LIÈGE université Library

http://lib.uliege.ac.be http://matheo.uliege.be

Prémices de l'optimisation topologique des échangeurs de chaleur

Auteur : Chevalier, Louise
Promoteur(s) : Duysinx, Pierre
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil électromécanicien, à finalité spécialisée en énergétique
Année académique : 2017-2018
URI/URL : http://hdl.handle.net/2268.2/4501

Avertissement à l'attention des usagers :

Tous les documents placés en accès ouvert sur le site le site MatheO sont protégés par le droit d'auteur. Conformément aux principes énoncés par la "Budapest Open Access Initiative"(BOAI, 2002), l'utilisateur du site peut lire, télécharger, copier, transmettre, imprimer, chercher ou faire un lien vers le texte intégral de ces documents, les disséquer pour les indexer, s'en servir de données pour un logiciel, ou s'en servir à toute autre fin légale (ou prévue par la réglementation relative au droit d'auteur). Toute utilisation du document à des fins commerciales est strictement interdite.

Par ailleurs, l'utilisateur s'engage à respecter les droits moraux de l'auteur, principalement le droit à l'intégrité de l'oeuvre et le droit de paternité et ce dans toute utilisation que l'utilisateur entreprend. Ainsi, à titre d'exemple, lorsqu'il reproduira un document par extrait ou dans son intégralité, l'utilisateur citera de manière complète les sources telles que mentionnées ci-dessus. Toute utilisation non explicitement autorisée ci-avant (telle que par exemple, la modification du document ou son résumé) nécessite l'autorisation préalable et expresse des auteurs ou de leurs ayants droit.

Université de Liège - Faculté des Sciences Appliquées

Prémices de l'optimisation topologique des échangeurs de chaleur

Travail de fin d'études réalisé en vue de l'obtention du grade de master "Ingénieur Civil en électromécanique"

Auteur :
Louise Chevalier

Promoteur :
P.Duysinx

Année académique 2017-2018

Prémices de l'optimisation topologique des échangeurs de chaleur

Auteur :
Louise Chevalier

Promoteur :
P.Duysinx

Université de Liège - section électromécanique

Année académique 2017-2018

Remerciement

Je tiens à remercier toute l'équipe ERT de Safran Aero Boosters de m'avoir accueillie, encadrée et permis de découvrir la vie en entreprise.

Je remercie le professeur P. Duysinx pour m'avoir conseillée et guidée. Son intervention a permis de préciser l'objet du présent travail et de créer une synergie entre l'entreprise et l'université.

Enfin, merci à ma famille et à mes amis pour m'avoir supportée et encouragée pendant cette période de travail absorbant.

Abstract

L'optimisation topologique, déjà bien présente en mécanique des structures, pourrait, si elle était appliquée aux échangeurs de chaleur, conduire à des designs inattendus mais plus performants. La fabrication additive permet d'envisager concrètement la réalisation de ces nouveaux designs.
Contrairement à la mécanique des structures, l'optimisation des échangeurs de chaleur nécessite de prendre en compte la conjugaison de l'écoulement de fluides et du transfert de chaleur ce qui complexifie le problème.
Des informations sur l'optimisation topologique des échangeurs de chaleur ont été rassemblées dans le but de définir ce qui est réalisable actuellement en entreprise sans nouveaux investissements. Au-delà de cette analyse, la définition d'une méthode et l'obtention de premiers résultats sont aussi souhaitables.
Un code Matlab pour l'optimisation topologique en mécanique des structures est modifié pour tenir compte du transfert de chaleur par conduction entre une source et un dissipateur de chaleur. Des résultats inattendus sur obtenus et présentés notamment pour ce qui pourrait être un échangeur surfacique. Pour aller plus loin et envisager de tenir compte de l'écoulement fluide et du transfert de chaleur par convection, un couplage entre deux logiciels -dont un solveur fluide- est étudié. Le logiciel Fluent est choisi pour différentes raisons dont la présence d'un solveur adjoint. Les liens entre Matlab et Fluent se font manuellement. Quelques problèmes persistent, notamment avec les résultats fournis par le solveur adjoint lesquels ne sont pas exactement les résultats espérés. Toutefois, l'étude est lancée et il sera certainement possible après des analyses plus poussées -et peut-être des nouvelles versions des logiciels- d'aboutir à ce couplage et de l'automatiser.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

L'optimisation topologique est bien ancrée dans le domaine de la mécanique des structures et se développe de plus en plus dans divers domaines comme la thermique, la biologie, et bien d'autres. De nombreuses recherches sont en cours concernant les échanges de chaleur et certains résultats obtenus semblent très prometteurs. Le couplage entre l'écoulement d'un fluide et le transfert thermique est également un sujet abordé dans quelques études et articles.

Le présent travail est basé sur un stage réalisé en entreprise chez "Safran Aero Boosters" au sein d'une équipe spécialisée dans les calculs thermiques.

Le but de celui-ci est, dans un premier temps de faire le point sur ce qui existe déjà en matière d'optimisation topologique des échangeurs de chaleur et sur la manière dont cela est réalisé. Ensuite d'investiguer sur ce qui serait réalisable au niveau de l'entreprise dans un futur plus ou moins proche, avec les moyens disponibles actuellement. Effectuer quelques essais et obtenir des premiers résultats figurent aussi parmi les objectifs. Les échangeurs considérés sont des échangeurs air-huile qui servent à transmettre une partie de la puissance de l'huile à l'écoulement d'air tout en minimisant les pertes de charge. Seule la partie air sera modélisée lors de ce projet.

Les premières questions qui se sont imposées lors de ce travail de fin d'étude furent : Qu'est-il possible de faire concrètement à ce jour et comment ? Le second chapitre de ce document tente donc d'y répondre ; d'abord en résumant les articles qui paraissent les plus pertinents par rapport aux attentes de l'équipe et ensuite en présentant les différentes méthodes disponibles tout en donnant quelques compléments d'informations. Il expose aussi les notions de base que sont l'optimisation et les échangeurs de chaleur ainsi que quelques logiciels commercialisés.

Dans un second temps, une partie plus pratique est envisagée à partir des logiciels mis à disposition par l'entreprise. Le but étant au final de déterminer si l'optimisation topologique pourrait être utilisée comme méthode de conception de nouveaux designs avec les moyens opérationnels, dans un avenir plus ou moins proche.

Le troisième chapitre s'appuie sur un code écrit en Matlab pour réaliser de l'optimisation topologique des structures avec pour but de maximiser la raideur de la structure sous certaines contraintes. Ce code a été adapté en vue d'une utilisation centrée sur des problèmes thermiques ne tenant compte que du transfert de chaleur sous forme de conduction dans la matière. Plusieurs cas intéressants sont présentés et l'influence de différents paramètres sur le processus d'optimisation et le design final est étudiée.
Dans ce même chapitre 3, une nouvelle propriété appelée "entransie" est utilisée dans de nombreux articles consultés mais néanmoins controversée est présentée. Les critiques la concernant

sont également énoncées.

Enfin, le quatrième et dernier chapitre de ce travail traite de l'optimisation poussée à un niveau supérieur dans le but de considérer un fluide et le transfert de chaleur par convection en plus du transfert par conduction. Le dialogue sur l'optimisation topologique avec l'université de Liège a pu être lancé par le biais de ce stage. Le souhait de la mise en commun du travail a été évoqué afin que le celui soit partagé : l'université se concentrant sur le processus d'optimisation implémenté sur Matlab et la considération du logiciel OpenFoam comme solveur fluide et Safran Aero Boosters envisageant l'utilisation d'un autre solveur plus fréquent en entreprise.
L'idée présentée ici est de coupler deux logiciels (Matlab et Fluent). Ce travail ne porte dont pas sur le processus d'optimisation à proprement parlé mais tente de déterminer si le couplage dont il est question ci-dessus est réalisable et dans quelle mesure il générerait des résultats exploitables. Le lien entre les deux logiciels et le fonctionnement du logiciel CFD sont décrits et quelques résultats sont exposés.

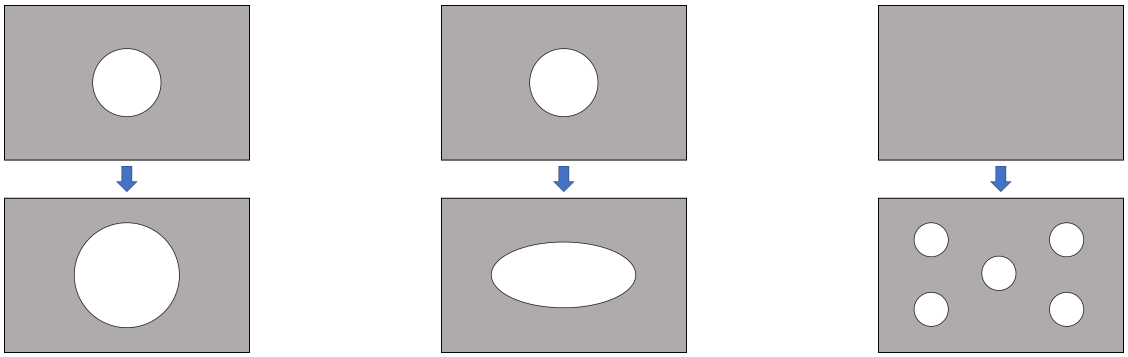
Chapitre 2

Recherches dans la littérature

2.1 L'optimisation en général

L'optimisation est un processus au cours duquel un design est modifié dans le but d'améliorer ses performances. Il existe trois types d'optimisation :

- l'optimisation **paramétrique** (Fig.2.1a) qui a pour but de modifier les dimensions des variables (longueurs, épaisseurs, ...) afin d'améliorer les performances de l'objet optimisé. Dans le cas des échangeurs de chaleur, ce type d'optimisation pourrait par exemple modifier la hauteur de l'échangeur, le diamètre des tubes dans lesquels un fluide passe ou encore l'épaisseur des ailettes s'il y en a.
- l'optimisation de **forme** (Fig.2.1b) qui modifie les frontières prédéterminées pour obtenir un design optimal. Ainsi, pour un échangeur de chaleur, cette optimisation pourrait modifier la forme des canaux ou celle des ailettes s'il y en a.
- l'optimisation **topologique** (Fig.2.1c) qui a pour but de déterminer, dans un domaine de conception fixé et avec des conditions aux limites données, où la matière devrait être placée pour minimiser une fonction objectif tout en vérifiant une ou plusieurs contraintes. Autrement dit, l'optimisation place de la matière aux endroits nécessaires et détermine comment celle-ci doit être connectée dans le domaine. Dans le cas d'un échangeur de chaleur, l'optimisation topologique pourrait déterminer la forme des canaux mais aussi leur nombre et la façon de les placer au mieux dans le domaine.



(a) Optimisation paramétrique (b) Optimisation de forme (c) Optimisation topologique

FIGURE 2.1 – Schémas explicatifs des différents types d'optimisation

Dans les deux premiers types d'optimisation, la topologie du design ne peut être modifiée durant l'optimisation, c'est-à-dire que la présence de trou ou non dans le design doit être décidé

à priori.
Le processus d'optimisation topologique permet d'obtenir des designs complètement nouveaux, souvent inenvisageables.
Bien sûr, le temps nécessaire à l'optimisation topologique est bien plus grand que celui pour l'optimisation de taille. C'est pourquoi, l'optimisation topologique ne peut être réalisée que pour des projets de longues durées que l'entreprise doit pouvoir planifier sur un long terme. Les différents types d'optimisation peuvent également être combinés et ainsi, l'optimisation topologique pourrait être utilisée dans un premier temps, sans forcément la faire tourner jusqu'au bout, pour définir la topologie du design avant d'appliquer de l'optimisation de forme et de taille sur celui-ci qui sont des processus beaucoup plus rapides.
Un autre atout de l'optimisation topologique est qu'elle se combine très bien avec la fabrication additive de plus en plus utilisée dans le monde industriel qui permet d'obtenir des designs plus complexes et de diminuer la quantité de matière nécessaire à la fabrication d'une pièce.

2.2 Les échangeurs de chaleur

Un échangeur de chaleur est un appareil utilisé pour transférer de l'énergie thermique d'un milieu vers un autre à plus basse température. Ces systèmes se retrouvent partout, du radiateur dans les maisons aux échangeurs industriels. Dans le domaine de l'aéronautique, les échangeurs sont, entre autres, utilisés pour refroidir l'huile et le carburant des moteurs d'avion et garder ces fluides dans un intervalle de température bien précis afin d'améliorer les performances du moteur.

Les échangeurs de chaleur utilisés dans l'aéronautique doivent être compacts et les plus légers possible. Pour ce faire, le choix du type d'échangeur se porte souvent sur les échangeurs à courant croisé. Comme le montre le schéma de la Fig.2.2, les fluides chaud et froid ont des directions orthogonales l'une à l'autre. La Fig.2.3 montre à quoi ressemble un échangeur de chaleur vu de l'extérieur.

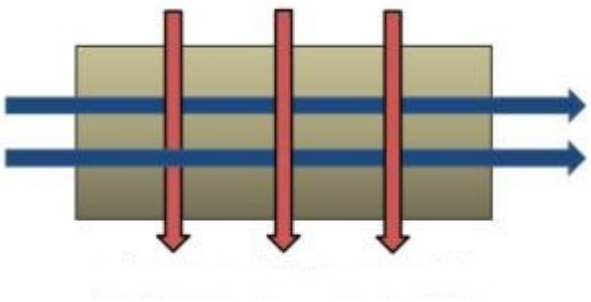


FIGURE 2.2 – Schéma d'un échangeur de chaleur à courant croisé [15]

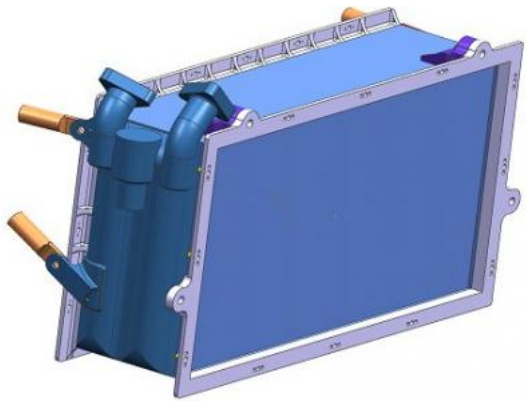


FIGURE 2.3 – Extérieur d'un échangeur de chaleur [41]

La plupart du temps, des ailettes sont également utilisées dans les échangeurs de chaleur pour augmenter la surface d'échange entre les deux fluides et ainsi améliorer leur performance. L'amélioration des performances des échangeurs de chaleur est un point clé dans de nombreux systèmes car beaucoup de pertes peuvent être réduites et la chaleur peut être transférée d'un endroit à un autre où elle est nécessaire. Cette récupération de la chaleur réduit les coûts auparavant dépensés pour réchauffer ces endroits.

2.3 État de l'art de l'optimisation topologique dans le cadre des échangeurs de chaleur

De nombreuses publications existent sur l'optimisation topologique des échangeurs de chaleur mais rares sont celles qui contiennent des détails sur l'implémentation du processus d'optimisation ou sur le logiciel utilisé pour obtenir les résultats présentés. D'autre part, si des logiciels ont été développés sur ce sujet les informations les concernant sont limitées.
La majorité des recherches déjà menées sur l'optimisation topologique avec transfert de chaleur et écoulement de fluide se cantonnent à des écoulements laminaires car la turbulence génère nombre de difficultés supplémentaires, notamment en raison du maillage qui doit être très fin pour prendre en compte les plus petits tourbillons. Afin de faciliter les équations qui entrent en jeu dans le processus, les écoulements sont la plupart du temps considérés stationnaires et les fluides incompressibles.
Certains articles sont plus intéressants que d'autres et les plus pertinents sont présentés dans la suite de cette section.

En 2009, E. Dede [13] utilise le logiciel COMSOL couplé à un optimiseur MMA ("Method of Moving Asymptotes") sur Matlab pour optimiser des problèmes de transfert de chaleur et d'écoulement de fluide. Dans un premier temps, un cas 3D de conduction pure d'un dissipateur de chaleur avec une génération de chaleur répartie dans le domaine de conception est résolu avec comme objectif de minimiser la température moyenne du domaine. Ensuite, un cas plus compliqué 2D avec du transfert de chaleur et de l'écoulement laminaire entre une entrée et deux sorties en régime établi de fluide incompressible est considéré. L'objectif est double, minimiser la température moyenne et la puissance totale dissipée par le fluide.

En 2012, E. Kontoleontos [20] applique la méthode d'optimisation topologique à la conception de conduites/collecteurs dans le but de minimiser la perte de charge et/ou maximiser la différence de température entre la sortie et l'entrée. L'écoulement est incompressible et laminaire ou turbulent, les variations de la viscosité turbulente sont prises en compte.

En 2013, G. Marck [22] étudie l'optimisation topologique des problèmes de transfert de chaleur et de masse dans les cas des écoulements laminaires avec pour objectif de minimiser les pertes de charge. Pour ce faire, il utilise la méthode des volumes finis, la méthode SIMP et la méthode MMA pour l'optimisation.

Cette même année, T. Matsumori [29] utilise l'optimisation topologique pour cette fois maximiser le transfert de chaleur dans un échangeur à une entrée et une ou deux sortie(s) tout en gardant une puissance en entrée constante. L'écoulement est incompressible et la source de chaleur est considérée d'abord dépendante et ensuite indépendante de la température. L'approche utilisée est basée sur la densité et l'optimisation se fait à l'aide du logiciel COMSOL Multiphysics
.

En 2014, T. Van Oevelen [39] optimise un dissipateur de chaleur sujet à un écoulement laminaire et en régime permanent par optimisation topologique dans le but de maintenir la température de la source de chaleur la plus basse possible. Le domaine est divisé en deux parties : la partie inférieure est composée de solide en contact avec une source de chaleur répartie et la deuxième partie, la couche supérieure, est composée d'ailettes. Seule la partie supérieure est optimisée afin de trouver la répartition optimale des ailettes. Le cas étudié dans l'article considère un écoulement de Stokes et tient compte aussi bien de la conduction que de la convection.

En 2015, E. Dede [21] recourt à l'optimisation topologique pour minimiser la résistance thermique d'un dissipateur de chaleur 3D refroidi par un jet d'air perpendiculaire à la surface tout en considérant la conduction et la convection sur les surfaces latérales en régime permanent. Pour ce faire, la méthode SIMP a été utilisée. Le design obtenu au final pour la pièce a été fabriquée via la méthode additive et ainsi a pu être comparée à des références sur des essais réels.

Toujours en 2015, J.H.K. Haertel [23] utilise COMSOL avec la méthode GCMMA ("Globally Convergent Method of Moving Asymptotes") comme méthode d'optimisation. Dans cet article, un dissipateur de chaleur 2D soumis à de la convection forcée pour refroidir une surface avec une production constante de chaleur est étudié. L'écoulement est stationnaire et laminaire et le fluide incompressible. L'objectif est de minimiser la température moyenne du dissipateur pour une perte de charge donnée. Le but de cet article est de montrer la facilité d'utilisation du logiciel COMSOL pour réaliser de l'optimisation topologique de systèmes multi physiques complexes.

K. Yaji [24] utilise la méthode à variation de frontières pour l'optimisation topologique d'un problème couplé fluide-thermique dans le but de maximiser les performances d'un dissipateur de chaleur. Le domaine de conception considéré est fixe avec une entrée et une sortie. La perte de charges entre l'entrée et la sortie est fixe. Le fluide est incompressible et l'écoulement est laminaire et stationnaire.

Enfin, P.Papazoglou [40] cherche à maximiser le transfert de chaleur dans un échangeur de chaleur à deux écoulements non mixés pour des pertes de charge fixées et des contraintes sur les dimensions. L'écoulement considéré est un écoulement de Stokes incompressible, le transfert de chaleur est caractérisé par un nombre de Péclet (le rapport du transfert par convection sur le transfert par conduction) relativement élevé et il n'y a pas de génération interne de chaleur. Il utilise une méthode spéciale (méthode à matériaux multiples) afin de garantir le non mélange des fluides. Cependant, la convergence du processus est difficile à obtenir. Cet article est le seul consulté qui considère l'échange de chaleur entre deux fluides différents.

En 2016, M.Zhou [26] applique l'optimisation topologique à des problèmes de transfert de chaleur conductif et convectif combinés en utilisant CATIA (CAD), SIMULIA-Abaqus (finite element analysis) et SIMULIA-Tosca (optimisation topologique). Les auteurs considèrent, entre autres, un dissipateur de chaleur 3D avec un flux de chaleur centré en bas du domaine, soumis à différentes contraintes (symétrie et/ou manufacture). L'objectif étant de minimiser la compliance thermique, ce qui revient à minimiser la température moyenne du domaine comme il sera expliqué dans la section 3.1.2, avec une contrainte sur le volume. Un des designs obtenus est vérifié et comparé à un cas de référence avec un modèle thermofluide sur le logiciel COMSOL-Multiphysics. La conclusion est que le design optimisé présente de meilleures performances thermiques que le design de référence. Cependant, la perte de charge entre l'entrée et la sortie est plus importante avec le design optimisé qu'avec celui de référence ce qui est certainement lié à la plus grande surface d'échange nécessaire à un meilleur échange thermique.

Le tableau 2.1, reprends les différents articles et les informations importantes.

Article	Ecoulement	Objectif	Méthode	Logiciel
[13]	Laminaire	minimiser la température moyenne et la puissance totale dissipée	MMA	COMSOL
[20]	Laminaire et Turbulent	minimiser ΔT et/ou maximiser ΔT entrée-sortie		
[22]	Laminaire	minimiser ΔP et maximiser Q	SIMP + MMA	
[29]	Laminaire	maximiser transfert de chaleur	Homogénéisation	COMSOL
[39]	Laminaire	minimiser T_{source}		
[21]		minimiser résistance thermique	SIMP	
[23]	Laminaire	minimiser T moyenne	GCMMA	COMSOL
[24]	Laminaire	maximiser transfert de chaleur	Level-set	
[40]	Laminaire	maximiser transfert de chaleur		
[26]		minimiser compliance thermique		Tosca

Tableau 2.1 – Tableau récapitulatif des informations présentées dans l'état de l'art

2.4 Descriptions de quelques méthodes d'optimisation topologique

Différentes méthodes d'optimisation peuvent être utilisées en optimisation topologique, en voici quelques unes (plusieurs méthodes sont décrites dans l'article [18]) :

- **Méthodes d'homogénéisation** :
  Dans ces méthodes, chaque élément est traité comme un matériau poreux dont les microstructures peuvent être modélisées et contrôlées. Ces méthodes opèrent sur un domaine fixe d'éléments finis et sont souvent associées à la méthode SIMP (Solid Isotropic Method with penalization) où chaque élément est défini par une seule variable d'optimisation, souvent la densité de matériau en mécanique des structures. D'autres méthodes comme les méthodes ESO et BESO ("(bi-directional) Evolutionary Structural Optimization") sont mentionnées dans la littérature. Dans ces méthodes, chaque élément est traité comme vide ou solide via une variable discrète. Au fur et à mesure des itérations, des éléments sont ajoutés ou retirés du domaine de conception. Selon [43], ces dernières méthodes sont des versions discrètes des méthodes basées sur la densité.

- **Méthodes à variation de frontières** (Level-set) :
  Méthodes où les frontières entre les zones solide et fluide du design sont définies comme la courbe de niveau annulant une fonction de niveau. Cette fonction définit implicitement la structure et dépend de la variable d'optimisation. Cette méthode est également appliquée sur un maillage fixe, ce qui permet un suivi précis des frontières fluide-solide . Celles-ci sont également plus lisses qu'avec les méthodes de densité. La méthode à variation de frontières se marie très bien avec les éléments finis étendus(XFEM). (Pour plus de détails

sur les méthodes à variation de frontières, le lecteur est renvoyé vers les articles [25], [24], [11] ou [34] par exemple)

2.5 Logiciels existant pour l'optimisation topologique

Au vu de la littérature, il existe plusieurs logiciels sur le marché permettant de réaliser de l'optimisation topologique d'échangeurs de chaleur. Le logiciel le plus souvent mentionné dans les articles ayant été consultés est **COMSOL Multiphysics** [12].

COMSOL est un logiciel de simulation basé sur la méthode des éléments finis qui a pour spécialité le fait de pouvoir résoudre des phénomènes couplés. Au logiciel de base peuvent s'ajouter des modules complémentaires afin de traiter des cas spéciaux. De nombreux modules sont disponibles dont, entre autres,

- *Heat Transfer Module* qui permet d'étudier le transfert de chaleur par conduction, convection et radiation ainsi que les écoulements non-isothermes conjugués au transfert de chaleur ;
- *CFD Module* qui permet la simulation de composants et de systèmes impliquant des écoulements de fluides ;
- *Optimization Module* qui doit être utilisé conjointement avec un autre module. L'optimisation se réalise en 4 étapes : (1) définir une fonction objectif, (2) définir un ensemble de variables de conception, (3) définir un ensemble de contraintes, de limites sur les variables ou des conditions d'opération qui doivent être satisfaites et enfin (4) utiliser le module d'optimisation pour améliorer le design en faisant varier les variables tout en satisfaisant les contraintes. La mise à jour des variables peut se faire par une technique sans dérivées qui permet d'approximer le gradient ou par une technique avec dérivées comme la méthode MMA (voir section 3.2.1) qui utilise le véritable gradient calculé analytiquement.

Ce logiciel est utile et fonctionne correctement si on se contente de faire ce pour quoi il est conçu. Il est par contre difficile de le modifier en profondeur car l'accès au code n'est pas possible. De plus, différents modules sont nécessaires et les licences sont assez onéreuses.

D'autres logiciels peuvent également être mentionnés tels que

- **SIMULIA TOSCA Fluid** [45] qui s'utilise avec des solveurs CFD standards de l'industrie. Cependant, l'optimisation topologique effectuée à l'aide de ce logiciel connait des limites et est plutôt utilisée comme première étape avant une optimisation de forme ;
- **HELYX** [16] qui consiste en un ensemble de logiciels CFD pour l'analyse et l'optimisation de design d'applications industrielles auquel on peut ajouter des modules d'extension pour couvrir des applications plus spécialisées.
- **InconCFD** [32] qui fournit une structure permettant de gérer un projet de sa préparation à l'analyse des résultats. Différents modules sont disponibles et distribués comme logiciels open source tels que *iconCFD Thermal* pour le traitement des problèmes de transfert de chaleur ou *iconCFD Optimize* pour l'optimisation de forme et topologique avec utilisation de la méthode adjointe et différentes fonctions objectifs pré-définies.

Tous les logiciels cités ont leurs avantages et leurs inconvénients et dans la plupart des cas peu d'informations sont disponibles pour se faire une idée précise de leur utilisation.
L'optimisation topologique pourrait aussi très bien être réalisée en couplant différents logiciels comme OpenFoam (ou Fluent) et Matlab. L'équipe ayant entouré ce travail se servant tous les jours de Fluent, c'est cette dernière solution qui a été retenue dans le cadre de ce stage et qui sera présentée plus en détails dans des chapitres ultérieurs.

2.6 Conclusion

Ce premier chapitre résume ce qui se fait en matière d'optimisation topologique des échangeurs de chaleur à l'heure actuelle et tente de présenter les moyens utilisés. Pour commencer, les différents types d'optimisation sont présentés et le principe d'un échangeur de chaleur est rapidement décrit. Différents articles sont ensuite cités et commentés brièvement. La recherche dans la littérature a également conduit à la découverte de plusieurs logiciels dont COMSOL le plus souvent mentionné. Ce logiciel est aussi le plus documenté cependant il est très onéreux.

Chapitre 3

Cas de la conduction pure sur Matlab

3.1 Description du problème

Le livre [14] de M.P. Bendsœ et O. Sigmund présente une implémentation sur Matlab d'un code d'optimisation topologique en mécanique des structures. Matlab [37] est un langage de programmation très utile et très répandu permettant entre autres d'effectuer des calculs numériques et d'analyser des données. Le but du processus est de minimiser la compliance, ce qui revient à maximiser la rigidité de la structure car la compliance est l'inverse de la raideur, avec des conditions aux limites telles que sur le schéma Fig.3.1. Ce code est basé sur les éléments finis et utilise le critère d'optimalité pour mettre à jour les variables de conception.

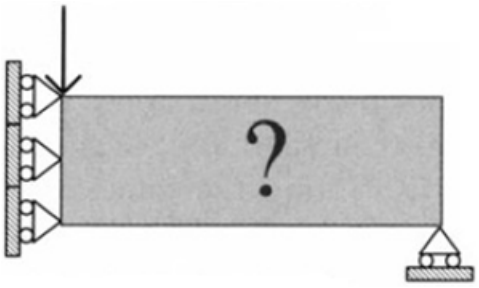
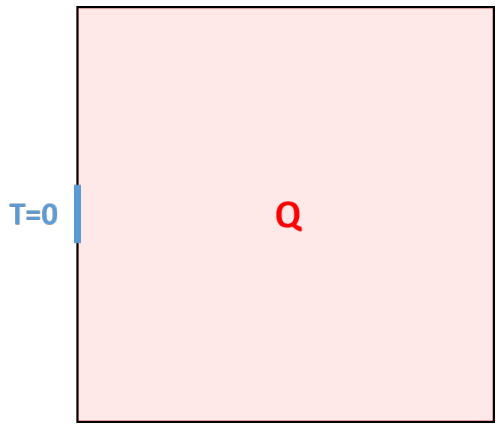
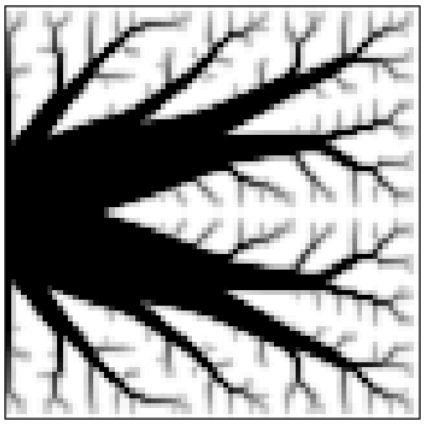


FIGURE 3.1 – Schéma de la poutre considérée de base dans le code Matlab [14]

Dans le cas de la conduction pure, certaines modifications (décrites dans le livre [14]) doivent être apportées au code. Le code utilisé pour obtenir les résultats présentés tout au long de ce chapitre est disponible en annexe A.
Le cas de base présenté dans [14] (ici nommé "cas 1") représente une source de chaleur distribuée sur l'entièreté du domaine de conception carré avec un dissipateur de chaleur (une source fixée à 0 $[K]$) situé sur sa frontière ouest comme illustré sur le schéma de la Fig.3.2a. Le résultat obtenu est présenté à la Fig.3.2b.



(a) Schéma des conditions aux limites



(b) Design obtenu

FIGURE 3.2 – Conditions aux limites et design obtenu pour le cas 1 [14]

Un problème d'optimisation général s'exprime comme suit :

$$\text{Trouver :} \quad \underline{x} = (\rho_1, \rho_2, \cdots, \rho_{N_e}) \tag{3.1}$$

$$\text{Minimiser :} \quad f_0 \tag{3.2}$$

$$\text{Sujet à :} \quad g_i = 0 \tag{3.3}$$

$$f_i \leq 0 \tag{3.4}$$

Où,
- $\underline{x}$ est un vecteur dont les composantes, au nombre d'éléments composant le domaine, sont les variables de conception ρ_e étant des "porosités" du milieu ;
- f_0 est la fonction objectif à minimiser ;
- g_i et f_i sont les contraintes d'égalité et d'inégalité respectivement.

Dans le problème d'optimisation topologique, les variables de conception ρ_e représentent la distribution de matière dans le domaine de conception. Lorsque cette variable a pour valeur 1, l'élément est solide et lorsqu'elle vaut 0, il est vide.

3.1.1 Contraintes

3.1.1.1 Contraintes d'égalité

Les contraintes d'égalité comprennent, entre autres, les expressions des équations qui gouvernent le problème. Dans le cas considéré, une seule équation gouvernante est nécessaire et elle est donnée par la théorie des éléments finis :

$$\underline{\underline{K}}\,\underline{T} = \underline{F} \tag{3.5}$$

où,
- $\underline{\underline{K}}$ est la matrice de conductivité globale $[N_{noeuds}, N_{noeuds}]$;
- $\underline{T}$ est le vecteur de température aux nœuds $[N_{noeuds},]$;
- $\underline{F}$ est le vecteur de charges thermiques appliquées, c'est-à-dire contenant les nœuds auxquels sont appliqués une source de chaleur $[N_{noeuds},]$.

La relation de l'Eq.3.5 est utilisée pour trouver les températures aux différents nœuds. Les charges appliquées sont fixées par l'utilisateur et vont dépendre du cas considéré. La matrice de conductivité globale quant à elle, ne dépend que du modèle. Par la méthode des éléments finis,

elle peut s'écrire comme la somme sur tous les éléments de la matrice de conductivité globale au niveau élémentaire $\underline{\underline{K}}_e$, comme suit :

$$\underline{\underline{K}} = \sum_{e=1} N_e \underline{\underline{K}}_e \tag{3.6}$$

La matrice $\underline{\underline{K}}_e$ peut être exprimée en utilisant la méthode SIMP :
La variable de conception ρ_e étant discrète, le problème est difficile à résoudre. Une manière de faciliter la résolution du problème est de relaxer celui-ci en rendant la variable continue, c'est-à-dire en acceptant les valeurs intermédiaires entre 0 et 1. Idéalement, la solution finale du problème d'optimisation devrait par contre uniquement contenir des variables discrètes, le domaine ne contenant ainsi que des zones solide et liquide afin d'être réalisable physiquement. La méthode **SIMP** ("Solid Isotropic Material with Penalization") (voir [**?**] par exemple) est utilisée pour pénaliser la variable et ainsi décourager les valeurs intermédiaires et diminuer la quantité de "matière grise" dans la pièce optimisée. En pratique, la méthode consiste à interpoler les propriétés de la matière en fonction de la variable de conception. Ainsi, la matrice $\underline{\underline{K}}_e$ peut s'écrire en fonction de la variable ρ_e de l'élément considéré et d'une matrice $\underline{\underline{K}}_e^0$, la matrice élémentaire, suivant la relation :

$$\underline{\underline{K}}_e(\rho_e) = (\mu_{min} + (1 - \mu_{min})\rho_e^p)\, \underline{\underline{K}}_e^0 \tag{3.7}$$

où, $\mu_{min} = 0.001$ est la fraction minimale de $\underline{\underline{K}}_e^0$ imposée afin d'éviter les difficultés numériques rencontrées avec une valeur nulle et p est l'exposant de pénalisation de la méthode SIMP permettant de réduire la quantité de matière grise dans le design (de pénaliser les valeurs intermédiaires de ρ_e).

La manière d'obtenir la matrice de conductivité élémentaire pour un élément rectangulaire subissant de la conduction sur son entièreté et de la convection sur sa face supérieure (voir Fig.3.3) est expliquée dans le livre [28].

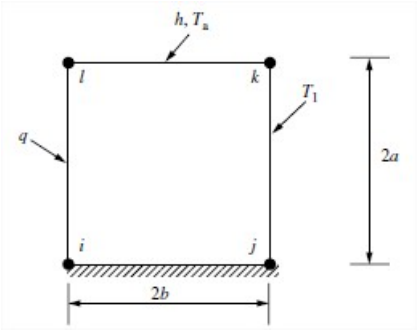


FIGURE 3.3 – Schéma d'un élément rectangulaire pour la méthode des éléments finis ([28])

Au final, la matrice est donnée par :

$$\underline{\underline{K}} = \frac{k_x a}{6b}\begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 2 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix} + \frac{k_y b}{6a}\begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 2 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix} + \frac{hl}{12}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 2 \\ 0 & 0 & 2 & 4 \end{bmatrix} \tag{3.8}$$

Lorsque, comme dans le cas considéré dans le code Matlab, les éléments sont carrés, que la conductivité thermique est considérée unitaire et que la convection n'est pas considérée du

tout, la matrice de conductivité élémentaire devient :

$$\underline{\underline{K}}_e^0 = \begin{bmatrix} 2/3 & -1/6 & -1/3 & -1/6 \\ -1/6 & 2/3 & -1/6 & -1/3 \\ -1/3 & -1/6 & 2/3 & -1/6 \\ -1/6 & -1/3 & -1/6 & 2/3 \end{bmatrix} \tag{3.9}$$

3.1.1.2 Contraintes d'inégalité

Deux contraintes d'inégalité sont appliquées au problème d'optimisation topologique. La première est la contrainte sur la variable de conception ρ_e. Cette variable est continue mais doit satisfaire les inégalités suivantes :

$$0 \leq \rho_e \leq 1 \tag{3.10}$$

En pratique, la borne inférieure est fixée à une valeur minimale ρ_{min} au lieu de 0 pour éviter tout problème numérique.

La seconde contrainte est une contrainte sur le volume. Une certaine fraction volumique $volfrac$ est fixée comme paramètre d'entrée du problème d'optimisation. En multipliant cette valeur par le nombre d'éléments constituant le domaine, on obtient un volume maximum de matière permis dans le design optimisé. Cette contrainte, en utilisant les éléments finis, s'écrit :

$$\sum_{e=1}^{N_e} v_e \rho_e \leq V_{max} \tag{3.11}$$

Où v_e est le volume d'un élément et est considéré égal à 1 dans le code.

3.1.2 Fonction objectif

Par analogie avec le cas de mécanique des structures de base traité par le code, la fonction objectif est la compliance thermique, aussi nommée "dissipation d'entransie" dans la littérature.

3.1.2.1 L'entropie

Pour la fonction objectif, l'idée principale est d'utiliser une propriété du système qui n'est pas conservée lors des transformations menant à l'équilibre. Ces propriétés sont alors des mesures des irréversibilités du processus.

La première propriété qui a été considérée est l'entropie :
Le principe de production minimale d'entropie a été développé par Prigogine et exprime le fait que la génération d'entropie dans un système thermique à l'état stationnaire devrait être minimale. Ce principe peut s'énoncer ([9]) "un état stationnaire possède le taux de production d'entropie minimale par rapport à d'autres états avec les mêmes conditions aux limites" et est souvent utilisé comme critère pour l'optimisation de systèmes thermiques. Cependant, quelques doutes planent sur le domaine de validité comme exprimé dans l'article [47].

Une autre propriété qui pourrait être utilisée comme mesure d'irréversibilités est l'exergie. Cette propriété permet de mesurer la qualité de l'énergie en jeu dans dans un processus et donc de comparer différentes sources de chaleur. Son expression fait intervenir les deux principes de la thermodynamique. Selon [35], "l'exergie du fluide correspond, au signe près, au travail maximal que l'on peut techniquement et théoriquement retirer de ce fluide qui évolue réversiblement entre un état quelconque et son état d'équilibre avec le milieu ambiant".

Alors que l'entropie d'un système augmente lors d'une transformation, l'exergie elle, diminue.

3.1.2.2 Concept d'entransie

L'entransie est une nouvelle quantité physique introduite par Z.-Y. Guo et son équipe dans l'article [30] publié en 2007. Selon eux, "plusieurs quantités existent pour décrire le taux de transfert de chaleur mais il n'existe aucun concept d'efficacité pour les processus de transfert car, dans les problèmes de transfert de chaleur, l'entrée et la sortie ont des unités différentes". Cela a pour résultat qu'"un processus de transfert de chaleur peut être amélioré mais il n'y a aucun moyen de savoir comment l'optimiser" [30]. Selon [27], "la quantité permettant d'optimiser un problème thermique va dépendre de l'objectif du transfert de chaleur. Il doit être différent selon que l'énergie thermique est convertie en travail ou qu'elle est directement utilisée pour chauffer ou refroidir un milieu. L'entransie va remplir ce rôle d'efficacité afin de pouvoir optimiser les processus de transfert de chaleur".
Les auteurs de [30] font l'analogie entre la conduction électrique et la conduction thermique en se basant sur le fait que la loi de Fourier pour la conduction thermique est analogue à la loi d'Ohm pour les circuits électriques. Dans cette analogie, le flux de chaleur et la température correspondent respectivement au courant et à la tension électriques. Il manque alors une quantité dans le système thermique qui correspondrait à l'énergie potentielle électrique d'une capacité dans le système électrique. Cette quantité est l'entransie.

Pour définir l'entransie, les auteurs de [30] considèrent un processus réversible de chauffage d'un objet à une température T et une chaleur spécifique à volume constant c_v. Pour que le chauffage soit continu, il faut un nombre infini de sources de chaleur de température de plus en plus élevée qui chauffent petit à petit l'objet. Donc, "l'énergie potentielle de l'énergie thermique augmente en parallèle à l'augmentation de l'énergie thermique lorsque la chaleur est ajoutée". L'augmentation de la partie potentielle de l'énergie thermique lorsqu'une quantité infinitésimale de chaleur est ajoutée au système peut s'écrire comme le produit de la charge thermique Q_{vh} (c'est-à-dire la chaleur stockée dans un volume constant) et de la variation infinitésimale de température (potentiel thermique) par analogie à l'énergie électrique dans une capacité.

$$\mathrm{d}E_{vh} = Q_{vh}\mathrm{d}T \tag{3.12}$$

Dans le cas où le zéro absolu est pris comme température de référence, le potentiel énergétique de l'énergie thermique dans l'objet à la température T et dans le cas d'une chaleur spécifique constante est donné par :

$$E_{vh} = \int_0^T Q_{vh}\mathrm{d}T = \int_0^T Mc_vT\mathrm{d}T = \frac{1}{2}Mc_vT^2 \tag{3.13}$$

Selon les auteurs toujours, "un objet peut être vu comme une capacité thermique qui stocke de la chaleur (charge thermique) et l'énergie potentielle thermique". Cette dernière est appelée entransie et est "une indication de la nature de l'énergie et de la capacité de transfert de chaleur du processus". Il est important de remarquer que l'entransie E_{vh} n'est pas une énergie à proprement parler car cette quantité s'exprime en $[J \cdot K]$.

Dans le cas de conduction thermique sans source de chaleur, l'équation de conservation de l'énergie thermique s'exprime :

$$\rho c_v \frac{\partial T}{\partial t} = -\nabla \cdot \dot{q} = \nabla \cdot (k\nabla T) \tag{3.14}$$

En multipliant cette dernière par T, on obtient l'équation de conservation de l'entransie :

$$T\rho c_v \frac{\partial T}{\partial t} = -\nabla \cdot (\dot{q}T) + \dot{q} \cdot \nabla T \tag{3.15}$$

Le terme de gauche représente la variation temporelle d'entransie stockée par unité de volume et les termes de droite représentent respectivement le transfert d'entransie associé au transfert de chaleur et la dissipation d'entransie par unité de temps et de volume. Selon [27], "l'entransie est donc dissipée avec le transfert de chaleur d'une zone à basse température vers une zone à haute température".
Le dernier terme de l'Eq.3.15 peut s'écrire en fonction de la conductivité thermique k et du gradient de température comme suit :

$$\Phi_h = -\dot{q} \cdot \nabla T = k(\nabla T)^2 \tag{3.16}$$

Encore selon [27], "lorsque le but du transfert thermique est de produire un travail, la génération d'entropie est une bonne mesure d'irréversibilité qui peut être utilisée lors de l'optimisation. Lorsque le but est simplement de réchauffer ou de refroidir un milieu par contre, la dissipation d'entransie est une mesure plus indiquée".

La dissipation d'entransie va pouvoir être utilisée comme une mesure de l'irréversibilité des transferts thermiques, c'est-à-dire de leur efficacité et donc comme fonction objectif à minimiser lors d'un processus d'optimisation, prenant ainsi la place de l'entropie ou de l'exergie.

La fonction objectif à minimiser du problème d'optimisation s'exprime :

$$C = \frac{1}{2} \int_\Omega k \left(\nabla T\right)^2 \, \mathrm{d}\Omega \tag{3.17}$$

Lorsque cette expression est discrétisée par éléments finis et en utilisant l'expression de l'Eq.3.7, elle peut être ré-écrite :

$$\frac{1}{2} \underline{T}^T \, \underline{\underline{K}} \, \underline{T} = \frac{1}{2} \sum_{e=1}^{N} \underline{T}_e^T \, \underline{\underline{K}}_e \, \underline{T}_e = \frac{1}{2} \sum_{e=1}^{N} \left(0.001 + 0.999 \rho_e^p\right) \underline{T_e}^T \, \underline{\underline{K}}_e^0 \, \underline{T}_e \tag{3.18}$$

3.1.2.3 Principes d'extremum de dissipation d'entransie

A partir du concept d'entransie et surtout de celui de la dissipation d'entransie, les auteurs de [30] définissent deux principes à appliquer pour la résolution du problème d'optimisation. Le principe choisi va dépendre du type de conditions aux limites imposées au problème.

Dissipation d'entransie minimale : "lorsque les conditions aux limites de flux sont données, la différence de température entre le point le plus chaud et le point le plus froid du domaine est minimale lorsque la dissipation d'entransie est minimale également" [30].

Dissipation d'entransie maximale : "lorsque les conditions aux limites portent sur les températures, le flux de chaleur échangé est maximal lorsque la dissipation d'entransie est maximale également"[30].

3.1.2.4 Résistance équivalente thermique

Les auteurs de [30] expriment également la résistance thermique, qui est définie comme "le rapport entre la différence de température et le flux de chaleur, à partir de la dissipation d'entransie et ce, encore une fois, par analogie à la conduction électrique". Comme la dissipation

d'énergie électrique par unité de temps s'exprime en fonction de la résistance, la dissipation d'entransie par unité de temps pour de la conduction de chaleur unidimensionnelle peut s'écrire :

$$\dot{E}_{vh\phi} = \int_V \phi_h \, \mathrm{d}V = \dot{Q}_h^2 R_h = \frac{(\Delta T)^2}{R_h} \tag{3.19}$$

$$R_h = \frac{(\Delta T)^2}{\dot{E}_{vh\phi}} = \frac{\dot{E}_{vh\phi}}{\dot{Q}_h^2} \tag{3.20}$$

Au vu de ces relations, "le minimum de dissipation d'entransie lorsque le flux de chaleur est fixé et le maximum de dissipation d'entransie lorsque ce sont des conditions sur les températures qui sont fixées, correspondent tous les deux au minimum de la résistance thermique équivalente. Les principes d'extremum présentés plus haut peuvent être combinés sous le **principe du minimum de la résistance thermique**" [30].

Minimiser la dissipation d'entransie revient donc à minimiser la température moyenne du domaine de conception.

3.1.2.5 Controverse sur l'entransie

Le concept d'entransie et de dissipation d'entransie est toujours en développement et ne fait pas l'unanimité parmi les scientifiques. Cette propriété est bien acceptée par certains scientifique mais complètement rejetée par d'autres. Les critiques les plus virulentes proviennent sans aucun doute de A. Bejan et son équipe qui ont publié plusieurs articles dont [8] dans lequel ils expliquent les raisons de leurs doutes. Leur acharnement sur le sujet s'explique par le fait qu'ils accusent l'équipe de [30] d'avoir plagié un de leur précédent article. Dans [8] ils accusent les auteurs de [30] de jouer deux tours aux lecteurs. Le premier est que dans la présentation de l'entransie, ils utilisent l'expression suivante pour l'énergie thermique :

$$Q_{vh} = Mc_vT \tag{3.21}$$

Selon [8], cette équation est une "fabrication qui n'appartient pas à la thermodynamique". Ils justifient leur choix en expliquant que "cette égalité provient de l'expression de la variation de l'énergie d'un système en l'absence toute forme macroscopique de variation d'énergie et sur un intervalle de température assez étroit pour que la chaleur spécifique à volume constant soit constante", soit :

$$U - U_0 = Mc_v \left(T - T_0\right) \tag{3.22}$$

Les deux équations 3.21 et 3.22 sont identiques lorsque le température de référence est fixée à 0 $[K]$. Or, si cela est bien le cas, l'intervalle de température $T - T_0$ devient trop grand que pour vérifier l'hypothèse de l'indépendance de la chaleur spécifique vis à vis de la température.

Le deuxième tour est que le principe "n'ajouterait rien à ce qui est déjà connu et utilisé pour le moment". Selon [8] toujours, "l'entransie n'est pas une quantité physique et la dissipation d'entransie ne serait qu'un multiple de la génération d'entropie". Ils reprochent à l'article sur l'entransie le manque d'explications sur la signification physique des différentes équations. Le principe de minimisation de la génération d'entransie n'est donc, selon eux, pas une nouvelle idée. Les auteurs de [30] ne feraient alors que "publier une idée existante en la faisant passer pour nouvelle". Les auteurs ajoutent que le problème d'aujourd'hui est que "tout le monde écrit et personne ne lit".

A. Bejan n'est pas le seul à critiquer l'entransie, H. Herwig [31] estime que "l'exergie est la quantité non-conservée dans un processus de transfert de chaleur et que par conséquent l'entransie n'est pas vraiment utile". Sekulic [19] lui, écrit que "l'entransie ne peut être utilisée que

dans un domaine très étroit, c'est-à-dire lors d'un transfert de chaleur sans conversion chaleur-travail". Il estime également que "contrairement à la génération d'entropie et à la destruction d'exergie qui s'appuient sur le second principe de la thermodynamique, l'entransie n'est pas universellement applicable car elle provient d'une analogie".

Enfin M. Kostic [33] est plus neutre et affirme que "l'entransie est bien liée à l'exergie et à l'énergie libre mais que celles-ci sont elles-mêmes liées à l'entropie. Ces 4 notions ont des unités différentes mais aussi des significations physiques différentes et cela n'empêche pas l'entransie d'être utile en optimisation thermique". Il ajoute que "certaines critiques qui ont été émises sont subjectives et incomplètes, et ne cherchent pas à comprendre cette nouvelle quantité avant de la rejeter".

Chacun se fait donc sa propre idée sur l'entransie et deux clans se sont formés dans la littérature et le monde scientifique. Le concept n'est peut être pas entièrement correct mais les critiques sont peut-être parfois rapides et pas entièrement impartiales. Il faut rester prudent et suivre les recherches sur cette quantité sans toutefois la rejeter catégoriquement.

3.1.3 Problème final

Le problème d'optimisation peut donc, au final, s'écrire :

$$\begin{aligned}
\text{Trouver :} \quad & \underline{x} = (\rho_1, \rho_2, ..., \rho_{N_e}) && (3.23)\\
\text{Minimiser :} \quad & C(\underline{x}) && (3.24)\\
\text{Sujet à :} \quad & \underline{\underline{K}}\ (\underline{x}))\ \underline{T} = \underline{F} && (3.25)\\
& \sum_{e=1}^{N} \rho_e V_e \leq V_{max} && (3.26)\\
& 0 \leq \rho_{e,min} \leq \rho_e \leq 1 && (3.27)
\end{aligned}$$

3.2 Description de l'outil Matlab

Le processus d'optimisation est implémenté dans un code Matlab disponible dans l'annexe A.

3.2.1 Processus d'optimisation

Le processus d'optimisation sert à mettre à jour les variables de conception d'une itération à l'autre dans l'algorithme.

Dans le code Matlab utilisé, le processus d'optimisation est le critère d'optimalité. Ce critère est dérivé des conditions d'optimalité de Karush–Kuhn–Tucker (KKT) qui sont des conditions nécessaires que doivent vérifier une solution d'un problème pour que cette solution soit optimale. Cependant, les relations de mise à jour sont en réalité des relations heuristiques qui s'écrivent [42] :

$$\rho_e^{(k)} = \begin{cases} 0 & \text{si} \quad \rho_e^{(k-1)}(B_e)^\eta \leq 0 \\ \rho_e^{(k-1)}(B_e)^\eta & \text{si} \quad 0 < \rho_e^{(k-1)}(B_e)^\eta < \rho_{max} \\ \rho_{max} & \text{si} \quad \rho_{max} \leq \rho_e^{(k-1)}(B_e)^\eta \end{cases}$$

où,

$$B_e = -\frac{\frac{\partial f_0}{\partial \rho_e}}{\lambda \frac{\partial f_1}{\partial \rho_e}} \tag{3.28}$$

Le critère d'optimalité tel qu'exprimé ici n'admet qu'une seule contrainte f_1 correspondant à la contrainte sur le volume décrite dans la section 3.1.1.2. Le terme B_e intervenant dans le critère d'optimalité et exprimé selon l'Eq.3.28 dépend des valeurs des dérivées de la fonction objectif et de la contrainte à l'itération en cours ainsi que d'un terme λ qui est déterminé par la méthode de la bisection en se basant sur la violation ou non de la contrainte sur le volume.

Bien que le critère d'optimalité soit utilisé de base dans le code Matlab, d'autres méthodes d'optimisation existent. Une méthode qui est souvent utilisée est la méthode **MMA** ("Method of moving asymptotes").
Cette méthode sert en particulier pour l'optimisation de systèmes non linéaires et son but est d'approximer les fonctions intervenant dans le problème d'optimisation et de résoudre des sous-problèmes convexes dans le but d'assurer l'obtention d'une réponse unique et optimale à chaque itération. Les fonctions sont linéarisées par rapport à des variables intermédiaires dépendantes du signe de la dérivée et l'expression de l'approximation est la suivante :

$$\tilde{f}_i^{(k)}(x) = f_i(x^{(k)}) + \sum_{j=1}^{n} \left(\frac{p_{ij}^{(k)}}{u_j^{(k)} - x_j} + \frac{q_{ij}^{(k)}}{x_j - l_j^{(k)}} \right) - \sum_{j=1}^{n} \left(\frac{p_{ij}^{(k)}}{u_j^{(k)} - x_j^{(k)}} + \frac{q_{ij}^{(k)}}{x_j^{(k)} - l_j^{(k)}} \right) \tag{3.29}$$

où les variables intermédiaires sont données par :

$$p_{ij}^{(k)} = \left(u_j^{(k)} - x_j^{(k)}\right)^2 \max\left\{0, \frac{\partial f_i}{\partial x_j}\left(x^{(k)}\right)\right\} \quad \text{et} \quad q_{ij}^{(k)} = \left(x_j^{(k)} - l_j^{(k)}\right)^2 \max\left\{0, -\frac{\partial f_i}{\partial x_j}\left(x^{(k)}\right)\right\} \tag{3.30}$$

Le degré de convexité de l'approximation dépend du taux de convergence. L'historique de la convergence du problème est donc nécessaire pour l'utilisation de cette méthode. Dans l'expression 3.29, les u_j et l_j sont les asymptotes du sous-problème convexe qui sont des limites sur le domaine dans la recherche de l'optimum.

Une implémentation de cette méthode sur Matlab est disponible et est expliquée dans l'article de K. Svanberg [44].

Il existe beaucoup de variante à la méthode MMA, une de celle-ci, qui a déjà été mentionnée dans l'état de l'art (2.3), est la méthode GCMMA ("Globally Convergent MMA"). Cette méthode permet de pallier le problème survenant lorsque la dérivée oscille autour de zéro en faisant intervenir les deux variables intermédiaires systématiquement. La recherche de l'optimum se fait alors sur un sous-domaine complètement borné.

3.2.2 Critère d'arrêt

Afin que le processus d'optimisation ne tourne pas indéfiniment, il faut définir un critère d'arrêt. Comme il sera mentionné plus loin, différents critères d'arrêt sont possibles et ne mènent pas forcément au même résultat (voir 3.3.4). Dans le cas considéré ici, le critère d'arrêt utilisé dans le code de départ pour la mécanique des structures a été conservé, c'est-à-dire :

$$\max\left(|\vec{x_k} - \vec{x_{k-1}}|\right) \leq 0.01 \tag{3.31}$$

L'optimisation s'arrête donc lorsque la plus grande différence observée sur le domaine entre la variable de conception à deux pas de temps successifs est inférieure à un certain seuil choisi.

Au lieu d'utiliser ce premier critère d'arrêt (Eq.3.31) et faisant intervenir les variables de conception, un critère se basant sur la valeur de la fonction objectif à minimiser pourrait être envisagé et pourrait s'écrire :

$$|f_0\left(x^k\right) - f_0\left(x^{k-1}\right)| \leq \epsilon \tag{3.32}$$

Le processus d'optimisation s'arrête donc lorsque la variation de la valeur de la fonction objectif d'une itération à l'autre devient inférieure à un seuil fixé au préalable.

3.2.3 Paramètres imposés en entrée

Les paramètres imposés en entrée sont les suivants (voir [14] pour plus d'informations) :

- Les nombres d'éléments horizontaux et verticaux (n_x et n_y) qui définissent la finesse du maillage du domaine de conception. La précision du design augmente donc avec le nombre d'éléments mais le temps de convergence également.
- La fraction volumique maximale ($volfrac$) intervenant dans la contrainte sur le volume exprimée par l'Eq.3.11 et représentant le rapport entre le volume de matière présente et le volume total du domaine de conception. Une fraction volumique plus élevée permet donc des designs plus épais, avec plus de matière.
- Le paramètre de pénalisation de la méthode SIMP p permettant de pénaliser les valeurs intermédiaires des variables de conception. Plus la valeur de p est élevée et plus le design est proche d'une solution "0-1". Cependant, la convergence n'est pas assurée et il est parfois nécessaire d'augmenter la pénalisation progressivement au cours du processus d'optimisation pour faciliter la convergence. La plupart du temps, néanmoins, ce paramètre sera fixé à 3.
- Le rayon du filtre (r_{min}) intervenant dans une méthode implémentée dans le code pour éviter les instabilités numériques notamment le problème de damier et pour assurer l'indépendance du design vis-à-vis du maillage. Ce filtre est appliqué à la sensibilité de la fonction objectif et effectue une moyenne pondérée de la sensibilité d'un élément à partir des sensibilités des éléments voisins se trouvant dans un rayon donné. Le rayon est exprimé par rapport à la taille d'un élément et doit donc être strictement supérieur à 1 pour que les éléments voisins soient pris en compte, le choix de sa valeur se fait par essais-erreurs.

3.3 Résultats

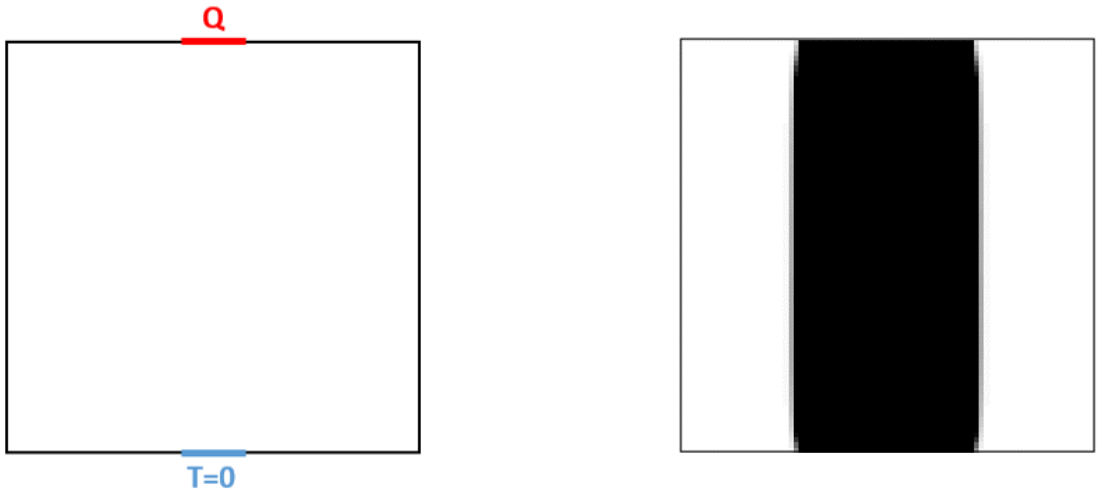
Le cas présenté dans [14] pour la conduction pure ne représente pas forcément un cas intéressant pour l'entreprise concernée par ce travail. Pour cette raison, les conditions aux limites du code de départ ont été modifiées afin de représenter des cas plus physiques bien que seule la conduction est considérée, rendant l'étude très simpliste.

3.3.1 Différents cas étudiés

Après discussion avec les membres de l'équipe sur les conditions aux limites qui pourraient être les plus représentatives de cas réellement applicables dans l'entreprise, différents cas ont

été étudiés et sont présentés dans cette section.

La première modification des conditions aux limites donne le cas présenté à la Fig.3.4a (cas 2) soit une source de chaleur centrée sur la face supérieure du domaine et un dissipateur de chaleur centré sur la face inférieure. Le design obtenu (Fig.3.4b) est simplement une "poutre" de matière reliant la source de chaleur et le dissipateur. Ce design confirme ce qui pouvait être attendu. En effet, avec de la conduction pure, la façon la plus directe de refroidir une source de chaleur à l'aide d'un dissipateur de chaleur est le chemin le plus court.



(a) Schéma des conditions aux limites (b) Design obtenu

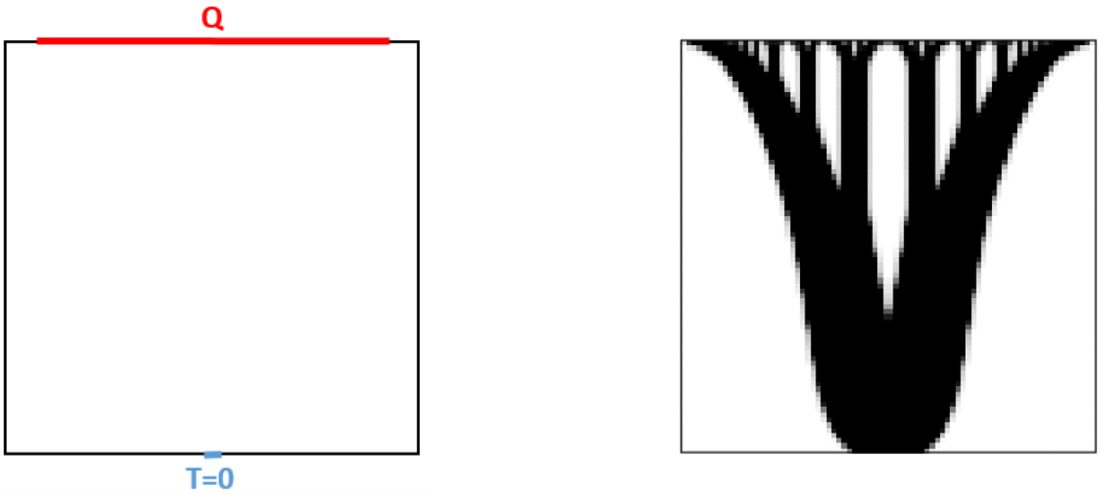
FIGURE 3.4 – Conditions aux limites et design obtenu pour le cas 2

Des cas plus spéciaux fournissent des designs plus inattendus qui pourraient permettre d'améliorer les performances du produit. Par exemple, pour un cas tel que celui représenté sur la Fig.3.5a (cas 3), une source de chaleur étendue sur presque l'entièreté de la face supérieure et un dissipateur de chaleur ponctuel centré sur la face inférieure, le design obtenu après optimisation topologique est celui de la Fig.3.5b. La quantité de matière étant limitée, celle-ci est distribuée en branches d'épaisseur variable et le design est symétrique. L'allure en forme de "V" de l'enveloppe extérieure du design semble logique au vu des conditions aux limites imposées mais les branches présentes au centre ne sont pas attendues.

Enfin, le cas de la Fig.3.6a (cas 4) est un cas plus intéressant car il pourrait être vu comme un échangeur surfacique avec une source de chaleur distribuée sur la face nord du domaine et un dissipateur de chaleur distribué sur la face sud. Encore une fois, le design obtenu (Fig.3.6b) avec la matière répartie en branches sur un arc est inattendu.

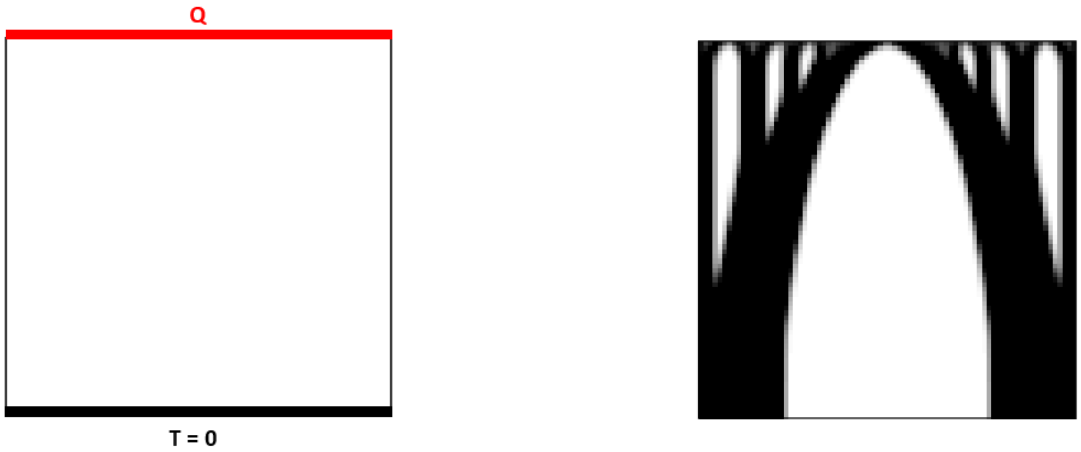
Les résultats quantifiés obtenus et les paramètres fixés en entrée pour obtenir ces résultats sont spécifiés dans la table 3.1. Les valeurs des paramètres sont choisis par essais-erreurs dans la plupart des cas pour obtenir le meilleur résultat possible avec la convergence la plus rapide possible.

La valeur de fonction objectif telle que donnée par le code est dépendante du nombre d'éléments composant le domaine. Afin de pouvoir comparer les différents cas, surtout lorsque le nombre d'éléments varie, cette valeur est divisée par le nombre total d'éléments et notée c/N_e.



(a) Schéma des conditions aux limites (b) Design obtenu

FIGURE 3.5 – Conditions aux limites et design obtenu pour le cas 3



(a) Schéma des conditions aux limites (b) Design obtenu

FIGURE 3.6 – Conditions aux limites et design obtenu pour le cas 4

Cas	volfrac	pénal	rmin	nbre itérations	c/N_e
2	0.4	1.6	1.2	38	$3.7781 \cdot 10^{-5}$
3	0.4	3	1.2	311	$3.8455 \cdot 10^{-4}$
4	0.5	3	1.6	193	$2.3077 \cdot 10^{-4}$

Tableau 3.1 – Nombre d'itérations nécessaires pour la convergence et valeur finale de la fonction objectif pour les différents cas avec $n_x = n_y = 80$

3.3.1.1 Gain de l'optimisation topologique

Il est logique de se demander ce qu'on gagne réellement à utiliser l'optimisation topologique pour trouver un design parfois complexe au lieu de se contenter de designs plus simples et familiers.

(a) Cas 3 (b) Cas 4

FIGURE 3.7 – Designs alternatifs obtenus sans l'optimisation topologique

Les designs présentés à la Fig.3.7, sont des exemples de designs moins complexes et plus classiques pour les cas 3 et 4 (le cas 2 étant déjà une droite, il n'est pas considéré ici). Ces deux designs sont construits de manière à ce que la fraction volumique de matière soit identique à celle considérée pour l'optimisation topologique et les designs présentés ne sont pas les seuls possibles.

Les informations et les résultats sont repris dans la table 3.2 qui sont à comparer avec les informations de la table 3.1. Cette comparaison montre que pour les deux cas l'optimisation topologique permet de diminuer la valeur de la fonction objectif et ce surtout pour le cas 4. Le nombre de barres composant le design du cas 4 peut être modifié pour obtenir des valeurs différentes de la fonction objectif mais, après quelques essais, la constatation est que cette valeur reste toujours de l'ordre de 10^{-3} et qu'il n'est pas évident d'obtenir la bonne fraction volumique.

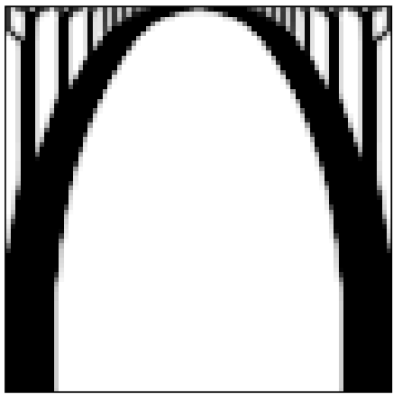
Cas	volfrac	pénal	c/N_e
3	0.4	3	$7.9570 \cdot 10^{-4}$
4	0.5	3	0.0019

Tableau 3.2 – Valeur finale de la fonction objectif pour les designs alternatifs des cas 3 et 4

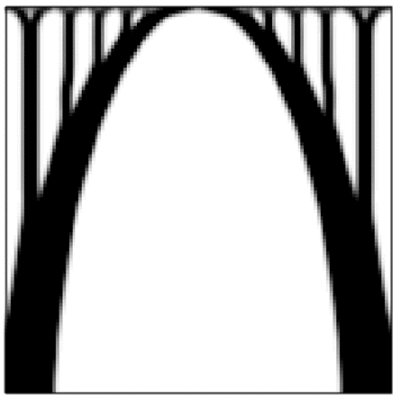
3.3.2 Influence du nombre d'éléments dans le domaine de conception

Le domaine de conception est divisé en éléments et le nombre d'éléments fixe le nombre de variables de conception. Ce nombre peut donc influencer le résultat final du processus d'optimisation. Le but de cette partie est de déterminer l'impact de ces paramètres d'entrée.
Le cas 4 étant le plus intéressant pour l'équipe, c'est sur celui-ci que l'étude des paramètres a été appliquée.

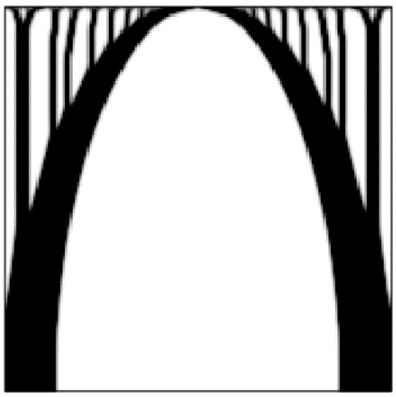
Dans un premier temps, le domaine est supposé carré et les nombres d'éléments horizontaux et verticaux sont identiques ($n_x = n_y$). Les designs obtenus et les résultats chiffrés de l'optimisation avec un nombre d'éléments progressivement augmenté sont présentés à la Fig.3.8 et à



(a) $n_x = n_y = 80$



(b) $n_x = n_y = 120$



(c) $n_x = n_y = 160$

FIGURE 3.8 – Designs obtenus pour le cas 4 en faisant varier le nombre d'éléments tel que $n_x = n_y$

la table 3.3. Pour les trois cas présentés, la fraction volumique et le paramètre de pénalisation sont respectivement fixés à 0.3 et 3. Par contre, pour des raisons de facilité de convergence, le rayon du filtre varie d'un cas à l'autre et est spécifié dans la table 3.3.

$n_x = n_y$	80	120	160
r_{min}	1.2	1.8	1.6
nbre itérations	383	563	549
c/N_e	$4.0759 \cdot 10^{-4}$	$4.1071 \cdot 10^{-4}$	$3.9629 \cdot 10^{-4}$

Tableau 3.3 – Nombre d'itérations nécessaires pour la convergence et valeur finale de la fonction objectif pour le cas 4 lorsque n_x et n_y varient ($n_x = n_y$)

Comme le montre la Fig.3.8, l'allure du design évolue quelque peu lorsque le nombre d'éléments augmente. La forme de l'arche est de plus en plus précise alors que dans un premier temps (entre les Fig.3.8a et 3.8b), le nombre de branches diminue et celles-ci s'épaississent. Cependant, dans un deuxième temps (entre les Fig.3.8b et 3.8c), les branches s'affinent et leur nombre augmente à nouveau.

Les nombres d'éléments verticaux et horizontaux ne doivent pas nécessairement être identique, le domaine de conception peut être rectangulaire ($n_x \neq n_y$). Ce cas ressemble encore plus au cas d'un échangeur de chaleur surfacique lorsque le nombre d'élément sur la hauteur de l'échangeur (n_y) reste constant et fixé à 80 alors que le nombre d'éléments sur la longueur (n_x) est augmenté d'une simulation à l'autre. C'est ce qui est réalisé pour 4 longueurs d'échangeurs et les résultats sont présentés à la Fig.3.9 et à la table 3.4. Encore une fois, les paramètres d'entrée $volfrac$ et p sont les mêmes pour tous les cas (respectivement 0.5 et 3) alors que le rayon de filtre est modifié d'un cas à l'autre de façon à faciliter la convergence du processus.

(a) $n_x = 120$

(b) $n_x = 160$

(c) $n_x = 200$

(d) $n_x = 240$

FIGURE 3.9 – Designs obtenus pour le cas 4 en faisant varier le nombre d'éléments n_x, n_y étant maintenu à 80

n_x	120	160	200	240
r_{min}	1.8	1.6	1.7	1.7
nbre itérations	157	585	335	335
c/N_e	$2.3041 \cdot 10^{-4}$	$2.2761 \cdot 10^{-4}$	$2.2986 \cdot 10^{-4}$	$2.287 \cdot 10^{-4}$

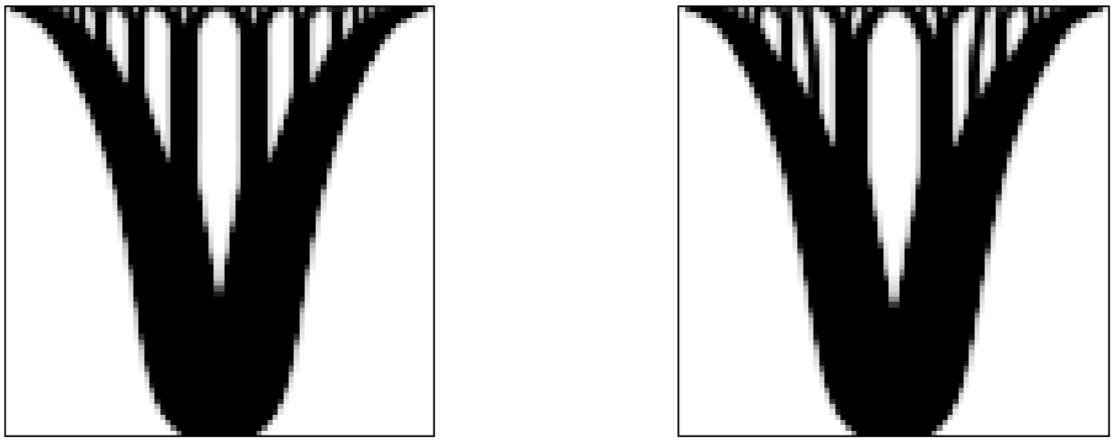
Tableau 3.4 – Nombre d'itérations nécessaires pour la convergence et valeur finale de la fonction objectif pour le cas 4 lorsque n_y varie

Les schémas de la Fig.3.9 montrent que le nombre d'arches augmente avec la longueur et que la largeur d'une arche augmente jusqu'à atteindre une certaine valeur fixe avant la création d'une nouvelle arche. Les valeurs données dans la table 3.4, quant à elles, montrent que la valeur de la fonction objectif pondérée par le nombre total d'éléments ne varie pas tellement d'un cas à l'autre. Rien ne peut être conclu à partir du nombre d'itérations nécessaire à la convergence du processus.

3.3.3 Influence du processus d'optimisation

Le processus d'optimisation choisi peut avoir une influence sur le résultat obtenu comme on peut le voir sur les Fig.3.10a à 3.11b.

Pour le cas 3, les deux premières figures montrent que les designs obtenus via les deux processus d'optimisation sont fort semblables même si les branches semblent plus droites et moins épaisses lorsque le critère d'optimalité est utilisé. De plus, les informations reprises dans la table 3.5 indiquent que la méthode MMA demande beaucoup plus d'itérations pour obtenir la convergence du résultat et que la valeur de la fonction objectif obtenue est légèrement plus élevée qu'avec le critère d'optimalité.



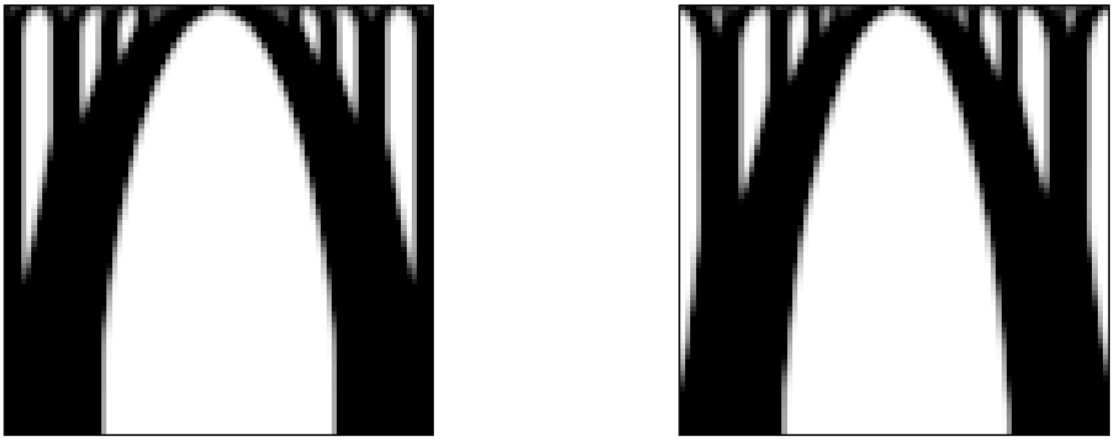
(a) Critère d'optimalité (b) MMA

FIGURE 3.10 – Designs obtenus pour le cas 3 avec deux processus d'optimisation différents

Cas		OC	MMA
3	N	311	506
	c/N_e	$3.8455 \cdot 10^{-4}$	$3.8505 \cdot 10^{-4}$
4	N	193	653
	c/N_e	$2.3077 \cdot 10^{-4}$	$2.2969 \cdot 10^{-4}$

Tableau 3.5 – Nombre d'itérations nécessaires pour la convergence et valeur finale de la fonction objectif pour les cas 3 et 4 en utilisant le critère d'optimalité ou la méthode MMA comme processus d'optimisation

Pour le cas 4, la différence entre les résultats obtenus est plus visible sur les Fig.3.11a et 3.11b que pour le cas 3. Les branches sont encore une fois plus épaisses avec le processus MMA mais elles sont aussi différemment réparties. Ce dernier processus demande aussi plus d'itérations que le critère d'optimalité mais, contrairement au cas 3, il mène à une fonction objectif de plus faible valeur comme on peut le constater dans la table 3.5.



(a) Critère d'optimalité (b) MMA

FIGURE 3.11 – Designs obtenus pour le cas 4 avec deux processus d'optimisation différents

Cette comparaison permet de conclure que le résultat obtenu dépend faiblement du processus d'optimisation pour ce qui est de la valeur de la fonction objectif et en dépend en partie pour l'allure du design obtenu bien que les différents résultats soient fort proches les uns des autres.

3.3.4 Influence du critère d'arrêt

La valeur du paramètre ϵ apparaissant dans l'expression du critère d'arrêt faisant intervenir la fonction objectif (Eq.3.32) n'est pas nécessairement la même que la valeur du seuil dans le critère de base (Eq.3.31). En effet, si ϵ est fixé à 0.01 (comme dans le cas du critère de base), le processus n'a pas le temps de converger correctement vers un design et la présence de matière grise (variables de conception de valeur intermédiaire) est importante. Par contre, si ϵ est fixé à 10^{-4}, les designs obtenus pour les cas 3 et 4 sont ceux présentés aux Fig.3.12 et 3.13 respectivement.

FIGURE 3.12 – Cas 3 avec critère d'arrêt sur f_0 et en entrée : $n_x = n_y = 80$, volfrac= 0.4, penal= 3 et rmin= 1.2

FIGURE 3.13 – Cas 4 avec critère d'arrêt sur f_0 et en entrée : $n_x = n_y = 80$, volfrac= 0.5, penal= 3 et rmin= 1.6

Le nombre d'itérations nécessaire pour obtenir la convergence du processus et la valeur finale de la fonction objectif pour les cas 3 et 4 avec le nouveau critère d'arrêt défini à l'Eq.3.32 sont repris dans la table 3.6.

cas	nbre itérations	c/N_e
3	82	$3.8706 \cdot 10^{-4}$
4	112	$2.3044 \cdot 10^{-4}$

Tableau 3.6 – Nombre d'itérations nécessaires pour la convergence et valeur finale de la fonction objectif pour les cas 3 et 4 obtenus aux Fig.3.12 et 3.13 respectivement

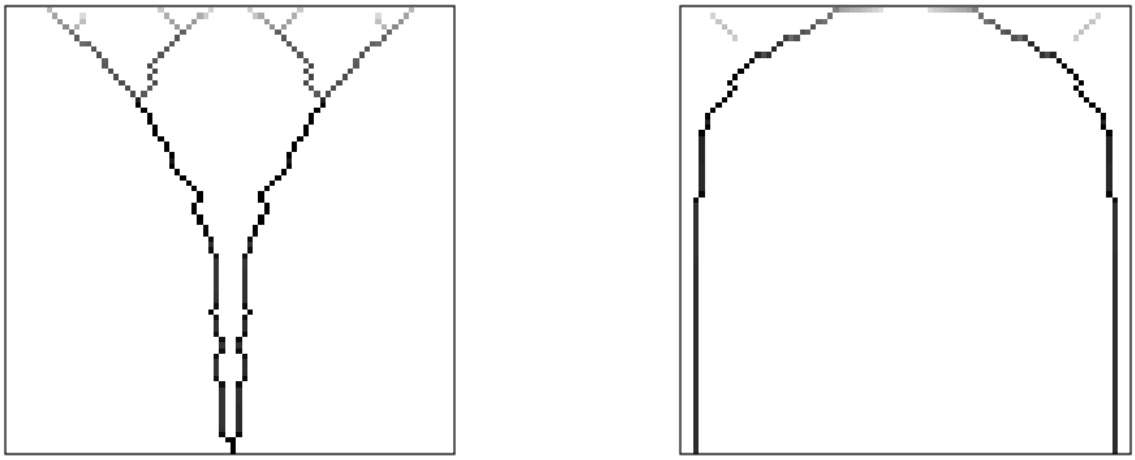
Ces résultats sont à comparer avec ceux repris dans la table 3.1 qui sont obtenus avec les mêmes conditions d'entrée mais avec le critère d'arrêt défini à l'Eq.3.31.

Dans les deux cas, le fait d'utiliser le deuxième critère d'arrêt (3.32) permet de diminuer le nombre d'itérations nécessaires à obtenir la convergence. En ce qui concerne la valeur de la fonction objectif après la dernière itération, elle ne varie que légèrement lorsqu'on passe d'un critère d'arrêt à l'autre (l'erreur relative entre les deux critères est inférieure à 1% dans les deux cas). Cependant, en fonction du cas, ce n'est pas le même critère qui mène à la valeur de fonction objectif la plus faible.
Au vu de ces constatations, le critère recommandé serait celui faisant intervenir f_0 puisqu'il permet une convergence plus rapide du processus.

3.3.5 Changement de fonction objectif

La plupart du temps, le but de l'optimisation d'un échangeur ou d'un autre produit est d'améliorer ses performances mais aussi de diminuer sa masse afin de diminuer son coût de production et d'utilisation. Il pourrait donc être envisagé de choisir le volume total de matière comme fonction objectif à minimiser plutôt que la dissipation d'entransie. La contrainte n'est

donc plus d'avoir une fraction volumique de matière maximale imposée par l'utilisateur mais plutôt un taux de dissipation d'entransie maximum permis et noté G_{max}. Après quelques essais, les designs présentés aux Fig.3.14a et 3.14b sont obtenus pour les cas 3 et 4.



(a) Cas 3 (b) Cas 4

FIGURE 3.14 – Designs obtenus pour les cas 3 et 4 avec le volume minimum comme fonction objectif et en entrée : $n_x = n_y = 80$, $G_{max} = 1$, penal= 3 et rmin= 1.6

Les designs obtenus présentent les mêmes allures qu'avec la minimisation de la dissipation d'entransie mais la quantité de matière est grandement diminuée et les designs présentent plus de matière grise.

En conclusion, il est indispensable de garder la contrainte sur le volume pour s'assurer d'avoir des designs assez épais pour être techniquement réalisables.

3.4 Conclusion

Dans ce chapitre, un code présenté dans la littérature a été utilisé et modifié pour permettre une première approche à l'optimisation topologique. Par analogie avec la mécanique des structures, le transfert de chaleur par conduction a pu être considéré et des premiers designs inattendus ont pu être générés. Ces designs, bien que trouvés dans le cadre de la conduction uniquement, pourraient déjà servir de base à une étude plus poussée sur les performances d'un nouvel échangeur de chaleur. Ainsi, même avec des hypothèses très simplificatrices, il est possible d'obtenir des résultats qui ne paraissent pas trop invraisemblables.

De plus, à partir de ce code, certains termes spécifiques à l'optimisation topologique et plus particulièrement à l'optimisation des échangeurs de chaleur, tel que le terme "entransie", ont pu être spécifiés.

Ce chapitre permet donc d'entamer l'optimisation topologique des échangeurs de chaleur en utilisant un code déjà existant et de se familiariser avec les différents concepts.

Chapitre 4

Couplage Matlab/Fluent

Dans la perspective de la prise par l'optimisation topologique du transfert de chaleur par convection et de l'écoulement du fluide dans l'échangeur, un solveur fluide est nécessaire. Le solveur n'est toutefois pas conçu pour lancer l'algorithme d'optimisation afin de mettre à jour les variables de conception. Pour cette raison, un couplage entre deux logiciels est considéré. Le but de ce chapitre est de déterminer la méthode à suivre pour réaliser ce couplage et progresser dans la réflexion pour une mise en application à venir.

A noté qu'une étude a été lancée par le professeur Boyan S. Lazarov de l'université de Manchester dans le but de tenir compte de la convection dans un code Matlab mais elle n'a pas été achevée. Selon lui, l'utilisation des volumes finis au lieu des éléments finis est nécessaire.

4.1 Description du couplage

Lors de l'optimisation, les deux logiciels sont utilisés l'un à la suite de l'autre, chacun transmettant à l'autre les informations nécessaires à son bon fonctionnement.

Le logiciel **Matlab** est utilisé pour réaliser l'algorithme d'optimisation topologique, c'est-à-dire la mise à jour des variables de conception d'une itération à l'autre jusqu'à ce que la solution converge vers un design optimal et qu'un critère d'arrêt soit vérifié. Les variables de conception sont identiques aux porosités apparaissant dans le problème d'optimisation décrit dans la section 3.1.

Cependant, dans le cas d'échangeur de chaleur, le problème est régi par le transfert de chaleur et l'écoulement du ou des fluide(s) présent(s) dans l'échangeur, et les équations correspondantes doivent être résolues à chaque itération pour pouvoir évaluer la valeur de la variable à améliorer et ainsi pouvoir optimiser le design. Ces équations interagissent entre elles et sont compliquées à résoudre et à implémenter sur Matlab. C'est pourquoi, un logiciel CFD est nécessaire dans le couplage.

Il existe plusieurs logiciels CFD qui pourraient convenir pour être couplés à Matlab, par exemple, OpenFoam ou Fluent. Le premier est un logiciel de simulation dédié à la mécanique des fluides. Il est libre d'accès et les codes sont accessibles, ce qui les rend personnalisables en fonction des besoins de l'utilisateur. Ce logiciel est cependant difficile à prendre en main et ne possède pas d'interface graphique. Le second, qui est très utilisé en entreprise, est un solveur permettant de modéliser des écoulements de fluide plus ou moins complexes à partir des volumes finis. Contrairement à OpenFoam, il possède une interface graphique et une aide

bien fournie facilitant la prise en main.

Le logiciel Fluent est utilisé comme solveur fluide chez Safran Aero Boosters. Le plus simple est donc de partir avec des logiciels déjà maitrisés dans l'entreprise de manière à perdre moins de temps à la prise en main. Il convient donc de vérifier que ce logiciel peut être intégré sans trop de difficultés à la boucle d'optimisation.

La Fig.4.1 montre le schéma du couplage envisagé entre les logiciels Matlab et Fluent afin de réaliser une optimisation topologique d'un échangeur de chaleur. Seule une partie du couplage a été réalisée lors de ce stage. Les étapes réalisées sont précédées d'un astérisque sur le schéma.

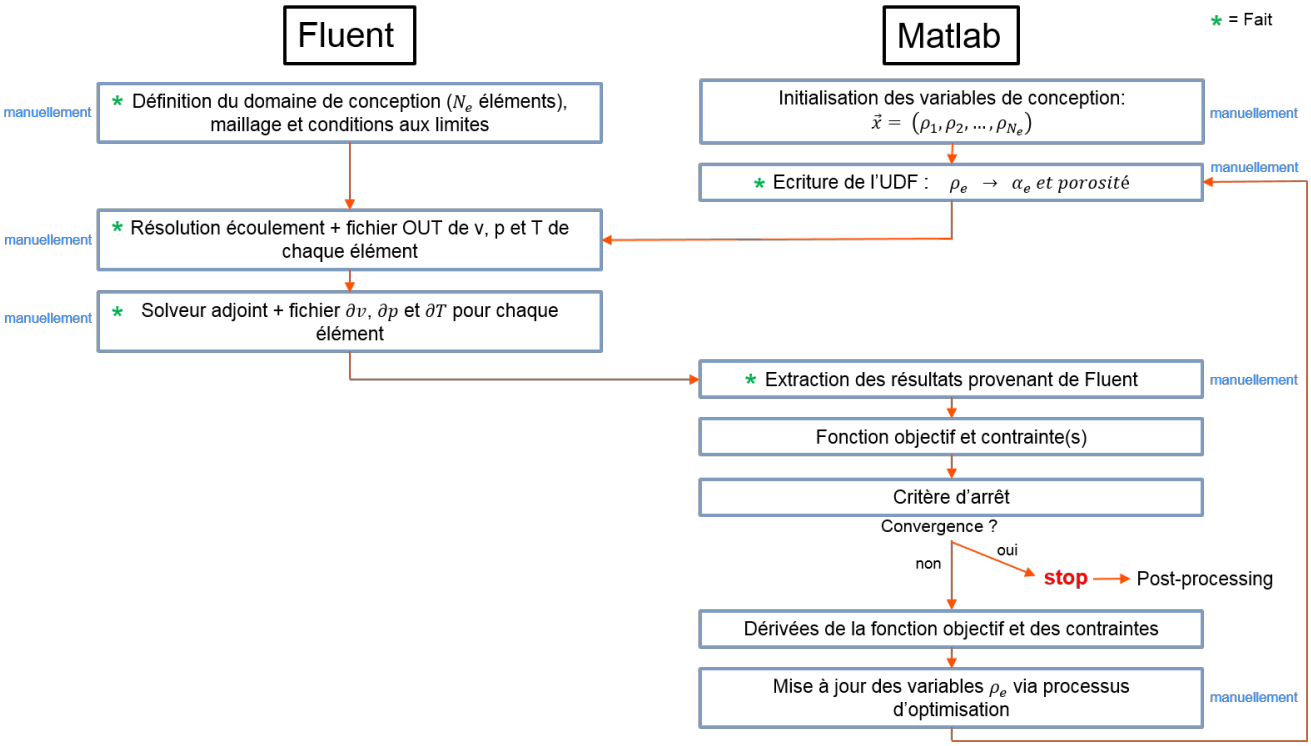


FIGURE 4.1 – Couplage Fluent-Matlab

4.1.1 Simulation de l'écoulement

Le domaine de conception est composé de zones solides et de zones fluides qui ne sont pas régies par les mêmes équations. Gérer ces zones séparément est complexe puisqu'elles varient d'une itération à l'autre. Le modèle de Brinkman peut alors être exploité pour n'utiliser qu'une seule équation de bilan de quantité de mouvement pour considérer les deux types de zones dans le domaine de conception. Selon [17], il est obtenu en combinant la loi de Darcy et l'équation régissant un écoulement de Stokes. Pour un fluide incompressible, ce modèle s'exprime :

$$\rho_f \left[\frac{\partial \underline{u}}{\partial t} + \nabla \cdot (\underline{u}\,\underline{u}) \right] - \nabla \cdot (\mu \nabla \underline{u}) = -\nabla p - \alpha \underline{u} \tag{4.1}$$

$$\nabla \cdot \underline{u} = 0 \tag{4.2}$$

Où, $\underline{u}$ est le champ de vitesse, ρ_f la masse volumique du fluide, μ la viscosité dynamique, p la pression et α un paramètre appelé l'"inverse de la perméabilité". Ce paramètre varie d'un

élément à l'autre du domaine en fonction de la variable de conception ρ_e fournie par le processus d'optimisation selon la relation de Borrvall [10] suivante :

$$\alpha(\rho_e) = \alpha_{max} + (\alpha_{min} - \alpha_{max})\rho_e \frac{1+q}{\rho_e + q} \tag{4.3}$$

Le paramètre q est le paramètre de pénalisation permettant d'obtenir des valeurs de α tendant soit vers 0 pour que l'Eq.4.1 corresponde à l'équation de quantité de mouvement régissant l'écoulement d'un fluide ou vers $+\infty$ de sorte que la vitesse soit réduite à 0 et que l'équation corresponde à celle d'un solide.

Les termes α_{max} et α_{min} apparaissant dans l'Eq.4.3, sont les bornes fixées pour la valeur de α et peuvent être déterminées comme dans l'article [10] par :

$$\alpha_{max} = \frac{2.5\mu}{0.01^2} \qquad et \qquad \alpha_{min} = \frac{2.5\mu}{100^2} \tag{4.4}$$

L'Eq.4.3 est donc l'équation qui fait le lien entre les résultats fournis par le code Matlab et les données à préciser en entrée au logiciel CFD.

4.1.2 Analyse de sensibilité pour la mise à jour des variables

La mise à jour des variables est une étape importante se faisant via un processus d'optimisation comme par exemple la méthode MMA décrite dans un chapitre antérieur. La majorité des méthodes existantes nécessitent les dérivées de la fonction objectif et de la (ou des) contrainte(s) afin de trouver la direction dans laquelle la fonction objectif diminue le plus. Les dérivées mentionnées sont les dérivées par rapport aux variables de conception, les ρ_e, qui sont très nombreuses. L'expression de ces dérivées n'est pas toujours simple et elles ne sont pas toujours possibles à écrire analytiquement.

Il existe plusieurs manières d'effectuer une analyse de sensibilité, c'est-à-dire d'estimer le gradient d'une quantité de sortie d'un problème par rapport à une ou plusieurs variables indépendantes (entrées du problème). La plus connue est la méthode des différences finies qui n'est pas particulièrement précise mais s'avère très simple à implémenter.
Les méthodes analytiques sont plus précises mais demandent plus d'informations concernant le problème et sont donc plus compliquées à implémenter. L'idée est que les fonctions gouvernant le système doivent toujours être vérifiées et que donc une perturbation des variables du problème ne devrait pas mener à une variation des résidus de ces fonctions.

En se basant sur l'article de J. Martins [36], un système gouverné par des équations d'état discrétisées est considéré. Ces équations peuvent être exprimées sous la forme de résidus s'annulant lorsque le système est à l'équilibre tels que :

$$R_l\left(x_j, y_k(x_j)\right) = 0 \tag{4.5}$$

où, x_j sont les variables de conception (avec $j = 1, \cdots n_x$) et y_k sont les variables d'état (avec $k = 1, \cdots n_y$). Le nombre total d'équations d'état étant égal au nombre de variables d'état.
Les équations gouvernant le problème devant être vérifiées pour que le système soit à l'équilibre, il en découle que toute perturbation dans les variables du système mène à une variation nulle des résidus :

$$\mathrm{d}R_l = \frac{\partial R_l}{\partial x_j}\mathrm{d}x_j + \frac{\partial R_l}{\partial y_k}\mathrm{d}y_k = 0 \tag{4.6}$$

En divisant cette dernière équation par $\mathrm{d}x_j$ on peut faire apparaitre les dérivées totales des variables d'état par rapport aux variables de conception :

$$\frac{\partial R_l}{\partial x_j} + \frac{\partial R_l}{\partial y_k}\frac{\mathrm{d}y_k}{\mathrm{d}x_j} = 0 \tag{4.7}$$

Le but de l'analyse de sensibilité est de déterminer la dérivée d'une fonction d'intérêt $f_i\left(x_j, y_k\right)$ qui s'écrit en divisant à nouveau l'expression par $\mathrm{d}x_j$:

$$\frac{\mathrm{d}f_i}{\mathrm{d}x_j} = \frac{\partial f_i}{\partial x_j} + \frac{\partial f_i}{\partial y_k}\frac{\mathrm{d}y_k}{\mathrm{d}x_j} \tag{4.8}$$

A partir de là, deux techniques, la méthode directe et la méthode adjointe, peuvent être appliquées.

Dans la **méthode directe**, la matrice jacobienne $\frac{\partial R_l}{\partial y_k}$ est factorisée en premier et son inverse est utilisée dans l'Eq.4.7 pour obtenir les dérivées totales des variables d'état par rapport à la variable de conception x_j considérée. Cette opération doit être réalisée pour chaque x_j puisque la matrice jacobienne varie d'une variable de conception à l'autre. Enfin, la sensibilité de la fonction d'intérêt s'obtient en substituant $\frac{\mathrm{d}y_k}{\mathrm{d}x_j}$ dans l'Eq.4.8.

Dans la **méthode adjointe**, une fonction Lagrangienne peut être définie pour le problème considérant les fonctions f_i soumises aux équations d'état comme contraintes. Cette fonction s'écrit :

$$\mathcal{L}\left(x_j, y_k, \lambda_k\right) = f_i\left(x_j, y_k\right) + \lambda_k^T R_l\left(x_j, y_k(x_j)\right) \tag{4.9}$$

où, λ_k est une composante du vecteur des multiplicateurs de Lagrange ou vecteur adjoint dans la cas présent.

La dérivée de la fonction Lagrangienne est égale à la dérivée de la fonction d'intérêt considérée et peut s'écrire :

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}x_j} = \frac{\mathrm{d}f_i}{\mathrm{d}x_j} = \frac{\partial f_i}{\partial x_j} + \frac{\partial f_i}{\partial y_k}\frac{\mathrm{d}y_k}{\mathrm{d}x_j} + \lambda_k^T\left[\frac{\partial R_l}{\partial x_j} + \frac{\partial R_l}{\partial y_k}\frac{\mathrm{d}y_k}{\mathrm{d}x_j}\right] \tag{4.10}$$

Le terme entre crochets est nul si le système se trouve à l'équilibre selon l'Eq.4.6. Dès lors, les valeurs des composantes du vecteur adjoint sont arbitraires.

En rassemblant les termes faisant intervenir les dérivées totales des variables d'état par rapport aux variables de conception, l'expression de la dérivée de la fonction d'intérêt devient :

$$\frac{\mathrm{d}f_i}{\mathrm{d}x_j} = \left(\frac{\partial f_i}{\partial x_j} + \lambda_k^T\frac{\partial R_l}{\partial x_j}\right) + \underbrace{\left(\frac{\partial f_i}{\partial y_k} + \lambda_k^T\frac{\partial R_l}{\partial y_k}\right)}_{A}\frac{\mathrm{d}y_k}{\mathrm{d}x_j} \tag{4.11}$$

Les dérivées $\frac{\mathrm{d}y_k}{\mathrm{d}x_j}$ qui ont été mises en évidence sont difficiles à évaluer et coûteuses en terme de calcul. Une solution est donc de choisir les composantes du vecteur adjoint de manière à annuler le terme A.

La première étape est donc de calculer pour chaque fonction d'intérêt f_i le vecteur adjoint via la relation suivante appelée équation adjointe :

$$\left[\frac{\partial R_l}{\partial y_k}\right]^T \lambda_k = -\left[\frac{\partial f_i}{\partial y_k}\right]^T \tag{4.12}$$

Cette étape, contrairement à la méthode directe, n'est réalisée qu'une seule fois pour tous les x_j puisqu'ils n'interviennent pas dans l'Eq.4.12.

Ce vecteur est ensuite utilisé dans l'Eq.4.11 qui se simplifie en :

$$\frac{\mathrm{d}f_i}{\mathrm{d}x_j} = \frac{\partial f_i}{\partial x_j} + \lambda_k^T \frac{\partial R_l}{\partial x_j} \tag{4.13}$$

Cette équation est bien l'expression de la sensibilité recherchée.

Contrairement à la méthode directe, la méthode adjointe est indépendante du nombre de variables de conception mais dépend du nombre de contraintes intervenant dans le problème d'optimisation. La méthode adjointe peut être continue ou discrète selon que la discrétisation des fonctions gouvernant le problème a lieu après l'analyse de sensibilité ou après.

Au final, une raison supplémentaire -et non négligeable- d'utiliser le logiciel ANSYS Fluent est la présence d'un solveur adjoint intégré.

4.2 Intégration du solveur

Le choix du solveur CFD se portant sur ANSYS Fluent, le projet est créé dans **ANSYS Workbench**, une plateforme permettant de faire une analyse complète d'un problème CFD. La procédure est divisée en une succession d'étapes, chacune sur un sous-programme particulier. Le schéma complet d'un projet est montré à la Fig.4.2.

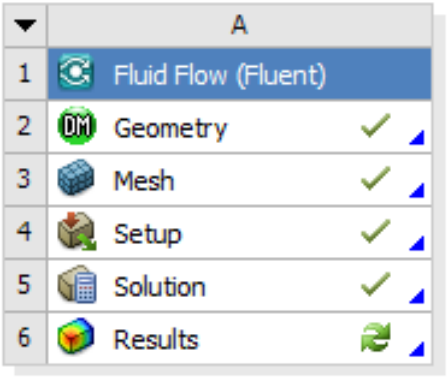


FIGURE 4.2 – Sous-programmes d'un projet dans la plateforme ANSYS Workbench

4.2.1 Équations d'état

Les équations de Navier-Stokes sont résolues dans Fluent pour simuler l'écoulement du fluide et sont exprimées aux Eq.4.14 à 4.17 selon [2].

- Équation de conservation de la masse :

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \underline{u}) = S_m \tag{4.14}$$

Où, ρ est la masse volumique du fluide dans les conditions d'écoulement et S_m un terme source représentant la masse qui pourrait s'ajouter et les sources définies par les utilisateurs.

- Équation de conservation de la quantité de mouvement :

$$\frac{\partial \rho \underline{u}}{\partial t} + \nabla \cdot (\rho \underline{u}\ \underline{u}) = -\nabla p + \nabla \cdot (\underline{\underline{\tau}}) + \rho \underline{g} + \underline{F} \tag{4.15}$$

  où le tenseur de contrainte $\underline{\underline{\tau}}$ s'écrit :

$$\underline{\underline{\tau}} = \mu \left[\left(\nabla \underline{u} + \nabla \underline{u}^T \right) - \frac{2}{3} \nabla \cdot \underline{u}\, \underline{\underline{I}} \right] \tag{4.16}$$

  Dans ces équations, p est la pression statique, $\underline{g}$ l'accélération gravitationnelle, $\underline{F}$ représente les forces extérieures, μ est la viscosité moléculaire et $\underline{\underline{I}}$ est le tenseur identité.
- Équation de conservation de l'énergie [3] :

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot (\underline{u}(\rho E + p)) = \nabla \cdot \left[k_{eff} \nabla T - \sum_j h_j \underline{J}_j + \left(\underline{\underline{\tau}}_{eff} \cdot \underline{u} \right) \right] + S_h \tag{4.17}$$

  Où, E est l'énergie totale, k_{eff} la conductivité effective, T la température, h_j l'enthalpie sensible du composant j du fluide, $\underline{J}_j$ le flux de diffusion du composant j et S_h comprend la chaleur provenant des réactions chimiques et les sources de chaleur définies par l'utilisateur.

Ces équations sont connues et ne demandent pas à être ré-expliquées ici. La seule subtilité se trouve dans l'équation de la conservation de la quantité de mouvement (Eq.4.15). Dans cette équation, le dernier terme $\underline{F}$ représente les forces extérieures et comprend également d'autres termes source dépendant du modèle comme ceux provenant du modèle de milieu poreux. En imposant correctement la valeur de $\underline{F}$ et en définissant chaque élément du domaine de conception comme une zone poreuse, il est possible d'obtenir les équations de Brinkman déjà mentionnées (Eq.4.1). Selon [5], pour un milieu poreux homogène :

$$F_i = -\left[\frac{\mu}{\beta} u_i + C_2 \frac{1}{2} \rho |u| u_i \right] \tag{4.18}$$

où le premier terme est la résistance poreuse et le deuxième la résistance d'inertie.

Dans ce cas, la résistance d'inertie est imposée nulle en fixant la constante C_2 à 0 et la résistance poreuse correspond au dernier terme de l'Eq.4.1, $\frac{\mu}{\beta}$ étant l'inverse de la perméabilité défini par l'Eq.4.3. Le paramètre à imposer dans Fluent pour chaque zone poreuse étant β, il est nécessaire de multiplier l'inverse des α obtenus à chaque itération par la viscosité dynamique μ fixée comme constante dans Fluent.

Une autre manière offerte par Fluent pour définir $\underline{F}$ est d'utiliser une loi de puissance à la place de l'équation faisant intervenir les deux termes de résistance (Eq.4.18). Le terme source s'ajoutant au bilan de quantité de mouvement s'écrit alors :

$$F_i = -C_0 |u|^{(C_1 - 1)} u_i \tag{4.19}$$

Dans ce cas, pour obtenir les équations de Brinkman, la constante C_1 est fixée à 1 et la constante C_0 est elle égale à l'inverse de la perméabilité fournie par l'Eq.4.3.

Ces deux méthodes pour faire apparaitre l'inverse de la perméabilité dans l'équation de la conservation de la quantité de mouvement devraient fournir les mêmes résultats. Les comparaisons ont été faites et l'équivalence des méthodes a bien été vérifiée.

4.2.2 Application de la méthode adjointe sur Fluent

Comme déjà mentionné dans la section 4.1.2, le logiciel Fluent contient un solveur adjoint qui peut être utilisé pour calculer les dérivées d'une quantité par rapport aux variables d'entrée.

La fonction objectif à minimiser durant l'optimisation topologique est choisie en fonction des circonstances mais elle fait très certainement intervenir la température, la pression, la vitesse ou bien même une combinaison des trois. Dès lors, l'expression de la dérivée de la fonction objectif sera composée en partie de la dérivée de ces résultats. Ce sont ces dérivées qui peuvent être calculées par la solveur adjoint de Fluent.

En fonction de la version de Fluent disponible, l'analyse de sensibilité par le solveur adjoint va pouvoir être réalisée ou non. En effet, ce n'est qu'à partir de la version **16.0** de ANSYS que le solveur adjoint est applicable à des zones poreuses.
Avec les versions précédentes de ANSYS, le solveur adjoint tourne mais en affichant un message d'avertissement indiquant que les zones poreuses sont bien actives mais qu'elles ne sont pas prises en compte par le solveur. Les zones poreuses seront considérées par le solveur adjoint comme des zones fluides non poreuses et les résultats seront de moins bonnes qualités. Une version **18.1** a cependant pu être installée et utilisée avant la fin du projet, permettant d'obtenir des résultats bien que d'autres problèmes, qui seront mentionnés ultérieurement, se soient présentés.

Selon [1], le solveur adjoint disponible dans Fluent utilise l'approche discrète de la méthode adjointe et permet de calculer la dérivée d'une fonction qui intéresse l'utilisateur par rapport aux paramètres spécifiés par lui. Les changements survenant dans les variables d'écoulement sont éliminées.

La fonction d'intérêt est un scalaire qui dépend de l'état de l'écoulement et éventuellement des variables de contrôle :

$$\mathcal{J}\left(\underline{q}^0, \underline{q}^1, \cdots, \underline{q}^{M-1}; \underline{c}\right) \tag{4.20}$$

Où, M est le nombre d'éléments dans le problème.

Cette fonction doit être différentiable par rapport aux variables dont elle dépend. Les variables de contrôle (composantes du vecteur $\underline{c}$) sont des paramètres spécifiés par l'utilisateur, c'est-à-dire des entrées du système telles que les conditions aux limites, le maillage du domaine, les propriétés du matériau ou des paramètres du modèle. Ces variables sont analogues aux variables de conception x_j intervenant dans les Eqs.4.5 à 4.13. L'état de l'écoulement $\underline{q}^\nu$ est un vecteur contenant les variables de sortie de l'élément ν du problème. Ces variables sont analogues aux variables d'état y_k dans les équations déjà mentionnées.

Les équations régissant l'écoulement (équations de conservation et autres contraintes), sont exprimées sous forme de résidus s'annulant à l'équilibre, lorsque la convergence du calcul est atteinte. Ces résidus s'écrivent :

$$\mathcal{R}_i^\mu\left(\underline{q}^0, \underline{q}^1, \cdots, \underline{q}^{M-1}; \underline{c}\right) = 0 \tag{4.21}$$

où, $\mu = 0, \cdots, M-1$ et $i = 0, \cdots, L-1$ avec L étant le nombre de conditions appliquées sur chaque élément.

En dérivant la fonction d'intérêt 4.20 et l'expression des résidus 4.21, on obtient des résultats similaires aux Eqs.4.8 et 4.6 respectivement.

$$\mathrm{d}\mathcal{R}_i^\mu = \frac{\partial \mathcal{R}_i^\mu}{\partial q_j^\mu}\mathrm{d}q_j^\mu + \frac{\partial \mathcal{R}_i^\mu}{\partial c_j}\mathrm{d}c_j = 0 \qquad (4.22) \qquad\qquad \mathrm{d}\mathcal{J} = \frac{\partial \mathcal{J}}{\partial q_j}\mathrm{d}_j + \frac{\partial \mathcal{J}}{\partial c_j}\mathrm{d}c_j \qquad (4.23)$$

Contrairement à la méthode décrite dans la section 4.1.2, les équations adjointes discrétisées ne sont pas obtenues en passant par la fonction Lagrangienne mais en prenant une combinaison linéaire pondérée de l'expression des dérivées des résidus (Eq.4.22), c'est-à-dire en multipliant tous les termes par une variable adjointe notée λ_i^μ. En identifiant alors le terme multipliant la variation des variables de sortie à celui de l'Eq.4.23, les équations adjointes permettant de définir les valeurs des variables adjointes sont obtenues et s'écrivent :

$$\frac{\partial \mathcal{R}_i^\mu}{\partial q_j^\mu}\lambda_i^\mu = \frac{\partial \mathcal{J}}{\partial q_j} \qquad (4.24)$$

Finalement, en éliminant le terme faisant intervenir les perturbations des variables de sortie dans l'Eq.4.23, on obtient l'expression suivante de la dérivée de la fonction d'intérêt en fonction uniquement des variables de contrôle.

$$\mathrm{d}\mathcal{J} = \left[\frac{\partial \mathcal{J}}{\partial c_j} - \lambda_i \frac{\partial \mathcal{R}_i}{\partial c_j}\right]\mathrm{d}c_j \qquad (4.25)$$

4.2.2.1 Possibilité de l'utilisation de la méthode adjointe sur OpenFoam

Un solveur adjoint nommé "adjointShapeOptimizationFoam" semble pouvoir être utilisé avec le logiciel OpenFoam mais peu d'informations sont réellement accessibles sur ce sujet. Ce solveur est cependant décrit dans [38]. D'autres solutions pour implémenter la méthode adjointe dans OpenFoam existe et sont expliquées dans quelques articles comme par exemple [46].

Une étude plus poussée sur le sujet et une prise en main du logiciel OpenFoam sont nécessaires avant d'envisager l'utilisation de ce logiciel dans le couplage avec Matlab.

4.2.3 Données en entrée et en sortie de Fluent

Les paramètres modifiés à chaque itération du processus d'optimisation sur Matlab peuvent être imposés comme données en entrée de Fluent via un **UDF** ("User-Defined Function").
Un UDF est un programme écrit en langage C qui accomplit une opération et renvoie le résultat lequel est alors utilisé par Fluent. Les UDF permettent d'exécuter des opérations non prévues de base dans le logiciel et de personnaliser le problème à résoudre.

Le résultat du processus d'optimisation réalisé sur Matlab est un vecteur dont les composantes sont les variables de conception (autant de variables que d'éléments dans le domaine). A partir de ce vecteur, le code Matlab va pouvoir rédiger un code en langage C, un UDF pour Fluent, qui va associer une valeur pour l'inverse de la perméabilité à chaque élément par l'Eq.4.3. L'UDF va également fixer la porosité (égal à la variable de conception) de chaque élément pour la définition des milieux poreux. Deux manières de caractériser les milieux poreux ont été évoquées à la section 4.2.1 mais il a été constaté que seuls les paramètres de l'Eq.4.18 peuvent être imposés par un UDF. Ceux de l'Eq.4.19 ne peuvent être fixés que manuellement dans le solveur. Le code Matlab générant l'UDF ainsi que qu'un exemple d'UDF sont disponibles dans l'annexe B.

Enfin, afin de limiter les problèmes, les UDF devraient être pris en compte par Fluent en étant "compilés" plutôt qu'"interprétés" (voir [4]) car ils sont alors plus stables selon le soutien

technique de ANSYS qui a pu débloquer le travail.

En ce qui concerne les sorties de Fluent nécessaires à l'algorithme implémenté dans Matlab, elles dépendent fortement de la fonction objectif et des contraintes du problème d'optimisation. Les résultats de Fluent vont en effet être utilisés pour calculer les valeurs des différentes fonctions et de leurs dérivées dans la but de permettre la mise à jour des variables et la continuation de la boucle d'optimisation. En général, pour un problème d'échangeur de chaleur, la fonction objectif est de minimiser la température moyenne sur le domaine ou de minimiser les pertes de charge entre l'entrée et la sortie de l'échangeur ou encore de combiner ces deux objectifs en leur imposant chacun un poids en fonction de la situation. Les données nécessaires sont donc la température, la pression et la vitesse moyenne de chaque élément constituant le domaine de conception. La dérivée de la fonction objectif va pouvoir être exprimée comme une combinaison des dérivées de ces variables de sortie (obtenue via le solveur adjoint) qui doivent donc également être transmises à Matlab. Le code Matlab pour extraire ces données est disponible dans l'annexe C.

Lorsque la simulation de l'écoulement sur Fluent est terminée et que les résultats ont convergé, l'historique de la convergence des résultats demandés est exporté dans un fichier lisible par Matlab. Un code Matlab est alors utilisé pour extraire les valeurs des données de sortie de Fluent, obtenues en calculant la moyenne sur les 100 dernières itérations du calcul, afin de pouvoir les utiliser. D'après ANSYS, pour s'assurer que le calcul a convergé, il faut que les résidus des équations de conservation soient au maximum égal à 10^{-3} mais en pratique, les utilisateurs cherchent plutôt à atteindre 10^{-6}. Après quelques essais, le nombre d'itérations est fixé à 500, ce qui assure des résidus assez faibles. Le nombre d'itérations étant fixe d'une itération à l'autre, le code Matlab extrayant les données ne demande pas de modification puisque les données seront toujours placées de la même façon dans les fichiers de sortie. Ce code est disponible dans l'annexe C.

Si le processus d'optimisation topologique n'a pas convergé, c'est-à-dire que le critère d'arrêt n'est pas vérifié, les variables de conception sont mises à jour et un nouvel UDF est écrit pour recommencer le cycle.

Actuellement, les lancements de Matlab et de Fluent ne se font pas automatiquement mais manuellement. L'automatisation supposerait que Matlab reconnaisse la clôture des calculs sur Fluent. Or, même si le nombre d'itérations de Fluent est fixe pour chaque itération de Matlab, le temps de calcul ne l'est pas forcément. Un moyen de détection serait par exemple de réaliser une boucle sur Matlab qui vérifierait à chaque fois si le résultat de l'itération 500 de Fluent est présent ou non (pour rappel ce résultat se retrouve toujours au même endroit dans le fichier de sortie).

4.2.4 Liaisons entre Matlab et Fluent

Le couplage entre Matlab et Fluent peut se faire via "ANSYS aaS Matlab Toolbox" qui permet de contrôler Fluent à partir de Matlab. Quant aux différentes actions à mener dans Fluent, elles peuvent être gérées à partir d'un "journal Fluent".

4.3 Résultats

4.3.1 Préparation de la résolution sur ANSYS Fluent

4.3.1.1 Définition de la géométrie

La première chose à faire est de définir un domaine de conception qui sera fixe tout au long du processus d'optimisation. Ce domaine est défini dans "**DesignModeler**" dans le plan XY et est composé de carrés de 10 mm de côté (contrainte sur les dimensions) dessinés dans une esquisse.
Chaque carré est considéré dans la suite comme un élément du domaine. Le nombre total d'éléments dans le domaine définit le nombre de variables de conception intervenant dans le problème d'optimisation implémenté sur le logiciel Matlab ; chaque élément se voit donc attribuer une variable ρ_e (porosité).
Ces éléments sont des zones indépendantes les unes des autres dans le calcul. Pour les définir comme telles, l'outil "*Surfaces à partir d'esquisses*" est utilisé en sélectionnant tous les carrés créés, en choisissant "*Ajouter un corps bloqué*". Le modèle contient alors une pièce et autant de corps que d'éléments.
C'est aussi dans ce sous-programme que les différentes parois et surfaces peuvent être nommées via "*sélection nommée*" pour simplifier les étapes ultérieures en leur donnant des noms appropriés et ainsi pouvoir les reconnaitre facilement dans les sous-programmes suivants. Comme montré sur la Fig.4.3, l'entrée et la sortie de l'écoulement fluide sont respectivement les parois de gauche et de droite du domaine (nommés "inlet" et "outlet"), la source de chaleur ("heat-gen") est distribuée sur la paroi inférieure alors que la paroi supérieure ("wallsup") est une simple paroi. Les différents éléments sont également nommés et distingués via un numéro (ils sont numérotés de haut en bas et de la gauche vers la droite comme montré sur la Fig.4.4 dans le cas d'un domaine composé de 4 éléments).

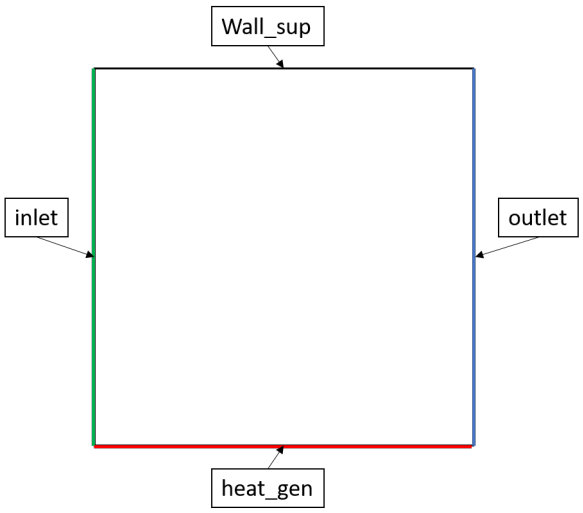


FIGURE 4.3 – Parois du domaine de conception

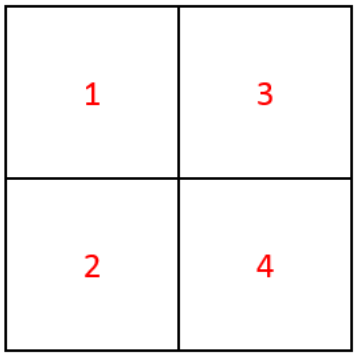


FIGURE 4.4 – Numérotation des éléments pour un domaine de 2x2 éléments

4.3.1.2 Maillage dans Meshing

La deuxième étape consiste à créer le maillage dans le sous-logiciel "**Meshing**".

Le domaine de conception étant un assemblage de carré, il est intéressant d'avoir un maillage composé uniquement de quadrilatères. Pour ce faire, il est possible d'imposer la méthode "à dominance de quadrilatères" et de rajouter une contrainte "maillage de face" pour forcer le mailleur à produire un maillage très ordonné de bonne qualité et ce sur tous les carrés.
Les intersections entre les carrés et les parois du domaine sont des endroits plus délicats et demandent plus de précisions que le centre des différents carrés puisque les contraintes d'écoulement sont différentes d'un élément à l'autre et qu'il risque d'y avoir des sauts. C'est pourquoi un dimensionnement est ajouté sur toutes les arêtes des éléments afin de fixer le nombre de divisions par arête. Les divisions ne seront pas uniformes de façon à obtenir des mailles plus larges au centre des arêtes et de plus en plus étroites lorsqu'elles se rapprochent des nœuds. Un rapport de taille est alors choisi pour fixer la différence entre la longueur de la plus grande maille et celle de la plus petite sur une arête.
Lorsque tout ceci est fixé, le maillage peut être généré.

La qualité du maillage est importante à vérifier car des éléments trop difformes pourraient mener à des erreurs. Les critères les plus importants devant être vérifiés sont :

- Le **rapport de Jacobien** qui est, selon [6], le rapport entre les valeurs maximale et minimale des déterminants des matrices Jacobiennes déterminés aux nœuds de coin des mailles. Lorsque ce rapport est négatif comme c'est le cas lorsqu'une maille est retournée sur elle-même, le maillage est inexploitable. Le rapport de Jacobien doit donc toujours être positif.
  Le rapport de jacobien est partout égal à 1 dans les maillages utilisés pour ce projet car toutes les faces opposées des mailles sont parallèles entre elles et aucune maille ne possède de nœuds au milieu d'un de ses côtés. Cela signifie que les mailles sont de forme idéale car le déterminant de la matrice jacobienne est identique pour tous les éléments.
  La Fig.4.5, présente trois mailles à rapport de Jacobien croissant. Plus le rapport de Jacobien est élevé et plus le risque que la maille ne casse augmente.
- Le **ratio de forme** qui est une mesure de l'étirement d'une maille. Ce ratio est le rapport entre la valeur maximale et la valeur minimale entre les distances normales entre le centre de la maille et les centres des faces et les distances entre le centre de la maille et les nœuds. Pour des mailles quadrilatères qui sont, en plus, rectangulaires, l'aspect ratio est le rapport entre la longueur du plus long côté sur celle du côté le plus court. L'aspect ratio d'un carré est donc 1.

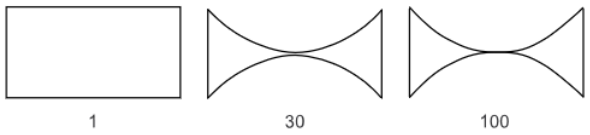


FIGURE 4.5 – Rapport jacobien [7]

Dans le cas des maillages générés lors de ce travail, ce rapport est compris entre 1 et 5, ce qui signifie que le maillage n'est pas beaucoup déformé.
Deux exemples de maille à rapport de forme différent sont montrés à la Fig.4.6. La maille de gauche à un ratio unitaire alors que la maille de droite à un ratio supérieur à 1.

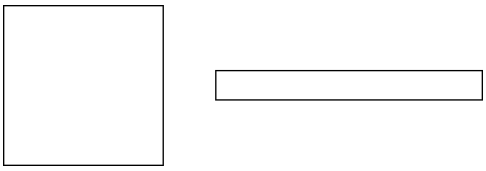


FIGURE 4.6 – Le rapport de forme

- L'**inclinaison** qui est la différence entre la forme de la maille considérée et la forme d'une maille quadrilatère équiangulaire de même volume. Pour que la qualité de la maille soit bonne, il faut que l'inclinaison soit la plus proche de 0 possible. Une maille de très mauvaise qualité a, quant à elle, une inclinaison proche de l'unité. Pour les maillages utilisés lors de ce travail, cette différence est toujours très proche de 0, ce qui correspond à une excellente qualité de maille selon les règles de bonnes pratiques de maillage. A la Fig.4.7, la maille de gauche a une inclinaison nulle puisqu'elle est quadrilatère équiangulaire alors que la maille de droite a une inclinaison plus élevée.

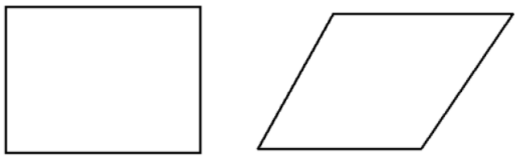


FIGURE 4.7 – L'inclinaison [6]

4.3.1.3 Initialisation du solveur Fluent

Avant de lancer le processus d'optimisation topologique, il faut initialiser le solveur. En effet, à chaque itération du processus implémenté dans Matlab, la modélisation de l'écoulement sera faite à partir des mêmes conditions aux limites et des mêmes paramètres du solveur, seul l'UDF écrit par Matlab sera modifié.

Les cas envisagés sont simples puisque ce projet n'est qu'un point de départ pour lancer l'utilisation de l'optimisation topologique via un couplage Matlab/Fluent. L'écoulement est donc considéré en régime établi et laminaire.

En ce qui concerne les matériaux en jeu dans la modélisation, l'air incompressible constitue le fluide tandis que l'aluminium a été choisi comme solide.

Pour les conditions aux limites, les choix ont été faits en se basant sur les conditions en pratique dans le cas d'un échangeur air-huile. Les conditions imposées dans le logiciel Fluent sont les suivantes, les valeurs imposées se trouvent elles dans la table 4.1.

- L'huile n'est pas représentée telle quelle mais on suppose que la paroi inférieure (Heat_gen) est le lieu de la génération de chaleur, c'est-à-dire la surface chaude à refroidir à l'aide de l'air, et qu'elle se trouve à la température estimée de l'huile. Cette paroi est définie comme un mur stationnaire et sans glissement avec une température constante.
- Pour l'entrée de l'écoulement, deux conditions doivent être fixées. La première condition est la température du fluide et la seconde est la vitesse de l'écoulement . Cette seconde condition pourrait être la pression manométrique en fonction des données fournies. La pression manométrique est une pression relative par rapport à une pression de référence qui est ici la pression atmosphérique, elle est donc égale à la différence entre la pression totale et la pression ambiante fixée à 101325 $[Pa]$ dans le logiciel.
- Pour la sortie de l'écoulement, la pression est fixée à la pression atmosphérique. Dans certains cas, un "backflow" (un écoulement rentrant dans le domaine par la sortie) est possible et sa température totale doit être fixée avec les conditions aux limites.
- La paroi supérieure est un mur stationnaire, sans glissement sans flux de chaleur ou température imposée. Cette dernière condition est formulée en imposant un flux de chaleur nul sur la surface.

	Inlet	Outlet	Wall_sup	Heat_gen
T [K]	310	300 (backflow)	-	397.5
P [Pa]	-	101325	-	-
c [2m/s]	5	-	-	-

Tableau 4.1 – Valeurs utilisées pour l'initialisation du solveur Fluent

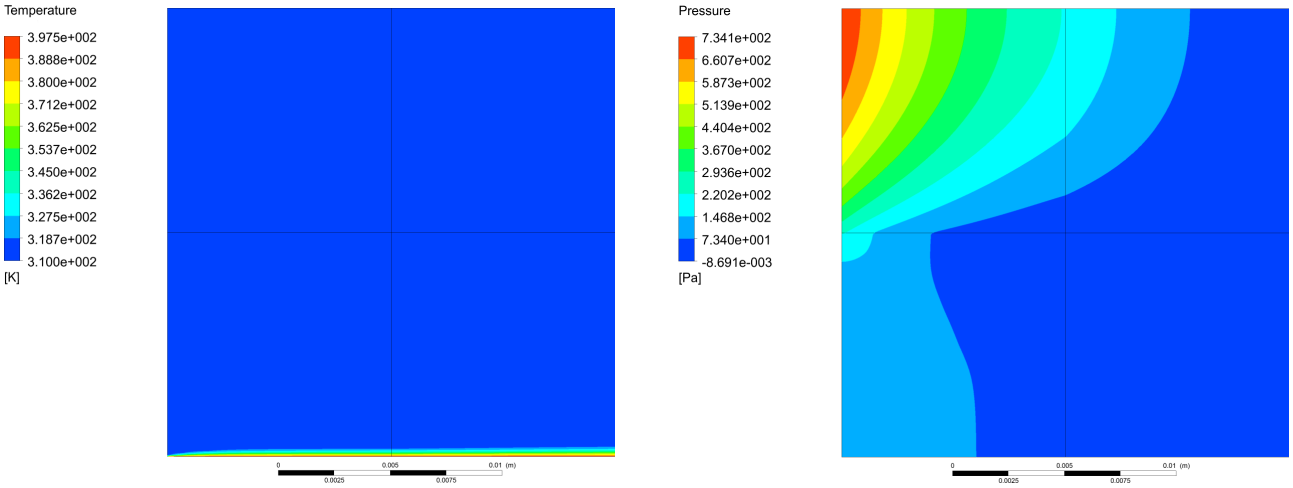
Ces 3 premières étapes (définition de la géométrie, du maillage et initialisation du solveur) permettent d'initialiser les différents sous-programmes qui seront utilisés tout au long du processus d'optimisation. Elles ne se font qu'une seule fois mais tout doit se faire manuellement. Lorsque le nombre d'éléments est faible cela ne pose pas trop de problèmes mais lorsqu'il devient important ces étapes exigent un investissement en temps extrêmement lourd.

4.3.2 Cas 2x2

Dans un premier temps, un cas simple composé d'un carré de 4 éléments comme représenté à la Fig.4.4 a été considéré pour prendre en main le logiciel et vérifier le bon fonctionnement de la méthode.

Une première simulation est faite pour un domaine de conception défini par les variables de conception présentées à la Fig.4.10d. L'écoulement est complètement libre dans les deux élé-

ments inférieurs ($\rho_e = 1$) alors qu'il passe difficilement dans les éléments supérieurs, l'écoulement étant normalement impossible dans l'élément supérieur droit caractérisé par un $\rho_e = 0$, c'est-à-dire que l'élément devrait être solide.
Les répartitions de température et de pression de la solution de cette étude sont présentées à la Fig.4.8 ; la répartition de vitesse ainsi que les lignes de courant sont représentés à la Fig.4.11d.



(a) Répartition de la température (b) Répartition de la pression

FIGURE 4.8 – Répartitions de la vitesse et de la pression pour le cas 2x2

Ce dernier graphique montre bien que l'écoulement du fluide rentre par toute la paroi gauche du domaine mais est dévié par les milieux poreux vers la partie inférieure du domaine. Une seule ligne de courant est présente dans l'élément supérieur droit, ce qui s'explique par le fait qu'un milieu poreux n'est jamais entièrement solide. Cependant, la vitesse dans cet élément est nulle ou en est très proche.

Le graphique de la température n'apporte pas grand chose d'inattendu. La température de la paroi inférieure du domaine étant fixe, le fluide s'écoule et se réchauffe sur une très faible couche limite le long de cette paroi. L'effet de la température de la plaque n'est pas ressenti par la plus grande partie de l'écoulement et la température moyenne de sortie de l'écoulement est très proche des 310 $[K]$ de l'écoulement en entrée.

Le graphique de la pression montre seulement que la pression en entrée n'est pas uniforme à cause de la différence de porosité entre les deux éléments de gauche. Elle évolue alors pour atteindre les conditions de sortie de l'écoulement (pression fixée).

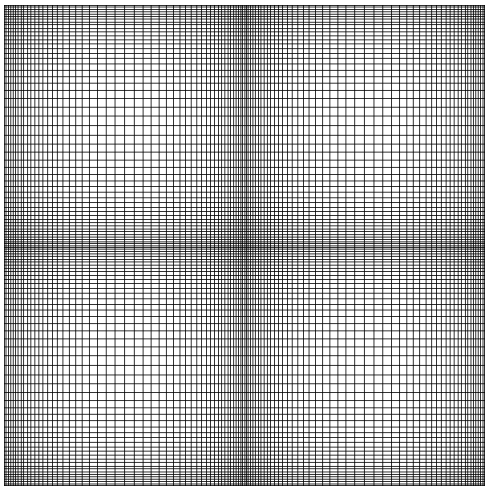
4.3.2.1 Influence du maillage

La maillage choisi pour le domaine de conception doit être assez fin pour ne pas influencer les résultats. Cependant, un maillage trop fin peut demander beaucoup d'itérations avant que les calculs ne finissent par converger vers un résultat. En pratique, il faut donc faire un compromis sur le maillage.

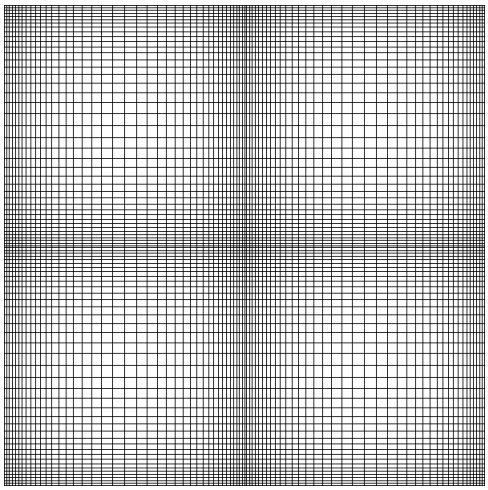
Plusieurs maillages ont été considérés pour le cas 2x2. Entre les différents maillages, seul le nombre de divisions sur les arêtes est modifié, les mailles étant toujours des quadrilatères et le rapport de taille restant fixé à 5. Les différences entre les maillages, c'est-à-dire le nombre de divisions par arête, le nombre de nœuds et le nombre de mailles, sont reprises dans la table 4.2. Les 3 maillages sont présentés à la Fig.4.9.

Maillage	nbre divisions	nbre nœuds	nbre mailles
fin	70	19881	19600
intermédiaire	50	10201	10000
grossier	20	1681	1600

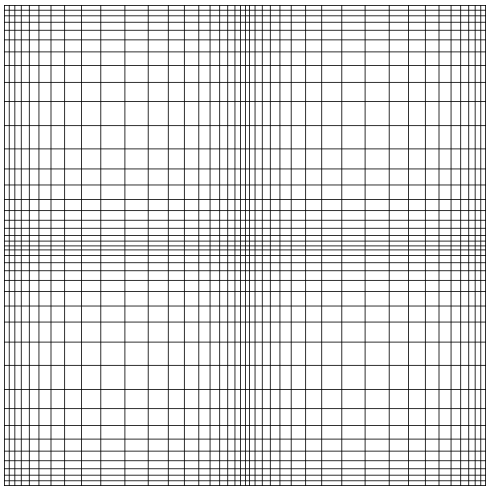
Tableau 4.2 – Informations sur les différents maillages comparés pour le cas 2x2



(a) Maillage fin



(b) Maillage intermédiaire



(c) Maillage grossier

FIGURE 4.9 – Différents maillages comparés

Avec le nombre de divisions par arête qui augmente, les séparations entre les différents éléments sont plus marquées et les résultats obtenus en ces endroits sont plus précis.

Les résultats obtenus pour les différents maillages sont repris dans la table 4.3. Les erreurs relatives commises en prenant le maillage intermédiaire ou grossier plutôt que le maillage fin sont toujours inférieures à 1% sauf dans le cas de la perte de charge moyenne avec le maillage grossier qui atteint les 2%, ce qui reste tout de même faible. Le maillage fin n'est donc pas forcément nécessaire s'il nécessite plus de temps pour la résolution de l'écoulement. Le maillage

intermédiaire suffit.

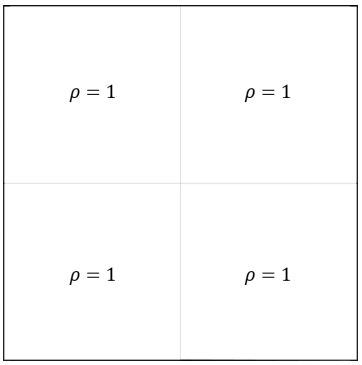
	$T_{t,out}$	ΔP_t	V_{out}
Maillage fin	311.3501	231.5712	5.0054
Maillage intermédiaire	311.3606	230.6792	5.0054
Erreur [%]	$3.37241 \cdot 10^{-3}$	-0.3852	0
Maillage grossier	311.4776	226.4119	5.0052
Erreur [%]	0.041	-2.228	$-3.99568 \cdot 10^{-3}$

Tableau 4.3 – Résultats pour les différents maillages et erreurs commises par rapport au maillage fin

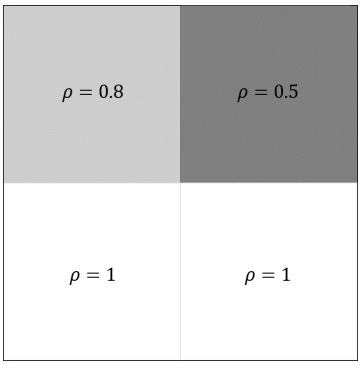
4.3.2.2 Différentes itérations (itérations successives)

Le but final de ce couplage des deux logiciels est de pouvoir résoudre automatiquement les équations de conservation à chaque itération de Matlab au cours du processus d'optimisation. Malheureusement, la partie Matlab servant à mettre à jour les variables de conception ρ_e du problème n'a pas été implémentée au cours de ce travail. Pour pouvoir observer l'effet de variation du vecteur $\underline{x}$ des variables de conception sur les résultats calculés par Fluent, il faut que l'utilisateur modifie ce vecteur manuellement dans le code Matlab générant le code en langage C de l'UDF. Ceci a été réalisé et les Fig.4.10 et 4.11 représentent respectivement le vecteur $\underline{x}$ à chaque itération et la répartition de vitesse et les lignes de courant obtenus après convergence des calculs du solveur Fluent.
La première itération représente le point de départ du processus d'optimisation topologique lorsque la totalité du domaine de conception est occupé par du fluide. Comme la Fig.4.11a le montre, la vitesse est uniforme, et son amplitude reste la même qu'en entrée, partout sur le domaine à l'exception du voisinage des parois latérales où une fine couche limite se forme.

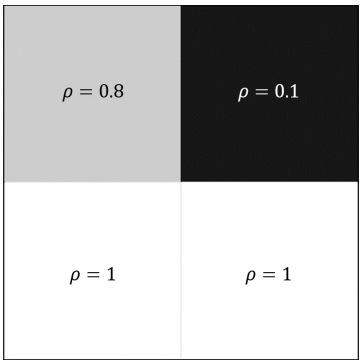
Les itérations suivantes représentent l'optimisation du domaine. La source de chaleur se trouvant sur la paroi inférieure, le fluide va être de plus en plus forcé dans les éléments inférieurs et les éléments supérieurs vont devenir de plus en plus solides. Ce constat peut être fait à partir des Fig.4.11b à 4.11d. Les lignes de vitesse évitent de plus en plus l'élément supérieur droit et la vitesse y diminue jusqu'à s'annuler sur tout l'élément. La vitesse augmente donc dans les éléments par lesquels le fluide doit passer.



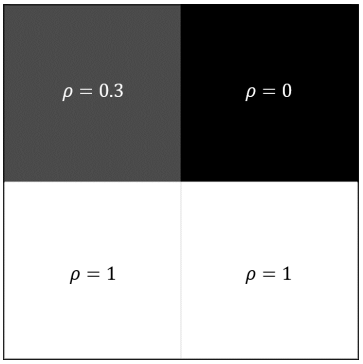
(a) Itération 1



(b) Itération 2

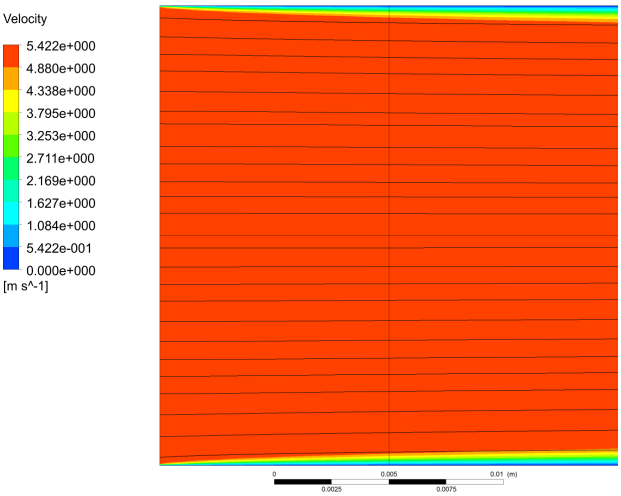


(c) Itération 3

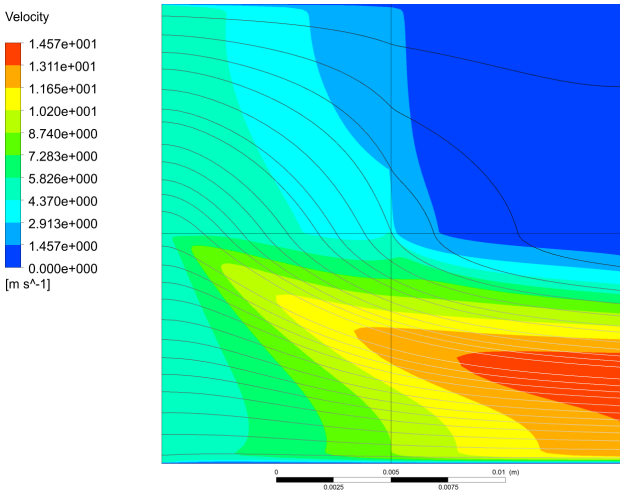


(d) Itération 4

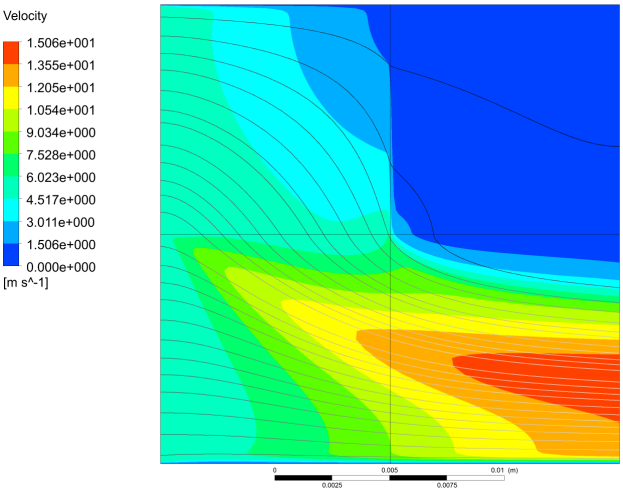
FIGURE 4.10 – Valeurs des variables de conception ρ_e du domaine 2x2 pour 4 itérations successives



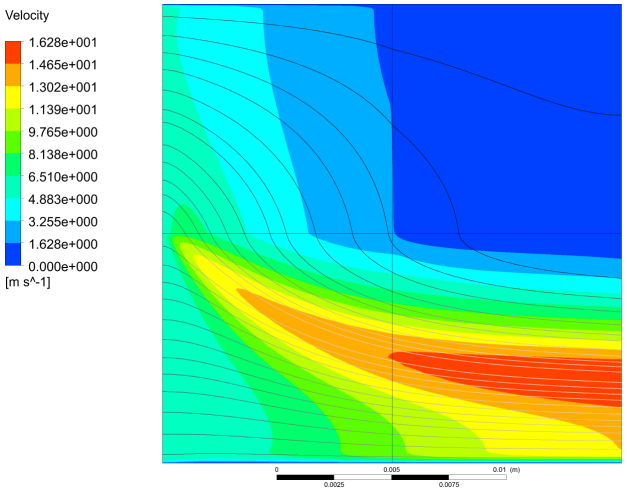
(a) Itération 1



(b) Itération 2



(c) Itération 3



(d) Itération 4

FIGURE 4.11 – Contour et ligne de vitesse du domaine 2x2 pour 4 itérations successives

4.3.2.3 Influence du paramètre q

Le choix de la valeur du paramètre de pénalisation q intervenant dans l'Eq.4.3, reliant l'inverse de la perméabilité à la variable de conception, est arbitraire.

La Fig.4.12 présente l'évolution de l'inverse de la perméabilité α en fonction de ρ_e pour différentes valeurs du paramètre de pénalisation q en utilisant l'Eq.4.3 avec, pour le calcul des bornes de α (Eq.4.4), une viscosité dynamique unitaire.

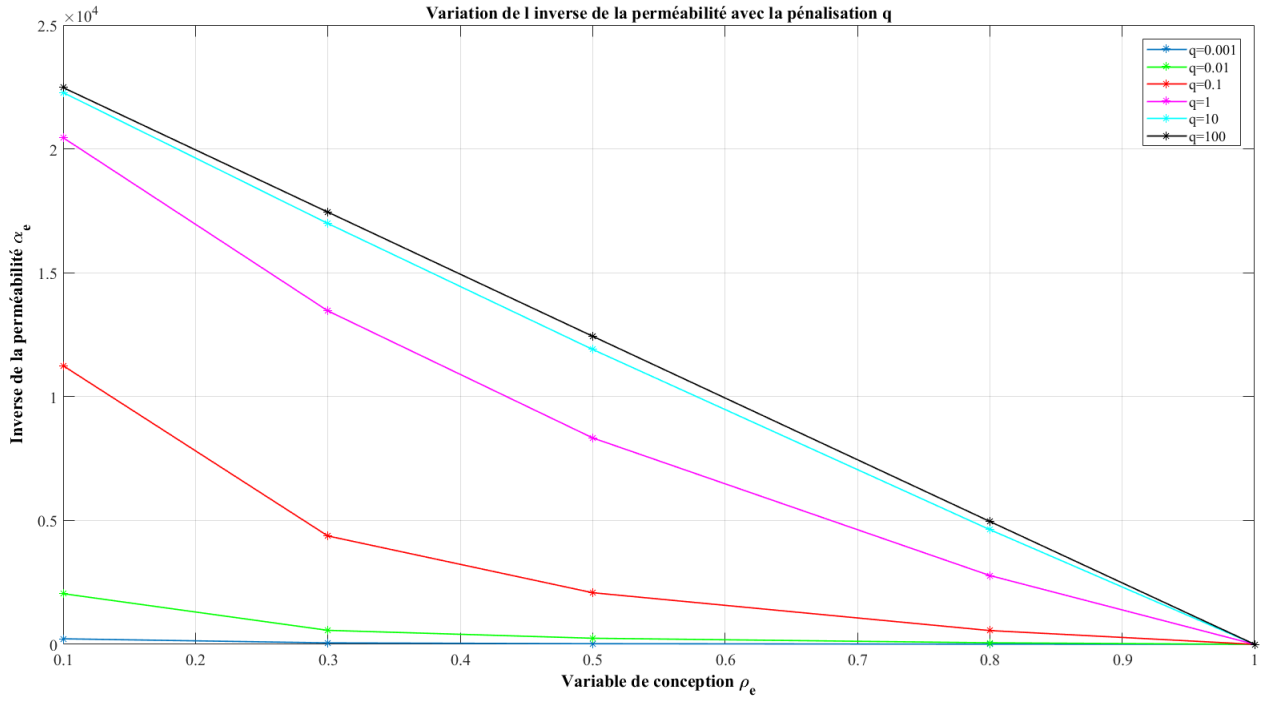
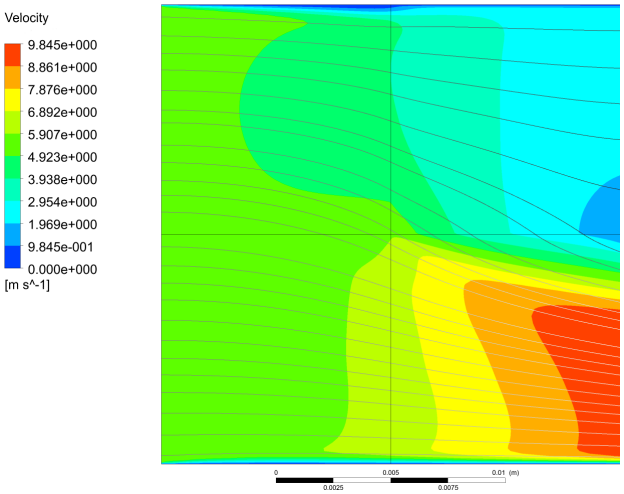


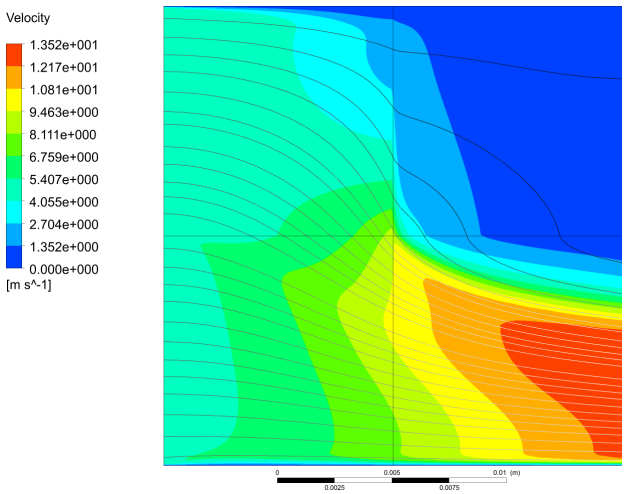
FIGURE 4.12 – Evolution de l'inverse de la perméabilité α en fonction de ρ_e pour différentes valeurs du paramètre de pénalisation q

Ce graphique montre que plus la valeur donnée à q est élevée, plus l'intervalle entre la valeur de α correspondant à du fluide ($\rho_e = 1$) et celle correspondant à du solide ($\rho_e = 0$) est grand.

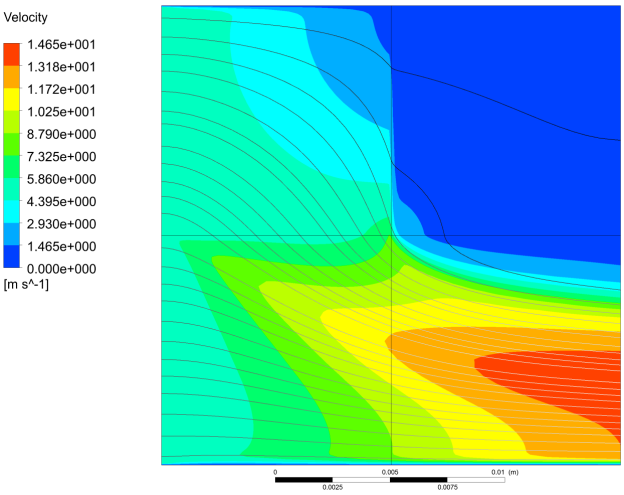
L'effet de l'augmentation du paramètre q peut se voir sur la Fig.4.13. Plus la valeur de q est élevée, plus la différence entre solide et liquide est marquée. En d'autres termes, plus la valeur de q est élevée et plus les valeurs intermédiaires des variables de conception sont pénalisées. Cependant, l'effet s'atténue avec l'augmentation de q et à partir d'une certaine valeur, il n'est plus utile de continuer à l'augmenter.



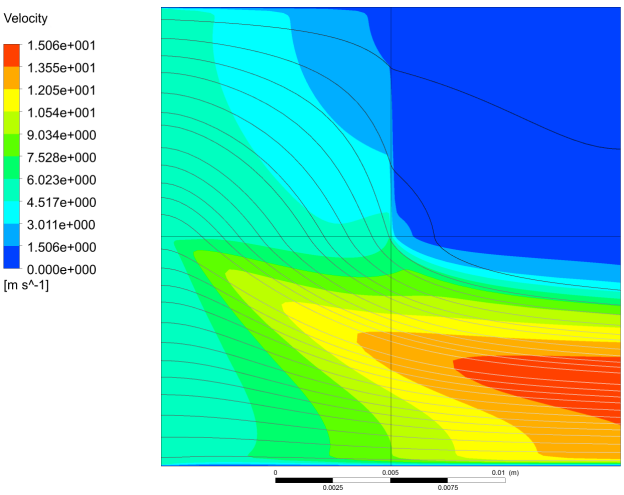
(a) $q = 0.01$



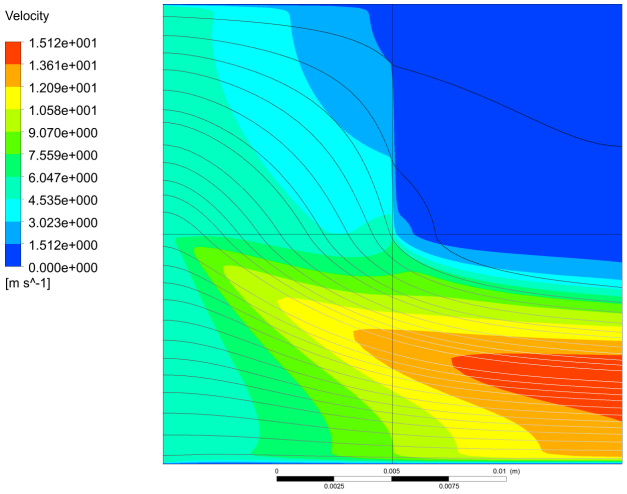
(b) $q = 0.1$



(c) $q = 1$



(d) $q = 10$



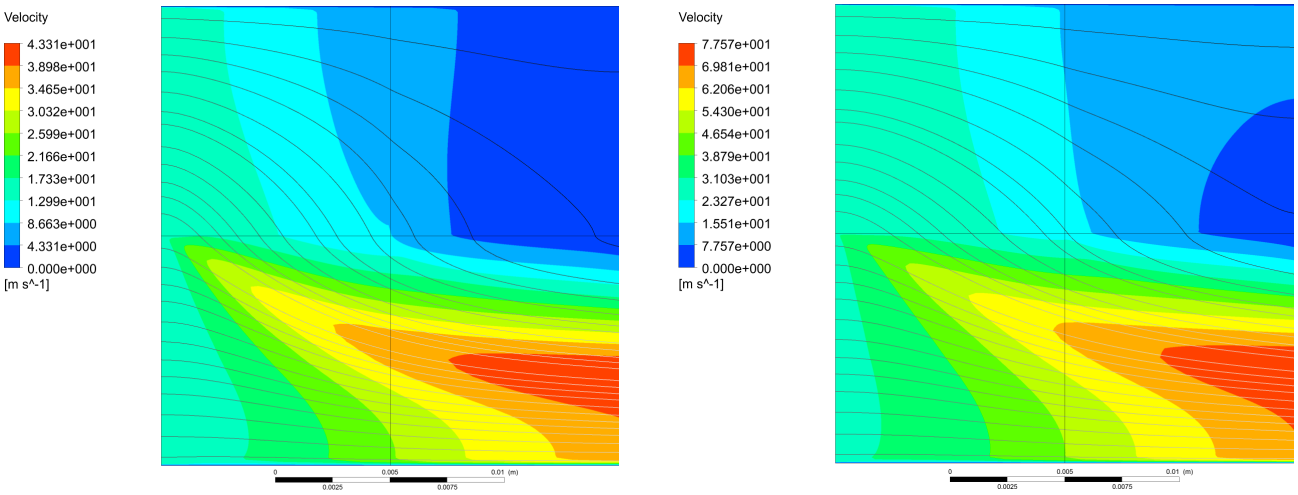
(e) $q = 100$

FIGURE 4.13 – Répartition de la vitesse et lignes de courant pour différents paramètres de pénalisation q dans le cas 2x2 avec le vecteur de variable de conception correspondant à la Fig.4.11c

4.3.2.4 Influence de la vitesse en entrée

La vitesse imposée à l'écoulement à l'entrée du domaine de conception peut également avoir une influence sur le parcourt du fluide à travers les différentes zones poreuses.

Pour un domaine dont les variables de conception sont celles de la Fig.4.10c, les répartitions de vitesse et les lignes de courant obtenues à la fin de la simulation du solveur Fluent pour trois vitesses d'écoulement différentes imposées en entrée sont montrées aux Fig.4.11c, 4.14a et 4.14b. Les vitesses étant respectivement 5, 15 et 30 $[m/s]$.



(a) $V_{in} = 15$ $[m/s]$ (b) $V_{in} = 30$ $[m/s]$

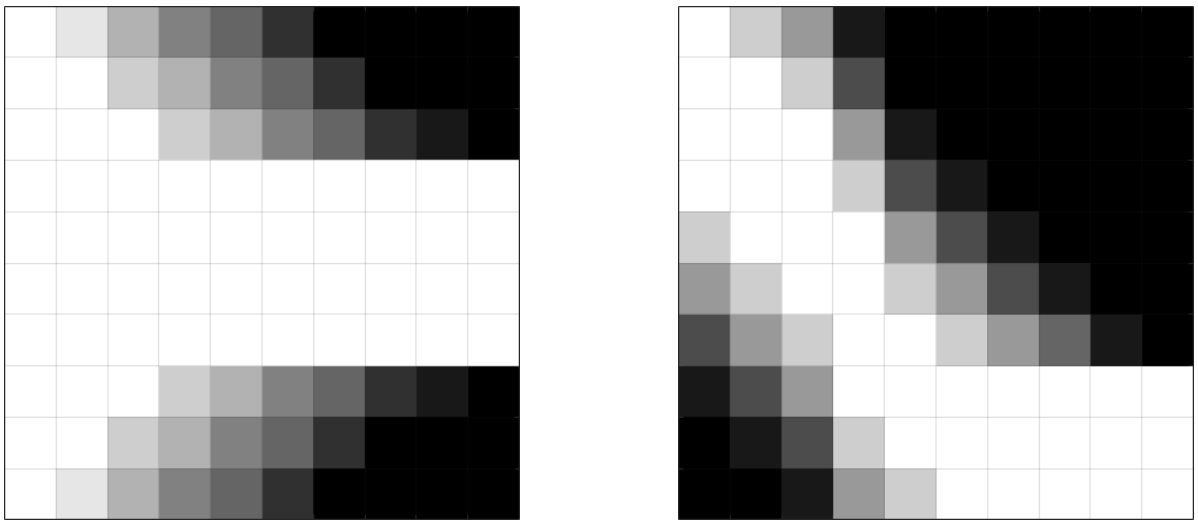
FIGURE 4.14 – Répartition de la vitesse et lignes de courant du domaine 2x2 pour différentes vitesses d'entrée avec les variables de conception de la Fig.4.10c

La comparaison des trois figures mène à la conclusion suivante : plus la vitesse imposée en entrée est élevée et moins le modèle de Brinkman est capable d'imiter un milieu solide. En effet, plus la vitesse d'entrée est élevée et plus il y a de lignes de vitesse passant par l'élément supposé être presque totalement solide (élément supérieur droit). Une plus grande distance est nécessaire pour une diminution de la vitesse.

4.3.3 Cas 10x10

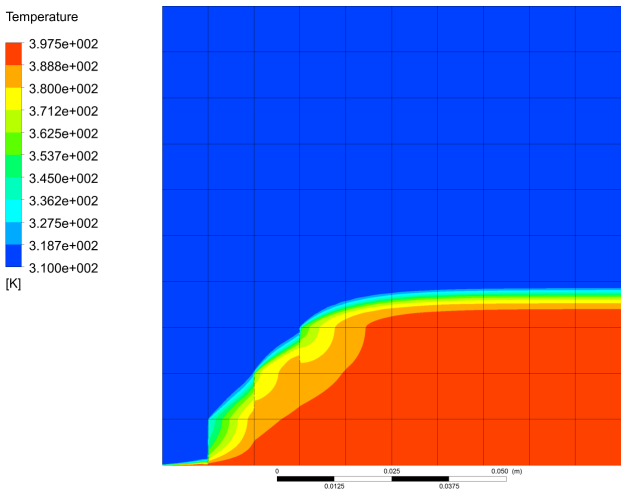
Un domaine de 10x10 éléments a ensuite été considéré pour permettre une plus grande liberté sur la répartition des variables de conception sur le domaine. Ce domaine est beaucoup plus important que le précédent et comme cité ci-dessus, l'initialisation des sous-programmes utilisés sur ANSYS exige plus de temps.

Deux designs représentés par les valeurs des variables de conception de chaque élément du domaine sont considérés et sont présentés à la Fig. 4.15. Le premier est simplement un écoulement en forme de convergent alors que le deuxième représente un écoulement guidé vers la paroi inférieure. Dans les deux cas, une source de chaleur à température fixe est placée sur la paroi inférieure. Les conditions aux limites imposées sur le domaine pour la résolution de l'écoulement sont toujours celles données dans la table 4.1.
Les répartitions de pression, de température et de vitesse ainsi que les lignes de courant sont visibles aux Fig.4.16 et 4.17 pour les deux designs respectivement.

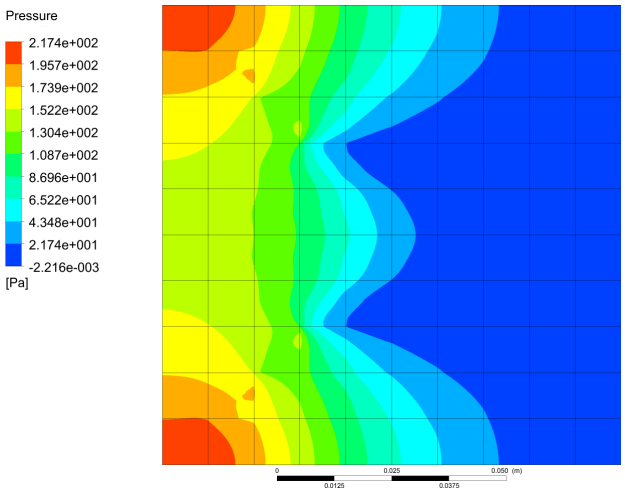


(a) Design 1 (b) Design 2

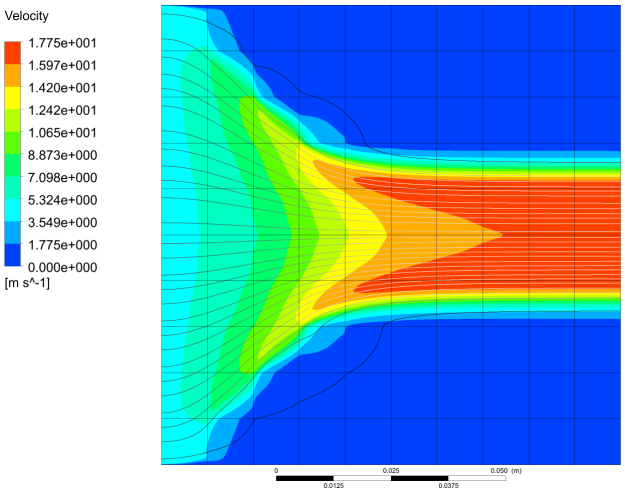
FIGURE 4.15 – Valeurs des variables de conception ρ_e du domaine 10x10 pour les deux designs considérés



(a) Répartition de la température

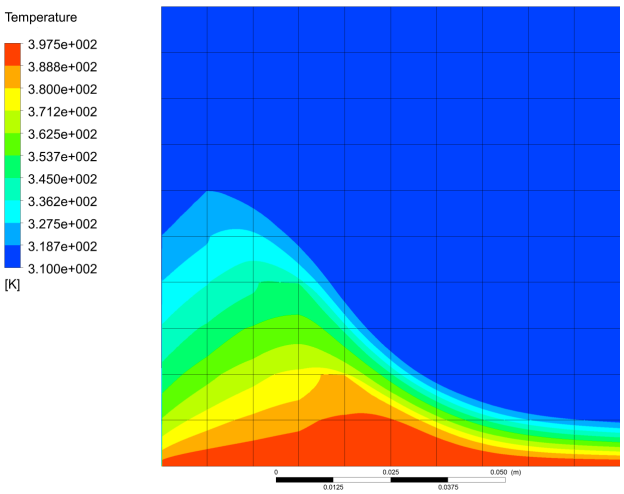


(b) Répartition de la pression

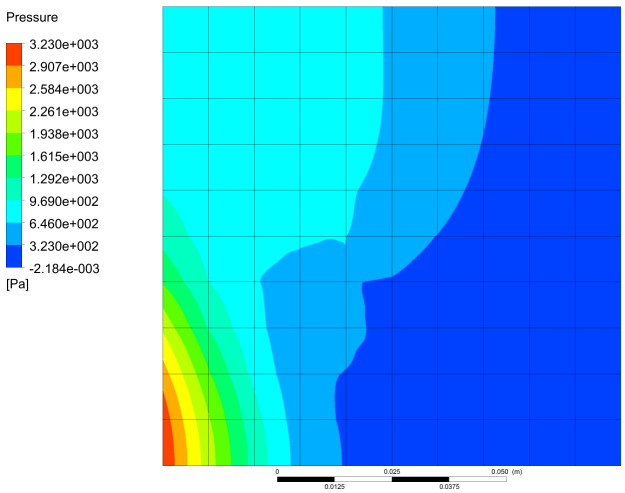


(c) Répartition de la vitesse et lignes de courant

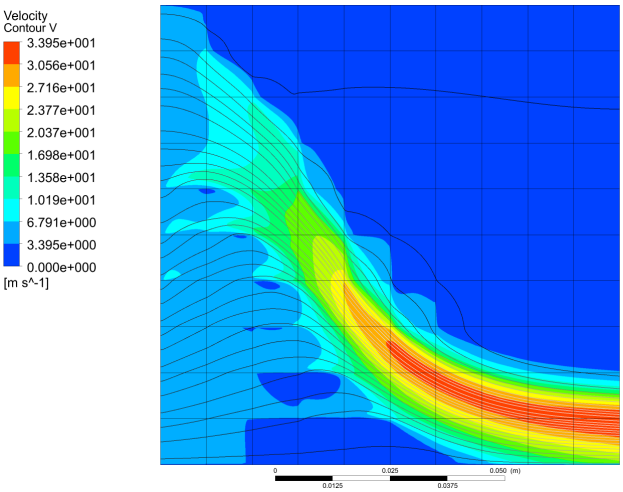
FIGURE 4.16 – Cas 10x10 avec les variables de conception de la Fig.4.15a



(a) Répartition de la température



(b) Répartition de la pression



(c) Répartition de la vitesse et lignes de courant

FIGURE 4.17 – Cas 10x10 avec les variables de conception de la Fig.4.15b

Pour le premier design, le domaine étant symétrique pour les variables de conception, les répartitions de pression (Fig.4.16b) et de vitesse (Fig.4.16c) le sont donc également. La pression est plus élevée aux extrémités de l'entrée, ce qui est certainement dû au rétrécissement de la section créé par les zones poreuses et au manque d'un collecteur pour amener le fluide vers cette entrée. Ensuite, la condition de sortie étant une pression manométrique nulle, la pression s'uniformise et sa valeur diminue.

La vitesse, quant à elle, est uniforme à l'entrée du domaine, encore une fois à cause de la condition imposée, et sa valeur augmente avec la section de passage qui diminue comme attendu pour un écoulement dans un entonnoir. Comme pour le cas 2x2, les lignes de courant suivent bien les zones où les variables de conception sont unitaires et la vitesse dans les zones solides est nulle ou presque.

La répartition de température montre bien le phénomène de conduction de chaleur qui a lieu dans la partie solide en contact avec la source de chaleur. Le fluide se réchauffe un peu au voisinage de cette partie solide mais pas assez pour impacter la partie solide supérieure du domaine. Celle-ci reste donc à la température d'entrée du fluide.

Pour le deuxième design, la répartition de la température (4.17a) montre -comme pour le premier design- le transfert de chaleur par conduction qui a lieu dans la partie solide inférieure du domaine. La température en entrée est bien celle fixée par les conditions aux limites (un zoom sur le graphique est indispensable pour le constater). Cependant, elle est très vite influencée par la source de chaleur sur la paroi inférieure du domaine.
La pression (4.17b) est maximale en entrée dans le coin inférieur gauche du domaine où le solide se trouve. Cela parait censé puisque le fluide a des difficultés à passer dans les zones moins poreuses.
Les lignes de courant suivent encore une fois bien le trajet créé avec les variables de conception.

Comme pour le cas 2x2, la finesse du maillage à utiliser doit être étudiée en comparant plusieurs maillages. Il en est de même pour l'étude de l'influence de la vitesse imposée en entrée. Cependant, les résultats sont semblables à ceux obtenus pour le cas 2x2 et ne seront, pour cette raison, pas explicités pour ce cas-ci.

4.4 Resultats du solveur adjoint

Le solveur adjoint est utilisé lorsque le calcul effectué par Fluent pour simuler l'écoulement est terminé. Ce solveur doit être lancé manuellement pour chaque calcul de dérivée qui doit être réalisé, ce qui peut prendre beaucoup de temps en fonction du nombre d'éléments étant entendu que l'optimisation sollicite les dérivées de la température, de la vitesse et de la pression de chaque élément du domaine.

Chaque fonction d'intérêt doit être définie comme "observable" du solveur adjoint. Le solveur adjoint permet d'afficher la valeur de chaque observable avant de calculer la dérivée, c'est-à-dire sa valeur à la fin de la simulation de l'écoulement faite par Fluent et, pour chaque observable, il peut être stipulé si le but de l'optimisation est de le minimiser ou de le maximiser. Cette information est utilisée par Fluent si le but est de réaliser une optimisation de forme directement sur le solveur.

Le solveur adjoint est initialisé en choisissant les méthodes de résolutions qui, dans la mesure du possible, devraient être les mêmes que pour le solveur Fluent. Le choix du critère d'arrêt de la résolution doit également être déterminé : seuil de convergence des résidus ou nombre fixé d'itérations.

Lorsque le calcul est terminé, la solution peut être extraite sous forme d'un fichier lisible par Matlab. Pour ce faire, le solveur demande de choisir une frontière sur laquelle est estimée la sensibilité. Seules les frontières extérieures du domaine (représentées à la Fig.4.3) sont accessibles alors que pour l'optimisation topologique il faudrait pouvoir choisir des volumes (chaque élément). Les séparations entre les éléments ne sont pas des parois et ne peuvent pas être définies comme telles sans influencer de manière négative le résultat final. Ceci est un **point bloquant** pour le couplage considéré.

Un exemple de résultat du solveur adjoint est tout de même donné à la Fig.4.18. L'observable principal est la perte de charge entre l'entrée et la sortie du domaine. La sensibilité est donnée sur la sortie de l'élément du coin inférieur droit (élément 100). Ce rapport de la sensibilité signifie que la pression manométrique en sortie est nulle après simulation par Fluent et qu'une variation de 1 [Pa] à la sortie entrainerait une diminution de 0.019 [Pa] de la perte de charge. Dans le cas considéré, la température du dernier élément est aussi indiquée comme observable (secondaire) et le rapport donne donc l'effet d'une variation de 1 [K] de cette température sur

la perte de charge (l'effet est nul). Enfin, pour chaque observation, un conseil est donné pour l'optimisation en fonction du but recherché pour l'observable.

```
Boundary condition sensitivity report: outlet_v2-elm100
Observable: pressure-drop
Gauge Pressure (pascal): 0  Sensitivity ((pascal)/(pascal)): -0.01900202
Increase Gauge Pressure to decrease pressure-drop
Temperature (k): 300  Sensitivity ((pascal)/(k)): 0
Decrease Temperature to decrease pressure-drop
```

FIGURE 4.18 – Exemple des informations renvoyées par le solveur adjoint

Bien que l'aide Fluent [1] affirme que les paramètres du modèle entrés par l'utilisateur font partie des variables de contrôle, aucun moyen n'est mis à disposition pour choisir ces variables lorsque la valeur de la sensibilité de la fonction d'intérêt est demandée.
Une solution pour obtenir la valeur des dérivées en fonction des porosités serait peut-être de fixer ces dernières comme observables secondaires constant. Lorsque le solveur adjoint calcule la sensibilité d'un observable recherché (température, vitesse ou pression), ses sensibilités par rapport aux observables secondaires sont également donnés. Cette solution n'est valable que lorsque le nombre d'éléments est limité car il faut modifier manuellement les valeurs des observables constants en fonction des porosités de l'itération en cours. Aucune manière d'automatiser cette étape n'a été trouvé.

Quelques problèmes se sont également présentés avec l'utilisation de l'UDF lors du lancement du solveur adjoint. Comme pour les milieux poreux, les UDF ne sont pas compatibles avec le solveur adjoint pour les versions antérieures à la version 16.0.

4.5 Conclusion

Ce chapitre se concentre sur le couplage envisageable entre les logiciels Matlab et ANSYS Fluent. Le couplage est d'abord décrit et les différents sous-programmes utilisés sont rapidement présentés. Le besoin de calculer les dérivées de la fonction objectif et des contraintes est exprimé et des solutions sont exposées.
Des codes Matlab sont écrits pour traduire les données d'un logiciel à l'autre et la partie Fluent est testée pour différents cas.

Chapitre 5

Conclusion générale

La première partie de ce travail a consisté à consulter la littérature afin de regrouper toutes les informations intéressantes relatives à l'optimisation topologique faisant intervenir des échanges de chaleur et l'écoulement de fluide. Ces recherches ont permis de découvrir quelques articles intéressants mais aussi certains logiciels capables à l'heure actuelle de réaliser de l'optimisation topologique. La plupart des articles rapporte leur résultats sans expliquer la méthode utilisée pour les obtenir, ce qui complique la mise en pratique de l'optimisation. En outre, au niveau des logiciels, peu d'informations sont accessibles : il est donc malaisé de se forger une idée précise à leur sujet. Enfin, un des logiciels découverts, à savoir COMSOL Multiphysics, paraissant le plus documenté, a fait l'objet d'une étude plus complète. En conclusion de cette étude il ressort que le logiciel pourrait s'avérer utile malgré un coût onéreux à la base et la nécessité d'acquérir des modules complémentaires spécifiques à l'optimisation topologique. Dès lors, si le but est de travailler sur des cas prévus par le logiciel, l'investissement s'avère sans doute rentable. Par contre, pour des cas non pris en compte par le logiciel, la modification des codes nécessaire à la résolution s'avère tellement complexe qu'une autre solution doit être envisagée.

L'utilisation d'un logiciel mieux connu est considéré dans le chapitre 3. Dans celui-ci, un code écrit dans le logiciel Matlab et accomplissant de l'optimisation topologique d'un système soumis à de la conduction de chaleur a été analysé et modifié afin de pouvoir l'utiliser pour des cas bien spécifiques. Une partie importante du chapitre est consacrée à la définition de la fonction objectif connue sous le nom de "dissipation d'entransie". L'entransie est une nouvelle propriété qui n'obtient pas l'assentiment de tous mais pourrait par contre se révéler fort utile si elle s'avérait fondée. De cette propriété et de sa dissipation découlent des principes d'extremum ainsi que l'expression d'une résistance thermique lesquels pourraient constituer des critères pour l'optimisation de systèmes impliquant des échanges de chaleur.
Le problème traité reste très simple ; quelques résultats ont déjà pu être obtenus et les différentes étapes du processus d'optimisation topologique ont pu être observées. Ce travail a principalement consisté à rassembler des idées de conditions aux limites à appliquer aux domaines et à les implémenter. De nombreux cas ont ainsi été optimisés mais tous n'ont pas conduits à des designs intéressants. Seuls trois d'entre eux méritent d'être retenus et présentés ici, c'est notamment le cas d'un échangeur surfacique. Le design obtenu pour ce dernier peut déjà être considéré tel quel et ses performances étudiées pour des cas faisant intervenir de la convection afin de déterminer s'il n'est pas déjà plus performant que les échangeurs utilisés actuellement. Les différents paramètres qui interviennent dans le processus sont cités et leurs effets sur le résultat de l'optimisation sont expliqués et dans quelques cas schématisés. Le gain obtenu sur la fonction objectif en utilisant l'optimisation topologique plutôt qu'en "devinant" le design est quantifié et le choix de la fonction objectif justifié. Ce chapitre 3 permet de démon-

trer que même en restant sur des cas simples, l'optimisation topologique permet d'obtenir des designs innovants qui pourraient, à terme, mener à une amélioration significative des performances des échangeurs de chaleur. Cela est encore plus vrai si la fabrication additive est utilisée.

Enfin, pour aller encore plus loin dans l'optimisation topologique et pouvoir considérer l'échange de chaleur entre un fluide et de la matière solide sous forme de conduction mais aussi de convection, le chapitre 4 envisage un couplage entre deux logiciels : Matlab et ANSYS Fluent. Le choix de Fluent en tant que solveur fluide se justifie par la présence d'un solveur adjoint dans ses modules complémentaires et par son utilisation déjà bien implantée au sein de l'entreprise.
Pour tenir compte de la variation de la topologie du domaine d'une itération à l'autre, les éléments sont définis comme des zones plus ou moins poreuses en fonction de leur rôle de fluide ou de solide. La porosité de chaque élément du domaine est imposée à Fluent par un UDF écrit par Matlab directement à partir des variables de conception du problème. Ce code Matlab et cet UDF sont présentés.
Les données échangées par les deux solveurs ainsi que la manière dont l'échange peut être réalisé sont décrites. Des premiers résultats du solveur fluide pour une itération fictive de l'optimisation sont obtenus et analysés dans la section (4.3). Ces résultats sont très prometteurs puisque le fluide se déplace effectivement dans le domaine en tenant compte des zones plus ou moins poreuses.
Après la résolution de l'écoulement fluide, le solveur adjoint de Fluent est utilisé pour calculer les dérivées des variables nécessaires à l'optimisation.
Sont également présentés des codes écrits en Matlab qui permettent d'extraire les données renvoyées par Fluent et de les stocker en vue de la mise à jour des variables par l'algorithme d'optimisation.

A l'heure actuelle, ce couplage reste complexe à implémenter. Quelques problèmes sont toujours en attente d'être réglés, notamment en ce qui concerne le logiciel Fluent. Sans parler du temps mis en œuvre pour initialiser le solveur -qui nécessite entre autres d'indiquer pour chaque élément le nom des variables dont il dépend- le solveur adjoint n'est toujours pas au point pour les résultats attendus. Ce solveur a déjà été amélioré puisque entre deux versions utilisées lors du stage, les zones poreuses ont pu être considérées et les observables définis sur des volumes. Malgré cela, les solutions données par le solveur ne peuvent être exprimées qu'en fonction de certaines variables pré-déterminées lesquelles ne contiennent pas les variables utiles pour l'optimisation. Une solution est envisagée mais ne serait optimale lors d'un processus d'optimisation réel. Cependant, au vu de la présence croissante de l'optimisation topologique dans le milieu des fluides et des échanges de chaleur, ces problèmes devraient être résolus dans un avenir proche.

Dès lors, à ce stade, l'utilisation d'un logiciel plus académique et facile à manipuler comme OpenFoam est sans doute préférable malgré le temps nécessaire à la prise en main et à l'implémentation.

L'optimisation topologique sera sans aucun doute un jour un outil indispensable pour la conception de nouveaux échangeurs de chaleur. Le partenariat entamé entre l'entreprise et l'université permettra d'accélérer la concrétisation de cette avancée technologique. Dans cette attente, certains cas simples et bien compris peuvent déjà être utilisés pour fournir des idées de nouveaux designs.

Bibliographie

[1] ANSYS Help 15.0. Fluent adjoint solver module manual.

[2] ANSYS Help 15.0. Fluent theory guide 1.2.

[3] ANSYS Help 15.0. Fluent theory guide 5.2.1.1.

[4] ANSYS Help 15.0. Fluent udf manual.

[5] ANSYS Help 15.0. Fluent user's guide 6.2.3.

[6] ANSYS Help 15.0. Meshing user's guide 20.5.

[7] Ansys. Jacobian ratio. `https://www.sharcnet.ca/Software/Ansys/17.0/en-us/help/wb_msh/msh_jacobian_ratio.html`. Accessed : 05-06-2018.

[8] Adrian Bejan. "entransy," and its lack of content in physics. *Journal of Heat Transfer*, 136, May 2014.

[9] V. Bertola and E. Cafaro. A critical analysis of the minimum entropy production theorem and its application to heat and fluid flow. *International journal of heat and mass transfer*, 51 :1907–1912, 2008.

[10] Thomas Borrvall and Joakim Petersson. Topology optimization of fluids in stokes flow. *International journal for numerical methods in fluids*, 41 :77–107, 2003.

[11] Peter Coffin and Kurt Maute. Level set topology optimization of cooling and heating devices using a simplified convection model. *Structural and Multidisciplinary Optimization*, 53 :985–1003, 2016.

[12] COMSOL. Produits. `https://www.comsol.fr/products`. Accessed : 05-06-2018.

[13] Ercan M. Dede. Multiphysics topology optimization of heat transfer and fluid flow systems. *Proceedings of the COMSOL Conference Boston*, 2009.

[14] M.P. Bendsøe and O. Sigmund. *Topology Optimization : Theory, Methods, and Applications*. Springer, 2004.

[15] Maison Energie. Echangeur de chaleur – échangeur thermique. `https://energie2maison.wordpress.com/2009/10/01/echangeur-de-chaleur-echangeur-thermique/`, octobre 2009. Accessed : 05-06-2018.

[16] Engys. Helyx - open-source cfd for enterprise. `https://engys.com/products/helyx`. Accessed : 05-06-2018.

[17] Adriano A. Koga et al. Development of heat sink device by using topology optimization. *International Journal of Heat and Mass Transfer*, 64 :759–772, 2013.

[18] Christian Mark et al. Heat exchanger design with topology optimization. 2017.

[19] Dusan P. Sekulic et al. Entransy : A misleading concept for the analysis and optimization of thermal systems. *Elsevier Energy*, 80 :251–253, 2015.

[20] E. A. Kontoleontos et al. Adjoint-based constrained topology optimization for viscous flows, including heat transfer. *Engineering Optimization*, 45(8) :941–961, 2013.

[21] Ercan M. Dede et al. Topology optimization, additive layer manufacturing, and experimental testing of an air-cooled heat sink. *Journal of Mechanical Design*, 137, November 2015.

[22] Gilles Marck et al. Topology optimization of heat and mass transfer problems : Laminar flow. *Numerical Heat Transfer, Part B : Fundamentals : An International Journal of Computation and Methodology*, 63 :6 :508–539, 2013.

[23] Jan Hendrik Klaas Haertel et al. Topology optimization of thermal heat sinks. *Proceedings. COMSOL conference*, 2015.

[24] Kentaro Yaji et al. A topology optimization method for a coupled thermal–fluid problem using level set boundary expressions. *International Journal of Heat and Mass Transfer*, 81 :878–888, 2015.

[25] Michael Yu Wang et al. A level set method for structural topology optimization. *Computer methods in applied mechanics and engineering*, 192 :227–246, 2003.

[26] Mingdong Zhou et al. Industrial application of topology optimization for combined conductive and convective heat transfer problems. *Structural and Multidisciplinary Optimization*, 54 :1045–1060, 2016.

[27] Qun Chen et al. An alternative criterion in heat transfer optimization. *Proceedings of the Royal Society A : Mathematical, Physical and Engineering Sciences*, 2010.

[28] R. W. Lewis et al. *Fundamentals of the Finite Element Method for Heat and Fluid Flow*. John Wiley and Sons, 2004.

[29] Tadayoshi Matsumori et al. Topology optimization for fluid–thermal interaction problems under constant input power. *Structural and Multidisciplinary Optimization*, 47 :571–581, 2013.

[30] Zeng-Yuan Guo et al. Entransy - a physical quantity describing heat transfer ability. *International journal of heat and mass transfer*, 50 :2545–2556, 2007.

[31] Heinz Herwig. Do we really need "entransy" ? a critical assessment of a new quantity in heat transfer analysis. *Journal of Heat Transfer*, 136, April 2014.

[32] Icon. iconcfd thermal. `http://www.iconcfd.com/en/products/simulation/thermal`. Accessed : 05-06-2018.

[33] Milivoje M. Kostic. Entransy concept and controversies : A critical perspective within elusive thermal landscape. *International Journal of Heat and Mass Transfer*, 115 :340–346, 2017.

[34] Sebastian Kreissl and Kurt Maute. Levelset based fluid topology optimization using the extended finite element method. *Structural and Multidisciplinary Optimization*, 46 :311–326, 2012.

[35] André Lallemand. Bilans entropiques et exergétiques. *Thermodynamique appliquée*, 2008.

[36] Joaquim R. R. A. Martins. Sensitivity analysis. *Multidisciplinary Design Optimization*, -.

[37] MathWorks. Matlab. `https://nl.mathworks.com/products/matlab.html`. Accessed : 05-06-2018.

[38] Ulf Nilsson. Description of adjointshapeoptimizationfoam and how to implement new objective functions. Technical report, Chalmers University of Technology, 2014.

[39] Tijs Van Oevelen and Martine Baelmans. Application of topology optimization in a conjugate heat transfer problem. *An International Conference on Engineering and Applied Sciences Optimization*, June 2014.

[40] Panagiotis Papazoglou. Topology optimization of heat exchangers. Master's thesis, Delft University of Technology, 2015.

[41] Safran. Heat exchangers to supplement its range of oil system equipment. `https://www.safran-aero-boosters.com/propulsion-equipment/heat-exchangers-supplement-its-range-oil-system-equipment`. Accessed : 05-06-2018.

[42] Avinash Shukla and Anadi Misra. Review of optimality criterion approach scope, limitation and development in topology optimization. *International Journal of Advances in Engineering and Technology*, 6(Issue 4) :1886–1889, Septembre 2013.

[43] Ole Sigmund and Kurt Maute. Topology optimization approaches. *Structural and Multidisciplinary Optimization*, 48 :1031–1055, 2013.

[44] Krister Svanberg. The method of moving asymptotes - modelling aspects and solution schemes. *Royal Institute of Technology (KTH), Lectures notes for the DCAMM course*, 1998.

[45] Dassault systemes. Tosca fluid. `https://www.3ds.com/fr/produits-et-services/simulia/produits/tosca/fluid/`. Accessed : 05-06-2018.

[46] M. Towara and U. Naumann. A discrete adjoint model for openfoam. *Procedia Computer Science*, 18 :429–438, 2013.

[47] Alexandre Vaudrey. Un nouveau concept pour l'optimisation des échanges de chaleur : l'entransie.

Annexe A

Codes Matlab d'optimisation topologique pour la conduction pure

```
%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, OCTOBER
    1999 %%

%%%%%% Fonction objectif : Minimiser la compliance thermique %%%%%%%
%%%%%%     Contrainte : volfrac = fraction volumique max     %%%%%%%
%%%%%%              Processus d'optimisation : OC            %%%%%%%

function toph_compli_vmax_OC(nelx,nely,volfrac,penal,rmin,whichcase)

% INITIALIZE

x(1:nely,1:nelx) = volfrac; % Matrice des variables de conception (
    porosite)
dc = zeros(nely,nelx); % Matrice de la derivee de la fonction
    objectif

loop = 0;
change = 1.;

% START ITERATION

while change> 0.01

        loop = loop + 1;
        xold = x;

% FE-ANALYSIS

        [U]=FE(nelx,nely,x,penal,whichcase); % Vecteur de
            temperature

% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS

        [KE] = lk;    % Matrice de rigidite thermique fonction de x
        c=0.; % Compliance thermique
```

```
        for ely = 1:nely
                for elx = 1:nelx
                n1 = (nely+1)*(elx-1)+ely;
                n2 = (nely+1)* elx +ely;
                Ue = U([n1; n2; n2+1; n1+1],1);
                        c = c + (0.001+0.999*x(ely,elx)^penal)*Ue'*
                           KE*Ue;
                        dc(ely,elx) = -0.999*penal*x(ely,elx)^(penal
                           -1)*Ue'*KE*Ue;
                end
    end
    vector_c(loop) = c;

% FILTERING OF SENSITIVITIES

    [dc] = check(nelx,nely,rmin,x,dc);

% DESIGN UPDATE BY TBE OPTIMALITY CRITERIA METHOD

        [x] = OC(nelx,nely,x,volfrac,dc);

% PRINT RESULTS
        change = max(max(abs(x-xold)));
        disp(['␣It.:␣' sprintf('%4i',loop) '␣Obj.:␣' sprintf('%10.4f
            ',c) ...
        '␣Vol.:␣' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ...
                '␣ch.:␣' sprintf('%6.3f',change ) '␣Penal.:␣'
                   sprintf('%i',penal)])

% PLOT DENSITIES
        colormap(gray); shape = imagesc(-x); axis equal; axis tight;
            axis off;pause(1e-6);

    if loop == 1000
        error('Error:␣ne␣converge␣pas');
    end

end

% Plot de la fonction objectif
figure
  compli = plot(1:1:loop,vector_c);
  grid on
  titlename = sprintf(['Compliance␣evolution␣\n(nelx␣=␣',num2str(
     nelx),',␣nely␣=␣',num2str(nely),',␣volfrac␣=␣',num2str(volfrac),
     ',␣penal␣=␣',num2str(penal),',␣rmin␣=␣',num2str(rmin),')']);
  title(titlename)
  xlabel('Number␣of␣iterations␣[-]','FontSize',14,'FontWeight','bold
     ','FontName','Times','Color','black');
```

```
   ylabel('Compliance','FontSize',14,'FontWeight','bold','FontName','
      Times','Color','black');
   set(gca,'fontsize',12,'fontname','Times','Linewidth', 0.5);

%%%%%%%%% OPTIMALITY CRITERIA UPDATE%%%%%%%%%%%%%%%%%%
function [xnew]=OC(nelx,nely,x,volfrac,dc)
l1 = 0; l2 = 100000; move = 0.2;
while (l2-l1 > 1e-4)
     lmid = 0.5*(l2+l1);
     xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)
        ))));
     if sum(sum(xnew)) - volfrac*nelx*nely > 0
          l1 = lmid;
     else
          l2 = lmid;
     end
end


%%%%%%%%%%%%%%%MESH-INDEPENDENCY FILTER%%%%%%%%%%%%%%%

function [dcn]=check(nelx,nely,rmin,x,dc)
dcn=zeros(nely,nelx);
for i = 1:nelx
     for j = 1:nely
          sum=0.0;
          for k = max(i-floor(rmin),1):min(i+floor(rmin),nelx)
               for l = max(j-floor(rmin),1):min(j+floor(rmin),nely)
                    fac = rmin-sqrt((i-k)^2+(j-l)^2);
                    sum = sum+max(0,fac);
                    dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
               end
          end
          dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
     end
end


%%%%%%%%%%%%%%%%%%%%FE-ANALYSIS%%%%%%%%%%%%%%%%%%%%%%

function [U]=FE(nelx,nely,x,penal,whichcase)
[KE] = lk;
K = sparse((nelx+1)*(nely+1), (nelx+1)*(nely+1)); %1 ddl par noeud
F = sparse((nely+1)*(nelx+1),1) ; U = sparse((nely+1)*(nelx+1),1);
for elx = 1:nelx
     for ely = 1:nely
          n1 = (nely+1)*(elx-1)+ely;
          n2 = (nely+1)* elx +ely;
          edof = [n1; n2; n2+1; n1+1];
```

```
            K(edof,edof) = K(edof,edof) + (0.001+0.999*x(ely,elx)^penal)
                *KE;
        end
    end

% DEFINE LOADS AND SUPPORTS (SQUARE PLATE WITH HEAT SINK)
switch  whichcase

    case 1 % (Cas 1) CAS DE BASE: source de chaleur uniforme
        distribuee sur le domaine et dissipateur sur le centre de la
        face ouest
         F(:,1) = 0.01;
         fixeddofs = nely/2+1-(nely/20):nely/2+1+(nely/20);

    case 2 % (Cas 2) Source centree sur la face nord, dissipateur
        centre sur la face sud
         F(1+(nely+1)*(ceil((nelx+1)/2)-1)-(ceil(nelx/5))*(nely+1):
            nely+1:1+(nely+1)*(ceil((nelx+1)/2)-1)+(ceil(nelx/5))*(
            nely+1),1) = 0.01;
         fixeddofs = (nely+1)*(ceil((nelx+1)/2))-(ceil(nelx/5))*(nely
            +1):nely+1:(nely+1)*(ceil((nelx+1)/2))+(ceil(nelx/5))*(
            nely+1);

    case 3 % (Cas 3) Source etendue sur la face nord, dissipateur
        ponctuel au centre sur la face sud
         F(1+nely+1:nely+1:1+(nely+1)*(nelx-1),1) = 0.01;
         fixeddofs = ceil((nelx+1)/2)*(nely+1);

    case 4 % (Cas 4) Source sur toute la face nord, dissipateur sur
        toute la face sud
         F(1:nely+1:nelx*(nely+1)+1,1) = 0.01;
         fixeddofs = (nely+1):nely+1:(nely+1)*(nelx+1);

    otherwise

         error('Provide a correct case number!!');
end
alldofs = 1:(nely+1)*(nelx+1);
freedofs = setdiff(alldofs,fixeddofs);

% SOLVING

U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
U(fixeddofs,:)= 0;


%%%%%%%%%%%%%%%%%ELEMENT STIFFNESS MATRIX%%%%%%%%%%%%%%%%%%

function [KE] =lk
KE = [ 2/3 -1/6 -1/3 -1/6
```

```
-1/6  2/3  -1/6  -1/3
-1/3  -1/6  2/3  -1/6
-1/6  -1/3  -1/6  2/3];
```

Annexe B

Code Matlab pour générer l'UDF de Fluent

```
% Fonction generant un UDF pour fixer les porosites et les
   resistances
% visqueuses des zones poreuses.

%ENTREES : vecteur x des porosites (variables de conception) et
   parametre
%de penalisation q

function udf_PorousZones(x,q)

fileID = fopen('udf_PorousZones.c','w');
fprintf(fileID,'#include "udf.h" \n\n');
fprintf(fileID,'real rho[100] = {');
fprintf(fileID,'%5.1f,',x(1:length(x)-1));
fprintf(fileID,'%5.1f',x(length(x)));
fprintf(fileID,'}; \n');

fprintf(fileID,'real b_max = 2.5/(0.01*0.01); \n');
fprintf(fileID,'real b_min = 2.5/(100*100); \n');
fprintf(fileID,'real q =');
fprintf(fileID,'%5.1f',q);
fprintf(fileID,'; \n');
fprintf(fileID,'real mu = 1.7894e-05; \n');
fprintf(fileID,'real k; \n');

for loop = 1:length(x)
    name_por = ['porosity_elm',num2str(loop)];
    name_VRes = ['ViscousRes_elm',num2str(loop)];
    por = ['rho[',num2str(loop-1),']'];

    fprintf(fileID,'DEFINE_PROFILE('); %UDF pour imposer la porosite
        des elements
    fprintf(fileID, name_por);
    fprintf(fileID,', t, nv) \n');
    fprintf(fileID,'{ \n');
```

```
    fprintf(fileID,' cell_t c; \n');
    fprintf(fileID,' begin_c_loop(c,t) \n');
    fprintf(fileID,'\t C_PROFILE(c,t,nv) =');
    fprintf(fileID,por);
    fprintf(fileID,'; \n');
    fprintf(fileID,' end_c_loop(c,t) \n');
    fprintf(fileID,'} \n');

    fprintf(fileID,'DEFINE_PROFILE('); %UDF pour imposer la
       resistance visqueuse des zones poreuses
    fprintf(fileID, name_VRes);
    fprintf(fileID,', t, nv) \n');
    fprintf(fileID,'{ \n');
    fprintf(fileID,' cell_t c; \n');
    fprintf(fileID,' begin_c_loop(c,t) \n');

    fprintf(fileID,'\t k = (b_max + (b_min - b_max)*');
    fprintf(fileID,por);
    fprintf(fileID,'*(1+q)/(');
    fprintf(fileID,por);
    fprintf(fileID,'+q))/mu; \n');
    fprintf(fileID,'\t if(k<=0) \n');
    fprintf(fileID,'\t C_PROFILE(c,t,nv) = 1e-10; \n');
    fprintf(fileID,'\t else \n');
    fprintf(fileID,'\t C_PROFILE(c,t,nv) = k; \n');

    fprintf(fileID,' end_c_loop(c,t) \n');
    fprintf(fileID,'} \n');
end
fclose(fileID);
```

Exemple d'UDF en langage C généré par Matlab pour un domaine composé de deux éléments :

```
#include "udf.h"

real rho[100] = {  0.3,  1.0};
real b_max = 2.5/(0.01*0.01);
real b_min = 2.5/(100*100);
real q = 10.0;
real mu = 1.7894e-05;
real k;
DEFINE_PROFILE(porosity_elm1, t, nv)
{
 cell_t c;
 begin_c_loop(c,t)
         C_PROFILE(c,t,nv) =rho[0];
 end_c_loop(c,t)
}
DEFINE_PROFILE(ViscousRes_elm1, t, nv)
{
```

```
 cell_t c;
 begin_c_loop(c,t)
          k = (b_max + (b_min - b_max)*rho[0]*(1+q)/(rho[0]+q))/mu;
          if(k<=0)
          C_PROFILE(c,t,nv) = 1e-10;
          else
          C_PROFILE(c,t,nv) = k;
 end_c_loop(c,t)
}
DEFINE_PROFILE(porosity_elm2, t, nv)
{
 cell_t c;
 begin_c_loop(c,t)
          C_PROFILE(c,t,nv) =rho[1];
 end_c_loop(c,t)
}
DEFINE_PROFILE(ViscousRes_elm2, t, nv)
{
 cell_t c;
 begin_c_loop(c,t)
          k = (b_max + (b_min - b_max)*rho[1]*(1+q)/(rho[1]+q))/mu;
          if(k<=0)
          C_PROFILE(c,t,nv) = 1e-10;
          else
          C_PROFILE(c,t,nv) = k;
 end_c_loop(c,t)
}
```

Annexe C

Code Matlab pour extraire les résultats de Fluent

Code pour extraire les résultats de la simulation de l'écoulement sur le solveur Fluent :

```
function [T_tout_m, P_tdiff_m, v_mout_m] = Fluent_DataOut

filename_Ttout = 't_t_out-rfile.out'; %Fichiers renvoyes par Fluent
filename_Ptdiff = 'p_t_diff-rfile.out';
filename_Vmout = 'v_m_out-rfile.out';

startRow = 4;
endRow = 503; %Toutes les iterations

%% Initialize variables.
delimiter = ' ';
if nargin<=2
    startRow = 4;
    endRow = inf;
end

%% Format for each line of text:
%   column2: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = '%*q%f%[^\n\r]';

%% Open the text file.
fileID_Ttout = fopen(filename_Ttout,'r');
fileID_Ptdiff = fopen(filename_Ptdiff,'r');
fileID_Vmout = fopen(filename_Vmout,'r');

%% Read columns of data according to the format.
dataArray_Ttout = textscan(fileID_Ttout, formatSpec, endRow(1)-
    startRow(1)+1, 'Delimiter', delimiter, 'MultipleDelimsAsOne',
    true, 'HeaderLines', startRow(1)-1, 'ReturnOnError', false, '
    EndOfLine', '\r\n');
dataArray_Ptdiff = textscan(fileID_Ptdiff, formatSpec, endRow(1)-
    startRow(1)+1, 'Delimiter', delimiter, 'MultipleDelimsAsOne',
```

```
    true, 'HeaderLines', startRow(1)-1, 'ReturnOnError', false, '
    EndOfLine', '\r\n');
dataArray_Vmout = textscan(fileID_Vmout, formatSpec, endRow(1)-
    startRow(1)+1, 'Delimiter', delimiter, 'MultipleDelimsAsOne',
    true, 'HeaderLines', startRow(1)-1, 'ReturnOnError', false, '
    EndOfLine', '\r\n');
for block=2:length(startRow)
    frewind(fileID_Ttout);
    frewind(fileID_Ptdiff);
    frewind(fileID_Vmout);
    dataArrayBlock_Ttout = textscan(fileID_Ttout, formatSpec, endRow
        (block)-startRow(block)+1, 'Delimiter', delimiter, '
        MultipleDelimsAsOne', true, 'HeaderLines', startRow(block)-1,
         'ReturnOnError', false, 'EndOfLine', '\r\n');
    dataArrayBlock_Ptdiff = textscan(fileID_Ptdiff, formatSpec,
        endRow(block)-startRow(block)+1, 'Delimiter', delimiter, '
        MultipleDelimsAsOne', true, 'HeaderLines', startRow(block)-1,
         'ReturnOnError', false, 'EndOfLine', '\r\n');
    dataArrayBlock_Vmout = textscan(fileID_Vmout, formatSpec, endRow
        (block)-startRow(block)+1, 'Delimiter', delimiter, '
        MultipleDelimsAsOne', true, 'HeaderLines', startRow(block)-1,
         'ReturnOnError', false, 'EndOfLine', '\r\n');
    dataArray_Ttout{1} = [dataArray_Ttout{1};dataArrayBlock_Ttout
        {1}];
    dataArray_Ptdiff{1} = [dataArray_Ptdiff{1};dataArrayBlock_Ptdiff
        {1}];
    dataArray_Vmout{1} = [dataArray_Vmout{1};dataArrayBlock_Vmout
        {1}];
end

%% Close the text file.
fclose(fileID_Ttout);
fclose(fileID_Ptdiff);
fclose(fileID_Vmout);

%% Post processing for unimportable data.
% No unimportable data rules were applied during the import, so no
    post
% processing code is included. To generate code which works for
% unimportable data, select unimportable cells in a file and
    regenerate the
% script.

%% Allocate imported array to column variable names
t_t_out = dataArray_Ttout{:, 1};
p_t_diff = dataArray_Ptdiff{:, 1};
v_mout = dataArray_Vmout{:, 1};

iterations = zeros(500,1);
iterations(:) = (1:500);
```

```
Fin_Ttout = t_t_out(400:500);
T_tout_m = mean(Fin_Ttout);

Fin_Ptdiff = p_t_diff(400:500);
P_tdiff_m = mean(Fin_Ptdiff);

Fin_Vmout = v_mout(400:500);
v_mout_m = mean(Fin_Vmout);

%% Generation des graphes

% figure
% plot(iterations,t_t_out);
% grid on
% titlename1 = sprintf('Evolution de T_{t,out}');
% title(titlename1)
% xlabel('Itrations [-]','FontSize',14,'FontWeight','bold','FontName
    ','Times','Color','black');
% ylabel('T_{t,out} [K]','FontSize',14,'FontWeight','bold','FontName
    ','Times','Color','black');
% set(gca,'fontsize',12,'fontname','Times','Linewidth', 0.5);
% %
% figure
% plot(iterations,p_t_diff);
% grid on
% titlename1 = sprintf('Evolution de P_{t,diff}');
% title(titlename1)
% xlabel('Itrations [-]','FontSize',14,'FontWeight','bold','FontName
    ','Times','Color','black');
% ylabel('P_{t,diff} [Pa]','FontSize',14,'FontWeight','bold','
    FontName','Times','Color','black');
% set(gca,'fontsize',12,'fontname','Times','Linewidth', 0.5);
%
%
% figure
% plot(iterations,v_mout);
% grid on
% titlename1 = sprintf('Evolution de V_{m,out}');
% title(titlename1)
% xlabel('Itrations [-]','FontSize',14,'FontWeight','bold','FontName
    ','Times','Color','black');
% ylabel('V_{m,out} [m/s]','FontSize',14,'FontWeight','bold','
    FontName','Times','Color','black');
% set(gca,'fontsize',12,'fontname','Times','Linewidth', 0.5);
```

Code pour extraire les résultats de l'analyse de sensibilité faite par le solveur adjoint :

```
function [DP_diff_DP_out, DP_diff_DT_elm100] = Adjoint_DataOut

%% Import data from text file.
```

```
%% Initialize variables.
filename = 'adjoint_Pdrop_out_elm100'; %Fichier renvoye par Fluent
delimiter = {'\t',':'}; %Delimitation des differentes colonnes du
    fichier
startRow = 3;
endRow = 6;

%% Format for each line of text:
%   column3: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = '%*s%*s%f%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to the format.
dataArray = textscan(fileID, formatSpec, endRow-startRow+1, '
    Delimiter', delimiter, 'MultipleDelimsAsOne', true, 'TextType', '
    string', 'EmptyValue', NaN, 'HeaderLines', startRow-1, '
    ReturnOnError', false, 'EndOfLine', '\r\n');

%% Close the text file.
fclose(fileID);

%% Create output variable
adjointPdropoutelm100 = table(dataArray{1:end-1}, 'VariableNames', {
    'VarName3'}); %Extraction des donnees necessaires

%% Clear temporary variables
clearvars filename delimiter startRow endRow formatSpec fileID
    dataArray ans;
%% Write the outputs
DP_diff_DP_out = adjointPdropoutelm100(2,1);
DP_diff_DT_elm100 = adjointPdropoutelm100(4,1);
```