UNIVERSITÉ DE MONTRÉAL

UNE ÉTUDE DE CAS DE VÉRIFICATION POUR UN CODE À SOURCE OUVERT DE SIMULATION NUMÉRIQUE

HAROLD NORIEGA
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INFORMATIQUE)
DÉCEMBRE 2016

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

UNE ÉTUDE DE CAS DE VÉRIFICATION POUR UN CODE À SOURCE OUVERT DE SIMULATION NUMÉRIQUE

présentée par : NORIEGA Harold
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. ANTONIOL Giuliano, Ph. D., président
M. GUIBAULT François, Ph. D., membre et directeur de recherche
M. REGGIO Marcelo, Ph. D., membre et codirecteur de recherche
M. CAMARERO Ricardo, Ph. D., membre
M. BLAIS Bruno, Ph. D., membre externe

DÉDICACE

À Olga, à Valentin et à Angelica,
À ceux qui m'ont soutenu dans la distance...

REMERCIEMENTS

Je tiens à remercier mon directeur de recherche, François Guibault qui a été mon mentor dans mon processus d'apprentissage, je le remercie aussi de m'avoir donné la liberté d'entreprendre ma recherche dans la direction que je souhaitais. Mes gratitudes s'étendent aussi à son appui économique, qui a permis de conclure ce projet. Je tiens à remercier aussi mon codirecteur de recherche Marcelo Reggio qui m'a aidé à mettre au point des aspects techniques, ainsi que pour sa grande collaboration dans les aspects linguistiques et de rédaction des deux articles.

Je tiens à remercier l'Institut de recherche d'Hydro-Québec pour sa contribution financière dans ce projet d'étude et plus particulièrement Paul Labbé et Robert Magnan qui m'ont indiqué le point de départ de cette recherche.

J'ai une profonde reconnaissance pour le travail de Marie-Gabrielle Vallet, merci d'avoir corrigé mon français dans le cadre de cette thèse.

Je tiens à remercier à Christophe Devals et Julien Dompierre qui avaient la patience d'écouter mes idées dans les laboratoires.

Enfin, je remercie Olga, Valentin et Angelica pour le soutien sans relâche toutes ces années.

RÉSUMÉ

Ces dernières décennies, les calculs numériques en mécanique des fluides, à l'aide d'outils de calcul tels que OpenFOAM®, se sont considérablement développés, grâce aux ressources informatiques qui sont de plus en plus puissantes et qui permettent ainsi la résolution numérique des équations aux dérivées partielles avec une grande précision. Cependant, malgré les grands progrès, il reste un défi majeur : la simulation des problèmes réels complexes. Pour la plupart de ces problèmes, il n'existe pas de solution analytique et la validation et la vérification des résultats deviennent une tâche difficile, mais nécessaire. OpenFOAM® est un outil de simulation principalement orienté sur la résolution numérique des équations de la mécanique des fluides. Distribué sous licence à source ouverte GNU/GPL, OpenFOAM® est largement utilisé dans les milieux universitaires et dans l'industrie. Bien que ce logiciel est composé d'un grand nombre de bibliothèques qui prennent en charge les différentes phases de la simulation numérique, peu d'études de vérification peuvent être trouvées aujourd'hui. Une étude de vérification en profondeur de ses librairies principales est justifiable. D'autant plus qu'on a observé une perte de convergence lors des simulations numériques de solutions analytiques des équations de Navier-Stokes sur des maillages non orthogonaux.

Cette thèse est composée principalement de deux contributions qui concernent une méthode de vérification des opérateurs de convection-diffusion de l'outil informatique OpenFOAM. Une première contribution est le développement de la méthode de vérification elle-même, qui a permis d'identifier la source de perte de convergence sur des maillages non orthogonaux. À cette fin, une méthode qui tient compte de la complexité du solveur (c.-à-d., la complexité des schémas), des conditions aux limites, ainsi que de la qualité du maillage est établie dans une première étape. La principale contribution de cette première partie est la détection de la source d'une perte de convergence due au traitement de la non-orthogonalité pour des conditions aux limites de type Dirichlet et de type Neumann. Autres résultats dans cette première phase sont : la vérification du bon fonctionnement des algorithmes à l'intérieur du domaine et sur des maillages orthogonaux, la confirmation de l'ordre théorique de convergence des conditions aux limites périodiques sur des maillages orthogonaux et non orthogonaux, ainsi que la vérification du bon fonctionnement des algorithmes concernant les critères de qualité du maillage, la non-uniformité et l'asymétrie (skewness en anglais). Pour étudier l'erreur, la méthode des solutions manufacturées a été appliquée pour obtenir des solutions analytiques de l'équation de Poisson. Les normes d'erreurs moyenne et maximale ont été utilisées pour calculer le taux de convergence.

La seconde contribution est constituée de différentes façons afin de corriger le problème de perte de convergence. Nous avons aussi appliqué le processus de vérification aux schémas corrigés. Dans cette deuxième partie, on montre mathématiquement la nécessité d'introduire une correction aux faces à la frontière. À l'aide de l'analyse d'erreur de troncature pour les conditions de type Dirichlet ou Neumann, nous avons montré la nécessité d'implémenter un traitement non orthogonal pour les deux types de conditions aux limites. Par la suite, différentes méthodes pour obtenir des conditions aux frontières de deuxième ordre ont été implémentées. Dans les cas de conditions aux limites de type Dirichlet, quatre alternatives pour atteindre une convergence d'ordre deux ont été présentées, dont trois peuvent être étendues afin de permettre de traiter la non-orthogonalité aux frontières de type Neumann. Les quatre variantes implémentent une correction non orthogonale pour les conditions aux limites de type Neumann et Dirichlet en reconstruisant le gradient dans les cellules voisines à la frontière et les quatre présentent des résultats similaires en termes de convergence et de précision. Plusieurs tests de convergence permettent de faire des comparaisons entre les résultats utilisant les nouveaux schémas et les résultats utilisant le code original. Enfin, nous avons détecté une "petite" perte de convergence provenant de la libraire de reconstruction du gradient pour une méthode aux moindres carrés, due à la reconstruction des vecteurs de moindres carrés (les vecteurs de liaison) à la frontière dans les cas de maillages non orthogonaux. On propose une correction du calcul de ces vecteurs.

Dans ce document, on fait une synthèse des deux articles qui expliquent les détails techniques et théoriques de la méthode de vérification qui a permis l'identification de la source de perte de convergence : le traitement de la non-orthogonalité pour des conditions aux limites Dirichlet ou Neumann. De plus, on décrit les différentes stratégies pour obtenir des conditions aux limites de deuxième ordre. Finalement, la contribution principale de cette thèse est une amélioration de la convergence des librairies OpenFOAM dans le traitement aux limites des types Dirichlet et Neumann, ce qui résulte en des simulations numériques plus fiables.

ABSTRACT

Based on practical observation, an *a* priori dependence of OpenFOAM® schemes on grid type and quality is supposed to exist. This dependency is characteristic of the numerical methods that support this and other softwares. In this regard, this study aims the assessment of OpenFOAM® libraries as well as the analysis of the influence of cell type on the numerical accuracy of convection-diffusion operators. OpenFOAM (for "Open source Field Operation And Manipulation") is a multi-physics code that is largely used in industry and academy, why is justifiable a study of verification of this tool. The code (that is writting in C++) is released as free and open source software under the GNU General Public License. This work is divided in two parts.

In the first part of this investigation, the analysis of schemes for convection-diffusion problems is focused only on the numerical diffusion, because this individually has been found to show a loss in convergence rate. Although second order schemes that are based on the cell centered finite volume approach underlying OpenFOAM should preserve a theoretical second order convergence rate, loss of convergence order is observed when applying non-orthogonal meshes at the boundaries. Theoretical convergence orders were observed for Poisson's equation solver on orthogonal hexahedral meshes using several boundary conditions. On non-orthogonal meshes, we have noticed that even slight mesh distortions reduce the theoretical second order convergence to just first-order. Our investigation has shown that this loss of convergence order takes place when Dirichlet and/or Neumman boundary conditions are used together with a non-orthogonal treatment on boundaries. To study the error, the method of manufactured solutions has been applied to yield analytical solutions for the Poisson equation. The average and maximum error norms were used to calculate the convergence rate. The root cause of the accuracy loss is identified and corrections to recover second order convergence are proposed.

In part two of this investigation which is the follow-up of the previous first part, some ways to achieve this theoretical second order convergence accuracy are shown. They are mainly based on a second order boundary approximation for Dirichlet and/or Neumann boundaries. Additionally, it is shown that a small reduction on the convergence order can also be observed when using the least square gradient scheme for the explicit part on the treatment for non-orthogonality. This seemed to be caused by the non-orthogonal correction in boundaries

inside the least square vector reconstruction library. The verification procedure is performed by means of the method of manufactured solutions for the Poisson equation or by means of analytical solutions for the Navier-Stokes equations. Comparative results are presented.

This document is a synthesis of two articles in whichs we explain the technical and the theoretical details of the implemented verification methodology. The overall contribution of this thesis is to have contributed to the current state of development in terms of convergence of the OpenFOAM software tool (In general the contribution is valid for the creation of second order boundary conditions based on the cell centered scheme in finite volumes). This contribution is of great importance for the numerical simulations.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

Caractères usuels

$\vec{u}$	champ de vitesse
$\vec{v}$	champ de vitesse
$U(x, y)$	composant horizontal de vitesse
p	pression
q	terme source d'une équation
Re	nombre de Reynolds
t	temps

Lettres grecques

α	facteur d'interpolation
β	facteur de relaxation
μ	viscosité dynamique
ν	viscosité cinématique
Γ	coefficient de diffusion
ϕ	propriété physique scalaire
ϕ_h	valeur numérique de ϕ
$\tilde{\phi}$	valeur analytique de ϕ
ρ_0	masse volumique du fluide

domaine de simulation

Ω	domaine de calcul
$\|\Omega\|$	volume du domaine de calcul
i	indice d'une cellule de calcul
P	barycentre d'une cellule de calcul
N	barycentre de la cellule voisine
$\vec{d}$	vecteur qui fait la liaison entre les barycentres P et N
$\vec{d_u}$	vecteur unitaire de $\vec{d}$
f	une face d'une cellule de calcul
C_f	barycentre de la face f
e_i	erreur numérique dans la cellule i
e_k	erreur numérique globale d'une partition uniforme du domaine (k fois)

L_p	espace de Lebesgue $\|\|f\|\|_p = \frac{1}{\|\Omega\|} \left(\int_\Omega \|f\|^p \, d\Omega\right)^{\frac{1}{p}}$
L_1	espace de Lebesgue $\|\|f\|\|_1 = \frac{1}{\|\Omega\|} \left(\int_\Omega \|f\| \, d\Omega\right)$
L_∞	espace de Lebesgue $\|\|f\|\|_\infty = \sup_{x \in \Omega} \|f(x)\|$
$\|\|e\|\|_{L_1(\Omega)}$	norme utilisée dans ce travail pour l'erreur moyenne
$\|\|e\|\|_\infty$	norme utilisée dans ce travail pour l'erreur maximale
R_k	taux de convergence
sup	valeur supremum dans un ensemble
max	valeur maximum dans un ensemble discret
V_p	volume d'une cellule de calcul
$\vec{S}$	vecteur normal à une face f
$\vec{\Delta}$	vecteur de contribution orthogonale
$\vec{k}$	vecteur de correction non orthogonale
F	flux de masse à travers une face f
α_N	mesure d'angle

Symboles

$\bigtriangledown$	opérateur gradient
$\bigtriangledown \cdot$	opérateur divergence
$\bigtriangledown \cdot \bigtriangledown$	opérateur de Laplace
$\frac{\partial}{\partial t}$	dérivée partielle en temps
$\frac{\partial}{\partial x}$	dérivée partielle en x
$\int_{V_p}$	intégral volumique en V_p
$\oint_{S_p}$	intégral de surface en V_p
$\sum_f$	somme sur les faces f d'une cellule de calcul

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Les simulations en mécanique des fluides sur des maillages complexes sont rapidement devenues incontournables pour la modélisation des problèmes réels. Ces simulations informatiques permettent d'éviter le coût et, dans certains cas, de limiter le risque d'une série d'épreuves réelles. Bien que les ressources informatiques sont devenues de plus en plus puissantes permettant la résolution numérique des équations aux dérivées partielles avec une grande précision, la simulation des problèmes réels reste un défi majeur. Pour être sûres que les résultats sont cohérents avec les modèles physiques et mathématiques sous-jacents, la validation et la vérification des résultats devraient être une partie incontournable de la modélisation physique. Même après avoir surmonté certains des problèmes communs des schémas numériques tels que la convergence et la stabilité, on peut se poser les questions : est-ce que la solution obtenue est une solution fiable par rapport au problème réel ? Est-ce que les algorithmes numériques ont été résolus avec la bonne exactitude ? Est-ce qu'il n'y a pas d'erreur de codage (tels que des erreurs de logique) ? Un processus de validation doit répondre à la première question, alors qu'un processus de vérification doit permettre de répondre aux deux dernières (voir (Oberkampf and Roy, 2010)).

Le contexte général de cette recherche est la vérification des opérateurs de convection-diffusion dans la librairie de calcul numérique OpenFOAM ®. OpenFOAM (acronyme anglais "Open Field Operation and Manipulation") est un outil de simulation multiphysique principalement axé sur la résolution des équations de la mécanique des fluides. Distribué sous licence open source GNU/GPL, il est largement utilisé dans les milieux universitaires et dans l'industrie, c'est pourquoi il est justifié de vérifier en profondeur ses librairies principales. OpenFOAM est basé sur une discrétisation de type volumes finis. L'observation pratique montre que l'ordre de convergence des schémas numériques d'OpenFOAM dépend du type et de la qualité des maillages. Dans le cadre de ce travail de recherche, nous nous limiterons au processus de vérification des librairies de résolution des opérateurs de convection-diffusion, ainsi qu'à l'analyse de l'influence du maillage sur la précision numérique de ces opérateurs.

1.1 Le processus de simulation numérique en mécanique des fluides

L'une des méthodes les plus importantes et les plus universelles de résolution des problèmes en physique est la méthode d'application des lois de conservation. Toutes les lois de conservation sont unies par le fait qu'une quantité physique reste constante dans certaines conditions. Étant

donné que les propriétés physiques du fluide à l'intérieur d'un volume V sont influencées par le milieu environnant, certains principes physiques comme la conservation de la masse, de la quantité de mouvement et de l'énergie doivent être respectés. Autrement dit, ces quantités peuvent se déplacer d'un endroit à un autre, mais ne peuvent pas apparaître ou disparaître spontanément. La forme générale d'une équation de transport, donnée ci-dessous, est utilisée pour décrire les lois de conservation d'un phénomène physique en mécanique des fluides (voir Jasak (1996)) :

$$\underbrace{\int_V \frac{\partial \rho\phi}{\partial t}\, d\tau}_{\text{dérivé temporelle}} + \underbrace{\int_V \nabla \cdot (\rho\vec{v}\phi)\; d\tau}_{\text{terme de convection}} - \underbrace{\int_V \nabla \cdot (\rho\Gamma_\phi \nabla\phi)\; d\tau}_{\text{terme de diffusion}} = \underbrace{\int_V q(\phi)\, d\tau}_{\text{terme source}} \tag{1.1}$$

où ϕ représente une composante tensorielle d'une propriété physique, $\vec{v}$ la vitesse du fluide, Γ_ϕ le coefficient de diffusion et ρ désigne la masse volumique du fluide. Le terme avec la dérivée temporelle représente la variation dans le temps de la propriété physique $\rho\phi$ à chaque point à l'intérieur du volume V. Le terme de convection représente que la propriété physique $\rho\phi$ à un endroit donné peut changer en raison de l'écoulement. L'interprétation du terme de diffusion dépend du type de phénomène physique. Dans le cas d'un fluide en mouvement, ce terme représente la propagation de la quantité du mouvement due à l'interaction entre les particules (généralement par pression ou par contrainte de cisaillement).

Les équations de Navier-Stokes sont un ensemble d'équations aux dérivées partielles dérivées des lois de conservation qui gouvernent la dynamique d'un fluide. Un certain nombre de fluides comme l'eau et l'air sont approchés par le modèle des fluides appelé fluide newtonien spécifié ainsi (voir Jasak (1996)) :

— l'équation de bilan de la masse

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho\vec{v}) = 0 \tag{1.2}$$

— l'équation de bilan de la quantité de mouvement

$$\frac{\partial \rho\vec{v}}{\partial t} + \nabla \cdot (\rho\vec{v}\vec{v}) = \rho\vec{g} - \nabla\left(p + \frac{2}{3}\mu\nabla \cdot \vec{v}\right) + \nabla \cdot \left[\mu\left(\nabla\vec{v} + (\nabla\vec{v})^T\right)\right] \tag{1.3}$$

où p désigne la pression, la viscosité dynamique et $\vec{g}$ désigne l'accélération de la gravité.

— l'équation de bilan de l'énergie

$$\begin{aligned}\frac{\partial \rho e}{\partial t} + \nabla \cdot ((\rho e + p)\vec{v}) &= \rho \vec{g} \cdot \vec{v} + \nabla \cdot \left(\frac{2}{3}\mu \left(\nabla \cdot \vec{v}\right)\vec{v}\right) \\ &+ \nabla \cdot \left[\mu \left(\nabla \vec{v} + (\nabla \vec{v})^t\right) \cdot \vec{v}\right] + \nabla \cdot (\lambda \nabla T) + \rho Q \end{aligned} \quad (1.4)$$

où T est la température du milieu, $-\lambda\left(\nabla T\right)$ est le flux de chaleur perdu par conduction thermique dans le volume de contrôle et Q est un terme source de chaleur dans le même volume.

Le problème qui résulte de la modélisation des équations de Navier-Stokes est un problème continu dans l'espace et dans le temps, et doit alors être approché par un problème discret lorsqu'il est résolu numériquement, c.-à-d., les variables physiques inconnues doivent être représentées par un nombre discret de valeurs. Cette procédure de discrétisation implique principalement deux tâches (voir Schafer (2006)) :

— la discrétisation du domaine de définition du problème à modéliser
— la discrétisation des équations

La discrétisation du domaine approche le domaine continu dans l'espace et dans le temps par une représentation discrète, c.-à-d., par un nombre fini de sous-domaines dans lesquels les lois de conservation sont appliquées et dans lesquels les valeurs numériques des quantités inconnues sont déterminées. La discrétisation spatiale du domaine se fait sous la forme d'un maillage, c.-à-d., une représentation géométrique discrète du domaine. Ce processus de discrétisation spatiale d'un domaine en sous-domaines, qui forment un recouvrement du domaine physique, est appelé génération du maillage. Les résultats de la solution numérique dépendent de la qualité du maillage. Un maillage bien construit peut améliorer la qualité de la solution alors qu'une déviation de la solution numérique peut être observée sur des maillages mal construits.

L'ensemble des relations pour le calcul des valeurs inconnues est obtenu par un système algébrique d'équations qui approchent le système d'équations différentielles initial. La solution numérique produit un ensemble de valeurs liées au domaine discrétisé du problème à partir duquel une approximation de la solution peut être construite. Il existe principalement trois différentes approches disponibles pour la procédure de discrétisation des équations :

— la méthode des différences finies (MDF)
— la méthode des volumes finis (MVF)
— la méthode des éléments finis (MEF)

La méthode des différences finies est l'une des plus anciennes et des plus simples. En fait, étant "plus simple" que les méthodes MVF et MEF, la méthode des différences finies est souvent la méthode préférée lorsqu'un cours de méthodes numériques est introduit. Actuellement, les méthodes MEF et MFV sont plus fréquemment utilisées pour résoudre des problèmes numériques. Bien que la méthode MEF soit utilisée principalement dans le domaine de la mécanique des structures, la méthode MFV est plutôt utilisée dans les applications de simulation des fluides (voir Schafer (2006)).

L'avantage des méthodes MEF par rapport aux méthodes MDF est que les géométries complexées, les conditions aux limites et les variables ou les propriétés des matériaux non linéaires peuvent être manipulés relativement de façon plus facile. Le MEF est un bon choix pour analyser des problèmes sur des domaines complexes, lorsque le domaine change, lorsque la précision souhaitée varie sur l'ensemble du domaine ou lorsque la solution manque de régularité. De plus, MEF a une base théorique solide qui donne une fiabilité accrue et permet dans de nombreux cas d'analyser et d'estimer mathématiquement l'erreur. Les applications initiales de FEM ont commencé par l'analyse structurale à la fin des années 1950 et ont principalement été basées sur des principes variationnels (voir Schafer (2006), Chung (2010)).

Une propriété importante des méthodes MVF est que le principe d'équilibre, qui est à la base de la modélisation mathématique des problèmes en mécanique des milieux continus, par définition, est également satisfait par les équations discrètes. Le flux entrant dans un volume donné à travers d'une face est identique à celui que quitte le volume adjacent, ces méthodes sont conservatives. Un autre avantage de MVF est qu'il est facilement formulé pour permettre des maillages non structurés. Une brève présentation des fondements mathématiques de MVF est donnée au chapitre 2.

Le système d'équations discret résultant obtenu par l'une des méthodes MDF, MFV ou MEF, est un grand système linéaire creux d'équations algébriques. L'étape suivante dans le processus de simulation consiste à trouver la solution du système d'équations algébriques. On le représente dans sa forme matricielle :

$$[A]\,[\phi] = [b]$$

La matrice du système $[A]$ résultant est généralement non singulière (sous certaines exigences pour la discrétisation), de telle façon que le système d'équations possède une solution unique.

Il y a deux possibilités pour la solution numérique d'un système matriciel, lesquelles sont amplement documentées dans la littérature (voir Schafer (2006), Ferziger and Peric (2002)) :

— des méthodes directes
— des méthodes itératives

La principale caractéristique des méthodes directes est que la solution exacte du système (en négligeant l'erreur d'arrondi) est obtenue par un algorithme non itératif. Généralement la solution est obtenue après un nombre fini de pas. Les méthodes les plus populaires de ce type sont : l'algorithme de Thomas, la méthode d'élimination de Gauss-Jordan et la méthode de décomposition LU. Les méthodes directes sont convenables pour les systèmes de petite taille. Pour les systèmes d'équations de très grande taille, le nombre d'opérations nécessaires augmente rapidement avec le nombre d'équations (sauf dans le cas de l'algorithme de Thomas, propre aux systèmes tri diagonaux, pour lequel la complexité est linéaire). Parallèlement à la complexité de calcul, la mémoire nécessaire pour les méthodes directes augmente également en ordre carré de complexité par rapport au nombre d'équations, rendant cette approche encore moins attrayante pour les systèmes d'équations de grande taille.

Les méthodes itératives (voir Hackbusch (1998), Schafer (2006), Ferziger and Peric (2002)) sont plus économiques en mémoire. Pour ces méthodes, une approche de solution initiale ϕ_0 est successivement améliorée par l'application répétitive d'une certaine règle d'itération :

$$\phi^{k+1} = P(\phi^k),\ k = 0, 1, ...$$

Les méthodes de Jacobi et de Gauss-Seidel appartiennent aux méthodes les plus simples de solution itérative. Par exemple, la méthode de Gauss-Seidel améliore la convergence du processus itératif en utilisant les valeurs déjà calculées dans l'expression récurrente :

$$\phi_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}\phi_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}\phi_j^k \right) \tag{1.5}$$

où a_{ij} sont les coefficients de la matrice.

Un critère suffisant (mais pas nécessaire) pour la convergence des méthodes de Jacobi et de Gauss-Seidel est que la matrice du système soit diagonale dominante, c.-à-d., si la valeur absolue de chacun des coefficients de la diagonale de la matrice est égale ou plus grande que la somme des valeurs absolues des coefficients non diagonaux pour chaque équation, et au moins pour un coefficient, l'inéquation est stricte ($>$) :

$$|a_{ii}| \geq \sum_j |a_{ij}|, \ j \neq i$$

où a_{ii} sont les coefficients de la diagonale de la matrice et a_{ij} $(j \neq i)$ sont les coefficients non diagonaux pour chaque équation.

Pour améliorer la convergence de la méthode de Gauss-Seidel, une sur-relaxation est introduite, ce qui entraîne la méthode de surrelaxation successive SOR ("successive over-relaxation" en anglais). Il existe plusieurs variantes de la méthode de surrelaxation successive qui diffèrent par l'ordre dans lequel les nœuds individuels sont traités. La convergence de ces algorithmes est généralement plus rapide. Étant donné le système matriciel :

$$[A]\,[\phi] = [b] \tag{1.6}$$

La méthode itérative consiste à déterminer ϕ^{k+1} à l'aide de ϕ^k selon la formule suivante :

$$\phi^{k+1} = (D + \beta L)^{-1}(\beta b - [\beta U + (\beta - 1)D]\phi^k) \tag{1.7}$$

où la matrice A est la somme d'une matrice diagonale notée D et de deux matrices triangulaires, inférieure L et supérieure U respectivement, c.-à-d., $A = D+L+U$. Le choix du facteur de relaxation β n'est pas trivial et dépend des coefficients de la matrice. Pour une matrice symétrique et définie positive, il a été démontré que l'algorithme est convergent pour $\beta \in (0, 2)$.

Une autre classe de méthodes itératives, qui est devenue populaire grâce à la bonne convergence et à la robustesse algorithmique, est basée sur la décomposition LU incomplète de la

matrice du système. La décomposition $A = LU$ est une méthode de décomposition d'une matrice comme produit d'une matrice triangulaire inférieure L (comme lower en anglais) et une matrice triangulaire supérieure U (comme upper en anglais). Une décomposition LU incomplète, comme pour une décomposition LU, consiste en un fractionnement multiplicatif de la matrice de coefficients A dans une matrice triangulaire inférieure L et une autre triangulaire supérieure U (à nouveau notées comme L et U) qui devraient se rapprocher de la matrice A autant que possible $A \approx LU$. Les matrices L et U devraient être déterminées pour que le processus d'itération soit aussi efficace que possible. Les méthodes de décomposition LU incomplète sont appliquées aux matrices creuses, les facteurs LU peuvent être beaucoup moins creux que la matrice d'origine.

La résolution du système obtenu $LU\phi = b$ est rapide, mais ne donne pas la solution exacte $A\phi = b$. Donc, la matrice $M = LU$ est plutôt utilisée comme un pré-conditionnement (qui est l'application d'une transformation qui conditionne un problème donné dans une forme qui est plus appropriée pour sa résolution numérique) dans un autre algorithme de solution matricielle itérative. La méthode du gradient conjugué et la méthode du gradient bi conjugué sont des exemples de ce type de méthode pré-conditionnée. Les détails techniques de mise en œuvre de ces méthodes itératives sont largement documentés dans la littérature spécialisée (voir Hackbusch (1998), Schafer (2006), Ferziger and Peric (2002)).

Dans une première instance, les résultats du calcul numérique produisent une grande quantité de données qui normalement n'a pas une compréhension intuitive. Par conséquent, pour l'évaluation des résultats obtenus, une visualisation appropriée des résultats est importante. À cet effet, des logiciels spécialisés sont disponibles.

D'autre part, il est essentiel d'inspecter les résultats numériques obtenus par rapport à leur qualité. Au cours de toutes les étapes du calcul numérique, des erreurs sont inévitablement introduites, et il est nécessaire d'obtenir une quantification de ces erreurs (par exemple, avec des données expérimentales de références, avec des solutions asymptotiques, etc.). À cet effet, deux aspects doivent être distingués :

— **La validation** répond à la question : les équations appropriées ont elles été résolues ? La validation est le processus de détermination du degré auquel un modèle est une représentation exacte du modèle physique or du monde réel, du point de vue des utilisations prévues du modèle (voir (Oberkampf and Roy, 2010)).

— **La vérification** répond à la question : les équations sont-elles résolues de façon correcte ? La vérification est le processus consistant à déterminer que la mise en œuvre d'un modèle représente la description conceptuelle du développeur. La vérification doit donner la preuve que le modèle informatisé représente un modèle conceptuel dans les limites spécifiées de précision. Une introduction au processus de vérification numérique est donnée au chapitre 2.3.

1.2 OpenFOAM : Une brève introduction

Basé sur le langage C++, OpenFOAM est un logiciel programmé selon le paradigme orienté objet avec toutes les propriétés de ce paradigme de programmation telles que l'encapsulation, le polymorphisme et l'héritage. De plus, OpenFOAM contient un très grand nombre de librairies pour résoudre les problèmes de la mécanique des fluides numériques ainsi qu'une variété de structures de données spécialisées pour traiter les différents champs (p. ex. les champs scalaire, vectoriel ou tensoriel) pour les différents types de propriétés physiques (voir (OpenFOAM, 2011)).

La distribution de OpenFOAM contient de nombreux solveurs et outils qui permettent de couvrir les différentes étapes d'une simulation numérique. Un des points forts de OpenFOAM est la possibilité de créer de nouveaux solveurs pour l'utilisateur qui possède des connaissances préalables de base des techniques, de la physique et de programmation sous-jacente impliquées. OpenFOAM est ainsi fourni avec des outils de pré-traitement et de post-traitement. Un survol des solveurs et des outils principaux est donné ci-dessous.

Pré-traitement. Les outils suivants sont fréquemment utilisés pour le pré-traitement d'une simulation :

— *blockMesh*, générateur de maillage pour géométries simples,
— *snappyHexMex*, générateur de maillage pour géométries complexes,
— des outils de conversion de maillage vers le format OpenFoam comme *fluentMeshToFoam, gambitToFoam, cfx4ToFoam, gmshToFoam, etc*,
— des outils pour faire la correspondance (mapping en anglais) d'un ou plusieurs champs physiques relatifs à une géométrie vers une autre géométrie comme *mapFields*,

— des outils pour la vérification de la qualité des maillages comme *checkMesh*, etc.

Post-traitement. Le principal outil de post-traitement fourni par OpenFOAM est un module de lecture des données produites par une simulation OpenFOAM qui fonctionne avec ParaView, qui est un logiciel à code source libre de visualisation de données fondé sur la bibliothèque VTK. D'autres outils de post-traitement incluent des outils pour calculer p.ex. les résidus matriciels, le nombre de Courant (à partir des fichiers de simulations), ainsi que des outils pour exporter les données numériques vers d'autres logiciels comme *fluent, EnSight, OpenDX, etc.*

Les solveurs. OpenFOAM couvre une vaste gamme d'applications à l'aide de plusieurs dizaines de solveurs (plus de quatre-vingts solveurs dans la version OpenFOAM 3.x). On énumère ici certains d'entre eux :

— *laplacianFoam*, solveur pour résoudre l'équation de Laplace d'une quantité scalaire
— *icoFoam*, solveur non stationnaire pour un fluide newtonien incompressible laminaire
— *nonNewtonianIcoFoam*, solveur non stationnaire pour un fluide non newtonien incompressible laminaire
— *pisoFoam*, solveur non stationnaire pour un fluide incompressible utilisant l'algorithme PISO.
— *simpleFoam*, solveur stationnaire pour un fluide incompressible avec modélisation de la turbulence utilisant l'algorithme SIMPLE.
— *rhoSimpleFoam*, solveur stationnaire pour un fluide compressible avec modélisation RANS de la turbulence utilisant l'algorithme SIMPLE.
— *sonicFoam*, solveur non stationnaire pour les régimes transsonique/ supersonique, laminaires ou turbulents d'un gaz compressible
— *sonicDyMFoam*, solveur non stationnaire pour les régimes transsonique/ supersonique, laminaires ou turbulents d'un gaz compressible avec un déplacement du maillage

Plusieurs autres solveurs sont disponibles pour les écoulements compressible et incompressible. Quelques autres dizaines de solveurs sont aussi disponibles pour d'autres types de phénomènes physiques comme la combustion, le transfert de chaleur, les écoulements polyphasiques (multiphasiques), les écoulements avec suivi de particules, l'électromagnétisme, les finances, les interactions fluide-structure, etc.

Les conditions aux limites. OpenFOAM permet de combiner sa vaste gamme de solveurs avec une autre vaste gamme de conditions de frontières, qui incluent les types standards de conditions aux limites comme les types Dirichlet, Neumann et périodique.

Optimisation du calcul matriciel. La résolution des systèmes matriciels dans OpenFOAM utilise les méthodes itératives qui sont plus économiques en ressources informatiques. Les solveurs utilisent un pré-conditionnement, de sorte que la convergence du système préconditionné est beaucoup plus rapide que pour l'original, et utilisent du lissage, qui sont des transformations pour réduire la dépendance itérations - maillage. Voici quelques options : ICC, solveur fondé sur la méthode du gradient conjugué avec pré-conditionnement de Cholesky incomplète ; BICCG, solveur avec pré-conditionnement diagonal et décomposition LU incomplète ; PCG, solveur avec pré-conditionnement du gradient bi-conjugué pour des matrices LDU symétriques qui utilisent un pré-conditionnement sélectionnable à l'exécution, etc.

1.3 Objectifs de la recherche

L'objectif de cette recherche est de vérifier les opérateurs de convection-diffusion disponibles dans les bibliothèques OpenFOAM et de corriger une perte de convergence qui a été mise en évidence lors de l'utilisation de l'outil de simulation numérique OpenFOAM sur des maillages non orthogonaux. Cet objectif peut être décomposé en 3 étapes : identifier la source (ou les sources) de ce problème, analyser les composantes mathématiques et logicielles impliquées et finalement proposer des alternatives de solution pour le résoudre.

1.4 Plan de la thèse

Le format choisi pour cette thèse est une thèse par articles où les articles sont placés en annexes. Le chapitre courant présente une introduction au processus de simulation numérique, ainsi qu'une brève introduction au logiciel OpenFOAM et aux objectifs de la thèse.

Dans le chapitre suivant, on présente une revue de la littérature permettant de comparer des schémas volumes finis centrés aux éléments (VFCE) aux schémas centrés aux sommets (VFCS). Des comparaisons de ces schémas seront faites en ce qui concerne la convergence, la performance et la précision. De plus, on présente les fondements mathématiques des schémas volumes finis centrés aux éléments. Enfin, on présente les concepts de base nécessaires pour

comprendre le processus de vérification comme la méthode de solutions manufacturées et l'erreur numérique.

Le troisième chapitre fait la synthèse des deux articles en annexe. Nous présenterons un résumé des principaux aspects des deux articles en présentant la méthodologie suivie, et les résultats pour chaque article.

Le quatrième chapitre présente une discussion générale de l'ensemble des résultats de cette étude de vérification. Le cinquième chapitre présente un résumé des principaux aspects liés à l'implémentation technique d'un traitement non orthogonal à la frontière. Enfin, le chapitre avec la conclusion et les recommandations est présenté.

CHAPITRE 2 REVUE CRITIQUE DE LA LITTERATURE

Ce chapitre présente les notions nécessaires à la compréhension des concepts et fondements mathématiques de la méthode des volumes finis, qui est à la base de la discrétisation utilisée dans le logiciel OpenFOAM. D'abord, nous allons comparer les schémas volumes finis centrés sur les éléments avec les schémas volumes finis centrés sur les sommets. Les deux schémas sont largement utilisés dans la méthode des volumes finis (MVF) pour stocker les valeurs des variables physiques d'un problème à résoudre. Des comparaisons des deux schémas en termes de performance et de précision seront montrées. Ensuite, on présente les schémas volumes finis centrés sur les éléments pour le gradient et pour les opérateurs de convection-diffusion, lesquels sont théoriquement d'ordre deux. Ce chapitre présente aussi une revue de la méthode de vérification des codes numériques en mécanique des fluides, la méthode des solutions manufacturées, ainsi que les normes d'erreur numérique.

2.1 Schémas de volumes finis

Dans la méthode des volumes finis, lors de la discrétisation d'un système d'équations, la première question est de savoir comment définir les volumes de contrôle et où trouver les variables de flux par rapport au maillage de calcul. Les schémas centrés sur les éléments VFCE, les schémas centrés sur les sommets VFCS et un schéma hybride (centrés sur les éléments et centrés sur les sommets) avec chevauchement de volumes de contrôle, seront d'abord présentés et comparés (voir Blazek (2006)).

2.1.1 Schéma VFCE

On parle de schéma VFCE, comme ceux utilisés à la Sec. 2.2, lorsque les volumes de contrôle sont identifiés aux éléments du maillage et que les variables de mouvement (la vitesse et la pression) sont associées à leurs barycentres. On l'illustre à la Fig. 2.1 où les points de coordonnées{(I,J), (I,J+1), (I,J-1), (I+1,J), (I-1,J)} représentent les barycentres des éléments. Dans ce type de schéma, le flux convectif et le flux diffusif doivent être fournis pour chacune des faces de l'élément de calcul. Ils peuvent être estimés de l'une des trois façons suivantes (voir Blazek (2006)) :

— en calculant la moyenne des flux. Ceux-ci sont calculés à partir des valeurs aux barycentres des éléments à gauche et à droite de la face en commun, mais en utilisant

le même vecteur normal à cette face (généralement appliqué uniquement aux flux convectifs).

— en calculant la moyenne des variables associées au flux à partir des valeurs de ces variables aux barycentres de l'élément à gauche et de l'élément à droite de la face en commun.

— en calculant séparément les flux à partir de l'extrapolation des valeurs à gauche et à droite de la face. En général, ces quantités extrapolées à partir des valeurs à gauche et à droite différent l'une de l'autre. Par la suite, une méthode (une fonction non-linaire) permet d'obtenir le flux à la face à partir de la différence des valeurs à gauche et à droite. Ce type de méthode permet de générer différents schémas de calcul appliqués uniquement aux flux convectifs.

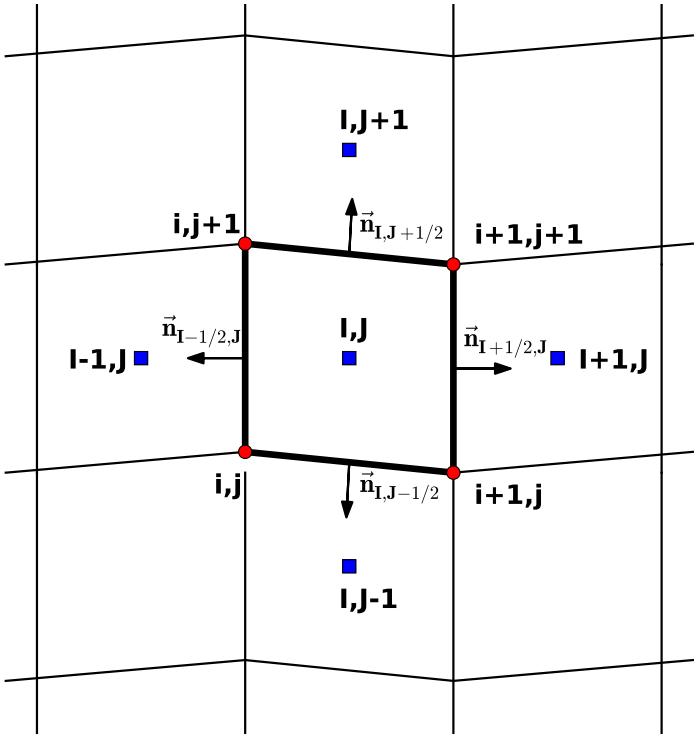


Figure 2.1 Volume de contrôle pour le schéma centré sur l'élément

2.1.2 Schéma centré aux sommets avec chevauchement des volumes de contrôle

Dans cette approche, les variables de mouvement sont attribuées aux sommets du maillage et les volumes de contrôle sont définis comme l'union de tous les éléments du maillage ayant un sommet en commun, typiquement 4 éléments en 2D (voir la Fig. 2.2) et 8 éléments en 3D.

Cela signifie que les volumes de contrôle associés à deux sommets voisins se chevauchent.

Dans l'approche avec chevauchement des volumes de contrôle, les éléments du maillage représentent encore les volumes de contrôle, tout comme dans le cas du schéma VFCE (voir Blazek (2006)). Le super volume de contrôle pour le sommet (i,j) devient donc le volume :

$$\Omega_{i,j} = \Omega_{I,J} + \Omega_{I-1,J} + \Omega_{I,J-1} + \Omega_{I-1,J-1} \tag{2.1}$$

où $\Omega_{I,J}$ est le volume de contrôle (voir Fig. 2.2) qui est défini, dans le cas 2-D, par les sommets :

$$\{(i,j),\quad (i+1,j),\quad (i+1,j+1),\quad (i,j+1)\}$$

Le flux convectif (ou diffusif) est calculé pour chacune des faces comme dans le cas VFCE. La somme des flux sur toutes les faces de l'élément de calcul permet d'obtenir l'expression du résidu de type VFCE au noeud (I, J).

Un résidu de type VFCE, dans le barycentre P d'une cellule de calcul, mesure le résidu de l'expression algébrique de l'équation de conservation, c.-à-d. :

$$R_P = \left| a_P \phi_P - \sum_{nb} a_{nb} \phi_{nb} - b \right| \tag{2.2}$$

où ϕ est une variable physique, l'index P représente le nœud (I, J), l'index nb représente les nœuds voisins (les nœuds (I,J+1), (I,J-1), (I-1,J), (I+1,J+1)), a_{nb} sont des constantes (la contribution des éléments voisins) produites par la discrétisation et b la contribution du terme source.

Afin de relier les résidus intermédiaires VFCE avec les résidus VFCS, une nouvelle approximation est faite au sommet (i,j) à partir des valeurs obtenues sur les noeuds des éléments {(I,J), (I,J-1), (I-1,J), (I-1,J-1)} en utilisant des approches de pondération. Cela permet essentiellement d'évaluer les résidus du sommet inconnu (i,j) à partir d'une somme pondérée de toutes les valeurs aux barycentres des éléments ayant ce sommet particulier en commun. Différentes approches de pondération peuvent être citées :

— une pondération par les volumes, proposée par Ni (1981)

— une somme non pondérée, proposée par Hall (1985)
— une procédure de pondération vers l'amont (upwind en anglais), proposée par Rossow (1990) et Rossow (1993)

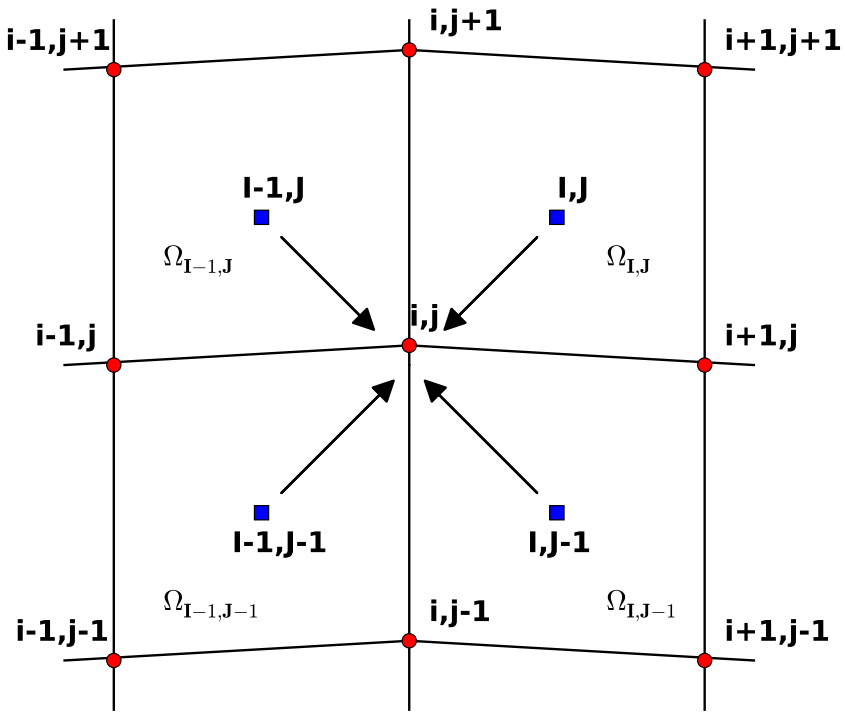


Figure 2.2 Volume de contrôle pour le schéma noeud centré pondéré

Des recherches théoriques sur l'erreur de troncature réalisées par Rossow (1989) suggèrent que le schéma de Ni est plus précis que l'approche de Hall. Toutefois, en pratique la formule d'interpolation de Ni conduit à des problèmes aux endroits où le maillage est fortement déformé et étiré. Parmi les trois approches, le schéma d'interpolation de Hall s'est avéré être le plus robuste (voir Schafer (2006)).

Le schéma centré aux sommets avec chevauchement est lourd en complexité de calcul et il est actuellement utilisé rarement.

2.1.3 Schéma centré aux sommets avec un volume de contrôle dual

Pour ce schéma, le volume de contrôle est centré autour d'un sommet sur lequel les variables de mouvement sont stockées. En 2-D le volume de contrôle dual peut être créé en reliant le barycentre d'un élément avec le point au milieu d'une arête et après avec le barycentre de l'élément voisin (décomposition de Voronoï). De cette façon, les points du maillage sont entourés par des volumes de contrôle qui ne se chevauchent pas. Ainsi, la face du volume de contrôle est composée de deux parties avec différents vecteurs normaux (voir la Fig. 2.3). Sur les frontières, la discrétisation de la face peut être modifiée, comme cela est illustré sur la Fig. 2.4 à gauche. Une redéfinition du volume de contrôle peut être faite en utilisant un sommet à la frontière telle qu'illustrée sur la Fig. 2.4 à droite (voir Blazek (2006)). Cependant, cette dernière approche peut générer des inconsistances comme on le verra plus tard (voir Sec. 2.1.4).

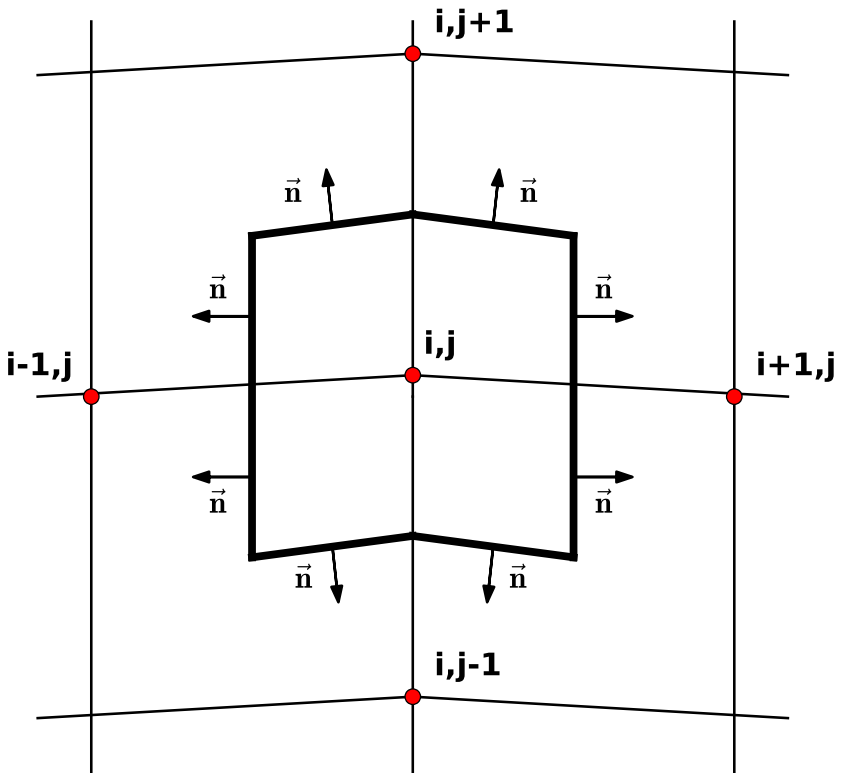
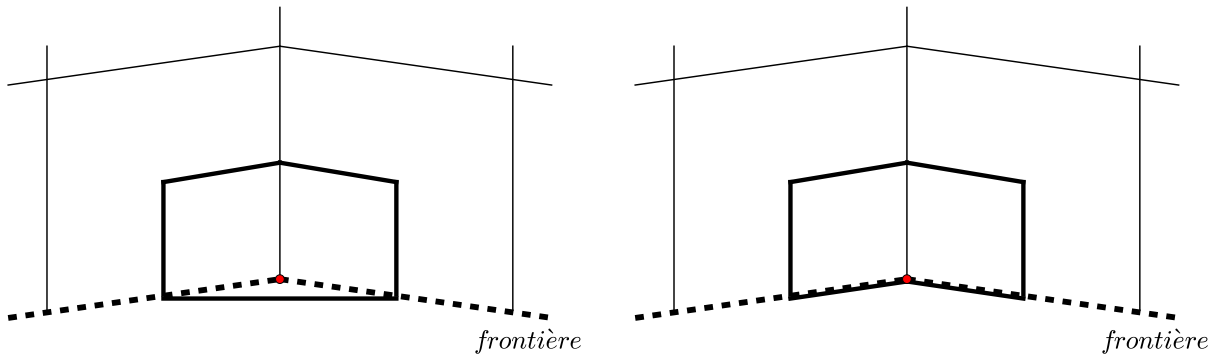


Figure 2.3 Volume de contrôle pour le schéma noeud centré dual

Les flux convectifs et visqueux doivent être calculés maintenant pour chaque partie des faces du volume de contrôle, cependant cette approche est nécessaire seulement pour les schémas



(a) Volume de contrôle reliant deux faces consécutives à la frontière

(b) Redéfinition du volume de contrôle des deux faces consécutives à la frontière

Figure 2.4 Volume de contrôle à la frontière pour le schéma noeud centré dual

d'ordre trois ou quatre (Blazek (2006)). Pour un schéma d'ordre deux, il est suffisant de définir un vecteur normal unitaire moyenné à partir des vecteurs normaux de toutes les parties reliées à une arête en 2-D (p.ex. il y a deux vecteurs normaux liés à l'arête $\overline{(i,j)(i,j+1)}$ sur la Fig. 2.3) ou de toutes les parties reliées à un point-médian en 3-D. Pour chaque face, l'aire est la somme des aires de toutes ses parties. Les valeurs des variables qui définissent les flux sont calculées comme dans le cas du schéma VFCE (voir la section 2.1.1)

2.1.4 VFCE vs VFCS

Dans cette section nous allons faire une comparaison du schéma centré aux éléments et du schéma centré aux sommets avec un volume de contrôle dual. La raison pour laquelle on limite la comparaison à ces deux schémas est que le schéma avec chevauchement des volumes de contrôle est rarement utilisé (Blazek (2006)). Pour comparer ces deux schémas, Blazek (2006) a montré une analyse qui examine certaines caractéristiques telles que la précision, l'effort de calcul numérique, la taille mémoire requise et le traitement aux frontières.

La précision. Un schéma VFCE sur un maillage triangulaire (respectivement tétraédrique) conduit à environ deux (six) fois plus de volumes de contrôle, et par conséquent plus de degrés de liberté qu'un schéma VFCS (voir Baxth (1992)). Sur des maillages mixtes typiques, constitués de prismes et de tétraèdres, un schéma VFCE donne à peu près trois fois plus d'inconnues qu'un schéma VFCS. Cela suggère que les schémas VFCE sont plus précis que les schémas VFCS sur un même maillage. Cependant, les résidus avec un schéma VFCE sont plus faible qu'avec un schéma VFCS, et les résidus plus faibles peuvent altérer la précision.

Ainsi, il n'y a pas de preuve claire montrant quel schéma pourrait être supérieur du point de vue de la précision(Blazek (2006)).

Le traitement aux frontières et l'asymétrie (*skewness*). Le schéma centré aux sommets souffre du problème d'être fortement asymétrique (*skewed*) sur des maillages fortement allongés. Une autre difficulté inhérente aux schémas VFCS est la discrétisation des frontières physiques du domaine tel qu'illustrée à la Fig. 2.4a à gauche ; pour ces cas, une redéfinition du volume de contrôle est montrée à la Fig. 2.4b à droite. Cependant, que se passe-t-il à la frontière alors que le volume de contrôle est tronqué de moitié ? Le résidu, qui doit être lié au centre d'un volume de contrôle, est lié maintenant au nœud qui se situe directement sur la frontière. Cette incohérence entraîne une augmentation de l'erreur de discrétisation par rapport aux schémas VFCE. Un volume de contrôle centré aux sommets provoque également des problèmes en montrant des pics de pression et de densité non physiques là où les volumes présentent des angles très aigus. D'autres complications surviennent dans le cas de frontières périodiques, où les flux des deux parties du volume de contrôle doivent être calculés correctement.

L'effort de calcul numérique. Un schéma VFCE boucle sur les faces de l'élément alors qu'un schéma VFCS dual boucle sur les arêtes d'un élément. Étant donné que l'évaluation des flux à l'interface est très similaire pour les deux schémas, le rapport du nombre de faces au nombre d'arêtes donne le rapport de l'effort de calcul. Ainsi, dans un maillage tétraédrique le schéma VFCE est deux fois plus cher en effort de calcul que le schéma VFCS sur un même maillage (Baxth (1992)). L'approche VFCE devient cependant plus compétitive sur des mailles mixtes contenant des éléments prismatiques. En dehors du traitement aux frontières, les deux méthodes sont équivalentes en ce qui concerne l'effort de calcul numérique sur des maillages hexaédriques.

La mémoire d'ordinateur requise. En ce qui concerne la mémoire requise, en moyenne le schéma VFCE doit stocker six fois plus de variables sur un maillage tétraédrique et environ trois fois plus de variables sur des maillages mixtes par rapport au schéma VFCS. En général, le schéma VFCE a besoin de plus de deux fois plus de mémoire que le schéma VFCS (Blazek (2006)).

En résumé, le schéma VFCS et le schéma VFCE sont numériquement très similaires à l'intérieur d'un champ d'écoulement stationnaire, les différences principales se produisent dans

le traitement aux frontières. Dans ce cas, l'approche VFCE présente des avantages sur le schéma VFCS qui se traduisent par une mise en œuvre plus simple dans un solveur (Blazek (2006)).

2.1.5 Des comparaisons de schémas VFCE et VFCS plus récentes

Dans cette section, nous compléterons la comparaison des schémas VFCE et VFCS en présentant un résumé des travaux de Diskin qui sont plus récents que ceux de Blazek avec des schémas centrés sur les éléments et centrés vers les sommets. Diskin a étudié la précision et la complexité des schémas de discrétisation volumes finis pour les flux visqueux et non visqueux sur des maillages généraux.

Dans l'étude de Diskin and Thomas (2011), une première étape concerne la précision et la performance en comparant les schémas VFCE et VFCS pour les flux visqueux modélisés par une équation de Poisson. Diskin et al. (2010) a considéré différents types de maillages : des maillages de quadrilatères réguliers, des maillages triangulaires structurés réguliers, des maillages triangulaires aléatoires (obtenus en introduisant des petites perturbations sur la position des sommets), et des maillages avec des éléments mixtes (quadrilatères et triangles, où les éléments triangulaires sont obtenus à partir des quadrilatères choisis de façon aléatoire). Les deux dernières catégories sont des maillages irréguliers non-structurés qui ne possèdent pas de patron régulier de connectivité.

Pour cette première étape, une première catégorie de tests implique des solutions manufacturées (technique pour créer des solutions analytiques pour un système d'équations. Voir la Sec. 2.3.1 en ce qui concerne une définition plus détailée des solutions manufacturées) lisses sur des maillages isotropes et sur des maillages fortement anisotropes. L'anisotropie, dans le contexte du maillage, peut être considérée comme la variation de la densité d'un maillage (définie comme le nombre moyen d'éléments du maillage par unité de volume) dépendamment de la direction. Une deuxième catégorie concerne des solutions sur des maillages anisotropes qui ont de fortes variations sur une frontière courbe, typique de ceux rencontrés en simulation d'écoulements turbulents à haut nombre de Reynolds. Quatre schémas considérés d'ordre deux ont été étudiés (voir Diskin et al. (2010)), un schéma VFCS, un schéma de type centré vers les sommets-centré aux éléments (CSCE) (c.-à-d. un schéma centré vers les sommets pondérés, où la valeur à un sommet est la valeur moyenne des valeurs barycentriques voisines qui ont été obtenues à partir d'un schéma centré aux éléments), ainsi que deux schémas

VFCE de reconstruction du gradient de type moindres carrés sur les faces de l'élément avec deux types de molécules de calcul (ou stencil) différents.

La principale conclusion est que la précision et la complexité du schéma VFCS et du meilleur schéma VFCE (incluant le schéma CSCE) sont comparables pour un nombre équivalent de degrés de liberté sur des maillages irréguliers. De plus, du point de point de vue de la complexité, le schéma VFCS a une complexité de stencil plus simple en ce qui concerne les degrés de liberté utilisés pour son calcul. Le schéma CSCE peut présenter des problèmes de convergence si on utilise une coupure sur le gradient. De plus, ce schéma est moins précis sur des maillages fortement anisotropes et sa molécule de calcul est plus complexe. Les résultats de la première catégorie de test indiquent que les schémas VFCE avec une reconstruction du gradient de type moindres carrés, soit le schéma avec pondération CSCE (sans coupure) et le schéma VFCS, montrent une convergence d'ordre deux pour l'erreur de discrétisation par degré de liberté avec des précisions très similaires. Les résultats de la deuxième catégorie ont été plus exigeants. Le schéma VFCS a toujours une convergence de second ordre avec une précision comparable au meilleur des schémas VFCE. Les schémas VFCE avec reconstruction du gradient de type moindres carrés sur les faces ont des stencils plus compacts avec une complexité similaire à la complexité du schéma VFCS. Pour les simulations sur des maillages fortement anisotropes qui suivent une surface courbe, le schéma avec reconstruction du gradient de type moindres carrés doit être modifié pour refléter la direction de forte densité.

Une deuxième étape de l'étude de Diskin and Thomas (2011) concerne le processus de vérification pour le cas de flux non visqueux en utilisant deux schémas qui sont des extensions du schéma VFCS (c.-à-d., des schémas avec des améliorations) et six schémas qui sont des extensions du schéma VFCE, tous théoriquement d'ordre deux. Les schémas ont été comparés par rapport à la complexité, à la précision, et au taux de convergence. Une variété de maillages qui comprend des maillages isotropes de géométries rectangulaires, des maillages anisotropes et des maillages avec des régions anisotropes pour la couche limite, lesquels incluent des perturbations aléatoires de noeuds, ont été étudiés. Un ensemble de résultats très variés dépendant du type de maillage, du type de schéma, de la solution manufacturée utilisée et de la courbure du domaine, entre autres, a été obtenu. Par exemple, dans le cas de maillages anisotropes, tous les schémas peuvent produire une erreur en $O(Ah)$ pour la reconstruction du gradient sur des maillages perturbés (où A représente l'étirement du maillage et h représente la plus grande variation dans le maillage). L'erreur de discrétisation des schémas qui utilisent une reconstruction du gradient de type moindres carrés pondérés diverge dans le processus de raffinement sur les maillages avec de petites perturbations. Pour tous les autres schémas,

l'erreur converge vers l'ordre deux et l'ordre deux est presque idéalement approché si les calculs se font avec le gradient exact. D'autres comparaisons ont été faites parmi les schémas dérivés du schéma VFCS et parmi les schémas dérivés du schéma VFCE. Cependant, une réponse claire qu'un des schémas (VFCE vs VFCS) soit mieux qu'un autre n'a pas été observée.

Ensuite, un survol rapide sera fait sur la discrétisation d'ordre deux du gradient et des opérateurs de convection-diffusion basée sur la méthode VFCE (voir Jasak (1996), Juretic (2004)). VFCE est le schéma par défaut des librairies dans OpenFOAM, néanmoins, OpenFOAM a aussi des structures de données pour stoker des données aux sommets, qui pourraient permettre la création des nouveaux solveurs.

2.2 Discrétisation d'ordre deux du gradient et des opérateurs de convection-diffusion en volumes finis

Discrétisation spatiale du gradient

La discrétisation spatiale du gradient peut être effectuée soit en utilisant le théorème de Gauss (c.-à-d. via une stratégie de discrétisation en volumes finis) ou en utilisant une reconstruction du gradient de type "Least-Square-Fit"(voir Jasak (1996), Juretic (2004)). On considère un volume de calcul V_p, délimité par une surface de contrôle S_p, composée de faces planes f. En faisant l'hypothèse d'une variation linéaire de la variable scalaire ϕ à l'intérieur de l'élément, la discrétisation VFCE d'ordre deux basée sur le théorème de Gauss peut être décrite comme suit :

$$\int_{V_p} \nabla\phi d\tau = \oint_{S_p} \phi d\vec{S} = \sum_f \int_{S_f} \phi d\vec{S} \approx \sum_f \vec{S}_{C_f}\phi_{C_f} \tag{2.3}$$

où ϕ_{C_f} est la valeur de ϕ évaluée au barycentre de la face f. Cette valeur peut être estimée en utilisant une interpolation (cette interpolation est le schéma de convection aux différences centrées) à partir des valeurs ϕ_P et ϕ_N des éléments qui partagent la face où on utilise la notation $P = \vec{x}_P$, $N = \vec{x}_N$ et $C_f = \vec{x}_{C_f}$ (voir Fig. 2.5) :

$$\phi_{C_f} = \alpha\phi_P + (1-\alpha)\phi_N \tag{2.4}$$

où α est un facteur d'interpolation qui est défini comme le ratio de la distance $\overline{C_f N}$ sur la distance $\overline{PN}$, comme ceci :

$$\alpha = \frac{\overline{C_f N}}{\overline{PN}} \tag{2.5}$$

Puisqu'on a fait l'hypothèse d'une variation linéaire de ϕ sur le volume de contrôle V_P, une approximation VFCE du gradient d'ordre deux est calculée comme suit :

$$(\nabla\phi)_P = \frac{1}{V_p}\sum_f \vec{S}_f \phi_{C_f} \tag{2.6}$$

Cette discrétisation génère le schéma de Gauss de OpenFOAM pour la reconstruction du gradient (voir Jasak (1996)).

L'approche de la méthode des moindres carrés, quant à elle, consiste à minimiser la fonctionnelle suivante par rapport au gradient (voir Juretic (2004)) :

$$L = \sum_f \left(\frac{\phi_N - \phi_P}{|\vec{d}|} - \frac{\vec{d}\cdot(\nabla\phi)_P}{|\vec{d}|} \right)^2 \tag{2.7}$$

où $\vec{d} = P\vec{N}$ est le vecteur qui fait la liaison entre le barycentre P d'un élément et le barycentre de l'élément voisin N. L'expression numérique :

$$\frac{\phi_N - \phi_P}{|\vec{d}|} \tag{2.8}$$

dans l'Eq. 2.7 représente une approximation linéaire du gradient de ϕ projeté sur le vecteur $\vec{d}$ (voir Fig. 2.5).

Le gradient reconstruit au point P $(\nabla\phi)_P$, défini comme le gradient qui minimise l'équation Éq. 2.7, peut être calculé en résolvant le système d'équations linéaires produit par la condition de minimisation suivante :

$$\frac{dL}{d(\nabla\phi)_P} = 0 \tag{2.9}$$

où la valeur qu'on obtient du gradient $(\nabla\phi)_P$ est une approximation numérique d'ordre deux.

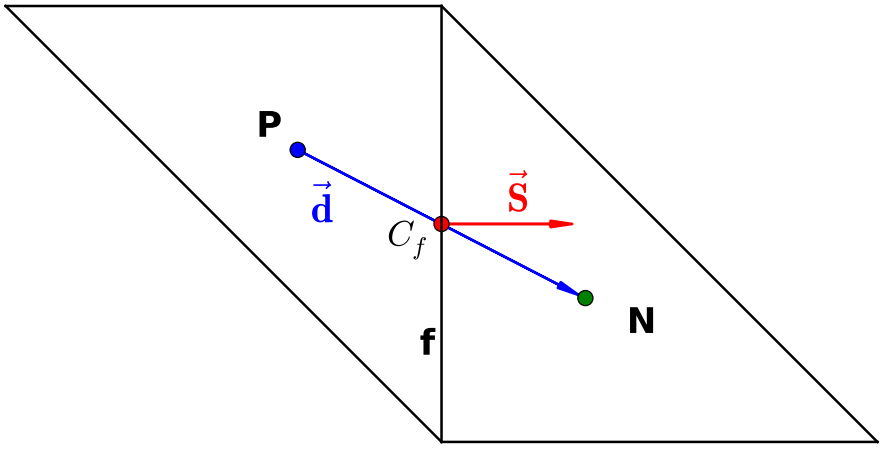


Figure 2.5 Une face non orthogonale entre deux éléments

Terme de Convection

Une discrétisation en volumes finis du terme de convection d'une variable scalaire ϕ, s'obtient en appliquant le théorème de Gauss sur un volume de contrôle V_p, délimité par une surface de contrôle S_p, et par la suite en utilisant les valeurs au barycentre C_f pour discrétiser les intégrales sur les faces f qui délimitent le volume de contrôle (voir Jasak (1996)) :

$$\begin{aligned} \int_{V_p} \nabla \cdot (\rho\vec{v}\phi)\, d\tau &= \oint_{S_p} d\vec{S} \cdot (\rho\vec{v}\phi) = \sum_f \int_{S_f} d\vec{S} \cdot (\rho\vec{v}\phi) \\ &\approx \sum_f \vec{S} \cdot (\rho\vec{v}\phi)_{C_f} = \sum_f \vec{S} \cdot (\rho\vec{v})_{C_f} \phi_{C_f} \\ &= \sum_f F\phi_{C_f} \end{aligned} \tag{2.10}$$

où $\vec{v}$ représente le champ de vitesse, ρ la densité volumique et F est le flux de masse à travers la face f : $F = \vec{S} \cdot (\rho\vec{v})_{C_f}$. Cette discrétisation définit le schéma de Gauss d'OpenFOAM d'ordre deux pour le terme de convection.

La valeur de ϕ_{C_f} au centre de la face f de l'expression à droite de l'Éq. 2.10 est calculée à partir des valeurs de ϕ dans les barycentres **P** et **N** des cellules qui partagent cette face. L'ensemble des cellules nécessaires pour calculer le schéma du terme de convection (à droite de l'Éq. 2.10) définissent sa "molécule" de calcul. Par exemple, étant donné un maillage triangulaire comme celui illustré à la Fig. 2.6, chaque valeur ϕ_{C_f} est calculée à partir de la valeur de ϕ dans les barycentres **P** et **N**.

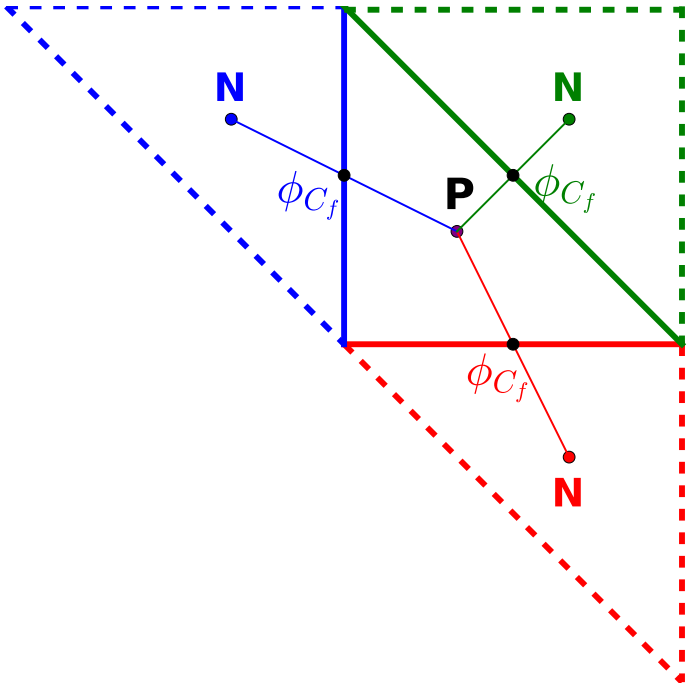


Figure 2.6 Molécule de calcul pour le schéma du terme de convection.

Terme de diffusion

De la même façon que dans le cas de l'opérateur de convection, en appliquant le théorème de Gauss sur un volume de contrôle V_p et par la suite en utilisant les valeurs au barycentre des faces pour discrétiser les intégrales de surface, on obtient le schéma de Gauss pour le Laplacien d'OpenFOAM (voir Jasak (1996)) :

$$\begin{aligned} \int_{V_p} \nabla \cdot (\rho\Gamma\nabla\phi)\, d\tau &= \oint_{S_p} d\vec{S} \cdot (\rho\Gamma\nabla\phi) = \sum_f \int_{S_f} d\vec{S} \cdot (\rho\Gamma\nabla\phi) \\ &\approx \sum_f (\rho\Gamma)_{C_f} \left(\vec{S}_{C_f} \cdot \nabla\phi_{C_f}\right) \end{aligned} \tag{2.11}$$

où Γ est le coefficient de diffusion, qui est considéré constant dans ce travail.

Le gradient $\nabla\phi_{C_f}$ au centre de la face f de l'expression à droite de l'Éq. 2.11 doit être ap-

proximé de manière à garantir un ordre deux de convergence. Une approche est d'approximer le gradient à la face f en interpolant sa valeur à partir des valeurs stockées au centre des éléments voisins :

$$\nabla\phi_{C_f} = \alpha\nabla\phi_P + (1-\alpha)\nabla\phi_N \tag{2.12}$$

où α est un facteur d'interpolation (voir Éq. 2.5). Ce schéma a une molécule de calcul lourde (voir Jasak (1996)). Étant donné un maillage triangulaire comme celui illustré à la Fig. 2.7, si chaque valeur du gradient $\nabla\phi_{C_f}$ est calculée en interpolant sa valeur utilisant l'Éq. 2.12, une molécule composée de deux couches, comme celui illustré à la Fig. 2.7, est nécessaire. Dans le cas d'un maillage en 3D, la molécule de calcul est plus lourde encore.

Comme il sera décrit à la section 2.2.1, OpenFOAM utilise un schéma de type correction différée (deferred-correction approach en anglais) composé de deux termes qui est considéré comme moins lourd. Un premier terme approche le gradient en utilisant une molécule plus simple, les valeurs aux centres P et N des deux éléments qui partagent une même face f, et un deuxième terme pour fournir une correction au gradient qui est appelé la contribution non orthogonale. Le traitement de la non-orthogonalité n'est pas une propriété applicable à tous les schémas formulés à partir de la discrétisation du schéma centré aux éléments décrits par l'Éq. 2.11, mais plutôt un besoin du schéma introduit dans OpenFOAM pour éviter une molécule de calcul lourde. Nous allons l'examiner ce schéma plus tard à la section 2.2.1.

Terme Source

Dans le cadre de ce travail, le terme source $q = q(\vec{x})$ s'obtient en implémentant des solutions manufacturées pour des fins de vérification seulement, c'est pourquoi q dépend seulement des coordonnées géométriques. Une approximation d'ordre deux est obtenue en intégrant sur le volume de contrôle V_P comme ceci :

$$\int_{V_p} q(\vec{x})d\tau \approx q_P V_P \tag{2.13}$$

où $q_P = q(\vec{x}_P)$ est le terme source évalué au barycentre P du volume de contrôle V_P.

En utilisant les Eqs. 2.10, 2.11 et 2.13, une discrétisation de deuxième ordre pour les opérateurs de convection-diffusion et du terme source est formulée ainsi :

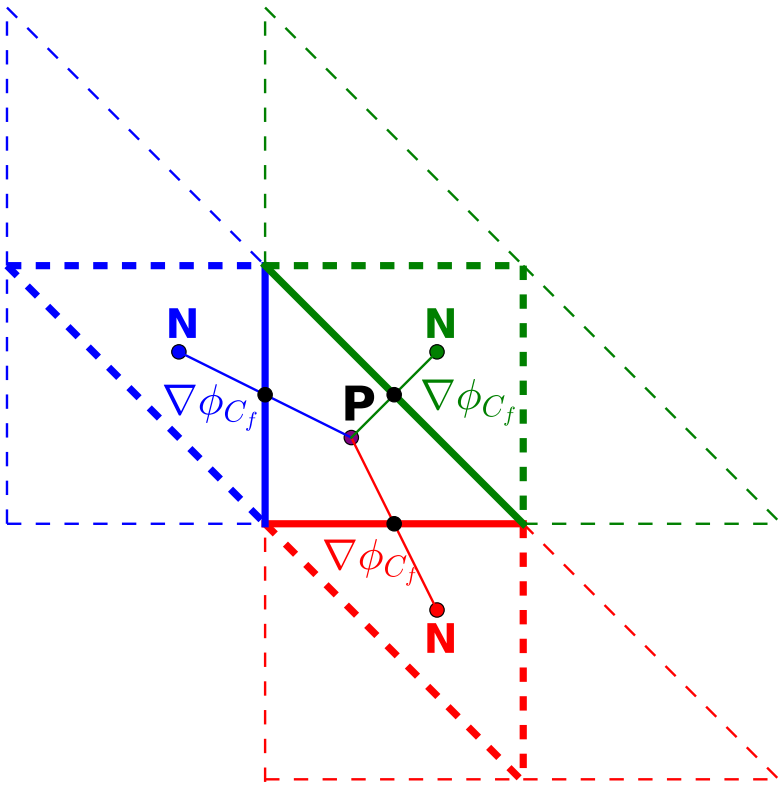


Figure 2.7 Molécule de calcul pour le schéma du terme de diffusion.

$$\underbrace{\sum_f F\phi_{C_f}}_{\text{terme de convection}} \underbrace{-\sum_f \vec{S}_{C_f}\cdot(\rho\Gamma\nabla\phi)_{C_f}}_{\text{terme de diffution}} = \underbrace{q_P V_P}_{\text{terme source}} \tag{2.14}$$

L'équation 2.14 produit une équation algébrique d'ordre deux au centre P de chaque élément (Jasak (1996)) pouvant se mettre sous la forme :

$$a_P\phi_P + \sum_N a_N\phi_N - R_P = 0 \tag{2.15}$$

où a_P, a_N and R_P sont de coefficients obtenus par les schéma de discrétisation spatiale utilisés pour approximer ϕ_{C_f} et $(\nabla\phi)_{C_f}$.

Nous pouvons assembler un système d'équations algébriques pour chaque nœud P dans l'Eq. 2.15 :

$$[A]\,[\phi] = [R] \tag{2.16}$$

Ce système d'équations algébriques (Eq. 2.16) produit une approximation numérique de deuxième ordre pour la variable ϕ à chaque nœud de calcul.

2.2.1 Le traitement de la non-orthogonalité dans OpenFOAM

Pour calculer le gradient $\nabla\phi_{C_f}$ aux faces d'un élément de calcul, un schéma centré composé de deux termes a été utilisé par Jasak (1996) qui est généralement considéré le contributeur majeur de l'outil OpenFOAM. Un premier terme approche le gradient en utilisant les valeurs aux centres P et N des deux éléments qui partagent une même face f (voir Fig. 2.8). Ce premier terme a été appelé la contribution orthogonale. Un deuxième terme est ensuite ajouté pour fournir une correction au gradient, qui est appelé la contribution (la correction) non orthogonale. L'expression mathématique de cette idée est présentée ainsi :

$$\vec{S}\cdot(\nabla\phi)_f = \underbrace{|\vec{\Delta}|\frac{\phi_N-\phi_P}{|\vec{d}|}}_{\text{contribution orthogonale}} + \underbrace{\vec{k}\cdot(\nabla\phi)_f}_{\text{contribution non orthogonale}} \tag{2.17}$$

où f désigne la face qui est partagée par deux volumes de contrôle et $(\nabla\phi)_f$ le gradient de ϕ. Les vecteurs $\vec{\Delta}$ et $\vec{k}$ sont reliés par la contrainte :

$$\vec{S} = \vec{\Delta} + \vec{k} \tag{2.18}$$

où le vecteur $\vec{\Delta}$ est choisi comme étant parallèle au vecteur $\vec{d}$ (voir Fig. 2.8)

Cette approche consistant à décomposer le gradient correspond à un schéma de type correction différée (deferred-correction approach en anglais), qui, comme cité dans Ferziger and Peric (2002), a déjà été utilisée par Muzaferija (1994).

Parmi les choix possibles pour l'Éq. 2.18, dans sa thèse, Jasak (1996) a considéré trois façons de calculer le vecteur $\vec{\Delta}$. Chacune d'elles permet de calculer le vecteur de correction $\vec{k}$ en utilisant l'équation 2.18. L'idée commune à ces trois approches est de décomposer le vecteur

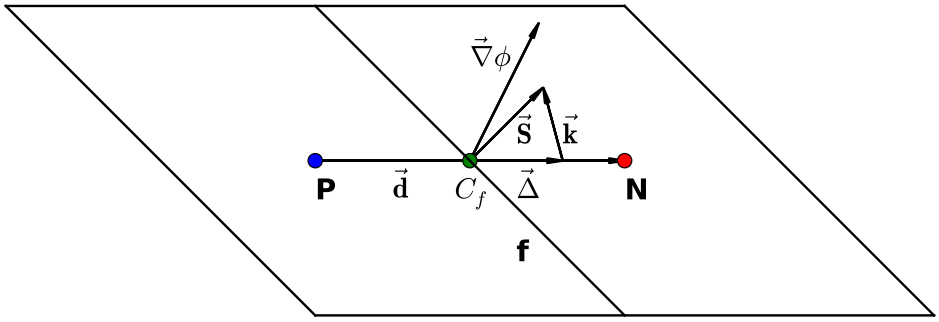


Figure 2.8 Vecteur de correction non orthogonale $\vec{k}$

normal à la surface $\vec{S}$ au centre de la face f. Ces trois manières de calculer la correction non orthogonale sont illustrées dans les Figs. 2.9, 2.10 and 2.11. Elles sont appelées approche de "correction orthogonale", "Correction minimale" et approche "Sur-relaxée" respectivement (en anglais Over-relaxed approach) (voir Jasak and Gosman (2000)) :

— Pour la "Correction minimale" (voir Fig. 2.9) le calcul est effectué de manière à maintenir la correction non orthogonale aussi petite que possible :

$$\vec{\Delta} = \cos(\alpha_N)|S|\vec{d_u} \tag{2.19}$$

— Pour la "Correction orthogonale" (voir Fig. 2.10) la contribution de ϕ_P et ϕ_N est maintenue comme dans le cas des maillages orthogonaux :

$$\vec{\Delta} = |S|\vec{d_u} \tag{2.20}$$

— Pour l'approche "Sur-relaxée" (voir Fig. 2.11), l'importance des termes ϕ_P et ϕ_N augmente avec la non-orthogonalité :

$$\vec{\Delta} = \frac{|S|}{\cos(\alpha_N)}\vec{d_u} \tag{2.21}$$

où le vecteur $\vec{d_u} = \frac{\vec{d}}{|d|}$ est le vecteur unitaire $\vec{d} = P\vec{N}$; $|S|$ l'aire de la face f, et $\cos(\alpha_N)$ est le cosinus de l'angle entre le vecteur unitaire $P\vec{N}$ et $\vec{n}$, le vecteur unitaire normal à la face.

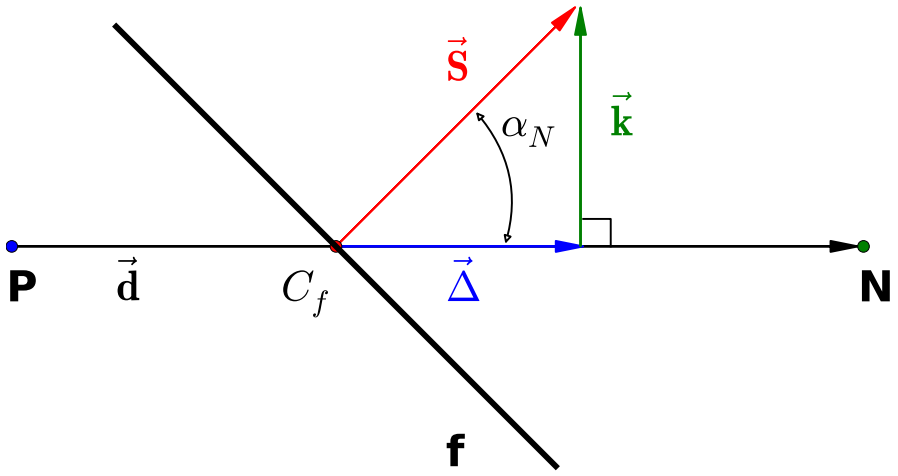


Figure 2.9 Approche “Correction minimale” pour la correction non orthogonale

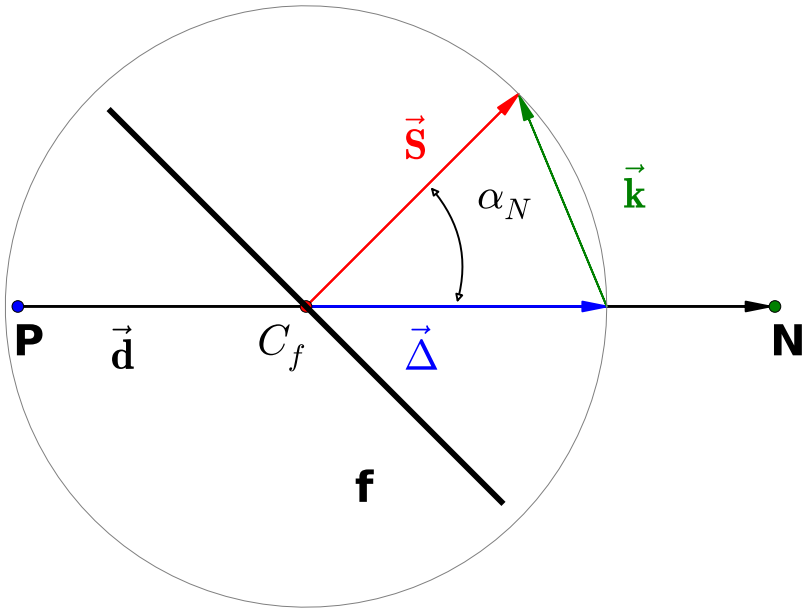


Figure 2.10 Approche “Correction orthogonale” pour la correction non orthogonale

Enfin, il faut remarquer deux choses. Premièrement, l’Éq. 2.17 est non-linéaire en raison de la contribution non orthogonale, de sorte que dans OpenFOAM un processus itératif est mis en œuvre pour obtenir la valeur de la projection du gradient sur la face. Deuxièmement,

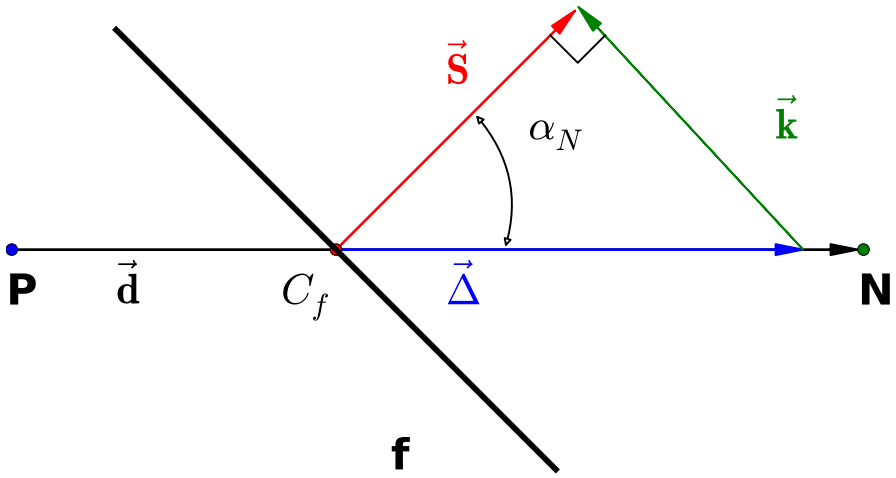


Figure 2.11 Approche “Sur-relaxée” pour la correction non orthogonale

dans OpenFOAM l’approche “Sur-relaxée” semble être la seule implémentée. Une contrainte λ sur la partie non orthogonale est mise en œuvre dans le code pour limiter la croissance du module de la partie non orthogonale par rapport à la partie orthogonale quand l’angle α_N croît :

$$\vec{S} \cdot (\nabla\phi)_f = |\vec{\Delta}| \frac{\phi_N - \phi_P}{|\vec{d}|} + \lambda \left(\vec{k} \cdot (\nabla\phi)_f \right) \tag{2.22}$$

2.3 Vérification des codes numériques

Dans le calcul scientifique, l’objectif de la vérification du code est de s’assurer que le code est une représentation fidèle du modèle mathématique sous-jacent. Le modèle mathématique est constitué d’équations aux dérivées partielles ou d’équations intégrales avec des conditions initiales, des conditions aux limites et éventuellement d’autres contraintes. La vérification d’un code considère donc à la fois l’exactitude des algorithmes numériques choisis et l’exactitude de l’implémentation de ces algorithmes dans le code source (c.-à-d., il faut s’assurer qu’il n’y ait pas d’erreur de codage ou “bugs”, voir Knupp and Salari (2003)).

Quelques définitions généralement acceptées de la vérification de code d’équations aux dérivées partielles, sont les suivantes :

— La justification (ou corroboration) qu'un modèle informatisé représente un modèle conceptuel dans les limites spécifiées de précision (Society for Computer Simulation (SCS, 1979)).

— Le processus de détermination qu'une mise en œuvre du modèle représente avec précision la description conceptuelle du développeur du modèle et la solution du modèle (American Institute of Aeronautics and Astronautics (AIAA), Guide for the Verification and Validation of Computational Fluid Dynamics Simulations (AIAA, 1998)).

— Le processus de détermination du fait que le modèle de calcul représente avec précision le modèle mathématique sous-jacent et sa solution. (American Society of Mechanical Engineers (ASME), Guide for Verification and Validation in Computational Solid Mechanics (ASME, 2006)).

Selon Oberkampf and Roy (2010), la vérification du code numérique peut être séparée en deux parties : la vérification des algorithmes numériques et l'assurance qualité des logiciels (SQA - acronyme anglais "sofware quality assurance").

La vérification des algorithmes numériques, quant à elle, traite l'exactitude mathématique de la mise en œuvre (dans un logiciel) de tous les algorithmes numériques qui affectent la précision numérique des résultats de calcul. L'objectif principal de la vérification des algorithmes numériques est d'accumuler des preuves qui montrent que les algorithmes numériques sont mis en œuvre correctement et qu'ils fonctionnent comme prévu. Par exemple, la vérification des algorithmes numériques devrait montrer qu'étant donnée une méthode de discrétisation spatiale pour une équation aux dérivées partielles, l'algorithme produit le taux de convergence attendu au fur et à mesure que le maillage est raffiné. La vérification des algorithmes numériques est fondamentalement empirique. Plus précisément, elle est basée sur des tests, des observations, des comparaisons et des analyses de résultats pour les exécutions individuelles du code. Elle se concentre sur des investigations minutieuses des aspects numériques, tels que les taux de convergence spatiale et temporelle, la convergence spatiale en présence de discontinuités, l'indépendance des solutions d'avec les transformations de coordonnées, et les tests de symétrie liés à divers types de conditions aux limites. L'analyse d'erreur analytique ou d'erreur formelle est insuffisante dans la vérification des algorithmes numériques parce que le code lui-même doit démontrer les résultats analytiques et formels de l'analyse numérique. La vérification des algorithmes numériques est généralement réalisée en comparant la solution numérique avec des solutions très précises, qui sont (communément) appelées vérifications de

référence (verification benchmarks en anglais).

Dans le cas de l'assurance qualité des logiciels, le but est de déterminer que le code, dans le cadre d'un système logiciel, est mis en œuvre correctement et qu'il produit des résultats reproductibles sur un matériel (hardware) et dans un environnement logiciel déterminé. L'assurance qualité des logiciels, quant à elle, consiste en des pratiques, des procédures et des processus qui sont principalement développés par les chercheurs, les ingénieurs informatiques et les communautés du génie logiciel. L'analyse de la qualité des logiciels et les méthodes de tests peuvent être divisées de la façon suivante : analyse statique, tests dynamiques, et analyse formelle. Les tests dynamiques peuvent être divisés en des éléments tels que les tests de régression, les tests à boîte noire, et les tests à boîte blanche.

2.3.1 La méthode des solutions manufacturées

La méthode des solutions manufacturées (MMS - acronyme anglais *method of manufactured solutions*) est une approche très puissante et générale pour créer des solutions exactes pour un système d'équations aux dérivées partielles. Plutôt que d'essayer de trouver une solution exacte au système d'équations aux dérivées partielles avec des conditions initiales et avec des conditions aux limites, l'objectif est de "fabriquer" une solution exacte pour une équation légèrement modifiée (voir Knupp and Salari (2003)). Pour la vérification de code, il n'est pas vraiment nécessaire que la solution fabriquée soit liée à un problème physique réaliste. Il ne faut pas oublier que la vérification ne concerne que la mathématique d'un problème donné. La méthode MMS nécessite la solution à un problème inverse : étant donné un ensemble d'équations et une solution choisie, trouver un ensemble d'équations modifiées qui satisfassent cette solution. L'implémentation pratique de cette méthode appliquée aux équations de Navier-Stokes et à l'équation de Poisson sera montrée au chapitre 3 dans ce travail.

Bien que l'approche MMS pour générer des solutions exactes ne soit pas nouvelle (voir p. ex. Zadunaisky (1976) et Stetter (1978)), Roache and Steinberg (1984) et Steinberg and Roache (1985) semblent être les premiers à utiliser ces solutions exactes aux fins de vérification de code. Leur travail a examiné le taux de convergence asymptotique de l'erreur de discrétisation avec raffinement de maillage systématique. Shih (1985) a développé, de façon indépendante, une procédure similaire pour le débogage des codes de calcul scientifique, mais sans employer le raffinement de maillage pour évaluer la convergence ou l'ordre de précision. Les concepts derrière la méthode MMS pour des propos de vérification de code ont ensuite été perfectionnés par Roache et al. (1990) et Roache (1998).

Le terme de “solution manufacturée” a été introduit par Oberkampf et al. (1995) et il se réfère au fait que la méthode génère (ou fabrique) un ensemble d'équations analytiques liées à la solution choisie. Une étude approfondie des solutions manufacturées pour la vérification de code est présentée par Knupp and Salari (2003) et comprend des détails sur la méthode ainsi que sur l'application à une variété d'équations aux dérivées partielles. Une discussion de fond, des définitions et des descriptions pour certains termes liés à la qualité de développement en CFD ont été proposées par Roache (1998) et par la suite complétés par Stern et al. (2006).

La vérification a été décrite comme la résolution des équations de façon correcte et la validation comme la résolution des équations appropriées. Différents aspects abordés dans ce travail comprennent la distinction d'une part entre la vérification du code et la validation de code, et d'autre part entre la vérification du code et la vérification des calculs, la convergence du maillage par rapport à la convergence itérative, et l'erreur numérique par rapport à l'erreur de modélisation conceptuelle. Stern et al. (2006) décrivent un ensemble de procédures de vérification, de validation et de méthodologies de certification pour les simulations numériques. Par exemple, des corroborations quantitatives du modèle RANS (acronyme anglais “Reynolds average Navier-Stokes”) ont été présentées en hydrodynamique navale.

Il n'est pas rare d'utiliser la méthode MMS pour la vérification de codes de calcul hydrodynamique (voir par exemple, Roache et al. (1990), Pelletier et al. (2004), Roy et al. (2004a), Eca et al. (2007a)). Abanto et al. (2005) ont présenté une étude de convergence spatiale sur certains cas CFD atypiques en utilisant un certain nombre de logiciels de CFD commerciaux. Leurs tests de vérification ont la particularité que les solutions exactes sont connues. Ils incluent le classique écoulement de Poiseuille, un flux de re-circulation incompressible, une solution manufacturée pour simuler l'écoulement de la couche limite laminaire incompressible, et un écoulement annulaire incompressible. Différents taux de convergence ont été obtenus selon qu'on utilisait des maillages structurés ou non structurés.

Tremblay et al. (2006) ont appliqué la méthode des solutions manufacturées pour la vérification du code à des interactions fluide-structure. Ils ont observé que lorsque la méthode MMS est utilisée avec un raffinement systématique du maillage, elle devient un outil très puissant de vérification. Roy et al. (2004b) ont utilisé la méthode MMS pour vérifier l'ordre de précision d'un code de volumes finis pour les équations d'Euler et de Navier-Stokes en évaluant avec précision l'erreur de discrétisation des solutions numériques. Une convergence

spatiale d'ordre deux a été observée pour une approche centrée aux sommets en utilisant des maillages non structurés.

La recherche sur les solutions manufacturées pour les équations de Navier-Stokes et la turbulence incompressibles en 2-D comprend les articles publiés par Eca and Hoekstra (2006, 2007); Eca et al. (2007b). Une étude concernant la vérification d'un code CFD en volumes finis non structurés a été publiée par Veluri and Roy (2010). La méthode MMS a été utilisée pour générer des solutions exactes des équations d'Euler et de Navier-Stokes afin de vérifier l'ordre de convergence du code sur différents types de maillages en 2-D et en 3-D. Ils ont vérifiés différents modèles, parmi lesquels les équations de mouvement stationnaires, des modèles de transport, des modèles de turbulence, des conditions aux limites et des équations non stationnaires.

Une étude plus récente de vérification pour des maillages structurés utilisant l'outil OpenFOAM a été publiée par Blais and Bertrand (2015). La méthode MMS a été utilisée pour vérifier un solveur des équations Navier-Stokes volume moyen (les équations VANS en anglais) lesquelles sont un élément clé dans de nombreux modèles utilisés pour étudier des problèmes complexes comme les écoulements dans les médias poreux ou les écoulements multiphasiques. Un cadre pour concevoir des solutions analytiques et pour vérifier des codes qui résolvent les équations VANS utilisant la méthode des solutions manufacturées est présenté dans ce travail.

Une description des étapes principales lors de l'implémentation d'une solution manufacturée est présentée ainsi (voir Knupp and Salari (2002)) :

— Proposer une solution manufacturée pour le système d'équations initiales.
— Obtenir un terme source pour les équations initiales.
— Insérer le terme source comme un terme supplémentaire dans le solveur.
— Résoudre la nouvelle équation.

2.3.2 Normes d'erreur numérique

Pour effectuer la vérification de code, nous comparons la solution numérique à une solution de référence, de préférence une solution analytique du modèle mathématique ou une solution manufacturée. On peut calculer la différence entre la sortie numérique et la solution de référence en utilisant une norme définie sur l'ensemble du domaine. Pour l'erreur moyenne,

on utilise la norme L_1 :

$$||e||_{L_1(\Omega)} = \frac{1}{|\Omega|} \int_{\Omega} |e| \, d\Omega \tag{2.23}$$

alors que pour l'erreur maximale on utilise la norme L_∞ qui retourne la valeur maximale sur l'ensemble du domaine :

$$||e||_{\infty} = \sup_{x \in \Omega} |e(x)| \tag{2.24}$$

Parmi ces deux normes, la mesure d'erreur maximale absolue est généralement la plus sensible. D'autres normes pour calculer l'erreur moyenne, p. ex. la norme euclidienne L_2, ne seront pas utilisées dans le processus de vérification. Le choix de la norme L_1 peut-être justifié par l'inéquation :

$$||e||_{L_p} \leq ||e||_{L_1} \tag{2.25}$$

où la convergence de la norme L_1 implique la convergence des normes L_p ($p = 2, 3, ..$). En effet, pour justifier l'Éq. 2.25 nous pouvons envisager un domaine discrétisé :

$$||e||_{L_p} = \left[\sum_{i=1}^{n} |e_i|^p \right]^{\frac{1}{p}} \leq \left[\sum_{i=1}^{n} |e_i| \right] = ||e||_{L_1} \tag{2.26}$$

Si une représentation de la solution analytique est disponible, l'erreur exacte $e = \tilde{\phi} - \phi_h$ locale dans un élément noté i, est la différence entre la solution de référence $\tilde{\phi}$ et la solution numérique ϕ_h d'une variable scalaire ϕ :

$$e_i = \tilde{\phi}_i - \phi_{hi} \tag{2.27}$$

Dans un domaine discrétisé, la norme pour l'erreur moyenne (Éq. 2.23) devient :

$$||e||_{L_1(\Omega)} \approx \frac{1}{|\Omega|} \sum_{i=1}^{N} \omega_i \, |e_i| \tag{2.28}$$

où ω_i est le volume de contrôle de l'élément i. La norme d'erreur maximale (Éq. 2.24) est approchée par :

$$||e||_\infty \approx \max_i |e_i|, \quad i = 1, ..., N \tag{2.29}$$

Dans ce travail, on utilise des domaines à deux dimensions, des parallélogrammes, qu'on maille uniformément en divisant chaque côté N_k fois :

$N_k = 10, 20, 40, 60, 80, 100.$

Si e_k est l'erreur globale produite en partitionnant chaque côté N_k fois, le taux de convergence peut être calculé comme :

$$R_k = \frac{\log(e_k/e_{k-1})}{\log(N_k/N_{k-1})} \tag{2.30}$$

2.3.3 Résumé de la revue de la littérature

— En termes de précision attendue, les schémas VFCS et VFCE présentent chacun des avantages et des inconvénients, mais globalement tous peuvent être conçus afin d'obtenir au moins une convergence théorique d'ordre deux.

— La méthode de solutions manufacturées est un outil puissant pour vérifier un code numérique.

— Malgré la très grande popularité du code à sources ouvertes OpenFOAM dans le milieu académique, il y a un manque de travaux de vérification sur des maillages complexes.

CHAPITRE 3 SYNTHÈSE DE L'ENSEMBLE DU TRAVAIL

Ce chapitre fait la synthèse des contributions de cette thèse. Il est organisé en deux thèmes. Le premier thème correspond à la présentation d'une méthode de vérification qui amène à détecter l'origine d'une dégradation de la convergence dans les simulations numériques où sont utilisés des maillages non orthogonaux. Le deuxième thème prouve d'abord, à l'aide de l'erreur de troncature, qu'une correction est nécessaire pour les conditions aux limites de types Dirichlet et Neumann. Ensuite, différentes façons d'implémenter une correction non orthogonale pour atteindre la convergence d'ordre deux ainsi que les résultats de ces implémentations sont documentés. Les détails précis concernant la méthodologie à suivre ne sont pas donnés dans ce chapitre. Pour plus de détails, les articles peuvent être consultés dans l'annexe de la thèse.

3.1 Thème 1 : Processus de détection de la source de dégradation de la convergence.

Premier objectif : Vérifier les opérateurs de convection-diffusion disponibles dans les bibliothèques OpenFOAM et détecter la source de dégradation de la convergence sur des maillages non orthogonaux.

L'article suivant répond à ce premier objectif :

Noriega, Guibault, Reggio, and R. ("forthcoming"a). "A Case-Study in Open-Source CFD Code Verification.
Part I : Convergence Rate Loss Diagnosis". (Voir Annexe B)

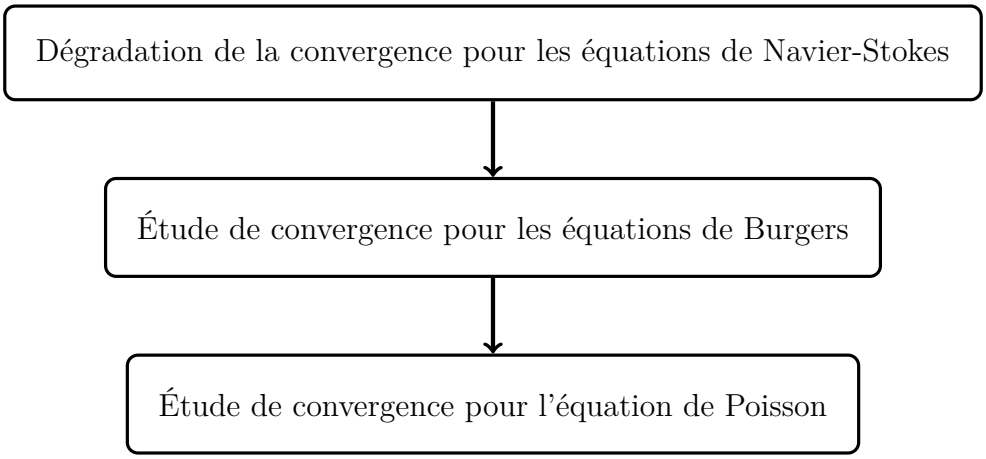
3.1.1 Méthodologie de vérification

Le point de départ de ce travail a été la détection d'une dégradation de la convergence lors de simulations numériques des équations de Navier-Stokes avec des solutions analytiques sur des maillages non orthogonaux. Dû à la complexité de mettre en œuvre un processus de vérification pour ces équations, une stratégie pour détecter l'origine du problème est implémentée dans la première partie de travail. Cette stratégie prend en compte les équations, les types de conditions aux limites ainsi que les critères de qualité des maillages.

Diminution de la complexité du solveur

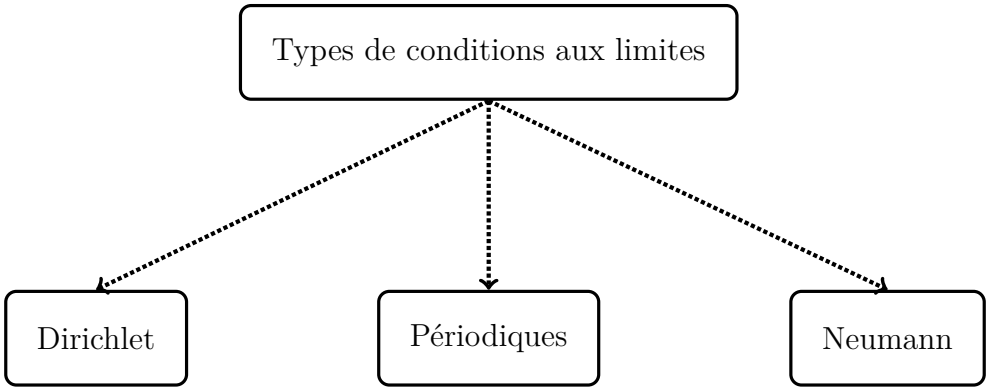
Une réduction de la complexité du solveur et du nombre de termes de l'équation aux dérivées partielles permet de diminuer le nombre des schémas numériques impliqués, et par conséquent d'isoler la source potentielle du problème.

Une réduction du nombre de termes des équations de Navier-Stokes est obtenue en choisissant les équations de Burgers pour vérifier les opérateurs de convection-diffusion seulement. Par la suite, après avoir détecté le même problème avec les équations de Burgers, l'équation de Poisson a été ciblée pour vérifier l'opérateur de diffusion seulement :



Types de conditions aux limites

L'analyse des conditions aux limites est d'une grande importance puisque, en absence de forces externes, ce sont les conditions aux limites qui déterminent la physique d'un problème à un moment donné. Dû à la complexité de l'outil de calcul OpenFOAM, pour lequel il existe plusieurs dizaines de types de conditions aux limites, la plupart étant dérivée de quelques types de base, nous nous limitons aux conditions de Dirichlet, Neumann et périodiques (cycliques dans la terminologie d'OpenFOAM) :



Types de Maillage

Bien qu'on ait déterminé très rapidement que le problème apparaissait lorsque des maillages non orthogonaux étaient utilisés, il fallait d'abord vérifier si d'autres propriétés que la non-orthogonalité pouvaient être à la source de la dégradation de la convergence. Juretic (2004) a défini trois propriétés pour la qualité du maillage :

— **La non-orthogonalité.** Un maillage orthogonal est défini comme un maillage pour lequel le vecteur $\vec{d}$, qui joint le centre P d'une cellule au centre N d'une cellule voisine, est parallèle au vecteur normal ($\vec{S}$) de la face f que partagent les deux cellules (voir Fig. 3.1). Un maillage qui ne satisfait pas cette condition est défini comme non orthogonal (voir Fig. 3.2).

— **L'asymétrie (skewness en anglais).** Si le segment $\overline{PN}$ n'intercepte pas le barycentre C_f de la face f, le maillage est défini comme asymétrique (voir Fig. 3.3).

— **L'uniformité.** Un maillage est dit uniforme lorsque le milieu de $\overline{PN}$ est sur la face f. La Fig. 3.4 montre une face f qui ne satisfait pas cette propriété.

Pour le processus de vérification, nous avons choisi trois types de maillages orthogonaux et cinq types de maillages non orthogonaux (voir l'Annexe A) qui permettent de vérifier les trois propriétés recommandées par Juretic. La vérification utilisant des maillages orthogonaux a été faite en utilisant les maillages suivants :

Hexaédrique. Maillage ayant des cellules rectangulaires identiques (voir Fig. A.1).

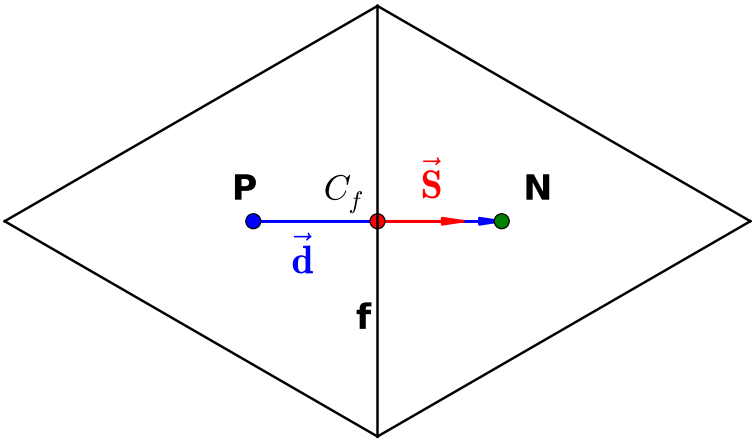


Figure 3.1 Maillage orthogonal, symétrique et uniforme

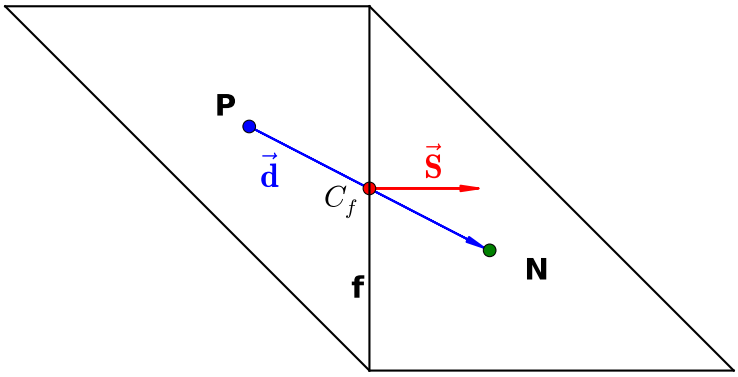


Figure 3.2 Maillage simplement non orthogonal (symétrique et uniforme)

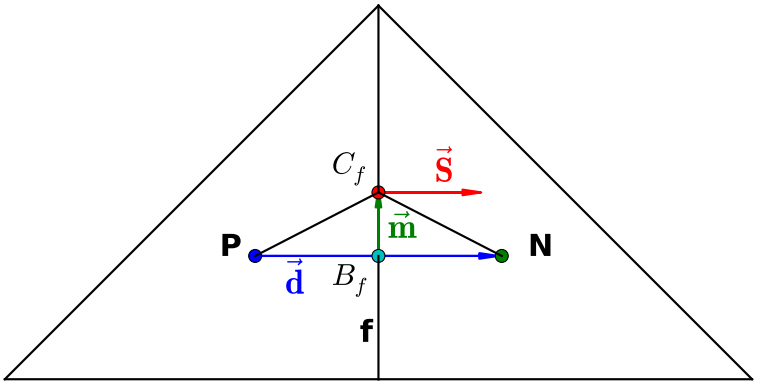


Figure 3.3 Maillage asymétrique (orthogonal et uniforme)

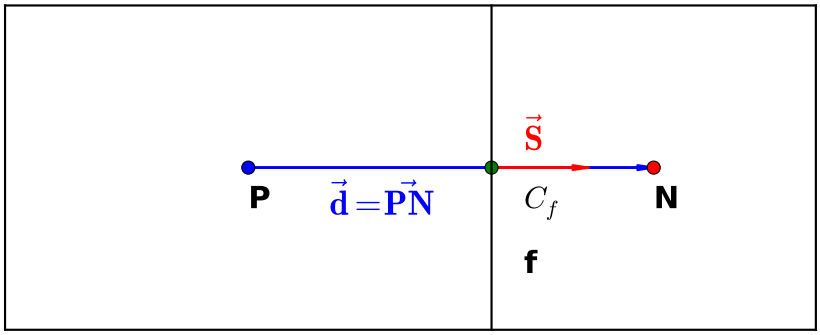


Figure 3.4 Maillage orthogonal non uniforme

Prismes équilatéraux. Maillage ayant des cellules triangulaires orthogonales (voir Fig A.2).

Allongé. Maillage ayant des cellules rectangulaires avec des poids non symétriques, cela pour tester la non-uniformité (voir Fig A.3).

Dans le cas de maillages non orthogonaux, nous avons utilisé des maillages (simplement) non orthogonaux uniformes et symétriques, ainsi que des maillages non orthogonaux avec asymétrie et des maillages non orthogonaux non uniformes (notons que l'uniformité ainsi que l'asymétrie peuvent être mesurées sur des maillages orthogonaux ou non orthogonaux) :

Left. Un maillage non orthogonal composé de prismes orientés vers la gauche (voir Fig. A.4).

Alternate. Un maillage non orthogonal à la frontière, composé de prismes orientés alternés (voir Figs. A.5). Ces cellules sont orthogonales (sauf aux faces à la frontière) avec asymétrie.

Delaunay. Un maillage non orthogonal de type Delaunay composé de prismes (voir Fig. A.6).

Inclined. Un maillage non orthogonal incliné composé d'hexaèdres (voir Fig. A.7).

Stretched. Un maillage non orthogonal incliné composé d'hexaèdres étirés (voir Fig.

A.8).

Les noms en gras sont utilisés par la suite pour identifier sans ambiguïté les différents types de maillage. Pour les maillages non orthogonaux, on a gardé les noms utilisés dans les articles en anglais.

Notons qu'OpenFOAM est un outil pour faire des simulations 3-D, et une seule couche dans la troisième dimension, avec des conditions aux limites pertinentes, permet l'émulation de simulations 2-D. Alors, un maillage avec des cellules carrées correspond à un maillage hexaédrique et un maillage avec des cellules triangulaires correspond en fait à un maillage composé de prismes.

Pour résumer, les trois facteurs choisis dans cette méthodologie qui permettent de détecter la source de la dégradation de la convergence sont les suivantes :

— **Le solveur** qui résout les équations en utilisant plusieurs schémas numériques qui approchent les dérivées partielles et en utilisant plusieurs schémas d'interpolation.

— **Les conditions aux limites.** Trois types de conditions aux limites de base, Dirichlet, Neumann et cycliques sont utilisés.

— **Le maillage.** Trois critères de qualité seront vérifiés : la non-orthogonalité, l'asymétrie et la non-uniformité.

Implémentation des solutions manufacturées

Certaines solutions manufacturées et solutions analytiques ont été implémentées dans le processus de vérification. Un synopsis de ces solutions est donné ci-dessous.

Pour vérifier l'équation de Burgers bidimensionnelle, on a utilisé la solution manufacturée suivante (Knupp and Salari (2002)) :

$$\begin{aligned} u(x,y,t) &= u_0 \left[\sin(x^2+y^2+\omega t)+\epsilon\right] \\ v(x,y,t) &= v_0 \left[\cos(x^2+y^2+\omega t)+\epsilon\right] \end{aligned} \tag{3.1}$$

où ϵ et ω sont des constantes. ϵ est une "petite" constante utilisée pour éviter la symétrie relative à l'origine du système de coordonnées. On prend la valeur $\omega = 0$ pour obtenir une solution stationnaire. Un domaine non symétrique $[-0.4 : 0.4]$x$[-0.3 : 0.4]$ est recommandé pour commencer le processus de vérification (voir Knupp and Salari (2002)).

Pour la vérification de l'opérateur de Laplace, on a utilisé une solution manufacturée scalaire qui permet d'une façon simple la vérification des conditions aux limites de Dirichlet et cycliques :

$$\phi(x,y) = \phi_0 \left[\cos(\omega x)\sin(\omega y)\right] \tag{3.2}$$

où ω et ϕ_0 sont la fréquence et l'amplitude de la sinusoïde respectivement ; par simplicité on prend les valeurs $\omega = 1.0$ et $\phi_0 = 1.0$ dans les cas de conditions aux limites de type Dirichlet. Dans le cas de conditions cycliques (périodiques), on change la valeur de la fréquence $\omega = 2.0$ pour éviter une discontinuité.

Pour vérifier les conditions aux limites de type Neumann, on a utilisé une solution manufacturée pour simuler un flux entrant d'un côté et la même quantité de flux sortant du côté opposé dans des domaines rectangulaires. Pour les deux autres côtés, on a utilisé des conditions de Dirichlet :

$$\phi(x,y) = \left[\frac{(1-x^2)(\frac{y^3}{3} - \frac{\pi^2 y}{2})}{\pi^2}\right] \tag{3.3}$$

Une autre solution manufacturée est utilisée pour tester des conditions de Neumann homogènes (ou Neumann zéro gradient dans la nomenclature de OpenFOAM). Du point de vue technique, OpenFOAM utilise une autre classe pour ce type de condition de Neumann, c'est pourquoi on a besoin de tester cette classe également. Deux côtés opposés d'un domaine rectangulaire ont été définis de type Neumann homogènes et les deux autres de type Dirichlet :

$$\phi(x,y) = \left[\frac{(1-x^2)(2y^3 - 3\pi y^2 + \pi^3)}{\pi^3}\right] \tag{3.4}$$

Les solutions manufacturées Éqs. 3.2, 3.3 et 3.4 ont permis de détecter une dégradation de la convergence sur des maillages non orthogonaux pour les conditions de types Dirichlet et Neumann, ce qui était le sujet de ce premier thème. De plus, elles seront utilisées dans les procédures de vérification dans la deuxième partie de ce travail de recherche pour vérifier les corrections introduites pour traiter la non-orthogonalité.

3.1.2 Contribution

La contribution principale de cet article est d'avoir réussi à détecter la source de la dégradation de la convergence sur des maillages non orthogonaux. Un synopsis des principales étapes de cette procédure de vérification ainsi qu'un ensemble de résultats additionnels, autres que la contribution principale, sont donnés ici.

Synopsis du processus de vérification

Le tableau 3.1 présente un synopsis du processus de vérification (de haut en bas). *A priori*, une dégradation de la convergence lors des simulations numériques des équations de Navier-Stokes semble se produire sur des maillages qui ne sont pas orthogonaux. Les dernières étapes montrent que le problème est dû au traitement des conditions aux limites de types Dirichlet et Neumann. Le tableau montre de gauche à droite, d'abord l'équation ciblée qui doit être vérifiée (en parenthèses la solution manufacturée utilisée), ensuite on regarde quels types de conditions aux limites ont été utilisées, par la suite les types de maillages et dans les deux dernières colonnes on montre l'ordre de convergence asymptotique ainsi que les types d'erreurs utilisés pour calculer la convergence. Par exemple, l'étape 3 nous indique une vérification pour l'équation de Burger avec condition de Dirichlet et Neumann utilisant des maillages orthogonaux composés d'hexaèdres et d'hexaèdres allongés :

Conv. = convergence ; Err. = erreur ; Avg = erreur moyenne ; Max = erreur maximale.

(†), pour les maillages non-orthogonaux, le nom du maillage en anglais est indiqué. Le nom en anglais correspond au nom des maillages dans les graphes de convergence dans les articles.

(∗) - Détection de la dégradation de la convergence.

(∗∗) - on utilise le gradient analytique pour une reconstruction non-orthogonale à la frontière.

À noter :

- pour résoudre une équation, plusieurs schémas numériques (p.ex. plusieurs schémas du gradient) sont possibles, les résultats sont montrés pour les plus représentatifs.
- la valeur montrée pour la convergence est une valeur asymptotique.

Tableau 3.1 Synopsis du processus de vérification de l'article I.

	Équation ciblée	**Condition aux limites**	**Maillages(†)**	**Con.**	**Err.**
1	Éq. Navier-Stokes (∗)				
2	Éq. de Burgers stationnaire (Eq. 3.1)	Dirichlet	Orthogonaux : hexaédrique	2.0	Avg, Max
3	Éq. de Burgers stationnaire (Eq. 3.1)	Dirichlet et Neumann	Orthogonaux : hexaédrique	2.0	Avg, Max
4	Éq. de Burgers stationnaire (Eq. 3.1)	Dirichlet	Non orthogonaux : *Left.*	1.0	Avg, Max
5	Éq. de Burgers stationnaire (Eq. 3.1)	Dirichlet et Neumann	Non orthogonaux : *Left.*	1.0	Avg, Max
6	Éq. de Poisson (Eq. 3.2)	Dirichlet	Orthogonaux : hexaédrique, prismes équilatéraux, allongé.	2.0	Avg, Max
7	Éq. de Poisson (Eq. 3.2)	Dirichlet et Neumann	Orthogonaux : hexaédrique, prismes équilatéraux, allongé.	2.0	Avg, Max
8	Éq. de Poisson (Eq. 3.2)	Cyclique	Orthogonaux : hexaédrique.	2.0	Avg, Max
9	Éq. de Poisson (Eq. 3.2)	Cyclique	Non orthogonaux : *Left, Alternate.*	2.0	Avg, Max
10	Éq. de Poisson (Eq. 3.2)	Dirichlet	Non orthogonaux : *Left, Delaunay, Inclined, Stretched.*	<1.0	Avg, Max
11	Éq. de Poisson (Eq. 3.2)	Dirichlet	Non orthogonaux : *Alternate.*	2.0	Avg
12	Éq. de Poisson (Eq. 3.2)	Dirichlet	Non orthogonaux : *Alternate.*	1.0	Max
13	Éq. de Poisson (Eq. 3.2)	Dirichlet Corrigée(∗∗)	Non orthogonaux : *Left, Delaunay, Inclined, Stretched.*	2.0	Avg, Max
14	Éq. de Poisson (Eq. 3.3)	Dirichlet Corrigée (∗∗) et Neumann	Non orthogonaux : *Left, Stretched.*	<2.0	Avg, Max
15	Éq. de Poisson (Eq. 3.4)	Dirichlet Corrigée (∗∗) et Neumann homogène	Non orthogonaux : *Left, Stretched.*	<2.0	Avg, Max

— : < signifie que la valeur asymptotique de la convergence est plus petite que le chiffre à droite.

Dans ce tableau, il y a deux aspects qui conduisent à identifier la source du problème : d'abord, les lignes 8 et 9 présentent des vérifications utilisant des conditions aux limites de type cyclique. On peut voir que l'ordre de convergence est l'ordre théorique pour les deux types de maillage, orthogonal et non orthogonal. Ensuite, aux lignes 11 et 12 on voit que pour le maillage que nous appelons "Alternate" (voir Fig. A.5) la convergence est d'ordre deux pour l'erreur moyenne et d'ordre un pour l'erreur maximum (qui se situe à la frontière). Ce maillage est orthogonal à l'intérieur du domaine, c'est pourquoi l'ordre de convergence

moyen est deux. Cependant, les faces qui sont à la frontière ne sont pas orthogonales, ce qui explique la convergence de l'erreur maximale d'ordre un.

3.1.3 Résultats principaux.

Le principal résultat de la vérification de cette première partie est d'avoir réussi à identifier la source de la dégradation de la convergence sur des maillages non orthogonaux. La stratégie implémentée pour vérifier les opérateurs de convection-diffusion nous a permis de vérifier une vaste quantité de schémas implémentés qui incluent les schémas du solveur, les schémas de trois types de conditions aux limites, ainsi que trois critères de qualité des maillages. Ces résultats et d'autres, également liés au processus de vérification, sont discutés au chapitre 4.

3.2 Thème 2 : Résultats de l'implémentation des frontières de deuxième ordre.

Deuxième objectif : Corriger le problème de la dégradation de la convergence sur des maillages non orthogonaux et effectuer un processus de vérification des schémas implémentés.

L'article suivant répond à ce deuxième objectif :

Noriega, Guibault, Reggio, and Magnan ("forthcoming"b)."A Case Study in Open-Source CFD Code Verification.
Part II : Boundary Condition Non-Orthogonal Correction". (Voir annexe C)

3.2.1 Méthodologie

Cette deuxième partie de la thèse est la continuation d'une première partie pour laquelle on a identifié que le traitement de la non-orthogonalité pour des conditions aux limites de type Dirichlet ou de type Neumann était la source d'une dégradation de la convergence de l'outil OpenFOAM, qui est censée être d'ordre deux. Dans ce deuxième article, le point de départ est la démonstration mathématique de la nécessité d'introduire une correction aux faces à la frontière. L'analyse de l'erreur de troncature pour les conditions aux limites de types Dirichlet et Neumann a montré que le traitement non orthogonal de l'outil OpenFOAM conduit à une convergence d'ordre un pour les deux types de conditions aux limites. Par la suite, différentes stratégies pour obtenir des conditions à la frontière d'ordre deux ont été implémentées et des comparaisons de nouveaux résultats avec les résultats originaux sur différents types de maillages ont été faites.

Pour cette deuxième partie, à nouveau, les paramètres de la méthodologie sont le solveur, les conditions aux limites ainsi que les différents types de maillages. Le processus de vérification a été fait dans l'ordre inverse en ce qui concerne le choix du solveur, d'abord un solveur pour l'équation de Poisson, puis un solveur pour les équations de Navier-Stokes. Pour les deux types d'équations, nous avons vérifié l'ordre de convergence en utilisant les nouveaux types de conditions aux limites de Dirichlet, de Neumann et de Neumann homogène (nouveaux dans le sens qu'on a introduit une correction de la non-orthogonalité). La vérification des opérateurs de convection-diffusion se fait en utilisant simpleFOAM, un solveur pour les équations de Navier-Stokes stationnaires. Cependant, nous avons accompli le processus de vérification seulement pour les schémas les plus représentatifs. Pour l'opérateur de Laplace,

le schéma ciblé comme représentatif est le schéma “Gauss linear corrected” qui donne les meilleurs résultats en termes de convergence avant les corrections. Pour l'operator de divergence, le schéma choisi est le schéma “Gauss linear” ; d'autres schémas d'ordre un (ou d'ordre deux) tels que le schéma amont (upwind), TVD et autres n'ont pas été testés. Pour l'operator de gradient, nous avons vérifié les schémas “Gauss linear” et “moindres carrés”.

La vérification des nouvelles conditions aux limites a été faite avec des maillages non orthogonaux de type prisme présentés aux Figs. A.4, A.5 et avec des maillages non orthogonaux de type hexaédriques présentés aux Figs. A.7 et A.8. Les tests de vérification et le calcul de l'erreur numérique ont été faits en utilisant une série de solutions manufacturées et de solutions analytiques pour les deux solveurs utilisés.

Implémentation des solutions manufacturées

Pour vérifier les nouvelles conditions aux limites, en plus des solutions manufacturées Eqs. 3.2, 3.3 et 3.4, on utilise une simulation réelle, celle d'un écoulement visqueux entre deux plaques (dont l'une peut être en mouvement par rapport à l'autre). On suppose un écoulement stationnaire incompressible gouverné par les équations de Navier-Stokes avec un profil de vitesse qui dépend uniquement de x :

$$\vec{v}(x,y) = \{U_x(y), 0\}$$

L'intégration de l'équation du mouvement des équations de Navier-Stokes dans un domaine rectangulaire en utilisant en haut et en bas des conditions aux limites de Dirichlet pour le champ de vitesse, et de Neumann pour le champ de pression et le contraire sur les frontières à droite et à gauche, nous amène à la solution analytique du champ de vitesse et du champ de pression :

$$\begin{aligned} \vec{v}(x,y) &= \left\{ \frac{1}{2\mu} \left[\underbrace{\left(\frac{\partial p}{\partial x}\right)(y^2 - by)}_{\text{écoulement de Poiseuille}} + \underbrace{U_0 + (U_b - U_0)\frac{y}{b}}_{\text{écoulement Couette}} \right], 0 \right\} \\ p(x,y) &= \left[\frac{\partial p}{\partial x} x + p_0 \right] \end{aligned} \tag{3.5}$$

où U_b et U_0 sont la vitesse des plaques supérieure et inférieure respectivement, et μ le coeffi-

cient de viscosité dynamique. La vitesse sous la forme $\vec{v} = \{U_x(y), 0\}$ satisfait à l'équation de continuité automatiquement. Les termes du champ de vitesse qui sont associés au gradient de pression sont appelés l'écoulement de Poiseuille. Les termes associés au mouvement des plaques sont appelés les termes de l'écoulement de Couette.

3.2.2 Contributions

Deux contributions principales sont présentées dans cet article, une première est l'identification théorique du fait que les conditions aux frontières actuelles de type Dirichlet et Neumann dans OpenFOAM ont besoin d'une correction non-orthogonale. Une deuxième contribution principale est d'avoir réussi à implémenter des corrections sur le traitement non orthogonal pour les conditions de Dirichlet et Neumann qui garantissent une convergence de deuxième ordre. On donne ici un synopsis des étapes principales de la procédure de vérification pour cette deuxième partie de la thèse ainsi que des contributions additionnelles aux contributions principales.

Synopsis des résultats

Le tableau 3.2 présente un synopsis des résultats de vérification qui, cette fois, se fait dans l'ordre inverse. La vérification des nouvelles conditions aux limites de Dirichlet et Neumann se fait d'abord pour l'équation de Poisson, et par la suite pour des solutions analytiques des équations de Navier-Stokes. Le tableau montre de gauche à droite, d'abord l'équation ciblée qui doit être vérifiée (entre parenthèses la solution manufacturée utilisée), ensuite le type de conditions aux limites utilisé, par la suite le type de maillage et, dans les deux dernières colonnes, on montre l'ordre de convergence asymptotique ainsi que les types d'erreurs utilisés pour calculer la convergence. Les simulations numériques ont été conduites en utilisant une reconstruction locale du gradient de type Gauss qui est implémenté comme une condition aux limites. Tel que montré dans le deuxième article, il n'existe pas de différences significatives en termes de précision et convergence si on utilise une reconstruction locale du gradient de type moindres carrés ou si on utilise les libraires OpenFOAM de reconstruction du gradient.

Tableau 3.2 Synopsis du processus de vérification de l'article II.

	Équation ciblée	**Condition aux limites**	**Maillages**	**Conv.**	**Err.**
1	Éq. de Poisson (Eq. 3.2)	Dirichlet corrigé	Non orthogonaux : *Left, Alternate, Inclined, Stretched.*	2.0	Avg, Max
2	Éq. de Poisson (Eq. 3.2)	Dirichlet corrigé	Non orthogonaux : *Delaunay*(†)	2.0(†)	Avg, Max
3	Éq. de Poisson (Eq. 3.3)	Dirichlet corrigé et Neumann	Non orthogonaux : *Left, Stretched.*	<2.0	Avg, Max
4	Éq. de Poisson (Eq. 3.4)	Dirichlet corrigé et Neumann homogène	Non orthogonaux : *Left, Stretched.*	<2.0	Avg, Max
5	Éq. de Poisson (Eq. 3.3)	Dirichlet corrigé et Neumann corrigé	Non orthogonaux : *Left, Stretched.*	2.0	Avg, Max
6	Éq. de Poisson (Eq. 3.4)	Dirichlet corrigé et Neumann homogène corrigé	Non orthogonaux : *Left, Stretched.*	2.0	Avg, Max
7	Éqs. de Navier-Stokes de type sinusoïdal (Eq. 3.2)	Dirichlet corrigé	Non orthogonaux : *Left, Alternate, Inclined, Stretched.*	2.0	Avg, Max
8	Éqs. de Navier-Stokes de type sinusoïdal (Eq. 3.2)	Dirichlet corrigé	Non orthogonaux : *Delaunay*(†)	2.0(†)	Avg, Max
9	Éqs. de Navier-Stokes (Eq. 3.5)	Dirichlet corrigé et Neumann	Non orthogonaux : *Left*	<2.0	Avg, Max
10	Éqs. de Navier-Stokes (Eq. 3.5)	Dirichlet corrigé et Neumann homogène	Non orthogonaux : *Left*	<2.0	Avg, Max
11	Éqs. de Navier-Stokes (Eq. 3.5)	Dirichlet corrigé et Neumann corrigé	Non orthogonaux. *Left*	2.0	Avg, Max
12	Éqs. de Navier-Stokes (Eq. 3.5)	Dirichlet corrigé et Neumann homogène corrigé	Non orthogonaux. *Left*	2.0	Avg, Max

Conv. = convergence ; Err. = erreur ; Avg = erreur moyenne ; Max = erreur maximale.

(†) - Pour ce type de maillage la convergence est d'ordre deux, sauf pour la dernière étape de raffinement ; il ne s'agit pas d'un raffinement, mais de la génération d'autre maillage plus dense.

3.2.3 Résultats principaux.

Les deux contributions principales de la vérification de cette deuxième partie du travail sont d'avoir montré la nécessité d'ajouter une correction non orthogonale, en analysant l'erreur de troncature, pour les conditions de Dirichlet et Neumann, et d'avoir réussi à implémenter ces

corrections pour le traitement non orthogonal qui garantissent une convergence de deuxième ordre.

D'autres résultats produits par cette étude incluent des approches alternatives de reconstruction du gradient au barycentre des cellules voisines de la frontière, une correction à la frontière pour la classe Least-squares de reconstruction du gradient, et la performance des nouvelles conditions, entre autres. Ces résultats sont décrits avec plus de détails au chapitre 4.

CHAPITRE 4 DISCUSSION GÉNÉRALE

Ce chapitre présente, plus en détail, l'ensemble des résultats de cette étude de vérification. Ces résultats montrent la nécessité d'un processus de vérification pour un logiciel complexe comme OpenFOAM. L'ensemble des résultats est divisé en deux parties pour chacun des articles inclus dans cette thèse.

Dans la première partie, un processus de vérification qui se concentre sur l'opérateur de Laplace a été mis en oeuvre. Bien qu'appliquée uniquement à l'opérateur de diffusion, cette tâche s'est avérée complexe. La vérification des algorithmes tient compte de trois types de conditions aux limites, ainsi que de trois critères de qualité des maillages.

Le principal résultat de la vérification de cette première partie est d'avoir réussi à identifier la source de la dégradation de la convergence sur des maillages non orthogonaux. Les autres résultats produits par cette recherche sont listés ici.

— sur des maillages orthogonaux, on a vérifié que l'ordre de convergence pour l'opérateur de diffusion avec des conditions aux limites de Dirichlet est la valeur théorique attendue.

— mêmes résultats pour des conditions aux limites mixtes (Dirichlet et Neumann).

— on a vérifié que l'ordre de convergence pour l'opérateur de diffusion avec des frontières périodiques (cycliques) est la valeur attendue, que ce soit sur des maillages orthogonaux ou sur des maillages non orthogonaux.

— on a détecté une dégradation de la convergence de l'opérateur de diffusion sur des maillages non orthogonaux avec des conditions de Dirichlet. On a identifié que cette dégradation est liée au traitement non orthogonal à la frontière avec des conditions de Dirichlet.

— on a détecté une dégradation de la convergence de l'opérateur de diffusion sur des maillages non orthogonaux avec des conditions de Neumann. On a identifié que cette dégradation est liée au traitement non orthogonal à la frontière avec des conditions de

Neumann.

— on a vérifié que le traitement non orthogonal semble bien fonctionner pour les faces à l'intérieur du domaine. À l'intérieur du domaine, la convergence donne les résultats théoriques attendus.

— le schéma de discrétisation de l'opérateur de diffusion a un paramètre ψ. Pour ce paramètre, quatre valeurs sont définies : $\psi = \{0.0, 0.333, 0.5, 1.0\}$. Pour $\psi = 0.0$, la valeur λ sur la correction non-orthogonale (voir Éq. 2.22) est égale à zéro, et il n'y a pas de traitement non orthogonal. Pour $\psi = 1.0$ la valeur λ est égale à un. Pour la valeur $\psi = 0.5$ le module de la partie non-orthogonale est limité par le module de la partie orthogonale. Enfin, pour la valeur $\psi = 0.333$ le module de la partie non-orthogonale est limité à une demie du module de la partie orthogonale. On a déterminé que pour les solutions manufacturées utilisées dans ce travail, la valeur $\psi = 1.0$ donne la meilleure convergence.

— on a vérifié que le traitement algorithmique de l'asymétrie donne les résultats théoriques attendus. Pour les maillages avec non-uniformité, les poids semblent être bien calculés.

Un résultat implicite, important, de la vérification effectuée dans cette étude, lequel justifie ce travail, a été de montrer la nécessité d'un processus de vérification pour un logiciel complexe comme OpenFOAM. Le processus de vérification appliqué à une équation simple, comme l'équation de Poisson, s'est révélé être une tâche complexe. Trois types de conditions aux limites, trois types de critères de qualité du maillage, différents schémas pour la reconstruction du gradient de la partie non orthogonal, plusieurs schémas d'interpolation, ainsi que plusieurs options qui limitent le terme non orthogonal forment un ensemble combinatoire de possibilités qui fait de la vérification de l'opérateur de diffusion une tâche difficile. Même après avoir conclu que la perte de convergence provient des conditions aux limites, le labeur de trouver quelles sont les classes impliquées parmi l'ensemble de classes des librairies OpenFOAM est une tâche délicate.

Dans la deuxième partie, les deux contributions principales sont d'avoir montré la nécessité d'ajouter une correction non orthogonale, en analysant l'erreur de troncature, pour les conditions de Dirichlet et Neumann, et d'avoir réussi à implémenter ces corrections pour le traitement non orthogonal qui garantissent une convergence de deuxième ordre (voir le cha-

pitre 3). D'autres résultats produits par cette étude sont listés ici.

— dans le cas de conditions de Dirichlet, trois méthodes de reconstruction du gradient au barycentre des cellules voisines de la frontière ont été développées. Elles assurent que la projection du gradient numérique à la frontière est d'ordre 1 (en effet, une convergence d'ordre au moins un du gradient est nécessaire pour obtenir une convergence d'ordre deux de la valeur au barycentre d'une cellule voisine de la frontière). La première méthode de reconstruction locale du gradient est de type Gauss, la seconde est de type moindres carrés, et la troisième utilise les librairies OpenFOAM de reconstruction globale du gradient. Pour cette troisième méthode, plusieurs schémas peuvent être utilisés. Une autre alternative (une quatrième) dans le cas de conditions de Dirichlet consiste à implémenter la correction non orthogonale via un terme source injecté dans le solveur.

— plusieurs comparaisons des quatre méthodes de reconstruction pour des conditions aux limites de Dirichlet ont été présentées. Elles permettent de conclure que les quatre méthodes donnent des résultats similaires en termes de précision.

— l'implémentation de la correction non orthogonale à la frontière, telle qu'elle a été mise en œuvre dans ce travail, n'interfère pas avec la correction non orthogonale de OpenFOAM à l'intérieur du domaine. Cette dernière a besoin d'une reconstruction du gradient sur les faces internes du maillage, laquelle peut être implémentée en utilisant les schémas p.ex. de Gauss ou des moindres carrés. Si on utilise une reconstruction locale du gradient (à la frontière) de type Gauss pour implémenter la correction non orthogonale et à l'intérieur du domaine une reconstruction du gradient de type Gauss, on obtient l'ordre de convergence attendu. Cependant, une légère dégradation de la convergence a été détectée dans le cas d'une reconstruction locale du gradient à la frontière de type Gauss, lorsqu'une reconstruction du gradient de type moindres carrés est utilisée pour l'intérieur du domaine. Pour surmonter ce problème un traitement de la non-orthogonalité de la librairie de reconstruction des vecteurs des moindres carrés est présenté comme solution alternative.

— dans le cas de conditions de Neumann les mêmes trois méthodes de reconstruction du gradient au barycentre des cellules voisines à la frontière ont été appliquées pour obtenir un ordre de convergence de deux.

— plusieurs cas de vérification utilisant les solutions manufacturées pour l'équation de Poisson, en utilisant des conditions aux limites de type Dirichlet et Neumann modifiées permettent de vérifier les nouveaux résultats.

— dans le cas des équations de Navier-Stokes, le traitement non orthogonal dans les conditions aux limites effectué dans ce travail est affecté par le solveur. Par exemple, en utilisant les nouvelles conditions aux limites avec un nombre de Reynolds $Re = 30$, la convergence tombe en dessous de deux. La version originale du code simpleFOAM calcule le flux convectif quelques pas après le calcul du champ de vitesse, après avoir modifié le champ de vitesse, ce qui contribue à accélérer la convergence, mais influe sur le traitement de la non-orthogonalité des conditions aux limites présentée ici. Si le calcul du flux convectif est fait immédiatement après avoir calculé le champ de vitesse, la convergence atteint le second ordre :

Code original du solveur :

```
#include UEqn.H
p.boundaryField().updateCoeffs();
volScalarField AU = UEqn().A();
U = UEqn().H()/AU;
UEqn.clear();
phi = fvc::interpolate(U) & mesh.Sf(); // On calcule le flux
adjustPhi(phi, U, p);
```

Nouvelle variante du code :

```
#include UEqn.H
phi = fvc::interpolate(U) & mesh.Sf(); // On calcule le flux
p.boundaryField().updateCoeffs();
volScalarField AU = UEqn().A();
U = UEqn().H()/AU;
UEqn.clear();
adjustPhi(phi, U, p);
```

— un autre problème se pose pour les résidus matriciels. Pour une simulation d'un écoulement de Poiseuille sur maillages orthogonal, même si l'erreur de la composante nulle du champ de vitesse U_y est égale à zéro, le résidu matriciel de cette composante ne converge pas vers zéro. Ce problème se produit même si on utilise différents solveurs pour la vitesse ou la pression.

CHAPITRE 5 RÉSULTATS COMPLÉMENTAIRES, MISE EN OEUVRE

Dans ce chapitre, nous allons examiner certains détails techniques de la mise en œuvre de la correction non orthogonale à la frontière. Ce qui justifie cette explication est le manque de documentation (au moment d'écrire cette thèse) des librairies d'OpenFOAM, ce qui rend la programmation difficile. Aussi, nous montrerons certains résultats sur la performance des nouvelles conditions aux limites.

5.1 Traitement non orthogonale à la frontière dans OpenFOAM

Le schéma de discrétisation centré sur les éléments appliqué aux faces d'un élément (comme celui implémenté dans OpenFOAM) est un schéma d'ordre deux qui approxime les intégrales de surfaces produites par l'application du théorème de Gauss sur les opérateurs de convection-diffusion. Dans le deuxième article de cette thèse, il a été montré que les valeurs numériques de la variable ϕ et de son gradient $(\nabla\phi)$, utilisées pour calculer les expressions produites par la discrétisation de ces intégrales, doivent satisfaire un critère de précision. Pour les conditions aux limites de type Dirichlet, la valeur numérique du gradient $(\nabla\phi)_{C_f}$ doit être calculée avec une précision d'ordre un. Pour les conditions aux limites de type Neumann, la valeur numérique de la variable ϕ_{C_f} doit être calculée avec une précision d'ordre deux. On a montré dans ce deuxième article qu'une correction est nécessaire dans le cas de maillages non orthogonaux pour les conditions aux limites de type Dirichlet ou Neumann implémentées dans OpenFOAM. La correction non orthogonale à la frontière, de la façon dont elle a été implémentée dans cette thèse, passe par la reconstruction du gradient $(\nabla\phi)_P$ aux barycentres des éléments qui ont une face frontière (voir la Fig. 5.1).

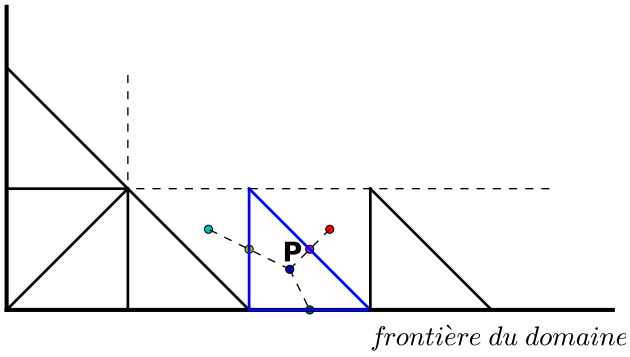


Figure 5.1 Éléments voisins à la frontière

Dans le ca d'une condition de type Dirichlet, nous avons ajouté une correction $\vec{k} \cdot (\nabla\phi)_P$ au traitement non orthogonal du terme de diffusion qui permet de corriger la projection sur la normale du gradient numérique évalué au barycentre d'une face f (voir Fig. 5.2) :

$$\begin{aligned} \left(\vec{S} \cdot (\nabla\phi)_{C_f}\right)_h &= \frac{|S|}{|d_n|}\left(\phi_{C_f} - \phi_P - \vec{k} \cdot (\nabla\phi)_P\right) \\ &= \frac{|S|}{|d_n|}\left(\phi_{C_f} - \phi_D\right) \end{aligned} \tag{5.1}$$

où ϕ est une propriété physique, $\vec{S}$ le vecteur aire de la face f, $\vec{k}$ le vecteur de correction non-orthogonale, $|\vec{d_n}|$ est la distance de D (projeté orthogonal de P sur la normale à la face) au barycentre C_f de la face frontière et ϕ_D est la valeur de ϕ au point D.

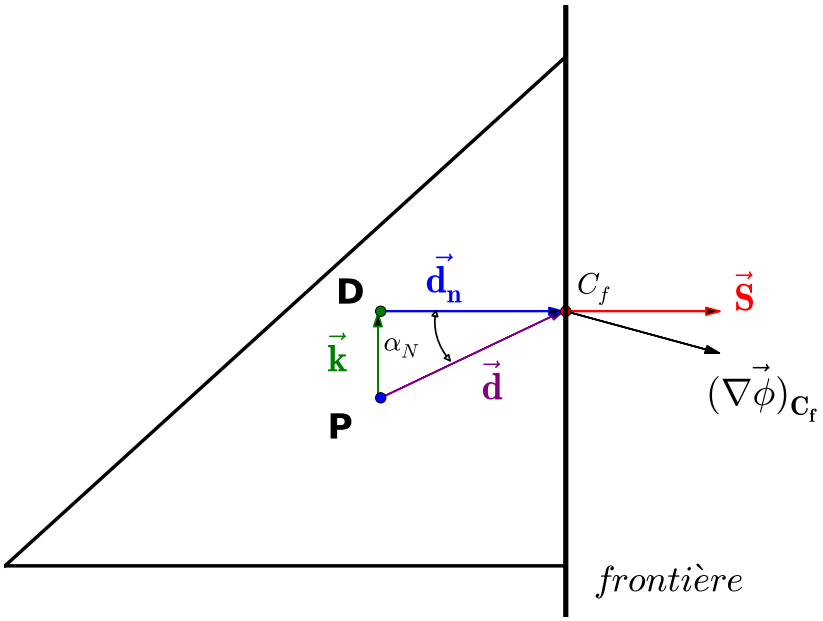


Figure 5.2 Non orthogonalité aux faces frontière

$$|\vec{d_n}| = |\vec{d}|cos(\alpha_N) \tag{5.2}$$

où $|\vec{d}|$ est la distance du barycentre P d'un élément au barycentre C_f de la face à la frontière.

Le second membre de l'Éq. 5.1 peut être réécrit de la façon suivante :

$$\frac{|S|}{|d_n|}\left(\phi_{C_f} - \phi_P - \vec{k}\cdot(\nabla\phi)_P\right) = -\frac{|S|}{|d_n|}(\phi_P) + \frac{|S|}{|d_n|}\left(\phi_{C_f} - \vec{k}\cdot(\nabla\phi)_P\right) \tag{5.3}$$

Le membre de droite de l'Éq. 5.3 est composé de deux termes. Le premier terme, ϕ_P est une valeur inconnue et le coefficient $-\frac{|S|}{|d_n|}$ ajoute une contribution à la matrice du système d'équations (une contribution à la partie implicite). Le deuxième terme est composé, d'abord de la contribution de la valeur $\left(\phi_{C_f}\right)$ à la frontière : $\frac{|S|}{|d_n|}\left(\phi_{C_f}\right)$, puis de la correction non orthogonale $\frac{|S|}{|d_n|}\left(-\vec{k}\cdot(\nabla\phi)_P\right)$. Ils forment le terme source, c.à-d., le second membre du système matriciel. De même, dans le cas des conditions de Neumann, on peut montrer que la contribution non orthogonale est incluse dans le deuxième terme de l'équation matricielle (cette procédure n'est pas présentée ici).

Une correction non orthogonale pour l'opérateur de diffusion avec des conditions de Dirichlet, telle que montrée à l'équation 5.1, a été présentée par Juretic (2004). Dans le deuxième article de ce travail, on a inclus dans la démonstration du schéma Éq. 5.1 les erreurs numériques de convergence aux barycentres des éléments avec une face frontière. Ces erreurs n'étaient pas incluses par Juretic (2004) dans sa démonstration, et nous avons mis en évidence leur importance.

5.1.1 Types de reconstruction du gradient pour le traitement non orthogonal

Dans la deuxième partie de ce travail, nous avons réalisé deux types de reconstruction du gradient $(\nabla\phi)_P$, une première que nous appelons reconstruction locale et une deuxième que nous appelons reconstruction globale. Pour la première approche, la reconstruction du gradient est faite seulement au voisinage de la frontière. Une structure de donnée est alors nécessaire pour garder les valeurs du gradient pour chaque section de frontière où une condition limite est définie. Le gradient peut être reconstruit localement en utilisant le schéma de Gauss ou le schéma des moindres carrés. Pour la reconstruction globale, on utilise les classes OpenFOAM qui reconstruisent le gradient de façon explicite sur tout le domaine (p.ex. on peut utiliser les classes soit du schéma de Gauss, ou bien du schéma des moindres carrés). Une façon pratique d'implémenter ces deux approches est d'introduire une nouvelle condition aux limites. Les schémas d'OpenFOAM pour les conditions aux limites fournissent tous les outils pour manipuler les termes internes, c.-à-d., ceux qui concernent les coefficients de la matrice du système d'équations linéaires, et pour manipuler les termes du second membre du système matriciel.

5.2 Les classes OpenFOAM pour la frontière

Dans cette section, on fournit une brève description de la hiérarchie des classes qui permettent de mettre en œuvre une nouvelle condition aux limites. Il convient de noter les avantages du langage de programmation C++ orienté objet (POO) à la base du code OpenFOAM. D'un côté, on a la complexité des expressions et des structures de données de OpenFOAM. Et, d'autre part, la POO permet une meilleure modularité dans la conception logicielle. La POO présente d'énormes avantages : la facilité d'organisation des classes, la réutilisation du code, la création des méthodes plus intuitive, la possibilité d'héritage, une grande facilité de correction des erreurs, les projets sont plus faciles à gérer, etc.

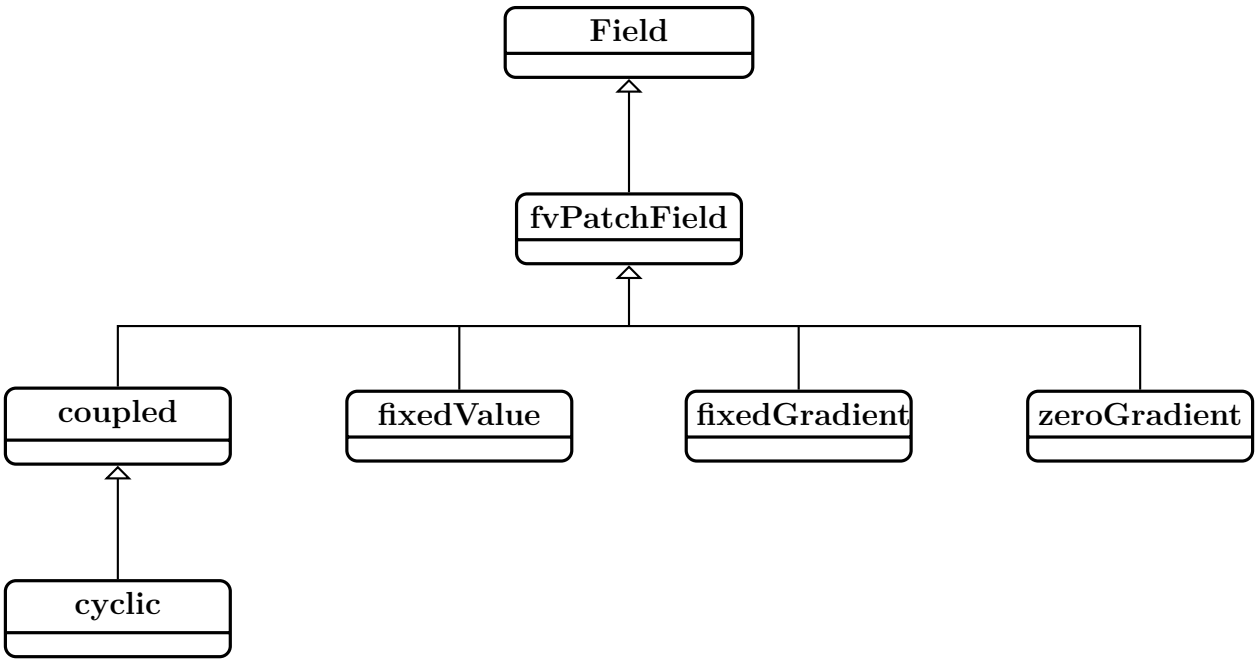
Dans OpenFOAM, une façon d'implémenter une correction non-orthogonale d'une condition aux limites à la frontière, sans modifier le code source, est d'implémenter une nouvelle condition aux limites avec correction (par exemple, elles peuvent être créées en tant que classes dérivées des conditions aux limites existantes). Dû à la grande quantité de conditions aux limites de la version actuelle d'OpenFOAM-3.x (plus d'une cinquantaine de types, la plupart dérivés des types de base), cette description est limitée aux types de base.

La classe de base de toute condition aux limites est la classe **fvPatchField** qui est une classe dérivée de la classe **Field** (champ en français) qui permet de définir un champ tensoriel. La classe **fvPatchField** est une structure de donnée qui permet de définir une section ou un morceau de frontière (appelée "Patch" dans OpenFOAM), à partir de laquelle certains types de base de conditions aux frontières sont obtenus par spécialisation :

- **fixedValueFvPatchField** pour les conditions aux limites de type Dirichlet
- **fixedGradientFvPatchField** pour les conditions de type Neumann
- **zeroGradientFvPatchField** pour les conditions de type Neumann homogène
- **cyclicFvPatchField** pour des conditions aux limites de type périodique

et plusieurs autres qui ne sont pas présentés dans ce travail.

Le diagramme UML suivant (où nous avons omis le suffixe du nom des classes dérivées) permet de visualiser la hiérarchie de classes :



5.2.1 Les méthodes des classes dérivées de fvPatchField

La classe **fvPatchField** et ses classes dérivées possèdent des méthodes qui permettent de modifier les valeurs de la matrice du système d'équations linéaires, qui concernent les schémas définis à la frontière. On peut modifier le coefficient de la valeur inconnue ϕ_P, c.-à-d., le coefficient ("interne") de la variable définie au barycentre d'un élément voisin de la frontière, ou on peut ajouter des modifications au second terme de la matrice (pour ajouter par exemple un terme de correction non-orthogonale). Dans le cas d'une frontière de type Dirichlet, pour ajouter une correction non orthogonale dans l'opérateur de diffusion, les deux méthodes suivantes permettent de modifier le coefficient de ϕ_P et le second terme, respectivement :

```
template<class Type>
tmp<Field<Type>>
    fixedValueFvPatchField<Type>::gradientInternalCoeffs() const

template<class Type>
tmp<Field<Type>>
    fixedValueFvPatchField<Type>::gradientBoundaryCoeffs() const
```

Dans le cas d'une frontière de type Dirichlet, on ne modifie pas le code dans la méthode "gradientInternalCoeffs()". On ajoute la correction non orthogonale à l'intérieur de la méthode "gradientBoundaryCoeffs()" :

```
fixedValueFvPatchField<Type>::gradientBoundaryCoeffs() const
{
    return ((*this) - skGrad())*this->patch().deltaCoeffs()
}
```

où :

- `*this` est le pointeur vers la condition aux limites de type Dirichlet,
- `skGrad()` est la correction non-orthogonale, c.-à-d., $(\vec{k} \cdot \nabla\phi)$, et
- `this->patch().deltaCoeffs()` sont des coefficients produits par la discrétisation volumes finis.

Dans le cas d'une frontière de type Neumann, les deux méthodes suivantes permettent de modifier le coefficient "interne" et le second terme de la matrice respectivement :

```
template<class Type>
tmp<Field<Type>>
    fixedGradientFvPatchField<Type>::
     valueInternalCoeffs(const tmp<scalarField>&) const

template<class Type>
tmp<Field<Type>>
    fixedGradientFvPatchField<Type>::
     valueBoundaryCoeffs(const tmp<scalarField>&) const
```

Dans les librairies originales, pour les frontières de type Neumann homogènes, la valeur d'une variable physique au barycentre ϕ_{C_f} de la face frontière est extrapolée à partir de la valeur au barycentre ϕ_P de l'élément. Cette approche demande une correction à la frontière dans le cas de maillages non orthogonaux, qui est ajoutée à l'intérieur de cette dernière méthode, et qui nous permet d'extrapoler la valeur au barycentre ϕ_{C_f} à partir de la valeur ϕ_D (voir la figure 5.2). Cette correction doit aussi être ajoutée pour les conditions de type Neumann non-homogènes. Pour les frontières de type Neumann, on a :

```
fixedGradientFvPatchField<Type>::valueBoundaryCoeffs(const tmp<scalarField>&)
{
    gradient_/this->patch().deltaCoeffs() + skGrad()
}
```

où :

- `gradient_` est la condition aux limites de type Neumann à la frontière,
- `skGrad()` est la correction non-orthogonale, et
- `this->patch().deltaCoeffs()` sont des coefficients produits par la discrétisation volumes finis.

La correction non orthogonale pour les frontières de type Neumann doit aussi être ajoutée dans la méthode :

```
template<class Type>
void fixedGradientFvPatchField<Type>::evaluate(const Pstream::commsTypes)
{
    Field<Type>::operator=
    (
    this->patchInternalField() + gradient_/this->patch().deltaCoeffs()
        + skGrad()
    );
}
```

qui permet de gérer l'actualisation de la valeur de ϕ_{C_f}. `this->patchInternalField()` est la valeur de ϕ_P au barycentre de l'élément voisin de la frontière.

5.3 Les classes OpenFOAM pour la reconstruction du gradient

Pour la reconstruction globale, l'approche la plus simple à mettre en œuvre pour reconstruire le gradient passe par l'utilisation des modules OpenFOAM d'interpolation et de reconstruction du gradient :

`fvc::interpolate(fvc::grad(`ϕ`))`
où les méthodes `fvc::grad(`ϕ`)` reconstruisent le gradient de la variable ϕ aux barycentres des éléments sur tout le domaine. Les méthodes `fvc::interpolate(`$\nabla\phi$`)` interpolent les valeurs du gradient sur les faces d'un volume de contrôle. Le produit scalaire de ce dernier et du vecteur correction $\vec{k}$ au barycentre des faces génère la correction non orthogonale pour l'opérateur de diffusion :

$\vec{k}$ · `fvc::interpolate(fvc::grad(`ϕ`))`

Cette correction doit être ajoutée dans les méthodes :

`fixedValueFvPatchField<Type>::gradientBoundaryCoeffs()`

`fixedGradientFvPatchField<Type>::valueBoundaryCoeffs(const tmp<scalarField>&)`

Pour ne pas interférer avec la correction non orthogonale à l'intérieur du domaine, et de plus pour améliorer la performance, le champ vectoriel de surface $\vec{k}$ ("de surface" dans le sens que le champ est défini sur les faces) doit être défini nul sur les faces intérieures au domaine.

Pour la reconstruction locale, il faut créer une nouvelle méthode, p.ex. dans la classe `fixedValueFvPatchField`, et utiliser le schéma de "Gauss" ou "moindres carres" pour faire une reconstruction seulement dans les éléments voisins à la frontière. Pour optimiser le code, la structure de données qui garde la correction non orthogonale doit être initialisée dans le constructeur de même classe `fixedValueFvPatchField`.

5.4 Correction de la non orthogonalité par un terme source

Une mise en œuvre d'une correction non-orthogonale à travers un solveur est possible, mais, de façon similaire au cas des conditions aux limites, à l'heure actuelle, il y a plus de 80 solveurs standards (ceux qui sont distribués dans la version officielle et qui ne sont pas programmés par les usagers) décrits dans la documentation. La correction non orthogonale peut être implémentée via un terme source dans le solveur dans le cas de conditions aux limites de type Dirichlet :

`solve(A == q + fvc::div (`$\nu|S|\vec{k}$ `·` `fvc::interplate(fvc::grad(`ϕ`))))`
où A est la matrice de coefficients du système linéaire d'équations, et $\nu|S|$ est le produit du coefficient de viscosité et de l'aire de la face. Cette approche a été testée et comparée à la précédente. Les deux approches donnent des résultats valides.

5.5 Performance des nouvelles conditions aux limites

Dans cette section, nous montrons certains résultats de l'application des nouvelles conditions aux limites en termes de performance. Nous regardons les résultats de la simulation d'un écoulement de Poiseuille et nous mesurons la performance en termes de temps.

5.5.1 Simulation d'un écoulement de Poiseuille

En utilisant les nouvelles conditions aux limites pour un écoulement de Poiseuille entre deux plaques, une convergence plus lente est observée. Pour obtenir une précision de 10^{-07} en utilisant les nouvelles conditions aux limites avec des mailles non orthogonales de 18 000, 32 000 et 50 000 éléments, le temps de convergence est augmenté jusqu'à 20 pour cent. Si en plus, on modifie le solveur simpleFOAM pour calculer le flux convectif juste après le calcul du champ de vitesse, l'augmentation du temps de calcul peut être plus significatif. Dans le tableau 5.1, on regarde l'augmentation du temps (en secondes) par rapport au code original sans correction aux frontières.

Nombre de Reynold :	30
Schéma temporel :	stationnaire
Schéma du gradient :	"Gauss linear"
Schéma de la divergence :	"Gauss linear"
Schéma du laplacien :	"Gauss linear limited 1.0"
Solveur :	simpleFoam (sans turbulence)
Maillage :	maillage non orthogonal régulier prismatique de type Left Fig. A.4
Domaine :	rectangulaire de 3 x 0.2 mètres

Tableau 5.1 Performance des nouvelles conditions aux limites.

Nombre d'éléments :	**Schéma original**	**Schéma avec correction**	%	**Schéma avec correction (∗)**	%
18 000	196	220	13	240	23
32 000	617	710	15	755	22
50 000	1 526	1 827	20	1 969	29

où (%) est le pourcentage du temps qui augmente par rapport au schéma original.
(∗) le flux convectif est calculé juste après le calcul du champ de vitesse.

CHAPITRE 6 CONCLUSION ET RECOMMANDATIONS

Cette thèse est composée principalement de trois contributions qui ont permis de vérifier et d'améliorer la convergence des opérateurs de convection-diffusion implantés dans le logiciel OpenFOAM. La première contribution est la définition des paramètres et la mise en œuvre de la méthodologie qui a permis de détecter une perte de convergence ; la deuxième contribution est l'identification théorique de la source d'erreur utilisant l'erreur de troncature, et la troisième est le développement de différentes alternatives d'implémentation des conditions aux limites d'ordre deux, ainsi que la vérification de ces nouveaux types de conditions aux limites à l'aide de la méthodologie introduite précédemment.

Dans la première contribution, une méthode de vérification a été mise au point pour identifier qu'une perte de convergence provient du traitement non orthogonal. Cette méthode vérifie les schémas numériques pour les opérateurs de convection-diffusion avec des conditions de frontières de type Dirichlet, de type Neumann et de type périodique. La vérification est faite sur des maillages composés de prismes et d'hexaèdres qui ont différentes caractéristiques, comme la non-orthogonalité, l'asymétrie et la non uniformité des poids. Le processus de vérification a permis de conclure que la source de la perte de convergence est le traitement des conditions aux limites de types Dirichlet et Neumann. Sur des maillages orthogonaux, les schémas implémentés donnent les résultats théoriques attendus avec les conditions aux limites de types Dirichlet et Neumann, mêmes si les maillages sont non-uniformes et asymétriques. Les résultats théoriques attendus sont aussi observés pour des maillages non-orthogonaux à l'intérieur du domaine. Cependant l'ordre de convergence peut diminuer à l'intérieur du domaine si on utilise une contrainte sur le traitement non orthogonal (voir Éq. 2.22).

Dans une deuxième étape, pour identifier la source du problème, une analyse de l'erreur de troncature est mise au point. Des démonstrations mathématiques basées sur l'erreur de troncature montrent la nécessité d'ajouter des corrections aux frontières. Ces démonstrations vont servir de garant du bon fonctionnement de l'implémentation pratique des nouveaux schémas pour les conditions de frontières.

La troisième contribution de cette thèse est d'avoir réussi à implémenter des corrections pour le traitement non orthogonal pour les conditions aux limites de Dirichlet et Neumann qui garantissent une convergence de deuxième ordre. Plusieurs alternatives d'implémentation de

conditions aux limites de deuxième ordre ont été mises en œuvre pour les conditions aux limites de type Dirichlet et Neumann. Elles impliquent la reconstruction du gradient au barycentre des cellules voisines à la frontière. Elles présentent des résultats similaires en termes de précision et convergence. Des cas tests (utilisant les schémas corrigés) sur tous les maillages non orthogonaux utilisés dans la première partie de cette recherche, convergent selon l'ordre théorique pour l'équation de Poisson et pour les équations de Navier-Stokes.

Enfin, nous avons détecté une "petite" perte de convergence provenant de la librairie de reconstruction du gradient par la méthode des moindres carrés. Une correction non-orthogonale à l'intérieur de ces classes sur les vecteurs des moindres carrés règle le problème sur tous les maillages testés. D'autre part, dans les simulations avec les équations de Navier-Stokes, le traitement non orthogonal pour les conditions aux limites implémenté dans ce travail est affecté par le solveur. Si le calcul du flux convectif est fait immédiatement après avoir calculé le champ de vitesse, la convergence atteint le second ordre.

6.1 Limitations de la solution proposée

Une étude de vérification en profondeur des librairies OpenFOAM est une tâche complexe, même si cette vérification n'est faite que pour les libraires de discrétisation en volumes finis des opérateurs de convection-diffusion. Dans cette étude, la taille de l'espace combinatoire du produit cartésien des paramètres utilisés pour la vérification a été réduite. Ces paramètres (que nous avons décrits dans la méthodologie) sont :

- les schémas numériques
- les conditions aux limites (plus précisément, les schémas numériques implémentés à la frontière)
- les caractéristiques du maillage

Tel qu'indiqué au début de la thèse, seulement trois types de base de conditions aux limites ont été vérifiés, les types de Dirichlet, Neumann et périodique. Au moment d'écrire cette thèse, des dizaines de conditions de frontières font partie de librairies OpenFOAM, la plupart d'entre elles sont dérivées des types de base.

OpenFOAM est un outil de simulations numérique 3-D par défaut, et quoique les maillages qui sont présentés à l'annexe A sont 2-D, les simulations ont été réalisées sur des maillages 3-D avec une seule couche (et des conditions de frontière "vide" pour les frontières dans la

troisième dimension) pour simuler le cas 2-D. Pour les simulations numériques, on n'a utilisé que des cellules de maillages de forme hexaédrique ou prismatique (voir l'annexe A). Une extension de présents travaux à des cas tridimensionnels complets serait à envisager.

En ce qui concerne les schémas numériques utilisés, nous avons vérifié en profondeur tous les schémas possibles qui discrétisent le laplacien pour détecter la source du problème. Cependant, le processus inverse de vérification a été fait uniquement pour les schémas les plus représentatifs. Pour l'opérateur laplacien, le schéma représentatif choisi est le schéma "Gauss linear corrected" qui donne les meilleurs résultats pour ce qui est de la convergence dans le code original. Pour l'opérateur de divergence le schéma choisi est le schéma linéaire de Gauss (Gauss linear). Les autres schémas tels que les schémas amont (upwind), TVD et autres n'ont pas été testés. Pour l'operateur du gradient, nous avons vérifié le schéma linéaire de Gauss (Gauss linear) et la version corrigée du schéma des moindres carrés.

6.2 Améliorations futures

Les limites de l'étude décrites dans le paragraphe précédent 6.1 sont un point de départ pour proposer des améliorations futures. Peut-être le cas le plus simple pour étendre les résultats actuels serait d'utiliser des maillages autres que les maillages hexaédrique ou prismatique.

Concernant les conditions aux limites, des dizaines de conditions aux limites, la plupart dérivées des types de base, auraient besoin d'être vérifiées. Ce n'est pas une tâche facile, car chaque modèle de conditions aux limites requiert une solution analytique ou une solution manufacturée.

L'autre avenue de recherche serait d'augmenter la complexité du solveur. Le processus de vérification peut être étendu aux modèles de turbulence, aux modèles des écoulements compressibles, aux modèles multiphasiques, aux modèles thermiques, aux modèles électromagnétiques, etc.

RÉFÉRENCES

J. Abanto, D. Pelletier, A. Garon, J. Trepanier, et M. Reggio, "Verification of some commercial cfd codes on atypical cfd problems", *43rd AIAA Aerospace Sciences Meeting and Exhibit*, vol. 1, pp. 15 693–15 725, 2005.

T. Baxth, "Aspects of unstructured grids and finite volume solvers for the euler and navier-stokes equations." *AGARD R-787, Special Courseon Unstructured Grid Methods for Advection Dominated Flows, Brussels, Belgium, 18-22 May*, vol. 6, pp. 1–61, 1992.

B. Blais et F. Bertrand, "On the use of the method of manufactured solutions for the verification of cfd codes for the volume averaged navier–stokes equations", *Computers and Fluids Journal*, vol. 114, no. 121 129, 2015.

J. Blazek, *Computational Fluid Dynamics : Principles and Applications, 2nd Edition.* Elsevier Science, 2006.

T. Chung, *Computational Fluid Dynamics.* Cambridge University Press, Cambridge, 2010.

B. Diskin et J. Thomas, "Comparison of node-centered and cell-centered unstructured finite volume discretizations : Inviscid fluxes", *AIAA JOURNAL*, vol. 49, no. 4, 2011.

B. Diskin, J. Thomas, E. Nielsen, H. Nishikawa, et J. White, "Comparison of node-centered and cell-centered unstructured finite-volume discretizations : Viscous fluxes", *AIAA JOURNAL*, vol. 48, no. 7, 2010.

L. Eca et M. Hoekstra, "An introduction to cfd code verification including eddy-viscosity models", 2006.

——, "Code verification of unsteady flow solvers with the method of the manufactured solutions", *International offshore and polar engineering ; ISOPE 2007*, pp. 2012–2019, 2007.

L. Eca, M. Hoekstra, A. Hay, et D. Pelletier, "On the construction of manufactured solutions for one and two-equation eddy-viscosity models." *International Journal for Numerical Methods in Fluids.*, vol. 54(2), pp. 119–154, 2007.

——, "Verification of rans solvers with manufactured solutions"', *Engineering with Computers*, vol. 23, pp. 253–270, 2007.

J. Ferziger et M. Peric, *Computational Methods for Fluid Dynamics, 3nd Edition.* Springer, 2002.

W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations.* Springer, 1998.

M. Hall, "Cell-vertex multigrid scheme for solution of the euler equations." *Proc. Int. Conf. on Numerical Methods for Fluid Dynamics, Reading, UK, Springer Verlag*, 1985.

H. Jasak, "Error analysis and estimation for the finite volume method with applications to fluid flows", Thèse de doctorat, Imperial College, Juin 1996.

H. Jasak et A. Gosman, "Automatic resolution control for the finite-volume method, part 2 : a-posteriori error estimates", *Numerical Heat Transfer, Part B : Fundamentals*, vol. 38, no. 3, pp. 237–256, 2000.

F. Juretic, "Error analysis in finite volume cfd." Thèse de doctorat, Department of Mechanical Engineering, Imperial College., 2004.

P. Knupp et K. Salari, *Verification of Computer Codes in Computational Science and Engineering.* Chapman and Hall/CRC, 2002.

——, *Verification of Computer Codes in Computational Science and Enginerring*, K. H. Rosen, éd. Chapman & Hall/CRC, 2003.

S. Muzaferija, "Adaptive finite volume method for flow predictions using unstructured meshes and multigrid approach." Thèse de doctorat, University of London, 1994.

R. Ni, "Multiple grid scheme for solving the euler equations." *AIAA Paper*, vol. 81, p. 1025, 1981.

H. Noriega, F. Guibault, M. Reggio, et R. Magnan, "A case study in open-source cfd code verification. part ii : Boundary condition non-orthogonal correction." "forthcoming".

H. Noriega, F. Guibault, M. Reggio, et M. R., "A case-study in open-source cfd code verification. part i : Convergence rate loss diagnosis." "forthcoming".

W. L. Oberkampf, F. G. Blottner, et A. D. P., "Methodology for computational fluid dynamics code verification/validation." *AIAA Paper.*, pp. 95–2226, 1995.

W. Oberkampf et C. Roy, *Verification and Validation in Scientific Computing*, Cambridge, éd. Cambridge University Press, 2010.

OpenFOAM, *OpenFOAM. The Open Source CFD Toolbox. User Guide.*, 12 2011.

D. Pelletier, E. Turgeon, et D. Tremblay, "Verification and validation of impinging round jet simulations using an adaptive fem. 44, 737-763." *International Journal for Numerical Methods in Fluids.*, vol. 44, pp. 737–763, 2004.

P. Roache, "Verification of codes and calculations", *AIAA Journal*, vol. 36, 5, no. 5, pp. 696–702, 1998.

P. J. Roache et S. Steinberg, "Symbolic manipulation and computational fluid dynamics." *AIAA Journal.*, vol. 22(10), pp. 1390–1394, 1984.

P. J. Roache, P. M. Knupp, S. Steinberg, et R. L. Blaine, "Experience with benchmark test cases for groundwater flow." *American Society of Mechanical Engineers, Fluids Engineering Division.*, vol. Vol. 93, Book No. H00598., pp. 49–56, 1990.

C. Rossow, "Berechnung von stromungsfeldern durch losung der euler-gleichungen mit einer erweiterten finite-volumen diskretisierungsmethode (calculation of flow fields b y the solution of euler equations using an extended finite volume discretisation scheme)", *DLR Research Report*, vol. 89, p. 38, 1989.

——, "Flux balance splitting - a new approach for a cell-vertex upwind scheme", *Proc. 12th Int. Conf. on Numerical Methods in Fluid Dynamics, Oxford, UK, Springer Verlag*, 1990.

——, "Accurate solution of the 2d euler equations with an efficient cell-vertex upwind scheme." *AIAA Paper*, vol. 93, p. 0071, 1993.

C. Roy, C. Nelson, T. Smith, et O. C., "Verification of euler/navier-stokes codes using the method of manufactured solutions." *International Journal for Numerical Methods in Fluids.*, vol. 44(6), pp. 599–620, 2004.

C. J. Roy, C. C. Nelson, T. M. Smith, et C. C. Ober, "Verification of euler/navier–stokes codes using the method of manufactured solutions", *International Journal for Numerical Methods in Fluids*, vol. 44, no. 6, pp. 599–620, 2004. DOI : `10.1002/fld.660`. En ligne : `http://dx.doi.org/10.1002/fld.660`

M. Schafer, *Computational Engineering. Introduction to Numerical Methods.* Springer, 2006.

T. M. Shih, "Procedure to debug computer programs." *International Journal for Numerical Methods in Engineering.*, vol. 21(6), pp. 1027–1037, 1985.

S. Steinberg et P. J. Roache, “Symbolic manipulation and computational fluid dynamics.” *Journal of Computational Physics*, vol. 57(2), pp. 251–284, 1985.

F. Stern, R. Wilson, et J. Shao, “Quantitative v-v of cfd simulations and certification of cfd codes”, *International Journal for Numerical Methods in Fluids*, vol. 50, no. 11, pp. 1335–1355, 2006. DOI : 10.1002/fld.1090. En ligne : http://dx.doi.org/10.1002/fld.1090

H. J. Stetter, “The defect correction principle and discretization methods.” *Numerische Mathematik.*, vol. 29(4), pp. 425–443, 1978.

D. Tremblay, S. Etienne, et D. Pelletier, “Code verification and the method of manufactured solutions for fluid-structure interaction problems”, *36th AIAA Fluid Dynamics Conference and Exhibit*, vol. 2, pp. 882–892, 2006.

S. Veluri et C. Roy, “Comprehensive code verification for an unstructured finite volume cfd code”, *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition 4 - 7 January 2010*, 2010.

P. Zadunaisky, “On the estimation of errors propagated in the numerical integration of ordinary differential equations.” *Numerische Mathematik.*, vol. 27, no. 1, pp. 21–39, 1976.

ANNEXES

— **ANNEXE A**. Maillages pour l'étude de vérification.

— **ANNEXE B**. ARTICLE 1. A Case-Study in Open-Source CFD Code Verification. Part I : Convergence Rate Loss Diagnosis.

— **ANNEXE C**. ARTICLE 2. A Case-Study in Open-Source CFD Code Verification. Part II : Boundary Condition Non-Orthogonal Correction.

ANNEXE A Maillages pour l'étude de vérification

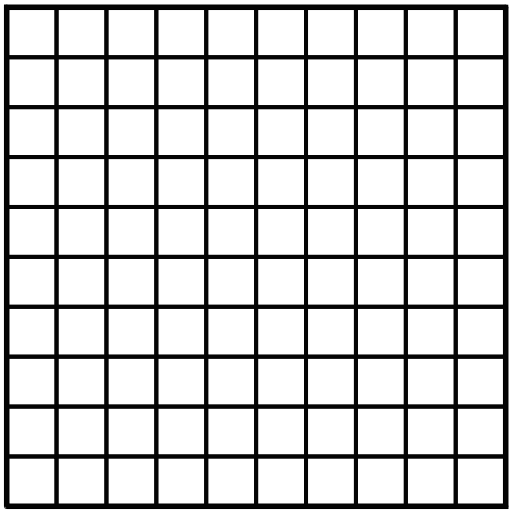


Figure A.1 Maillage orthogonal composé d'hexaèdres

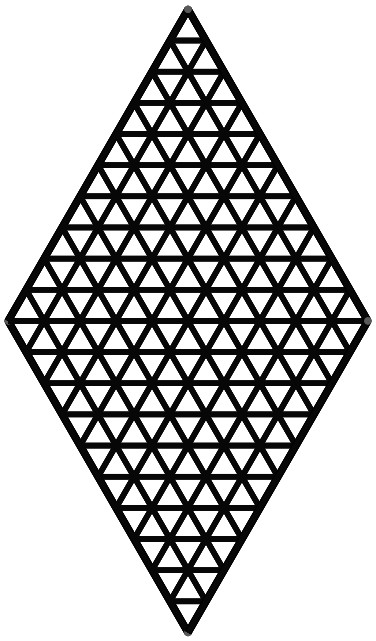


Figure A.2 Maillage orthogonal composé de prismes orthogonaux

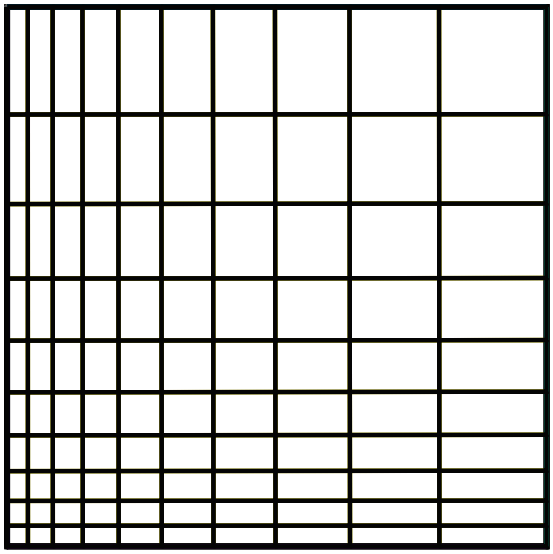


Figure A.3 Maillage orthogonal composé d'hexaèdres allongés

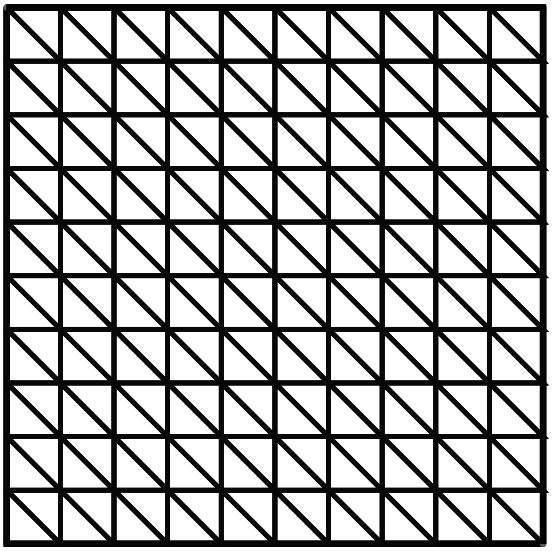


Figure A.4 Maillage non orthogonal composé de prismes orientés (vers la gauche) *Left*

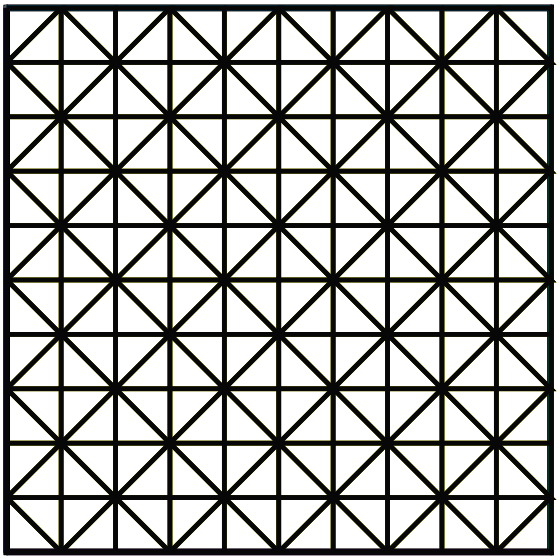


Figure A.5 Maillage non orthogonal composé de prismes alternés *Alternate*

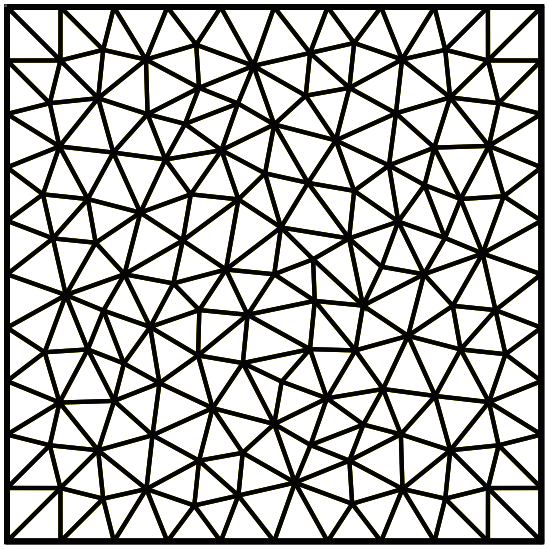


Figure A.6 Maillage non orthogonal composé de prismes Delaunay *Delaunay*

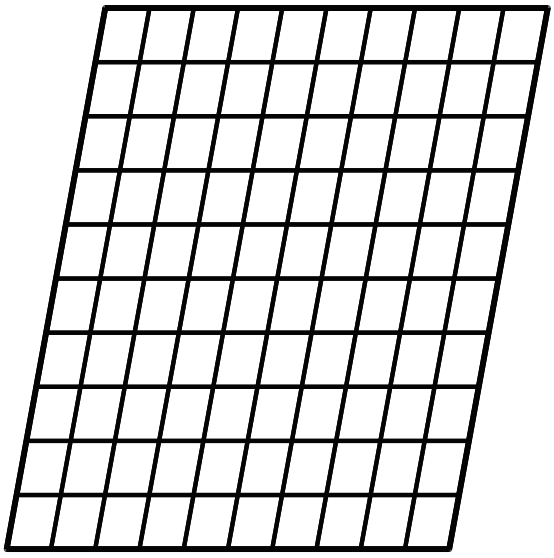


Figure A.7 Maillage non orthogonal incliné composé d'hexaèdres *Inclined*

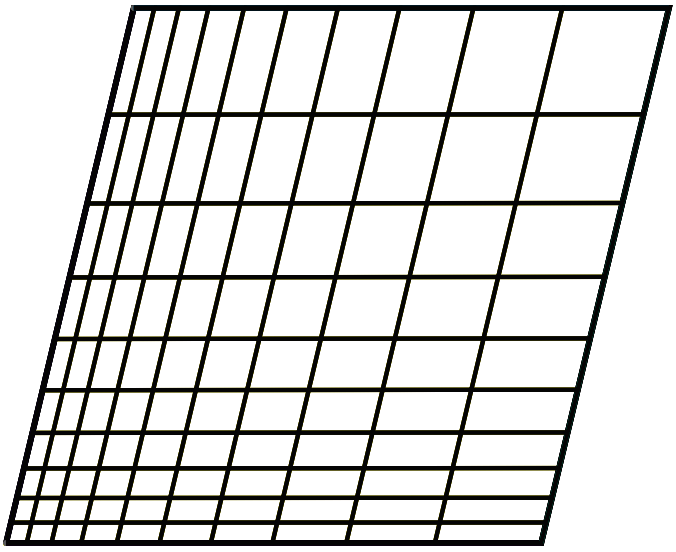


Figure A.8 Maillage non orthogonal incliné composé d'hexaèdres étirés *Streched*

ANNEXE B ARTICLE 1. A Case-Study in Open-Source CFD Code Verification. Part I : Convergence Rate Loss Diagnosis.

Dans l'article I, l'objectif est de vérifier les opérateurs de convection-diffusion disponibles dans les bibliothèques OpenFOAM et de détecter la source de dégradation de la convergence sur des maillages non orthogonaux.

Nom de l'article en anglais :

"A Case-Study in Open-Source CFD Code Verification. Part I : Convergence Rate Loss Diagnosis."

Auteurs :

Harold Noriega ; François Guibault ; Marcelo Reggio ; Robert Magnan.

L'article a été soumis au journal :

Mathematics And Computers in Simulations. 12-10-2016.

A Case-Study in Open-Source CFD Code Verification. Part I: Convergence Rate Loss Diagnosis.

H. Noriega[a,*], F.Guibault[a], M.Reggio[a], R.Magnan[b]

[a] *2900, boul. Édouard-Montpetit, Campus de l'Université de Montréal, 2500, chemin de Polytechnique, Montréal (Québec) H3T 1J4*

[b] *Institut de recherche d'Hydro-Québec, 1800 Boulevard Lionel-Boulet, Varennes, QC J3X 1S1, Québec, Canada.*

Abstract

This study analyzes the influence of cell geometry on the numerical accuracy of convection-diffusion operators in OpenFOAM. The large variety of solvers and boundary conditions in this tool, as well as the precision of the finite-volume method in terms of mesh quality, calls for a verification process performed in steps. The work is divided into two parts. In the first (the current manuscript), we focus on the diffusion operator, which has been found to exhibit a loss in convergence rate. Although the cell-centered finite volume approach underlying OpenFOAM should preserve a theoretical second order convergence rate, loss of convergence order is observed when non-orthogonal meshes are used at the boundaries. To investigate the origins of this problem, the method of manufactured solutions is applied to yield analytical solutions for the Poisson equation and compute the numerical error. The root cause is identified and corrections to recover second-order convergence are proposed. In part two of this investigation, we show how convergence can be improved, and present results for problems described by the Poisson and Navier-Stokes equations.

Keywords: Verification, Manufactured solution, OpenFOAM, CFD, Poisson equation, Meshing.

1. Introduction

Numerical calculations in fluid mechanics have developed considerably in recent decades, in part due to advances in computational power and in part because of advances in numerical methods. However, despite this progress, the simulation of real-world problems with complex geometries remains a major challenge. In fact, analytical solutions do not exist for these problems, and instead the equations representing a particular physical problem are solved via numerical simulation, which involves the use of approximations

*Corresponding author

Email addresses: harold.noriega@polymtl.ca (H. Noriega), francois.guibault@polymtl.ca (F.Guibault), marcelo.reggio@polymtl.ca (M.Reggio), magnan.robert@ireq.ca (R.Magnan)

from various sources.

The finite volume method (FVM), based on the laws of conservation, is currently one of the methods most often used to solve the Navier-Stokes equations for predicting industrial flows. Theoretical support for FVM on complex geometries can be found in the Computational Fluid Dynamics (CFD) literature [1], [2], and [3].

OpenFOAM®, a trademark of The OpenFOAM Foundation, is a free, open source software package that is widely used for fluid flow simulations in both industry and academia. OpenFOAMs theoretical support, based on cell-centered finite volume discretization, includes a large variety of solvers, schemes, and processing tools for solving a wide range of problems in computational fluid mechanics. OpenFOAM support and theoretical background can be found in [4, 5, 6, 7, 8, 9, 10, 11, 12].

Flow simulations are routinely performed in industry, and OpenFOAM accuracy must be verified on the meshes that usually accompany the complex geometries in these simulations. Because of the complexity and extent of OpenFOAM libraries and tools, the verification process has to be performed in steps. However, there is little in the literature on the verification of OpenFOAM schemes, which is why we are addressing this important issue here. A background discussion, along with definitions and descriptions for some terms related to confidence building in CFD, was presented by Roache [13] and extended by Stern et al. [14]. There, validation is described as solving the right equations, and verification as solving the equations in the right way. The various aspects discussed in Roache's paper include the distinction between code verification and code validation, between the verification of code and the verification of calculations, grid convergence vs. iterative convergence, and numerical error vs. conceptual modeling error.

Along these lines, Abanto et al. [15] presented a grid convergence study on some atypical CFD cases using a number of commercial CFD packages. Their verification test cases are distinctive in that exact solutions are known. Verification test cases include the classical Poiseuille flow, an incompressible recirculating flow, a manufactured incompressible laminar boundary layer flow, and an incompressible annular flow. Different convergence rates are determined using structured and unstructured meshes. Stern et al. [14] describe a set of verification, validation, and certification methodologies and procedures for numerical simulations. Examples of the application of quantitative certification of Reynolds-Averaged Navier-Stokes (RANS) codes are presented for ship hydrodynamics.

In Tremblay et al. [16], the method of manufactured solutions (MMS) for fluid-structure interaction code verification is applied. These researchers observed that, when used with systematic grid refinement, MMS provides strong code verification. Roy et al. [17] used MMS to verify the order of accuracy of two finite-volume Euler and Navier-Stokes codes. These exact solutions were used to accurately evaluate the discretization error in the numerical solutions. Through global discretization error analysis, the spatial order of accuracy was observed to be second order for a node-centered approach using unstructured meshes. More recently, Ianelli [18] compared an exact solution and CFD solutions of the Navier-Stokes equations. They determined the convergence rates and

orders of accuracy of these solutions and illustrated the utility of the exact solution developed for verification purposes. There is a series of published papers on MMS for 2D incompressible Navier-Stokes and turbulent models, which includes those by Eca and Hoestra [19, 20] and Eca et al. [21].

A recent study regarding verification for an unstructured finite volume CFD code has been published by Veluri and Roy [22], in which MMS is used to generate exact solutions to both the Euler and Navier-Stokes equations to verify the order of accuracy of the code on different grid types in 2D and 3D. The various options for code verification include the baseline steady-state governing equations, transport models, turbulence models, boundary conditions, and unsteady flows. Diskin et al. [23] studied the accuracy and complexity in finite volume discretization schemes for viscous fluxes on general grids using a node-centered scheme and three cell-centered schemes (a node-averaging scheme and two schemes using least-squares face-gradient reconstruction). Among several interesting results, they found that the node-averaging scheme has the highest complexity and can fail to converge to the exact solution when the node-averaged values are clipped. On highly anisotropic grids, the least-squares schemes, the node-averaging scheme without clipping, and the node-centered scheme demonstrated similar second-order accuracies per degree of freedom. Overall, the accuracies of the node-centered and the best cell-centered schemes are comparable at an equivalent number of degrees of freedom on isotropic and curved anisotropic grids. A similar work for inviscid fluxes is presented by Diskin and Thomas [24], the second-order cell-centered and node-centered approaches for finite volumes were compared for unstructured grid discretizations in two dimensions. Some weaknesses were observed in all these schemes, such as instability, accuracy degradation, and/or poor convergence of defect-correction iterations. In a more recent work, Diskin and Thomas [25] studied the effects of mesh regularity on the accuracy of unstructured node-centered finite-volume discretizations. In their paper, they focused on an edge-based approach which uses unweighted least-squares gradient reconstruction with a quadratic fit.

Along these lines, the goal of the present two-part study is the verification of OpenFOAM libraries used by solvers, and detection of the influence of the cell geometry on the numerical accuracy of OpenFOAM Convection-Diffusion operators. Here, in the first part of our study, we focus on the diffusion operator. To ensure the correctness of the code libraries, the theoretical second order convergence rate for the cell-centered approach must be preserved. We have considered the verification of the OpenFOAM convergence rate under three types of boundary condition (Dirichlet, Neumann, and Periodic). Three criteria of mesh quality (non-orthogonality, non-uniformity, and skewness) are taken into consideration in the verification process. For this purpose, we investigate how the error induced by mesh distortions affects the convergence rate of numerical simulations under systematic mesh refinement.

The paper is organized as follows: first, we present the formulation of the mathematical models, followed by the meshes used, the manufactured solutions applied, and the errors introduced. The following section presents numerical results: first on orthogonal meshes, and then on non-orthogonal ones. For the latter, a reduction in convergence rate compared to what is predicted by theory is observed. This loss of accuracy is further

explored and related to the treatment of non-orthogonality at the boundary. The root cause of this loss of accuracy is identified and a solution is proposed. The final section presents our concluding remarks and perspectives.

2. Mathematical Models and Discretization

Here, in this first part of our study, we reduce the verification procedure for the convection and diffusion schemes to a convergence analysis on the diffusion scheme, for which a reduction in convergence rate has already been observed on non-orthogonal meshes, as we will see later. We will begin by considering the Poisson equation, written as:

$$-\nabla \cdot (\nu \nabla \phi) = q \tag{1}$$

in which ϕ represents a scalar field, ν the diffusion coefficient, and q the source term.

To discretize the equation 1 using the FVM, we integrate over the elementary control volume V_p limited by the control surface S_p, as follows:

$$\int_{V_p} \nabla \cdot (\nu \nabla \phi)\, d\tau = \oint_{S_p} d\vec{S} \cdot (\nu \nabla \phi) = \int_{V_p} q_\phi(\vec{x}) d\tau \tag{2}$$

A brief introduction to the FVM discretization of equation 2 is provided below.

2.1. Numerical Schemes

An overview of the theoretical basis of the OpenFOAM schemes analyzed in this work is presented below (see [4], [10]).

2.1.1. Spatial Schemes

OpenFOAM is based on finite volume schemes that use the center $\vec{x}_P$ of the cell to define the variation in space of a scalar function $\phi = \phi(\vec{x})$. In order to yield second order accurate schemes, this variation needs to be linear inside a control volume V_p (see [4], [10]). So,

$$\phi(\vec{x}) = \phi(\vec{x}_P) + (\vec{x} - \vec{x}_P) \cdot (\nabla \phi)(\vec{x}_P) + O(|\vec{x} - \vec{x}_P|^2) \tag{3}$$

where $\cdot$ is the inner product.

In fact, considering $P = \vec{x}_P$ as the centroid of V_P to perform a volume integral on Eq. 3, the integral of the linear term on the right-hand side disappears and a second order integral approximation is obtained:

$$\int_{V_p} \phi(\vec{x}) d\tau = \phi_P V_P + O(|\vec{x} - \vec{x}_P|^2) \tag{4}$$

where $\phi_P = \phi(\vec{x}_P)$.

Similarly, considering a scalar quantity ϕ_{C_f} evaluated at the center of a face (Fig. 1a), we can obtain:

$$\int_{S_f} \phi(\vec{x}) d\vec{S} = \phi_{C_f} \vec{S}_{C_f} + O(|\vec{x} - \vec{x}_{C_f}|^2) \tag{5}$$

in which $\vec{S}_{C_f}$ indicates the surface area vector $\vec{S}$ evaluated at the center C_f of the face f, and the magnitude of S_{C_f} is the area of the face.

Along a face f, a scalar property ϕ_{C_f} can be evaluated by linear interpolation of ϕ using information from adjacent elements stored on nodes $P = \vec{x}_P$ and $N = \vec{x}_N$ (see [4], [10]):

$$\phi_{C_f} = \alpha \phi_P + (1 - \alpha) \phi_N \tag{6}$$

where α is an interpolation factor that can be defined as the ratio of distances $\overline{C_f N}$ over $|\vec{d}| = \overline{PN}$, and so,

$$\alpha = \frac{\overline{C_f N}}{\overline{PN}} \tag{7}$$

Alternatively, we can use midpoint interpolation, that is, a symmetric weighting ($\alpha = 0.5$) interpolation from nodes P and N, or a cubic interpolation derived from the linear interpolation, with the addition of an explicit higher correction. This correction is calculated using the gradient values $(\nabla\phi)_P$ and $(\nabla\phi)_N$.

If the control volume V_p is bounded by a control surface composed of flat faces f, the divergence terms can be approximated using the Gauss-Ostrowsky theorem (see [4], [10]). The divergence term of the vector property $\vec{v}$ can be approximated as follows:

$$\int_{V_p} (\nabla \cdot \vec{v}) d\tau = \oint_{S_p} d\vec{S} \cdot \vec{v} = \sum_f \int_{S_f} d\vec{S} \cdot \vec{v} \approx \sum_f \vec{S}_{C_f} \cdot \vec{v}_{C_f} \tag{8}$$

where the components $\vec{v}_{C_f}$ can be evaluated using the interpolation Eq. 6.

For a scalar, the gradient term can be obtained using either the divergence theorem or a least-squares fit (see [4], [10]). Discretization based on the divergence theorem can be written as follows:

$$\int_{V_p} \nabla\phi d\tau = \oint_{S_p} d\vec{S}\phi = \sum_f \int_{S_f} d\vec{S}\phi \approx \sum_f \vec{S}_{C_f} \phi_{C_f} \tag{9}$$

where ϕ_{C_f} is evaluated using the interpolation Eq. 6.

Taking into account a linear variation of ϕ over the control volume P, the cell-centered approach for the gradient at P is calculated as follows:

$$(\nabla\phi)_P = \frac{1}{V_p} \sum_f \vec{S}_f \phi_{C_f} \tag{10}$$

This discretization generates the OpenFOAM Gauss scheme for the gradient.

The least squares fit approach is based on minimizing the following functional with respect to the gradient $(\nabla\phi)_P$ (see [10]):

$$L(\nabla\phi_P) = \sum_f \left(\frac{\phi_N - \phi_P}{|\vec{d}|} - \frac{\vec{d}\cdot(\nabla\phi)_P}{|\vec{d}|} \right)^2 \tag{11}$$

The numerical expression $\frac{\phi_N - \phi_P}{|\vec{d}|}$ in Eq. 11 represents a linear approximation of the gradient $(\nabla\phi)_P$ projected on vector $\vec{d}$ (see Fig. 1b). This approximation generates a second order least squares scheme for the gradient. The optimum gradient $(\nabla\phi)_P$, defined as the gradient that minimizes Eq. 11, can be found by solving the system of linear equations generated by the minimizing condition:

$$\frac{dL}{d(\nabla\phi)_P} = 0 \tag{12}$$

The diffusion term is discretized using Eq. 8 (see [10]):

$$\int_{V_p} \nabla\cdot(\nu\nabla\phi)\, d\tau = \oint_{S_p} d\vec{S}\cdot(\nu\nabla\phi) = \sum_f \int_{S_f} d\vec{S}\cdot(\nu\nabla\phi) \approx \sum_f \vec{S}_{C_f}\cdot\left(\nu_{C_f}\nabla\phi_{C_f}\right) \tag{13}$$

where ν represents the diffusivity (considered constant in this work). This discretization defines the Gauss Laplacian scheme, which is the only means available in OpenFOAM for discretizing the Laplacian operator.

In our case, the source term $q = q(\vec{x})$ on the right-hand side of Eq. 1 depends only on spatial coordinates, and is obtained by implementing a manufactured solution for verification purposes. A second order integral approximation for the control volume V_P is obtained as follows:

$$\int_{V_p} q(\vec{x})d\tau = q_P V_P + O(|\vec{x} - \vec{x}_P|^2) \tag{14}$$

where $q_P = q(\vec{x}_P)$.

The second order discretized expression for the Poisson Eq. 2 can now be written as:

$$\sum_f \vec{S}_{C_f}\cdot(\nu\nabla\phi)_{C_f} = q_P V_P \tag{15}$$

The gradient $(\nabla\phi)_{C_f}$ in Eq. 15 is expressed through the values of ϕ_P and the values of ϕ in the surrounding cells. Accordingly, Eq. 15 leads to an algebraic equation (see [4]) for cell P:

$$a_P\phi_P + \sum_N a_N\phi_N = R_P \tag{16}$$

where a_P, a_N and R_P are constants produced by the discretization process.

Assembling Eq. 16 for all cells centers P, a system of algebraic equations is obtained:

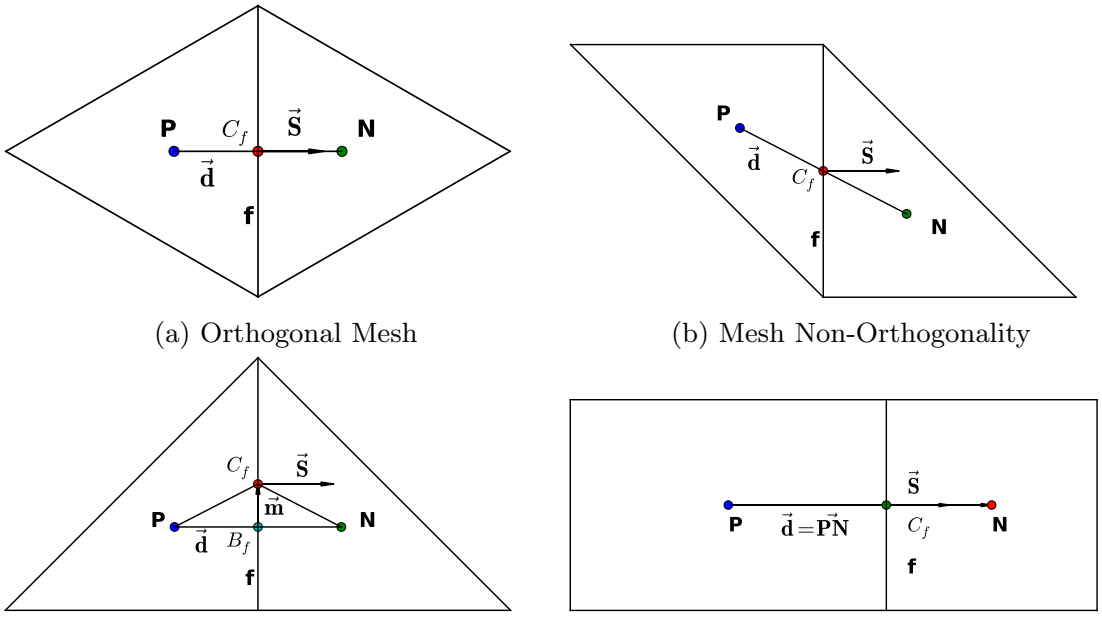
$$[A]\,[\phi] = [R] \tag{17}$$

which is solved iteratively.

2.2. The Meshes.

Mesh shape and size are key factors determining the accuracy of a numerical approximation. Juretic [10] defines three properties that affect mesh quality:

- **Non-Orthogonality.** An orthogonal mesh can be defined as a mesh for which the vector $\vec{d}$ joining the cell center P and the center N of an adjacent cell is parallel to the vector $\vec{S}$ normal to a cell face f (see Fig. 1a). A mesh that does not satisfy the orthogonality condition is defined as non-orthogonal (see Fig. 1b)
- **Mesh Skewness.** When the segment $\overline{PN}$ does not intercept the centroid point C_f of the face f, the mesh is defined as skewed (see Fig. 1c).
- **Non-Uniformity.** A mesh is uniform when $\vec{d}$ intersects the face midway between the nodes P and N. A mesh that does not satisfy the uniformity condition is defined as non-uniform (see Fig. 1d)



(a) Orthogonal Mesh

(b) Mesh Non-Orthogonality

(c) Mesh Skewness

(d) Mesh Non-Uniformity

Figure 1: Mesh properties.

We have chosen a variety of meshes for the verification procedure to enable us to test these three properties. After performing some basic tests, we will focus on the non-orthogonality of the grid for which a loss of convergence order is observed. We use orthogonal hexahedral meshes (shown in Figs. 2a, 2b, and 2c), and non-orthogonal prism meshes (shown in Figs. 2d, 2e, and 2f). Non-orthogonal hexahedral meshes are depicted in Figs. 2g and 2h. A physical domain $[0,\pi][0,\pi]$ is used for the square domains.

For the slanted domains (parallelograms), the upper boundary is moved to the right by approximately 16% of the characteristic length, and the rhombus domain is chosen in order to obtain equilateral prisms.

2.3. Treatment of Non-Orthogonality

For non-orthogonal grids (see Fig. 3), the diffusive flux on a given face f has more than one component, as is the case for orthogonal grids. Several multi-point face schemes have been presented by Schafer [1], however using these schemes may increase the complexity of the computations and cause difficulties in converging the solvers.

To deal with non-orthogonality, a one-point centroid-based scheme has been proposed by Jasak [4]. This scheme uses a first term that is calculated using the cell centers P and N neighboring the face f (see Fig. 1b). This term is called the orthogonal contribution. A second term is then added to provide the gradient projection of the outward-pointing face area vector $\vec{S}_f$ with a "correction". This is called the non-orthogonal contribution. The mathematical expression of this idea (see Eq.18) is:

$$\vec{S} \cdot (\nabla\phi)_f = |\vec{\Delta}| \frac{\phi_N - \phi_P}{|\vec{d}|} + \vec{k} \cdot (\nabla\phi)_f \tag{18}$$

where f denotes the face of a control volume, $(\nabla\phi)_f$ the gradient of a physical property ϕ at face f, $|\vec{\Delta}| \frac{\phi_N - \phi_P}{|\vec{d}|}$ the orthogonal contribution and $(\vec{k} \cdot (\nabla\phi)_f)$ the non-orthogonal correction.

The non-orthogonal contribution term in Eq. 18 corresponds to a deferred correction approach. As explained in [3], deferred correction approaches are used in FVM to, among other things, define higher-order schemes. Muzaferija [26] suggested a second order deferred correction scheme to deal with non-orthogonality that uses a correction vector.

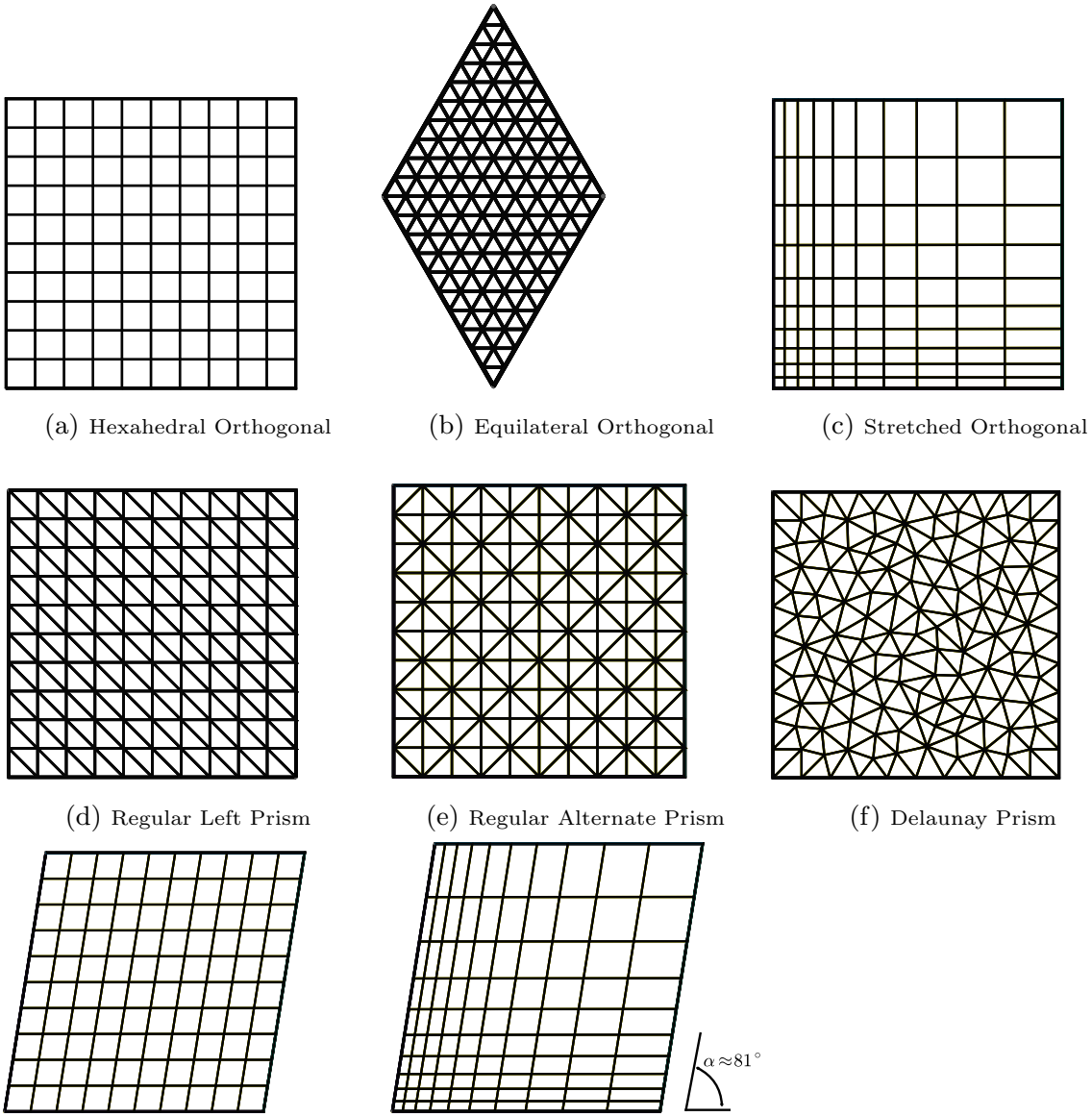
Among a set of possible non-orthogonal decomposition contributions in equation 18, Jasak [4] examines three, which are called the "minimum correction", "orthogonal correction", and "over-relaxed" approaches (see [5]). These three ways to compute the non-orthogonal correction are illustrated in Figs. 4, 5, and 6 respectively. What these three approaches have in common is that they decompose the surface normal vector $\vec{S}$ at the center of face f as follows (see [4]):

$$\vec{S_f} = \vec{\Delta} + \vec{k} \tag{19}$$

The vector $\vec{\Delta}$ is specifically calculated for each of the three approaches:

- For the "minimum correction" (see Fig.(4)), the calculation is performed in such a way as to keep the non-orthogonal correction as small as possible:

$$\vec{\Delta} = \cos(\alpha_N)|S|\vec{d_u} \tag{20}$$



(a) Hexahedral Orthogonal (b) Equilateral Orthogonal (c) Stretched Orthogonal

(d) Regular Left Prism (e) Regular Alternate Prism (f) Delaunay Prism

(g) Hexahedral Slanted (h) Hexahedral Slanted Stretched

Figure 2: Orthogonal and Non-Orthogonal Prism Meshes in 2D Representation

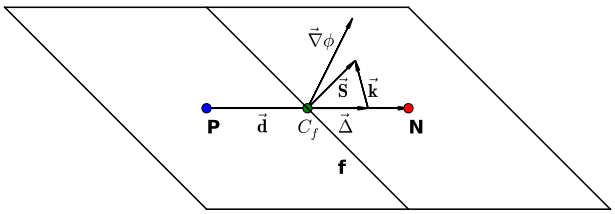


Figure 3: Non-Orthogonality Treatment

- For the "orthogonal correction" (see Fig.(5)), the contribution from ϕ_P and ϕ_N is kept the same as on the orthogonal mesh:

$$\vec{\Delta} = |S|\vec{d_u} \tag{21}$$

- In the "over-relaxed approach" (see Fig. 6), the importance of the terms ϕ_P and ϕ_N increases with the increase in non-orthogonality:

$$\vec{\Delta} = \frac{|S|}{\cos(\alpha_N)}\vec{d_u} \tag{22}$$

where vector $\vec{d_u} = \frac{\vec{PN}}{|\vec{PN}|}$ is the unit vector $\vec{d}$, $|S|$ is the area of face f, and $\cos(\alpha_N)$ is the cosine of the angle between the unitary vector $\vec{d_u}$ and the normal face vector $\vec{n} = \frac{\vec{S}}{|S|}$.

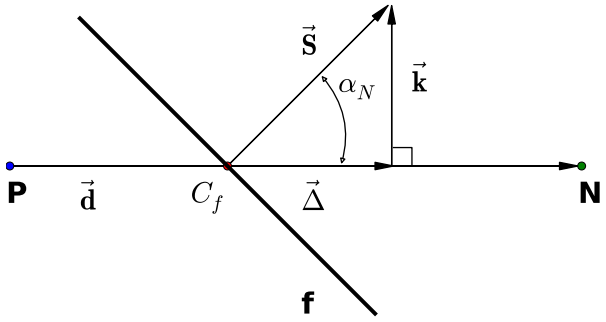


Figure 4: Non-Orthogonality Treatment: "Minimum Correction" Approach

The vector $\vec{k}$ is calculated from the equation:

$$\vec{k} = \vec{S_f} - \vec{\Delta} \tag{23}$$

Finally, note that the scheme presented in Eq. 18 is a non-linear equation owing to the non-orthogonal contribution. The implementation of this scheme needs the value of the gradient in the non-orthogonal contribution. This is achieved by means of a recursive process in which the non-orthogonal contribution is explicitly calculated on

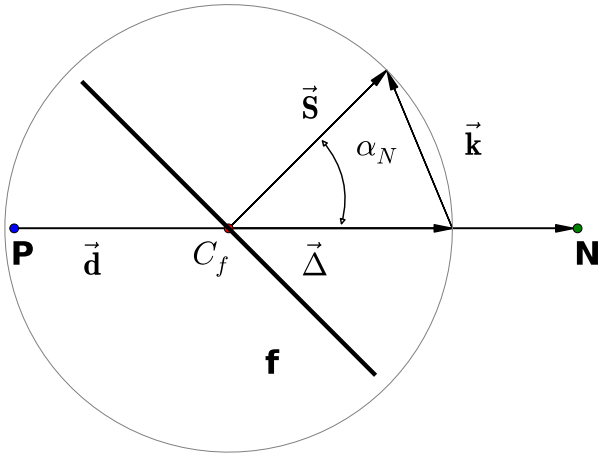


Figure 5: Non-Orthogonality Treatment: "Orthogonal Correction" Approach

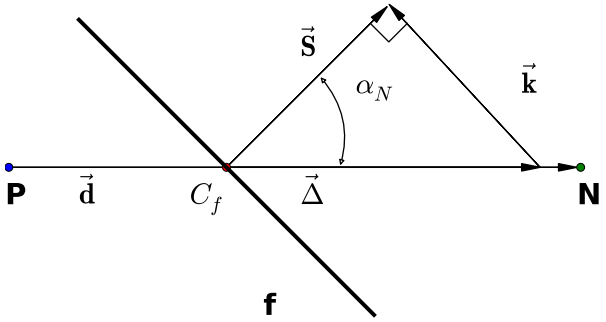


Figure 6: Non-Orthogonality Treatment: "Over-relaxed" Approach

the right-hand side of the matrix of the algebraic system of equations (see Eq. 17). Another important aspect is the limitation of the non-orthogonal contribution, which can slow down the convergence of the iterative method solving the matrix system. All three methods shown by Jasak for computing $\vec{k}$ yield a constraint on such a vector. An alternative that will ease this problem is to apply a limiter to the non-orthogonal contribution. In OpenFOAM, a constraint is added to the over-relaxed approach in order to prevent the non-orthogonal part from becoming dominant, as we will see later in this work (see Eq. 30).

3. Methodology

In this section, we begin by recalling the Method of Manufactured Solutions (MMS), which is a powerful verification tool for determining the convergence rate through the calculation of exact numerical error using mesh refinement. Next, we present the norms used to obtain the average and maximum errors. Finally, we present the mathematical expression for calculating the order of accuracy in the logarithmic scale.

3.1. Method of Manufactured Solution

The general idea behind MMS is to use a custom built source term that allows Eq. 1 to have an analytic solution. The steps of the MMS methodology can be described as follows (see [27]):

- Propose a manufactured solution for a system of equations.
- Obtain a source term for the initial equations.
- Implement that source term in the solver.
- Solve the new equation.
- Compute the error.

In the present work, a scalar manufactured solution is used to verify the Poisson equation when using Dirichlet and Periodic (Cyclic) boundaries:

$$\phi(x, y) = \phi_0 \left[\cos(\omega x) \sin(\omega y)\right] \tag{24}$$

where the frequency ω and the amplitude ϕ_0 of the sinusoid are set to 1 for Dirichlet boundaries. Periodic boundaries are easily tested if $\omega = 2.0$ is set to ensure the periodicity of the solution.

The advantage of this choice of solution is that the source term obtained is multiplied by ω^2, which is a product that can be used to avoid convergence problems in the explicit part of the matrix system equation 17. Other manufactured solutions will also be used in this work to test the Neumann boundary condition. The source term produced by the manufactured solution for Poisson's Eq. 1 is:

$$q(x, y) = \quad 2\nu\omega^2\phi(x, y) \tag{25}$$

3.2. Error Estimation

The exact error e_i at the cell center "i" is the difference between the analytical manufactured solution $\tilde{\phi}$ and the numerical solution ϕ_h:

$$e_i = \tilde{\phi} - \phi_h \tag{26}$$

To calculate the order of convergence, we use two types of norm. The first, for the average error, is calculated as follows:

$$||e||_{L_1(\Omega)} = \int_\Omega |e| \, d\Omega \approx \frac{1}{|\Omega|} \sum_{i=1}^{NC} \Omega_i \, |e_i| \tag{27}$$

where Ω indicates the domain definition, Ω_i is the control volume of cell i, and NC indicates the number of cells. The second norm, for the maximum error, is calculated as follows:

$$||e||_\infty = \sup_{x \in \Omega} |e(x)| \approx \max_i |e_i| \tag{28}$$

where $i = 1, ..., NC$.

3.3. Convergence Rate

The correctness of a computer code is verified via systematic mesh refinement. In our case, we uniformly divide a parallelogram domain, with N_k being the number of nodes along each boundary, taking the values $N_k = [10, 20, 40, 60, 80, 100]$. This sequence refines the control cell at each step for all meshes except for the Delaunay mesh (see Fig. 2f).

For this mesh, the refinement is applied only on boundary faces, after which the cells are automatically constructed using a Delaunay algorithm without a strict guarantee that the cells will be systematically refined by a constant factor inside the domain. Strictly speaking, the simulations using this mesh only illustrate the trend of convergence when the boundary is refined, and do not constitute a measure of convergence based on the systematic internal refinement of cells.

If e_k represents the error produced by partitioning the domain N_k times (with [10, 20,-40, 60, 80, 100]), convergence is defined as:

$$R_k = \frac{\log\left(e_k/e_{k-1}\right)}{\log\left(N_k/N_{k-1}\right)} \tag{29}$$

*3.4. Boundary Conditions (***BCs***)*

In this work, the verification procedure is only performed for three basic types of BC: the Dirichlet, Neumann, and Periodic boundaries. The convergence rate issues revealed with respect to the Dirichlet and Neumann BCs explain why we initially restrict the analysis to these types of BC.

3.5. OpenFOAM Spatial Schemes For The Diffusion Operator

In this section, the OpenFOAM implementation of the Laplacian scheme for the diffusion operator is described. This scheme is based on the expression on the right-hand side of equation 13 for which an interpolation, a gradient, and a non-orthogonal scheme are required. A number of options for each of these schemes are presented below.

As emphasized in the OpenFOAM documentation (see [28], [29]) the "Gauss scheme" is the only choice for the Laplacian operator. The Gauss scheme syntax is composed of the name "Gauss", an interpolation scheme for the diffusion coefficient ν considered constant in this study, and a selection of a limiter to the surface normal gradient for the non-orthogonal treatment:

```
Gauss <interpolationScheme> <snGradScheme>
```

The many options for the surface normal gradient scheme, snGradScheme, are the following (see [28], [29]):

- corrected - unbounded, second order, conservative
- uncorrected - bounded, first order, non-conservative
- fourth - fourth order, conservative
- limited ψ - blend of corrected and uncorrected

where the parameter ψ limits the non-orthogonal part. Typical values are:

- $\psi = 1.0$ corresponding to a corrected, unbounded, second order, conservative scheme
- $\psi = 0.5$ corresponding to a non-orthogonal correction $\leq$ of the orthogonal part
- $\psi = 0.333$ corresponding to a non-orthogonal correction $\leq \frac{1}{2}$ of the orthogonal part

- $\psi = 0.0$ corresponding to an uncorrected, bounded, first order, non-conservative scheme

The role of the parameter ψ is to serve as a constraint to the non-orthogonal part, as follows:

$$\lambda \left(\vec{k} \cdot \nabla \phi \right) \tag{30}$$

where λ is calculated as:

$$\lambda = \min \left(\frac{(\psi)|\texttt{orthogonal part}|}{(1-\psi)|\texttt{non-orthogonal correction}| + \epsilon}, \ 1 \right) \tag{31}$$

where ϵ is a "small" value.

The many schemes of the gradient operator, which is part of the non-orthogonal treatment, are: Gauss, least squares, and fourth, and include CellLimited and FaceLimited versions of each. The Gauss and least-squares schemes are based on Eqs. 10 and 11 respectively. The fourth scheme is a fourth order least squares approach. Following is a brief description of each of these (see [28], [29]):

- Gauss second order, based on Gaussian integration
- Least Squares second order, based on the least-squares approach
- Fourth fourth order, fourth order least squares approach

OpenFOAM offers three main schemes to interpolate the values from cell centers to face centers: "linear", based on Eq. 6; "midpoint" which is linear with symmetrical weights $\alpha = 0.5$ and "cubic", which is a cubic scheme implemented as a correction to the linear scheme.

Before ending this section, we wish to point out that the verification tests presented in this work were performed on OpenFOAM-2.3.x and Foam-Extend-3.1, which serve as references for other OpenFOAM versions

4. Verification Results

This section reports on the convergence tests that make it possible to detect and track the origin of the loss of convergence accuracy. First, we comment on the verification procedure for orthogonal meshes. Then, we verify the convergence rate on various non-orthogonal meshes for different BCs. Finally, we show the verification strategy for detecting the origin of the convergence loss when non-orthogonal meshes are used.

In the figures presented in this section, the error is represented as a function of the refinement steps on a logarithmic scale for which the slope of functions represents the convergence order (see section 3.3). The labels Avg. and Max. indicate that the average and the maximum error norms respectively are used to calculate the convergence rate for the numerical solutions. To perform the tests on 2D meshes, we use a single layer of cells with empty BCs in the front and back. Note that, unless explicitly stated otherwise,

the manufactured solution illustrated by Eqs. 24 is used hereafter on a square domain $[0;\pi][0;\pi]$, applying the Dirichlet BC along the boundaries. The diffusion coefficient is supposed to be constant and its interpolation scheme is not tested.

4.1. Initial Test Verification

The strategy for verifying the convection-diffusion operators follows the verification methodology proposed by Knupp and Salari [27] to test Burger's equation using the Dirichlet and Neumann BCs. In this way, an initial set of tests for Burger's equation is successfully performed in terms of the theoretical convergence rate on orthogonal meshes (see Fig. 2a - 2c). In these tests, the Dirichlet BC is first applied on all the boundaries, and then simulations are performed combining the Dirichlet and Neumann BCs on alternating boundaries. Theoretical second order accuracy is obtained for all the BC combinations. These results are not illustrated in this work.

This first set of verification tests is also performed for Left non-orthogonal regular prism meshes (see Fig.2d) using second order schemes for Burger's equation. A reduction to first-order accuracy is observed when only the Dirichlet BC is used. For this reason, to simplify the problem, the initial goal of verifying the convection-diffusion operators focuses on verifying the diffusion term via a Poisson equation.

4.2. Verification of the Poisson Equation on Orthogonal Meshes for the Dirichlet BC

In this section, a set of verification tests for Poisson's equation on orthogonal meshes using the Dirichlet BC is presented. To perform these tests, the orthogonal meshes presented in Fig.(2a - 2c) are used. The first is composed of regular hexahedra, the second of equilateral prisms (for this, the domain $[-1;1][-\sqrt{3};\sqrt{3}]$ is used), and the third of stretched hexahedra with a ratio of 4:1 (i.e. each edge of the bottom-left cell is one-quarter the length of each side of the upper-right cell). The Gauss linear orthogonal scheme is used for the Laplacian.

In Fig. 7, the convergence rates for the Avg. and Max. errors on hexahedral (Hex.), hexahedral stretched (Stret.), and equilateral (Equil.) prism meshes are presented. The six curves in this plot have a slope of 2, which corresponds to the predicted theoretical second order accuracy. The results obtained for the Stret. meshes verify that, for these simulations, the convergence is not impacted by non-uniformity. Following this step, tests on elongated orthogonal meshes are presented (on domain $[-0.4:0.4];[-0.3:0.4]$), first with a refinement by a factor of 10 in the x-direction using 40x4, 80x8, 160x15, 320x32, 640x64, and 1280x128 partitions, and then with refinement by a factor of 10 in the y-direction. Although the Hex. meshes are strongly elongated in one direction, the theoretical convergence rate of 2 is preserved for the Avg. and Max. errors, as observed in Fig. 8. In this graph, [10:1] means 10 times more refined in the x-direction. Second-order accuracy is also observed in simulations combining the Dirichlet and Neumann BCs on alternating boundaries on square domains, however those results are not illustrated here.

4.3. Grid Refinement Results on Non-Orthogonal Meshes

On non-orthogonal meshes, we test all combinations of the scheme options supported for the Laplacian scheme, as explained in section 3.5: the interpolation, gradient, and non-orthogonal schemes. For this reason, the verification process is divided into several stages. Next, we present the most representative results. For these simulations, unless otherwise specified, the Gauss linear scheme is used for the gradient appearing in the non-orthogonal contribution.

4.3.1. Testing the OpenFOAM Laplacian Non-Orthogonal Constraint Using the Dirichlet BC.

In the first step, the parameter $\psi = [1.0, 0.5, 0.333, 0.0]$, explained in section 3.5, for the non-orthogonal treatment is tested. The verification tests are performed using only a Left non-orthogonal regular prism mesh, as presented in Fig. 2d.

Significant differences in the results are found for the various values of ψ, and the best performance in terms of convergence rate is obtained for the value $\psi = 1.0$, but, unfortunately, this is only a first-order convergence. As can be appreciated in Fig. 9, the Left Avg. $\psi = 1.0$ and Left Avg. $\psi = 0.5$ curves are very close in terms of convergence and accuracy, as a convergence rate of 1.0 is observed. The Left Avg. $\psi = 0.333$ curve has an initial slope of 1.0, but, when refined, its convergence rate decreases considerably. The curve with non-orthogonal correction, Left Avg. $\psi = 0.0$, has a zero convergence rate. In these plots, first-order accuracy is also observed for the maximum error in the Left Max 1.0 curve (using $\psi = 1.0$), but this latter has less precision relative to that of the Left Avg. 1.0 curve.

When using Gauss cubic and Gauss midpoint instead of Gauss linear for the gradient reconstruction implicitly appearing in the non-orthogonal contribution, no improvement in convergence is observed. Neither is there any improvement in convergence when other gradient schemes are used, such as Fourth and Least Squares, including their CellLimited and FaceLimited versions. Some of these results are presented in Fig. 10. In this plot, the Gauss, Least Squares, and Fourth labels indicate the scheme used for the gradient inside the non-orthogonal term. The labels linear and cubic indicate the interpolation scheme used for the Gauss scheme. The value for ψ indicates the value in the limited scheme. As can be appreciated in Fig. 10, these curves present a convergence rate of 1. In this graph, we can also compare some curves when the Least Squares scheme is used. The Least Squares curves with $\psi = 1.0$ and $\psi = 0.5$ present a convergence rate of 1.0. For the least squares curve with $\psi = 0.333$, an asymptotic convergence rate of less than 1.0 is observed.

Unless otherwise explicitly stated, the verification tests are hereafter conducted using the Gauss linear limited scheme 1.0 for the Laplacian.

4.3.2. Results on Different Types of Non-Orthogonal Triangular Mesh.

In this section, the influence of non-orthogonality is illustrated for a variety of non-orthogonal meshes that include a Left regular prism (see Fig. 2d), an Alternate regular prism (see Fig. 2e); a Delaunay regular prism (see Fig. 2f); a Slanted hexahedral (see Fig. 2g), and a Slanted stretched hexahedral (see Fig. 2h) mesh.

In Fig. 11, we see some convergence curves for the average error and one curve for the maximum error. As presented in this figure, an asymptotic convergence rate of 1 is observed in the Left Avg. curve. The convergence rate for the Alternate Avg. curve is the expected theoretical value of 2.0, but this value is only 1.0 for the Alternate Max. curve. That is because the Alternate regular mesh keeps the orthogonality inside the domain, but this property is not preserved for the cells neighboring the boundaries (see Fig. 2e). With this result, we are able to detect the origins of non-orthogonality issues. It is also remarkable that the Alternate mesh verifies the skewness inside the domain. So, it seems that neither uniformity nor skewness participates in the convergence issues, which are likely caused only by non-orthogonal treatment on boundaries.

The Delaunay Avg. curve presents a critical loss in convergence rate for the more refined meshes. As explained in section 3.3, the simulations using this mesh only illustrate the convergence trend when the boundary is refined. Finally, the Slanted Avg. curve and the Stretched Avg. curve, using hexahedral meshes, present an asymptotic convergence accuracy of 1.0 for the average error, which is in agreement with the trend of the previous non-orthogonal meshes.

4.3.3. Testing Non-Orthogonality that Systematically Inclines the Border.

To illustrate the influence of non-orthogonality on the convergence order loss, the initial square domain $[0;\pi][0;\pi]$ is systematically inclined moving the upper boundary to the right. Initially, the upper boundary is moved one percent of the characteristic distance L (i.e., $\Delta x = 0.01L$) to the right, then this process is repeated taking $\Delta x = n(0.01L)$, with n = 2,3,4,5.

The results for average error are presented in Fig. 12. In this representation, the index 0,1,2,3,4,5 corresponds to the number of steps moved, i.e. Avg-0 corresponds to the initial orthogonal mesh, Avg-1 is obtained from the initial orthogonal moving the top boundary one step, etc. As seen in Fig. 12, the convergence rate, which is initially 2 (as we have seen for the Avg-0 curve) decreases when the number of steps is increased, and this convergence loss is strengthened as non-orthogonality increases. These results clearly indicate that convergence is influenced by non-orthogonality.

4.3.4. Non-orthogonal Mesh with a Periodic (Cyclic) BC.

In this section, simulations with a periodic BC are performed. Issues with these types of BC have not been found, either for orthogonal or non-orthogonal meshes. Two pairs of neighboring periodic BCs are used (left and right, and up and down boundaries). An Orthogonal hexahedral mesh, and two non-orthogonal Left and Alternate prism meshes, which are presented in Figs. 2a, 2d, and 2e respectively, are used. The manufactured solution specified by Eq. 24 is implemented, using the space frequency $\omega = 2.0$ to guarantee continuity of the solution.

Some results are shown in Fig. 13. In this representation, the Orthogonal Avg. and Orthogonal Max. curves correspond to the average and maximum error convergence on orthogonal meshes. Both curves have a slope of 2. The same trend is observed for some curves on a Left regular and an Alternate regular prism mesh for the average and maximum errors, as we have seen in the Left Avg. $\psi = 1.0$, Left Max. $\psi = 1.0$,

and Alternate Avg. $\psi = 1.0$ curves (note that the Orthogonal Avg. and Alternate Avg. $\psi = 1.0$ curves are superposed. The value ψ refers to the parameter used for the Laplacian scheme). Good convergence is also observed for $\psi = 0.5$ on Left regular meshes. However, as we observed in other verification tests, a poor convergence rate is observed on Left meshes for the average and maximum error (this latter result is not presented here) when $\psi = 0.333$, as can be seen in the Left Avg. $\psi = 0.333$ curve. These results on a Periodic BC confirm that the convergence issues originated at the boundaries.

4.4. Convergence Rate for the Gradient

Generally speaking, the trend seen up to now for the convergence rate of the basic variable ϕ in Poisson's equation is preserved for the gradient ($\nabla\phi$): using theoretically second order schemes, the convergence rate of 2.0 is observed for both gradient components [$\partial\phi/\partial x$, $\partial\phi/\partial y$] on orthogonal meshes, whereas that on non-orthogonal meshes is 1.0 for both components. The numerical gradient is calculated with the help of the OpenFOAM libraries fvc::grad on an Orthogonal regular hexahedral mesh and on a Left regular prism mesh. The gradient reconstruction is performed using the Gauss Linear, Least Squares, and Fourth schemes, but the results are presented only for the Gauss Linear scheme. The Fourth and Least Squares schemes do not show any significant improvement in terms of convergence behavior.

In Fig. 14, we can see the curves with labels Avg. $\partial\phi/\partial x$ Ortho, Avg. $\partial\phi/\partial y$ Ortho, Max. $\partial\phi/\partial x$ Ortho, and Max. $\partial\phi/\partial y$ Ortho for the two gradient components for the average and maximum errors on orthogonal meshes. For these, we observe a theoretical convergence rate of 2.0. For non-orthogonal meshes, the curves with labels Avg. $\partial\phi/\partial x$ nonOrtho and Avg. $\partial\phi/\partial y$ nonOrtho for the average error have an asymptotic convergence rate of 1. By contrast, the Max. $\partial\phi/\partial x$ nonOrtho and Max. $\partial\phi/\partial y$ nonOrtho curves have an asymptotic convergence rate of 0, which originates from the boundaries.

4.5. Implementing a Non-Orthogonal Treatment at the BC

A few factors guide us in identifying the source of convergence reduction on non-orthogonal meshes: theoretical convergence rates are observed in simulations using a Periodic BC and in simulations using Alternate meshes.

Convergence can be improved by using an external treatment, i.e. by not modifying OpenFOAM FVM libraries, and this approach is based on the use of the manufactured solution. We use the fact that the non-orthogonal term is implemented explicitly on the right-hand side of matrix equations (Eq. 17) as a source term, and so this correction can be implemented as a source term in the Poisson equation using the manufactured gradient. For the non-orthogonal term $\vec{k} \cdot (\nabla\phi)_f$, both the non-orthogonal correction vector $\vec{k}$ and the gradient $(\nabla\phi)_f$ need to be reconstructed. For the latter, the gradient produced by the MMS can be used directly. We only perform a reconstruction on boundary faces for these two fields, so this implementation does not interfere in any way with the non-orthogonal scheme for the Laplacian inside the domain. For the Surface Field vector $\vec{k}$, an over-relaxed correction approach at the BC is computed. By implementing $\vec{k} \cdot (\nabla\phi)_f$ on the laplacianFoam solver, an immediate improvement in the convergence rate is obtained.

The results using the Left regular prism, the Alternate regular prism, and the Delaunay prism meshes (see Figs. 2d, 2e, and 2f respectively) are presented. In Fig. 15, some curves for the average error Avg. are shown. In the Left Avg., Alternate Avg., and Delaunay Avg. curves, the surface gradient $(\nabla\phi)_f$ is replaced by the analytical gradient evaluated at the faces. For these curves, we obtain the expected theoretical convergence rate of 2.0.

4.6. Non-orthogonal Mesh with Neumann BC

In this section, we consider the Dirichlet BC combined with the Neumann BC (in the upper and in the lower boundaries), alternating opposite sides in a square domain $[0;\pi][0;\pi]$. We verify two ways of handling the Neumann BC: zeroGradient and fixedGradient, which are separate classes in OpenFOAM. For zeroGradient, the projection of the gradient along the normal to a boundary face is 0.

First, we test zeroGradient using the following manufactured solution. The projection along the normal on the upper and lower boundaries is nil:

$$\phi(x,y) = \left[\frac{\left(1-x^2\right)\left(2y^3 - 3\pi y^2 + \pi^3\right)}{\pi^3}\right] \tag{32}$$

Some results using a Left non-orthogonal mesh type (see Fig. 2d) are shown in Fig. 16. This representation shows the curves labeled Left Avg. and Left Max., which show a convergence in the order of 1.0 (using the original code). The plots Left Avg. Corr. and Left Max. Corr. were obtained by applying the non-orthogonal analytical correction to the Dirichlet BC on the left and right boundaries. An improvement in accuracy and convergence rate can clearly be seen. However, both curves display an asymtotic convergence in the order of 1.0. This makes it possible to conclude that the zeroGradient condition is the cause of a lower order of convergence.

Second, we applied a fixedGradient BC on the upper and lower boundaries using the following manufactured solution:

$$\phi(x,y) = \left[\frac{\left(1-x^2\right)\left(\frac{y^3}{3} - \frac{\pi^2 y}{2}\right)}{\pi^2}\right] \tag{33}$$

The results are shown in Fig. 17. Once again, the Left Avg. and Left Max. curves obtained with non-orthogonal grids of the Left type, show a convergence rate of the first order only (using the original code). The Left Avg. Corr. and Left Max. Corr. plots with non-orthogonal analytical correction at the Dirichlet BC show better accuracy and convergence; however, the asymptotic convergence rate is only 1.0.

5. Conclusions

A manufactured solution procedure has been applied to verify the OpenFOAM diffusion operator on different types of mesh. To achieve this, a manufactured Poisson's equation was solved using OpenFOAM schemes. Convergence analysis was performed by applying systematic grid refinement. Predicted second order accuracy was found when orthogonal grids and various alternatives for handling BCs (Dirichlet, Neumann, and periodic) are used.

On non-orthogonal grids, convergence deterioration was observed for several OpenFOAM schemes. It was found that even minor non-orthogonality significantly reduces the convergence order. It was also discovered that, in the domain's core, second-order accuracy is maintained, and that convergence reduction essentially derives from boundary treatment.

The verification procedure brought to light two factors which enable us to detect the origin of convergence reduction. First, it was found that when periodic BCs are used, second-order accuracy is obtained for all types of grid, orthogonal and non orthogonal. Second, a convergence loss was found for grids of the Alternate type, but only in the neighborhood of the boundary (In fact, the theoretical second order convergence is preserved in the domain). Thus, convergence reduction derives from non-orthogonal treatment at boundaries using the Dirichlet BC. A strategy for the treatment of non-orthogonality has been proposed by taking into account the analytical gradient (i.e. the gradient calculated from the manufactured solution).

Besides, on the one hand, when applying a combination of the Neumann BC and improved treatment of the Dirichlet BC, it was found that the BC zeroGradient and fixedGradient implementations also lead to a convergence degradation for non-orthogonal boundaries. On the other hand, it was found that the presence of skewness and non-uniformity seems to be well handled.

Another factor that has a significant impact on convergence reduction, and which does not originate from the boundaries, involves the choice of the constraint of the non-orthogonal correction scheme. In particular, the Gauss limited ψ scheme shows the best results for the value $\psi = 1.0$ for simulations using a Laplacian solver.

Alternatives for reconstructing numerical schemes on boundaries for second-order convergence will be presented in the second part of this work. The well-founded nature of the proposed options will be verified by performing convergence tests on problems described by the Laplace and Navier-Stokes equations.

This work has revealed the need for a verification of the numerical algorithms, as these are strongly influenced by the kind of mesh involved. It has been shown that verification is a complex task, even though it is only applied to the diffusion operator when a simple solver like laplacianFOAM is used, and which has been applied on prisms and regular hexahedrons with the Dirichlet, Neumann, and Periodic boundary conditions.

6. Figures

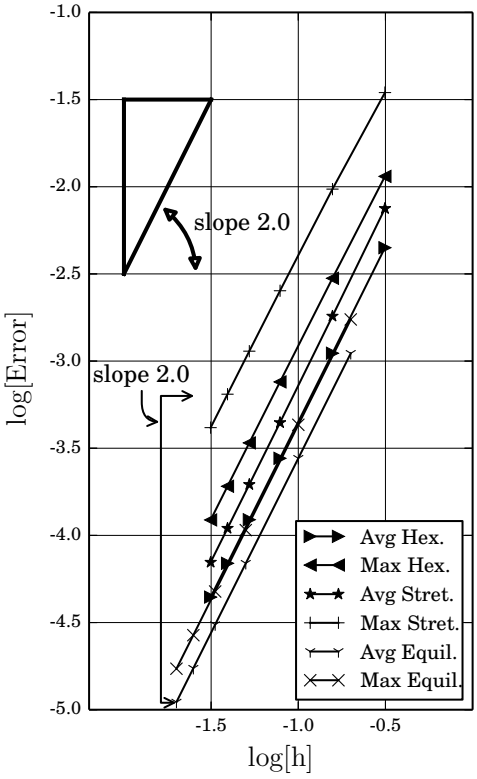


Figure 7: Curves on Orthogonal Meshes.

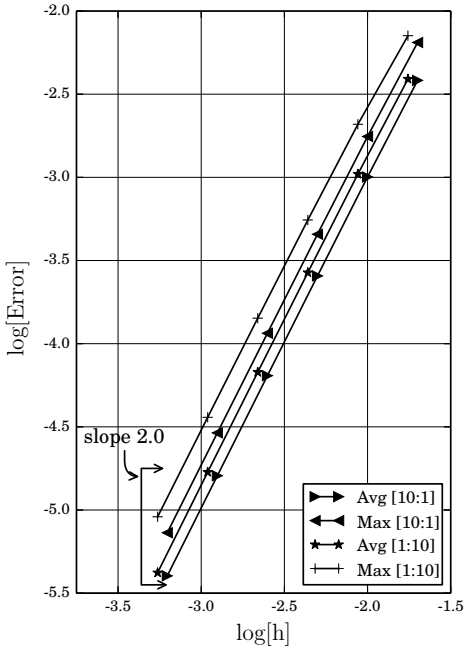


Figure 8: Curves on Orthogonal Elongated Meshes.

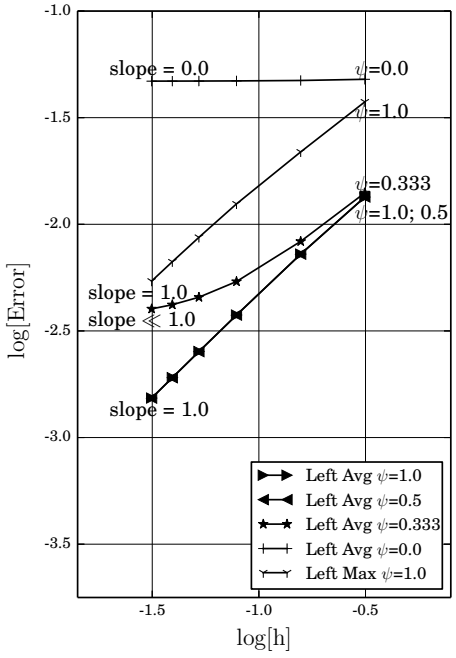


Figure 9: Curves on Non-Orthogonal Meshes with Varying Limited Parameter ψ.

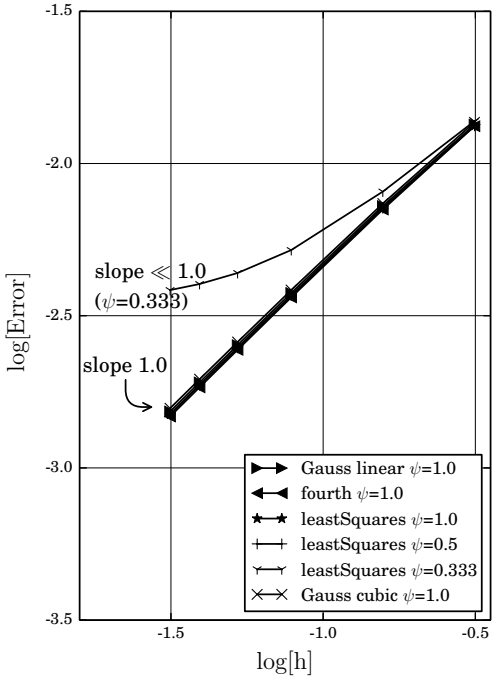


Figure 10: Curves on Non-Orthogonal Meshes Using a Set of Gradient Schemes for Non-Orthogonal Treatment.

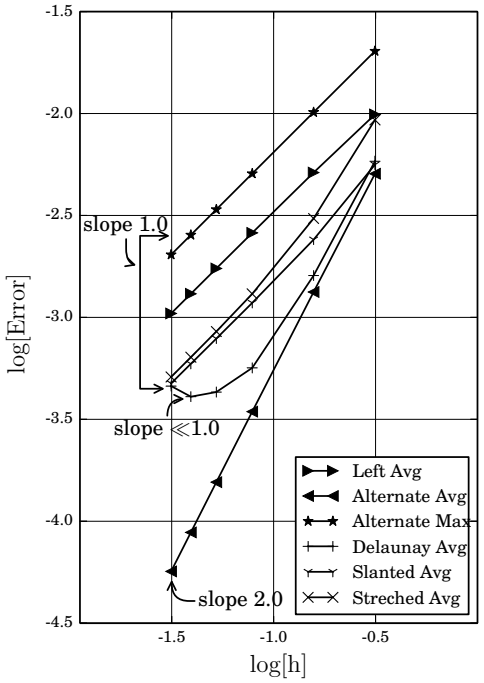


Figure 11: Curves on Non-Orthogonal Meshes.

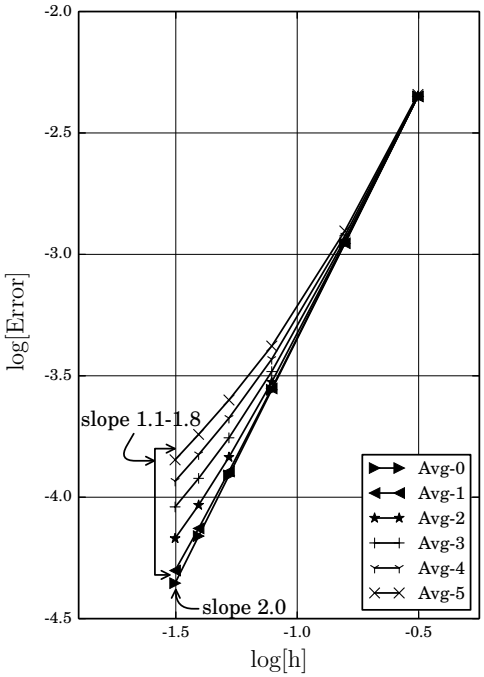


Figure 12: Incidence on Non-Orthogonal Meshes Inclining an Initial Square Domain.

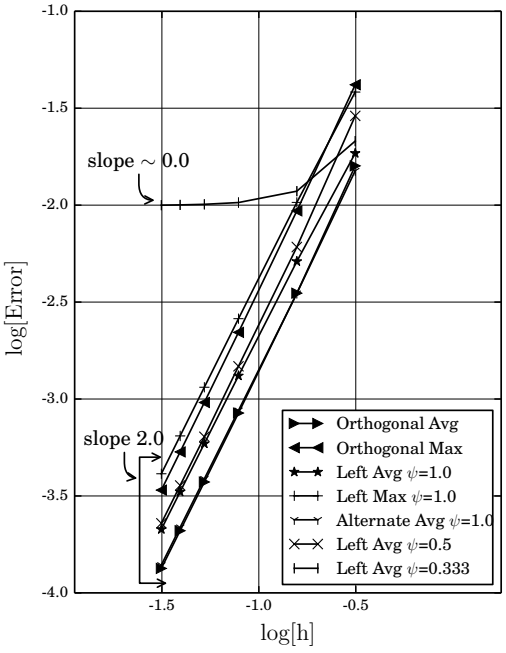


Figure 13: Curves Using Periodic (Cyclic) BCs.

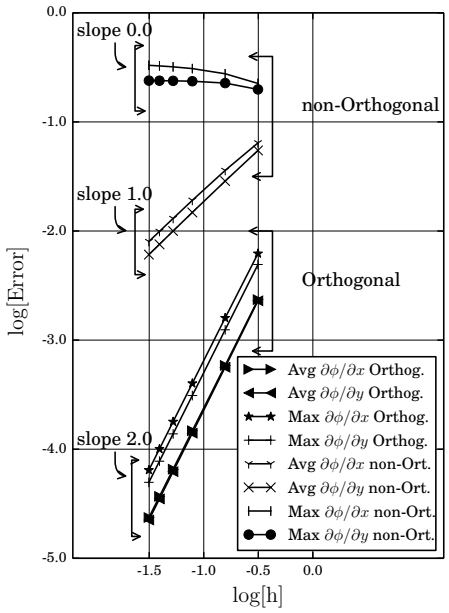


Figure 14: Curves on Non-Orthogonal Meshes for the Gradient of ϕ.

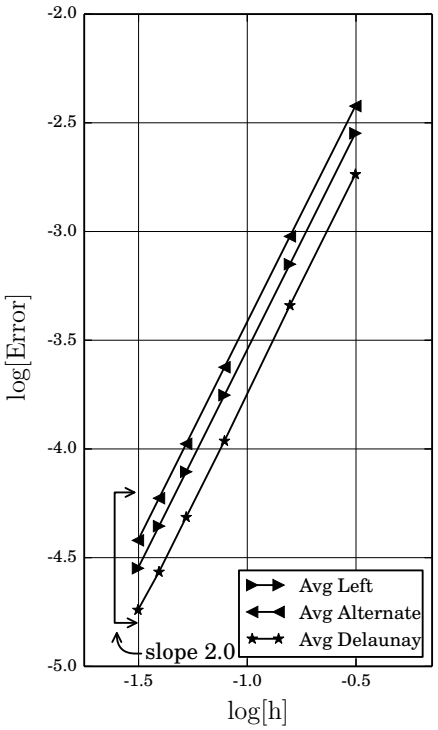


Figure 15: Curves Using an Analytical Gradient on the Non-Orthogonal Correction.

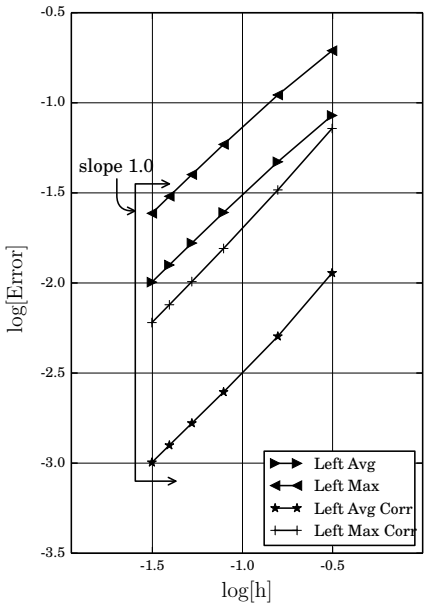


Figure 16: Curves Using Dirichlet and Adiabatic BCs.

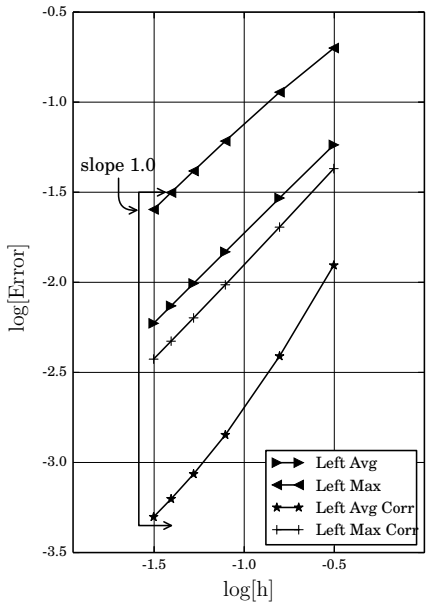


Figure 17: Curves Using Dirichlet and Neumann BCs.

7. References

[1] M. Schafer, Computational Engineering. Introduction to Numerical Methods, Springer, 2006.

[2] J. Blazek, Computational Fluid Dynamics: Principles and Applications, 2nd Edition, Elsevier Science, 2006.

[3] J. Ferziger, M. Peric, Computational Methods for Fluid Dynamics, 3nd Edition, Springer, 2002.

[4] H. Jasak, Error analysis and estimation for the finite volume method with applications to fluid flows, Ph.D. thesis, Imperial College (Jun. 1996).

[5] H. Jasak, A. Gosman, Automatic resolution control for the finite-volume method, part 2: a-posteriori error estimates, Numerical Heat Transfer, Part B: Fundamentals 38 (3) (2000) 237–256.

[6] H. Jasak, A. Gosman, Automatic resolution control for the finite-volume method, part 2: Adaptive mesh refinement and coarsening, Numerical Heat Transfer Part B Fundamentals 38 (3) (2000) 257–271.

[7] H. Jasak, A. Gosman, Rsidual error estimate for the finite volume method, Numerical Heat Transfer, Part B: Fundamentals 39 (1) (2001) 1–19.

[8] H. Jasak, A. Gosman, Element residual error estimate for the finite volume method, Computers and Fluids (2003) 223–248.

[9] H. Jasak, A. Gosman, Automatic resolution control for the finite-volume method, part 3: Turbulent flow applications, Numerical Heat Transfer Part B Fundamentals 38 (3) (2000) 273–290.

[10] F. Juretic, Error analysis in finite volume cfd., Ph.D. thesis, Department of Mechanical Engineering, Imperial College. (2004).

[11] F. Juretic, A. Gosman, Error analysis of the finite-volume method with respect to mesh type, Numerical Heat Transfer, Part B: Fundamentals: An International Journal of Computation and Methodology Fundamentals 57 (6) (2010) 414–439.

[12] F. Moukalled, L. Mangani, M. Darwish, The Finite Volume Method in Computaional Fluid Dynamic, Springer, 2015.

[13] P. Roache, Verification of codes and calculations, AIAA Journal 36, 5 (5) (1998) 696–702.

[14] F. Stern, R. Wilson, J. Shao, Quantitative v-v of cfd simulations and certification of cfd codes, International Journal for Numerical Methods in Fluids 50 (11) (2006) 1335–1355. doi:10.1002/fld.1090.
URL `http://dx.doi.org/10.1002/fld.1090`

[15] J. Abanto, D. Pelletier, A. Garon, J. Trepanier, M. Reggio, Verification of some commercial cfd codes on atypical cfd problems, 43rd AIAA Aerospace Sciences Meeting and Exhibit 1 (2005) 15693–15725.

[16] D. Tremblay, S. Etienne, D. Pelletier, Code verification and the method of manufactured solutions for fluid-structure interaction problems, 36th AIAA Fluid Dynamics Conference and Exhibit 2 (2006) 882–892.

[17] C. J. Roy, C. C. Nelson, T. M. Smith, C. C. Ober, Verification of euler/navierstokes codes using the method of manufactured solutions, International Journal for Numerical Methods in Fluids 44 (6) (2004) 599–620. doi:10.1002/fld.660.
URL `http://dx.doi.org/10.1002/fld.660`

[18] J. Iannelli, An exact non-linear navierstokes compressible-flow solution for cfd code verification, International Journal for Numerical Methods in Fluids 72 (2) (2013) 157–176. doi:10.1002/fld.3731.
URL `http://dx.doi.org/10.1002/fld.3731`

[19] L. Eca, M. Hoekstra, An introduction to cfd code verification including eddy-viscosity models, 2006.

[20] L. Eca, M. Hoekstra, Code verification of unsteady flow solvers with the method of the manufactured solutions, International offShore and Polar Engineering; ISOPE 2007 (2007) 2012–2019.

[21] L. Eca, M. Hoekstra, A. Hay, D. Pelletier, Verification of rans solvers with manufactured solutions', Engineering with Computers 23 (2007) 253–270.

[22] S. Veluri, C. Roy, Comprehensive code verification for an unstructured finite volume cfd code, 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition 4 - 7 January 2010.

[23] B. Diskin, J. Thomas, E. Nielsen, H. Nishikawa, J. White, Comparison of node-centered and cell-centered unstructured finite-volume discretizations: Viscous fluxes, AIAA JOURNAL 48 (7).

[24] B. Diskin, J. Thomas, Comparison of node-centered and cell-centered unstructured finite volume discretizations: Inviscid fluxes, AIAA JOURNAL 49 (4).

[25] B. Diskin, J. Thomas, Effects of mesh regularity on accuracy of finite-volume schemes, 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition 09 - 12 January 2012, Nashville, Tennessee.

[26] S. Muzaferija, Adaptive finite volume method for flow predictions using unstructured meshes and multigrid approach., Ph.D. thesis, University of London (1994).
[27] P. Knupp, K. Salari, Verification of Computer Codes in Computational Science and Engineering, Chapman and Hall/CRC, 2002.
[28] User-Guide, Openfoam. the open source cfd toolbox. user guide. version 2.2. september 2013. openfoam foundation (2013).
[29] Programmer-Guide, Openfoam. the open source cfd toolbox. programmers guide. version 2.2. september 2013. openfoam-foundation (2013).

ANNEXE C ARTICLE 2. A Case-Study in Open-Source CFD Code Verification. Part II : Boundary Condition Non-Orthogonal Correction.

Dans l'article II, l'objectif est de corriger le problème de la dégradation de la convergence sur des maillages non orthogonaux et d'effectuer un processus de vérification des schémas implémentés.

Nom de l'article en anglais :

"A Case Study in Open-Source CFD Code Verification. Part II : Boundary Condition Non-Orthogonal Correction".

Auteurs :

Harold Noriega ; François Guibault ; Marcelo Reggio ; Robert Magnan.

L'article a été soumis au journal :

Mathematics And Computers in Simulations. 12-10-2016.

A Case-Study in Open-Source CFD Code Verification. Part II: Boundary Condition Non-Orthogonal Correction.

H. Noriega[a,*], F.Guibault[a], M.Reggio[a], R.Magnan[b]

[a] *2900, boul. Édouard-Montpetit, Campus de l'Université de Montréal, 2500, chemin de Polytechnique, Montréal (Québec) H3T 1J4*

[b] *Institut de recherche d'Hydro-Québec, 1800 Boulevard Lionel-Boulet, Varennes, QC J3X 1S1, Québec, Canada.*

Abstract

This investigation constitutes the follow-up to a previous work in which we applied the manufactured solution procedure to verify the OpenFOAM diffusion operator on different types of meshes. Theoretical convergence orders were observed for Poisson's equation solver on orthogonal hexahedral grids using several boundary conditions. We noticed that on non-orthogonal grids even slight mesh distortions reduce the theoretical second order convergence rate to first order. Our investigation showed that this loss in convergence order takes place when Dirichlet and/or Neumman boundary conditions are used on non-orthogonal meshes. In this paper, we introduce ways to achieve theoretical second order convergence accuracy, mainly based on applying non-orthogonal correction to Dirichlet and/or Neumann boundary condition schemes. A peculiarity was discovered for the Least-Squares gradient scheme that slightly affects second-order convergence, and which we attribute to the non-orthogonal correction at boundaries in the Least-Squares vector reconstruction library. The verification procedure is performed using the method of manufactured solutions for the Poisson equation and an analytical solution, the SIMPLE solver, for the Navier-Stokes equations. The average and maximum error norms were used to calculate the convergence rate. Comparative results are presented.

Keywords: Verification, Manufactured Solution, OpenFOAM, Poisson Equation, Navier-Stokes Equations.

1. Introduction

OpenFOAM®, a trademark of The OpenFOAM Foundation, is free, open source Computational Fluid Dynamics (CFD) software that is widely used in both industry and academia. The number of flow simulations based on this tool has grown considerably in recent years, particularly in academia. The theoretical foundation for OpenFOAM is the finite volume method (FVM), which is a well-known approach for solving a variety

*Corresponding author

Email addresses: harold.noriega@polymtl.ca (H. Noriega), francois.guibault@polymtl.ca (F.Guibault), marcelo.reggio@polymtl.ca (M.Reggio), magnan.robert@ireq.ca (R.Magnan)

of partial differential equations. Theoretical information regarding the FVM is widely available, and can be found in [1], [2], and [3], among many others. Schafer [1] presents an introduction to several approaches for approximating surface and volume integrals, which make it possible to obtain a number of second-order schemes for orthogonal and non-orthogonal meshes. Blazek [2] compares cell-centered and median-dual schemes on unstructured meshes in terms of accuracy, flexibility, and computational time. A detailed description of OpenFOAM based on a finite volume cell-centered approach can be found in [4, 5, 6, 7, 8, 9, 10, 11, 12].

OpenFOAM convection-diffusion schemes include a non-orthogonal treatment, which, for the diffusion operator case, corresponds to a deferred correction approach. Examples of applications of this kind of methodology are explained in [3]. Deferred correction approaches are applied in the FVM for higher order schemes, among many other uses. Khosla and Rubin [13] first suggested an explicitly computed higher order approximation which is then combined with an implicit lower order approximation to calculate FVM fluxes (see [3]). As cited in [3], Muzaferija [14] suggested a second order deferred correction scheme to deal with non-orthogonality. Jasak [4] uses three different ways to calculate a correction expression for a non-orthogonal treatment (see [5]).

In [15], we developed a verification procedure based on the method of manufactured solutions (MMS). As shown in that work, the non-orthogonal treatment at the boundaries modifies the convergence order in solving the Poisson equation, and the theoretical second order is not reached. As in the previous study, we follow the verification procedure proposed by Knupp and Salari [16]. In [16], the MMS is used to obtain exact solutions for code verification and for studying the order and accuracy of numerical solutions of Burgers' and Navier-Stokes equations. A background discussion, along with definitions and descriptions of some terms related to confidence building in CFD, is proposed by Roache [17] and extended by Stern et al. [18]. The various aspects discussed in these papers include the distinction between code verification and code validation, verification of code and verification of calculations, grid convergence vs. iterative convergence, and numerical error vs. conceptual modeling error.

In this context, Abanto et al. [19] presented a grid convergence study on some atypical CFD cases using a number of commercial CFD packages. Their verification test cases are distinctive in that exact solutions are known. These include an incompressible laminar Poiseuille flow, an incompressible recirculating flow, a manufactured incompressible laminar boundary layer flow, and an incompressible annular flow. Different convergence rates were determined using structured and unstructured meshes. Stern et al. [18] describe a set of verification, validation, and certification methodologies and procedures for numerical simulations. Examples of the application of the quantitative certification of Reynolds-averaged Navier-Stokes (RANS) codes are presented in that work for ship hydrodynamics.

Roy et al. [20] used MMS to verify the order of accuracy of two finite-volume Euler and NavierStokes codes. These exact solutions were used to accurately evaluate the discretization error in the numerical solutions. Through global discretization error analyses, the spatial accuracy was observed to be second order for a node-centered approach using

unstructured meshes. More recently, Iannelli [21] obtained exact and CFD solutions of the NavierStokes equations. They determined the convergence rates and orders of accuracy of these solutions and illustrated the usefulness of the exact solution for verification purposes. Eca and Hoekstra [22, 23] and Eca et al. [24] published a series of papers on Manufactured Solutions for the 2D incompressible Navier-Stokes equations with turbulent models.

Along these lines, the objective in this work is to show ways to achieve a theoretical second order convergence rate on non-orthogonal meshes using the Dirichlet and Neumann boundary conditions (BCs). One approach is based on a local boundary gradient reconstruction, i.e. a cell centroid gradient reconstruction in cells that have face boundaries. A second alternative involves inserting a source term directly into the solver. This compensates for the non-orthogonal treatment at a given boundary. Also, we show that there is a reduction in the convergence rate resulting from non-orthogonal treatment in the least squares vector reconstruction libraries. Finally, we extend the verification procedure to the SIMPLE solver for the steady Navier-Stokes equation.

The paper is organized as follows: we first present the formulation of the mathematical models, followed by a brief summary of the verification methodology. The numerical results are presented next for the Dirichlet and Neumann BCs, obtained before and after implementing the local boundary gradient reconstruction on non-orthogonal meshes, and after implementing the non-orthogonal treatment as a source term. Similarly, we present numerical results obtained before and after implementing an over-relaxed correction in the Least-Squares vector reconstruction library. Results for the Poisson and steady Navier-Stokes equations are also shown. The final section provides concluding remarks and perspectives.

2. Mathematical Models and Discretisation

We focus on two equations here. The first, Poisson's equation, is used to test the non-orthogonality corrections on Dirichlet or Neumann BC schemes, the macroscopic governing equation of which can be given as:

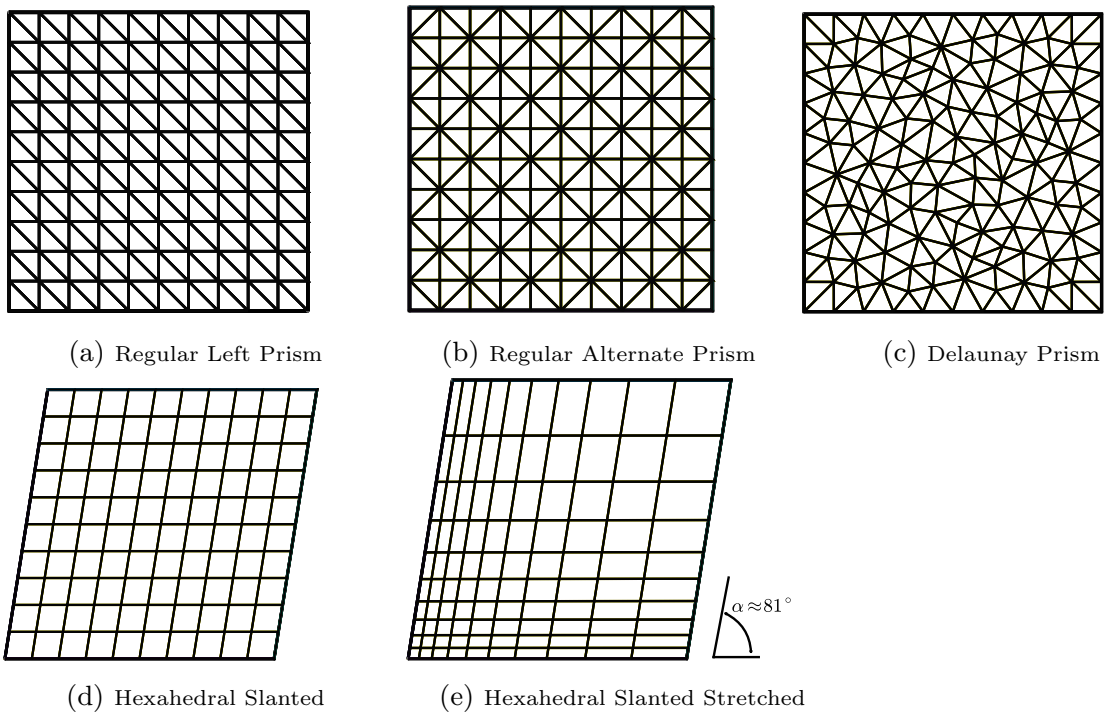
$$-\nabla \cdot (\nu \nabla \phi) = q \tag{1}$$

in terms of a scalar field ϕ, the scalar diffusivity coefficient ν, and the scalar source term q.

The new correction approaches for the SIMPLE solver involve the second equation, which we use to tackle the steady incompressible Navier-Stokes equation, the governing equation for which can be given as:

$$\begin{aligned} (\vec{v} \cdot \nabla)\vec{v} - \nabla \cdot (\nu \nabla \vec{v}) &= -\nabla \left(\frac{p}{\rho_0} \right) + \vec{q} \\ \nabla \cdot \vec{v} &= 0 \end{aligned} \tag{2}$$

where $\vec{v}$ is the flow velocity, p is the pressure, ρ_0 is the uniform density and $\vec{q}$ is the vector source term.



(a) Regular Left Prism (b) Regular Alternate Prism (c) Delaunay Prism

(d) Hexahedral Slanted (e) Hexahedral Slanted Stretched

Figure 1: Non-Orthogonal Prism Meshes in 2D Representation

The discretization of the convection-diffusion operators, based on the cell-centered approach of the finite volume method (FVM), was introduced in the first part of this work [15] (see also [4, 10, 12]). Further details on the discretization issue can be found in the first part of this work, while some concepts specific to this second part of the work are presented below.

2.1. The Mesh

To verify the correctness of the BC corrections implemented, we use non-orthogonal meshes only: We consider three types of prism mesh, as shown in Figs. 1a, 1b and 1c; a hexahedral slanted mesh, depicted in Fig. 1d, and a hexahedral slanted stretched mesh, depicted in Fig. 1e.

2.2. Non-Orthogonal Treatment at Boundaries

Non-orthogonal treatment at boundaries, which we implement in this work, requires a face boundary vector reconstruction, $\vec{k}$. For this, we use an over-relaxed correction vector approach (see [4]), according to which, the correction vector is orthogonal to the surface area vector $\vec{S}$, as shown in Fig. 2. In this graph, $\vec{d} = P\vec{N}$ is the vector linking the cell centers P with N and $\vec{\Delta} = \frac{|S|}{\cos(\alpha_N)} \frac{\vec{d}}{|d|}$. The correction vector is defined from the relation $\vec{S} = \vec{d} + \vec{k}$.

2.3. Truncation Errors at Boundaries

In this section, we show that a correction needs to be added to the original formulas proposed in OpenFOAM for handling the Dirichlet and Neumann BCs when

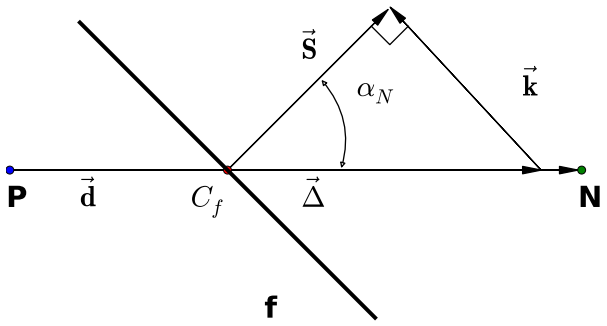


Figure 2: Non-Orthogonality Treatment: "Over-relaxed" Approach

non-orthogonal grids are used. The results in this section constitute our theoretical contribution, and they are confirmed later in this work by the results of a convergence study.

For the most common CFD problems, BCs are given by prescribing the value of ϕ at the boundary in the case of the Dirichlet BC and/or by prescribing the gradient value in the case of the Neumann BC.

Convective fluxes require a value of ϕ at a face boundary, which is known in the case of the Dirichlet BC. For the Neumann BC, however, the implementation in OpenFOAM extrapolates the quantity from the cell with the boundary face using the following expression (see Fig. 3 and Ref. [4]):

$$(\phi_{C_f})_h = \phi_P + \vec{d_n} \cdot (\nabla\phi)_{C_f} \tag{3}$$

where the index $_h$ indicates the numerical value.

The truncation error can be estimated using the following Taylor expansion (as backward approximation):

$$\phi_{P_A} = \phi_{C_f} - \vec{d} \cdot (\nabla\phi)_{C_f} + \frac{1}{2}\vec{d^2} : (\nabla\nabla\phi)_{C_f} \tag{4}$$

where ϕ_{P_A} is the exact solution (the analytical value) of the function ϕ at P and : represents the product of two tensor expressions.

From Eqs.(3 and 4) the following truncation error is obtained (see Fig. 3):

$$\begin{aligned} e_{extraPol} &= \phi_{C_f} - (\phi_{C_f})_h \\ &= -\vec{d_n} \cdot (\nabla\phi)_{C_f} + \vec{d} \cdot (\nabla\phi)_{C_f} - \frac{1}{2}\vec{d^2} : (\nabla\nabla\phi)_{C_f} + e_{\phi_P} \\ &= \vec{k} \cdot (\nabla\phi)_{C_f} - \frac{1}{2}\vec{d^2} : (\nabla\nabla\phi)_{C_f} + e_{\phi_P} \end{aligned} \tag{5}$$

where $e_{\phi_P} = (\phi_{P_A} - \phi_P)$ is the approximation error of function ϕ at P.

Decomposing the vector $\vec{d}$ into $\vec{d} = \vec{d_n} + \vec{k}$, using the spatial Taylor approximation for $(\nabla\phi)_{C_f} = (\nabla\phi)_{P_A} + \vec{d} \cdot (\nabla\nabla\phi)_{P_A}$ and $(\nabla\nabla\phi)_{C_f} = (\nabla\nabla\phi)_{P_A}$, and inserting that values

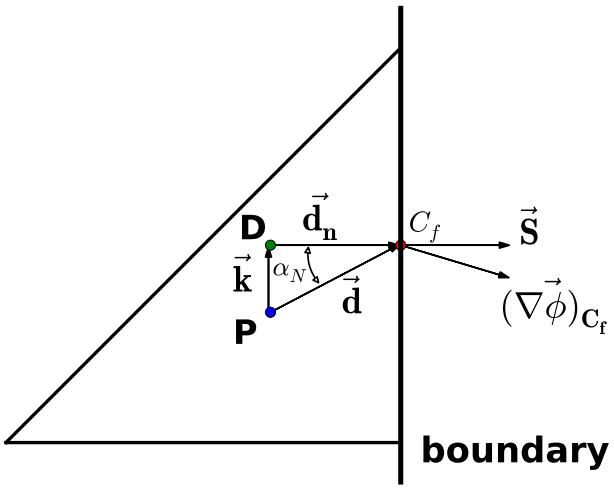


Figure 3: Non-Orthogonality Treatment in Patch

in Eq. 5 leads to the following:

$$\begin{aligned} e_{extraPol} &= \vec{k}\cdot(\nabla\phi)_{P_A} + \vec{k}\vec{d}:(\nabla\nabla\phi)_{P_A} - \frac{1}{2}\vec{d}^2:(\nabla\nabla\phi)_{P_A} + e_{\phi_P} \\ &= \vec{k}\cdot(\nabla\phi)_{P_A} + \frac{1}{2}\vec{k}\vec{d}:(\nabla\nabla\phi)_{P_A} - \frac{1}{2}\vec{d_n}\vec{d}:(\nabla\nabla\phi)_{P_A} + e_{\phi_P} \end{aligned} \tag{6}$$

Eq. 6 can be expressed in terms of numerical values at point P:

$$\begin{aligned} e_{extraPol} &= \underbrace{\vec{k}\cdot(\nabla\phi)_P + \frac{1}{2}\vec{k}\vec{d}:(\nabla\nabla\phi)_P - \frac{1}{2}\vec{d_n}\vec{d}:(\nabla\nabla\phi)_P}_{spatial\ error\ contribution} \\ &+ \underbrace{\left(e_{\phi_P} + \vec{k}\cdot e_{(\nabla\phi)_P} + \frac{1}{2}\vec{k}\vec{d}:e_{(\nabla\nabla\phi)_P} - \frac{1}{2}\vec{d_n}\vec{d}:e_{(\nabla\nabla\phi)_P}\right)}_{approximation\ errors\ at\ P} \end{aligned} \tag{7}$$

In Eq. 7, we have the contribution of the spatial error, as well as that of the error generated by the accuracy of the numerical values at location P. To simplify the analysis, we suppose that, in Eq. 7, the following relations hold: $e_{\phi_P} \sim \mathcal{O}(|d|^2)$, $e_{(\nabla\phi)_P} \sim \mathcal{O}(|d|)$, and $e_{(\nabla\nabla\phi)_P} \sim \mathcal{O}(|1|)$. This assumption is necessary in order to avoid analyzing the approximation error. In other words, this means that the convergence loss may be caused by the increase in the approximation error, which can, for example, be produced by a poor interpolation (or extrapolation) of a quantity on the internal faces. So, we have the following:

$$e_{extraPol} = \underbrace{\vec{k}\cdot(\nabla\phi)_P}_{first\ order\ accurate} + \frac{1}{2}\vec{k}\vec{d}:(\nabla\nabla\phi)_P - \frac{1}{2}\vec{d_n}\vec{d}:(\nabla\nabla\phi)_P \tag{8}$$

On non-orthogonal meshes, the vector $\vec{k}$ is not zero, the boundary error in Eq. 8 has a lead first order term, which causes a loss of accuracy when Neumann conditions are

used. To obtain a second-order rate in Eq. 3, we can add the correction term $\vec{k} \cdot (\nabla\phi)_P$, as follows:

$$(\phi_{C_f})_h = \phi_P + \vec{d_n} \cdot (\nabla\phi)_{C_f} + \vec{k} \cdot (\nabla\phi)_P \tag{9}$$

OpenFOAM handles the Neumann BC in two different ways: "zeroGradient" and "fixedGradient", both based on Eq. 3. In the zeroGradient handling, the projection of the gradient along the normal is zero $\left(\vec{n} \cdot (\nabla\phi)_{C_f}) = 0\right)$ and the extrapolation is performed directly from the neighboring cell, i.e. $(\phi_{C_f})_h = \phi_P$. Note here that the correction is necessary in the case of a zeroGradient on a non-orthogonal grid. For this type of BC, when the correction is taken into account, we have the following (see Fig. 3):

$$(\phi_{C_f})_h = \phi_P + \vec{k} \cdot (\nabla\phi)_P = \phi_D \tag{10}$$

For the "fixedGradient" handling, the corresponding equation is:

$$(\phi_{C_f})_h = \phi_P + \vec{k} \cdot (\nabla\phi)_P + \vec{d_n} \cdot (\nabla\phi)_{C_f} = \phi_D + \vec{d_n} \cdot (\nabla\phi)_{C_f} \tag{11}$$

where $(\nabla\phi)_{C_f}$ is the user specified gradient.

Diffusive fluxes require the normal gradient projection of ϕ at a boundary face, which is a known quantity for the Neumann BC. For the Dirichlet BC, the following expression is implemented in OpenFOAM to calculate the gradient projection at the face center C_f (see Fig. 3 and Ref.[4]):

$$\left(\vec{S} \cdot (\nabla\phi)_{C_f}\right)_h = |S| \frac{\phi_{C_f} - \phi_P}{|d_n|} \tag{12}$$

where the index $_h$ indicates the numerical value.

The formula given by Eq. 12 may also be re-written as follows:

$$\left(\vec{S} \cdot (\nabla\phi)_{C_f}\right)_h = |S| \frac{\phi_{C_f} - \phi_{P_A}}{|d_n|} + |S| \frac{\phi_{P_A} - \phi_P}{|d_n|} \tag{13}$$

where ϕ_{P_A} indicates the exact (analytical) value computed at P.

Eq. 12 is clearly of order 1 for orthogonal meshes. However, this scheme has a constant convergence order when non-orthogonal meshes are used. To show this, we use Taylor's expansion:

$$\phi_{C_f} = \phi_{P_A} + \vec{d} \cdot (\nabla\phi)_{P_A} + \frac{1}{2}\vec{d^2} : (\nabla\nabla\phi)_{P_A} \tag{14}$$

where $(\nabla\phi)_{P_A}$ and $(\nabla\nabla\phi)_{P_A}$ are the more highly derived analytical values evaluated at P.

Replacing Eq. 14 in Eq. 13, we obtain:

$$\left(\vec{S} \cdot (\nabla\phi)_{C_f}\right)_h = |S| \frac{\left(\vec{d} \cdot (\nabla\phi)_{P_A} + \frac{1}{2}\vec{d^2} : (\nabla\nabla\phi)_{P_A}\right)}{|d_n|} + |S| \frac{\phi_{P_A} - \phi_P}{|d_n|} \tag{15}$$

Using the Taylor approximation for gradient $(\nabla\phi)_{C_f} = (\nabla\phi)_{P_A} + \vec{d}\cdot(\nabla\nabla\phi)_{P_A}$ and inserting that value in Eq. 15 yields:

$$\begin{aligned}\left(\vec{S}\cdot(\nabla\phi)_{C_f}\right)_h &= |S|\frac{\left(\vec{d_n}\cdot(\nabla\phi)_{P_A} + \vec{k}\cdot(\nabla\phi)_{P_A} + \frac{1}{2}\vec{d^2} : (\nabla\nabla\phi)_{P_A}\right)}{|d_n|} + \frac{|S|}{|d_n|}e_{\phi_P} \\ &= \vec{S}\cdot(\nabla\phi)_{C_f} + \frac{|S|}{|d_n|}e_{\phi_P} \\ &+ |S|\frac{\left(\vec{k}\cdot(\nabla\phi)_{P_A} + \frac{1}{2}\vec{k}\vec{d} : (\nabla\nabla\phi)_{P_A} - \frac{1}{2}\vec{d_n}\vec{d} : (\nabla\nabla\phi)_{P_A}\right)}{|d_n|}\end{aligned} \tag{16}$$

If we define the error of the gradient projection e_{snGrad} as:

$$e_{snGrad} = \vec{S}\cdot(\nabla\phi)_{C_f} - \left(\vec{S}\cdot(\nabla\phi)_{C_f}\right)_h \tag{17}$$

then we can express the error of the gradient projection as:

$$e_{snGrad} = -|S|\frac{\left(\vec{k}\cdot(\nabla\phi)_{P_A} + \frac{1}{2}\vec{k}\vec{d} : (\nabla\nabla\phi)_{P_A} - \frac{1}{2}\vec{d_n}\vec{d} : (\nabla\nabla\phi)_{P_A}\right)}{|d_n|} - \frac{|S|}{|d_n|}e_{\phi_P} \tag{18}$$

Eq. 18 can be expressed in terms of numerical values (using the index P):

$$\begin{aligned}e_{snGrad} &= -|S|\underbrace{\frac{\left(\vec{k}\cdot(\nabla\phi)_P + \frac{1}{2}\vec{k}\vec{d} : (\nabla\nabla\phi)_P - \frac{1}{2}\vec{d_n}\vec{d} : (\nabla\nabla\phi)_P\right)}{|d_n|}}_{spatial\ error\ contribution} \\ &- |S|\underbrace{\frac{\left(e_{\phi_P} + \vec{k}\cdot e_{(\nabla\phi)_P} + \frac{1}{2}\vec{k}\vec{d} : e_{(\nabla\nabla\phi)_P} - \frac{1}{2}\vec{d_n}\vec{d} : e_{(\nabla\nabla\phi)_P}\right)}{|d_n|}}_{approximation\ errors\ at\ P}\end{aligned} \tag{19}$$

Note that it is important to include the approximation error in Eqs. 7 and 19: for example, the numerical gradient $(\nabla\phi)_P$ at the centroid of the cell neighboring the boundary is calculated using the values of ϕ_P. The numerical value of ϕ_P could be affected by an external source. In the 4.2.1 section, we will show that there is a small reduction in the convergence order when the least squares gradient scheme is used. This can be attributed to the non-orthogonal correction at boundaries in the least squares vector reconstruction library which affects the numerical value of ϕ_P.

To simplify the analysis (as we did for the Neumann BC), we suppose that $e_{\phi_P} \sim$

$\mathcal{O}(|d|^2)$, $e_{(\nabla\phi)_P} \sim \mathcal{O}(|d|)$, and $e_{(\nabla\nabla\phi)_P} \sim \mathcal{O}(|1|)$. Then,

$$\begin{aligned} e_{snGrad} &= -|S| \frac{\left(\vec{k}\cdot(\nabla\phi)_P + \frac{1}{2}\vec{k}\vec{d}:(\nabla\nabla\phi)_P - \frac{1}{2}\vec{d_n}\vec{d}:(\nabla\nabla\phi)_P\right)}{|d_n|} \\ &= -|S| \left(\underbrace{\tan(\alpha_N)(\vec{\hat{k}}\cdot(\nabla\phi)_P)}_{constant\ accurate} \right) \\ &+ |S| \left(\frac{1}{2}\tan(\alpha_N)\vec{\hat{k}}\vec{d}:(\nabla\nabla\phi)_P + \frac{1}{2}\vec{\hat{d}_n}\vec{d}:(\nabla\nabla\phi)_P \right) \end{aligned} \tag{20}$$

where $\vec{\hat{k}}$ and $\vec{\hat{d}_n}$ are the unitary vectors corresponding to $\vec{k}$ and $\vec{d_n}$ respectively.

For non-orthogonal meshes when $\vec{k}$ is a non-zero vector, the boundary error in Eq. 20 has a term with a constant convergence order, which causes a loss of accuracy on the Laplacian scheme for the Dirichlet BC. To obtain a first order rate in Eq. 20, we add the correction $\vec{k}\cdot(\nabla\phi)_P$:

$$\left(\vec{S}\cdot(\nabla\phi)_{C_f}\right)_h = \frac{|S|}{|d_n|}\left(\phi_{C_f} - \phi_P - \vec{k}\cdot(\nabla\phi)_P\right) \tag{21}$$

Note that, in [10], Juretic analyzed the truncation error directly for the 21 scheme. In his study, the approximation error was not included.

Finally, Eq. 21 can be written as:

$$e_{snGrad} = -|S| \left(\frac{1}{2}\tan(\alpha_N)\vec{\hat{k}}\vec{d}:(\nabla\nabla\phi)_P + \frac{1}{2}\vec{\hat{d}_n}\vec{d}:(\nabla\nabla\phi)_P \right) \tag{22}$$

which is first order accurate. This first order accurate approximation of the gradient projection at the boundary face is equivalent to a first order accurate approximation of the gradient at the centroid of the face:

$$(\nabla\phi)_{C_f} = \left((\nabla\phi)_{C_f}\right)_h + \mathcal{O}(|d|) \tag{23}$$

Now note that at least a first-order error for the gradient projection at the face boundary is necessary for there to be a second-order contribution at the centroid of the neighboring cell for ϕ_P. To see this, we use Eq. 23 and, assuming that

$$(\nabla\phi)_P = (\nabla\phi)_{C_f} + \left(\vec{d}\cdot(\nabla\nabla\phi)_{C_f}\right) + \mathcal{O}(|d|^2)$$

Then, using Eq. 23, we re-write Eq. 21 (which is first order accurate) as follows:

$$\begin{aligned} \phi_P &= \phi_{C_f} - |d_n|\left(\vec{n}\cdot(\nabla\phi)_{C_f}\right)_h - \left(\vec{k}\cdot(\nabla\phi)_P\right) - \frac{|d_n|}{|S|}\mathcal{O}(|d|) \\ &= \phi_{C_f} - \left(\vec{d_n}\cdot(\nabla\phi)_{C_f}\right)_h - \left(\vec{k}\cdot(\nabla\phi)_{C_f}\right)_h - \left(\vec{k}\vec{d}:(\nabla\nabla\phi)_{C_f}\right)_h + \frac{|d_n|}{|S|}\mathcal{O}(|d|) \\ &= \phi_{C_f} - \left(\vec{d}\cdot(\nabla\phi)_{C_f}\right)_h + \mathcal{O}(|d|^2) \end{aligned} \tag{24}$$

where $\vec{n}$ is the normal vector of the face f. This approximation of ϕ_P is clearly a second order approximation from ϕ_{C_f} at the boundary.

Note that Eq. 21 can be rewritten as (see Fig. 3):

$$\left(\vec{S}\cdot(\nabla\phi)_{C_f}\right)_h = \frac{|S|}{|d_n|}\left(\phi_{C_f}-\phi_D\right) \tag{25}$$

which is the gradient at C_f calculated as a backward difference.

We have shown theoretically that the accuracy on non-orthogonal meshes can be improved using the correction $(\vec{k}\cdot(\nabla\phi)_P)$ for both the Dirichlet and the Neumann BC. The practical implementation can be carried out in different ways. One way is to place a non-orthogonal compensation on the right-hand side of the system matrix equation, which can be implemented via a new BC, as explained below.

2.4. Implementation of a Non-Orthogonal Correction at a BC

The non-orthogonal contribution $(\vec{k}\cdot(\nabla\phi)_P)$ is calculated explicitly, as a source term, on the right-hand side of the matrix equation formed. Based on this, we can expect to implement this correction as a source term via a new BC or as a source term inside the solver (for the Dirichlet BC only). A gradient reconstruction $(\nabla\phi)_P$ can be implemented locally, i.e. a reconstruction only in cells that have a face on a boundary (see Fig. 4) or via the use of OpenFOAM gradient libraries, which involves a gradient reconstruction over the whole domain.

2.4.1. Local Gradient Reconstruction.

Gauss or Least-Squares approaches can be used to reconstruct a local gradient $(\nabla\phi)_P$. In this work, "local" means that we reconstruct the gradient only in cells neighboring the boundary (see Fig. 4). Note that this local gradient reconstruction must undergo both a non-orthogonal and a skewness treatment to ensure an efficient and accurate gradient reconstruction.

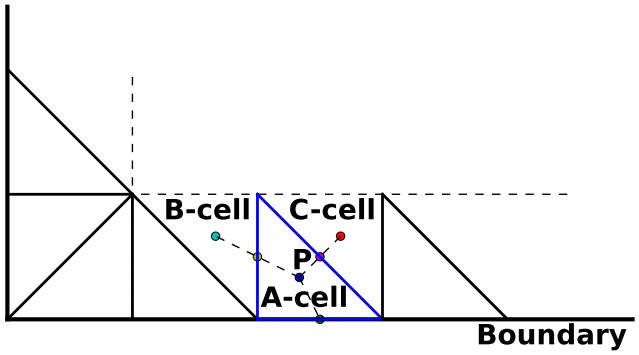


Figure 4: A cell with a face on boundary.

2.4.2. Full Gradient Reconstruction Using OpenFOAM Libraries.

The BC implementation using OpenFOAM gradient libraries performs a gradient reconstruction on the whole domain. The gradient projection is calculated as follows:

$$\vec{k} \cdot \text{fvc::interplate}(\text{fvc::grad}(\phi)) \tag{26}$$

where "$\cdot$" is a dot product, $\vec{k}$ is the non-orthogonal correction vector, and "fvc::interpolate" and "fvc::grad" are OpenFOAM interpolation and gradient reconstruction libraries, respectively. A gradient reconstruction is built automatically by "fvc::gradient" libraries in the "full domain", but we extract only the gradient values of cells neighboring the boundary to implement the non-orthogonal treatment. The gradient fvc::interpolate(fvc::grad(ϕ)) is a surface gradient which is projected over the correction vector $\vec{k}$ in all faces inside the domain.

2.4.3. Non-Orthogonal Correction as a Source Term.

When using the Dirichlet BC, the non-orthogonal correction can be easily implemented by applying OpenFOAM libraries to the solver matrix equation:

$$\text{solve}\left(A == q + \text{fvc::div}\left(\nu|S|(\vec{k} \cdot \text{fvc::interpolate}(\text{fvc::grad}(\phi)))\right)\right)$$

where A is the matrix of coefficients of the linear system and $\nu|S|$ the product of the diffusion coefficient and the module of the surface normal vector. This approach requires modification of the solver, but it has one advantage, which is that we have access to all the data structure components representing one specific field.

3. Verification Methodology

In first part of this work [15], we introduced the method of manufactured solutions, along with the error norms used in the verification process. In this section, we briefly recall the error norms, and the analytical and the manufactured solutions used in this work.

3.1. Error Estimation and mesh refinement

For the average and maximum errors, we use the same norms as in the first part of this work. These are, respectively:

$$||e||_{L_1(\Omega)} = \int_\Omega |e| \, d\Omega \approx \frac{1}{|\Omega|} \sum_{i=1}^{N} \Omega_i \, |e_i| \tag{27}$$

$$||e||_\infty = \sup_{x \in \Omega} |e(x)| \approx \max_i |e_i| \tag{28}$$

where e_i is the exact error, and is the difference between the analytical and numerical solutions; Ω indicates the domain definition; Ω_i is the control volume of cell i; and N indicates the number of cells. A systematic mesh refinement is performed which divides a parallelogram domain, with N_k being the number of nodes along each boundary, taking the values $N_k = [10, 20, 40, 60, 80, 100]$.

3.2. Manufactured and Analytical Solutions

As in [15], we use the next three functions to verify the Poisson Eq. 1:

- A sinusoid to test the Dirichlet BC:

$$\phi(x,y) = \phi_0 \left[\cos(\omega x)\sin(\omega y)\right] \tag{29}$$

  where ϕ_0 is a constant and ω is the frequency of the sinusoid. Both ϕ_0 and ω are set to 1.0.
- For the zeroGradient Neumann BC, the following manufactured solution is used:

$$\phi(x,y) = \left[\frac{(1-x^2)(2y^3 - 3\pi y^2 + \pi^3)}{\pi^3}\right] \tag{30}$$

- For the Neumann fixedGradient BC (with non-zero flux) yields the following:

$$\phi(x,y) = \left[\frac{(1-x^2)(\frac{y^3}{3} - \frac{\pi^2 y}{2}}{\pi^2}\right] \tag{31}$$

To verify the Navier-Stokes equations, with only one Dirichlet condition at the boundaries (for the pressure and velocity field), the following manufactured solution is proposed:

$$\begin{aligned} \vec{v}(x,y) &= \{U_x(x,y), U_y(x,y)\} \\ &= \{U_{0x}\left[\cos(\omega x)\sin(\omega y)\right], -U_{0y}\left[\sin(\omega x)\cos(\omega y)\right]\} \\ P(x,y) &= P_0\left[\cos(\omega x)\cos(\omega y)\right] \end{aligned} \tag{32}$$

where U_{0x}, U_{0y}, and P_0 are constants, and all, for simplicity are set to 1.0.

In addition, to verify the Navier-Stokes equations with combined the Dirichlet and Neumann BCs, we consider two classical flows between parallel plates, Poiseuille and Couette. The analytical solution for the velocity and pressure fields are:

$$\begin{aligned} \vec{v}(x,y) &= \{U_x(y), 0\} \\ &= \left\{\frac{1}{2\mu}\left[\left(\frac{\partial p}{\partial x}\right)(y^2 - by) + U_0 + (U_b - U_0)\frac{y}{b}\right], 0\right\} \\ P(x,y) &= \left[\frac{\partial p}{\partial x}x + P_0\right] \end{aligned} \tag{33}$$

where U_b and U_0 are the upper (at $y = b$) and lower(at $y = 0$) plate velocities, and μ is the dynamic viscosity. The velocity in the form $\vec{v} = \{U_x(y), 0, 0\}$ automatically satisfies the continuity equation. The terms in the velocity that are associated with the pressure gradient are called the Poiseuille flow terms. The terms associated with the motion of the boundaries are called the Couette flow terms.

4. Results

This section reports convergence results obtained using a non-orthogonal correction at boundaries. We present convergence rate results for the Poisson equation with the

Dirichlet BC using the original code as a reference for comparison purposes. Still considering Poisson's equation, we present the results of implementing a non-orthogonal correction via a new BC (see Sec. 2.4), in which a local Gauss gradient reconstruction is performed at boundaries when a Gauss gradient is used by the OpenFOAM Laplacian scheme inside the domain (note that the term "inside the domain" means on the internal faces). Next, we applied a local Gaussian gradient reconstruction at boundaries using a Least-Squares gradient inside the domain. In this case, we observed a small diminution in convergence over the previous reconstruction with the Gauss gradient inside the domain. This issue has since been resolved. Comparative results are presented below for both cases. Then, we present the results obtained when implementing the correction via a BC using OpenFOAM gradient reconstruction libraries. Finally, the results of the approach based on implementing the non-orthogonal treatment as a source term in the solver are given. At the end of this section, the results of the convergence rate using the steady SIMPLE solver for the Navier-Stokes equations are presented. Results obtained with the standard BC are compared with those using the new BC with corrections, first using only the Dirichlet BC (non-physical test), and then using the Dirichlet and Neumann BCs.

For the simulations in this section, we used the Gauss linear limited gradient, $\psi = 1.0$, in the Laplacian scheme. This scheme corresponds to the non-orthogonal OpenFOAM treatment, and is an approach that does not interfere with the BC treatment introduced in this work. Unless otherwise explicitly indicated, the Gaussian approach is used for the local gradient reconstruction. The least-squares local reconstruction approach was also tested, but owing to the similarity of its results with those of the Gauss alternative, we only present the latter.

In the figures shown in this section, the error is displayed on a logarithmic scale as a function of the refinement steps, with the slope representing the convergence rate. The labels Avg and Max indicate the curve for the average and maximum errors respectively. As explained in [15], the curve labeled Avg Delaunay exhibits an unexpected behavior during the final steps of the refinement. This has been attributed to the fact that refinement is only applied on boundary faces, and thereafter the cells are automatically constructed using a Delaunay algorithm, which does not guarantee that the cells will be systematically refined by a fixed factor inside the domain. This behavior will be observed in almost all plots presented below. Some explanation of the labels we use is provided below:

- Avg Left: the average error on regular prism meshes, as shown in Fig. 1a.
- Avg Alternate: the average error on regular alternate oriented prism meshes, as shown in Fig. 1b.
- Max Alternate: the maximum error on alternate oriented prism meshes, as shown in Fig. 1b.
- Avg Delaunay: the average error on Delaunay prism meshes, as shown in Fig. 1c.
- Avg Slanted: the average error on slanted hexahedral meshes, as shown in Fig. 1d.
- Avg Stretched: the average error on slanted stretched hexahedral meshes, as shown in Fig. 1e.

- Max Stretched: the maximum error on slanted stretched hexahedral meshes, as shown in Fig. 1e.

Note that we present only seven curves, in order to avoid over-populating the graphs. To reproduce most of the graphs below, we use the Poisson equation with the Dirichlet BC. We impose an analytical solution using the manufactured solution specified by Eq. 29. Other manufactured or analytical solutions used in the verification process will be cited for each specific case.

4.1. Grid Refinement on Non-Orthogonal Meshes with Original Code

In Fig. 5, we observe the results of convergence using the original code:

- Avg Left: convergence rate of 1.0 for the average error.
- Avg Alternate: convergence rate of 2.0 for the average error, because the internal faces of this mesh are orthogonal.
- Max Alternate: convergence rate of 1.0 for the maximum error, because this mesh is non-orthogonal to the boundaries.
- Avg Delaunay: no asymptotic convergence rate towards 2.0, because this mesh is only refined uniformly at the face boundaries, not at internal faces.
- Avg Slanted: asymptotic convergence rate of 1.0 for the average error.
- Avg Stretched: convergence rate of 1.0 for the average error.
- Max Stretched: convergence rate of 1.0 for the maximum error, with less precision than the average stretched error.

These results, from [15], show an asymptotic convergence rate of the first order in almost all the error curves for a variety of meshes. They are presented for comparison purposes, with the new ones obtained using the proposed boundary treatment.

4.2. Dirichlet Boundary Conditions

Here we present results of verifying the Diffusion Operator with the Dirichlet BC.

4.2.1. New BC Implementation: Local Gauss Gradient Scheme.

In this section, we use a local Gauss gradient boundary reconstruction. Inside the domain, Gauss and Least-Squares gradient schemes have been applied.

The results using the Gauss scheme inside the domain are presented in Fig. 6. If we compare these results with those shown in Fig. 5, which were obtained with the original code, it is clear that the correction on boundaries increases the slope of the functions. The error functions are straight lines with a slope of two (except for final step for the Avg Delaunay curve). The Max Alternate and Avg Stretched curves are superposed.

The results using the Least-Squares scheme inside the domain are presented in Fig. 7. Using this scheme, some curves, which are shown in bold dashed lines, do not have the expected asymptotic slope of 2.0. These are:

- Avg Left: asymptotic convergence rate of 1.0 for the average error.

- Avg Slanted: convergence rate oscillating between 1.88 and 1.47 for the average error.
- Avg Stretched: convergence rate oscillating between 1.96 and 1.49 for the average error.
- Max Stretched: convergence rate oscillating between 1.90 and 1.72 for the maximum error.

We found that this convergence loss is due to the non-orthogonal treatment at boundary faces in the Least-Squares Vector library. This non-orthogonal treatment influences the approximation error (see the Eq. 19) in the centroid of the cells neighboring the boundaries. Programming an over-relaxed non-orthogonal correction approach into the non-orthogonal treatment at face boundaries in the Least-Squares vector library led to a better convergence rate. These results are presented in Fig. 8. Comparing them with those in Fig. 7, we not only observe an improvement in convergence rate, but an improvement in accuracy too. The functions Avg Left, Avg Slanted, Avg Stretched, and Max Stretched, which showed a diminution in convergence rate, are now theoretically of the second order. The pair Avg Left and Avg Slanted, and the pair Max Alternate and Avg Stretched, are superposed.

4.2.2. New BC Implementation. Results Using OpenFOAM Libraries.

In this section, we use OpenFOAM libraries for reconstructing the local gradient in cells neighboring the boundaries (see Sec. 2.4.2). The Gauss skewCorrected scheme is specified for the library fvc::gradient.

Those results are presented in Fig. 9. They are similar, for each curve, to those obtained using local Gauss reconstruction, but in this case with a small loss of accuracy. It can be seen that all seven curves are straight lines with a slope of 2.

4.2.3. Grid Refinement with a Source Term.

The results using the source term implementation are presented in Fig 10 (see Sec. 2.4.3). These solutions are numerically identical to those obtained in the previous section, Sec.4.2.2. This can be explained by the fact that the explicit OpenFOAM libraries $fvc :: interpolate$ and $fvc :: grad$ have also been used to implement the treatment at the BC. However, this kind of implementation is only valid for the Dirichlet BC with a fixed value at the boundary. For the Neumann BC, quantities are extrapolated and updated in the class that defines this type of BC.

4.3. Neumann Boundary Conditions.

Here, we present our results with respect to the Dirichlet and Neumann BCs. As we have shown the well-founded non-orthogonal correction for the Dirichlet BC, we now shift our focus to the practical need to implement a correction for the Neumann BC (see Sec.2.3).

4.3.1. Convergence Rate using the zeroGradient or fixedGradient Neumann BC.

First, we present the results of applying the zeroGradient Neumann BC (with the projection along the boundary face being zero). For this, a square domain was chosen

with the Dirichlet BC on the left-hand and right-hand sides, and the Neumann BC applied at the top and bottom of the square. In this case, however, the manufactured solution, Eq. 30, satisfying the Neumann zeroGradient BC, is used.

The results are shown in Fig. 11. The Avg Left and Max Left plots display the convergence for the average and maximum error obtained with the original code for the Dirichlet and Neumann zeroGradient BCs on a Left mesh type (see Fig. 1a). For the Left mesh (see Fig. 1a), the Avg Left Dir-Corr and Max Left Dir-Corr plots show the convergence obtained for the average and maximum errors when the non-orthogonal correction is applied on the Dirichlet BC only. Although accuracy is increased, the asymptotic convergence order is still 1. Finally, the curves Avg Left All-Corr" and "Max Left All-Corr" display the convergence results when the non-orthogonal correction is applied on all boundaries. The convergence order corresponds to what was theoretically expected.

In Fig. 12, the curves resulting from the application of the non-orthogonal correction on all boundaries for the average error on different non-orthogonal grids are shown: Left, Alternate, Delaunay, Slanted, and Stretched see Figs.(1a-1e). All these curves display second-order convergence.

The results of applying the fixedGradient Neumann BC instead of the zeroGradient Neumann BC are shown in Fig. 13. Note the similarity in the results with those shown in Fig. 11. In Fig. 14, the curves resulting from the application of the non-orthogonal correction on all the boundaries in the Left, Alternate, Delaunay, Slanted, and Stretched grids are shown. All these curves display second-order convergence.

4.3.2. Convergence for the Navier-Stokes Equation (a non-physical model).

To test the Navier-Stokes equations (Eq. 2), we first verified a non-physical model in which we defined only the Dirichlet BC at boundaries. Two types of mesh are considered: a Regular Left Prism (see Fig 1a) and a Hexahedral Stretched (see Fig 1e).

The results using the Regular Left Prism mesh are presented in Fig. 15. In these figures, the curves labeled with the Corr string are the curves in which the new BC with non-orthogonal correction has been implemented. The values on the left-hand side of the curves are the asymptotic values of the convergence rate. The convergence rate using the standard BC handling for the two velocity components U_x and U_y is slightly greater than 1. Using the new correction at boundaries, they are asymptotically at least 2. The results using the Hexahedral Stretched mesh, not presented here, are very similar to those obtained for the Regular Left Prism, in terms of convergence and accuracy.

4.3.3. Convergence for Navier-Stokes Equations with the Dirichlet and Neumann BCs.

In this section, the verification results given apply to the simultaneous use of the Dirichlet and Neumann-type zeroGradient. For this, a typical BC for obtaining Poiseuille flow (see Eq. 33) in a rectangular domain $[0; 3.0][0; 0.2]$ is applied. P_{left} and P_{right} pressure levels are applied on the left and right boundaries respectively with $P_{left} > P_{right}$. For the velocity field, the Dirichlet BC is applied (zero velocity values) on the top and bottom boundaries, while the zeroGradient ($\frac{\partial \vec{v}}{\partial n} = 0$) is applied on the left and right sides. Simulations are performed under a non-turbulent regime ($Re = 30$ and $\nu = 10^{-04}$)

using a modified version of the simpleFOAM solver.

The results obtained using orthogonal grids (not shown here) have theoretical second order convergence. The results using the Left non-orthogonal grid (see 1a) for the average error Avg can be seen in Fig. 16. In this plot, the asymptotic convergence order is shown on the left of each curve. The curves for simulations with standard boundaries Avg U_x and Avg P have a slow rate of convergence to 1. Using the BC with the correction, we observe an improvement in the convergence rate in curves Avg U_x Corr and Avg P Corr. However, the convergence order for the U_x velocity component is not the value that was theoretically expected.

Below, we review some of the points that have an impact on the results of using the simpleFOAM algorithm:

- The most important point to consider is that the second order for U_x can be obtained modifying the simpleFOAM algorithm, if we calculate the convective flux immediately following calculation of the velocity field. The results for this approach are observed in curves Avg U_x Corr2 and Avg P Corr2. These plots show better precision and also a better convergence order. The original version of simpleFOAM does not compute the convective flow immediately after the matrix system for the velocity field has been solved, and this has a positive impact on convergence acceleration. However, this does influence the non-orthogonality handling of the BCs introduced in this study.

- When using orthogonal grids, even though the error of the second velocity component $U_y = 0$ is zero, the matrix residual of this component does not converge to zero. This problem has been found for various matrix solvers available in OpenFOAM for the velocity and pressure fields. This problem does not arise when non-orthogonal grids are used.

- The computation of Poiseuille flow with the non-orthogonal treatment applied at the boundaries shows a slower convergence than with the original method (this is almost unnoticeable for the Poisson equation). To yield a residual of 10^{-08} with the modified BC treatment on Left non-orthogonal grids of 18000, 32000, and 50000 cells, the convergence time is increased by 24, 34 and 48 percent respectively. Furthermore, if the simpleFOAM solver is modified to yield second-order accuracy, by computing the convective flux immediately after the velocity field has been computed, the computing time can be twice that required for the original BC treatment.

5. Conclusions

In this work, we have shown that the boundary treatment in OpenFOAM libraries needs to be modified to preserve the theoretical second order convergence rate. Currently, non-orthogonal grids at the boundary lead to a degraded asymptotic convergence when

the Dirichlet and Neumann BCs are imposed. The origin of this loss of convergence has been explained via a mathematical development based on truncation error analysis. This development constituted the basis for the implementation of a non-orthogonal treatment in this work.

Various correction alternatives have been developed to compensate for the convergence order degradation that occurs when non-orthogonal grids are used. With the first of these, a local reconstruction of the gradient with Gauss and least-squares schemes is applied, taking into account elements neighboring the boundary. This approach is implemented via a BC. It is then shown that a local reconstruction can be implemented using OpenFOAM's libraries for interpolating and reconstructing a gradient. A treatment for handling non-orthogonality in the Dirichlet BC by means of a source term acting directly on the solver has also been implemented. Several tests have been carried out to confirm that the proposed methods yield the theoretical convergence order for the laplacianFOAM and simpleFOAM solvers for Poisson and Navier-Stokes equations respectively. A number of tests have been carried out on different non-orthogonal grid configurations. Following a truncation error analysis, a way of optimizing the gradient computation at boundaries, $\nabla\phi$, for Dirichlet BC and to calculate the value of ϕ at boundaries for Neumann BC has been proposed.

Based on the various tests and analysess carried out in this study, we conclude that, when non-orthogonal grids are used, the current schemes found in OpenFOAM for handling the Dirichlet and Neumann BCs reduce the convergence order from 2 to 1. We have solved this by improving the BC class in the software. As a result, diffusion and convection-diffusion problems now yield second-order accuracy. Nevertheless, care has to be taken with all the solvers in addressing the Navier-Stokes equations. Specifically, some pressure correction schemes have extra boundary conditions embedded in the scheme, such as the SIMPLE method, for example. If these are not well handled, the general improvement in the Dirichlet and Neumann BCs could well be masked by the scheme, and the convergence order would then return to 1.

This work has shed some light on solving convergence-order issues when OpenFOAM is used. There is clearly a need to pursue the verification process for other schemes and models, those for unsteadiness and turbulence, for example. The nature of the treatment of the BC in each of these solvers needs to be carefully studied.

6. Figures

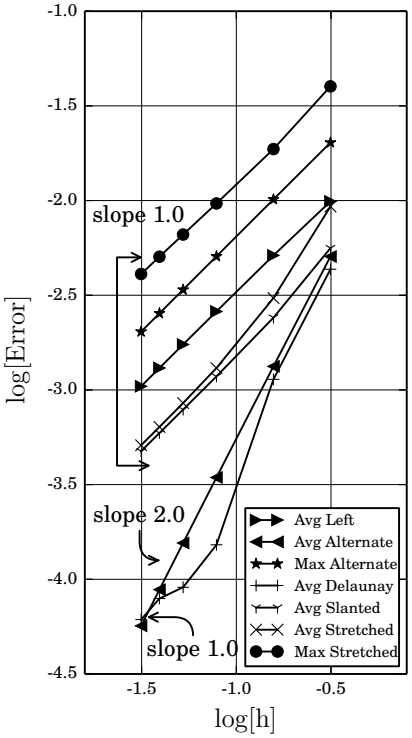


Figure 5: Convergence For The Original Code.

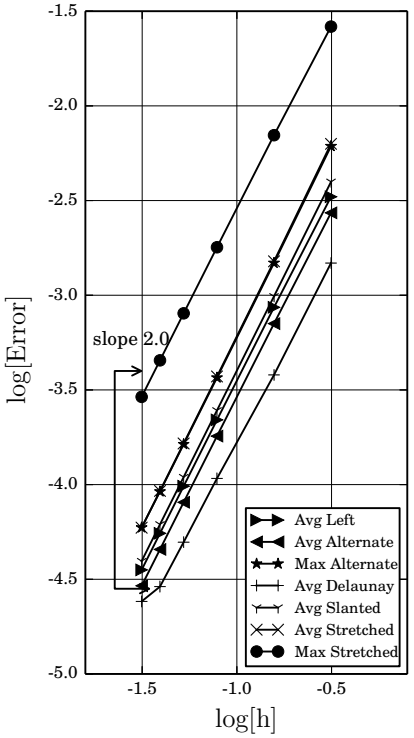


Figure 6: Convergence For Corrected Dirichlet BCs.

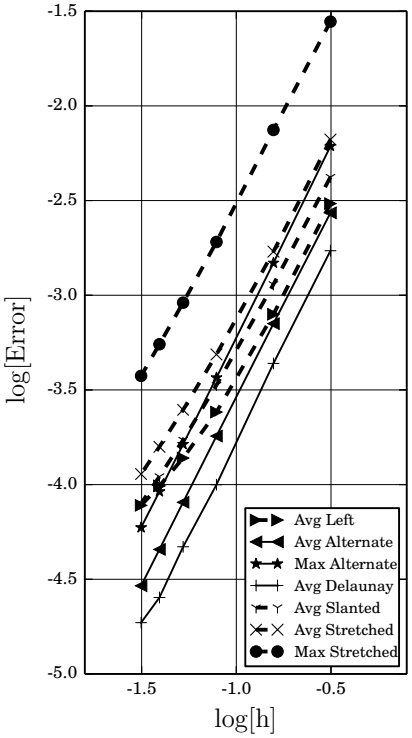


Figure 7: Convergence With Least-Squares Gradient Inside the Domain.

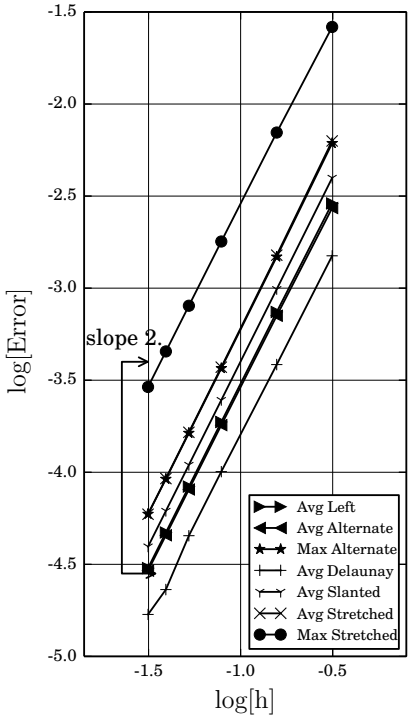


Figure 8: Convergence With Corrected Least-Squares Gradient Inside the Domain.

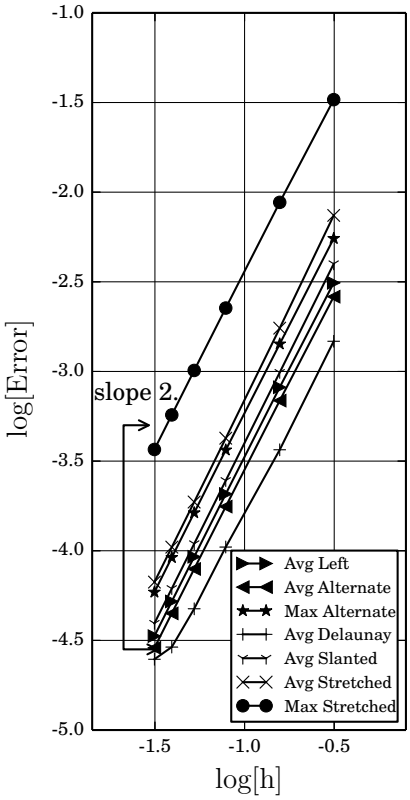


Figure 9: Convergence With Boundary Correction Using OpenFOAM Libraries.

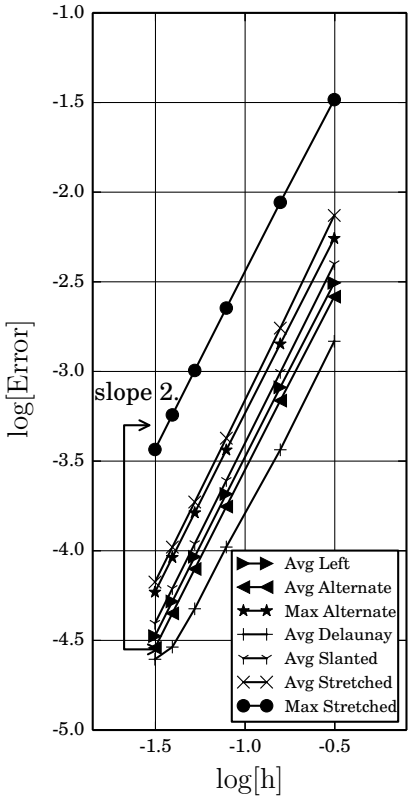


Figure 10: Convergence With Boundary Correction Using a Source Term Inside The Solver.

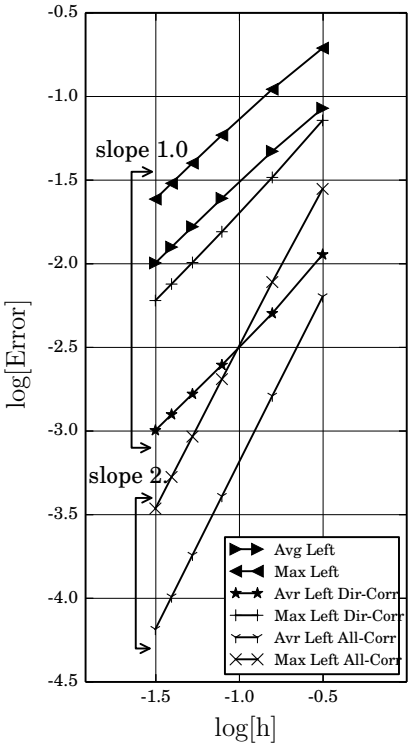


Figure 11: Convergence For Poisson Eq. With Corrected Dirichlet-Neumann ZeroGradient BC.

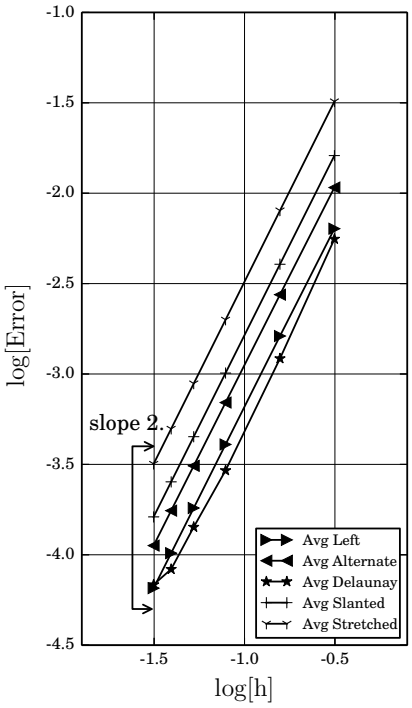


Figure 12: Convergence With Corrected Dirichlet-Neumann ZeroGradient BC on All the Boundaries.

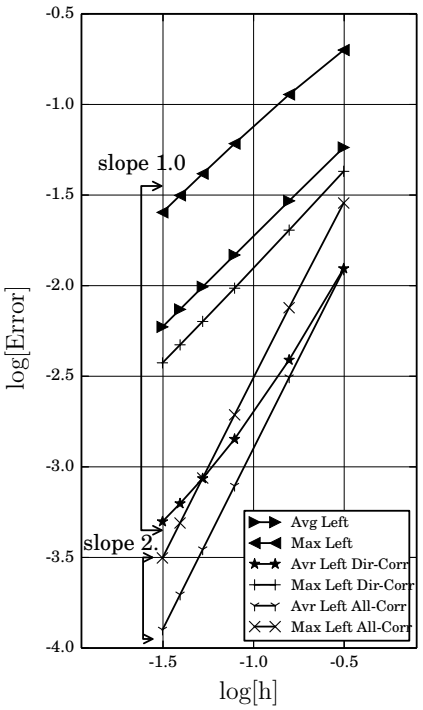


Figure 13: Convergence For Poisson Eq. With Corrected Dirichlet-Neumann FixedGradient BC.

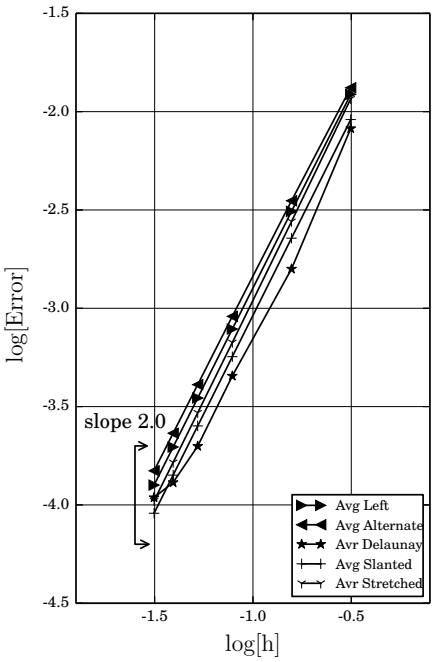


Figure 14: Convergence With Corrected Dirichlet-Neumann FixedGradient BC on All the Boundaries.

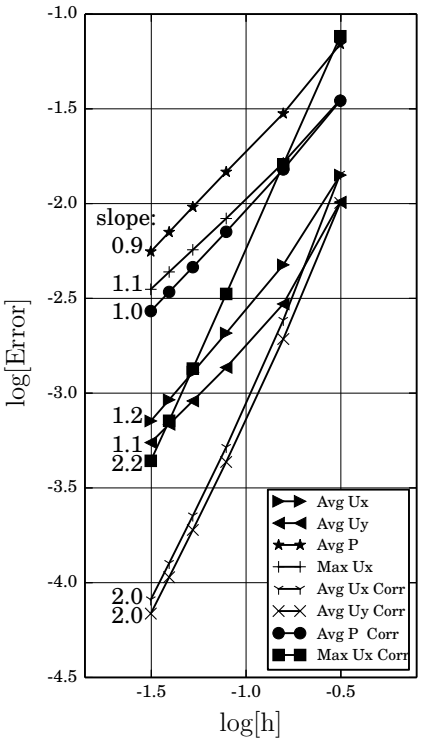


Figure 15: Convergence For Navier-Stokes Eq. With Dirichlet BC Only.

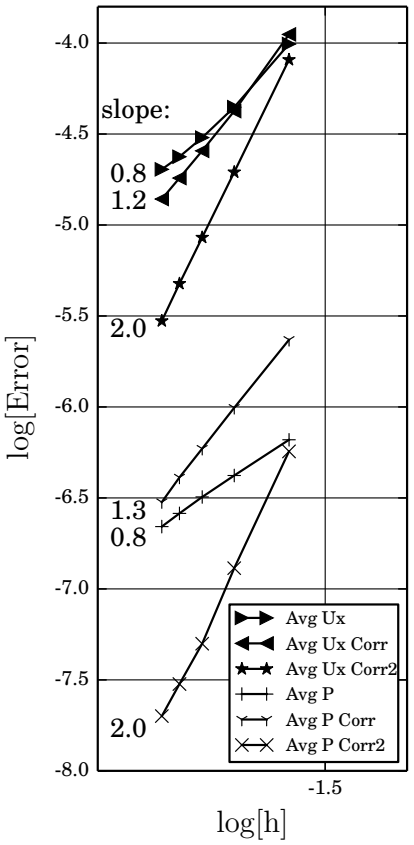


Figure 16: Convergence For Navier-Stokes Eq. With Dirichlet And ZeroGradient BC.

7. References

[1] M. Schafer, Computational Engineering. Introduction to Numerical Methods, Springer, 2006.

[2] J. Blazek, Computational Fluid Dynamics: Principles and Applications, 2nd Edition, Elsevier Science, 2006.

[3] J. Ferziger, M. Peric, Computational Methods for Fluid Dynamics, 3nd Edition, Springer, 2002.

[4] H. Jasak, Error analysis and estimation for the finite volume method with applications to fluid flows, Ph.D. thesis, Imperial College (Jun. 1996).

[5] H. Jasak, A. Gosman, Automatic resolution control for the finite-volume method, part 2: a-posteriori error estimates, Numerical Heat Transfer, Part B: Fundamentals 38 (3) (2000) 237–256.

[6] H. Jasak, A. Gosman, Automatic resolution control for the finite-volume method, part 2: Adaptive mesh refinement and coarsening, Numerical Heat Transfer Part B Fundamentals 38 (3) (2000) 257–271.

[7] H. Jasak, A. Gosman, Automatic resolution control for the finite-volume method, part 3: Turbulent flow applications, Numerical Heat Transfer Part B Fundamentals 38 (3) (2000) 273–290.

[8] H. Jasak, A. Gosman, Rsidual error estimate for the finite volume method, Numerical Heat Transfer, Part B: Fundamentals 39 (1) (2001) 1–19.

[9] H. Jasak, A. Gosman, Element residual error estimate for the finite volume method, Computers and Fluids (2003) 223–248.

[10] F. Juretic, Error analysis in finite volume cfd, Ph.D. thesis, Department of Mechanical Engineering, Imperial College. (2004).

[11] F. Juretic, A. Gosman, Error analysis of the finite-volume method with respect to mesh type, Numerical Heat Transfer, Part B: Fundamentals: An International Journal of Computation and Methodology Fundamentals 57 (6) (2010) 414–439.

[12] F. Moukalled, L. Mangani, M. Darwish, The Finite Volume Method in Computaional Fluid Dynamic, Springer, 2015.

[13] P. Khosla, S. Rubin, Diagonally dominant second-order accurate implicit scheme, Computers Fluids 2 (1974) 207–209.

[14] S. Muzaferija, Adaptive finite volume method for flow predictions using unstructured meshes and multigrid approach., Ph.D. thesis, University of London (1994).

[15] H. Noriega, F. Guibault, M. Reggio, R. Magnan, A case-study in open-source cfd code verification. part i: Convergence rate loss diagnosis.Unpublished.

[16] P. Knupp, K. Salari, Verification of Computer Codes in Computational Science and Engineering, Chapman and Hall/CRC, 2002.

[17] P. Roache, Verification of codes and calculations, AIAA Journal 36, 5 (5) (1998) 696–702.

[18] F. Stern, R. Wilson, J. Shao, Quantitative v-v of cfd simulations and certification of cfd codes, International Journal for Numerical Methods in Fluids 50 (11) (2006) 1335–1355. doi:10.1002/fld.1090.
URL `http://dx.doi.org/10.1002/fld.1090`

[19] J. Abanto, D. Pelletier, A. Garon, J. Trepanier, M. Reggio, Verification of some commercial cfd codes on atypical cfd problems, 43rd AIAA Aerospace Sciences Meeting and Exhibit 1 (2005) 15693–15725.

[20] C. J. Roy, C. C. Nelson, T. M. Smith, C. C. Ober, Verification of euler/navierstokes codes using the method of manufactured solutions, International Journal for Numerical Methods in Fluids 44 (6) (2004) 599–620. doi:10.1002/fld.660.
URL `http://dx.doi.org/10.1002/fld.660`

[21] J. Iannelli, An exact non-linear navierstokes compressible-flow solution for cfd code verification, International Journal for Numerical Methods in Fluids 72 (2) (2013) 157–176. doi:10.1002/fld.3731.
URL `http://dx.doi.org/10.1002/fld.3731`

[22] L. Eca, M. Hoekstra, An introduction to cfd code verification including eddy-viscosity models, 2006.

[23] L. Eca, M. Hoekstra, Code verification of unsteady flow solvers with the method of the manufactured solutions, International offshore and polar engineering; ISOPE 2007 (2007) 2012–2019.

[24] L. Eca, M. Hoekstra, A. Hay, D. Pelletier, Verification of rans solvers with manufactured solutions, Engineering with Computers 23 (2007) 253–270.