} -v.\nabla b_K^* = 1 & \text{dans } K \\ b_K^* = 0 & \text{sur } \partial K^+ \end{cases} \qquad (61)$$

où le bord aval de l'élément *K* est défini par $\partial K^+ = \{x \in \partial K, v(x).n(x) > 0\}$, et l'espace fonctionnel de b_K^* est :

$$B_h^* = \bigoplus_{K \in T_h(\Omega)} span(b_K^*) \qquad (62)$$

De même manière que dans *(59)*, cet espace vient enrichir A^0 pour obtenir :

$$A_{RFB}^* = A_h^0 \oplus B_h^* \qquad (63)$$

La solution $\alpha_h \in A_{RFB}$ de l'équation de convection pure *(58)* peut être écrite d'une façon unique comme la somme d'une partie linéaire $\alpha_L \in A_h$ et d'une fonction bulle $\alpha_B \in B_h$:

$$\alpha_h = \alpha_L + \alpha_B \qquad (64)$$

où $\alpha_B = \sum_{K \in T_h(\Omega)} c_K b_K$. De même manière, la fonction test $\alpha_h^* \in A_{RFB}^*$ peut s'écrire de la façon suivante :

$$\alpha_h^* = \alpha_L^* + \alpha_B^* \qquad (65)$$

où $\alpha_B^* = \sum_{K \in T_h(\Omega)} c_K^* b_K^*$.

Avec la décomposition *(65)*, on peut séparer ce problème de façon suivante :
Trouver $\alpha_h \in A_{RFB}$ tel que :

$$\begin{cases} \left(\frac{\partial \alpha_h}{\partial t}, \alpha_L^*\right) + \left(\nabla \alpha_h.v, \alpha_L^*\right) = 0 & \forall \alpha_L^* \in A_h^0 \\ \left(\frac{\partial \alpha_h}{\partial t}, \alpha_B^*\right) + \left(\nabla \alpha_h.v, \alpha_B^*\right) = 0 & \forall \alpha_B^* \in B_h^* \end{cases} \qquad (66)$$

4.3.5.5. RFB ≈ SUPG

Dans [Franca, 2000], une technique de condensation de bulle est montrée dans la cas d'une équation de convection stationnaire du type de :

$$\left(\nabla \alpha_h . v, \alpha_h^*\right) = 0 \qquad \forall \alpha_h^* \in A_{RFB}^0 \tag{67}$$

où $\alpha_h \in A_{RFB}$.

Avec la décomposition *(65)*, on peut séparer ce problème de façon suivante :

$$\begin{cases} \left(\nabla \alpha_h . v, \alpha_L^*\right) = 0 & \forall \alpha_L^* \in A_h^0 \\ \left(\nabla \alpha_h . v, \alpha_B^*\right) = 0 & \forall \alpha_B^* \in B_h \end{cases} \tag{68}$$

Nous présentons ici cette technique de condensation de bulle qui, au final, se trouve très proche de la méthode SUPG.

- La première étape est de prendre comme fonction test, une fonction qui vaut b_K^* sur l'élément *K*, et qui s'annule partout ailleurs :

$$\alpha_h^* = \begin{cases} b_B^* & \text{sur } K \\ 0 & \text{ailleurs dans } \Omega_h \end{cases} \tag{69}$$

L'équation *(67)* devient donc :

$$\left(\nabla \alpha_h . v, b_K^*\right)_K = 0 \tag{70}$$

Avec la décomposition *(64)*, on obtient :

$$\begin{gathered} \left(\nabla\left(\alpha_L + \alpha_B\right) . v, b_K^*\right)_K = 0 \\ \left(\nabla\left(\alpha_L + c_K b_K\right) . v, b_K^*\right)_K = 0 \\ c_K \left(\nabla b_K . v, b_K^*\right)_K = -\left(\nabla \alpha_L . v, b_K^*\right)_K \end{gathered} \tag{71}$$

Après intégration par parties :

$$\left(\nabla b_K . v, b_K^*\right)_K = -\left(b_K, v . \nabla b_K^*\right)_K + \int_{\partial K} b_K b_K^* . v . n \tag{72}$$

Par définition de la bulle, b_K s'annule sur le bord ∂K de l'élément *K*. Aussi, avec la définition *(61)* pour b_K^*, nous avons, en moyenne sur l'élément *K*:

$$v . \nabla b_K^* = -1 \tag{73}$$

En combinant *(71)* et *(73)*, on obtient :

$$c_K = -\left(\nabla \alpha_L . v\right)_{|K} \frac{\int_K b_K^*}{\int_K b_K} \tag{74}$$

A noter que le choix de la fonction de bulle b_K assure la propriété $\int_K b_K \neq 0$. Il suffit donc de répéter cette opération sur chaque élément $K \in T_h(\Omega)$ pour avoir toutes les valeurs de c_K.

- La deuxième étape consiste à prendre α_L^* comme fonction test de l'équation *(67)* :

$$\alpha_h^* = \alpha_L^* \tag{75}$$

L'équation *(67)* devient donc :

$$\left(\nabla \alpha_h . v, \alpha_L^*\right) = 0 \tag{76}$$

Avec la décomposition *(64)*, on obtient :

$$\begin{gathered} \left(\nabla\left(\alpha_L + \alpha_B\right).v, \alpha_L^*\right) = 0 \\ \left(\nabla\left(\alpha_L + \sum_K c_K b_K\right).v, \alpha_L^*\right) = 0 \\ \left(\nabla \alpha_L . v, \alpha_L^*\right) + \sum_K c_K \left(\nabla b_K . v, \alpha_L^*\right)_K = 0 \end{gathered} \tag{77}$$

Avec une intégration par parties :

$$\left(\nabla b_K . v, \alpha_L^*\right)_K = -\left(b_K, v.\nabla \alpha_L^*\right)_K + \int_{\partial K} b_K . \alpha_L^* . v.n \tag{78}$$

où la dernière intégrale s'annule grâce à la propriété de la bulle $b_K = 0$ sur ∂K. Donc, nous avons :

$$\left(\nabla b_K . v, \alpha_L^*\right)_K = -\left(v.\nabla \alpha_L^*\right)_{|K} \int_K b_K \tag{79}$$

La formule *(77)* devient alors :

$$\left(\nabla \alpha_L . v, \alpha_L^*\right) - \sum_K c_K \left(v.\nabla \alpha_L^*\right)_{|K} \int_K b_K = 0 \tag{80}$$

Nous pouvons maintenant substituer la valeur de c_K donnée par *(74)* :

$$\left(\nabla\alpha_L.v,\alpha_L^*\right)+\sum_K\left(\nabla\alpha_L.v\right)_{|K}\frac{\int_K b_K^*}{\int_K b_K}\left(v.\nabla\alpha_L^*\right)_{|K}\int_K b_K=0 \qquad (81)$$

Après simplification, on obtient une équation de la forme suivante :

$$\left(\nabla\alpha_L.v,\alpha_L^*\right)+\sum_K\left[\frac{1}{|K|}\int_K b_K^*\right]\int_K\left(\nabla\alpha_L.v\right)\left(v.\nabla\alpha_L^*\right)=0 \qquad (82)$$

Cette équation coïncide exactement celle de la méthode SUPG avec cependant un nouveau paramètre de stabilisation τ_K^{RFB}. Sa valeur peut être facilement obtenue lorsque l'on considère que la fonction de base b_K^* de la bulle est une pyramide (en 2D) de hauteur $h_K^v/\left|v_{|K}\right|$, où $v_{|K}$ est un vecteur vitesse supposée constante sur l'élément *K*, et h_K^v est la taille du plus grand segment parallèle à $v_{|K}$ contenu dans l'élément *K*.

$$\tau_K^{RFB}=\frac{1}{|K|}\int_K b_K^* \qquad (83)$$

En considérant la hauteur de la bulle de l'élément *K* :

$$\int_K b_K^*=\frac{1}{3}\frac{h_K^v}{\left|v_{|K}\right|}|K| \qquad (84)$$

Et donc, le paramètre de stabilisation pour *RFB* est :

$$\tau_K^{RFB}=\frac{1}{3}\frac{h_K^v}{\left|v_{|K}\right|} \qquad (85)$$

Au final, nous avons donc la forme générale suivante :

$$\left(\nabla\alpha_L.v,\alpha_L^*\right)+\sum_K\tau_K\int_K\left(\nabla\alpha_L.v\right)\left(v.\nabla\alpha_L^*\right)=0 \qquad (86)$$

où le paramètre de stabilisation τ_K dépend de la méthode :

$$\tau_K=\tau_K^{SUPG}=\frac{1}{2}\frac{h_K}{\left|v_{|K}\right|} \qquad (87)$$

ou bien :

$$\tau_K=\tau_K^{RFB}=\frac{1}{3}\frac{h_K^v}{\left|v_{|K}\right|} \qquad (88)$$

4.3.6. Résolution du problème de convection

Rappelons que, conformément à la mise en place des formulations du paragraphe précédent, une méthode éléments finis est utilisée pour approcher le champ scalaire α par l'inconnue α_h interpolée linéairement sur le domaine de calcul Ω_h maillé avec une triangulation de tétraèdres $T_h(\Omega)$. Dans le cadre d'une simulation instationnaire d'écoulement multi-fluide, le temps est discrétisée avec un pas de temps Δt. Ainsi, à chaque incrément de temps, une résolution de l'écoulement par le solveur Navier-Stokes est suivit par un solveur d'évolution de phase dans lequel se situe la résolution de l'équation de transport.

Similairement au solveur de Navier-Stokes, une interpolation *P1* est utilisée, et donc, une inconnue par nœud doit être résolue. La création des matrices locales est réalisée par notre librairie CimLib, sous forme de solveur local. Ensuite, elles sont prisent en charge par l'outil PETSc pour le préconditionnement et la résolution des systèmes linéaires. La nature de ce problème de convection fait qu'il n'est pas très lourd à résoudre. Aussi, un préconditionnement par Block Jacobi est largement suffisant, et se montre très robuste. Enfin, puisque la matrice du système linéaire n'est pas symétrique à cause du terme de convection, c'est la méthode de type GMRES fournie par PETSc qui convient le mieux pour résoudre l'équation de transport.

4.4. Validations du solveur de convection pure

4.4.1. Cas test de la rotation d'un cercle

Afin de valider notre solveur de l'équation de transport *(38)* continue basé sur une méthode éléments finis stabilisée par SUPG, nous étudions le cas test utilisé précédemment pour valider notre solveur de l'équation de transport continue. Rappelons qu'il s'agit d'effectuer la rotation complète d'un cercle autour d'un axe à partir d'un champ de vitesse fixe dans une cavité carré en 2 dimensions. La figure *(2)* montre le champ de vitesse fixé pour faire transporter par convection le cercle dans le carré de dimension (1x1) avec les valeurs suivantes en fonction des coordonnées (*x,y*) des nœuds du maillage :

$$v = \begin{pmatrix} -\frac{\pi}{2}\left(y-\frac{1}{2}\right) \\ \frac{\pi}{2}\left(x-\frac{1}{2}\right) \end{pmatrix} \qquad (89)$$

Dans les formulations de l'équation de convection pure dressées de ce paragraphe, nous considérons deux schémas de discrétisation temporelle : le schéma implicite *(44)* d'ordre 1, et le schéma semi implicite *(47)* d'ordre 2. Donc, nous nous servons de ce cas test pour comparer ces deux modèles.

Nous considérons une géométrie circulaire avec un rayon de *0.1* centré sur le point (*0.7, 0.5*). L'intérieur de ce cercle défini un sous domaine qui va être transporté tout autour du point central de la cavité (*0.5, 0.5*) avec le champ de vitesse *(89)*, jusqu'à ce qu'il revienne à se place initiale (*0.7, 0.5*).

Comme dans le cas discontinu, nous prenons un pas de temps $\Delta t = 0.01$. Avec la vitesse définie par *(89)*, il faut 400 incréments de temps pour que le cercle effectue un tour complet : analytiquement, le sous domaine circulaire doit retrouver sa position initiale centrée sur le point (*0.7, 0.5*) au temps $t = 4$.

Initialement, à $t = 0$, une fonction Level Set α est définie de façon à ce que son niveau zéro soit exactement le cercle que nous considérons. Cette iso surface représente l'interface entre le sous domaine qui doit être transporté (l'intérieur du cercle) et le reste de la cavité. α est initialisé comme une fonction distance signée à l'interface, mais seulement dans une zone de taille $\varepsilon = 0.5$ autour de celle-ci :

$$\begin{cases} \alpha(x,0) = \min\left[dist(x,\Gamma(0)),\varepsilon\right] & \text{dans } \Omega_{\text{cercle}}(0) \\ \alpha(x,0) = 0 & \text{sur } \Gamma(0) \\ \alpha(x,0) = -\min\left[dist(x,\Gamma(0)),\varepsilon\right] & \text{dans } \Omega - \Omega_{\text{cercle}}(0) \end{cases} \qquad (90)$$

Le profil d'une telle fonction Level Set est dessiné sur la figure *(11)*. Elle est appelée fonction Level Set locale, car elle n'a ces caractéristiques de fonction distance seulement localement, autour de l'interface.

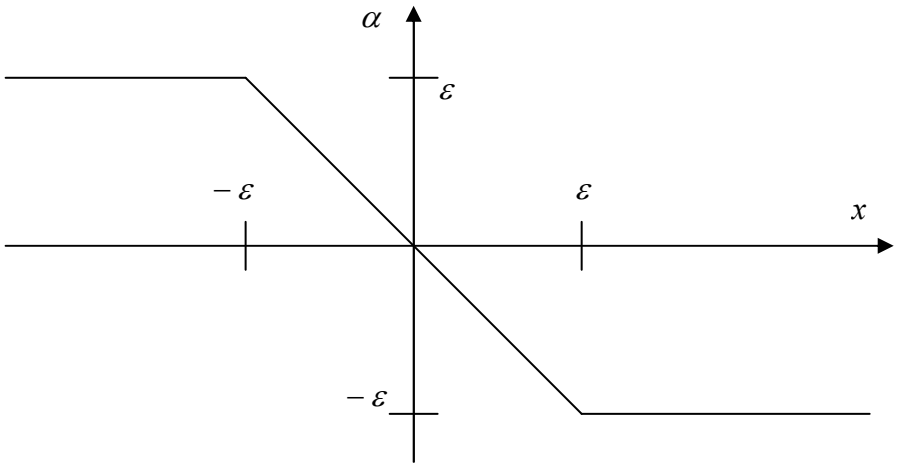


Figure 11 – Fonction Level Set locale

La motivation principale de l'utilisation d'une telle fonction vient de la simplification considérable des conditions limites. Nous verrons que ce choix n'affecte pas la qualité de la solution.

4.4.2. Schéma implicite en temps

La figure *(12)* montre la fonction Level Set α aux temps $t = 1$, $t = 2$, $t = 3$, et $t = 4$ de la rotation. Le trait noir est le niveau zéro de α, et donc l'interface qui décrit le cercle.

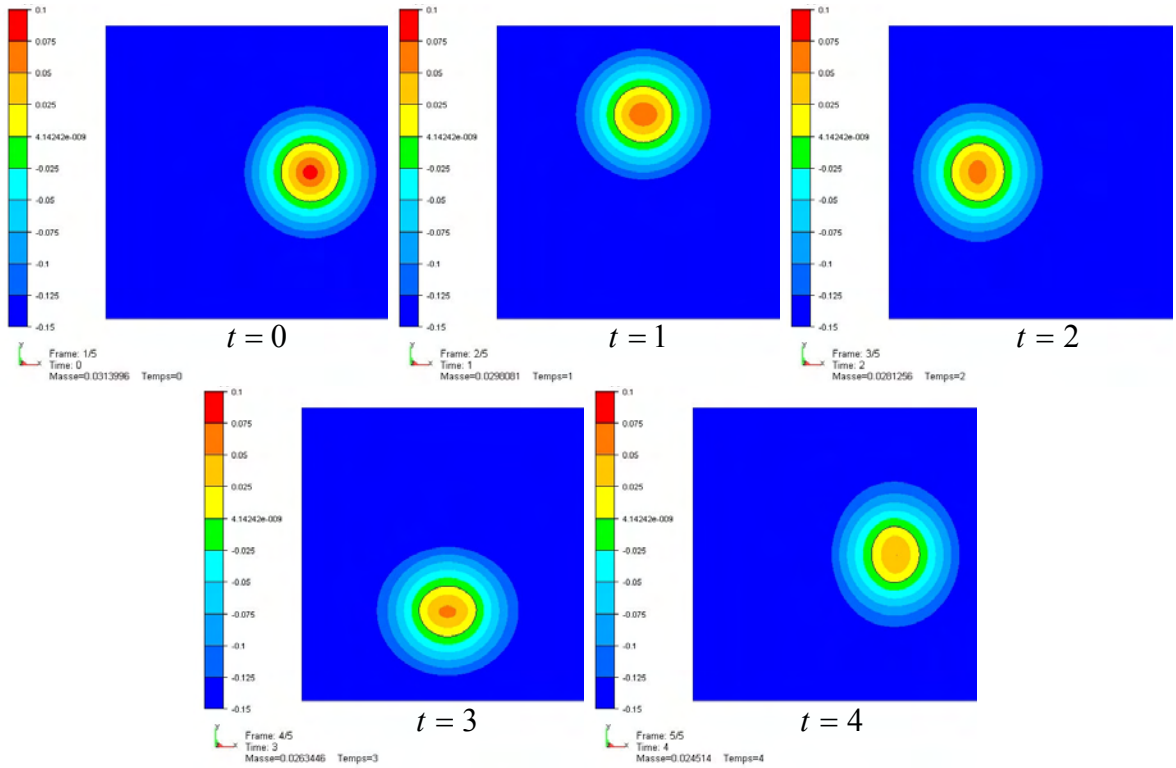


Figure 12 – Rotation d'un cercle avec un schéma implicite

Nous constatons que le transport du cercle n'est pas très bien réalisé. Non seulement, il se déforme, mais en plus, son volume n'est pas très bien conservé. Au départ, il occupe *3.14%* de la totalité de la cavité, mais à l'arrivée (à $t = 4$), il n'occupe plus que *2.45%*. Cette perte est loin d'être négligeable car elle représente *22%* de volume en moins.

Par contre, la solution ne comporte aucune trace d'oscillation, bien que la fonction initiale ne soit pas complètement régulière. L'utilisation d'une méthode de Galerkin standard ferait, dans ce cas, apparaître des oscillations, mais l'application d'un effet *upwind* apporté par SUPG suffit pour stabiliser ce transport. Ce résultat justifie donc la fonction de Level Set locale : sa définition, bien que non régulière, permet tout de même un transport stable.

4.4.3. Schéma semi implicite en temps

Ici, l'apport du schéma temporel semi implicite est flagrant. Le cercle est parfaitement transporté : il revient à sa position initiale, sans qu'aucune déformation vienne le changer. De plus, son volume de *3.14%* est très bien conservé puisqu'à la fin de la simulation (à $t = 4$), il occupe toujours la même portion de la cavité. Ces résultats sont donc bien meilleurs que les précédents, mais seul le schéma en temps constitue une différence. Cela est expliqué par l'ordre 2 de cette technique, par rapport à l'ordre 1 du schéma implicite.

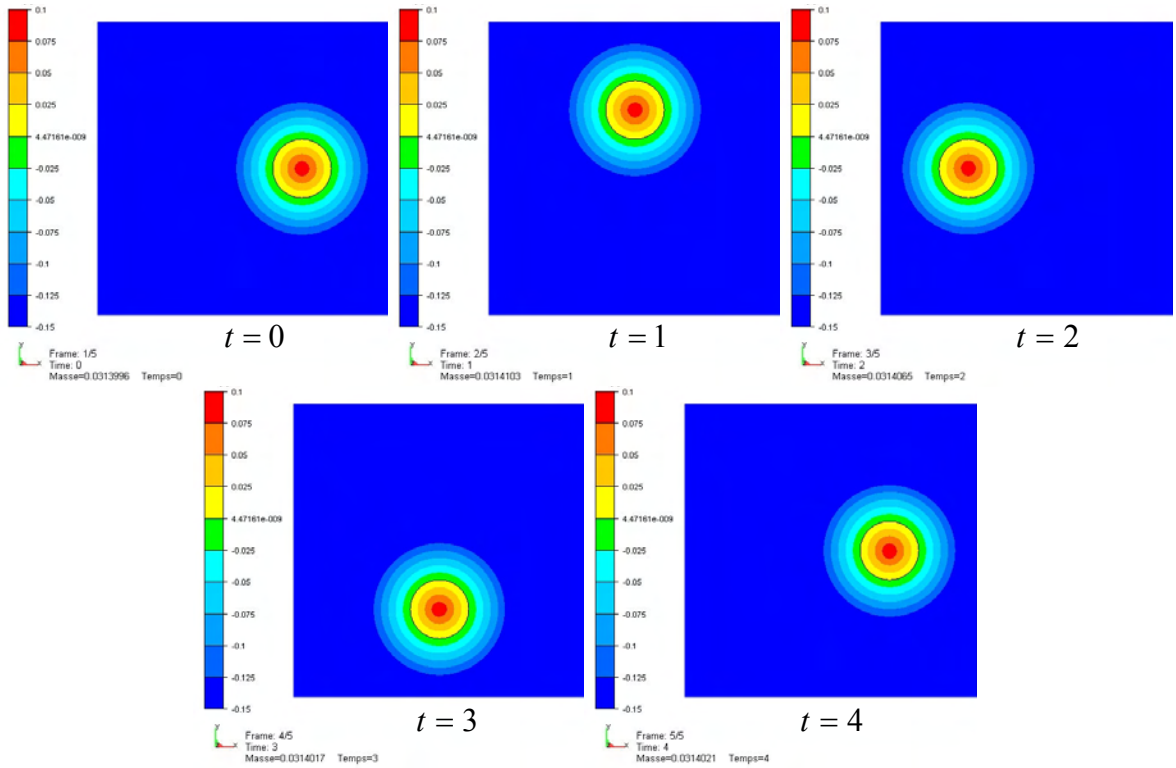


Figure 13 – Rotation d'un cercle avec un schéma semi implicite

Pour conclure, le schéma en temps semi implicite est adopté et sera utilisé par la suite, dans les diverses applications, car il montre de très bons résultats en comparaison avec le schéma implicite.

4.4.4. Temps de calcul

Dans cette simulation, 400 incréments de temps sont nécessaires pour que le cercle effectue un tour complet. A chaque incrément, un système linéaire non symétrique contenant 35036 inconnues (1 inconnue par nœud du maillage) est résolu. Après avoir préconditionné les matrices avec une méthode de type Jacobi par blocs, PETSc résous les systèmes avec GMRES (résidu minimal généralisé). A la fin de la simulation, le solveur a passé 339 secondes dans l'assemblage des matrices locales, et 86 secondes dans la résolution des systèmes linéaires. Le tableau suivant compare ces temps de calcul avec ceux obtenus dans le cas discontinu.

Solveur Convection	Continu	Discontinu
Nombre d'inconnues	35 036	138 756
Assemblage (s)	339	171
Résolution (s)	86	392
Total (s)	428	563

Tableau 1 – Temps de calcul pour les solveurs convection continu et discontinu

Bien que le nombre d'inconnues soit 4 fois inférieur dans le cas continu, le temps d'assemblage des matrices y est 2 fois plus long. Donc, les termes à assembler pour résoudre l'équation de transport de façon continu sont plus complexes, et notamment, la formulation SUPG demande plus d'opérations. Cependant, le temps de résolution des systèmes linéaires est 4.6 fois plus rapide en continu. Au final, la méthode continue est 1.3 fois plus rapide, mais

surtout, son nombre d'inconnues étant beaucoup moins grand, elle demande beaucoup moins de ressource de stockage.

4.4.5. Cas test du vortex

Un second test intéressant est celui du transport et de la déformation d'un cercle dans un écoulement cisaillé. Ce test va permettre d'étudier la capacité de notre méthode à décrire des structures filamentaires longues et très fines. Cette simulation a été proposée par [Rider, 1995]. Dans leur étude, les auteurs ont comparé quatre méthodes de suivi d'interface différentes : la méthode Level Set, la méthode *VOF*, et deux méthodes lagrangiennes. Leurs résultats mettent en évidence les défauts de la méthode Level Set. Elle semble peu satisfaisante pour la simulation de structures d'interface très fines. Nous allons voir dans cette partie que, suivant le schéma que l'on utilise pour la convection de l'interface, les résultats peuvent être améliorés, mais ils restent imparfaits.

Dans cette simulation, une cavité carrée en 2D est maillée avec un maillage à 16400 nœuds. Un champ de vitesse est fixé comme suit :

$$v = \begin{pmatrix} -2\sin^2(\pi x)\sin(\pi y)\cos(\pi y) \\ 2\sin(\pi x)\cos(\pi x)\sin^2(\pi y) \end{pmatrix} \quad \text{avec} \quad (x,y) \in [0,1]\times[0,1] \qquad (91)$$

A l'instant initial, l'interface est un cercle centré en (0.50, 0.75) de rayon 0.15 unités spatiales. Le champ de vitesse cisaillé et le disque à l'instant initial sont représentés sur la figure *(14)*.

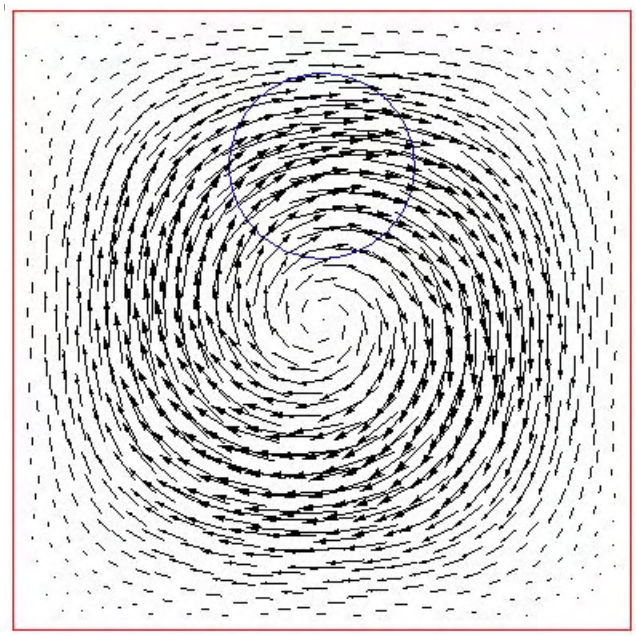


Figure 14 – Interface initiale et champ de vitesse

La fonction Level Set est initialisée comme la fonction distance au cercle à l'instant initial. Nous avons adopté la même démarche que précédemment pour tester les schémas de discrétisations spatiale et temporelle (on utilise toujours une stabilisation par SUPG et des schémas temporels de type d'Euler implicite et semi implicite). Une solution de référence

obtenue avec une méthode analytique de particules est présentée dans [Rider, 1995], et est représentée sur la figure suivante au temps de simulation *t=3* :

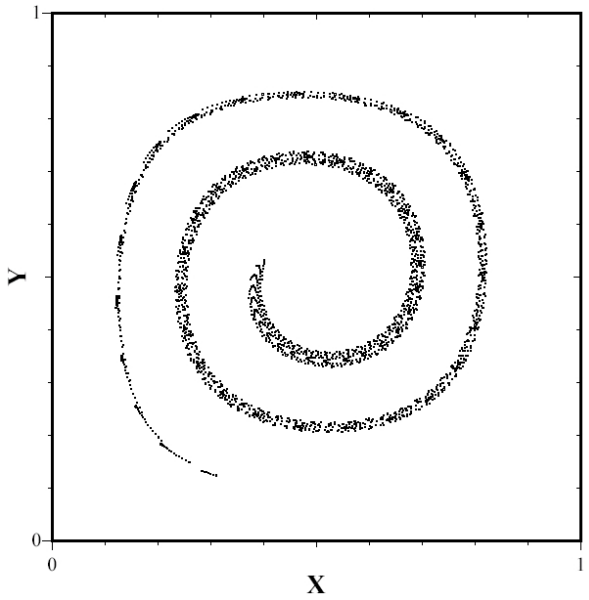


Figure 15 – Solution de référence

La figure *(16)* présente cette solution de référence, et les résultats obtenus au temps *t=3* unités temporelles pour les méthodes de Level Set avec schémas d'Euler implicite, et Level Set avec Euler semi implicite.

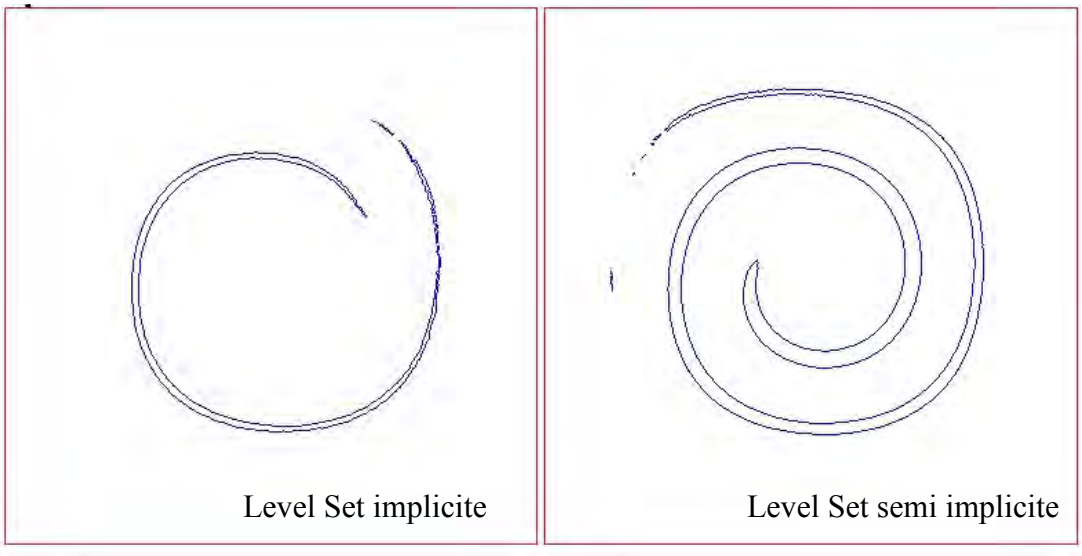


Figure 16 – Cas test du vortex : solutions obtenues à t=3

Les observations suivantes sont dressées :

- Solution de référence : On y observe que le cercle initial se transforme en un long serpentin qui s'affine au fur et à mesure du temps.
- Level Set implicite : De façon générale, la forme du serpentin est fortement altérée : il est beaucoup plus court que dans le solution de référence. De la même manière que dans le cas test précèdent, cette méthode montre de mauvais résultats sur le point de vue de la conservation de la masse. Une perte de 76 % de la masse initiale est observée, c'est-à-dire que plus du trois quart de la masse totale initiale a disparu.

➢ Level Set semi implicite : Encore une fois, cette méthode est bien meilleure que la précédente qui utilise un schémas implicite en temps. Toutefois, une perte de 5.5 % de la masse initiale est toujours observée. Cela montre les limites de la méthode Level Set qui a du mal à transporter exactement les quantités d'interface.

La solution obtenue sur le maillage à 16400 nœuds est plus ou moins dégradée par rapport à la solution exacte suivant le schéma utilisé. Lorsque l'épaisseur du filament devient faible, les erreurs numériques dues à la sous-résolution se caractérisent par des pertes de surface. Notre méthode Level Set semi implicite conduit cependant à une solution largement plus précise que les résultats exposés dans [Rider, 1995] et [Enright, 2002].

Nous allons maintenant vérifier la convergence spatiale de nos méthodes numériques, afin de s'assurer que la solution de notre calcul converge vers la solution exacte présentée dans [Rider, 1995]. Pour cela, nous utilisons un maillage plus fin à 65500 nœuds.

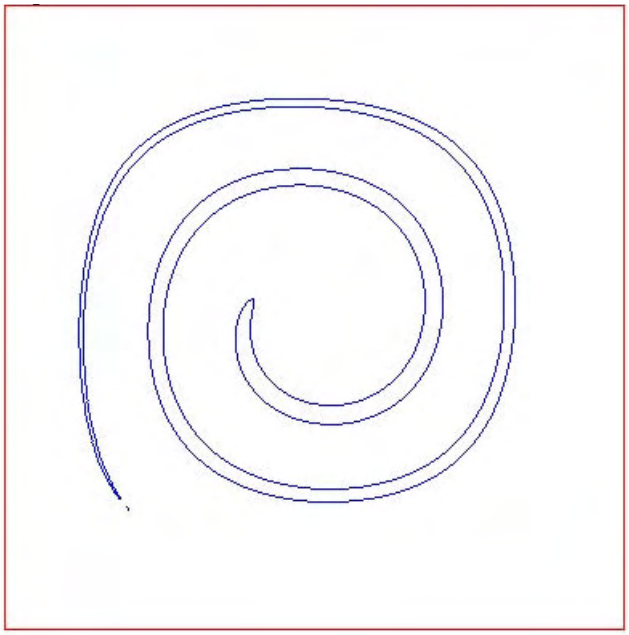


Figure 17 – Level Set semi implicite sur un maillage à 65500 nœuds

La solution obtenue sur la figure *(17)* avec un maillage fin est très proche de la solution de référence de la figure *(15)*.

Contrairement au cas test précèdent, dans le cas du serpentin le champ de vitesse est fortement cisaillé. Un défaut de la méthode apparaît lorsque l'on visualise plusieurs lignes de niveau, on observe sur la figure *(18)* que ces lignes vont être resserrées dans certains endroits et étirées dans d'autres. Ce phénomène risque de dégrader la précision avec laquelle le transport sera réalisé. Nous reviendrons par la suite sur ce comportement.

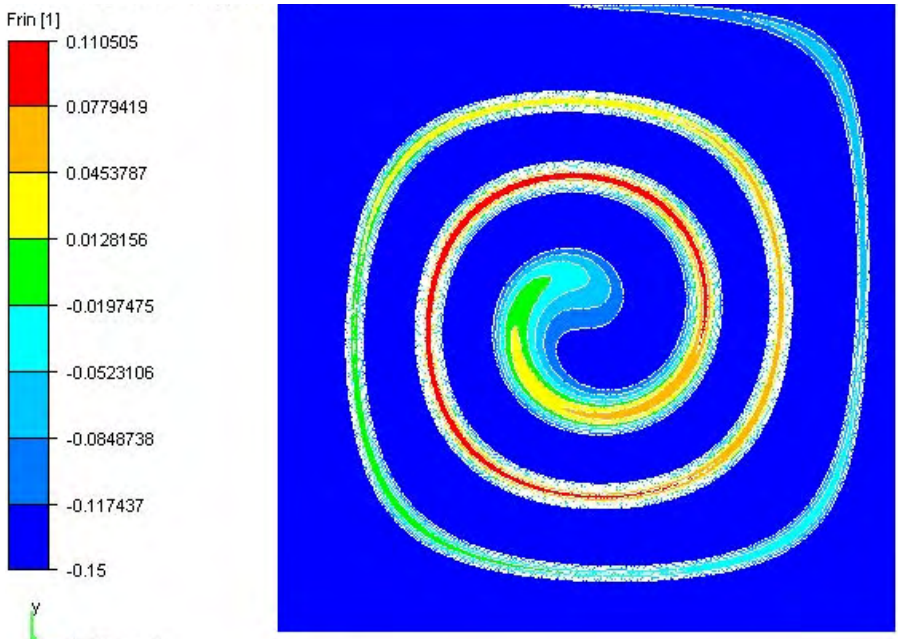


Figure 18 – Visualisation des lignes de niveau

4.5. Réinitialisation Level Set

4.5.1. Introduction

4.5.1.1. Les défauts de la méthode

Avec une méthode de type Level Set globale présentée précédemment, une fonction Level Set $\alpha(x,t)$ est initialisée comme une fonction distance signée à l'interface :

$$\alpha(x,0) = dist(x,\Gamma(0)) \qquad (92)$$

Ainsi, l'interface initiale $\Gamma(0)$ est représentée par le niveau zéro de $\alpha(x,0)$, c'est-à-dire l'ensemble des points qui correspondent à une valeur nulle. Puis, alors que cette fonction est transportée par l'équation de convection pure *(38)* vue précédemment, l'interface est capturée implicitement à l'instant *t* avec le niveau zéro de $\alpha(x,t)$:

$$\Gamma(t) = \{x, \alpha(x,t) = 0\} \qquad (93)$$

Cependant, le champ de vitesse solution d'un solveur d'écoulements incompressibles étant généralement fortement cisaillé dans les applications que l'on veut simuler, le transport de l'interface fait que $\alpha(x,t)$ peut se détériorer rapidement. Le cas test du vortex présenté précédemment met en évidence cette propriété obtenue par le cisaillement. Finalement, la fonction Level Set devient trop altérée pour que l'interface soit transportée efficacement, et ce, même avec une stabilisation de type SUPG. De fait, une telle fonction fait apparaître des gradients trop importants localement, proche de l'interface, et cause des oscillations résultantes d'une instabilité numérique grandissante.

4.5.1.2. Solutions apportées

La notion de distance algébrique à l'interface est essentielle. Il est donc numériquement souhaitable de garder $\alpha(x,t)$ proche d'une fonction distance signée, qui elle, avec un gradient d'ordre 1, se transporte de façon acceptable. Pour cela, la fonction Level Set doit être remise à jour lorsque celle-ci s'éloigne trop d'une fonction distance. Mais, est-ce vraiment légitime de vouloir changer ainsi la fonction de phase ? Ce problème est justifié de façon théorique dans [Evans, 1991] et [Chen, 1991], où il est prouvé que l'interface $\Gamma(t) = \{x, \alpha(x,t) = 0\}$ ne dépend pas du choix initial sur $\alpha(x,0)$, tant que son niveau zéro coïncide avec $\Gamma(0)$.

Le principe de la méthode Level Set est de transporter la ligne de niveau zéro de la fonction α avec la vitesse dictée par la physique du problème, puis de recalculer les autres lignes de niveau de manière à ce qu'elles respectent la propriété de distance algébrique signée à l'interface. Ce processus appelé « réinitialisation » consiste à remplacer $\alpha(x,t)$ par une autre fonction $\phi(x,\tau)$ qui a le même niveau zéro que $\alpha(x,t)$ mais qui est une fonction distance.

Il existe plusieurs méthodes de réinitialisation qui ont pour but de redonner une forme régulière à la fonction. Une façon naturelle serait de localiser explicitement l'interface avec une technique d'interpolation, puis de calculer la distance à cette interface en tout point du maillage [Merriman, 1994]. Cette approche brutale a l'avantage de ne pas changer l'interface au-delà de la précision du schéma d'interpolation ; mais elle a l'inconvénient d'être très coûteuse et de pouvoir introduire des irrégularités approximatives aux données, et notamment dans le cas de fortes courbures. Une procédure de lissage est donc souvent nécessaire pour améliorer cette approche. D'autre part, une solution plus élégante est présentée dans [Sussman, 1994], et c'est celle-là que nous choisissons d'étudier.

4.5.2. Algorithme de réinitialisation

Dans [Sussman, 1994], un algorithme de réinitialisation de la fonction Level Set α est mis au point. Son principe est de corriger de façon itérative la position des lignes de niveau par rapport à la ligne de niveau zéro, de manière à ce qu'elle respecte la propriété de distance algébrique. L'algorithme se présente sous la forme d'une équation aux dérivées partielles instationnaire que l'on doit résoudre jusqu'à ce que l'on aboutisse à un état stationnaire qui correspondra à une réinitialisation complète de la fonction α dans l'ensemble du domaine.

L'idée de base est de redonner à la fonction Level Set α ses deux propriétés les plus importantes :

- $|\nabla \alpha| = 1$ et
- $\alpha(x,t) = 0$ si et seulement si $x \in \Gamma(t)$

En d'autres termes, il s'agit de rendre une forme régulière à α tout en gardant son niveau zéro afin de garder l'interface inchangée.

Et ceci est réalisable en résolvant jusqu'à un état stationnaire une équation de type Hamilton-Jacobi initialisée avec la fonction Level Set α :

1. $\phi(x,0) = \alpha(x,t)$
2. Pour $\tau = 0...\varepsilon$, résoudre $\dfrac{\partial \phi}{\partial \tau} = S(\phi)(1 - |\nabla \phi|)$
3. $\alpha(x,t) = \phi(x,\varepsilon)$

La nomenclature utilisée dans le cadre de cet algorithme est la suivante :

- $\alpha(x,t)$ est la fonction Level Set qui est utilisée dans la loi de mélange du solveur d'écoulement ; aussi, elle est paramètre et solution du transport d'interface ; et enfin, elle sert d'initialisation à la phase de réinitialisation Level Set.
- $\phi(x,\tau)$ est la fonction paramètre et solution de la phase de réinitialisation.

Cette équation, qui ne traduit aucun principe physique, peut être utilisée, car d'un point de vue formel, elle ne modifie pas la position de la ligne de niveau zéro, qui, conformément à l'équation *(28)*, est la seule ligne de niveau dont on connaît la vitesse de propagation. Les autres lignes peuvent alors être convectées de manière indépendante, en particulier en imposant de respecter cette propriété de distance algébrique à l'interface.

Dans l'équation d'Hamilton-Jacobi :

$$\frac{\partial \phi}{\partial \tau} = S(\phi)(1 - |\nabla \phi|) \qquad (94)$$

τ est un temps fictif, et $S(\phi)$ est une fonction de signe définie par :

$$S(\phi) = \begin{cases} -1 & si\ \phi < 0 \\ 0 & si\ \phi = 0 \\ 1 & si\ \phi > 0 \end{cases} \qquad (95)$$

Néanmoins, cela défini une fonction non différentiable en $\phi = 0$, il est donc préférable de trouver une fonction de signe lissée et différentiable partout. Dans [Sussman, 1994], une approximation de $S(\phi)$ par :

$$S(\phi) = \frac{\phi}{\sqrt{\phi^2 + h^2}} \qquad (96)$$

est proposée avec de bons résultats. Cette méthode marche généralement très bien lorsque ϕ n'est ni trop plat, ni trop abrupt près de l'interface. Mais, dans le cas contraire, quelques inconvénients apparaissent. En effet, si le gradient de ϕ est trop faible (ϕ est trop plat), alors la vitesse de propagation de la fonction distance est plus petite, et donc, le nombre d'incréments nécessaires et le temps passé dans l'étape de réinitialisation sont plus grands. Par contre, si le gradient de ϕ est trop fort (ϕ est trop abrupt), le signe de ϕ peut alors changer, et donc, le niveau zéro se trouve changé pendant la réinitialisation. Pour remédier à ces problèmes, une autre définition est proposée dans [Peng, 1999] :

$$S(\phi) = \frac{\phi}{\sqrt{\phi^2 + |\nabla\phi|^2 h^2}} \qquad (97)$$

Par la suite, les figures *(22)* et *(23)* illustreront l'amélioration portée par cette fonction de signe.

La fonction distance signée que l'on recherche est la solution de l'équation *(94)* lorsque l'état stationnaire est atteint : dans ce cas, ϕ ne varie plus en fonction du temps fictif τ puisque $\partial\phi/\partial\tau = 0$ et donc $|\nabla\phi| = 1$. Aussi, le fait que $S(0) = 0$ force $\phi(x, \tau + \Delta\tau)$ à rester nul lorsque $\phi(x,\tau) = 0$, et cela montre bien que la solution garde le même niveau zéro que la fonction initiale.

Lorsque cette méthode est correctement implémentée, elle converge rapidement au voisinage de l'interface, ce qui représente un fort atout, comme nous le montrerons plus tard. La raison pour laquelle la fonction distance se reforme en premier lieu autour de l'interface est discutée dans le paragraphe suivant : ϕ se propage de part et d'autre de l'interface avec une vitesse de 1 dans une direction normale à l'interface et converge en un temps ε vers une fonction distance dans le voisinage ε de l'interface. Notons que la complexité de ce type de réinitialisation est proportionnelle au nombre de nœuds dans la zone de taille 2ε autour de l'interface ; ce qui correspond à un ordre *N* lorsque ε est petit par rapport à la taille totale de la cavité.

Propriété 1 : Si $\tau \in [0, \varepsilon]$, alors $|\nabla\phi(x,\varepsilon)| = 1$ dans la zone définie par $dist(x,\Gamma) < \varepsilon$.

4.5.3. Résolution des équations d'Hamilton-Jacobi

$$\begin{cases} \dfrac{\partial\phi}{\partial\tau} = S(\phi)(1 - |\nabla\phi|) \\ \phi(x,0) = \alpha(x,t) \end{cases} \qquad (98)$$

Pour résoudre cette équation de Hamilton-Jacobi, nous appliquons une méthode par éléments finis similaire à celle utilisée pour une équation de convection pure, car ces deux types d'équations sont semblables. En effet, avec un schémas temporel implicite du type Euler :

$$\frac{\partial\phi}{\partial\tau} = \frac{\phi - \phi^-}{\Delta\tau} \qquad (99)$$

où $\Delta\tau$ est un pas de temps fictif, et ϕ^- est la valeur connue pour ϕ à l'incrément de temps fictif précédent $(\tau - \Delta\tau)$. Nous pouvons ainsi remplacer $|\nabla\phi|$ par :

$$|\nabla\phi| = \frac{|\nabla\phi|^2}{|\nabla\phi|} = \frac{\nabla\phi^-.\nabla\phi}{|\nabla\phi^-|} \qquad (100)$$

Et nous obtenons :

$$\frac{\phi}{\Delta\tau} + S(\phi^-)\frac{\nabla\phi^-}{|\nabla\phi^-|}\nabla\phi = \frac{\phi^-}{\Delta\tau} + S(\phi^-) \qquad (101)$$

Etablir un parallèle avec une équation de convection pure est évident : cela équivaut à avoir une vitesse de convection (ou de propagation) w telle que :

$$w = S(\phi^-)\frac{\nabla\phi^-}{|\nabla\phi^-|} \qquad (102)$$

et un second membre F supplémentaire :

$$F = S(\phi^-) \qquad (103)$$

A noter que la vitesse de convection w est en fait un vecteur unitaire normal à l'interface qui pointe vers l'extérieure. Ceci donne une direction normale à l'interface et une vitesse de norme un à la propagation de la fonction distance pendant la réinitialisation. C'est-à-dire que le processus de réinitialisation se propage des points les plus proches de l'interface vers les points les plus éloignés. Et c'est pourquoi, la solution converge vers une fonction distance dans le voisinage ε de l'interface, lorsqu'on prend $\tau \in [0, \varepsilon]$.

La formulation forte du problème à résoudre est :

$$\frac{\phi}{\Delta\tau} + w.\nabla\phi = \frac{\phi^-}{\Delta\tau} + F \qquad (104)$$

Sous cette nouvelle forme, on reconnaît aisément une équation hyperbolique qui ressemble beaucoup à l'équation de transport *(38)*. Des méthodes spécifiques similaires à celles présentées pour résoudre cette équation de transport peuvent donc être adaptées à l'équation linéaire hyperbolique *(104)*. Et notamment, une technique de type SUPG est utilisée pour stabiliser la méthode numérique par éléments finis. Les relations *(102)* et *(103)* nous permettent de connaître le sens de propagation des courbes caractéristiques, on peut ainsi appliquer une procédure « upwind » en se référant au signe des composantes de vitesse du vecteur w, comme on l'a déjà fait pour la résolution de l'équation *(38)*.

Les espaces fonctionnels suivants sont utilisés pour écrire la formulation faible correspondante :

$$L^2(\Omega) = \left\{ q, \int_\Omega q^2 dV < \infty \right\}$$
$$H^1(\Omega) = \left\{ q \in L^2(\Omega), \nabla q \in L^2(\Omega) \right\}$$
$$V = \left\{ q \in H^1(\Omega), q = g \text{ sur } \partial\Omega \right\}$$
$$V_0 = \left\{ q \in H^1(\Omega), q = 0 \text{ sur } \partial\Omega \right\}$$

Dans cette formulation variationnelle, il suffit de trouver $\phi \in V$ tel que :

$$\int_{\Omega} \frac{\phi}{\Delta\tau}.\phi^* + \int_{\Omega} \nabla\phi.w.\phi^* = \int_{\Omega} \frac{\phi^-}{\Delta\tau}.\phi^* + \int_{\Omega} F.\phi^* \qquad \forall \phi^* \in V_0 \tag{105}$$

Ensuite, on construit un espace fonctionnel de dimension finie V_h tel que $\lim_{h\to 0} V_h = V$. Le domaine Ω est décomposé en tétraèdres *k* disjoints (triangles en 2D) qui vérifient $\Omega = \bigcup_K k$ pour qu'une méthode d'éléments finis puisse être appliquée. Pour cela, une interpolation continue (P1) est choisie. A l'aide de cette discrétisation, ϕ peut être approchée par la solution ϕ_h de ce problème :

$$\left(\frac{\phi_h}{\Delta\tau}, \phi_h^*\right)_{\Omega} + \left(\nabla\phi_h.w, \phi_h^*\right)_{\Omega} = \left(\frac{\phi_h^-}{\Delta\tau}, \phi_h^*\right)_{\Omega} + \left(F, \phi_h^*\right)_{\Omega} \tag{106}$$

Ainsi, en substituant *v* par *w* et en ajoutant *F* au second membre dans notre solveur de convection continue *(38)*, cette équation est résolue de la même façon que le transport ; notamment avec une stabilisation SUPG qui résout les problèmes du même ordre que rencontrés dans le paragraphe qui traite de l'équation de transport.

4.5.4. Étude de la réinitialisation

Dans cette partie, la phase de réinitialisation Level Set par la résolution de l'équation de Hamilton-Jacobi est étudiée. Le temps virtuel introduit par cette méthode est analysé, ainsi que l'influence de la taille de maille, et du pas de temps virtuel. Enfin, des conclusions sont tirées sur l'utilisation de cette méthode et de ses paramètres numériques.

4.5.4.1. Temps fictif

Dans une cavité rectangulaire de largeur 1 et de longueur 4 centimètres, une fonction ϕ trois fois plus « pentue » qu'une fonction distance ($|\nabla\phi| = 3$) est réinitialisée par la méthode discutée précédemment. Pour résoudre l'équation de Hamilton-Jacobi *(98)*, le pas de temps fictif $\Delta\tau$ est fixé à 0.1. Ce cas test est exécuté deux fois : la première fois avec un maillage grossier (1500 nœuds et une taille de maille de 0.06 cm) ; et la deuxième fois avec un maillage quatre fois plus fin (24000 nœuds et une taille de maille de 0.015 cm). Initialement, le niveau zéro (représentée sur la figure en noir) est sur $x = 0$, et les 10 lignes de niveau sont assez resserrées sur l'intervalle $\phi(x,0) \in [0,4]$.

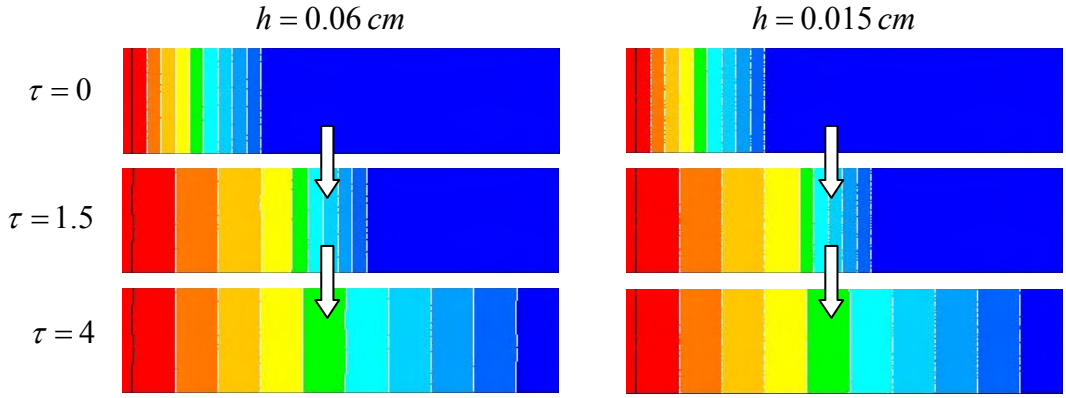


Figure 19 – Vitesse de propagation de ϕ en fonction de la taille de maille

Après 15 incréments en temps fictif, $\tau = 1.5$ et la solution a trouvé la stabilité avec $|\nabla\phi(x,1.5)| = 1$ pour $dist(x,\Gamma) \le 1.5\,cm$; puis, lorsque $\tau = 4$ et $\phi(x,\tau)$ est devenue une fonction distance avec $|\nabla\phi(x,4)| = 1$ pour $dist(x,\Gamma) \le 4\,cm$, c'est-à-dire sur tout le domaine. Cela vérifie bien la propriété *(1)* : $\phi(x,\tau)$ est réinitialisé d'abord autour de le niveau zéro, et même, un état stationnaire est atteint dans une zone de taille ε de part et d'autre du niveau zéro lorsque $\tau = \varepsilon$. Aussi, les résultats sont identiques pour les deux maillages utilisés, bien que ceux-ci aient des tailles très différentes (avec un facteur 4) ; ce qui indique que cette propriété est indépendante du maillage.

Maintenant, la dépendance au pas de temps virtuel est étudié. Pour cela, le même test est appliqué avec le même maillage (h=0.06), mais avec des pas de temps virtuel différents : 0.03 et 0.12.

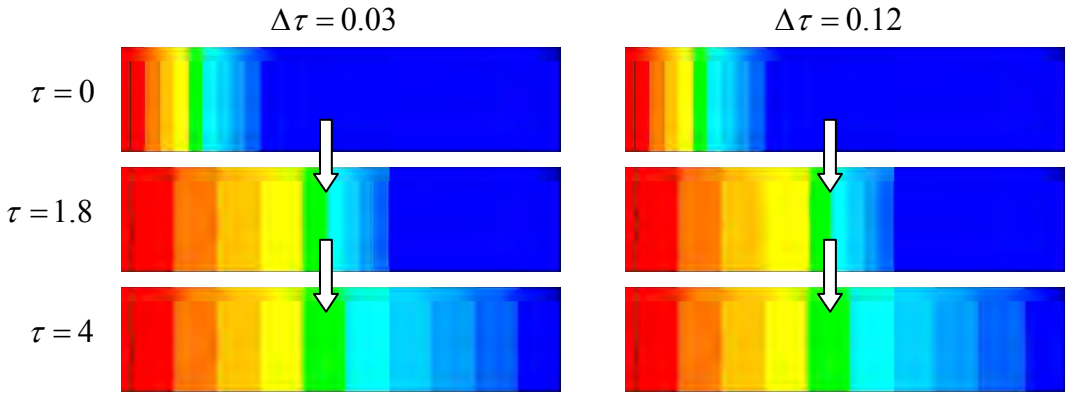


Figure 20 – Vitesse de propagation de ϕ en fonction du pas de temps fictif

Pour $\Delta\tau = 0.03$, la stabilité est atteinte dans toute la cavité (jusqu'à $x = 4\,cm$) après 134 incréments en temps virtuel ($\tau = 4$) ; tandis que pour $\Delta\tau = 0.12$, il n'a fallu que 34 incréments (mais toujours $\tau = 4$).

Pour conclure, ces cas test illustrent la propriété *(1)*, et montrent qu'elle est vraie indépendamment du pas de temps virtuel $\Delta\tau$ et la taille de maille h. En revanche, $\Delta\tau$ doit vérifier certaines conditions pour que la stabilité du système soit maintenue, mais il reste difficile à définir précisément. Ce sujet est développé dans [Peng 1999] où il est proposé la propriété suivante :

$$\Delta\tau \le h \qquad (107)$$

4.5.4.2. Vitesse de propagation

Afin de bien comprendre comment la propagation se réalise, la vitesse de propagation est analysée :

$$w = S(\phi)\frac{\nabla\phi}{|\nabla\phi|} \qquad (108)$$

Le cas test présenté ici est introduit dans [Peng, 1999] et constitue un support destiné à illustrer l'étude de *w*. Dans une cavité de dimensions 2x2 cm, une fonction est initialisée à $\phi(x,y) = x^2/0.3^2 + y^2/0.2^2 - 1$. Le pas de temps virtuel est fixé à $\Delta\tau = h/2 = 0.015/2$ sur un maillage non structuré à 22 000 nœuds et 44 000 éléments. Puis, l'équation de Hamilton-Jacobi *(98)* est résolue avec $\tau \in [0,1]$. La figure *(21)* montre le champ de la vitesse de propagation *w* qui va déterminer la direction et la vitesse de réinitialisation de la fonction ϕ tout au long des incréments en temps fictif.

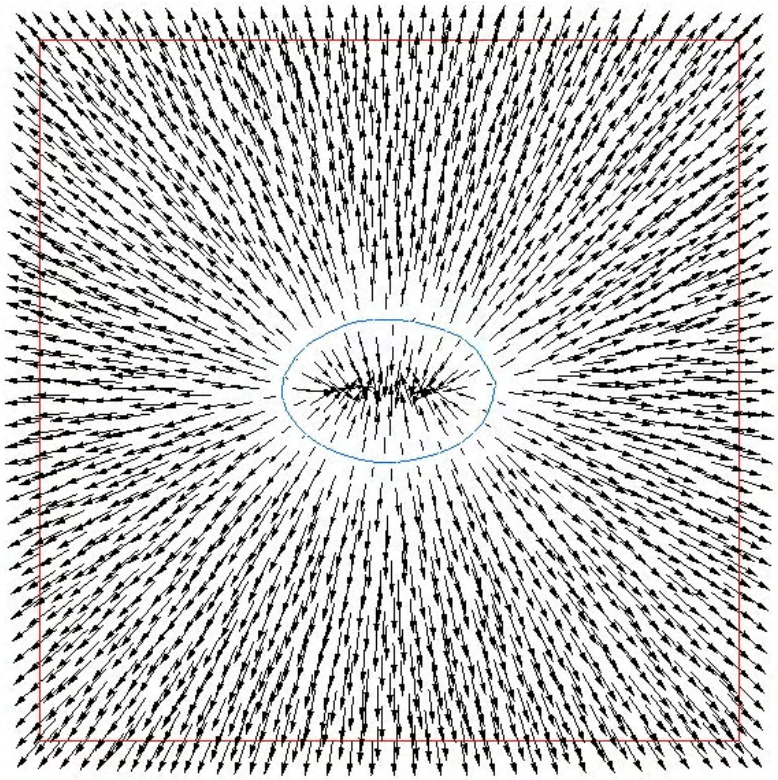


Figure 21 – Vitesse de propagation w

Cette vitesse de propagation *w* est normale au niveau zéro (l'ellipse bleue sur la figure) de la fonction ϕ. Sa norme est généralement très proche de 1, mais elle diminue dans le voisinage du niveau zéro. Voici comment peuvent être expliquées les principales caractéristiques de ce champ vectoriel :

- $\nabla\phi$ est un vecteur normal au niveau zéro de ϕ.
- $\dfrac{\nabla\phi}{|\nabla\phi|}$ est un vecteur unitaire normal au niveau zéro de ϕ ($\hat{n}$ dans la propriété *31*)

- $S(\phi)\dfrac{\nabla\phi}{|\nabla\phi|}$ est un vecteur unitaire normal qui pointe dans la direction opposée au niveau zéro de ϕ.

La définition *(97)* qui offre une forme plus régulière à $S(\phi)$ a pour conséquence de diminuer la norme de *w* dans un voisinage proche du niveau zéro (zone de taille h de part et d'autre de Γ) ; et prend même une valeur nulle sur les points qui vérifient exactement $\phi(x,\tau)=0$.

4.5.4.2. Approximation et étude de la fonction de signe

Sur les figures suivantes, la fonction ϕ initialement très raide, est représentée par 10 lignes de niveau sur l'intervalle $\phi\in[-0.15,0.15]$. Elles sont initialement très resserrées autour du niveau zéro (en noir) ce qui correspond à de forts gradients. Puis, après 5, 10 et 20 pas de temps fictifs, ces lignes de niveau se desserrent petit à petit et se fixent à intervalles réguliers en partant du niveau zéro. La fonction ϕ retrouve ainsi un gradient proche de 1 qui caractérise une fonction distance. Sur la figure *(22)*, la définition *(96)* est utilisée pour approcher la fonction signe, tandis que c'est la définition améliorée *(97)* qui est utilisée sur la figure *(23)*. On remarque que dans le premier cas, le niveau zéro représenté en noir s'altère de façon assez prononcée : la réinitialisation provoque un changement non négligeable du niveau zéro. Par contre, dans le deuxième cas, elle reste parfaitement identique à son état initial après 5, 10 et 20 pas de temps fictifs.

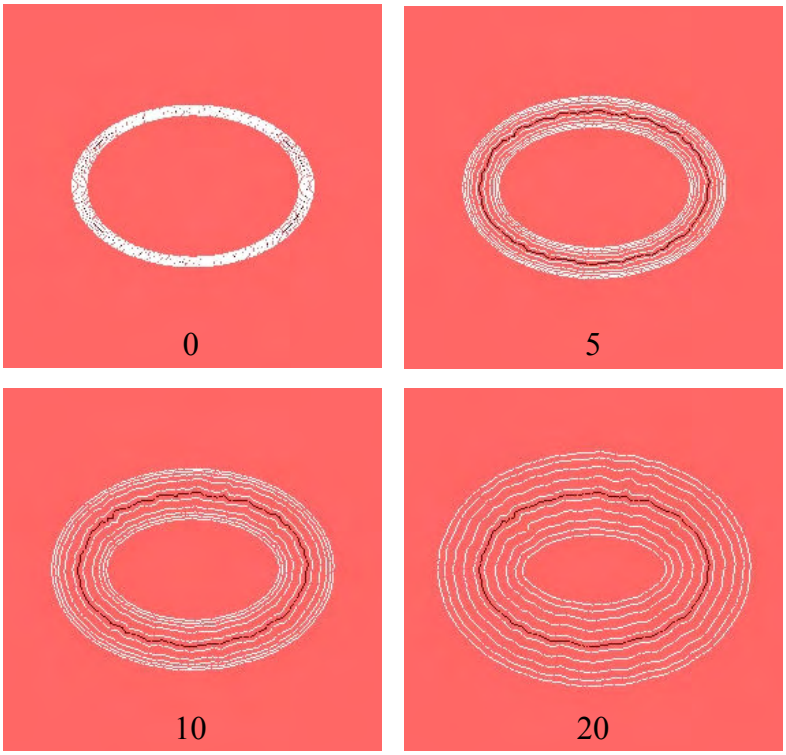


Figure 22 – $S(\phi)=\phi/\sqrt{\phi^2+h^2}$

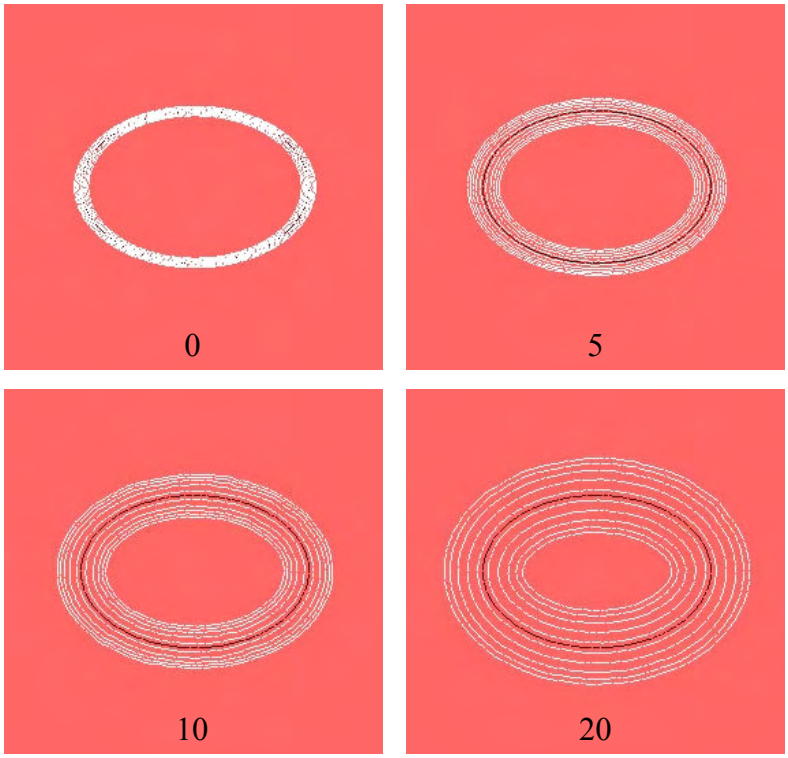


Figure 23 – $S(\phi) = \phi \Big/ \sqrt{\phi^2 + |\nabla \phi|^2 h^2}$

Le choix de l'approximation de $S(\phi)$ par la formule *(97)* résout le problème de changement signe de ϕ (qui déplace le niveau zéro à travers la parois des éléments du maillage) pendant l'étape de réinitialisation dans le cas où ϕ a un gradient trop élevé ; mais aussi, accélère la convergence dans le cas où ϕ a un gradient trop faible.

4.5.5. Validation du transport d'interface avec réinitialisation

On a vu précédemment que lorsque la fonction α était convectée par un champ de vitesse fortement cisaillé, les lignes de niveau pouvaient devenir très resserrées ou très étirées, ce qui a motivé la mise au point d'un algorithme de réinitialisation de la fonction distance. Dans le cas du vortex, le champ de vitesse est fortement cisaillé, c'est donc le cas idéal pour vérifier l'efficacité de l'algorithme de réinitialisation sur le transport d'une interface. Dans ce cas test, une réinitialisation est réalisé à chaque pas de temps lors du transport.

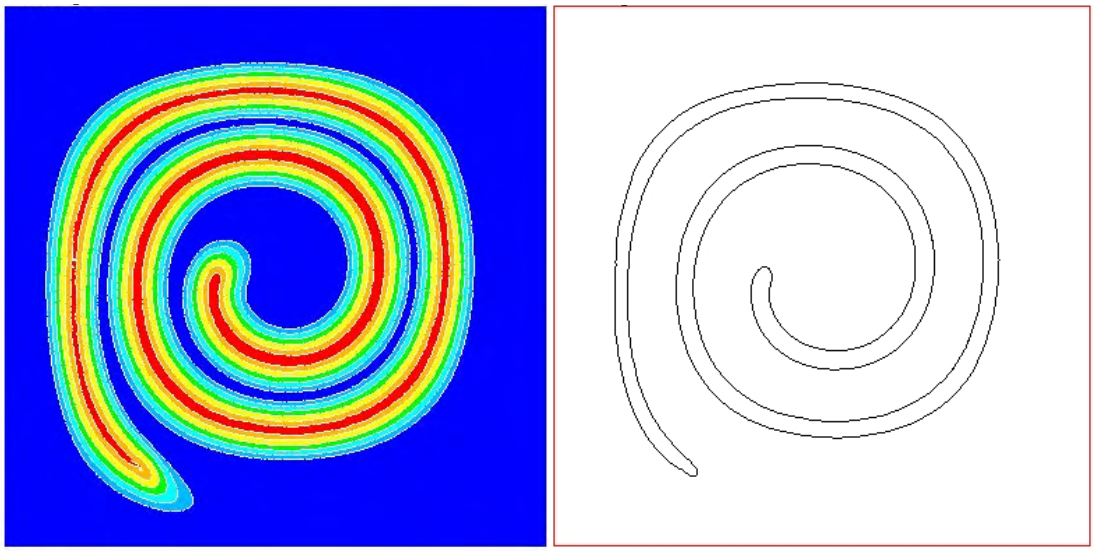


Figure 24 – Solution finale avec réinitialisation : lignes de niveau & niveau zéro

Sur la figure *(24)* sont tracés 6 lignes de niveau au temps *t=3*. On y observe que l'espacement entre les lignes de niveau reste régulier jusqu'à la fin de la simulation (l'intervalle est constant le long des lignes de niveau). L'algorithme de réinitialisation permet donc de prévenir l'étalement et le resserrement des lignes de niveau.
Si l'on s'intéresse maintenant aux propriétés de résolution de la méthode, on observe un résultat intéressant : la longueur du panache du serpentin correspond à peu près à la solution exacte *(15)*, en ce sens l'algorithme de réinitialisation améliore les propriétés de résolution de notre méthode de suivi d'interface. En revanche l'épaisseur du panache est clairement surestimée par rapport à la solution exacte. On a donc un problème de conservativité. Sur un maillage à 16400 nœuds, le gain de volume est estimé à 50%.

Il ressort de cette étude que l'algorithme est performant pour recalculer une fonction distance, mais qu'il n'aide pas à la conservation de la quantité de surface. Ainsi, bien que la fonction Level Set soit résolue dans des conditions de discrétisation optimales, il apparaît que des erreurs numériques assez importantes peuvent altérer la qualité de la solution. Dans le cas test du vortex, des gains de surface assez importants surviennent. Cependant, on observe une meilleure prédiction de la longueur du filament, la méthode devient alors intéressante pour décrire des structures fines, même si l'on sait qu'un épaississement artificiel peut survenir.
Le problème de non-conservation de la quantité de surface reste donc entier. Les « gains de surface » calculés sur ce cas sont importants, mais il faut modérer ce résultat car le filament décrit est très long, donc un faible épaississement sur une longueur importante peut conduire à des gains de surface assez importants. Dans les simulations d'écoulements à deux phases que nous effectuerons, il est peu probable que l'on rencontre des structures filamentaires aussi longues.

4.6. Méthode Level Set locale

En regardant la loi de mélange *(33)* qui reste très locale autour de l'interface, c'est-à-dire dans une épaisseur d'un élément traversé par l'interface, il est suffisant de s'intéresser à une fonction distance très proche de l'interface. C'est ainsi que la fonction de phase α peut être initialisée comme une fonction distance uniquement dans une certaine zone de taille $2\varepsilon_0$

autour de l'interface, contrairement à la formule *(26)* qui définie une fonction distance sur tout le domaine de calcul. En dehors de cette zone appelée *zone Level Set*, la fonction peut rester constante ($|\alpha| = \varepsilon_0$) :

$$\alpha(x,0) = \begin{cases} -\varepsilon_0 & si\ \ dist(x,\Gamma) < -\varepsilon_0 \\ dist(x,\Gamma) & si\ \ -\varepsilon \le dist(x,\Gamma) \le \varepsilon_0 \\ \varepsilon_0 & si\ \ dist(x,\Gamma) > \varepsilon_0 \end{cases} \qquad (109)$$

Avec une telle initialisation, certaines propriétés de la méthode de réinitialisation avec l'équation de Hamilton-Jacobi *(98)* peuvent être misent à profit. Notamment, l'étude portée sur cette méthode montre qu'au cours d'une phase de réinitialisation effectuée sur un temps fictif $\tau \in [0, \varepsilon]$, α retrouve les propriétés d'une fonction distance dans le voisinage ε de part et d'autre de l'interface. Autrement dit, α est réinitialisé à $|\nabla \alpha| = 1$ près de l'interface en premier. Par conséquent, le fait d'avoir une fonction telle que dans *(109)*, évite d'avoir à réinitialiser α sur tout le domaine : si la zone Level Set a initialement une taille de $2\varepsilon_0$, alors la méthode de réinitialisation peut être exécutée sur un petit intervalle de temps fictif $\tau \in [0, \varepsilon_0]$ afin de trouver un « état stationnaire local » :

$$|\nabla \phi| = 1 \quad \text{pour} \quad |\phi| \le \varepsilon_0 \qquad (110)$$

Ainsi, beaucoup moins d'incréments en temps fictif sont nécessaires, et le temps de calcul consacré à la réinitialisation Level Set est fortement diminué. Ailleurs dans la cavité, c'est-à-dire en dehors de la zone Level Set $\alpha \in [-\varepsilon_0, \varepsilon_0]$, nous nous préoccupons uniquement de la fonction α pour son signe : elle doit seulement rester assez stable, pour qu'aucune instabilité numérique ne vienne perturber le modèle et faire changer le signe de la fonction, et donc altérer l'interface.

Un autre avantage de cette méthode Level Set locale est qu'il est maintenant très facile de déterminer le nombre d'incréments en temps fictif nécessaires pour assurer la propriété de fonction distance dans un voisinage de l'interface prédéfini. Par exemple, si le pas de temps fictif est $\Delta\tau$, et $2\varepsilon_0$ est la taille initiale de la zone Level Set, alors α doit être réinitialisé avec $\varepsilon_0 / \Delta\tau$ incréments en temps fictif.

Les endroits non réguliers d'une fonction Level Set locale pourraient poser des problèmes, car nous avons vu que le Galerkin standard prend mal en compte de telles caractéristiques. Seulement, nous verrons par la suite que la stabilisation par SUPG permet de contrebalancer ce problème avec succès, dès lors que certaines conditions sont vérifiées (notamment en fonction de la taille de zone Level Set, la taille de maille, le pas de temps, et la vitesse de convection). C'est cette réflexion qui introduit le concept de critères automatiques de réinitialisation basés sur ces conditions de stabilité.

4.7. Critères automatiques de réinitialisation

Après avoir évalué le besoin de réinitialiser une fonction distance (localement autour de l'interface est suffisant) et étudié un moyen efficace d'y parvenir, il faut maintenant trouver un critère de déclanchement automatique qui vise à détecter – plus ou moins précisément – le moment au cours de la simulation où une réinitialisation est nécessaire pour que le transport de la fonction α se fasse correctement. En effet, il n'est pas nécessaire de réinitialiser α à tous les incréments de temps, car cela serait à la fois trop lourd et superflue : une fonction distance peut être transportée avec de bons résultats sur plusieurs incréments sans qu'elle ne se détériore trop ni qu'elle fasse apparaître des oscillations.

Une première méthode serait de fixer arbitrairement une fréquence de réinitialisation, mais, lors des simulations numériques qui peuvent être longues et complexes, il n'est pas optimal de fixer une telle fréquence étant donné que des écoulements de fluides ne sont que rarement constants au cours du temps. Une deuxième méthode pourrait être entièrement manuelle, c'est-à-dire que le choix de réinitialiser pourrait être donné par l'utilisateur lorsqu'il observe que la fonction transportée est trop altérée ; mais ceci serait bien sûr trop lourd compte tenu des contraintes évidentes que cela comporte (gérer de multiples reprises de calculs, nécessité d'un suivi continu tout au long de la simulation, … etc.). Enfin, une troisième technique serait d'étudier un certain nombre de paramètres afin de détecter automatiquement, et le plus précisément possible, les moments où une réinitialisation est nécessaire. Nous cherchons donc ici des critères potentiels pour ce genre de problème.

- Rappelons-nous que α est initialisé suivant l'équation *(109)*, et donc que la propriété suivante est vraie au début de la simulation :

$$\alpha(x,0) \in [-\varepsilon_0, \varepsilon_0] \qquad (111)$$

sur tout le domaine de calcul. Le premier critère de réinitialisation que nous pouvons imaginer est plutôt simple. En partant de l'idée que l'on réinitialise la fonction α dans le but de limiter les oscillations de celle-ci, nous pouvons dire que le critère dépend des valeurs maximales et minimales de α. En effet, l'apparition des oscillations va, entre autres, avoir pour conséquence de faire passer α au dessus de ε_0 et au dessous de $-\varepsilon_0$. Ainsi, lorsque l'équation de transport n'est pas assez stable, la propriété suivante n'est plus satisfaite :

$$|\alpha| \leq \varepsilon_0 \qquad (112)$$

Aussi, plus il y aura d'oscillations, plus les valeurs de $|\alpha|$ vont aller au-delà de ε_0. De ce fait, nous pouvons dire que α a besoin d'être réinitialisée dès le moment où des valeurs dépassent les bornes ε_0 et $-\varepsilon_0$ d'une manière trop importante. En pratique, nous observons que le transport ne s'effectue correctement uniquement lorsque :

$$|\alpha| \leq \frac{125}{100}\varepsilon_0 \qquad (113)$$

Autrement dit, dès que α atteint localement une valeur supérieure à 125% de ε_0 (ou inférieure à 125% de $-\varepsilon_0$), la fonction ne permet plus d'être transportée de façon efficace, et

donc, doit subir un rajeunissement pour lui donner une forme plus régulière dans la zone Level Set. Cette réinitialisation recadre aussi α dans ses bornes initiales : entre $-\varepsilon_0$ et ε_0.

- Un deuxième critère de réinitialisation peut ensuite être implémenté, mais cette fois-ci, en prenant en compte le gradient de α. La raison principale vient du fait que pendant le transport d'une fonction régulière, de forts gradients peuvent apparaître à proximité de l'interface, ce qui rend difficile la résolution de l'équation de convection pure. α étant initialisée comme le montre l'équation *(109)*, elle a un gradient égal à un dans la zone Level Set de taille $2\varepsilon_0$ autour de l'interface. Il est donc évident que le fait d'augmenter le gradient de α revient à donner une pente plus raide à la fonction, et ainsi, à diminuer la taille de la zone Level Set initialement fixée à $2\varepsilon_0$. Les deux figures suivantes illustrent cette affirmation : lorsque $|\nabla\alpha| = 1$, la zone Level Set est de taille $2\varepsilon_0$; mais lorsque $|\nabla\alpha| = 2$, la zone Level Set est de taille ε_0, et donc deux fois plus petite.

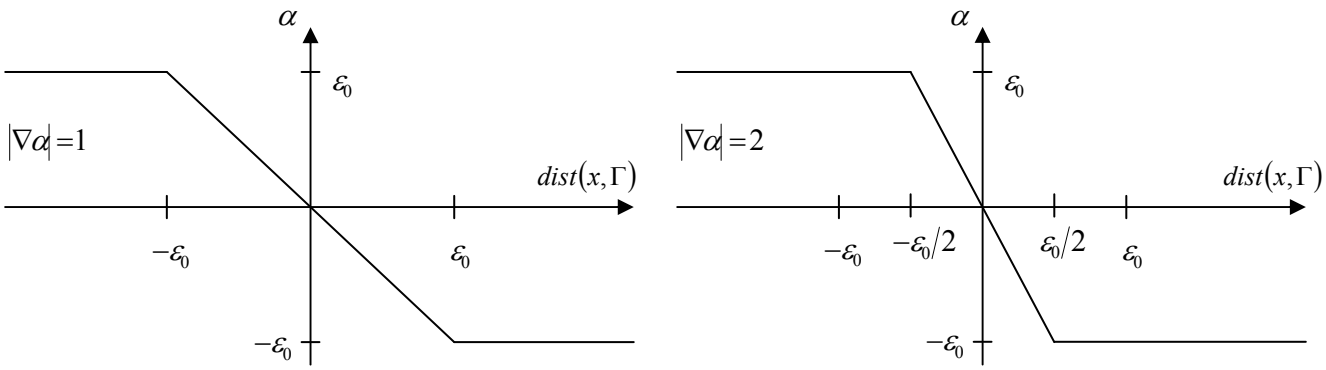


Figure 25 – Graphe de fonctions avec zones Level Set de $2\varepsilon_0$ et de ε_0

A partir de là, nous établissons une relation entre la taille ε de la zone Level Set et le gradient de α à un moment *t* donné :

$$\varepsilon(t) = \frac{\varepsilon_0}{|\nabla\alpha(x,t)|} \qquad (114)$$

Nous cherchons ici à trouver une valeur de $|\nabla\alpha|$ critique qui servira comme critère du déclanchement d'une réinitialisation. A partir de l'équation *(114)* précédente, nous pouvons donc dire que le gradient de α doit rester inférieur au rapport de la taille de la zone Level Set initiale ε_0 sur une taille minimale $\varepsilon_{\min}$ en dessous de laquelle la taille de la zone Level Set ne peut pas aller :

$$|\nabla\alpha| \le \frac{\varepsilon_0}{\varepsilon_{\min}} \qquad (115)$$

Ceci peut être un critère que l'on recherche : dès que la fonction α ne satisfait plus cette propriété, une réinitialisation est nécessaire. Mais il reste maintenant à déterminer la plus petite taille que la zone Level Set puisse avoir pour que le transport de α se passe bien, et donc que l'équation de convection pure reste stable, sans causer d'oscillations dans la solution.

Une hypothèse que l'on peut formuler sur ce problème est que le déplacement de la fonction α sur un incrément de temps ne doit pas dépasser la taille de la zone Level Set. A noter que le déplacement subit par α est :

$$\Delta\alpha = |v|\Delta t \qquad (116)$$

et donc

$$\varepsilon_{\min} = |v|\Delta t \qquad (117)$$

Afin de valider cette hypothèse, voici un exemple en deux dimensions dans lequel plusieurs fonctions α avec des zones Level Set de différentes tailles sont transportées suivant une vitesse constante :

$$v = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad (118)$$

Les images de la figure *(26)* montrent les fonctions à $t = 0$ et $t = 0.25s$, c'est-à-dire après seulement un incrément en temps où $\Delta t = 0.25s$. Ici, α est initialisé suivant la loi *(109)* : $\alpha = \varepsilon_0$ dans les zones rouges, $\alpha = -\varepsilon_0$ dans les zones bleues, et $|\nabla\alpha| = 1$ dans $[-\varepsilon_0, \varepsilon_0]$. Le niveau zéro des fonctions se trouve dans la zone verte, initialement au centre de la zone Level Set. Afin d'apprécier l'influence du maillage dans cette démonstration, les premiers tests ont été effectués avec un maillage à 1850 nœuds et une taille de maille de 0.06 cm ; puis, les seconds tests ont été effectués avec un maillage à 7200 nœuds et une taille de maille de 0.03 cm.

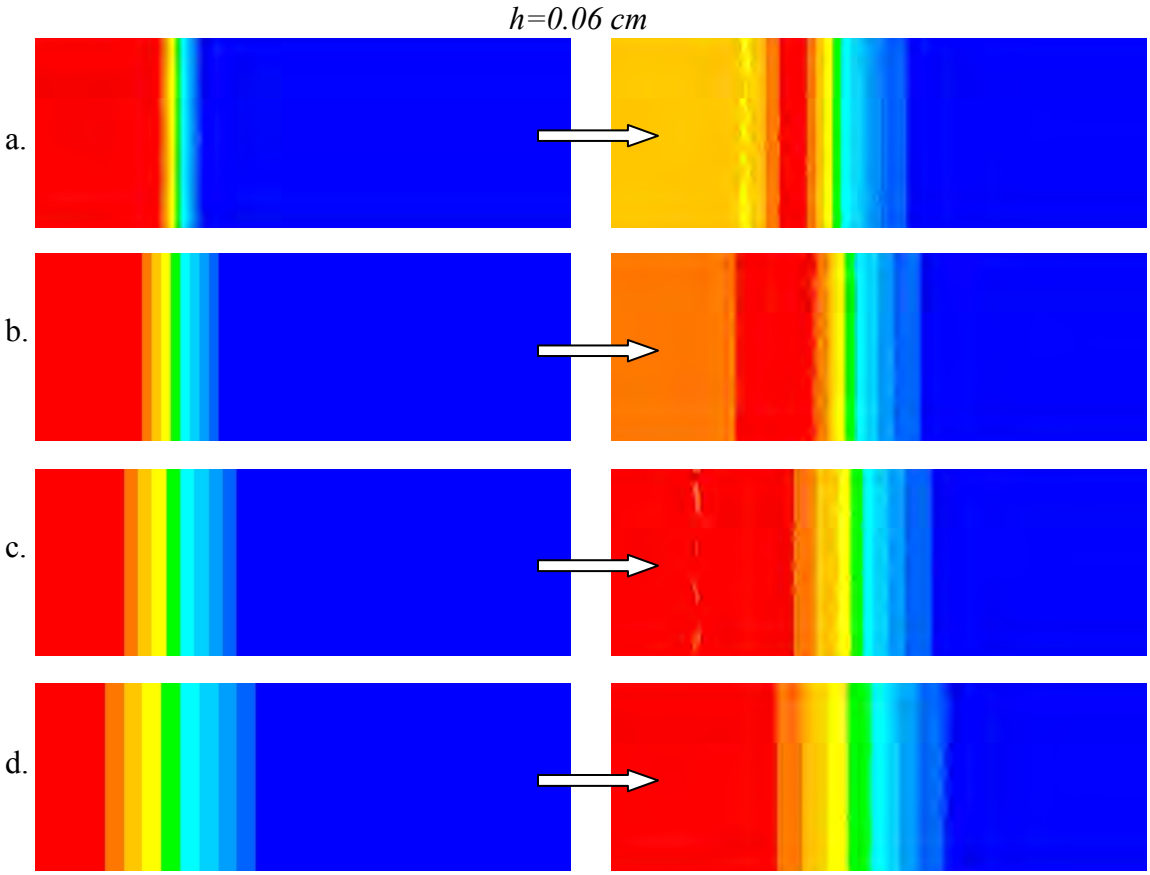


Figure 26 – Transport d'une fonction avec différentes tailles de zone Level Set – h=0.06cm

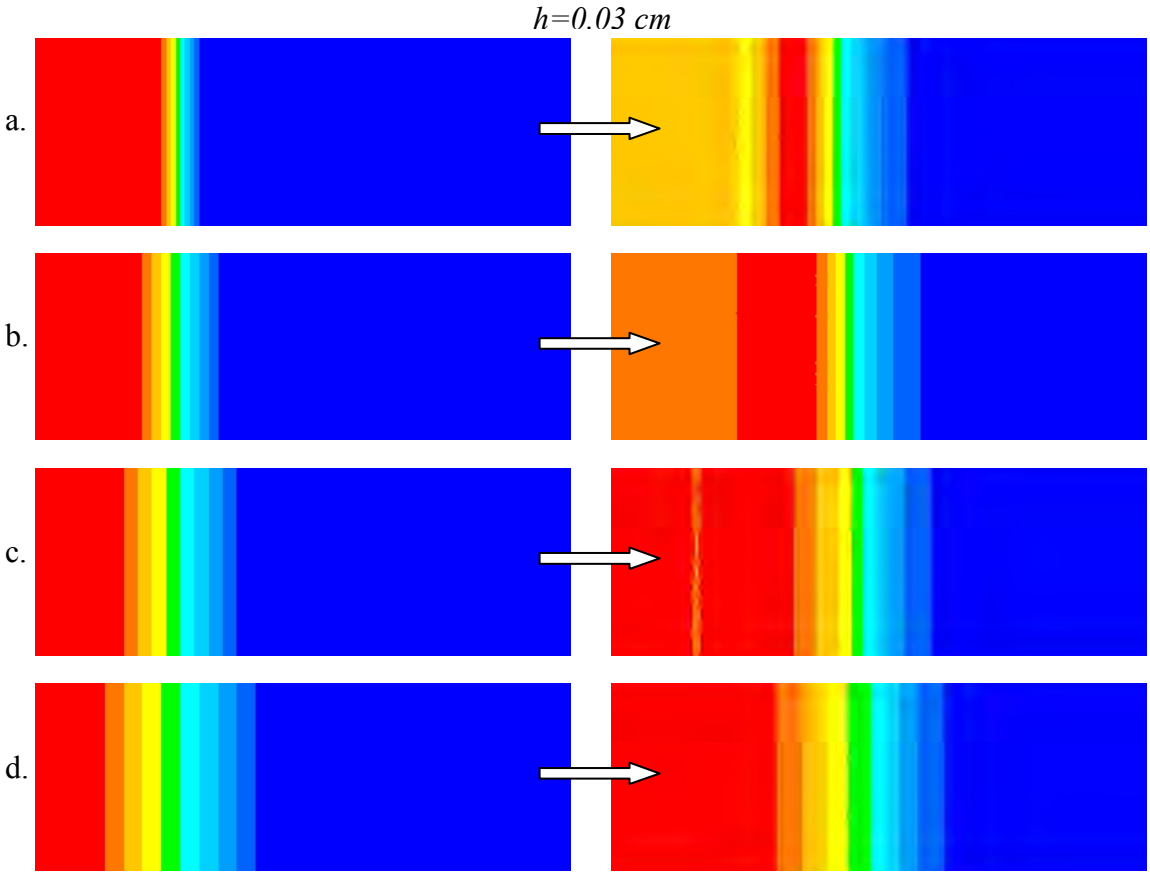


Figure 27 – Transport d'une fonction avec différentes tailles de zone Level Set – h=0.03 cm

A noter que ces résultats sont indépendants de la taille de maille : le même cas test réalisé avec des maillages différents donne des résultats largement similaires.
Nous observons ensuite que lorsque ε est inférieure à $|v|\Delta t = 0.25\,\text{cm}$ (images a. et b.), le transport sur un incrément de temps donne de mauvais résultats : d'importantes oscillations apparaissent aux abords du niveau zéro qui est ici à l'intérieure de la zone verte. Puis, lorsque ε est supérieure à $0.25\,\text{cm}$ (images c. et d.), le transport sur un incrément de temps ne montre pas – ou peu – d'oscillations.

Une autre différence importante peut être trouvée en comparant les cas où $\varepsilon_0 < |v|\Delta t$ et $\varepsilon_0 > |v|\Delta t$. En effet, à $t = 0$, le niveau zéro de la fonction est sur $x = 1$; donc, avec une vitesse de 1cm/s et un pas de temps de 0.25s, ce niveau zéro doit se retrouver sur $x = 1.25$. Or, lorsque $\varepsilon_0 < |v|\Delta t$, nous voyons clairement que la zone verte est moins avancée que lorsque $\varepsilon_0 > |v|\Delta t$. De fait, dans le cas a., le niveau zéro de α est sur $x = 1.205$, alors que dans les cas c. et d., le niveau zéro de α est sur $x = 1.25$, ce qui est la valeur exacte. La conclusion est que, non seulement le fait d'avoir une zone Level Set trop petite ($\varepsilon_0 < |v|\Delta t$) cause des oscillations dans la solution, mais aussi, cela rend le transport non conservatif.

Le critère *(117)* précédemment formulé sur $\varepsilon_{\min}$ ne tient pas compte de la taille, et donc, $\varepsilon_{\min}$ peut potentiellement être inférieur à *h* lorsque la vitesse et le pas de temps sont petits. Cette éventualité étant clairement non souhaitable, il est indispensable d'ajouter une condition à notre critère qui détermine la taille minimale qu'une zone Level Set puisse avoir :

$$\varepsilon_{\min} = \max\left(|v|\Delta t, h\right) \qquad (119)$$

La propriété *(119)* permet de déterminer un critère de réinitialisation automatique. Dès que l'inégalité suivante est vérifiée, la fonction est trop altérée et doit être réinitialisée :

$$|\nabla\alpha| \times \max\left(|v|\Delta t, h\right) > \varepsilon_0 \qquad (120)$$

Une amélioration peut encore être apportée à ce modèle. Pour cela, une réflexion est menée sur la direction de la vitesse de transport v par rapport à la direction de variation de α : ce n'est pas uniquement la variation de α dans la direction de v qui importe vraiment et qui influe sur la qualité du transport. L'hypothèse suivante peut légitimement être formulée à ce sujet : si v et $\nabla\alpha$ sont orthogonaux, alors α ne varie pas dans la direction de v, et donc, le transport peut se réaliser sans problème, quelque soit la taille de la zone Level Set. A l'inverse, lorsque v et $\nabla\alpha$ ont la même direction, cela représente le pire des cas, puisque toute la variation de α se fait dans la direction de la vitesse. Or, jusqu'à présent, c'est uniquement ce dernier cas de figure qui a été implicitement pris en compte. Cette hypothèse se vérifie-t-elle dans la pratique ? Pour en juger, un cas simple en deux dimensions est étudié : dans une cavité en forme de marche, un champ de vitesse fixe est calculé au préalable avec un solveur de Stokes incompressible stationnaire (voir figure *28*).

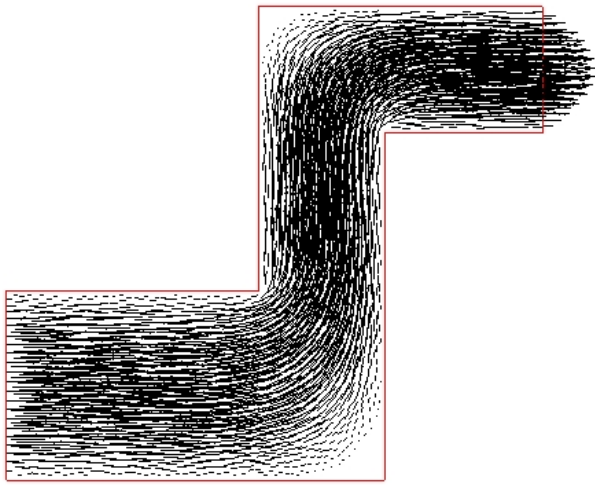


Figure 28 – Champ des vitesses

Avec ce champ de vitesse fixe, une fonction α avec zone Level Set inférieure à la taille minimum indiquée par la formule *(120)* est transportée sur un incrément de temps. Dans le premier cas test *(*figure *29)*, le gradient de α est proche d'être colinéaire à la vitesse (direction verticale sur la figure). Comme prévus, cela résulte en des oscillations qui polluent la solution. Par contre, dans le cas *(30)*, α a une zone Level Set de même taille (et donc théoriquement trop petite), et son transport ne pose aucun problème apparent. Cela est dû au fait que les directions du gradient de α et de la vitesse dans la zone Level Set sont proches d'être orthogonales : la vitesse a une direction plutôt verticale, tandis que $\nabla\alpha$ a une direction horizontale. Cet exemple valide bien l'hypothèse posée.

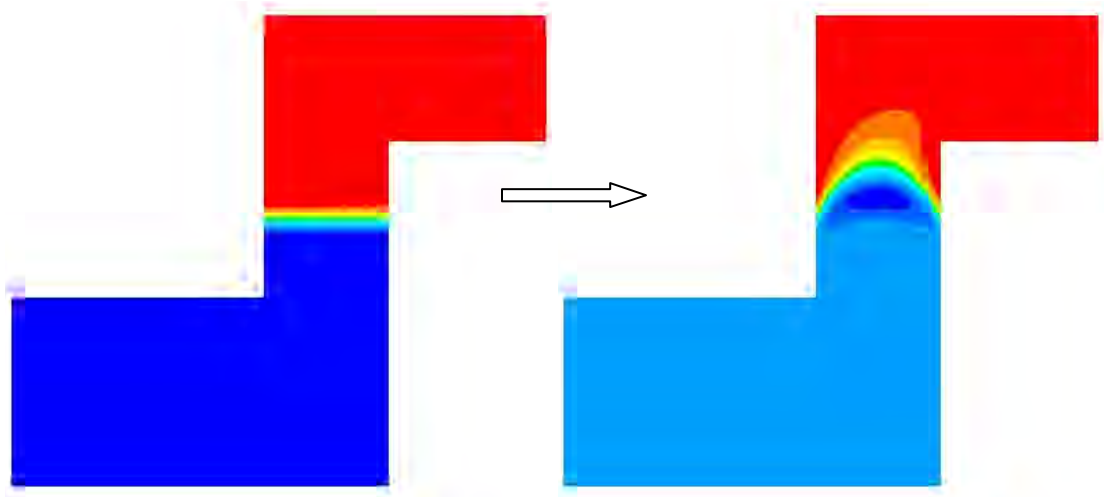


Figure 29 – Transport d'une zone Level Set avec $\nabla\alpha$ dans le sens de v

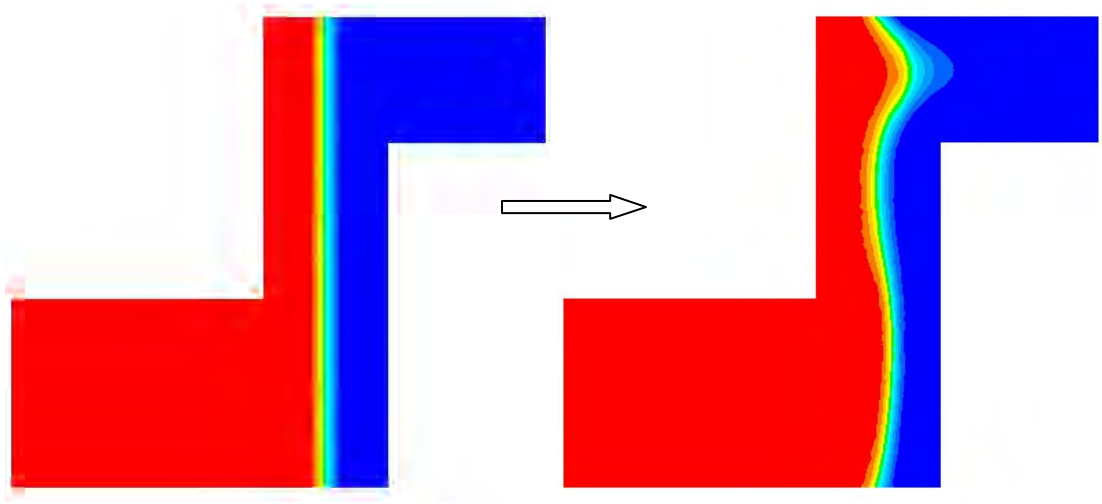


Figure 30 – Transport d'une zone Level Set avec $\nabla\alpha$ orthogonal à v

Imaginons qu'à un moment au cours d'une simulation, la taille de la zone Level Set passe en dessous de $\varepsilon_{\min}$ définit comme *(119)*. Cela a pour conséquence de déclancher une réinitialisation, alors que si, à ce moment précis, v et $\nabla\alpha$ sont proches d'être orthogonaux, le transport peut encore d'effectuer proprement. Afin d'éviter ce genre d'opération superflue, il faut prendre en compte cette dernière observation. Un bon moyen pour y parvenir est de remplacer $|\nabla\alpha|$ dans *(120)* par :

$$\frac{|\nabla\alpha.v|}{|v|} \qquad (121)$$

Pour mieux voir ce que cela représente :

$$\frac{|\nabla\alpha.v|}{|v|} = \frac{|\nabla\alpha||v|\cos(\theta)}{|v|} = |\nabla\alpha|\cos(\theta) \qquad (122)$$

où θ est l'angle entre les vecteurs v et $\nabla\alpha$. Avec cette expression, la valeur $|\nabla\alpha|$ est atteinte lorsque $\cos(\theta)=1$ et donc quand v et $\nabla\alpha$ sont colinéaires, ce qui constitue le pire des cas. Par contre, lorsque v et $\nabla\alpha$ ne sont pas colinéaire, c'est une valeur inférieure à $|\nabla\alpha|$ qui est atteinte ; et même, dans le meilleure des cas, si $\cos(\theta)=0$ et donc v et $\nabla\alpha$ sont orthogonaux, la valeur est nulle. Tout ceci est en parfait accord avec se que nous recherchons, et le critère de réinitialisation devient :

$$\frac{|\nabla\alpha.v|}{|v|} \times \max(|v|\Delta t, h) > \varepsilon_0 \qquad (123)$$

Dès que cette propriété est vérifiée, la fonction Level Set doit être remise à jour.

4.8. Une méthode de couplage transport-réinitialisation d'une fonction distance : Leveller

Une perspective d'optimisation du transport d'interface par Level Set est envisageable lorsque l'on considère un couplage fort entre l'équation de transport et l'algorithme de réinitialisation. Cette technique, proposée dans [Coupez, 2006], part de l'observation que seule la vitesse du phénomène physique autour de l'interface est importante. En effet, la seule information que l'on peut retirer de la fonction Level Set pour localiser l'interface est son niveau zéro. Partout ailleurs dans le domaine de calcul, les valeurs de la fonction Level Set ne correspondent à aucune quantité physique. Cependant, pour que le transport se réalise efficacement, elle doit seulement garder une forme à peu près régulière.

4.8.1. Formulation

Considérons les deux équations de transport et de réinitialisation appliquées à la fonction Level Set α :

$$\begin{cases} \dfrac{\partial \alpha}{\partial t} + \nabla \alpha . v = 0 \\ \dfrac{\partial \alpha}{\partial \tau} = S(\alpha)\left(1 - |\nabla \alpha|\right) \end{cases} \qquad (124)$$

Afin d'éliminer le temps fictif τ , nous substituons localement le pas de temps fictif $\Delta\tau$ par la taille de maille *h* pour obtenir :

$$\frac{\partial \tau}{\partial t} = \frac{h}{\Delta t} \qquad (125)$$

Comme précédemment, on réécrit le terme suivant :

$$|\nabla \alpha| = \nabla \alpha \frac{\nabla \alpha}{|\nabla \alpha|} \qquad (126)$$

Enfin, le couplage fort entre les deux équations de *(124)* donne l'équation unique suivante :

$$\frac{\partial \alpha}{\partial t} + \nabla \alpha . v + \frac{h}{\Delta t} S(\alpha) \nabla \alpha \frac{\nabla \alpha}{|\nabla \alpha|} = \frac{h}{\Delta t} S(\alpha) \qquad (127)$$

4.8.2. Interprétation

(127) est une équation de transport où la vitesse de convection est inchangée autour de l'interface, c'est-à-dire que c'est exactement celle donnée par le champ de vitesse de l'écoulement. De cette manière, l'interface est transportée correctement d'un point de vue physique. Par contre, la vitesse de convection est modifiée partout ailleurs dans la cavité de telle façon que la fonction Level Set se contente de garder une forme régulière (comme une fonction distance) sans prendre en compte la vitesse de l'écoulement qui a tendance à la

déformer. A la place, c'est une vitesse de convection similaire à celle utiliser dans l'algorithme de réinitialisation pour propager la propriété de distance à partir du niveau zéro de α .

En théorie, il suffit donc de résoudre l'équation *(127)* pour transporter l'interface d'un écoulement multi-fluide tout en gardant une fonction Level Set régulière, proche d'une fonction distance. Puisqu'il s'agit du même type de problème que l'on a vu jusqu'à présent (équation linéaire dominée par la convection), la résolution s'effectue de la même façon : par une méthode éléments finis stabilisée par SUPG ou *Residual-Free Bubbles*.

4.8.3. Premiers résultats

Le but ici est de montrer que les premiers tests réalisés sur le solveur d'interface Leveller (couplage transport-réinitialisation) sont encourageants, et demandent donc un approfondissement et une étude plus complète.

Le cas test du vortex constitue ici encore une application intéressante pour valider le couplage entre le transport de l'interface et la réinitialisation Level Set. La figure *(31)* montre le niveau zéro de α à la fin de la simulation.

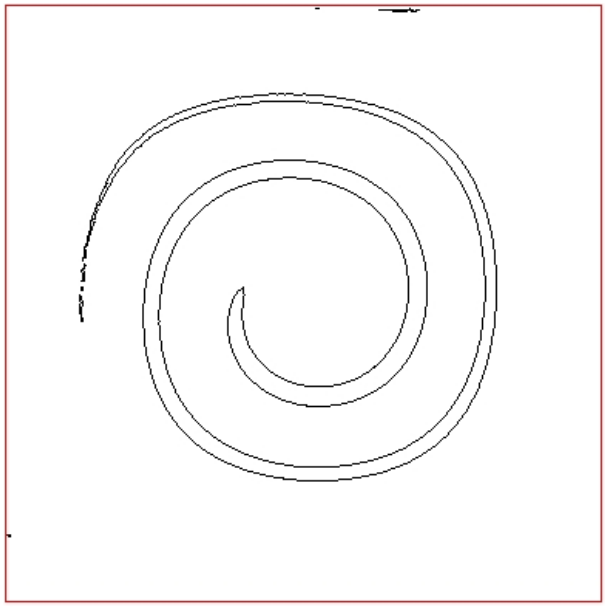


Figure 31 – Solution finale avec Leveller

La forme globale du serpentin semble assez bonne par rapport à la solution de référence *(figure 15)*, bien que toujours un peu trop courte. En tout cas, ce résultat est meilleur que dans le cas du transport sans réinitialisation *(figure 16)* car la longueur du filament est plus proche de celle de la solution de référence. Aussi, il est meilleur que dans le cas du transport avec réinitialisation Level Set classique *(figure 24)* car l'épaisseur du serpentin n'est pas surestimée. La conservation est plutôt bonne, puisqu'un gain de 5% est constaté dans ce cas.

Par contre, des problèmes aux bords de la cavité apparaissent avec des perturbations dans l'interface des sous domaines. Ces problèmes viennent des conditions aux limites lors de l'étape de réinitialisation : selon le champ des vitesses de convection des équations de Hamilton-Jacobi, des conditions limites doivent être ajoutées pour tenir compte des vitesses

rentrantes dans la cavité. Ce point peut donc constituer une amélioration, et doit donc être étudié.

Les premiers tests de cette méthode Level Set innovante semble donner des résultats prometteurs, mais une étude plus complète et approfondie doit toujours être menée sur ce sujet. Cela représente en tout cas une perspective intéressante. D'autant plus que cette méthode de couplage condense les étapes de transport et de réinitialisation. Ainsi, un seul et même solveur est nécessaire, contrairement à la méthode Level Set standard qui exige deux étapes distinctes. Les conséquences sur la performance sont donc importantes.

5. Conclusion sur les techniques de capture d'interface

5.1. *VOF* - Level Set

Le grand point fort de la méthode *VOF* est sa très bonne conservation de la masse puisqu'elle est précisément basée sur ce principe. Aussi, le caractère discontinu de cette approche permet l'élaboration d'une formulation éléments finis par Galerkin discontinu qui se comporte très bien avec les problèmes de convection pure. Ainsi, l'équation de transport est résolue avec une méthode de résolution robuste en termes de stabilité et de pas de temps.
Par contre, elle comporte plusieurs inconvénients de taille qui nous a poussé à élaborer une autre méthode. Premièrement, il y a un nombre élevé d'inconnues à résoudre à cause de leur nature discontinue : il y en a deux fois plus que le nombre d'éléments dans le maillage. Aussi, la matrice du système linéaire contient une largeur de bande importante, ce qui alourdie non seulement la résolution, mais aussi la quantité de mémoire nécessaire. Deuxièmement, le bas degré d'interpolation spatiale *P0* engendre une localisation géométrique peu précise de l'interface. Et ce phénomène est d'autant plus sérieux qu'une importante diffusion numérique vient « étaler » l'interface pour la rendre extrêmement diffuse, et donc, difficilement détectable. Une technique d'adaptation dynamique du maillage permet néanmoins de limiter efficacement cette diffusion, mais elle implique d'importante manipulations, à la fois lourdes et peu flexibles.

L'approche Level Set permet de faciliter la résolution avec beaucoup moins d'inconnues à résoudre : avec une interpolation spatiale continue de type *P1*, il y a autant d'inconnues que de nœuds dans le maillage. Cela rend donc la résolution plus rapide et moins lourde en quantité de mémoire nécessaire. De plus, la considération du couplage entre notre solveur d'écoulement (avec le problème de Navier-Stokes incompressible) de type *P1+/P1* et la capture d'interface de type *P1* continu est plus naturelle qu'avec du *P0* discontinu. En ce qui concerne l'interface elle-même, le degré d'interpolation plus élevé permet une description géométrique plus précise. Mais aussi, notre loi de mélange pour la modélisation multi-fluide interdit toute diffusion numérique, puisque seulement les éléments du maillage traversés par l'interface contiennent un mélange des différents fluides de l'écoulement. En comparaison avec la méthode *VOF*, ce dernier point est très important car, à priori, il n'y a pas nécessité d'ajouter une adaptation dynamique du maillage.
Cependant, quelques problèmes ont été mis en évidence par l'étude de cas tests. Tout d'abord, dans le contexte de résolution par éléments finis, les problèmes linéaires hyperboliques représentent une difficulté pour le Galerkin standard. Ainsi, une stabilisation de type SUPG ou *Residual-Free Bubbles* doit être ajoutée pour que la solution du problème discret converge vers celle de la formulation forte. Ensuite, le cas test du vortex montre les deux points faibles de la méthode Level Set qui apparaissent avec un champ de vitesse de convection fortement cisaillé, ou avec une géométrie d'interface trop fine. En effet, la fonction Level Set se détériore d'autant plus rapidement que le cisaillement de l'écoulement est fort. Entre autres, ses lignes de niveau se resserrent ou s'écartent les unes des autres pour finalement altérer la régularité de la fonction. Autrement dit, la propriété de distance sur la fonction Level Set n'est pas maintenue par le transport de l'interface, et donc une technique de réinitialisation est implémentée pour lui redonner une régularité de fonction distance. Enfin, le plus gros désavantage est le manque de conservation. Non seulement le cas du vortex fait apparaître des défauts lorsque la forme de l'interface devient trop fine, mais aussi, l'étape de réinitialisation peut introduire quelques mouvements d'interfaces. L'algorithme utilisé présente de nombreux

avantages : c'est une procédure robuste, qui permet de calculer avec précision les caractéristiques géométriques de l'interface. Cependant, son utilisation peut dégrader les propriétés de conservation de masse de la méthode. Plusieurs auteurs ont travaillé sur ce problème afin d'améliorer la résolution de la fonction, mais ce problème reste encore ouvert à l'heure actuelle.

5.2. Perspectives d'améliorations

Des tentatives d'amélioration de la méthode Level Set sont présentées dans la littérature, et surtout sur le sujet de la conservation qui représente son point faible principal. Parmi elles, on trouve un algorithme de réinitialisation avec contrainte sur la conservation de la surface dans [Sussman, 1998]. Cette contrainte se traduit par l'ajout d'un terme tenant compte de conservation de la surface par rapport au processus de réinitialisation :

$$\frac{\partial}{\partial \tau}\left(\int_K 2S(\phi) - 1 \right) = 0 \qquad (128)$$

Dans [Chang, 1996], un autre algorithme est développé pour essayer de rajouter ou d'enlever la quantité de surface manquante ou en excès, suivant le signe du défaut de masse, mais cette technique semble assez lourde.

D'autres améliorations peuvent aussi être apportées en dressant des méthodes de couplages entre différentes techniques de calcul d'interfaces existantes. Ainsi, un couplage Level Set - *Volume Of Fluid* appelé *CLSVOF* est présenté dans [Sussman, 2000]. L'algorithme obtenu est plus complexe mais semble prometteur. Le but de leurs travaux est de combiner les bonnes propriétés géométriques de la méthode Level Set (calcul de la normale et de la courbure à l'interface immédiat), à une méthode qui conserve la quantité de surface implicitement. Une procédure différente est appliquée pour recalculer la fonction distance à l'interface afin d'éviter les mouvements d'interface pendant la réinitialisation. Cette méthode peut donc représenter une perspective intéressante de développement de notre travail.

Finalement, un couplage entre la méthode Level Set et une méthode Lagrangienne de type de *Marker And Cell* est considéré. Cette méthode récente nommée *Particle Level Set* a été introduite dans [Enright, 2002] et vient du constat que les méthodes lagrangiennes apportent une bonne précision numérique. Il parait donc naturellement bénéfique de combiner la finesse de résolution des méthodes lagrangiennes, tout en conservant la robustesse et la précision de la méthode Level Set. L'idée de base est simple, puisqu'il s'agit de corriger les erreurs numériques induites sur la position de la ligne de niveau zéro.
Les résultats présentés par [Enright, 2005] montrent que l'algorithme est d'une grande précision, mais cette méthode semble cependant un peu plus difficile à mettre en œuvre et à gérer sur des simulations d'écoulements turbulents 3D effectuées sur des machines parallèles. En effet, l'un des inconvénients que l'on rencontre est d'optimiser la distribution des particules dans les zones où il est nécessaire d'avoir une résolution très fine, afin d'alléger les contraintes en temps de calcul et en espace mémoire.

5.3. La conservation

On constate clairement à la suite de cette revue bibliographique des études sur la conservation de la masse que le problème reste particulièrement difficile à résoudre. Parmi les différentes méthodes numériques développées par différents auteurs, le couplage de la méthode Level Set avec la méthode *VOF* ou avec une méthode lagrangienne peuvent être une réponse à ce problème. Il semble cependant que le couplage avec une méthode lagrangienne soit rapidement prohibitif en temps de calcul sur des configurations 3D. Le couplage avec une méthode *VOF* reste également à être étudié plus précisément afin de déterminer sa pertinence. Dans ce travail de thèse, les méthodes utilisées montrent des résultats satisfaisants, même si la conservation de la masse n'est pas garantie. Nous n'utilisons pas de méthodes de correction des pertes de masse, et plusieurs perspectives d'améliorations sont envisageables. En revanche, nous cherchons toujours à quantifier les défauts de masse dans nos simulations afin de montrer que, même si la conservation n'est pas parfaire, les pertes restent « raisonnables » dans le sens ou elles n'altèrent pas trop la solution calculée.

Références

[Adalsteinsson, 1995] D. Adalsteinsson, and J. A. Sethian, "A fast level set method for propagating interfaces", *J. Comput. Phys.*, 118, 269 (1995).

[Batkam, 2002] S. Batkam. "Thermique multidomaine en simulation numérique du remplissage 3D", *Thèse de doctorat*, Ecole Nationale supérieure des Mines de Paris (2002).

[Bigot, 2000] E. Bigot and T. Coupez. "Capture of 3D moving free surfaces and material interfaces by mesh deformation", *In European Congress on Computational Methods in Applied Sciences and Engineering* (2000).

[Bigot, 2001] E. Bigot. "Simulation tridimensionnelle du remplissage de corps mince par injection", *Thèse de doctorat*, Ecole Nationale Supérieure des Mines de Paris (2001).

[Brezzi, 1994] F. Brezzi and A. Russo, Choosing bubbles for advection-diffusion problems, Math. Models Meth. Appl. Sci., 4, pp. 571--587 (1994).

[Brezzi, 1997] F.Brezzi, L.P. Franca, T.J.R. Hugues, A.Russo. "$b = \int g$", *Computer methods in applied mechanics and engineering*, 145, 329-339 (1997).

[Brooks, 1982] A. N. Brooks, T. J. R. Hughes, "Streamline upwind/Petrov-Galerkin formulations for convective dominated flows with particular emphasis on the incompressible Navier-Stokes equations", *Comp. Meth. in Appl. Mech. Eng.*, 32, 199-259 (1982).

[Bruchon, 2003] J. Bruchon et T. Coupez, "Etude de la formation d'une structure de mousse polymère par simulation de l'expansion anisotherme de bulles de gaz", *Mécanique & Industries,* Volume 4, 331 (2003).

[Bruchon, 2004] J. Bruchon, "Etude de la formation d'une structure de mousse par simulation directe de l'expansion de bulles dans une matrice liquide polymère", *Thèse,* Ecole Nationale Supérieure des Mines de Paris (2004).

[Chang, 1996] Y. C. Chang, T. Y. Hou, B. Merriman, and S. J. Osher, "A Level Set Formulation of Eulerian Interface Capturing Methods for Incompressible Fluid Flows", *Jour. Comp. Phys.*, 124, pp. 449-464 (1996).

[Chen, 1991] Y. G. Chen, Y. Giga, and S. Goto, "Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations", *J. Diff. Geom.*, 33, 749 (1991).

[Coupez, 2000] T. Coupez, J. Bruchon, et S. Batkam, "Space-Time Finite Element Method for 3D Process Modelling", *18th Annual Meeting of the Polymer Processing Society*, PPS-18, Guimarães, Portugal (2002).

[Coupez, 2006] T. Coupez, "Réinitialisation convective et locale des fonctions Level Set pour le mouvement de surfaces et interfaces", *Journées Activités Universitaires de Mécanique*, La Rochelle, France (2006).

[Enright, 2002] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, "A Hybrid Particle Level Set Method for Improved Interface Capturing", *Journal of Computational Physics*, 183, 83 (2002).

[Enright, 2005] D. Enright, F. Losasso, and R. Fedkiw, "A fast and accurate semi-Lagrangian particle level set method", *Computers & Structures*, 83, 479 (2005).

[Evans, 1991] L. C. Evans, and J. Spruck, "Motion of Level Set via mean curvature I", *J. Diff. Geom.*, 33, 635 (1991).

[Franca, 2000] L.P. Franca, A. Russo: On Petrov-Galerkin formulations for the linear hyperbolic equation, appeared with the title ``Recovering SUPG using Petrov-Galerkin formulations enriched with adjoint residula-free bubbles" on *Comput. Methods Appl. Mech. Engrg.*, 182, 333-339 (2000).

[Harabetian, 1996] E. Harabetian, S. Osher, and C. W. Shu, "An Eulerian approach for vortex motion using a level set regularization procedure", *J. Comput. Phys.*, 127, 15 (1996).

[Hughes, 1979] T. J. R. Hughes, A. N. Brooks, "A multi-dimensional upwind scheme with no crossing diffusion", *Finite Element Methods for Convection Dominated Flows*, vol. 34, ASME, New York, 19-35 (1979).

[Hughes, 1982] T. J. R. Hughes, A. N. Brooks, "A theoretical framework for Petrov-Galerkin methods with discontinuous weighting functions: Applications to the streamline-upwind procedure", *R.H.G. et al. (Ed.), Finite Elements in Fluids*, vol. IV, Chichester, Wiley (1982).

[Magnaudet, 1995] Magnaudet J. , Rivero M. , Fabre J. "Accelerated flows around a rigid sphere or a spherical bubble. Part 1: Steady straining flow", *J. Fluid Mech.* 284, 97-136 (1995)

[Magnin, 1994] B. Magnin. "Modélisation du remplissage des moules d'injection pour les polymères thermoplastiques par une méthode eulérienne-lagrangienne arbitraire", *Thèse de doctorat,* Ecole Nationale supérieure des Mines de Paris (1994).

[Merriman, 1994] B. Merriman, J. Bence, and S. Osher, "Motion of multiple junctions: A Level Set approach", *J. Comput. Phys.*, 112, 334 (1994).

[Osher, 1988] S. Osher, and J. A. Sethian, "Fronts propagating with curvature dependant speed: Algorithms based on Hamilton-Jacobi formulation", *J. Comput. Phys.*, 79, 12 (1988).

[Peng, 1999] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang, "A PDE-Based Fast Local Level Set Method", *J. Comput. Physics*, 155, 410 (1999).

[Pichelin, 1998] E. Pichelin and T. Coupez. "A Taylor discontinuous Galerkin method for the thermal solution in 3D mold filling" *Computer methods in applied mechanics and engineering* (1998).

[Rider, 1995] W. J. Rider, and D. B. Kothe, "Stretching and tearing interface tracking methods", *in 12th AIAA CFD Conference Paper*, 95, 1 (1995).

[Sethian, 1984] J.A. Sethian, "Turbulent combustion in open and closed vessels", *J. Comp. Phys.*, 54, 425-456 (1984).

[Sussman, 1994] M. Sussman, P. Smereka, and S. Osher, "A level set method for computing solutions to incompressible two-phase flow", *J. Comput. Phys.*, 114, 146 (1994).

[Sussman, 1998] M. Sussman, E. Fatemi, P. Smereka, and S. Osher, "An Improved Level Set Method for Incompressible Tow-Phase Flows", *Computers & Fluids*, 27, 663 (1998).

[Sussman, 1999] M. Sussman, A. Almgren, J. Bell, P. Colella, L. Howell, and M. Welcomey, "An Adaptive Level Set Approach for Incompressible Two-Phase Flows", *Journal of Computational Physics*, 148, 81 (1999).

[Sussman, 2000] M. Sussman, E.G. Puckett, "A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows", *J. Comp. Phys.*, 162, 301 (2000).

[Vincent, 1998] S. Vincent and J-P Caltagirone, "Efficient solving method for unsteady incompressible interfacial flow problems". *International Journal for numerical methods in fluids*, 30, 795 (1998).

[Zhao, 1996] H. K. Zhao, T. Chan, B. Merriman, and S. Osher, "A variational level set approach to multi-phase motion", *J. Comput. Phys.*, 122, 197 (1996).

[Zhao, 1998] H. K. Zhao, B. Merriman, S. Osher, and L. Wang, "Capturing the behavior of bubbles and drops using the variational level set approach", *J. Comput. Phys.*, 143, 495 (1998).

APPLICATIONS

1. Introduction

Dans ce dernier chapitre, les méthodes numériques implémentées tout au long de cette thèse sont utilisées pour simuler des écoulements multi fluides. Pour cela, les solveurs d'écoulement et d'interface sont employés dans un contexte parallèle et de grille de calcul. Des benchmarks reconnus et prédéfinis en 2 dimensions et permettent de valider le couplage entre nos solveurs qui ont déjà été validés individuellement. De plus, des applications de surfaces libres variées en 3 dimensions sont présentées comme exemples d'utilisation de la librairie CimLib.

Après avoir implémenté et comparé un certain nombre de méthodes numériques dans le cadre de simulations d'écoulement multi fluides, le but est de les utiliser pour effectuer des calculs à plus ou moins grande échelle, qui serviront à la fois à tester nos techniques dans leur globalité, mais aussi à montrer des applications possibles en mécanique des fluides hétérogènes. C'est avec notre code de calcul implémenté en C++ que les simulations numériques sont exécutées. La librairie CimLib développée au CEMEF contient un ensemble de solveurs numériques qui s'appuient sur des méthodes de type éléments finis. Elle a été parallélisée en utilisant la librairie *MPI* (*Message Passing Library*) largement reconnue lorsque l'on parle de programmation parallèle par envois de messages. Les matrices locales qui sont calculées dans CimLib sont ensuite prisent en charge par PETSc (*Portable and Extensible Toolkit for Scientific Computation*), librairie qui permet de résoudre en parallèle et de façon itérative des systèmes d'équations linéaires et non linéaires impliquant des matrices creuses. Les matrices locales que PETSc reçoit, sont assemblées en une matrice globale ; puis, plusieurs méthodes de préconditionnement sont à disposition pour permettre une meilleure résolution : selon la nature du système à résoudre, des préconditionneurs de type ILU(k) ou Jacobi par blocs sont souvent choisis. Enfin, le système linéaire global est résolu en utilisant une méthode du résidu conjugué, soit MINRES, soit GMRES, selon les propriétés du système.

Les méthodes d'éléments finis qu'utilisent les solveurs de CimLib s'appuient sur un maillage : la cavité qui correspond au domaine de calcul est maillée avec des tétraèdres (ou des triangles en 2D). Puis, ce maillage est partitionné pour les calculs parallèles, dont les communications inter processeurs effectuées lors du produit matrice - vecteur dépendent directement du nombre de nœuds à l'interface des partitions.

Conformément aux méthodes choisies et développées tout au long de cette thèse, les inconnues de tous nos systèmes à résoudre sont exclusivement situées sur les nœuds du maillage. Ainsi, dans le problème de Navier Stokes (solveur P1+/P1 pour les écoulements), il y a 4 inconnues par nœud en 3D (3 pour la vitesse, et 1 pour la pression) et 3 inconnues par noeud en 2D (2 pour la vitesse, et 1 pour la pression). Le transport, lui (solveur pour l'évolution des phases), résulte en un système à 1 inconnue par nœud. En effet, la fonction de phase transportée est une fonction Level Set qui est à la fois scalaire et continue (P1). Ces deux systèmes sont résolus l'un après l'autre, et ce, à chaque incrément de temps, tout au long de la simulation. Par conséquent, il y 5 inconnues à trouver par nœud et par incrément de temps. Toutefois, une comparaison pourra être dressée avec d'autres types de méthodes et d'interpolations spatiales et temporelles.

2. Benchmark du Soliton

Le benchmark du Soliton est un cas test reconnus pour évaluer des codes de simulations numériques en mécanique des fluides. Il propose une problématique de cassure de vague sur laquelle certaines modélisations peuvent être comparées. Il s'agit de simuler la cassure d'une vague solitaire (dite soliton) due à la gravité sur un récif en forme de marche. Plus précisément, un soliton est considéré comme une onde solitaire qui se propage sans se déformer dans un milieu non-linéaire et dispersif. Le cas test est proposé et décrit avec précision dans la référence [Yasuda, 1997] qui fourni des résultats expérimentaux et numériques. Il est classique de supposer dans cette simulation que la globalité de l'écoulement (à la fois l'eau et l'air) est incompressible, ce qui fait que cette étude se porte bien à nos travaux. Ce cas test permet d'évaluer le comportement de solveurs numériques appliquées à une cassure de vague : plusieurs publications, comme [Helluy, 2005], montrent les résultats obtenus par des codes différents et permettent donc une comparaison avec notre code de calcul.

2.1. Conditions Initiales

Les conditions de l'application doivent être exactement celles décrites dans [Helluy, 2005] pour pouvoir comparer correctement notre solveur avec ceux dont les résultats sont présentés dans ce même article.

Au temps *t=0*, la pression et la vitesse dans l'eau sont connues et sont celles d'un soliton incompressible stable, calculées par la méthode de Tanaka présentée dans [Tanaka, 1986]. Dans l'air, la pression est de 105 Pa et la vitesse peut être négligée. La densité de l'eau est de 1000 kg/m^3, et celle de l'air est de 1 kg/m^3. Le sommet du soliton est initialement situé sur la position *x=0*, et se déplace vers la droite. Conformément aux études consacrées aux vagues solitaires, un soliton a une vitesse de $1.18\sqrt{gh_1}$, où h_1 est la hauteur initiale de l'eau. Dans notre cas, la surface de l'eau « calme » se trouve à $h_1 = 0.31m$ du fond de la cavité, alors que le haut de la vague se situe à $H_1 = 0.0131m$ au dessus du niveau de l'eau « calme ». Enfin, le soliton se propage à $c = 2.06\,m/s$. La figure suivante, tirée de [Tanaka, 1986], décrit le cas test et la cavité utilisée.

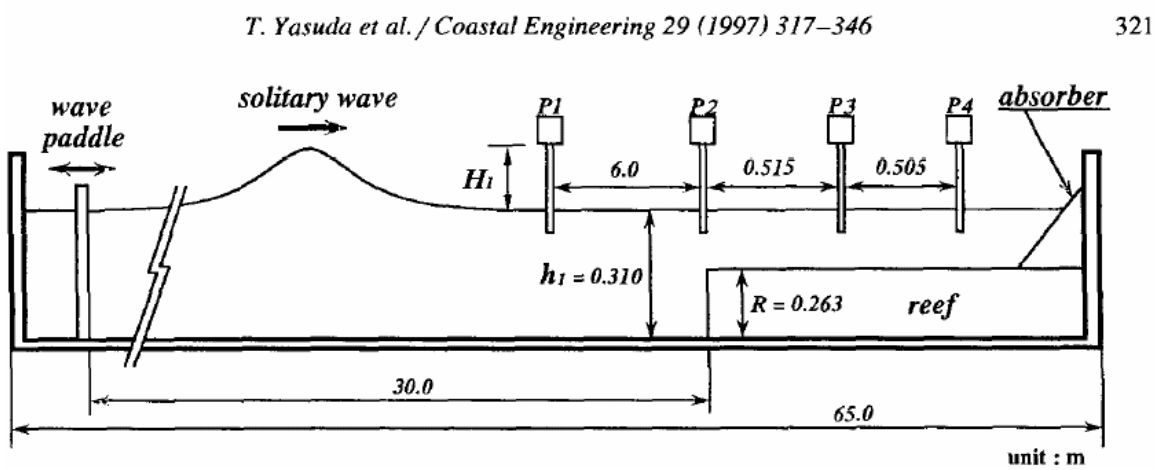


Figure 1 – Position de la marche et des jauges dans la cavité du soliton

Parmi les différents modèles comparés dans [Helluy, 2005], on trouve des solveurs compressibles, incompressibles, et des calculs d'interfaces basés sur du VOF ou du Level Set.

2.2. Évaluation numérique de la cassure d'une vague

Afin d'évaluer la précision des résultats obtenus numériquement, les données expérimentales de [Yasuda, 1997] sont utilisées comme une référence. De plus, [Helluy, 2005] met à disposition certains résultats numériques obtenus par différent codes de calcul et méthodes numériques. Deux formes de comparaisons sont possibles. La première concerne l'élévation de la surface de l'eau en fonction du temps au niveau des jauges *P1*, *P2*, *P3* et *P4* respectivement situées sur *x=0*, *x=2*, *x=2.515* et *x=3.02* (où l'unité est le mètre). La seconde consiste à visualiser les profils de la surface de l'eau à deux moments clés de la simulation : le point de cassure de la vague et l'initialisation de la chute.

2.3. Résultats numériques

2.3.1. Simulation

Nous utilisons un maillage isotrope en 2 dimensions qui contient 27000 nœuds et 53000 éléments. Pour avoir des résultats suffisamment précis sans pour autant trop rallonger les temps de calcul, le maillage est optimisé. De cette manière, il est raffiné dans les zones susceptibles d'être traversées par la surface de l'eau, et grossier partout ailleurs : nous choisissons des tailles de maille allant de *0.006 m* à *0.09 m* sur tout le domaine de calcul.

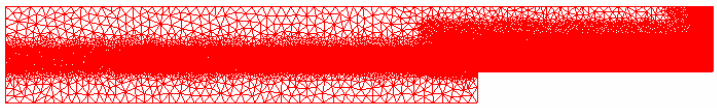


Figure 2 – Maillage 2D

Nous montrons ici plusieurs étapes de la simulation aux temps *t=0, t=1.2*, *t=1.4*, *t=1.6* et *t=1.8* (en secondes). Initialement, le soliton se propage vers la droite, mais tout en gardant le même profil. Lorsque celui-ci s'approche du récif en forme de marche, une vague se forme, puis se casse.

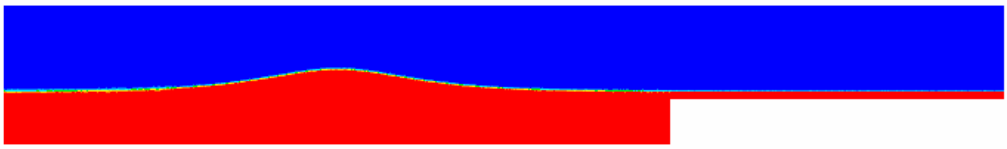


Figure 3 – Profil de surface libre à t=0 s

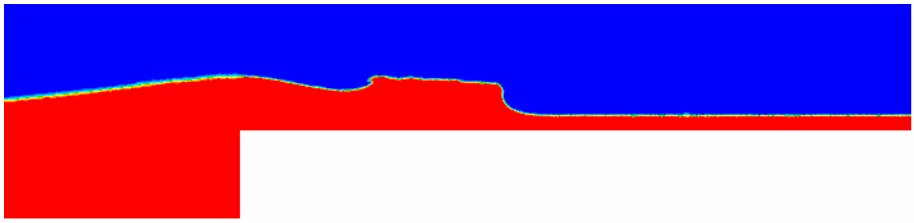


Figure 4 – Profil de surface libre à t=1.2 s

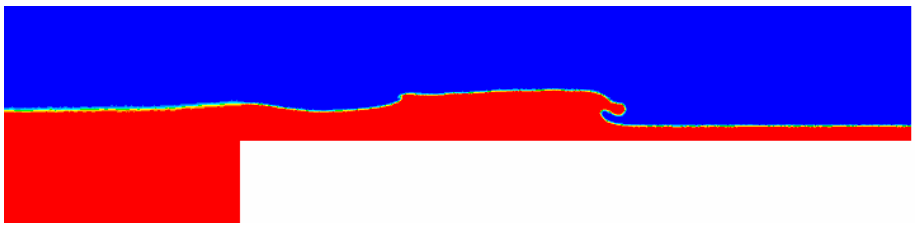


Figure 5 – Profil de surface libre à t=1.4 s

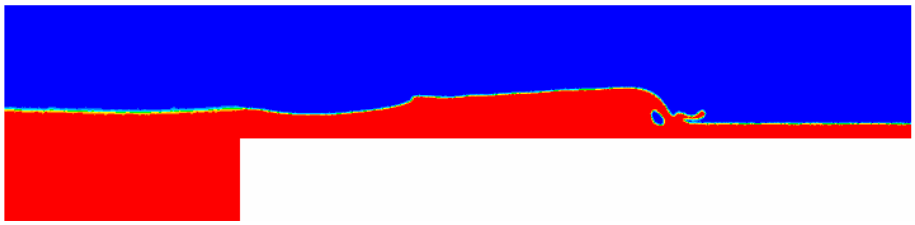


Figure 6 – Profil de surface libre à t=1.6 s

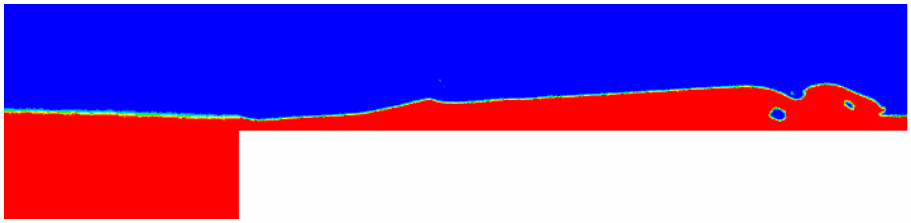


Figure 7 – Profil de surface libre à t=1.8 s

Compte tenu de la taille de ce problème en 2 dimensions, un PC (processeur Pentium 4 2.80 GHz et 1 Go de RAM) suffit à l'exécution de ce cas test. Le pas de temps est fixé à 0.001 seconde, et 2000 incréments en temps sont donc nécessaires pour obtenir une simulation sur 2 secondes. Avec des préconditionneurs adaptés à la résolution de l'écoulement par le problème de Navier Stokes (ILU 4) et à la résolution des mouvements d'interfaces méthode du Level Set (Jacobi par bloc), le temps de calcul total s'élève à environ 22 heures.

2.3.2. Jauges

Les figures suivantes montrent les graphes de l'élévation de la surface de l'eau en fonction du temps au niveau des jauges *P1*, *P2*, *P3* et *P4*. On peut y comparer les résultats obtenus par notre solveur et la référence que représentent les données issues d'expérimentations de [Yasuda, 1997].

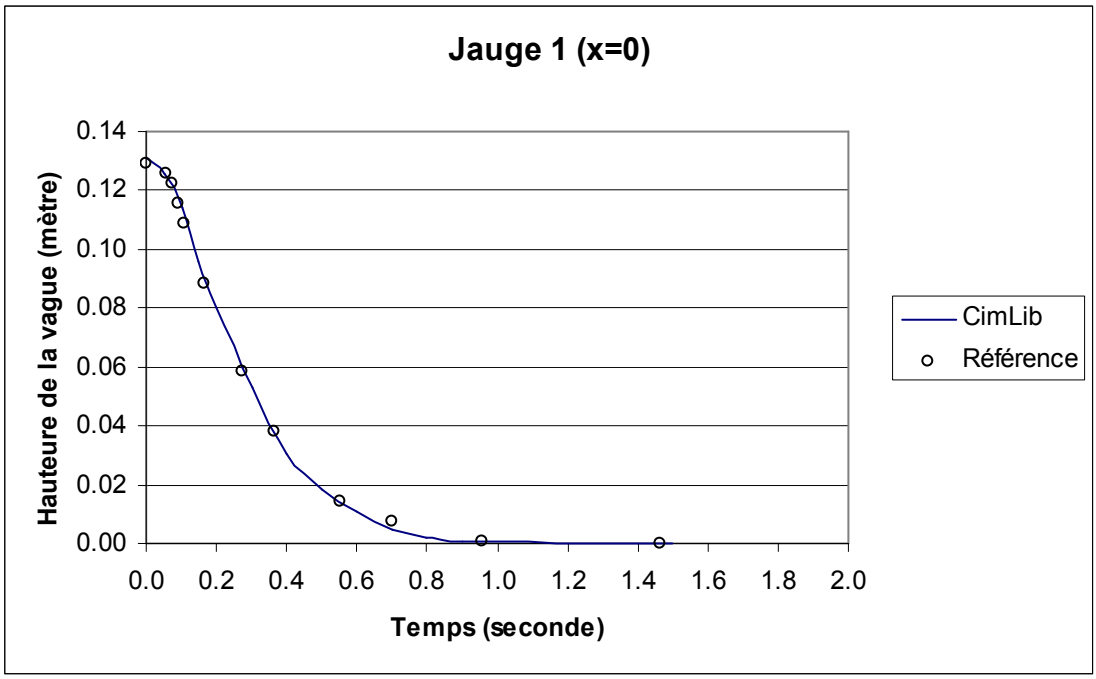


Figure 8 – Évolution de la surface de l'eau en fonction du temps : Jauge P1

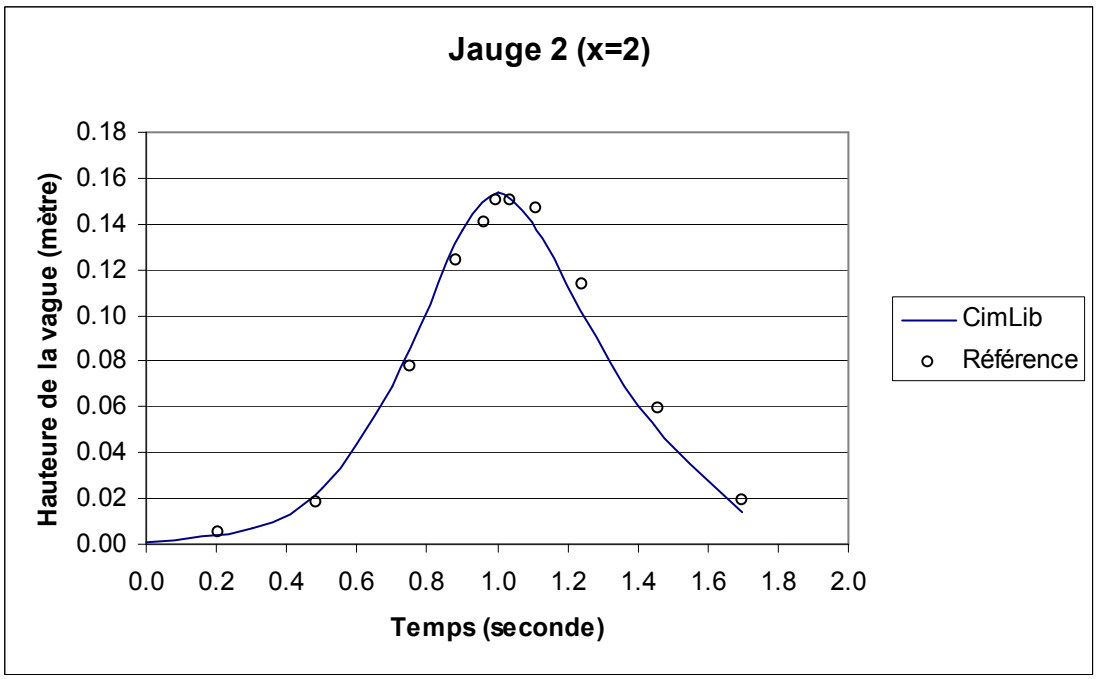


Figure 9 – Évolution de la surface de l'eau en fonction du temps : Jauge P2

- A la jauge *P1*, nos résultats numériques (la courbe bleue) sont bien en accord avec les données expérimentales de [Yasuda, 1997] (les points blanc). Cela montre que la propagation du soliton (à la fois sa vitesse et son élévation) est bien prédite avant la cassure de la vague.
- A la jauge *P2*, la courbe bleue est encore très proche des points blanc représentant les données expérimentales. Donc, l'élévation de la vague à l'endroit exact du commencement du récif est bien prédite, et de même pour le timing de la simulation.

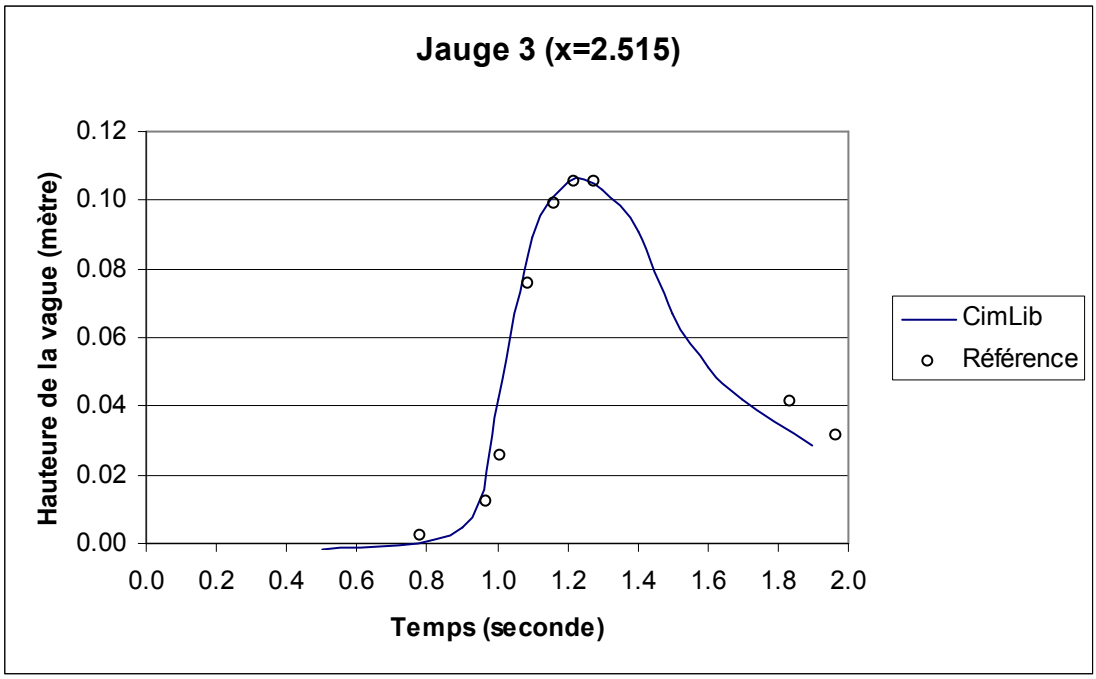


Figure 10 – Évolution de la surface de l'eau en fonction du temps : Jauge P3

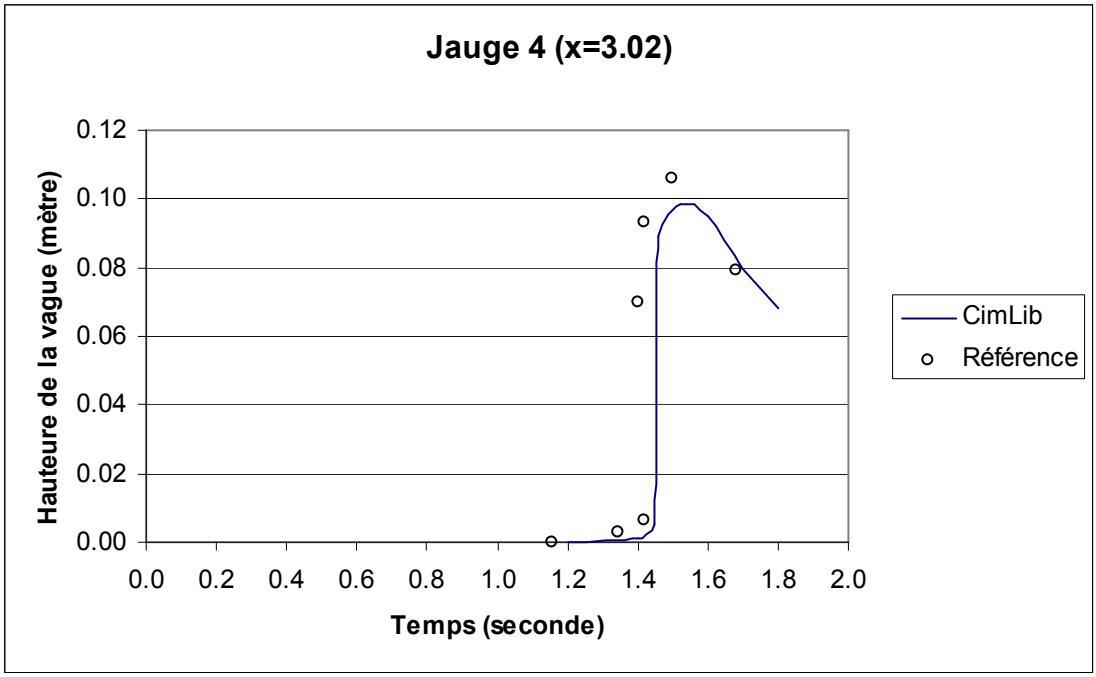


Figure 11 – Évolution de la surface de l'eau en fonction du temps : Jauge P4

- A la jauge *P3*, la courbe bleue monte en même temps que les points blanc jusqu'à *t=1.4s*, puis elle redescend trop vite par rapport à l'expérience. La phase montante de la vague est donc bien prédite par notre simulation. Notamment, la hauteur de la vague, ainsi que son timing, sont en accord avec les données expérimentales. Par contre, la phase descendante de la vague semble un peu plus rapide dans notre simulation que lors de l'expérience.
- Enfin, à la jauge *P4*, la courbe bleue est décalée vers la droite, ce qui retranscrit un retard de la cassure de la vague dans notre simulation par rapport à l'expérience. Aussi, le sommet de la vague à ce moment est sous-estimé.

2.3.3. Profil de la vague

Les deux graphes suivants représentent les profils de la vague au point de cassure et à l'initialisation de la chute. Des données expérimentales sont là aussi disponibles et permettent une vérification de nos résultats numériques.

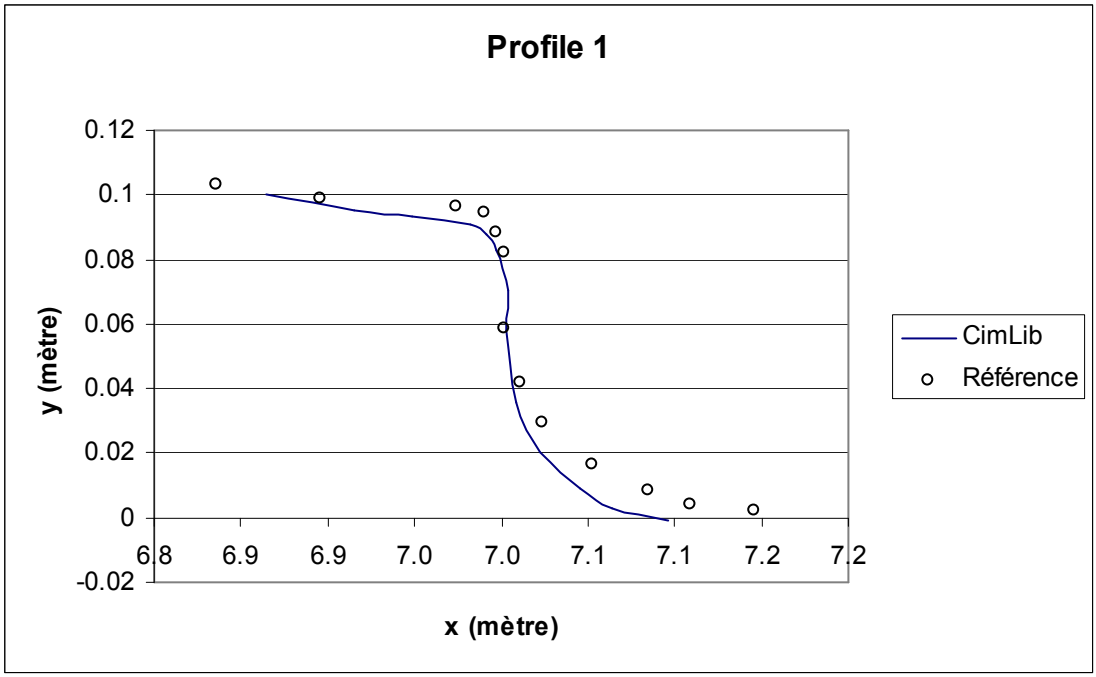


Figure 12 – Profil de la vague au point de cassure

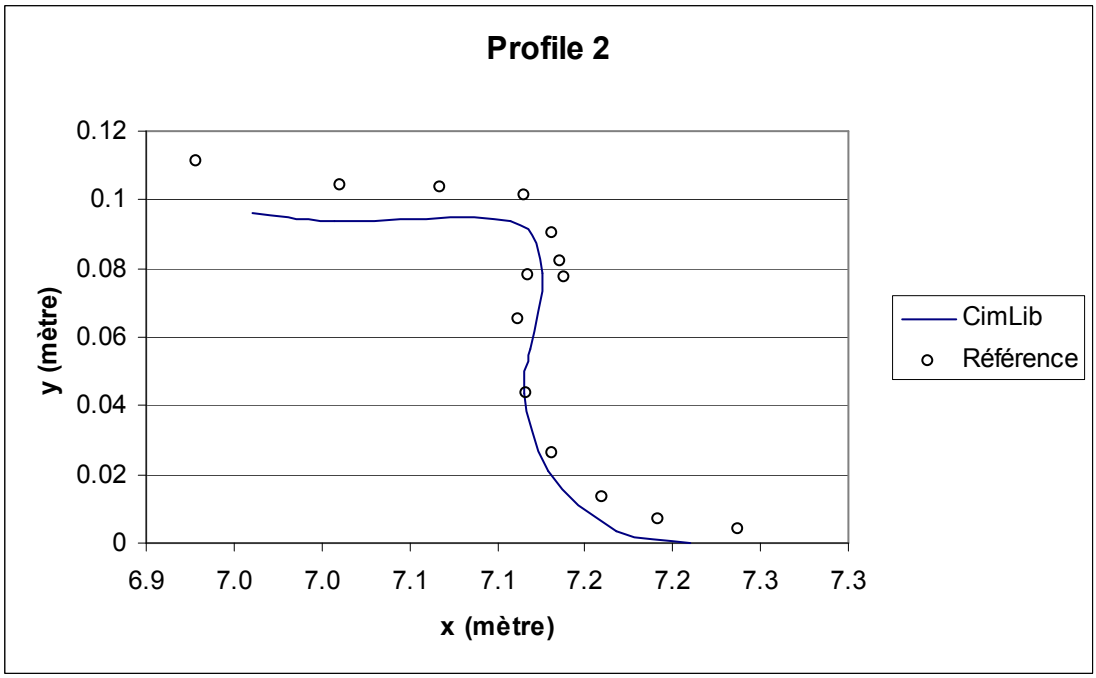


Figure 13 – Profil de la vague à l'initialisation de la chute

Les résultats de ce cas test que l'on peut trouver dans la littérature diffèrent énormément en fonction des méthodes numériques utilisées [Helluy, 2005]. Cela témoigne notamment de la complexité de cette application. Par conséquent, même si les profils de

vague des figures (*12*) et (*13*) montrent une différence assez importante entre nos résultats et l'expérience, on peut dire que notre simulation est qualitativement suffisamment proche de la solution attendue.

Plus précisément, on observe qu'en général, les simulations basées sur des méthodes de type Level Set montrent toutes, à peu près, les mêmes caractéristiques : elles présentent une forme trop arrondie au sommet de la vague au moment de la cassure.

2.4. Conclusion

Grâce à ce cas test du soliton, on peut évaluer les méthodes numériques développées dans cette thèse dans une application complexe de cassure de vague en 2 dimensions. Les résultats obtenus par notre code de calcul s'approchent plutôt bien de la référence que constituent les données expérimentales misent à disposition dans la littérature. Ainsi, nous pouvons valider à la fois le profil de la vague et le timing de la simulation. Toutefois, nous retrouvons ce qui semble être un défaut causé par les méthodes Level Set sur la forme de la vague au moment de sa cassure qui est plus « ronde » qu'avec d'autres méthodes numériques.

3. L'écroulement du barrage

3.1. Description

L'application de l'écroulement d'un barrage est un benchmark souvent utilisé pour tester des codes de simulation d'écoulements instationnaires de fluides et de surfaces libres, aussi bien en 2D qu'en 3D (voir [Martin, 1952], [Gaston, 1997], [Anagnostopoulos, 1999] et [Maronnier, 2000]). Il s'agit d'une colonne d'eau qui s'effondre sous son propre poids dans une cavité en forme de rectangle (ou parallélépipède rectangle en 3D). En approximant le comportement de l'air par un fluide incompressible moins dense que l'eau, il est possible de considérer cette application comme une simulation d'écoulement de fluide newtonien hétérogène et instationnaire. Ainsi, le comportement de l'interface entre ces deux fluides approche celui d'une surface libre.

Cette application est considérée comme complexe car non seulement les mouvements d'interface sont importants, mais aussi, les écoulement sont potentiellement turbulents : le nombre de Reynolds peut localement atteindre jusqu'à 10 000.

Notre code de calcul CimLib et les méthodes numériques développées dans cette thèse sont utilisés ici. Brièvement, les équations de Navier Stokes qui tiennent compte des termes d'inertie sont généralement utilisées dans ce cas, et l'évolution des deux phases en présence (l'eau et l'air) est assurée par le transport d'une fonction Level Set qui permet de capturer les interfaces. Nous pourrons aussi comparer les résultats obtenus avec d'autres méthodes de calcul d'interface comme le Volume Of Fluid (VOF).

3.2. 2 Dimensions

La cavité est de dimensions rectangulaire (5×2.2) mètres, et le temps total de la simulation est 5 secondes. On choisis d'augmenter artificiellement la viscosité cinématique de l'eau, en la prenant égale à 10^{-3} m^2/s, ce qui correspond à une viscosité dynamique de 1 Pa.s et à une masse volumique de 1000 kg/m^3. L'air est approximé par un fluide newtonien incompressible de masse volumique de 1 kg/m^3. Les conditions aux limites sont de type glissant, c'est-à-dire que, sur toutes les surfaces de la géométrie, seule la composante tangentielle de la vitesse est imposée à zéro, l'autre composante constituant donc un degré de liberté. Initialement, la totalité du fluide présent dans la cavité se trouve sous forme de colonne de dimension (1×2) mètres.

Cette même application est exécutée en utilisant les deux méthodes de capture d'interface que l'on dispose dans le but de pouvoir les comparer.

3.2.1. Volume of Fluid

Dans un premier temps, on utilise la méthode Volume of Fluid décrite dans le chapitre des calculs d'interfaces. On y associe une technique de r-adaptation (adaptation dynamique du maillage) qui limite les effets néfastes de la diffusion numérique, et qui améliore la définition des interfaces.

Sur la figure (*14*), l'interface entre le fluide et l'air est montrée aux moments *t=0s*, *t=0s*, *t=1s*, *t=2s*, *t=3s*, et *t=4s*.

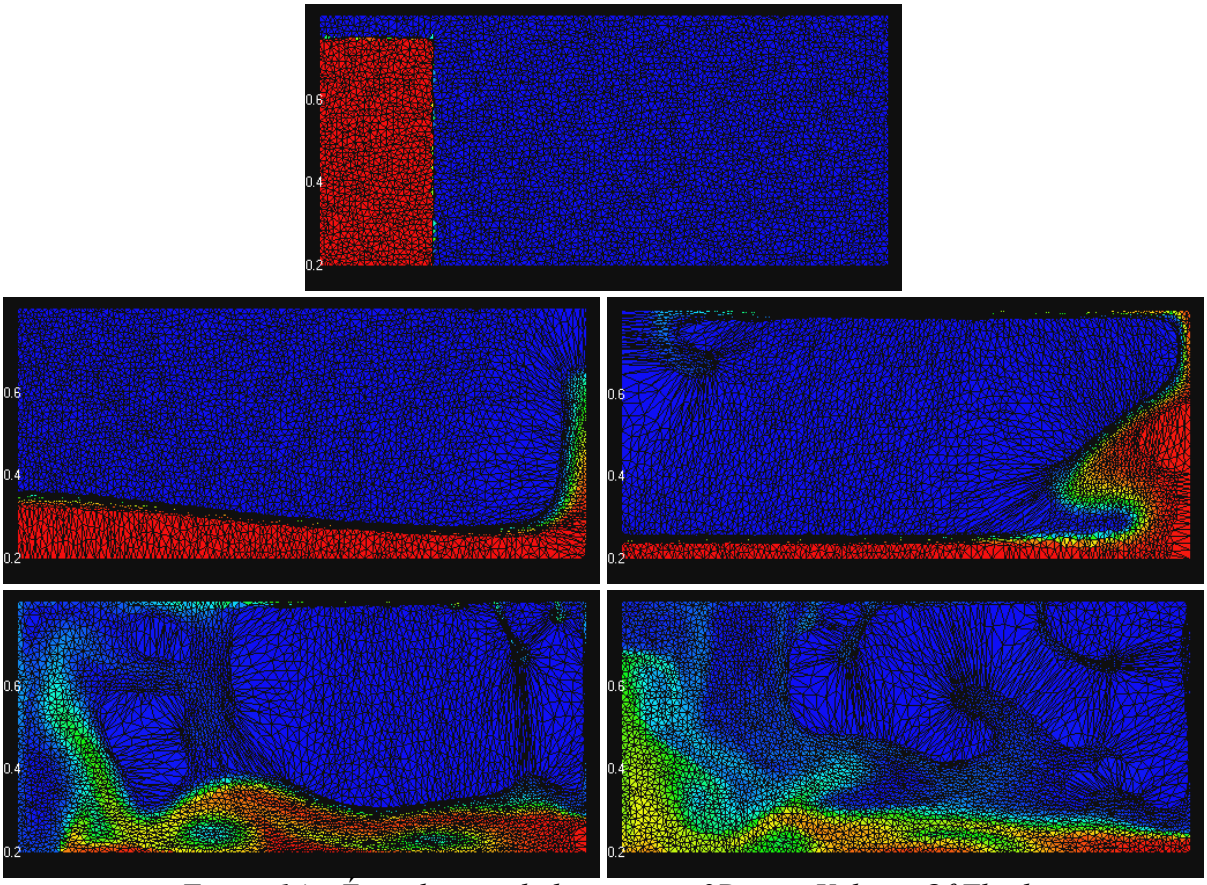


Figure 14 – Écroulement du barrage en 2D avec Volume Of Fluid

Dans la première partie de la simulation (jusqu'à environ *2.5s*), l'adaptation de maillage est clairement très performante, puisque seuls les éléments les plus rapprochés de l'interface sont raffinés. Mais ensuite, la longueur et la complexité de la simulation font que la qualité du raffinement du maillage se détériore considérablement, et c'est ainsi que l'on perd de la précision de façon considérable : après *1s*, il reste *99.5%* de la masse totale initialement présente dans la cavité ; ensuite, il reste *98%* après *2s*, *95.1%* après *3s* et *92%* après *4s*. Ces résultats deviennent de moins en moins bon tout au long de la simulation, et témoignent de la détérioration du raffinement de maillage, et donc de la diffusion numérique grandissante.

3.2.2. Level Set

Dans un deuxième temps, on utilise la méthode de type Level Set développée dans cette thèse. Ici, le domaine de calcul est maillé avec *35 000* nœuds et *70 000* éléments. La taille de maille correspondante est de *0.02 mètre*. Le temps total de la simulation est *5* secondes, et le pas de temps utilisé est *0.01* seconde. *500* incréments en temps sont donc nécessaires.

Une fonction Level Set α est initialisée telle que son niveau zéro corresponde au contour de la colonne d'eau à l'instant *t=0*. Ensuite, elle est transportée suivant la vitesse de l'écoulement hétérogène en utilisant la méthode Level Set décrite dans cette thèse. La figure (*15*) montre l'évolution de l'eau dans la cavité aux moments *t=0s*, *t=1s*, *t=2s*, *t=3s*, *t=4s* et

t=5s. L'eau y est représentée par la couleur rouge, et l'air par la couleur bleue. Les zones de mélange où coexistent les deux phases sont représentées par la couleur verte. Rappelons qu'avec notre méthode, elle ne dépasse pas l'épaisseur d'un élément.

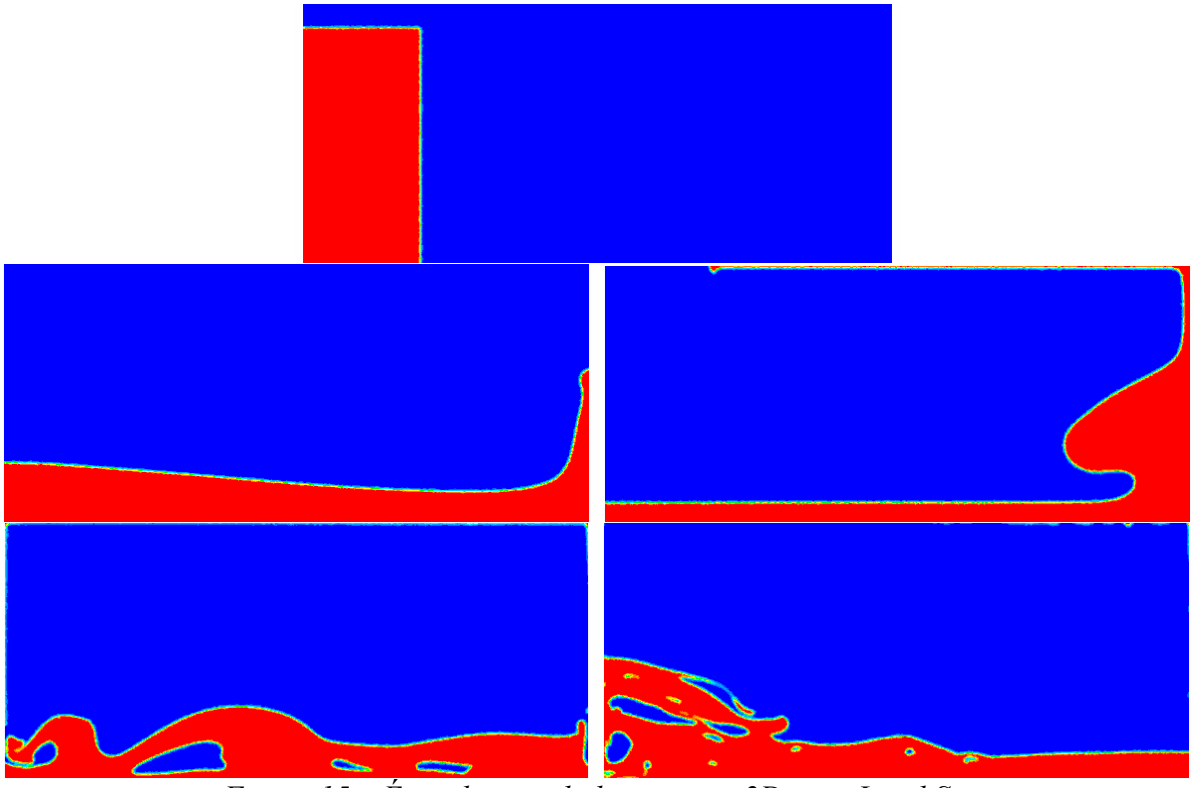


Figure 15 – Écroulement du barrage en 2D avec Level Set

Cette simulation exécutée sur 16 processeurs d'une grappe de PC a duré environ 2 heures. Comme nous l'avons vu dans le chapitre qui traite des calculs d'interfaces, la conservation de la masse est un des points faibles majeurs de la méthode Level Set. Nous vérifions donc ici les pertes de masse observées dans cette application : à la fin des *5* secondes de simulation, il reste environ *92%* de la masse totale initiale. Plus précisément, il reste *99%* après *1s*, *99 %* après *2s*, *97%* après *3s*, et *94.5%* après *4s*.

En comparant ces deux simulations qui, pour une même application, utilisent deux méthodes numériques distinctes, on s'aperçoit qu'elles donnent des résultats assez différents. Pourtant, il existe des points communs assez surprenants qui montrent la pertinence des deux techniques. Notamment, au moment *t=2.7s*, une goutte qui a traversé tout le plafond de la cavité retombe sur la parois de gauche avec une vitesse assez élevée, et chasse l'eau qui repose sur le sol.

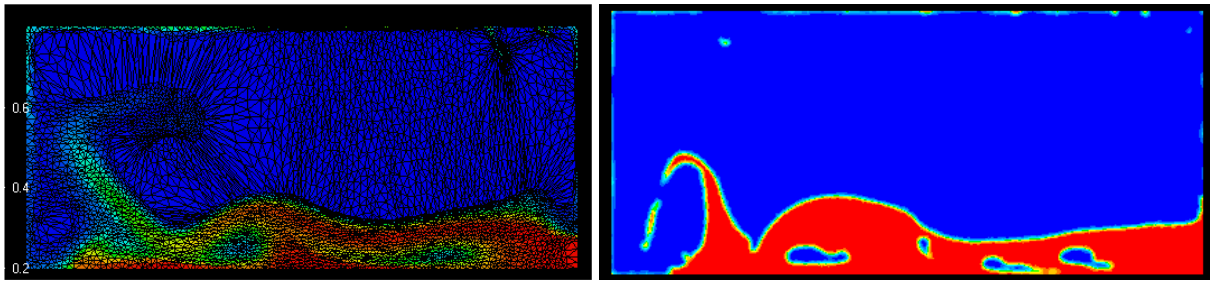


Figure 16 – Point commun entre le VOF et le Level Set

3.2.3. Résultats expérimentaux

Afin de valider nos résultats numériques, nous prenons comme référence les résultats expérimentaux obtenus dans [Martin, 1952]. Les auteurs ont enregistré le mouvement du fluide lors d'une expérience, et ont notamment noté la position de l'extrémité de la surface libre sur le plan horizontal. Ce sont ces donnés que l'on cherche à comparer ici avec nous résultats numériques. Pour cela, nous utilisons les données adimensionnelles suivantes :

$$T = nt\left(\frac{g}{a}\right)^{1/2} \qquad \text{et} \qquad Z = \frac{z}{a}$$

où T représente le temps et Z représente la position de l'extrémité de la surface libre sur le plan horizontal. a est la largeur initiale de la colonne de fluide (soit 1 mètre). n^2a est sa hauteur initiale (soit 2 mètres). z est la position de l'extrémité de la surface libre sur le plan horizontal au temps T.

Le graphe suivant montre Z en fonction de T pendant la première phase de la simulation où le fluide avance vers la droite de la cavité.

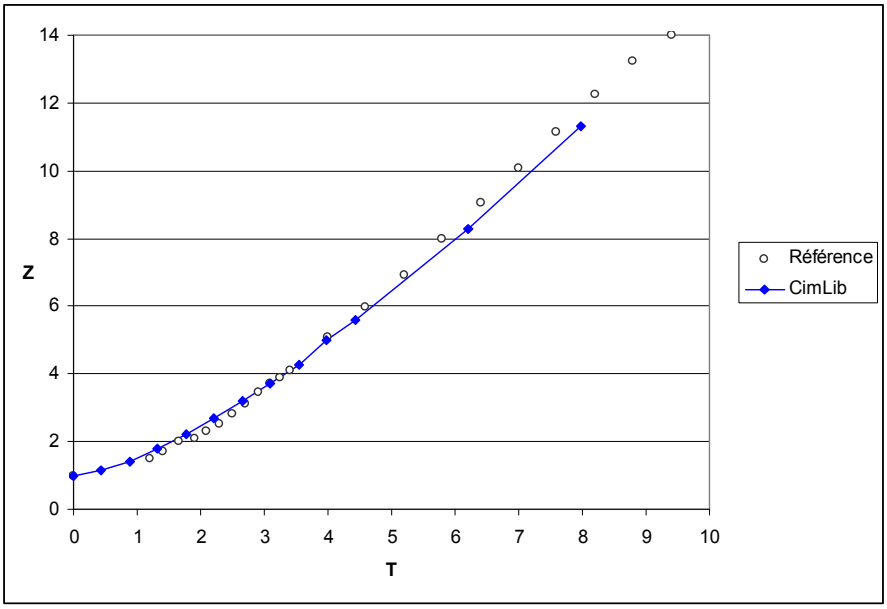


Figure 17 – Position de la surface libre

Sur la figure (*17*), nous observons que nos résultats numériques sont très proches des données expérimentales de référence [Martin, 1952]. Elle montre cependant qu'il y a un léger retard dans notre simulation qui s'accentue avec le temps, ce qui est probablement d'effet d'une légère perte de volume.

3.3. 3 Dimensions

L'extension de cette application en 3 dimensions se fait naturellement avec la librairie CimLib. Il suffit de mailler une cavité 3D à base de tétraèdres. Pour cela, nous choisissons un domaine de forme parallélépipède rectangle de dimension (*1.5 × 1.0 × 2.0)* mètres. Nous utilisons ici un maillage à *2 150 000* nœuds et *12 500 000* éléments. Le temps total de la

simulation est de 3 secondes, et le pas de temps utilisé est de *0.005* seconde. *600* incréments en temps sont donc nécessaires. Les paramètres utilisés sont les suivants : le fluide a une viscosité de *5 Pa.s* et une masse volumique de *1000 kg/m*^{3}, l'air est approximé par un fluide newtonien incompressible de masse volumique de *1 kg/m*^{3}. Les conditions aux limites sont de type glissant. Deux plans de symétrie permettent d'étendre la simulation à une cavité plus grande, sans pour autant alourdir les calculs.

Initialement, le fluide se trouve dans une colonne de dimension (*0.6 × 0.6 × 1.8*) contre les deux plans de symétrie de la cavité. Au temps *t=0*, elle s'effondre sous son propre poids, puis le fluide s'écoule librement dans la « bassine » que représente le domaine de calcul. Les figures suivantes montrent l'interface entre l'eau et l'air aux moments *t=0s*, *t=0.6s*, *t=1.2s*, *t=1.8s*, *t=2.4s* et *t=3s* dans le cas du Volume of Fluid (*figure 18*) et du Level Set (*figure 19*).

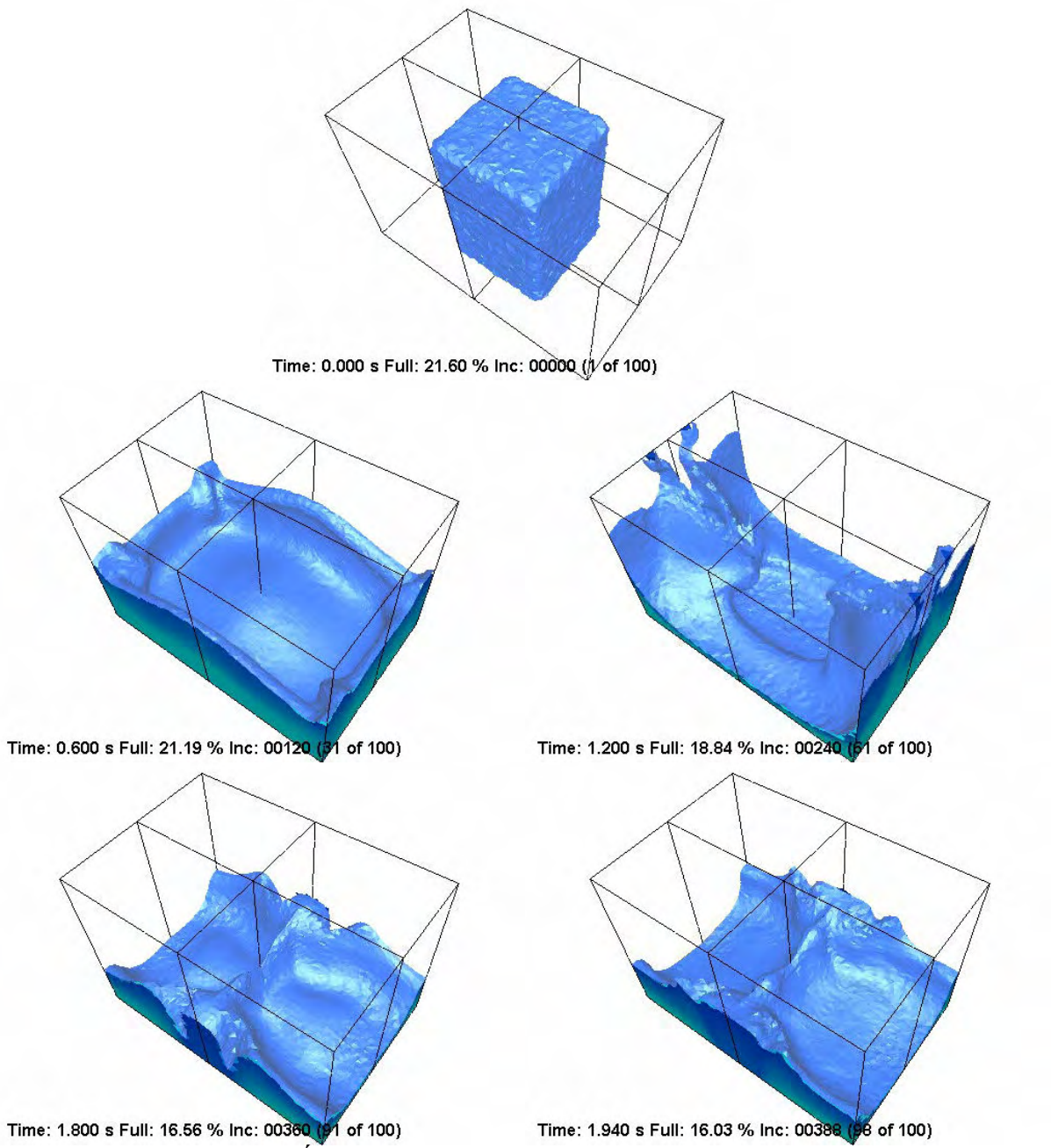


Figure 18 – Écroulement du barrage en 3D avec Volume of Fluid

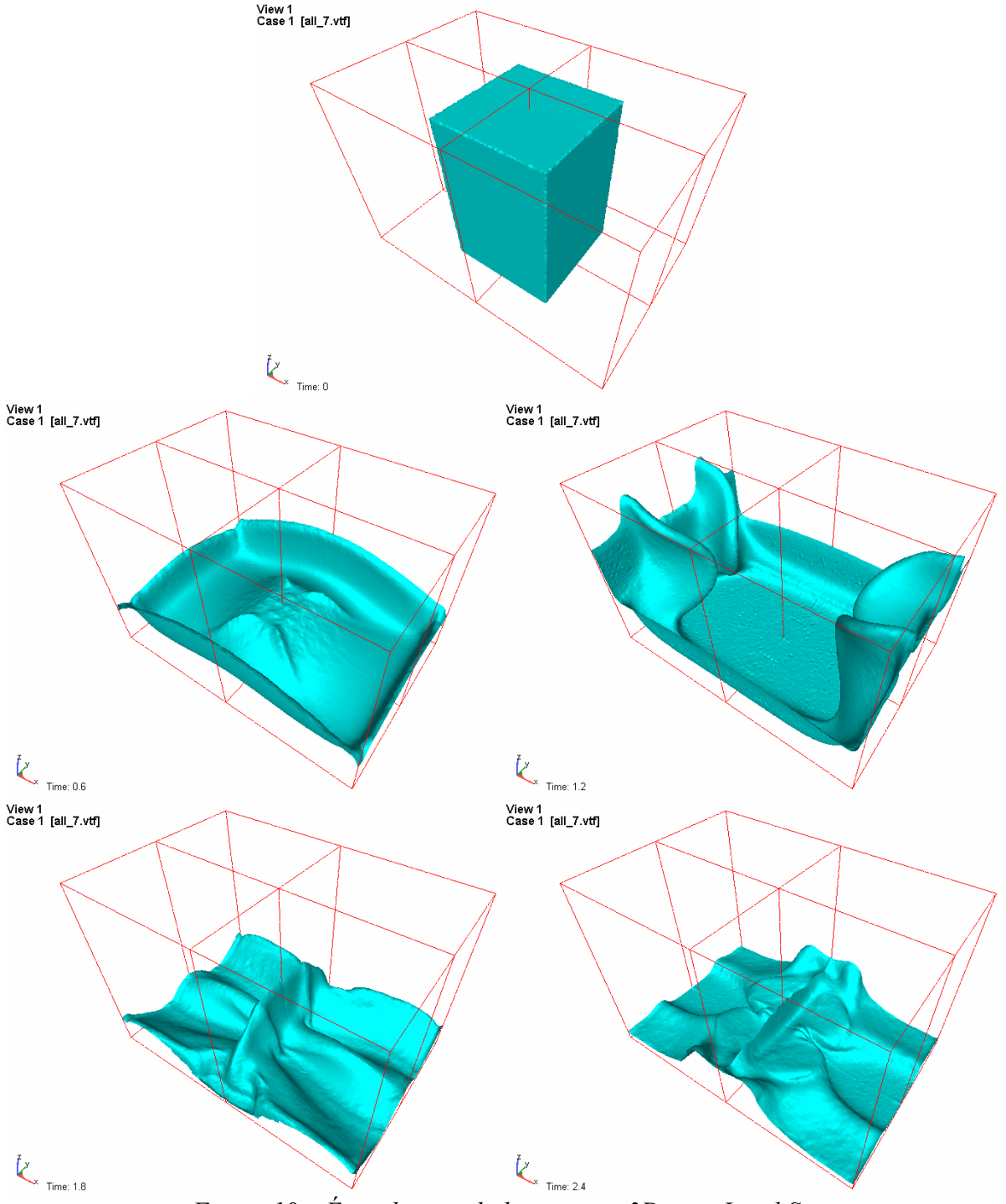


Figure 19 – Écroulement du barrage en 3D avec Level Set

Cette application a été exécutée en parallèle sur 32 processeurs d'une ferme de PC, et dura environ 5 jours et 1 heure. Pour une telle application, nous obtenons une efficacité parallèle très proche de 1 sur notre ferme de PC.

4. Chute d'une bille dans un fluide

Le but est ici de comparer des résultats expérimentaux avec nos résultats numériques dans le cadre d'une application de chute d'une bille dans un fluide.

4.1. Dispositif expérimental

La première étape de cette étude est de réaliser un dispositif expérimental permettant de mettre en évidence le jet provoqué par la chute d'une bille dans un liquide appelé le jet de Worthington.

Le rapport de stage de Morgane Pascal [Pascal, 2005] réalisé au CEMEF traite de cette expérience. Elle a reproduit le dispositif expérimental étudié dans [Cheny, 1996], [Cheny, 1999] et [Nigen, 2001] pour visualiser le jet de Worthington produit par la chute d'une bille dans une bassine de fluide. Conformément à [Nigen, 2001], ce bac doit être assez grand pour que le jet ne soit pas influencé par les bords ou par la profondeur du liquide (*40×40×20 cm*). Un dispositif permet de lancer une bille à différentes hauteurs avec une vitesse initiale nulle. Enfin, une caméra rapide filme l'expérience en capturant *1000* images par seconde.

Le fluide contenu dans le bac est du sirop de glucose obtenu par dissolution de sucre dans de l'eau. Le sirop utilisé contient *70%* de sucre en masse et a une viscosité d'environ *0.2 Pa.s*. Compte tenu du caractère exponentiel de la loi d'évolution de la viscosité en fonction de la masse de sucre, il existe une marge d'erreur assez importante sur la viscosité expérimentale du fluide. De plus, le lâché manuel de la bille entraîne une approximation assez importante sur la chute libre, avant qu'elle ne rencontre le fluide. Ainsi, il sera difficile d'exploiter les résultats de cette expérience de façon très rigoureuse. Toutefois, une comparaison qualitative avec des résultats numériques reste possible.

La figure (*20*) montre la chute d'une bille d'acier (avec une masse volumique de *7800* $kg.m^{-3}$) de *1 cm* de diamètre lancée d'une hauteur de *10 cm* par rapport à la surface du fluide.

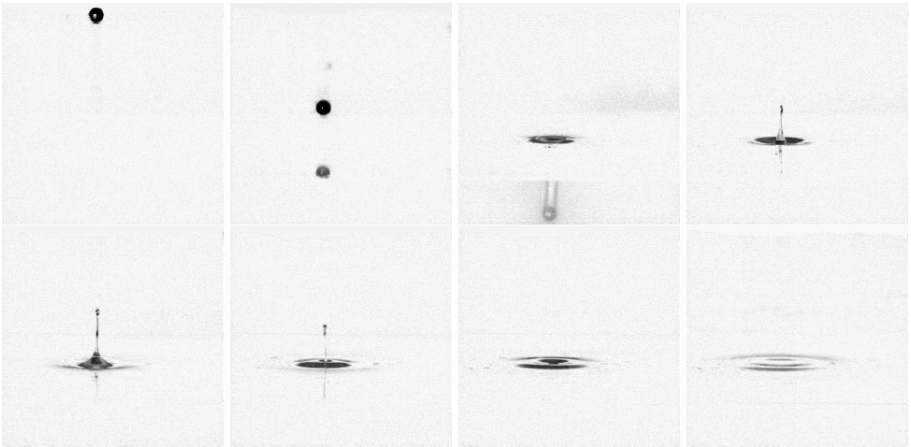


Figure 20 – Chute d'une bille en acier

La figure (*21*) montre une bulle capturée lors de l'impact de la bille en acier qui remonte à la surface du fluide.

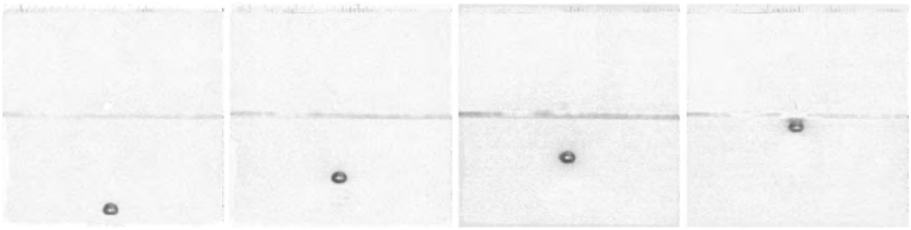


Figure 21 – Bulle qui remonte à la surface après l'impact

Nous renouvelons maintenant cette expérience avec une bille de même taille mais d'un autre matériau pour pouvoir observer l'influence de la masse volumique de la bille sur l'impacte qu'elle a avec le fluide. La figure (*22*) compare la hauteur maximale du jet de Worthington entre *2* billes de *1 cm* de diamètre lancée d'une hauteur de *10 cm* ; la première étant en acier (masse volumique de *7800 kg.m*$^{-3}$), et la deuxième étant en verre (masse volumique de *2700 kg.m*$^{-3}$).

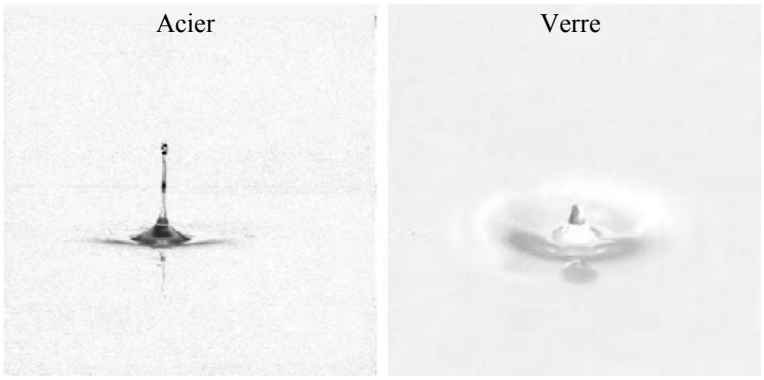


Figure 22 – Hauteur maximale du jet pour les billes en acier et en verre

La hauteur du jet est d'environ *6 cm* dans le cas de la bille en acier contre *1.5 cm* pour la bille en verre. La figure (*23*) compare la profondeur maximale du « cratère » formé par les billes en acier et en verre au moment de l'impacte avec le fluide.

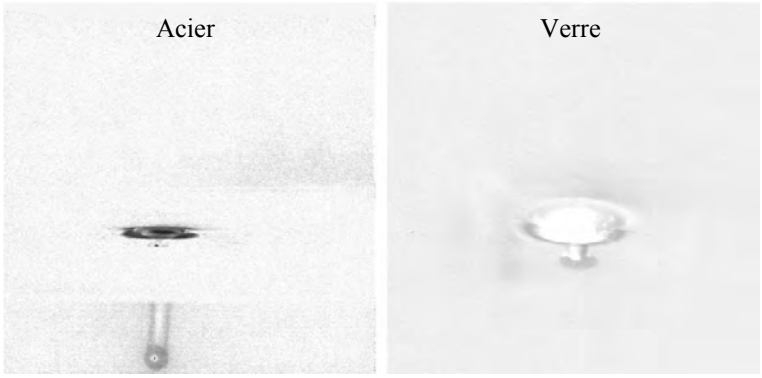


Figure 23 – Profondeur maximale de l'impacte pour les billes en acier et en verre

La profondeur maximale du cratère est de l'ordre de *7 cm* dans le cas de la bille en acier contre *2 cm* pour la bille en verre.

Nous pouvons maintenant chercher à comparer ces résultats expérimentaux avec ceux obtenus numériquement.

4.2. Résultats numériques

Nous créons ici une application numérique conforme à l'expérience décrite précédemment. Ainsi, une bille d'acier (puis une autre en verre) est lâchée d'une hauteur de *10 cm* au dessus de la surface d'un fluide contenu dans une bassine. Afin de pouvoir rentrer dans le cadre de cette thèse (c'est-à-dire des simulations numériques d'écoulement de fluides hétérogènes), nous considérons les trois domaines en présence comme des fluides incompressibles. La bille est représentée comme une goutte de *1 cm* de diamètre d'un fluide très visqueux (*2000 Pa.s* pour les calculs) et ayant une masse volumique de *7800 kg.m*$^{-3}$ pour l'acier et *2700 kg.m*$^{-3}$ pour le verre. L'air est représenté comme un fluide incompressible de masse volumique *1 kg.m*$^{-3}$. Enfin, le fluide a une viscosité de *0.22 Pa.s* et une masse volumique de *1000 kg.m*$^{-3}$.

4.2.1. 2 Dimensions

Tout d'abord, nous créons un maillage 2D contenant 7000 nœuds et 14000 éléments. Avec un axe de symétrie sur *x=0*, nous réalisons les calculs sur la moitié du domaine pour obtenir une cavité totale de *(20×18) cm* où le récipient a une profondeur de *6 cm*. Afin d'alléger la masse de calcul, la taille de maille est raffinée dans les zones où une interface est susceptible d'aller (c'est-à-dire sur la trajectoire de la bille, et autour de la surface du fluide), et déraffinée partout ailleurs. Initialement, la bille (ou plutôt, la goutte) est placée sur *(x=0,y=16)*, soit *10 cm* au dessus de la surface du fluide. A *t=0*, elle commence sa chute libre, jusqu'à l'impact avec le fluide. Là, nous pouvons observer et comparer les grandeurs que nous avons mesurées expérimentalement, soit la hauteur du jet et la profondeur du cratère.

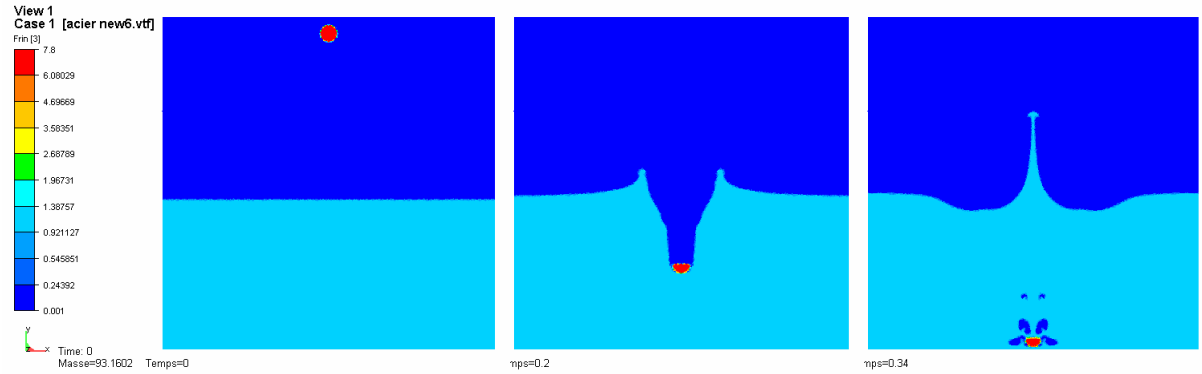


Figure 24 – Chute d'une bille d'acier

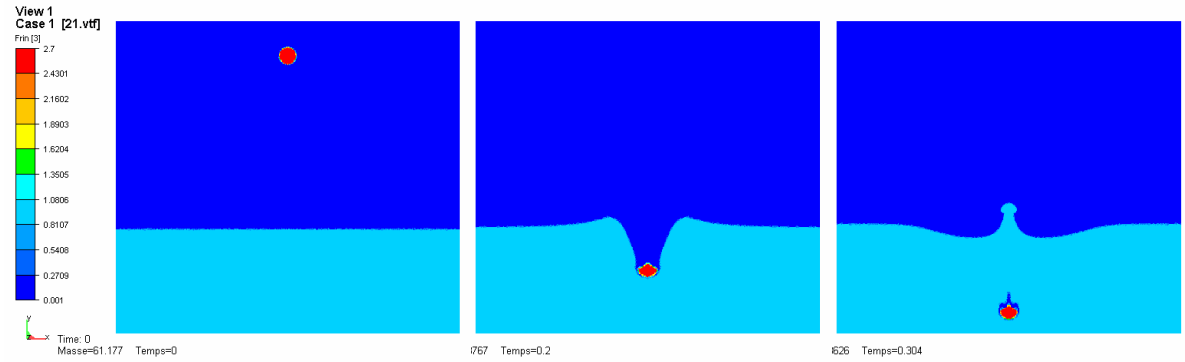


Figure 25 – Chute d'une bille de verre

Dans le cas de la bille d'acier (avec une masse volumique de *7800 kg.m*$^{-3}$), la hauteur du jet est de *5.5 cm*, contre *6 cm* mesuré lors de l'expérience ; et la profondeur maximale du cratère formé par l'impact est elle de *6 cm* contre *7 cm* expérimentalement. Dans le cas de la bille en verre (avec une masse volumique de *2700 kg.m*$^{-3}$), la hauteur du jet est de *1.5 cm*, la même valeur que celle mesurée lors de l'expérience ; et la profondeur maximale du cratère formé par l'impact est elle de *2 cm*, conformément à l'expérimentalement.

4.2.2. 3 Dimensions

Les mêmes calculs sont maintenant effectués en 3 dimensions avec un maillage 3D contenant *150 000* nœuds et *850 000* éléments. La cavité représentée par le domaine de calcul est de taille *10×10×15 cm*. Elle n'est pas tout à fait aussi grande qu'elle devrait être pour reproduire exactement l'expérience, mais elle est assez grande pour que les effets de bords n'influent pas trop sur l'impact de la bille dans le fluide. Avec les mêmes paramètres qu'en 2D, nous obtenons les résultats illustrés dans la figure suivante, où les surfaces de la bille d'acier et de l'eau sont représentées aux instants *t=0*, *t=0.16*, *t=0.30*, *t=0.40* et *t=0.47* secondes. Dans le cas de la bille de verre, la figure (*26*) montre les résultats de la simulation aux instants *t=0*, *t=0.16*, *t=0.30*, *t=0.37* et *t=0.52* secondes.

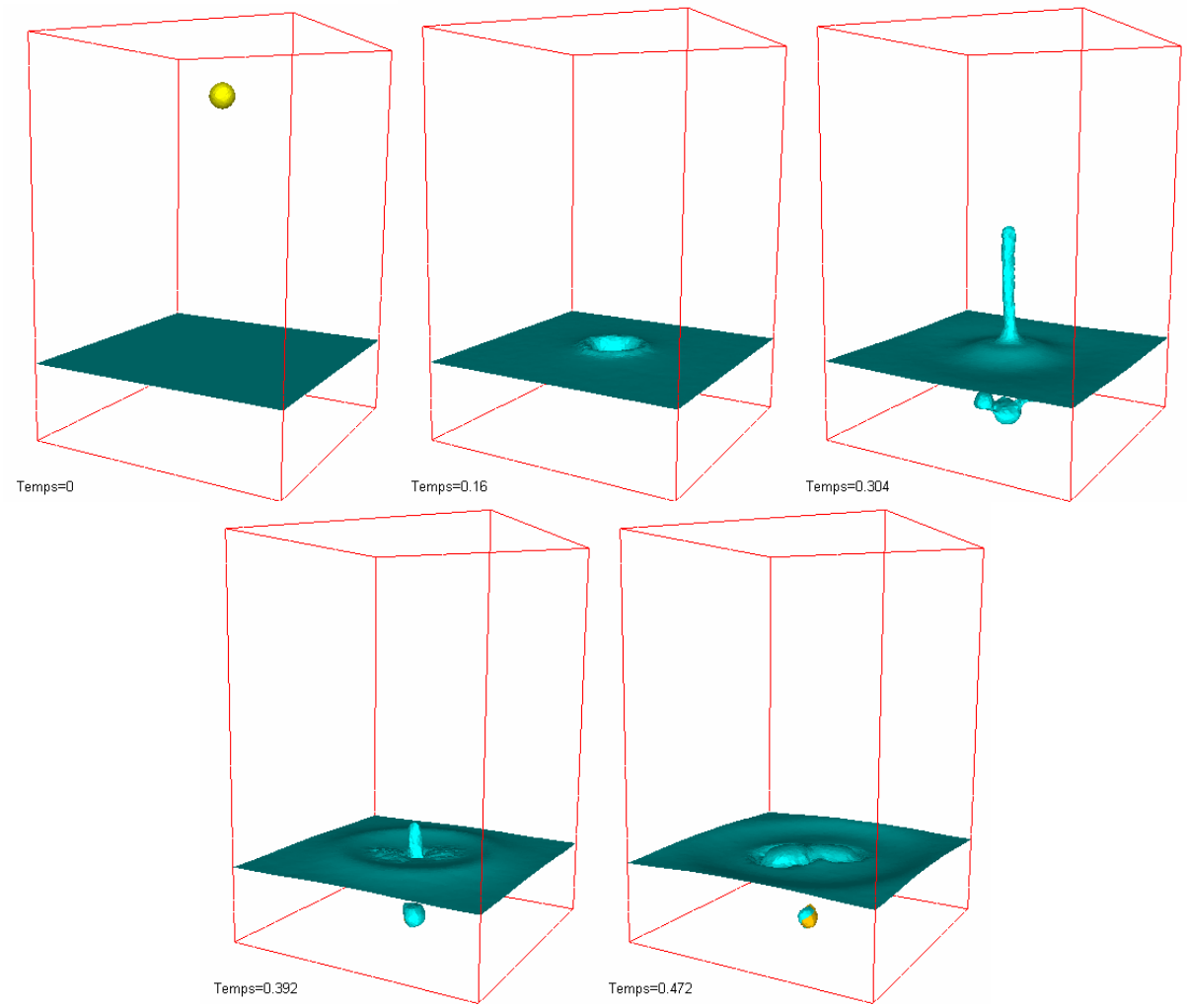


Figure 26 – Chute d'une bille d'acier

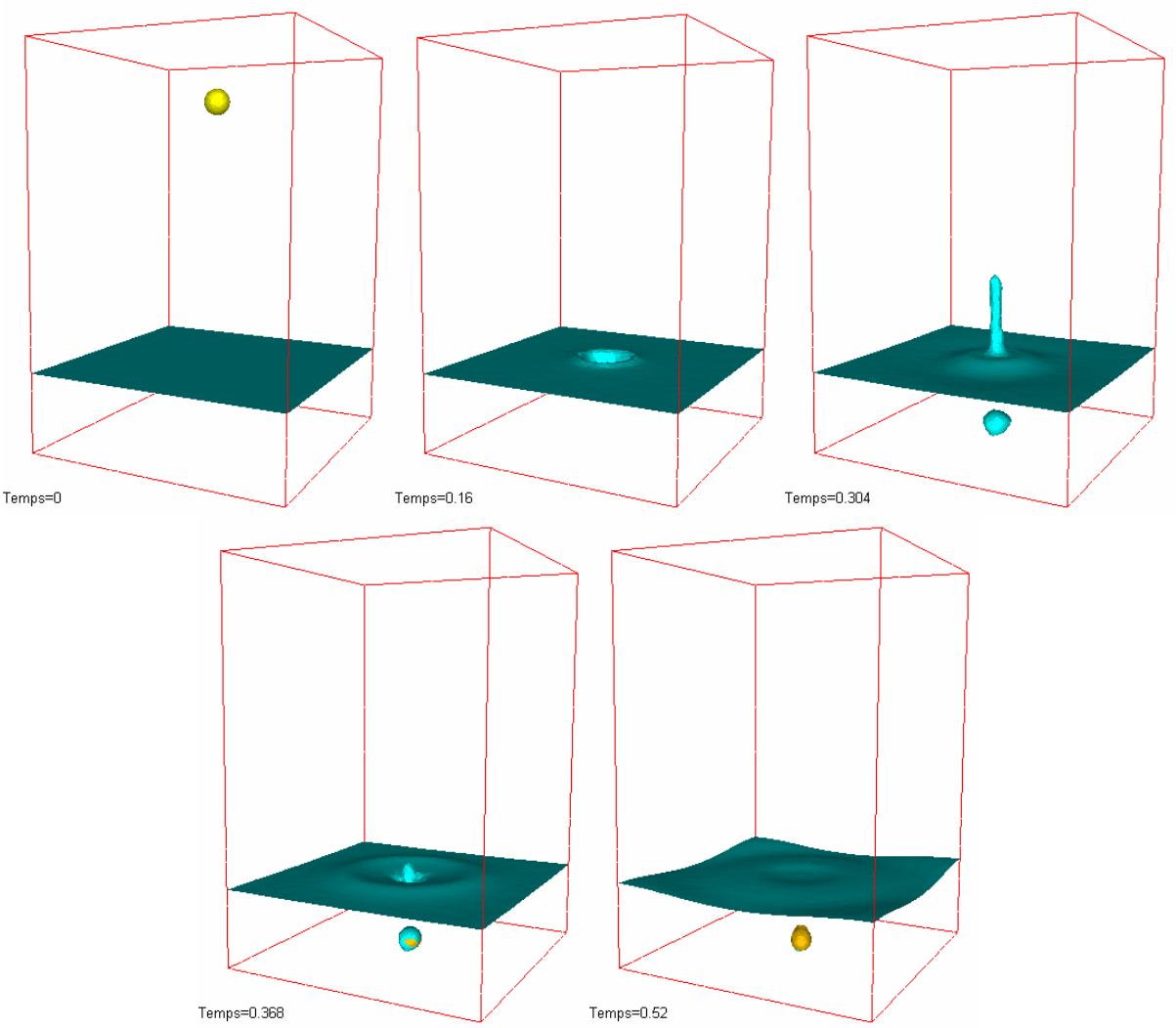


Figure 27 – Chute d'une bille de verre

Dans le cas de la bille d'acier, la hauteur du jet est de *4.9 cm*, contre *6 cm* mesuré lors de l'expérience ; et dans le cas de la bille en verre, la hauteur du jet est de *2.2 cm*, contre *1.5 cm* mesuré expérimentalement.

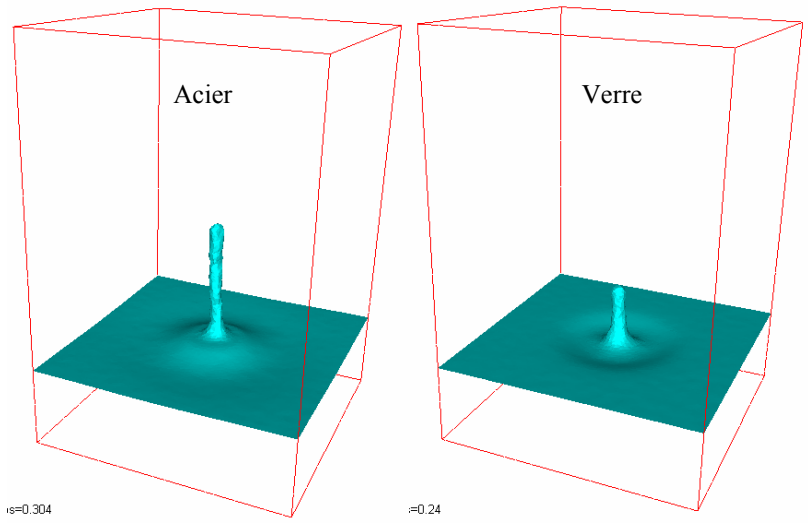


Figure 28 – Jet de Worthington pour les billes en acier et en verre

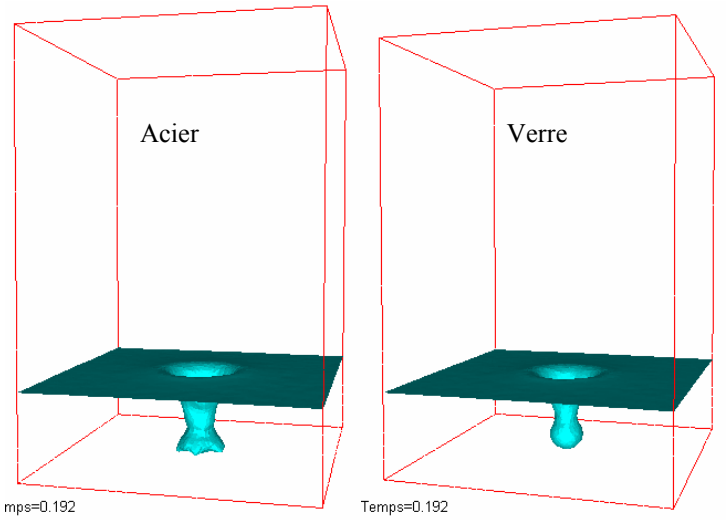


Figure 29 – Cratère de l'impacte pour les billes en acier et en verre

Nous observons ici que jets de Worthington sont bien reconnaissables par rapport à l'expérience, mais que les cratères formés par l'impact des billes ne le sont pas vraiment. En effet, la profondeur de la cavité utilisée pour la simulation numérique n'est visiblement pas assez profonde pour permettre aux cratères de se former sans que la parois n'intervienne : les billes cognent sur le sol avant même la formation entière des cratères.

Dans le cas de la bille en acier, nous voyons clairement la formation de bulles d'air qui se trouvent emprisonnées dans le cratère. Celle-ci remontent ensuite à la surface et forment un remout.

.

5. Conclusion

Dans ce dernier chapitre, des applications diverses en 2 et 3 dimensions sont présentées afin de valider les méthodes numériques développées (et notamment grâce à des résultats expérimentaux de référence) ; et aussi, de montrer des exemples variés qui témoignent de la diversité des applications possibles de nos modèles de simulation. De plus, l'implémentation en parallèle de notre code de calcul permet l'utilisation intensive de machines parallèles telles que des clusters et des grilles de calcul. C'est ainsi que des applications spécifiques aux grilles de calcul sont étudiées dans le chapitre qui traite du projet MecaGrid.

Même si certaines applications présentées ici demandent une étude plus profonde et une meilleure approche des conditions expérimentales, les résultats obtenus semblent encourageants.

Ainsi, la modélisation d'écoulements de fluides hétérogènes par couplage d'un solveur de Navier Stokes avec un solveur d'interface de type Level Set est justifiée, non seulement par la validation rigoureuse de ses deux solveurs séparément, mais aussi par la mise en place d'applications permettant d'évaluer leur couplage.

Références

[Anagnostopoulos, 1999] J. Anagnostopoulos, G.Bergeles. "Three-Dimensional Modeling of the Flow and the Interface Surface in a Continuous Casting Mold Model", *Metallurgical and materials transaction* N, 30B, 1095-1105, (1999).

[Cheny, 1996] J-M Cheny, and K. Walters, "Extravagant viscoelastic effects in the Worthington jet experiment", *Journal of Non-Newtonian Fluid Mechanics*, vol. 67, pp. 125-135 (1996).

[Cheny, 1999] J-M Cheny and K. Walters, "Rheological influences on the splashing experiment", *Journal of Non-Newtonian Fluid Mechanics*, vol. 86, pp. 185-210 (1999).

[Cocchi, 1997] J. P. Cocchi, and R. Saurel, "A Riemann problem based method for the resolution of compressible multimaterial flows", *Journal of Computational Physics*, 137-2, 265–298 (1997).

[Gaston, 1997] L. Gaston. "Simulation numérique par éléments finis bidimensionnels du remplissage de moule de fonderie et étude expérimentale sur maquette hydraulique", *Thèse de doctorat*, Ecole Nationale Supérieure des Mines de Paris, (1997).

[Helluy, 2005] P. Helluy, F. Golay, J.P. Caltagirone, P. Lubin, S. Vincent, D. Drevard, R. Marcer, P. Fraunié, N. Seguin, S. Grilli, A.C. Lesage, A. Dervieux, and O. Allain, "Numerical simulations of wave breaking", *Mathematical Modelling and Numerical Analysis*, 39-3, 591-607 (2005).

[Maronnier, 2000] V. Maronnier. "Simulation numérique d'écoulements de fluides incompressibles avec surface libre", *Thèse de doctorat*, Ecole polytechnique Fédérale de Lausanne (2000).

[Martin, 1952] J-C. Martin et M.J. Moyce. "Some gravity wave problems in the motion of perfect liquids", *Philos. Trans. Roy. Soc. London Ser*. A, 244, 231-334 (1952).

[Nigen, 2001] S. Nigen and K. Walters, "On the two-dimensionnal splashing experiment for Newtonian and slightly elastic liquids", *Journal of Non-Newtonian Fluid Mechanics*, vol. 97, pp. 233-250 (2001).

[Pascal, 2005] M. Pascal, "Interaction fluide-structure. Chute d'un solide dans un liquide", Rapport de stage, CEMEF (2005).

[Tanaka, 1986] M. Tanaka, "The stability of solitary waves", *Phys. Fluids*, 29, 650-655 (1986).

[Yasuda, 1997] T. Yasuda, H. Mutsuda, and N. Mizutani, "Kinematic of overtuning solitary waves and their relations to breaker types", *Coastal Engineering*, 29, 317-346 (1997).

CONCLUSION GENERALE

Nous avons implémenté un code de calcul parallèle qui permet de traiter des applications de grande taille dans le domaine de la simulation d'écoulements de fluides hétérogènes sur une grille de calcul.

Le projet MecaGrid

Il a fallu dans un premier temps construire une grille de calcul en reliant quatre clusters de trois centres de recherche indépendants grâce au middleware Globus [Foster, 1997], et l'étudier ensuite afin de mieux comprendre comment bien utiliser cet outil de calcul si particulier. Au-delà des difficultés techniques et administratives dues au regroupement de grappes de PC (ou clusters) géographiquement dispersées, les premiers résultats ont montré que la conception même de la grille engrange des problèmes de performance, surtout au niveau des communications de données inter sites. Ces connections internet se montrent extrêmement plus lentes que les communications au sein même d'un site géographique. De plus, l'importante diversité des caractéristiques de chaque cluster (en terme de puissance de processeur, de mémoire vive ou de réseau par exemple), résulte en une grande hétérogénéité globale, qui fait de la grille de calcul un outil très différent de celui d'un cluster classique beaucoup plus homogène. Et cela a bien sûr un impact important sur la manière de l'employer.

Nous avons donc mis en place plusieurs techniques d'optimisation qui visent à rendre envisageable l'exécution de lourdes applications, tant au point de vue technique que des performances. Notamment, nous avons montré qu'un bon partitionnement de maillage [Basermann, 2000], qui répartit la masse de calcul et les communications, s'avère cruciale dans ce genre de projet afin de s'adapter le mieux possible aux caractéristiques de la grille.

Au final, nous avons élaboré un outil de calcul très hétérogène dans sa conception, avec des processeurs et des réseaux très différents, et qui peut offrir une grande quantité de ressources de calculs à moindre coût. Les différentes études ont permis d'acquérir une connaissance approfondie de cette infrastructure et de définir avec succès une utilisation performante qui favorise sa mise en valeur avec des calculs scientifiques appliqués à la mécanique des fluides.

Les Perspectives

Une première perspective envisageable se trouve au niveau des ressources de calcul de la grille. Sa conception initiale permet en effet de la faire évoluer, et il est donc possible de rassembler un plus grand nombre de clusters afin d'obtenir une grille plus importante.

Il est ensuite très important d'améliorer les connexions internet entre les sites géographiques qui sont actuellement très pénalisantes, certaines communications étant 100 fois plus lentes que celles observées au sein d'un cluster. Avec l'amélioration rapide et importante que voit le développement d'internet, cela semble réalisable d'en bénéficier ici.

Enfin, notre outil de partitionnement optimisé, qui répartit la masse de calcul et les communications en fonction des caractéristiques de la machine, pourrait être amélioré de manière à être plus facile à utiliser, et plus adaptatif. Ainsi, les calculs réalisés lors d'une simulation numérique pourraient être instrumentés de manière à monitorer les performances exactes des ressources de calculs qui sont potentiellement variables dans le temps, et en

particulier dans le cas de longues simulations. En effet, ces performances sont extrêmement dépendantes des clusters utilisés, du nombre de processeurs par cluster, et de l'activité générale de la grille de calcul.

Les méthodes numériques

Dans le cadre de la résolution directe des équations de Navier-Stokes pour des écoulements incompressibles avec capture d'interface, nous avons développé, étudié et comparé plusieurs méthodes numériques. L'implémentation de ces solveurs en C++ dans CimLib est entièrement parallélisée avec la librairie MPI, et des algorithmes optimisés de l'outil de calcul PETSc sont utilisés pour préconditionner et résoudre les systèmes linéaires.

Nous avons, dans un premier temps, présenté une méthode de résolution directe du problème de Navier-Stokes incompressible basée sur les éléments finis mixtes stabilisés par des fonctions bulle [Coupez, 1997] qui rappellent les techniques de type *multiscale* [Hughes, 1998]. Un schéma temporel d'Euler implicite est utilisé, en association avec un algorithme de Newton pour linéariser le problème. Les validations numériques, réalisées au travers du cas test de la cavité entraînée [Ghia, 1982], montrent que nos méthodes numériques permettent de simuler efficacement des écoulements complexes de fluides newtoniens aux nombres de Reynolds allant au-delà de 20000, sans pour autant avoir à traiter un modèle de turbulence généralement utilisé pour simuler des écoulements à hauts Reynolds.

Dans un deuxième temps, nous nous sommes intéressés aux techniques de capture d'interface, et au développement d'un solveur basé sur une approche *Level Set* et une interpolation spatiale continue (P1). Nous avons présenté une méthode éléments finis stabilisés par *SUPG* ou *Residual-Free Bubbles* [Brezzi, 1997] pour résoudre une équation de transport (équation de convection pure). Ensuite, nous avons comparé cette approche avec une méthode de type Galerkin discontinue proche du *Volume of Fluid* qui a été développée précédemment dans notre laboratoire [Coupez, 2000]. Ainsi, les grands points forts de la méthode *VOF* se situent dans sa très bonne conservation et sa robustesse. Par contre, elle cause un problème important de diffusion numérique, qui peut néanmoins être limité par une technique d'adaptation dynamique du maillage. La méthode *Level Set* présentée ici a l'avantage d'être moins lourde et de résoudre le problème de diffusion numérique de l'interface. Cependant, quelques problèmes ont été mis en évidence par l'étude de cas test. Nous avons d'abord souligné la nécessité d'utiliser un algorithme de réinitialisation [Sussman, 1994] qui assure de garder de bonnes conditions de transport de l'interface ; puis, nous avons mis en évidence un manque de conservation, propre aux méthodes *Level Set*.

Après les validations du solveur Navier-Stokes incompressible et de la méthode *Level Set*, le couplage des deux méthodes a été développé, basé sur une loi de mélange qui permet la définition précise des interfaces, sans diffusion numérique. Plusieurs applications permettent de juger et de comparer l'efficacité et la robustesse de ces méthodes. Dans les écoulements étudiés, les grandes déformations ont bien été gérées, et les pertes de masses dues au problème de conservation sont restées minimes, voire négligeables. Des applications de grande taille avec maillage fin ont pu être réalisées sur les ressources de calcul disponibles à travers nos clusters et la grille de calcul de MecaGrid.

Les Perspectives

Des améliorations restent à effectuer, et notamment au niveau de la modélisation numérique de l'approche *Level Set*. En effet, son manque de conservation constitue son principal désavantage, et cela reste encore aujourd'hui un problème ouvert et particulièrement difficile à résoudre. Après une brève revue des études existantes qui permettent d'améliorer cet inconvénient, il semble qu'un couplage avec une méthode Lagrangienne [Enright, 2005] ou *Volume of Fluid* [Sussman, 2000] peut conduire à une méthode plus conservative, car ces approches sont précisément basée sur le principe de conservation de la masse. Si le couplage avec une méthode Lagrangienne parait lourd à mettre en place (et surtout en 3 dimensions), celui avec une méthode *VOF* reste à être étudié plus précisément afin d'en déterminer la pertinence.

Une autre perspective d'optimisation du transport d'interface par *Level Set* est envisageable lorsque l'on considère un couplage fort entre l'équation de transport et l'algorithme de réinitialisation. Il s'agit ici de calculer le transport de l'interface et de réinitialiser la fonction *Level Set* dans un seul et même système. Cette technique innovante introduite dans [Coupez, 2006] part de l'observation que seule la vitesse du phénomène physique autour de l'interface est importante. En effet, l'information que l'on peut retirer de la fonction *Level Set* pour localiser l'interface est son niveau zéro. Partout ailleurs dans le domaine de calcul, les valeurs de cette fonction ne correspondent à aucune quantité physique. Cette technique permettrait d'optimiser les performances du solveur de capture d'interface, et d'alléger les calculs.