

UFR

de mathématique
et d'informatique

Université de Strasbourg

Université
de Strasbourg



Université de Strasbourg

UFR de Mathématiques et Informatique

Department of Mathematics

Master de Calcul Scientifique et
Mathématiques de l'Innovation (CSMI)

MODELES REDUITS POUR LA CONSTRUCTION D'UN JUMEAU NUMERIQUE DE L'EXTRUSION DE POLYMERES

Présenté par :

CHAHID RAHOUTI

Encadré par :

Dr. Pascal TREMBLAY

Soutenance :

27 août 2025

Remerciements

Je tiens, de prime abord, à remercier chaleureusement mon encadrant au sein de l'entreprise ACOME, Dr Pascal TREMBLAY, pour la confiance qu'il m'a accordé en acceptant de m'encadrer pour mon stage de Master, et j'espère que les résultats obtenus soient à la hauteur de ses attentes. Ensuite, je le remercie pour ses remarques, son soutien, ses conseils précieux et son encouragement, ainsi que sa disponibilité.

J'adresse tout particulièrement mes remerciements à M le professeur, Christophe PRUD'HOMME, coordonnateur du Master Calcul Scientifique et Modélisation de l'Innovation, pour tous les efforts déployés pour notre formation pendant les deux années écoulées.

Je tiens également à exprimer mes remerciements à tous les membres du jury ; de m'avoir honoré de leur présences et en acceptant de juger et évaluer mon travail.

Mes sincères remerciements vont également à tous les professeurs du Master Calcul Scientifique et Modélisation de l'Innovation, qui m'ont accompagné et aidés à m'améliorer durant mon cursus de formation.

Enfin je ne saurais oublier de remercier tous ceux qui m'ont aidé ou soutenu de toute manière que ce soit.

Résumé

Ce mémoire de fin d'études est consacré à la modélisation et à la simulation de l'extrusion de polymères plus spécifiquement à travers l'outillage pour l'enrobage du câble. Dans ce but, nous cherchons à construire un modèle réduit de cette sous-partie du système de fabrication du câble qui fera éventuellement partie d'un jumeau numérique. Dans ce cadre, nous avons mené une étude approfondie sur l'extrusion de polymères, en nous intéressant notamment à la transformation du problème physique en un modèle mathématique, en mettant particulièrement l'accent sur la problématique de la rhéologie des polymères.

Dans la première phase de ce travail, nous avons préparé une modélisation sous OpenFOAM de la simulation de la mécanique des fluides et de la thermique qui utilise un modèle de comportement Carreau-Yasuda pour décrire cette interaction.

Dans la deuxième phase, nous avons effectué une revue bibliographique sur les méthodes de réduction telles que Décomposition en Modes Propres (POD) et Autoencoder (AE). Ces méthodes ont ensuite été appliquées aux résultats numériques obtenus par la simulation de l'extrusion de polymères à l'aide de la librairie EZyRB, que nous analysons et discutons les résultats obtenus.

Table des matières

Résumé	3
Introduction	10
1 Étude bibliographique et exploration de l'extrusion	14
1.1 Extrusion de polymère	14
1.1.1 Principe de l'extrusion	14
1.1.2 Description de la tête d'extrusion	17
1.2 Rhéologie des polymères	18
1.2.1 Fluides non-newtoniens	19
1.2.2 Polymère fondu et le rhéomètre capillaire	20
1.2.3 Modèle de Carreau-Yasuda	22
1.2.4 Paramètres physiques	22
1.3 Modélisation physique mathématique du problème	23
1.3.1 Équations fondamentales de la mécanique des fluides	25
1.3.2 Incompressibilité et équation de continuité	26
1.3.3 Conservation de la quantité de mouvement	27
1.3.4 Conservation de l'énergie	29
2 Méthodes de simulation	31
2.1 Méthodes de volume fini	31
2.2 OpenFOAM	32

2.2.1	La méthode des volumes finis dans OpenFOAM	32
2.2.2	Résolution numérique du système d'équations	35
2.2.3	Algorithme SIMPLE	36
2.3	Implementation dans OpenFOAM	39
2.3.1	Extension de la loi de Carreau-Yasuda	39
2.3.2	Solveur simpleFoam	40
2.4	Automatisation des simulations	41
3	Méthodes de modèles réduits	43
3.1	Introduction	43
3.1.1	Exploration des méthodes de modèle réduit	45
3.1.2	Paramétrisation du problème	47
3.2	La méthode de réduction de modèle POD	47
3.2.1	La méthode POD classique	48
3.2.2	La méthode SVD (Singular Value Decomposition)	49
3.2.3	La méthode POD pour un problème paramétrique	49
3.3	Autoencodeur et bibliothèque EyzRB	51
3.3.1	Principe de l'autoencodeur	51
3.3.2	Bibliothèque EZyRB	53
4	Construction de modèles réduits et résultats	56
4.1	Simulation et résultats de la cavité 2D	56
4.1.1	Géométrie et conditions de bord de la Cavity 2D	56
4.1.2	Simulation de la cavité 2D	57
4.1.3	Résultats de réduction de modèle POD et AE	58
4.2	Géométrie et conditions aux bords de l'extrusion de profilé	62
4.2.1	Cas simple de profilé	62
4.2.2	Résultats de réduction de modèle POD et AE	62
4.3	Géométrie et conditions aux bords de l'extrusion de profilé cas axi-symétrique	67
4.3.1	Géométrie en axisymétrie	67
4.3.2	Simulation de l'extrusion de profilé	69

4.3.3 Résultats de réduction de modèle POD et AE pour l'axisymétrie	70
Conclusion	77
Perspectives	78
A Annexes	79
A.1 Code de la loi de Carreau-Yasuda	79
A.2 Code d'ajouter l'équation d'energie	86
A.3 Géometrie extruder sur GMSH	89
Référence	92

Table des figures

1	Extrudeuse [1]	10
2	Câble électrique dans le fief du spécialiste ACOME	12
1.1	Granulés de polymère	15
1.2	Extrudeuse monovis[7]	16
1.3	Schéma simplifié d'une extrudeuse[7]	16
1.4	La tête d'extrusion utilisée pour l'extrusion de câbles chez ACOME.	17
1.5	Dépendance de la contrainte de cisaillement par rapport au taux de cisaillement pour différents fluides <i>lien</i>.	19
1.6	Dépendance de la contrainte de cisaillement par rapport au taux de cisaillement pour différents fluides.<i>lien</i>	20
1.7	Plateforme de rhéométrie capillaire chez ACOME.	21
1.8	Dépendance de la contrainte de cisaillement par rapport au taux de cisaillement pour différents matériaux utilisés dans cette étude.	23
2.1	Discrétisation du domaine d'étude en volumes de contrôle.	33
2.2	Illustration du théorème de Gauss.	34
2.3	Schéma de l'algorithme SIMPLE.	38
3.1	Illustration de la variété des solutions et de l'approximation RB	46
3.2	Gain de temps du modèle RB dans les scénarios multi-requêtes.	46
3.3	Schéma d'un autoencodeur : encodeur, espace latent, décodeur, avec même nombre de neurones en entrée et en sortie.	52
3.4	Le diagramme de la structure complète de la bibliothèque EZyRB	54

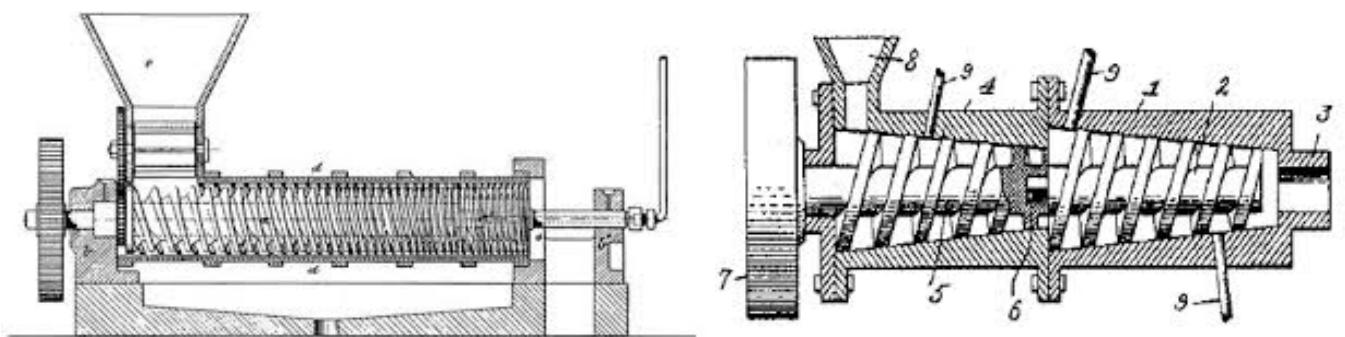
4.1	Maillages utilisés pour tester la convergence de la méthode	57
4.2	Simulation de la cavité 2D pour les trois variables U en m/s, P en Pa et T en K.	58
4.3	Solution de Haut fidélité	59
4.4	Solution de POD	60
4.5	Solution de AE	60
4.6	Erreur entre la solution de référence et la solution prédite par les méthodes POD	60
4.7	Erreur entre la solution de référence et la solution prédite par les méthodes AE	61
4.8	Comparaison de la précision entre les méthodes POD et AE	61
4.9	Géométrie 3D du profilé d'extrusion	62
4.10	Solution de Haut fidélité	64
4.11	Solution de POD	64
4.12	Solution de AE	65
4.13	Erreur entre la solution de référence et la solution prédite par les méthodes POD	65
4.14	Erreur entre la solution de référence et la solution prédite par les méthodes AE	66
4.15	Comparaison de la précision entre les méthodes POD et AE	67
4.16	Géométrie 3D du profilé d'extrusion	68
4.17	Géométrie 2D du profilé d'extrusion, avec indication des conditions aux bords	68
4.18	Maillage plus fin autour de la sortie du profilé d'extrusion	69
4.19	Solution de validation de l'extrusion de profilé	70
4.20	Solution de Haut fidélité	71
4.21	Solution de POD	72
4.22	Solution de AE	73
4.23	Erreur entre la solution de référence et la solution prédite par les méthodes POD	74

4.24 Erreur entre la solution de référence et la solution prédite par les méthodes AE	74
A.1 La structure des fichiers	79
A.2 Structure du solveur tempSimpleFoam avec tous ses fichiers à adapter	86

INTRODUCTION

L'extrusion, une technologie de fabrication en continu, remonte à la seconde moitié du XIX^e siècle. Dès 1879, Gray dépose le premier brevet concernant l'extrusion à vis, initialement utilisée pour l'isolation de câbles télégraphiques en caoutchouc naturel. Les années suivantes voient l'émergence des premières fabriques d'extrudeuses et de vis, marquant ainsi le début d'une évolution technologique majeure. Des avancées significatives ont été réalisées au fil des décennies, notamment avec l'introduction d'extrudeuses à pas variable pour le malaxage et le pétrissage en 1883, ou encore l'utilisation d'extrudeuses à dégazage à deux étages en 1913 pour éliminer l'air, l'humidité et les gaz des matériaux.

En 1954, Alex Doumak brevète un procédé mécanique révolutionnaire pour la production de guimauves, illustrant la polyvalence de l'extrusion. Ce procédé, qui consiste à transporter, fondre, malaxer, plastifier et comprimer les matières avant de les extruder sous pression à travers une filière, permet d'obtenir des produits aux formes précises et aux surfaces de haute qualité. Aujourd'hui, l'extrusion est une technologie en pleine expansion, appliquée dans des secteurs variés tels que l'agroalimentaire, l'industrie plastique et la métallurgie.



Desgoffe et Digiorgio

Price

FIGURE 1 – Extrudeuse [1]

Cependant, la complexité des phénomènes physiques impliqués dans l'extrusion, tels que les interactions thermiques et mécaniques, nécessite des outils numériques avancés pour leur modélisation et leur simulation. Dans ce contexte, la construction de jumeaux numériques

basés sur des modèles réduits (Reduced Order Models, ROM) s'impose comme une solution prometteuse. Ces modèles permettent de réduire la complexité des systèmes discrétisés d'ordre élevé, tout en conservant une précision suffisante pour les simulations. Ils sont particulièrement utiles dans les problèmes d'optimisation numérique en mécanique des fluides, où les systèmes matriciels de grande taille doivent être résolus de manière itérative.

Ces techniques remontent aux années 1960, dans le cadre de la théorie des systèmes et du contrôle, avec les travaux pionniers de Kalman et Zadeh. Les premières méthodes formelles, comme la troncature équilibrée introduite par Moore (1981), ont vu le jour dans les années 1980 [4]. L'introduction de la Proper Orthogonal Decomposition (POD), notamment par Sirovich (1987), marque un tournant en mécanique des fluides, en permettant d'extraire les structures dominantes d'un système à partir de données [2]. Par la suite, Holmes, Lumley et Berkooz (1996) ont largement contribué à la formalisation mathématique de la POD et à son application aux systèmes dynamiques [3]. Ces techniques se sont démocratisées avec l'essor du calcul numérique et trouvent aujourd'hui des applications cruciales dans la construction de jumeaux numériques, notamment grâce aux travaux de Antoulas (2005) sur la réduction de grands systèmes dynamiques [5], ainsi que ceux de Ito et Ravindran (1998) dans le contexte des équations aux dérivées partielles [6].

Considérons une classe de problèmes dépendant d'un paramètre $\mu \in \mathcal{D}$, où \mathcal{D} est un ensemble défini (par exemple, un sous-espace de \mathbf{R} , \mathbf{R}^n ou un espace fonctionnel). Ces problèmes peuvent être formulés comme suit : trouver $\mathbf{u} = \mathbf{u}(\mu) \in \mathbf{X}$ tel que $\mathcal{F}(\mathbf{u}, \mu) = 0$. À ce stade, l'ensemble \mathcal{D} n'est pas spécifié précisément. De tels problèmes se posent dans de nombreuses situations, telles que l'optimisation, le contrôle, l'identification de paramètres, l'analyse de surface de réponse ou l'analyse de sensibilité. Lorsque \mathcal{F} est exprimée à l'aide d'équations aux dérivées partielles, le problème peut être stationnaire ou dépendant du temps. Dans tous les cas, une solution $\mathbf{u}(\mu)$ doit être évaluée ou calculée pour de nombreuses instances de $\mu \in \mathcal{D}$. Même avec une méthode de discréttisation bien optimisée, le calcul de toutes ces solutions peut entraîner des coûts de calcul très élevés, rendant difficile la prise de décision en raison de temps de simulation trop longs pour obtenir des résultats fiables.

Dans ce mémoire, nous nous intéressons à la dynamique non linéaire des écoulements dans le cadre de l'extrusion de polymères. Nous explorons l'utilisation de modèles réduits basés sur des fonctions POD (Proper Orthogonal Decomposition), ainsi que la méthode de réduction de

dimensionnalité Autoencoder (AE), pour optimiser les temps pour différents paramètres rhéologiques et conditions aux limites. Ces approches visent à fournir des outils performants pour la construction d'un jumeau numérique de l'extrusion, ouvrant ainsi la voie à des avancées significatives dans la conception en fournissant des outils accessibles aux non experts en simulation avec des temps de calcul compatibles avec les métiers des bureaux d'études.

Objectif du stage

À cet égard, l'entreprise ACOME est considérée comme un fleuron technologique de l'industrie française des câbles, évoluant dans un environnement international concurrentiel. ACOME est un groupe industriel spécialisé dans la conception et la fabrication de fibres optiques et de câbles de haute technicité pour les réseaux d'infrastructures télécoms, les réseaux de données du bâtiment, ainsi que les réseaux embarqués automobiles et ferroviaires.



FIGURE 2 – Câble électrique dans le fief du spécialiste ACOME

Fondée en 1932, l'entreprise est aujourd'hui présente dans huit pays répartis sur quatre continents. ACOME est un acteur majeur de la filière câbles et connectique, avec un chiffre d'affaires de 535 millions d'euros en 2024 et 1700 collaborateurs. Son siège social est à Paris, mais c'est en Normandie, à Romagny-Fontenay, que l'entreprise possède les principales usines de ses six unités de production réparties dans les quatre coins du globe, toutes étant spécialisées dans un domaine spécifique de la fabrication de câbles. À ce niveau, la diversité

des domaines d'activité implique une diversité des matériaux utilisés, chaque matériau ayant ses propres propriétés rhéologiques, thermiques et mécaniques.

Cela nous amène à l'objectif du stage, qui consiste à construire un modèle réduit de la mécanique des fluides pour représenter le flux de matière dans l'outillage de mise en forme de l'enrobage d'un câble sur une ligne de production. Le modèle développé devra représenter l'essentiel de la physique de l'extrusion avec en sortie les champs de vitesse, pression et température dans une géométrie 2D/3D de l'outillage pour un large éventail de propriétés rhéologiques de matériaux.

À ce stade, on va décliner notre objectif en sous-objectifs :

- Exploration de la littérature sur l'extrusion de polymères et la modélisation mathématique des écoulements non linéaires.
- Construction des mailles sur Gmsh pour différents géométries de l'extrusion.
- Comprendre les outils d'OpenFoam pour la simulation de l'extrusion de polymères qui sont basés sur la méthode des volumes finis (MVF) et le langage de programmation C++.
- Développement d'un algorithme de simulation numérique pour l'extrusion de polymères, en tenant compte des interactions thermiques et mécaniques.
- Application des méthodes de réduction (POD, AE) aux résultats numériques obtenus par la simulation d'OpenFoam et en utilisant la librairie de modèles réduits EZyRB.
- Analyse et discussion des résultats obtenus, en mettant en évidence les avantages et les limites des modèles réduits par rapport aux simulations complètes.

Chapitre 1

Étude bibliographique et exploration de l'extrusion

Dans ce chapitre, nous allons effectuer une étude bibliographique sur l'extrusion de polymères. Nous aborderons les notions de transformation du problème physique en un modèle mathématique, en nous intéressant notamment aux équations de Navier-Stokes et à la thermique. Nous caractériserons également le nombre de Reynolds à partir d'une normalisation mathématique. Enfin, nous traiterons de la rhéologie des polymères et de leurs propriétés.

1.1 Extrusion de polymère

1.1.1 Principe de l'extrusion

La principale fonction de l'extrudeuse est de convoyer les granulés Figure 1.1 de polymère, de le fondre et de le mettre en pression, afin qu'il puisse franchir la filière placée à son extrémité. D'un point de vue industriel, l'objectif est d'obtenir à la sortie de la machine un débit régulier, avec un matériau homogène, à une température contrôlée, et dans des conditions de production optimales (débit maximal et consommation énergétique limitée)[7].

L'extrudeuse monovis Figure 1.2 courante se compose principalement des éléments suivants :

- **Trémie d'alimentation** : Approvisionnée avec le produit à extruder, généralement sous forme de granulés solides.



FIGURE 1.1 – Granulés de polymère

- **Vis sans fin (ou vis d'Archimète)** : Assure le transport, le mélange et la mise en pression du polymère.
 - **Fourreau** : Cylindre dans lequel tourne la vis sans fin, permettant le chauffage et le maintien de la température du polymère.
 - **Groupe d'entraînement de la vis** : Mécanisme qui entraîne la rotation de la vis.
 - **Filière** : Située à l'extrémité de l'extrudeuse, elle donne la forme finale au produit (gaines, tubes, plaques, profilés, etc.).
 - **Dispositifs de chauffage et de refroidissement** : Régulent la température du fourreau et, éventuellement, de la vis.
 - **Système de commande** : Permet de contrôler les différents paramètres de l'extrusion, tels que la vitesse de rotation, la température et la pression.
- La vis est responsable du transport de la polymère jusqu'à la filière, de sa plastification, de la mise en pression de la masse fondu, ainsi que de la régularité de la température et de la pression, qui conditionnent la régularité des dimensions du profilé.
- D'après les observations réalisées sur l'état du polymère dans la machine, on peut distinguer

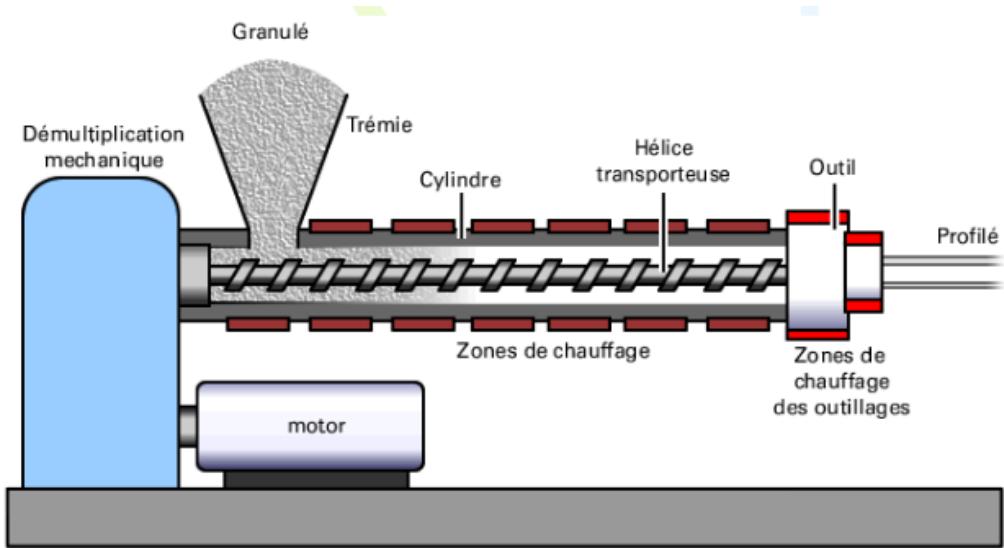


FIGURE 1.2 – Extrudeuse monovis[7]

trois zones phénoménologiques 1.3 :

- **Zone de convoyage solide** : Le polymère est entièrement solide.
- **Zone de fusion** : Coexistence de polymère encore solide et de polymère déjà fondu grâce au chauffage.
- **Zone de pompage** : Le polymère est totalement fondu et prêt à être extrudé.

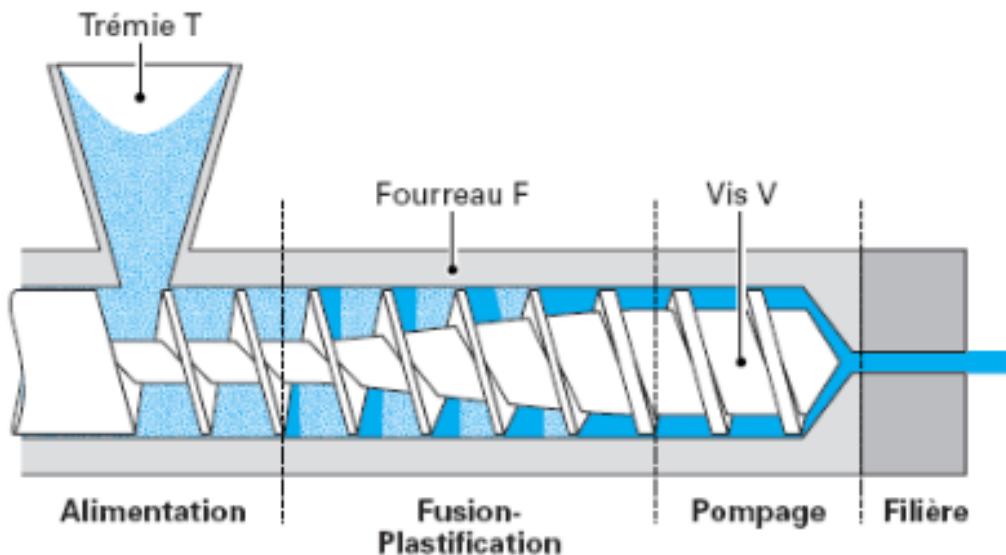


FIGURE 1.3 – Schéma simplifié d'une extrudeuse[7]

Les avantages de cette technologie sont nombreux. Elle permet d'importants gains de place et des économies d'énergie notables par rapport aux procédés traditionnels. De plus, les

extrudeuses offrent une grande adaptabilité : la quasi-totalité des paramètres du processus peut être ajustée en fonction du type de produit souhaité. Cela inclut la vitesse de rotation, la configuration des vis, la température, la pression, les débits de matière et le profil de l'outillage de mis en forme. Ces caractéristiques permettent de faire varier, à la demande, la texture, la forme et le taux d'humidité des produits. Toutefois, la maîtrise de ces paramètres reste complexe, et la texture des produits obtenus peut parfois être déficiente ou inégale. Ce procédé permet néanmoins de produire des pièces précises avec des états de surface excellents.

1.1.2 Description de la tête d'extrusion

La tête d'extrusion est un élément clé de l'extrudeuse, responsable de la transformation du polymère fondu en un profilé de forme et de dimensions spécifiques Figure 1.4 à gauche montre une tête d'extrusion typique coupé en deux pour en montrer l'intérieur et à droite est un exemple de tête d'extrusion utilisée pour l'extrusion de câbles chez ACOME.



Exemple chez ACOME

L'intérieur de la tête [lien](#)

FIGURE 1.4 – La tête d'extrusion utilisé pour l'extrusion de câbles chez ACOME.

- La filière a pour mission de donner la forme et les cotes voulues à l'extrudat dans les meilleures conditions de régularité et d'aspect.
- Si le profilé est creux, il faut utiliser, en plus, un poinçon pour obturer la partie correspondant à la cavité.

- La figure 1.3 illustre schématiquement le principe de l'extrusion d'un tube. On imagine bien que, si le poinçon est omis, on obtient un jonc de même diamètre extérieur que le tube.
- Le diamètre intérieur de la filière et le diamètre extérieur de l'extrudat n'ont pas exactement la même valeur, car le polymère, du fait de son retour à la pression atmosphérique dès sa sortie de la filière, se relaxe et donne lieu à une dilatation dite « gonflement à la filière ».
- En admettant que le polymère arrive à la filière dans de bonnes conditions de température, de viscosité et de pression, il est également nécessaire de concevoir la filière et le poinçon de manière à ce que la réduction de section entre le fourreau et l'entrefer filière/poinçon soit très progressive. Pour garantir que l'extrudat se forme régulièrement à la sortie de l'extrudeuse, il convient de veiller à obtenir des profils de vitesse et de température aussi uniformes que possible, ainsi qu'une pression stable dans le temps, sans fluctuations notables.

La tête d'extrusion a pour rôle de répartir de façon homogène la matière tout autour de la torpille, en prenant la forme poinçon/filière.

- L'entrefer est réglable, ce qui permet d'obtenir une épaisseur variable de l'extrudât.
- De l'air passe à l'intérieur de la torpille pour éviter que le tube ne s'écrase par aspiration.

1.2 Rhéologie des polymères

La rhéologie est définie comme "l'étude de la déformation et de l'écoulement de la matière", en particulier des polymères (plastiques et caoutchouc [1]). L'écoulement des matériaux à l'état liquide (ou fondu) dans une filière d'extrusion dépend entièrement de leurs propriétés mécaniques et rhéologiques. Ces propriétés sont étroitement liées à la viscosité, qui mesure le frottement entre les couches adjacentes du fluide lors de son écoulement. Mathématiquement, la viscosité est définie comme une relation entre la contrainte de cisaillement τ et le taux de cisaillement $\dot{\gamma}$

$$\tau = \mu \frac{\partial u}{\partial y} = \mu \dot{\gamma}, \quad (1.1)$$

La viscosité joue un rôle crucial dans l'écoulement du fluide, car elle conditionne son comportement lors de son déplacement. Dans le cas des thermoplastiques à l'état liquide, ces derniers se comportent comme des fluides non-newtoniens, car leur viscosité dépend du taux de cisaillement $\dot{\gamma}$ et du temps.

1.2.1 Fluides non-newtoniens

La modélisation de la viscosité dans les fluides non-newtoniens peut dépendre de plusieurs paramètres comme le taux de cisaillement et la température pour les polymères fondus. De plus, certains critères physiques et mathématiques doivent être pris en compte pour garantir la validité du modèle. Par conséquent, un modèle universel pour tous les comportements non-newtoniens ne serait pas réaliste. Par exemple, les lois de Carreau-Yasuda, de puissance (Power-law) ou encore la loi de Cross-Power-law sont couramment utilisées pour décrire la dépendance de la viscosité au taux de cisaillement. Dans ces modèles, les paramètres sont généralement déterminés par des mesures rhéométriques. Parmi ces paramètres, on trouve notamment le paramètre n , appelé indice d'écoulement, dont la valeur peut varier en fonction du comportement du fluide. On peut alors les classer de la manière suivante :

- $n < 1$: comportement pseudoplastique (amincissement par cisaillement),
- $n = 1$: comportement newtonien,
- $n > 1$: comportement dilatant (épaississement par cisaillement).

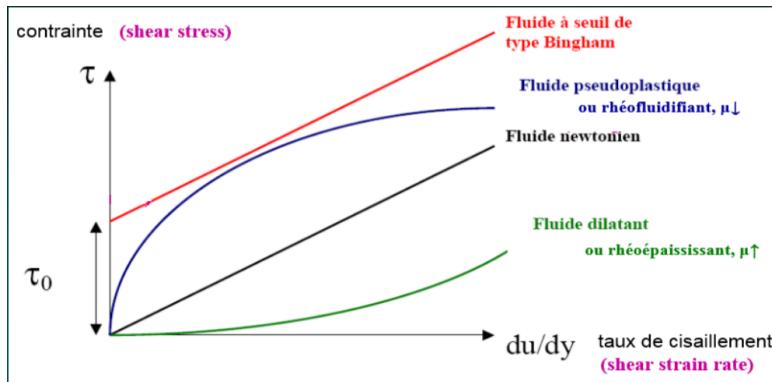


FIGURE 1.5 – Dépendance de la contrainte de cisaillement par rapport au taux de cisaillement pour différents fluides [lien](#).

On peut observer cette dépendance sur la figure 1.5, qui illustre la variation de la contrainte de cisaillement τ en fonction du taux de cisaillement $\dot{\gamma}$ pour différents fluides. On constate que, pour les fluides pseudoplastiques, la contrainte de cisaillement augmente moins rapidement avec le taux de cisaillement (amincissement par cisaillement), tandis que pour les fluides dilatants, elle augmente plus rapidement (épaississement par cisaillement). Les valeurs des autres paramètres du modèle dépendent des propriétés du matériau, telles que sa rigidité, sa température et sa structure interne. D'un point de vue physique et mathématique, il est

important que le taux de cisaillement $\dot{\gamma}$ soit constraint entre une limite inférieure et une limite supérieure, et que la valeur de n reste strictement positive.

1.2.2 Polymère fondu et le rhéomètre capillaire

Le polymère fondu est un matériau viscoélastique qui présente un comportement complexe lors de son écoulement. Chez ACOME, nous travaillons principalement avec la première catégorie, à savoir les polymères fluidifiants (pour lesquels $n < 1$). Dans ce cadre, les vitesses de cisaillement sont généralement faibles, ce qui implique un nombre de Reynolds bas (nous détaillerons ce point dans la section suivante) et conduit à un régime d'écoulement laminaire.



FIGURE 1.6 – Dépendance de la contrainte de cisaillement par rapport au taux de cisaillement pour différents fluides.[lien](#)

Pour caractériser ce comportement, nous utilisons un rhéomètre capillaire, illustré à la figure 1.6, qui est un instrument permettant de mesurer la viscosité des fluides. Il permet de déterminer la viscosité dynamique du polymère fondu en fonction du taux de cisaillement et de la température.

Le rhéomètre capillaire utilisé chez ACOME composé de deux canaux (Twin bore), permettant de réaliser des essais simultanés avec une filière capillaire longue (shear die) et une filière orifice (orifice die), cette dernière étant utilisée pour la correction de Bagley (prise en compte de la pression d'entrée), souvent il n'y a qu'un seul canal, ce qui oblige à faire des mesures avec deux filières différentes pour faire l'équivalent de ce qui est fait en une fois sur un rhéomètre à deux canaux. Pour mesurer la viscosité, des granulés de polymère préchauffés sont injectés dans le rhéomètre, qui les fait passer à travers une filière de petit diamètre. La pression nécessaire pour faire passer le polymère à travers la filière est mesurée, et la viscosité est calculée en fonction de cette pression et de la température.

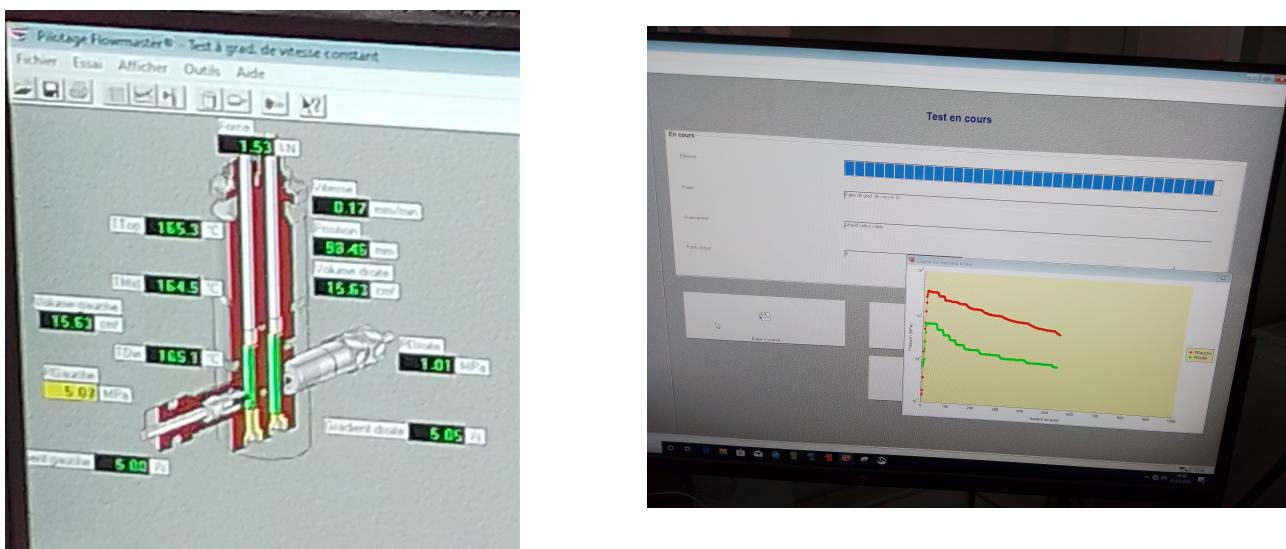


FIGURE 1.7 – Plateforme de rhéométrie capillaire chez ACOME.

Étant donné la diversité des matériaux utilisés, différents tests sont réalisés à diverses températures et pour différents taux de cisaillement. Après une analyse statistique approfondie des résultats obtenus, les paramètres les plus adaptés sont sélectionnés pour la simulation et la loi de comportement à utiliser.

1.2.3 Modèle de Carreau-Yasuda

Le modèle de Carreau-Yasuda est un modèle rhéologique couramment utilisé dans l'industrie pour décrire le comportement des fluides non-newtoniens, en particulier les polymères fondus. Il relie la viscosité dynamique μ au taux de cisaillement $\dot{\gamma}$ et à la température T à l'aide de ces équations suivantes :

$$\mu(\dot{\gamma}, T) = \mu_\infty \alpha_T + \alpha_T (\mu_0 - \mu_\infty) [1 + (\lambda \alpha_T \dot{\gamma})^a]^{\frac{n-1}{a}}. \quad (1.2)$$

$$\alpha_T = \exp(-b(T - T_0)) \quad (1.3)$$

Dans ces équations :

- μ_0 est la viscosité à faible taux de cisaillement (viscosité au repos),
- μ_∞ est la viscosité à haut taux de cisaillement,
- λ est un paramètre de temps caractéristique,
- a et n sont des paramètres du modèle de Carreau-Yasuda,
- $\dot{\gamma}$ est le taux de cisaillement,
- b coefficient de dépendance de la température,
- T_0 est la température de référence.

Ce modèle est particulièrement adapté pour les polymères, car il capture à la fois le plateau de viscosité à faible taux de cisaillement et la diminution de la viscosité à haut taux de cisaillement (effet d'amincissement par cisaillement). Ce comportement est essentiel pour modéliser correctement les écoulements dans l'extrudeuse, où le polymère est soumis à des gradients de vitesse importants.([8],[9])

1.2.4 Paramètres physiques

Les paramètres physiques du modèle de Carreau-Yasuda sont généralement déterminés à partir de mesures rhéométriques. Au sein de l'entreprise ACOME, un rhéomètre capillaire (voir figure 1.6) est utilisé pour obtenir les données expérimentales nécessaires à l'identification de cette loi. La calibration des paramètres du modèle est ensuite réalisée à l'aide du logiciel CompuPlast, qui permet d'ajuster précisément les courbes de viscosité en fonction des résultats expérimentaux.

Cette calibration repose sur plusieurs hypothèses de base, telles que l'homogénéité du matériau, la validité du modèle de Carreau-Yasuda sur la plage de taux de cisaillement et de

températures étudiée, ainsi que l'absence d'effets de glissement aux parois lors des mesures. Les paramètres obtenus sont ensuite utilisés dans nos simulations numériques, où ils sont comparés aux résultats expérimentaux afin de contrôler l'erreur et d'optimiser les algorithmes et approches de simulation. Ces travaux visent à étudier l'impact de la diversité de la rhéologie des matériaux sur le comportement à l'extrusion.

Dans le tableau 1.1 ci-dessous, nous présentons les paramètres de la loi de Carreau-Yasuda utilisés dans nos simulations pour différentes matériaux avec ses courbes de viscosité en fonction du taux de cisaillement dans la figure 1.8.

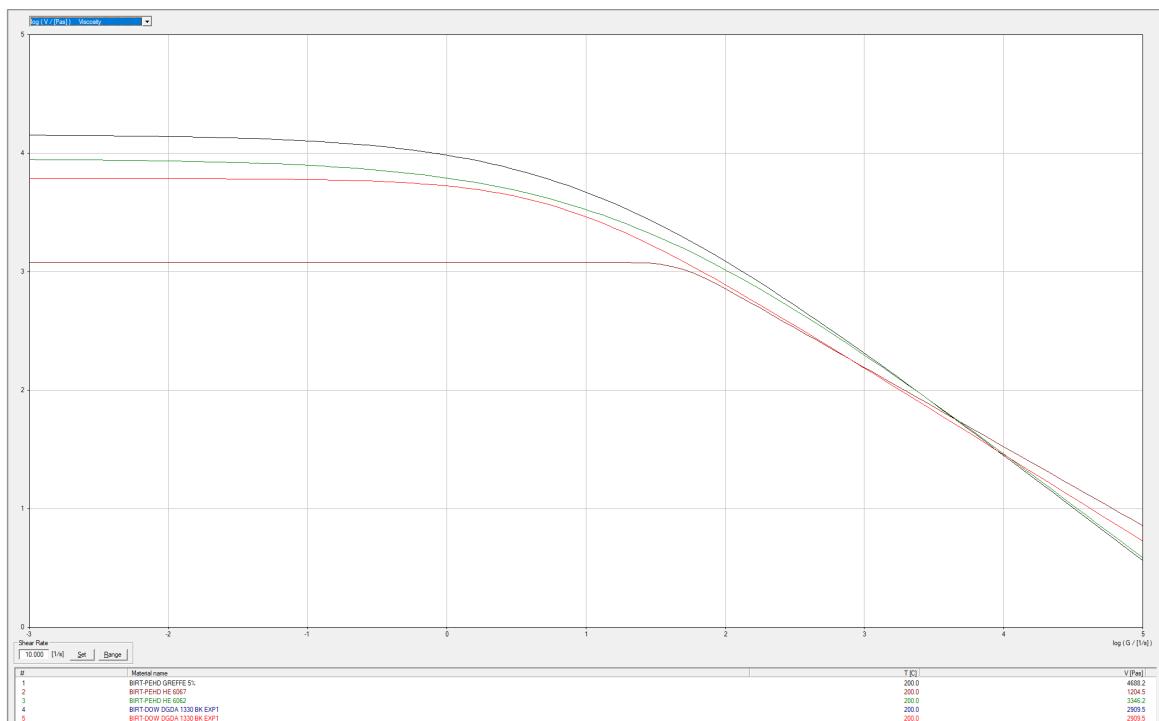


FIGURE 1.8 – Dépendance de la contrainte de cisaillement par rapport au taux de cisaillement pour différents matériaux utilisés dans cette étude.

1.3 Modélisation physique mathématique du problème

L'extrusion de polymères est un processus complexe qui implique des interactions thermiques, mécaniques et hydrodynamiques. Ce procédé consiste à faire passer un polymère fondu à travers une filière pour lui donner une forme spécifique. La modélisation mathématique de ce problème repose sur la description des écoulements du polymère, qui est généralement

TABLE 1.1 – Paramètres de la loi de Carreau-Yasuda pour différents matériaux.

Matériaux	Paramètre	Valeur	Unité
BIRT-DOW	μ_0	7325.89	Pa · s
	λ	0.18211	s
	n	0.26850	-
	a	0.88184	-
	b	0.01805	K ⁻¹
BIRT -PEHD	μ_0	10715.5	Pa · s
	λ	0.06409	s
	n	0.1	-
	a	0.50188	-
	b	0.01794	K ⁻¹
BIRT -PEHD	μ_0	1687.88	Pa · s
	λ	0.03003	s
	n	0.33230	-
	a	3.9756	-
	b	0.03370	K ⁻¹
GREFFE 5%	μ_0	17111.9	Pa · s
	λ	0.11656	s
	n	0.1	-
	a	0.55329	-
	b	0.01776	K ⁻¹
Paramètres fixes	T_0	463.15	K
	μ_∞	0	Pa · s
	ρ	770	Kg.m ⁻³
	κ	0.25	W.Kg ⁻¹ .K ⁻¹
	C_p	2250	J.Kg ⁻¹ .K ⁻¹

considéré comme un fluide visqueux et incompressible.

1.3.1 Équations fondamentales de la mécanique des fluides

Les équations qui décrivent le comportement d'un fluide sont un ensemble d'équations aux dérivées partielles non linéaires. Ces équations régissent des phénomènes impliquant des fluides non-newtoniens. Elles se composent de trois équations principales l'équation de continuité, qui représente la conservation de la masse dans un domaine ; l'équation de la conservation de la quantité de mouvement ou équation de Navier-Stokes, qui représente la deuxième loi de Newton appliquée à un fluide ; l'équation de l'énergie, qui permet d'analyser le transfert de chaleur dans le fluide, et la loi de Carreau Yasuda 1.2 , qui décrit l'interaction entre le fluide et la thermique et pour l'effet de la température la loi d'Arrhenius 1.3 . Les équations sont présentées ci-dessous :

Équation de continuité (conservation de la masse)

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \quad (1.4)$$

Équation de Navier-Stokes (conservation de la quantité de mouvement)

$$\frac{\partial(\rho \mathbf{u})}{\partial t} + \rho \mathbf{u} \cdot \nabla(\mathbf{u}) = -\nabla p + \rho \mathbf{g} + \nabla \cdot \bar{\tau} \quad (1.5)$$

Équation de l'énergie (conservation de l'énergie thermique)

$$\rho C_p \frac{\partial T}{\partial t} + \rho C_p \mathbf{u} \cdot \nabla T = \nabla \cdot (\kappa \nabla T) + \hat{\Phi} \quad (1.6)$$

où :

- ρ est la densité du fluide,
- t représente la variable temporelle,
- p est la pression,
- \mathbf{g} représente l'accélération gravitationnelle,
- $\bar{\tau}$ est le tenseur des contraintes de cisaillement,
- κ représente la conductivité thermique du fluide,
- T est la température,
- C_p est la capacité thermique massique,
- $\hat{\Phi}$ représente le terme de source de chaleur.

Ces équations permettent de modéliser le comportement du polymère fondu dans l'extrudeuse, en tenant compte des contraintes imposées par la géométrie de la filière et les conditions aux limites (par exemple, la vitesse imposée à l'entrée et la pression à la sortie). La résolution de ces équations, souvent réalisée à l'aide de méthodes numériques comme les volumes finis, permet de prédire les profils de vitesse, de pression et de température, essentiels pour optimiser le processus d'extrusion.

1.3.2 Incompressibilité et équation de continuité

Un fluide incompressible est un fluide dans lequel la densité d'une particule donnée de fluide reste constante, c'est-à-dire :

$$\frac{D\rho}{Dt} = 0. \quad (1.7)$$

Il est important de noter qu'un fluide incompressible n'implique pas nécessairement que la densité soit uniforme partout dans l'écoulement. Par exemple, un écoulement stratifié dans l'océan peut présenter des variations de densité entre différentes couches d'eau en raison des variations de salinité et de température. Cependant, un fluide avec une densité constante et uniforme est également considéré comme incompressible.

L'équation de continuité pour un fluide incompressible peut être dérivée en utilisant l'Eq. (1.7) :

$$\frac{D\rho}{Dt} = 0 = \frac{\partial\rho}{\partial t} + (\mathbf{u} \cdot \nabla)\rho \implies \frac{\partial\rho}{\partial t} = -(\mathbf{u} \cdot \nabla)\rho. \quad (1.8)$$

En substituant cette relation dans l'équation de continuité générale (1.4) :

$$\frac{\partial\rho}{\partial t} + \nabla \cdot (\rho\mathbf{u}) = 0, \quad (1.9)$$

on obtient :

$$-(\mathbf{u} \cdot \nabla)\rho + \nabla \cdot (\rho\mathbf{u}) = 0, \quad (1.10)$$

$$-(\mathbf{u} \cdot \nabla)\rho + \rho(\nabla \cdot \mathbf{u}) + (\mathbf{u} \cdot \nabla)\rho = 0, \quad (1.11)$$

$$\nabla \cdot \mathbf{u} = 0. \quad (1.12)$$

Ainsi, un fluide incompressible satisfait la même équation de continuité qu'un fluide avec une densité constante et uniforme.([10])

1.3.3 Conservation de la quantité de mouvement

L'équation de conservation de la quantité de mouvement pour les écoulements incompressibles, exprimée sous forme différentielle (1.5), peut être simplifiée en tenant compte des hypothèses formulées dans la section précédente. En supposant que le fluide est incompressible, la densité ρ est constante. De plus, l'écoulement visqueux, lourd et incompressible alors ($\mathbf{g} = 0$), donc le deuxième terme du côté droit de l'équation peut être négligé.

Le troisième terme du côté droit, $\bar{\tau}$, représente le tenseur des contraintes de cisaillement.

Lorsque le fluide n'est pas au repos, des variations de vitesse produisent du cisaillement (et/ou de l'elongation), qui sont contrebalancées par des forces de résistance visqueuses. Dans le cadre de l'extrusion de polymères, les écoulements sont laminaires et se produisent à de faibles nombres de Reynolds, ce qui est un élément essentiel pour la modélisation de l'extrusion. Le nombre de Reynolds est défini comme suit :

$$\text{Re} = \frac{\rho U L}{\mu}, \quad (1.13)$$

où U est la vitesse de l'écoulement, L une longueur caractéristique, et μ la viscosité dynamique du fluide. Dans notre cas, ce nombre doit être petit, c'est-à-dire $\text{Re} \ll 1$. En effet, dans le cas de l'extrusion de polymères chez ACOME, nous avons $L = 3.15 \times 10^{-4}$ m, $U = 480$ m/min, $\mu = 15219.5$ Pa · s et $\rho = 936$ kg/m³. Alors le nombre de Reynolds est :

$$\text{Re} = \frac{936 \times 480 \times 3.15 \times 10^{-4}}{15219.5 \times 60} \approx 1.5498 \times 10^{-4} \ll 1. \quad (1.14)$$

En considérant que le fluide est non-newtonien, le tenseur des contraintes de cisaillement $\bar{\tau}$ exprimé en fonction du gradient de vitesse $\nabla \mathbf{u}$ et de la viscosité dynamique $\mu(\dot{\gamma})$ ([11]) :

$$\bar{\tau} = 2\mu(\dot{\gamma}, T)\mathbf{D}, \quad (1.15)$$

où $\dot{\gamma}$ est le taux de cisaillement, et \mathbf{D} est le tenseur de déformation défini comme suit :

$$\mathbf{D} = \frac{1}{2} (\nabla \mathbf{u} + (\nabla \mathbf{u})^T). \quad (1.16)$$

Après simplification, l'équation de la quantité de mouvement peut être écrite sous la forme suivante :

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nabla \cdot (\mu(\dot{\gamma}, T)(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)). \quad (1.17)$$

En termes de contributions physiques, cette équation peut également être exprimée sous la forme suivante :

$$\{\text{Forces d'inertie}\} = \{\text{Forces de pression}\} + \{\text{Forces de contrainte}\}. \quad (1.18)$$

Et puisque le nombre de Reynolds est faible, on peut négliger le terme d'inertie alors l'équation de Navier-Stokes devient :

$$\nabla p = \nabla \cdot (\mu(\dot{\gamma}, T)(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)). \quad (1.19)$$

Exemple 1.3.1 (Adimensionalisation et réduction de l'équation de Navier-Stokes). Soit $U \in \mathbb{R}$ une vitesse caractéristique de l'écoulement étudié (par exemple liée à la vitesse du cable) et L une longueur caractéristique (par exemple le diamètre de la tête). On considère le temps caractéristique $T = \frac{L}{U}$ et on pose les variables adimensionnées suivantes :

$$\tilde{x} = \frac{x}{L}, \quad \tilde{t} = \frac{t}{T}, \quad \tilde{u}(\tilde{x}, \tilde{t}) = \frac{u(x, t)}{U}, \quad \tilde{p}(\tilde{x}, \tilde{t}) = \frac{p(x, t)}{\rho U^2}$$

Les nouvelles vitesse et pression \tilde{u} et \tilde{p} vérifient alors :

$$\rho \left(\frac{U}{L} \frac{\partial \tilde{u}}{\partial \tilde{t}} + \frac{U^2}{L} (\tilde{u} \cdot \nabla) \tilde{u} \right) = - \frac{U^2}{L} \nabla \tilde{p} + \frac{1}{L} \nabla \cdot (\mu(\dot{\gamma}, T)(\nabla \tilde{u} + (\nabla \tilde{u})^T))$$

Les opérateurs différentiels ∇ et Δ ci-dessus sont relatifs à la variable adimensionnée \tilde{x} .

On obtient ainsi :

$$\frac{\partial \tilde{u}}{\partial \tilde{t}} + (\tilde{u} \cdot \nabla) \tilde{u} = - \nabla \tilde{p} + \frac{1}{Re} \nabla \cdot (\mu(\dot{\gamma}, T)(\nabla \tilde{u} + (\nabla \tilde{u})^T)) \quad (1.20)$$

$$\nabla \cdot \tilde{u} = 0 \quad (1.21)$$

avec Re le nombre de Reynolds défini par :

$$Re = \rho \frac{LU}{\mu}$$

Le nombre de Reynolds caractérise le type d'écoulement étudié. Plus le nombre de Reynolds est petit, plus les forces de viscosité sont importantes et les effets inertIELS négligeables. À l'inverse, plus le nombre de Reynolds est grand, plus les forces d'inertIE sont importantes. Pour une présentation générale et approfondie de l'équation de Navier-Stokes, on pourra se référer à [39].

À partir des équations de Navier-Stokes, on obtient :

- Les équations de **Stokes** pour $Re \ll 1$ (régime laminaire)
- Les équations d'**Euler** pour $Re \gg 1$ (régime turbulent)

Cas $Re \ll 1$ (Stokes) : Si on pose $p' = LU\rho\tilde{p} = \nu Re\tilde{p}$, l'équation devient :

$$Re\left(\frac{\partial u}{\partial t} + (u \cdot \nabla)u\right) = -\frac{1}{\nu}\nabla p' + \nabla \cdot (\mu(\dot{\gamma}, T)(\nabla u + (\nabla u)^T))$$

En faisant tendre $Re \rightarrow 0$ et puisque les effets d'inerties égigeables devant les forces de viscosité, on obtient les équations de Stokes stationnaires :

$$-\nabla p + \nabla \cdot (\mu(\dot{\gamma}, T)(\nabla u + (\nabla u)^T)) = 0 \quad (1.22)$$

$$\nabla \cdot u = 0 \quad (1.23)$$

Cas $Re \gg 1$ (Euler) : Le terme de convection non linéaire $(u \cdot \nabla)u$ est dominant. En faisant tendre $Re \rightarrow +\infty$ (ou en prenant directement $\mu = 0$), on obtient les équations d'Euler :

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u + \nabla p = 0 \quad (1.24)$$

$$\nabla \cdot u = 0 \quad (1.25)$$

1.3.4 Conservation de l'énergie

L'extrusion est un processus non isotherme comme nous l'avons expliqué dans la section de la rhéologie, ce qui signifie que la température du polymère fondu varie tout au long de l'extrusion, ce qui souligne l'important de la prise en compte de la thermique pour que la simulation soit pertinente pour aider à la maîtrise de la qualité de l'extrusion. Par conséquent, il est nécessaire de considérer l'équation de l'énergie (1.6) pour modéliser le transfert de chaleur dans le polymère. Cette équation est souvent couplée à l'équation de Navier-Stokes afin de prendre en compte l'interaction entre les phénomènes thermiques et hydrodynamiques. L'équation de l'énergie contient quatre termes principaux qui contribuent physiquement au transfert de chaleur :

$$\left\{ \begin{array}{l} \text{Variation} \\ \text{d'énergie} \\ \text{interne} \end{array} \right\} + \left\{ \begin{array}{l} \text{Transport} \\ \text{convectif} \end{array} \right\} = \left\{ \begin{array}{l} \text{Transfert de} \\ \text{chaleur par} \\ \text{conduction} \end{array} \right\} + \left\{ \begin{array}{l} \text{Dissipation} \\ \text{visqueuse} \end{array} \right\}. \quad (1.26)$$

Le terme de dissipation visqueuse $\hat{\Phi}$ représente l'énergie mécanique du fluide transformée en énergie interne. Ce terme joue un rôle essentiel dans le cas de l'extrusion de polymères ([11]).

Pour calculer l'auto-échauffement générer par le cisaillement de la matière, on peut utiliser la relation suivante :

$$\hat{\Phi} = \bar{\tau} : \nabla \mathbf{u} = \frac{1}{2} \mu(\dot{\gamma} : \dot{\gamma, T}) = \eta \dot{\gamma}^2, \quad (1.27)$$

où η est la viscosité dynamique du fluide et $\dot{\gamma}$ est le taux de cisaillement.

Le terme $\hat{\Phi}$ est souvent considéré comme une source de chaleur dans l'équation de l'énergie. Il représente la chaleur générée par la dissipation visqueuse due à la déformation du fluide. En effet, lorsque le fluide s'écoule, il subit des déformations qui entraînent des pertes d'énergie sous forme de chaleur. Ce terme est particulièrement important dans les écoulements non newtoniens, où la viscosité dépend du taux de cisaillement.

Après division de l'équation de l'énergie par la masse volumique ρ multipliée par la capacité thermique massique C_p , elle devient :

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = \alpha \Delta T + \frac{\mu(\dot{\gamma}, T)}{\rho C_p} \dot{\gamma}^2, \quad (1.28)$$

où α est la diffusivité thermique du fluide. Nous pouvons également écrire l'équation de l'énergie en fonction de la loi de comportement rhéologique cela devient :

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = \alpha \Delta T + \frac{\mu_\infty \alpha_T}{C_p} \dot{\gamma}^2 + \frac{(\mu_0 - \mu_\infty) \alpha_T}{C_p} [1 + (\lambda \dot{\gamma})^2]^{\frac{n-1}{2}} \dot{\gamma}^2, \quad (1.29)$$

Chapitre 2

Méthodes de simulation

2.1 Méthodes de volume fini

La méthode des volumes finis est une technique numérique largement utilisée pour résoudre les équations aux dérivées partielles (EDP) qui modélisent des phénomènes physiques tels que la mécanique des fluides, la chaleur et la diffusion. Cette méthode repose sur le principe de conservation de la quantité de mouvement et de l'énergie dans un volume de contrôle discret. Elle est particulièrement adaptée aux problèmes où les solutions peuvent présenter des discontinuités ou des gradients importants. La méthode des volumes finis consiste à diviser le domaine de calcul en un maillage constitué de volumes de contrôle. Pour chaque volume, on applique le principe de conservation à la quantité d'intérêt (par exemple, la masse, la quantité de mouvement ou l'énergie) en intégrant les équations de conservation sur ce volume. Cela permet de relier les flux entrants et sortants à travers les faces du volume de contrôle. Cette approche garantit que les quantités conservées sont correctement prises en compte, même en présence de discontinuités, sous réserve d'une erreur de discrétisation. De plus, la méthode des volumes finis est particulièrement adaptée aux géométries complexes et aux maillages non structurés, ce qui en fait un choix privilégié pour de nombreuses applications en ingénierie et en sciences appliquées. Dans le cadre de l'extrusion de polymères, la méthode des volumes finis est utilisée pour résoudre les équations de Navier-Stokes et l'équation de la chaleur. Elle permet de modéliser le comportement des fluides non newtoniens dans des géométries complexes, telles que les filières d'extrusion. En utilisant cette méthode, il est possible de simuler avec précision les profils de vitesse, de pression et de température du polymère fondu.

dans la filière d'extrusion. En parallèle, il existe aussi la méthode des éléments finis (FEM), qui est une autre approche numérique utilisée pour résoudre les EDP et qui est particulièrement adaptée aux problèmes de mécanique des structures et de thermique par exemple dans cette article [43]. La méthode des éléments finis divise le domaine en éléments finis et utilise des fonctions de forme pour approximer la solution dans chaque élément. Cette méthode est utilisée dans des logiciels de simulation comme COMSOL, ANSYS Polyflow, etc. Dans notre travail, nous allons utiliser OpenFOAM qui se base sur la méthode des volumes finis.

2.2 OpenFOAM

Pour les simulations de CFD (Computational Fluid Dynamics) en mécanique des fluides, il existe de nombreux logiciels commerciaux et open source permettant de simuler les écoulements de polymères rhéofluidifiants, tels qu'ANSYS POLYFLOW, COMSOL, ou encore COMPUPLAST, ce dernier étant utilisé chez ACOME. Parmi ces solutions, OpenFOAM (Open Field Operation and Manipulation) est une boîte à outils de simulation multi-physiques, principalement dédiée à la résolution des équations de la mécanique des fluides. Distribué depuis 2004 sous licence GNU GPL par la société britannique OpenCFD Ltd (acquise par SGI en 2011, puis par ESI Group en 2012), ce logiciel open source a été initialement développé en C++ à l'Imperial College London pour exploiter la méthode des volumes finis et les avancées récentes en programmation.

OpenFOAM permet de traiter des écoulements complexes, incluant la turbulence, le transfert de chaleur, les réactions chimiques, ainsi que les écoulements multiphasiques.

OpenFOAM propose également des outils de maillage, de pré- et post-traitement, ainsi que la possibilité d'effectuer des calculs en parallèle. Sa grande flexibilité permet à l'utilisateur de modifier ou d'ajouter des fonctionnalités, ce qui constitue un avantage notable face aux solutions commerciales[16].

2.2.1 La méthode des volumes finis dans OpenFOAM

La méthode des volumes finis (FVM) est la technique de discréétisation la plus couramment utilisée en dynamique des fluides numérique. Pour illustrer son fonctionnement, prenons

comme exemple l'équation de transport scalaire pour une variable dépendante ϕ :

$$\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{\text{terme transitoire}} + \underbrace{\nabla \cdot (\rho \mathbf{u} \phi)}_{\text{terme convectif}} = \underbrace{\nabla \cdot (\Gamma \nabla \phi)}_{\text{terme diffusif}} + \underbrace{S}_{\text{terme source}} \quad (2.1)$$

Comme mentionné précédemment, dans cette méthode, le domaine d'étude (par exemple, le canal rectangulaire illustré à la Figure 2.1) est divisé en plusieurs volumes de contrôle.

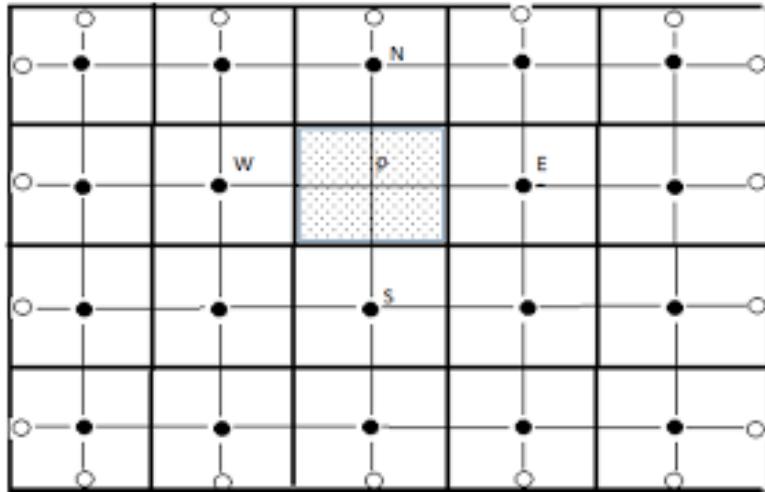


FIGURE 2.1 – Discrétisation du domaine d'étude en volumes de contrôle.

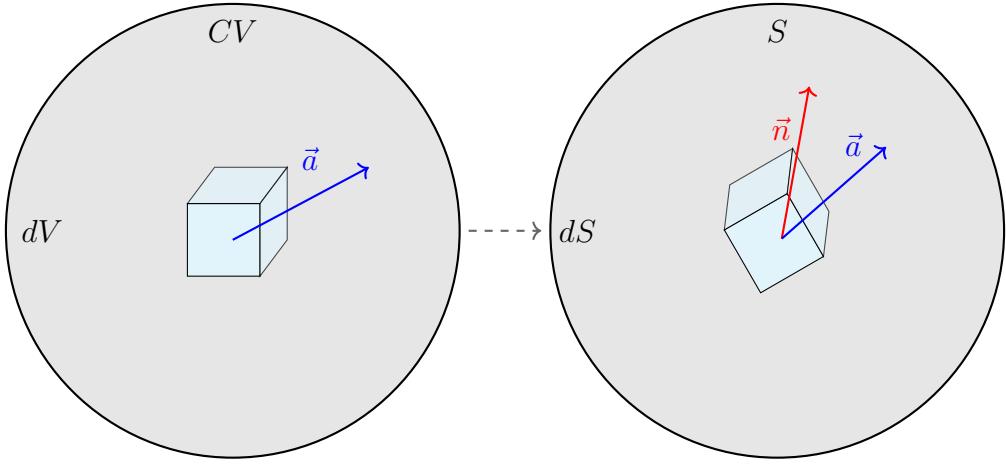
Les équations fondamentales sont ensuite intégrées sur l'espace (et sur le temps pour les problèmes transitoires), ce qui permet d'assurer la conservation des grandeurs dans chaque cellule :

$$\int_{CV} \frac{\partial(\rho\phi)}{\partial t} dV + \int_{CV} \nabla \cdot (\rho \mathbf{u} \phi) dV = \int_{CV} \nabla \cdot (\Gamma \nabla \phi) dV + \int_{CV} S dV \quad (2.2)$$

Pour un problème stationnaire, le terme transitoire de l'équation disparaît. Si le problème est instationnaire, il faut également intégrer l'équation sur le temps.

Afin de simplifier les termes convectif et diffusif, on utilise le théorème de Gauss (illustré à la Figure 2.2), qui permet de transformer les intégrales volumiques en intégrales de surface. Ce théorème indique qu'il est possible d'étudier le comportement de la variable sur la surface du volume de contrôle, plutôt que dans tout le volume.

En appliquant le théorème de Gauss, l'équation précédente (pour un problème stationnaire) peut s'écrire :



$$\int_{CV} \nabla \cdot \vec{a} dV = \int_S \vec{a} \cdot \vec{n} dS$$

FIGURE 2.2 – Illustration du théorème de Gauss.

$$\int_S \rho\phi \mathbf{u} \cdot \mathbf{n} dS = \int_S \Gamma \nabla \phi \cdot \mathbf{n} dS + \int_{CV} S dV \quad (2.3)$$

Pour chaque volume de contrôle, on obtient une équation similaire, et l'ensemble de ces équations doit être discrétisé, ce qui conduit à un système d'équations algébriques.

L'approximation de l'intégrale est simple : on suppose que les valeurs des variables sous l'intégrale sont connues au centre des faces du volume de contrôle.

Par exemple, pour le volume de contrôle o , on a :

$$\int_S \rho\phi \mathbf{u} \cdot \mathbf{n} dS \approx A(\rho\phi u)_e - A(\rho\phi u)_w + A(\rho\phi v)_n - A(\rho\phi v)_s$$

et pour le terme diffusif :

$$\int_S \Gamma \nabla \phi \cdot \mathbf{n} dS \approx A(\Gamma \nabla \phi)_e - A(\Gamma \nabla \phi)_w + A(\Gamma \nabla \phi)_n - A(\Gamma \nabla \phi)_s$$

où e, w, n, s désignent respectivement les faces est, ouest, nord et sud du volume de contrôle, A surface de la face et pour les signes dépendent de l'orientation. Cela signifie qu'il faut approximer \mathbf{u} , ϕ et $\nabla \phi$ aux faces du volume de contrôle, puisque toutes les variables sont placées au centre du volume de contrôle (en supposant Γ constant).

Par exemple, en considérant que la surface de chaque face (A) est identique pour tous les volumes de contrôle, ainsi que la distance entre les nœuds (δ), et en utilisant l'équation

précédente, l'équation pour ϕ au nœud o , avec l'approximation de la méthode des différences centrales (CDS) (en supposant le champ de vitesse connu), s'écrit :

$$\begin{aligned} & \rho \left[\frac{A}{2} u_e (\phi_e + \phi_o) - \frac{A}{2} u_w (\phi_o + \phi_w) + \frac{A}{2} v_n (\phi_n + \phi_o) - \frac{A}{2} v_s (\phi_s + \phi_o) \right] \\ &= \Gamma \left[\frac{A}{\delta} (\phi_e - \phi_o) - \frac{A}{\delta} (\phi_o - \phi_w) + \frac{A}{\delta} (\phi_n - \phi_o) - \frac{A}{\delta} (\phi_o - \phi_s) \right] + S \Delta V \end{aligned}$$

où u et v représentent respectivement les composantes de la vitesse selon les axes x et y . Le flux à travers une face, par exemple à l'est, s'écrit $flux = \rho u \phi_e \cdot A$, où l'on approxime généralement $\phi_e \approx \frac{\phi_e + \phi_o}{2}$, et de même pour les autres faces. Si l'on suppose que le terme source ne dépend pas de la variable au nœud o , c'est-à-dire $S_u \Delta V = S$, l'équation peut alors être simplifiée comme suit :

$$a_P \phi_o = a_E \phi_e + a_W \phi_w + a_N \phi_n + a_S \phi_s + S$$

où les coefficients a_P, a_E, a_W, a_N, a_S sont définis en fonction des paramètres physiques et géométriques du problème.

À noter que l'équation précédente s'écrit sous forme canonique :

$$a_P \phi_P = a_E \phi_E + a_W \phi_W + a_N \phi_N + a_S \phi_S + S_u \quad (2.4)$$

où les indices P, E, W, N, S désignent respectivement le nœud central discréte et ses voisins Est, Ouest, Nord et Sud.

Cette équation a été obtenue en utilisant des approximations pour les dérivées et la vitesse aux faces des cellules. Dans la bibliothèque de calcul OpenFOAM®, il est possible de choisir les schémas de discréétisation à utiliser pour chaque terme de l'équation (fichier `fvSchemes`).

2.2.2 Résolution numérique du système d'équations

Comme mentionné précédemment, après la discréétisation de l'équation 2.3, on obtient un système d'équations algébriques, qui peut être écrit sous la forme matricielle suivante :

$$\left[\begin{array}{cccccccccccccc}
a_P & -a_E & 0 & 0 & -a_S & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & \phi_1 & S_{u,1} \\
-a_W & a_P & -a_E & 0 & 0 & -a_S & 0 & \cdots & \ddots & \ddots & \ddots & \vdots & \phi_2 & S_{u,2} \\
0 & -a_W & a_P & -a_E & 0 & 0 & -a_S & 0 & \cdots & \ddots & \ddots & \vdots & \phi_3 & S_{u,3} \\
0 & 0 & -a_W & a_P & 0 & 0 & 0 & -a_S & 0 & \cdots & \ddots & \vdots & \phi_4 & S_{u,4} \\
-a_N & 0 & 0 & 0 & a_P & -a_E & 0 & 0 & -a_S & 0 & \cdots & \vdots & \vdots & \vdots \\
0 & -a_N & 0 & 0 & -a_W & a_P & -a_E & 0 & 0 & -a_S & 0 & \vdots & \vdots & \vdots \\
\vdots & \ddots & \vdots & \vdots & \vdots \\
0 & \cdots & 0 & -a_N & 0 & 0 & -a_W & a_P & -a_E & 0 & 0 & -a_S & \vdots & \vdots & \vdots \\
\vdots & \ddots & \cdots & 0 & -a_N & 0 & 0 & 0 & a_P & -a_E & 0 & 0 & \phi_{13} & S_{u,13} \\
\vdots & \ddots & \ddots & \cdots & 0 & -a_N & 0 & 0 & -a_W & a_P & -a_E & 0 & \phi_{14} & S_{u,14} \\
\vdots & \ddots & \ddots & \ddots & \cdots & 0 & -a_N & 0 & 0 & -a_W & a_P & -a_E & \phi_{15} & S_{u,15} \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & -a_N & 0 & 0 & -a_W & a_P & \phi_{16} & S_{u,16}
\end{array} \right] = \left[\begin{array}{c} \mathbf{u} \\ \mathbf{v} \end{array} \right]$$

Le système d'équations algébriques peut être résolu soit en utilisant des solveurs directs ou itératifs. En raison de la taille du système, les méthodes itératives sont généralement préférées car elles sont plus rapides, mais moins robustes si le système est mal conditionné.

Pour résoudre le système d'équations algébriques, nous utiliserons l'algorithme de SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) qui est adapté aux problèmes stationnaires de la mécanique des fluides.

2.2.3 Algorithme SIMPLE

L'algorithme SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) a été présenté par Patankar et Spalding en 1972. Il s'agit d'une procédure itérative pour le calcul faiblement couplé de la pression et de la vitesse, basée sur une approche de prédiction-correction.

Pour faciliter la compréhension, nous expliquons ici le fonctionnement de l'algorithme sur un maillage décalé (*staggered mesh*), où les variables ne sont pas toutes stockées au centre du volume de contrôle (VC), comme illustré à la Figure 4. Dans ce schéma, les vitesses sont définies sur les faces des VC.

Étape 1 : Discrétisation des équations de quantité de mouvement

Les équations discrétisées pour les deux composantes de la vitesse u et v s'écrivent :

$$\begin{cases} a_{i,J}u_{i,J}^* = \sum_{nb} a_{nb}u_{nb}^* - (p_{I-1,J}^* - p_{I,J}^*)A_{i,J} + b_{i,J} \\ a_{I,j}v_{I,j}^* = \sum_{nb} a_{nb}v_{nb}^* - (p_{I-1,J}^* - p_{I,J}^*)A_{I,j} + b_{I,j} \end{cases} \quad (2.5)$$

où u^* et v^* sont les vitesses calculées à partir d'une estimation initiale de la pression p^* , b est le terme source, nb désigne les voisins de la cellule, et A est la surface de la face, ces équations sont dans la forme canonique (2.4).

À cette étape, on résout les équations de quantité de mouvement en utilisant une pression initiale (devinée). Cela donne des champs de vitesse provisoires.

Étape 2 : Introduction des corrections

Pour améliorer la solution, on introduit des champs de correction pour la pression et la vitesse :

$$p = p^* + p', \quad u = u^* + u', \quad v = v^* + v'$$

où p' , u' , v' sont les corrections à appliquer.

On cherche à corriger les champs provisoires pour satisfaire la conservation de la masse (continuité).

Étape 3 : Calcul des corrections de vitesse On injecte $u = u^* + u'$ dans l'équation de continuité :

$$\nabla \cdot \mathbf{u} = \nabla \cdot (\mathbf{u}^* + \mathbf{u}') = 0 \implies \nabla \cdot \mathbf{u}' = -\nabla \cdot \mathbf{u}^* \quad (2.6)$$

Et comme on peut exprimer \mathbf{u}' en fonction de $\nabla p'$, on obtient une équation pour la pression corrigée p' .

$$\nabla \cdot \left(\frac{\nabla p'}{a_P} \right) = \nabla \cdot \mathbf{u}^* \quad (2.7)$$

Dans la forme canonique les corrections de vitesse sont obtenues à partir des équations de quantité de mouvement, en négligeant le terme de somme (simplification principale du SIMPLE) :

$$a_{i,J}u'_{i,J} = (p'_{I-1,J} - p'_{I,J})A_{i,J} + b_{i,J} \quad (2.8)$$

$$a_{I,j}v'_{I,j} = (p'_{I-1,J} - p'_{I,J})A_{I,j} + b_{I,j} \quad (2.9)$$

On relie la correction de vitesse directement à la correction de pression, ce qui permet de formuler une équation pour la pression corrigée [16].

L'ensemble de l'algorithme SIMPLE peut être visualisé dans l'organigramme de la Figure 2.3 .

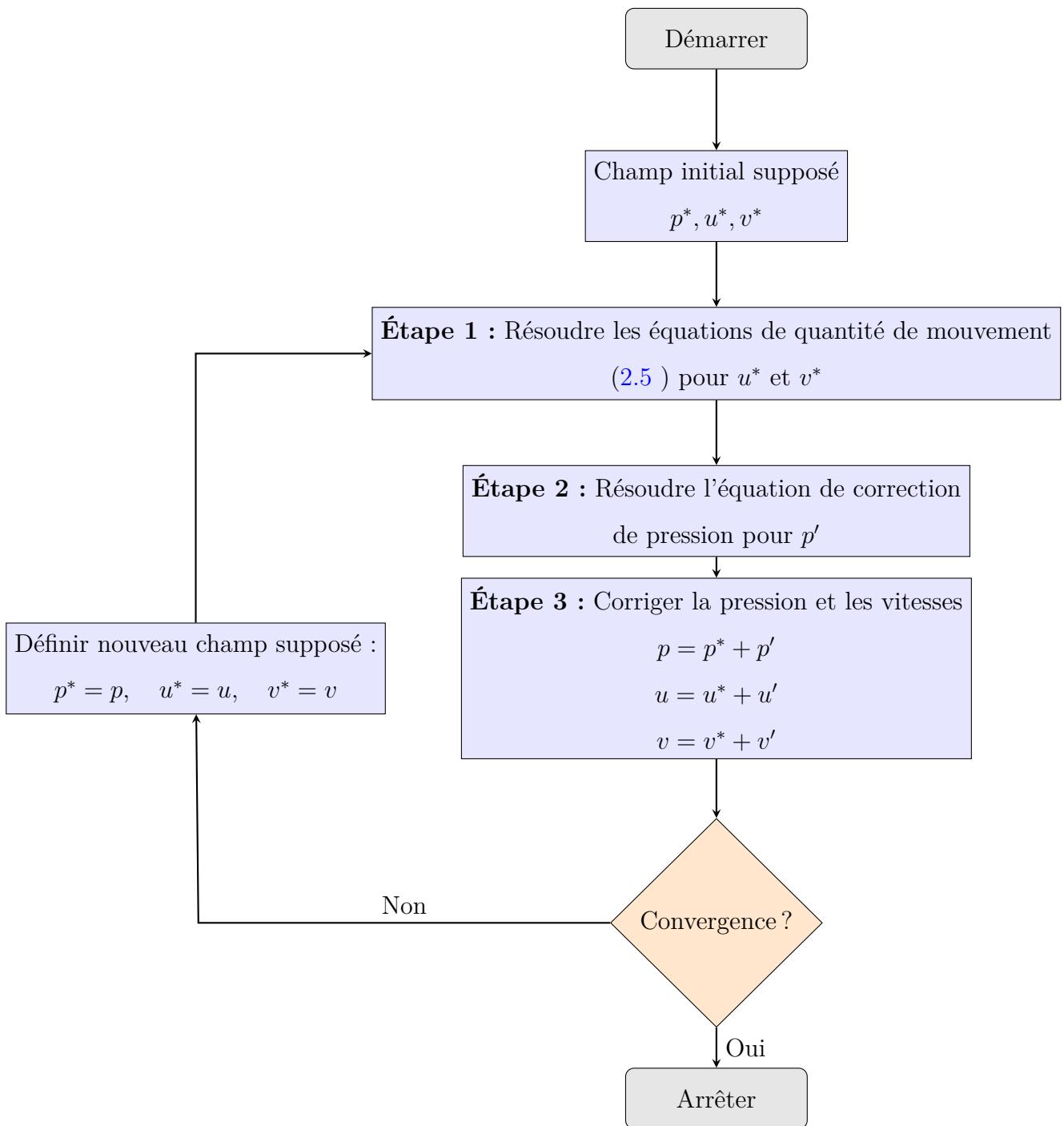


FIGURE 2.3 – Schéma de l'algorithme SIMPLE.

2.3 Implementation dans OpenFOAM

2.3.1 Extension de la loi de Carreau-Yasuda

Dans OpenFOAM, la loi de Carreau-Yasuda est implémentée dans le modèle de viscosité `BirdCarreau` qui est situé dans le répertoire `viscosityModels` de la bibliothèque OpenFOAM :

```
$FOAM_SRC_DIR/viscosityModels
```

Ce modèle est utilisé pour simuler des fluides non newtoniens. Pour étendre cette loi afin d'inclure la dépendance à la température, il existe dans OpenFOAM un modèle de viscosité qui prend en compte la dépendance thermique, appelé loi d'Arrhenius, situé également dans le répertoire `viscosityModels` de la bibliothèque OpenFOAM. À ce niveau, cette approche présente des limitations car elle consiste en une multiplication entre la loi d'Arrhenius et la loi de Carreau-Yasuda, ce qui n'est pas conforme à notre approche qui consiste à considérer la dépendance à la température directement intégrée dans la loi de Carreau-Yasuda comme nous l'avons mentionné dans la formule 1.2.

À ce stade, nous avons développé un nouveau modèle de viscosité appelé `tempdepBirdCarreau` qui constitue une extension de la loi de Carreau-Yasuda pour inclure la dépendance à la température selon la formulation 1.2. Pour cette implémentation, nous avons créé et modifié les fichiers suivants :

- `tempdepBirdCarreau.C` : implémente la loi de Carreau-Yasuda avec la dépendance à la température
- `tempdepBirdCarreau.H` : définit les coefficients et les paramètres de la loi thermodynamique
- `Make` : fichier de compilation du modèle de viscosité

Cette implémentation permet de simuler des fluides non newtoniens dont la viscosité dépend de la température et prend en compte le terme d'auto-échauffement dans l'équation de la chaleur, ce qui est essentiel pour notre étude des écoulements non newtoniens laminaires lors de l'extrusion de polymères. Pour voir l'ensemble de l'implémentation, vous pouvez consulter la section A.1 de l'Annexes.

2.3.2 Solveur simpleFoam

Le solveur `tempSIMPLEFoam` a été créé comme une extension du solveur intégré `SIMPLEFoam`. Ce solveur trouve ses applications dans les écoulements incompressibles, modifié ici pour une nouvelle application adaptée à l'étude des écoulements non newtonien liminaire depend de la température en utilisant loi de Carreau-Yasuda. Le solveur utilise l'algorithme SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) pour des simulations instationnaires résolues en temps. Après l'initialisation d'OpenFOAM 24, ce solveur se trouve dans :

```
$FOAM_APP_DIR/solvers/multiphase/simpleFoam
```

Le dossier contient les fichiers et sous-dossiers suivants :

- **Make/** : contient les fichiers `files` et `options` nécessaires à la compilation du solveur,
- **createFields.H** : initialise tous les champs et objets utilisés dans le solveur,
- **pEqn.H** : résout l'équation de pression,
- **UEqn.H** : résout l'équation de quantité de mouvement,
- **TEqn.H** : résout l'équation de température,
- **tempSimpleFoam.C** : le fichier principal du solveur qui contient toutes les étapes à exécuter.

Maintenant que nous disposons d'un modèle de viscosité qui dépend de la température, nous implémentons l'équation de température dans le solveur `tempSimpleFoam` en ajoutant le fichier `TEqn.H` dans le répertoire du solveur. Ce fichier contient l'équation de température, puisque le nombre de Reynolds est faible, alors le terme d'inertie est négligé de même puisque nous travaillons avec SimpleFoam qui est résolue à chaque itération jusqu'à la convergence selon la formulation présentée dans l'équation 1.28. L'implémentation inclut notamment le terme de dissipation visqueuse qui génère de l'auto-échauffement lors du cisaillement du polymère. Finalement, après avoir modifié l'ensemble des fichiers nécessaires pour intégrer la dépendance thermique dans le solveur `tempSimpleFoam`, nous sommes en mesure de lancer les simulations couplées fluid-thermiques. Cette approche permet de prendre en compte simultanément l'évolution de la viscosité avec la température et les effets de chauffage par dissipation visqueuse, essentiels pour une modélisation réaliste de l'extrusion de polymères. Pour plus de détails sur cette configuration, se référer aux travaux de ([18], [19]) et pour l'ensemble de l'implémentation, voir la section A.2 de l'Annexes.

2.4 Automatisation des simulations

Pour automatiser les simulations, nous avons développé un ensemble de scripts Python qui permettent de lancer les simulations de manière automatique et de collecter les résultats. Ces scripts sont conçus pour être exécutés dans un environnement OpenFOAM, ce qui facilite la gestion des simulations et l'analyse des résultats. Ils permettent de configurer les paramètres de simulation, de lancer les solveurs, et de collecter les données nécessaires pour l'analyse post-simulation. Cette automatisation est essentielle pour gérer efficacement les simulations complexes et répétitives, notamment dans le cadre d'études paramétriques ou d'optimisation et pour relier les simulations à la modélisation par ordre réduit (ROM) (que nous détaillons dans les chapitres suivants). Les scripts permettent également de configurer les constantes de la loi de Carreau-Yasuda et les conditions aux bords associées, à partir de fichiers JSON et de scripts bash.

Listing 2.1 – Structure des fichiers

```
openfoam-case-manager/
|-- src/openfoam_case_manager/          # Core Python modules
|   |-- core/                          # Case generation and mesh utilities
|       |-- step_to_geo_converter.py    # NEW: STEP-to-GEO converter
|       |-- mesh.py                   # Enhanced meshing with extrusion
|       '-- case.py                  # JSON-based case generation
|   '-- ROM/                         # Reduced Order Modeling
|-- scripts/                         # Automation scripts
|   |-- step_to_geo.sh                # NEW: Ergonomic STEP converter
|   |-- run_openfoam_case.sh         # Main simulation workflow
|   '-- compare_pod_vs_ae.py        # ROM performance analysis
|-- examples/                        # Ready-to-run examples
|   |-- cable_coating_2d/           # 2D axisymmetric cable coating
|   |-- cavity2d/                  # Driven cavity with ROM
|   '-- extrusion_case_steady/     # Multi-zone steady flow
|-- docs/                            # Comprehensive documentation
|   |-- step_to_geo_converter_guide.md # NEW: Geometry processing guide
|   '-- enhanced_meshing_workflow_guide.md # Meshing documentation
`-- openfoam-extensions/            # Custom solvers and libraries
```

Listing 2.2 – Exemple de configuration Json

```
"fluid": {  
    "type": "tempdepBirdCarreau",  
    "properties": {  
        "density": 936,  
        "A_final": 15219.5,  
        "A_initial": 152.195,  
        "A_ramp_iterations": 100,  
        "k": 0.11758,  
        "n": 0.15332,  
        "a": 0.53710,  
        "b": 0.03003,  
        "T0": 453.25,  
        "CN": 453.25,  
        "Cp": 1949,  
        "self_heating_correction_factor": 0.005,  
        "self_heating_max_strain_rate": 1000  
    }  
}
```

Chapitre 3

Méthodes de modèles réduits

3.1 Introduction

De nombreux domaines de sciences appliquées, de technologie, de l'ingénierie et de l'industrie s'appuient sur la simulation numérique pour leur capacité prédictive, ainsi que pour la conception et l'optimisation des systèmes (par exemple, la conservation de l'énergie, les champs électriques ou magnétiques, etc.). Cependant, l'augmentation constante et la diversité des ressources de calcul disponibles ne suffisent pas à répondre à la demande croissante de simulations numériques rapides et de haute fidélité, tout en réduisant les coûts de calcul. Le développement de méthodes numériques et d'algorithmes efficaces est donc essentiel, non seulement au sens de garantir la fiabilité des simulations, mais aussi pour exploiter au mieux la puissance de calcul disponible.

Dans ces domaines, on est fréquemment confronté à des Équations aux Dérivées Partielles (EDP) qui modélisent les phénomènes physiques étudiés. Les techniques de discrétisation conduisent alors à des modèles numériques de très grande dimension, impliquant des exigences importantes en termes de ressources matérielles, de mémoire et de temps de calcul. C'est le cas pour différents types de discrétisation, tels que les éléments finis (EF), les volumes finis (VF), ou les méthodes de différences finies (DF), entre autres. Ces défis majeurs rendent les coûts de calcul particulièrement élevés, notamment dans le contexte de scénarios à simulations multiples, en temps réel ou de calculs intégrés.

Les scénarios à simulations multiples correspondent à des situations où l'on ne cherche pas un unique résultat de simulation, mais où le problème varie et nécessite de multiples simulations,

selon le contexte étudié. Cela se rencontre par exemple lors d'études paramétriques (variation des paramètres géométriques ou physiques), de conception, d'optimisation, de problèmes inverses ou d'analyses statistiques.

Les scénarios en temps réel concernent des situations où le résultat de la simulation doit être obtenu très rapidement, tant en temps de calcul qu'en stockage. Cela peut concerter l'interaction avec des processus réels (contrôle, prédition, optimisation) ou avec des utilisateurs humains (par exemple, dans l'industrie pour réduire les temps de production, ou pour des ingénieurs de développement utilisant des logiciels de simulation et nécessitant des réponses rapides et efficaces).

Les scénarios de calcul intégré correspondent à des situations où l'on exige la même efficacité et fiabilité des simulations (par exemple pour des contrôleurs techniques simples, des applications mobiles, etc.), mais où les capacités de calcul sont très limitées en termes de vitesse ou de mémoire.

Pour cette classe de problèmes, la modélisation par réduction de modèle constitue une approche performante. Cette dernière désigne de manière générale toute méthode visant à remplacer un problème haute fidélité par un problème de complexité numérique bien moindre. Pouvoir évaluer la solution de ce modèle réduit, pour toute nouvelle instance de paramètre, à un coût indépendant de la dimension du problème haute fidélité d'origine (par exemple, issu d'une discrétisation EF ou VF), est la clé du succès de tout modèle réduit (ROM).

Les méthodes à base réduite constituent un exemple remarquable de techniques de modélisation par ordre réduit. Elles exploitent la dépendance paramétrique de la solution du problème en combinant un petit nombre de solutions haute fidélité (ou snapshots) calculées pour un ensemble (éventuellement restreint) de valeurs de paramètres. Par cette approche, un très grand système algébrique (matrices, vecteurs, etc.) est remplacé par un système beaucoup plus petit.

Les méthodes RB permettent ainsi de traiter un large éventail de problèmes, dans différentes disciplines, grâce à des temps de calcul très courts et à des besoins de stockage limités. Cette accélération considérable est due à la capacité des méthodes RB à approximer efficacement la variété des solutions paramétriques tout en réduisant le coût de calcul([21], [22],[23],[24]).

3.1.1 Exploration des méthodes de modèle réduit

Dans les applications de simulation numérique, on s'intéresse souvent à des variables qui décrivent le problème, appelées paramètres, qui sont regroupées dans un vecteur de paramètres $\mu \in \mathcal{P}$ (snapshots). Ici, on suppose que $\mathcal{P} \subset \mathbb{R}^p$ est un ensemble de paramètres de faible dimension p . La solution paramétrique sera alors notée $u(\mu)$ et provient généralement d'un espace de solutions X qui peut être de très grande dimension. Ainsi, la procédure de calcul standard consiste à partir d'un paramètre à calculer une solution $u(\mu)$ généralement dépendante de la discrétisation. Pour que la solution converge approximativement vers la solution exacte avec une précision élevée, la solution discrète doit être de grande dimension. Il est clair que la partie la plus coûteuse en calcul dans cette chaîne est la résolution de $u(\mu)$. L'objectif de la réduction de modèle est donc de développer des techniques permettant d'obtenir des approximations de faible dimension, et donc rapidement calculables, pour la solution $u(\mu)$ et les sorties éventuelles.

L'idée clé permettant de simplifier les problèmes paramétriques est le fait que la variété des solutions \mathcal{M} , c'est-à-dire l'ensemble des solutions paramétriques, peut souvent être bien approchée par un sous-espace de faible dimension $X_N \subset X$. Dans les méthodes RB, une façon courante de construire ce sous-espace est d'utiliser les snapshots, c'est-à-dire que X_N est engendré par les solutions $u(\mu^{(i)})$ pour des paramètres appropriés $\mu^{(i)}$, $i = 1, \dots, N$. Un choix judicieux de ces paramètres est crucial pour construire un espace d'approximation de qualité. Après la construction de l'espace, le modèle réduit est obtenu, par exemple, par projection de Galerkin. Il fournit alors une approximation $u_N(\mu) \in X_N$ de la solution, ainsi qu'une approximation $s_N(\mu)$ de la quantité de sortie d'intérêt, selon le problème étudié. Voir la Figure 3.1 pour une illustration du scénario d'approximation RB. En plus de la réduction de la complexité et du gain de temps de calcul, il est également possible d'estimer l'erreur d'approximation.

Le processus de calcul est principalement décomposé en deux phases : une phase hors-ligne (offline) et une phase en ligne (online). Lors de la phase hors-ligne, effectuée une seule fois, une base réduite est générée à partir de la matrice des snapshots (espace X_N) qui est construite pour différentes valeurs de paramètres μ . Ensuite, lors de la phase en ligne, pour chaque nouveau vecteur des paramètres μ , la solution approchée peut être obtenue rapidement. La complexité de calcul de la phase hors-ligne dépend généralement de façon

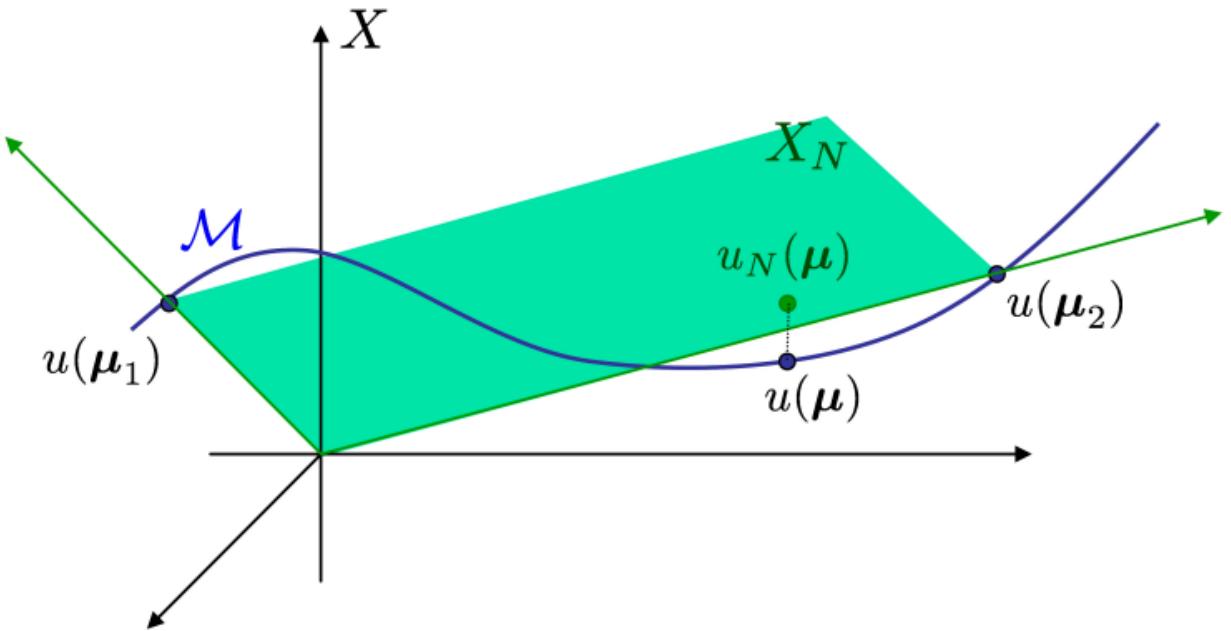
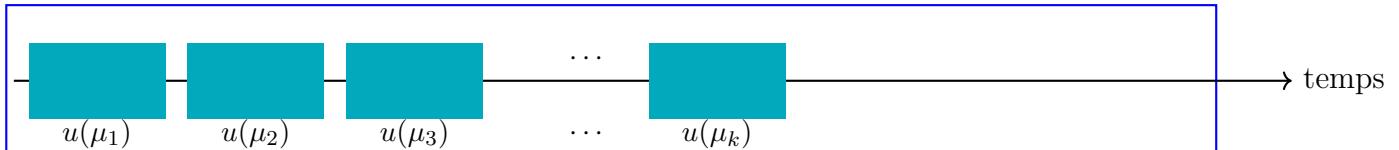


FIGURE 3.1 – Illustration de la variété des solutions et de l’approximation RB

polynomiale de la dimension de l'espace complet X , qui peut donc être supposé arbitrairement précis et aussi le nombre des paramètres p car si cette dernière est grande alors on doit calculer un grand nombre de snapshots pour couvrir la variété des solutions.

En revanche, la complexité de la phase en ligne dépend typiquement de façon polynomiale de N , la dimension de l'espace réduit. L'avantage en temps de calcul d'un modèle à base réduite (RB) dans le contexte d'un scénario multi-simulations est illustré à la Figure 3.2 : la phase hors-ligne est généralement bien plus coûteuse que plusieurs simulations complètes.

Multi-simulations avec modèle haute fidélité :



Multi-simulations avec modèle réduit :



FIGURE 3.2 – Gain de temps du modèle RB dans les scénarios multi-requêtes.

3.1.2 Paramétrisation du problème

Nous considérons les équations aux dérivées partielles ((1.12), (1.19), (1.28)) qui modélisent notre problème, en tenant compte des conditions spécifiques utilisées chez ACOME. Avec la loi de comportement Carreau-Yasuda ((1.2)), la viscosité dépend de plusieurs paramètres physiques. Or, chez ACOME, différents matériaux sont utilisés comme ceux présentés dans le tableau 1.1, ce qui pose un réel défi scientifique pour choisir les bons paramètres pour aider dans l'anticipation de l'extrudabilité qui est la problématique qui motive cette étude. En effet, le matériau doit être à la fois compatible avec le cahier des charges du produit final destiné au client (résistance au feu, endurance, etc.) et présenter une extrudabilité optimale pour maximiser la rentabilité industrielle et minimiser les problèmes de qualité. Cela conduit à des différences de comportement entre les matériaux, notamment en fonction de la viscosité. Puisque nos équations dépendent de la viscosité, il est essentiel de pouvoir prédire des variables telles que la vitesse, la pression et la température en fonction des paramètres du matériau. À ce stade, nous allons donc paramétriser notre problème en nous basant sur les paramètres physiques du matériau et aussi la vitesse de cable. Nous considérons un ensemble de paramètres $\mu = (v, T, \mu_0, \mu_\infty, \lambda, n, a, b)$, qui caractérisent le matériau. Ces paramètres sont regroupés dans un vecteur $\mu \in \mathcal{P}$, où $\mathcal{P} \subset \mathbb{R}^p$ est un ensemble de dimension p relativement faible (en pratique, on peut choisir un ou plusieurs paramètres selon les besoins de l'étude). La solution paramétrique sera alors notée $U(\mu)$ et appartient généralement à un espace de solutions X . Notre problème s'écrit alors :

$$F(U(\mu), \mu) = 0, \quad (3.1)$$

où F est un opérateur non linéaire qui dépend de la solution $U(\mu)$ et des paramètres μ , avec $U(\mu) = (u(\mu), p(\mu), T(\mu))$ représentant respectivement la vitesse, la pression et la température. L'objectif est de résoudre l'équation (3.1) pour différentes valeurs du paramètre μ , afin d'obtenir des solutions approchées $U_N(\mu)$ dans un espace réduit $X_N \subset X$ [29].

3.2 La méthode de réduction de modèle POD

Dans le domaine de modélisation par ordre réduit, de nombreuses méthodes existent et la méthode la plus connue sous le nom de **POD** (Proper Orthogonal Decomposition), et la plus

couramment utilisée dans divers domaines tels que l'analyse de données, le traitement d'images, l'aeroacoustics, et la mécanique des fluides. Pour savoir et prendre une idée sur son importance, il suffit regarder le nombre de publications scientifiques qui référence par l'intitulé (Proper Orthogonal Decomposition) sur le site de référencement scopus (<https://www.scopus.com/>).

Dans les années 1960, Lumley [25] a introduit la méthode POD, que l'on qualifie aujourd'hui de classique, dans le domaine de la mécanique des fluides, dans le but d'identifier les structures cohérentes d'un écoulement turbulent. Cette méthode consiste à déterminer les vecteurs propres d'un opérateur de corrélation spatiale du champ de vitesse. Cependant, la méthode POD classique est limitée aux problèmes de faible dimension, ce qui a conduit au développement d'une approche permettant de réduire la taille du problème à résoudre. Ainsi, la POD a commencé à se développer en mécanique des fluides grâce à la méthode des snapshots [2]. La POD est connue sous différents noms, tels que la méthode de Karhunen-Loève (KL), la méthode de SVD (Singular Value Decomposition) ou la méthode de PCA (Principal Component Analysis) en fonction de l'approche utilisée pour la construction de la base réduite.

3.2.1 La méthode POD classique

Considérons une fonction paramétrique $u(x, \mu)$, qui est la solution d'un problème paramétrique, et qui est définie sur un domaine $\Omega \subset \mathbb{R}^d$ avec $d = 1, 2, 3$.

$$\begin{aligned} u : \Omega \times R &\rightarrow \mathbb{R}^n \\ (x, \mu) &\mapsto u(x, \mu) \end{aligned} \tag{3.2}$$

Le principe de la POD classique est de rechercher les champs ϕ qui maximisent leur projection sur l'ensemble des champs connus u . Nous notons $\lambda(\phi)$ le quotient de cette projection :

$$\lambda(\phi) = \max_{\phi} \frac{\langle (\phi, u)^2 \rangle}{\langle (\phi, \phi) \rangle}$$

Où $\langle (a, b)^2 \rangle = \frac{1}{N} \sum_i^N (a, b)^2$ est l'opérateur moyen sur le domaine Ω et (a, b) est le produit scalaire de $L^2(\Omega)$. Généralement, la recherche des champs ϕ qui maximisent ce produit scalaire est coûteuse en termes de calcul, ce qui explique pourquoi cette approche directe est rarement utilisée en pratique. C'est pourquoi, en général, on privilégie la méthode POD basée sur les snapshots, qui permet de réduire significativement le coût de calcul [26].

3.2.2 La méthode SVD (Singular Value Decomposition)

La méthode SVD (Singular Value Decomposition) est une méthode très générale de factorisation, qui offre une nouvelle interprétation géométrique de toute application linéaire $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$. Elle consiste à factoriser une matrice quelconque $A \in \mathbb{R}^{n \times N}$ en un produit de matrices. La SVD a été introduite dans les années 1870 par Beltrami et Jordan. Elle est particulièrement utile dans le contexte de la POD, car elle permet de trouver les vecteurs propres et les valeurs propres d'une matrice de covariance associée aux snapshots. La SVD est souvent utilisée pour réduire la dimensionnalité des données tout en préservant l'information la plus significative. Considérons un ensemble de snapshots $u_i \in \mathbb{R}^n$ pour $i = 1, \dots, N$, où chaque snapshot est un vecteur représentant une solution paramétrique à un instant donné. On peut organiser ces snapshots en une matrice $U \in \mathbb{R}^{n \times N}$, où chaque colonne correspond à un snapshot.

$$A = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_N \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times N} \quad (3.3)$$

Pour appliquer la SVD sur matrice A , de dimension $n \times N$, consiste à écrire sa factorisation sous la forme :

$$A = \sum_{i=1}^N u_i \sigma_i v_i^T = U \Sigma V^T \quad (3.4)$$

où $U \in \mathbb{R}^{n \times n}$ est une matrice orthogonale dont les colonnes sont les vecteurs propres de AA^T , $\Sigma \in \mathbb{R}^{n \times N}$ est une matrice diagonale dont les éléments diagonaux sont les valeurs singulières de A telles que $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_p)$ et $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p \geq 0$ pour $p = \min(n, N)$, et $V \in \mathbb{R}^{N \times N}$ est une matrice orthogonale dont les colonnes sont les vecteurs propres de $A^T A$. Les valeurs singulières σ_i sont les racines carrées des valeurs propres de la matrice $A^T A$ et sont ordonnées de manière décroissante, c'est-à-dire $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_N \geq 0$ [27].

3.2.3 La méthode POD pour un problème paramétrique

La méthode POD classique comme nous l'avons vu précédemment, est coûteuse en termes de calcul, c'est pourquoi Sirovich [2] a proposé une solution qui représente à résoudre une version équivalente de la formulation originale, appelée méthode des snapshots (ou méthode des

échantillons). Elle consiste à ne pas résoudre le système sur la grille de points, mais sur l'ensemble des échantillons utilisés pour construire le tenseur d'autocorélation. Elle s'appuie sur le fait que dans la pratique le tenseur d'autocorrélation est calculé comme une moyenne finie de N échantillons. Le problème est alors dégénéré et les solutions peuvent s'écrire comme combinaison linéaire des champs u_N constituant l'ensemble des échantillons. Dans le contexte d'un problème paramétrique, on part d'un ensemble de paramètres $\Xi = \{\mu^{(1)}, \mu^{(2)}, \dots, \mu^{(N)}\}$ et on considère les solutions snapshots correspondantes $u^{(i)} = u(\mu^{(i)}) \in \mathbb{R}^n$ pour $i = 1, \dots, N$ du problème paramétrique (3.1). On définit la matrice des snapshots A comme suit :

$$A = \begin{bmatrix} u^{(1)} & u^{(2)} & \dots & u^{(N)} \end{bmatrix} \in \mathbb{R}^{n \times N} \quad (3.5)$$

D'après la section précédente, on peut appliquer la SVD sur la matrice des snapshots A pour obtenir sa factorisation

$$A = U\Sigma V^T \quad (3.6)$$

où $U = [\phi_1, \phi_2, \dots, \phi_n]$ et $V = [\psi_1, \psi_2, \dots, \psi_N]$ sont des matrices orthogonales, $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_p)$. Notons que $p = \min(n, N)$ est le rang de la matrice A , qui est strictement inférieur à N si les vecteurs des snapshots ne sont pas linéairement indépendants. Alors, on peut écrire

$$A\psi_i = \sigma_i\phi_i, \quad \text{et} \quad A^T\phi_i = \sigma_i\psi_i, \quad i = 1, \dots, p \quad (3.7)$$

ou de manière équivalente,

$$A^T A\psi_i = \sigma_i^2\psi_i, \quad \text{et} \quad A A^T\phi_i = \sigma_i^2\phi_i, \quad i = 1, \dots, p \quad (3.8)$$

où $\sigma_i^2, i = 1, \dots, p$ sont les valeurs propres non nulles de la matrice $A^T A$ (et aussi de $A A^T$). Pour tout $m \leq p$, la base de POD, $Z \in \mathbb{R}^{n \times m}$ de dimension m est construite en prenant les m premiers vecteurs singuliers droits ϕ_1, \dots, ϕ_m de U , ou de manière équivalente

$$\phi_i = \frac{1}{\sigma_i} A\psi_i, \quad i = 1, \dots, m \quad (3.9)$$

obtenus par les premiers m vecteurs propres ψ_1, \dots, ψ_m de la matrice de corrélation $A^T A$. Par construction, la base POD est orthonormée. Elle minimise la somme de l'erreur quadratique entre les solutions paramétriques $u(\mu)$ et les projections de ces solutions sur la base $W = \{w_1, w_2, \dots, w_m\} \in \mathbb{R}^{n \times m}$. Précisément, la projection est donnée par

$$\Pi_W x = \sum_{i=1}^m (x, w_i)_2 w_i = W W^T x$$

Théorème 3.2.1 (Propriété de la base POD). Soit $\mathcal{V}_m = \{W \in \mathbb{R}^{n \times m} : W^T W = I_m\}$ l'ensemble de toutes les bases orthonormées de dimension m . Alors,

$$\sum_{i=1}^N \|u_i - VV^T u_i\|_2^2 = \min_{W \in \mathcal{V}_m} \sum_{i=1}^N \|u_i - WW^T u_i\|_2^2 = \sum_{i=m+1}^p \sigma_i^2 \quad (3.10)$$

Démonstration. Voir [21] page 124 pour la démonstration. □

3.3 Autoencodeur et bibliothèque EyzRB

Dans le monde actuel, l'intelligence artificielle occupe une place centrale dans de nombreux domaines, notamment la modélisation des phénomènes physiques, l'analyse de données, le traitement d'images, la reconnaissance vocale, et bien d'autres. Dans le contexte de la modélisation, différentes méthodes sont utilisées pour résoudre ces systèmes, telles que les PINNs (Physics Informed Neural Networks) ou les auto-encodeurs. Cette dernière est particulièrement utilisée pour la réduction de modèle, la réduction de dimensionnalité, la compression et la génération de nouvelles données.

Un auto-encodeur est un type de réseau de neurones qui apprend à reproduire ses entrées en les compressant dans un espace latent de dimension inférieure. Il se compose de deux parties principales : l'encodeur, qui réduit la dimensionnalité des données d'entrée, et le décodeur, qui reconstruit les données à partir de cette représentation compressée. Cette méthode a été introduite par Rumelhart, Hinton et Williams (1986) [31], qui ont proposé l'idée d'un réseau de neurones avec un codage interne comprimé (bottleneck), capable de reconstruire l'entrée. Plus récemment, Pichi et Moya [32] ont utilisé cette méthode pour des écoulements de fluide non linéaire, tandis que Fu et Xiao [33] l'ont appliquée avec des mécanismes d'attention pour la réduction de dimensionnalité dans les écoulements de fluide. Ces approches ont montré des performances comparables à la méthode POD, tout en étant capables de capturer des structures complexes dans les données.

3.3.1 Principe de l'autoencodeur

L'auto-encodeur (AE) de base est un réseau de neurones non supervisé, de type feed-forward, qui peut être utilisé pour réduire la dimensionnalité des données.

Définition 3.3.1 (Auto-encodeur). *Un auto-encodeur est un type d’algorithme dont le but principal est d’apprendre une représentation « informative » des données, utilisable pour différentes applications, en apprenant à reconstruire correctement un ensemble d’observations en entrée.*¹

Au lieu d’associer des étiquettes de sortie pour l’apprentissage, le réseau auto-encodeur (AE) cherche à reproduire ses entrées en sortie, avec le même nombre de neurones, la même forme et les mêmes valeurs. Il commence par projeter les données d’entrée dans un espace latent de dimension réduite (représenté par un certain nombre de codes dans la couche centrale du réseau), puis il reconstruit cette représentation latente vers la sortie, comme illustré à la Figure 3.3.

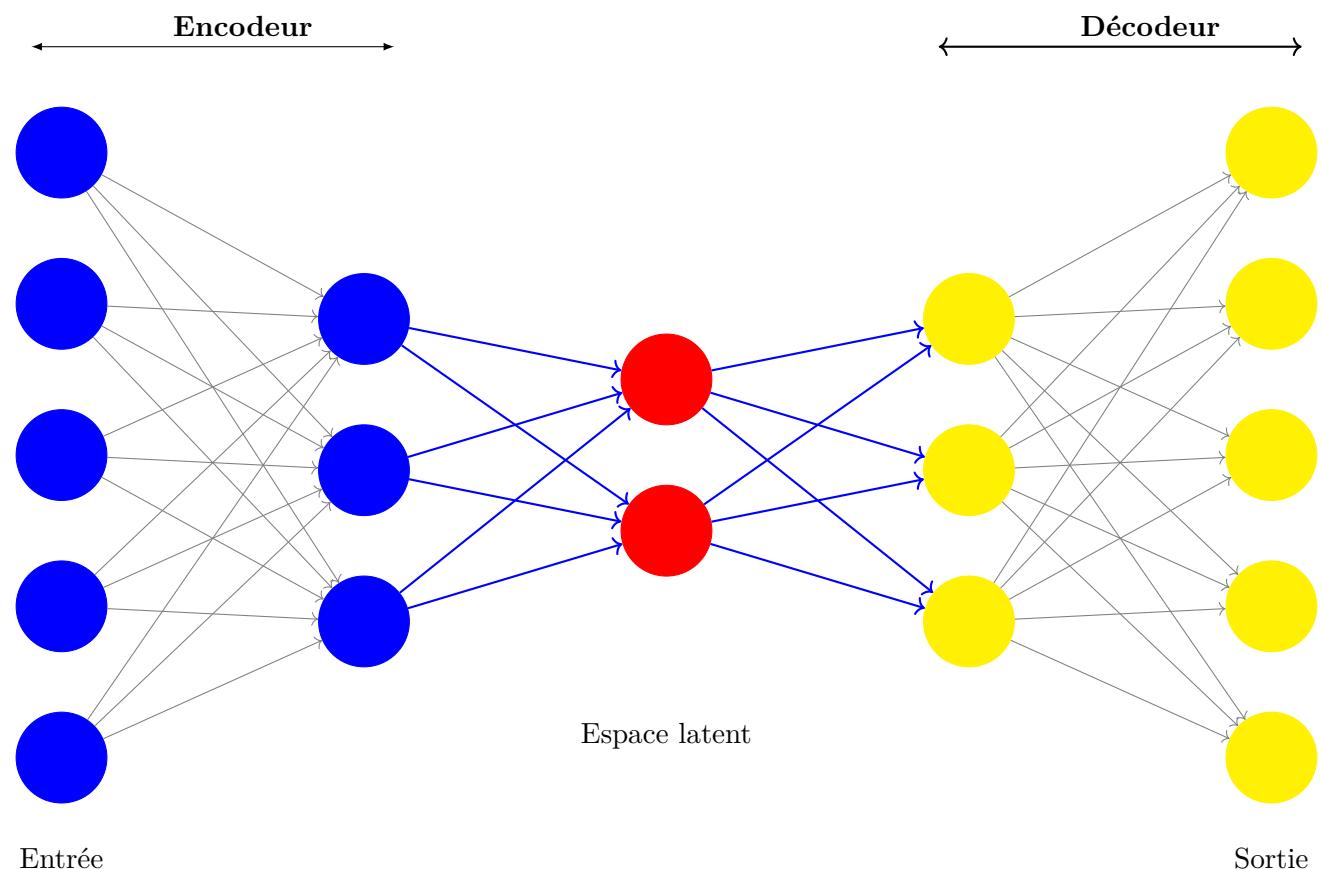


FIGURE 3.3 – Schéma d’un autoencodeur : encodeur, espace latent, décodeur, avec même nombre de neurones en entrée et en sortie.

L’objectif de cette architecture est donc de reconstruire les données d’entrée sous une forme

1. Adapté de : Bank, D., Koenigstein, N., et Giryes, R., *Autoencoders*, <https://arxiv.org/abs/2003.05991>

de dimension réduite, ce qui permet, par exemple, de compresser l'information ou d'extraire des caractéristiques essentielles. Les données d'entrée peuvent être, par exemple, la vitesse u , la pression p et la température T dans le cas d'un écoulement autour d'un outillage.

Un auto-encodeur se compose de deux parties principales : l'encodeur et le décodeur.

Mathématiquement, on considère des données $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^d$ que l'on cherche à approximer sous la forme $x = \mathcal{D}(y)$, avec $y \in \mathbb{R}^m$ où $m \ll d$ et \mathcal{D} une fonction non linéaire.

L'enjeu est donc de trouver une représentation latente z (issue d'une transformation de x) et de construire à la fois cette transformation et une approximation de l'opérateur \mathcal{D} .

Définition 3.3.2. Soit $X \subset \mathbb{R}^d$ et $Z \subset \mathbb{R}^m$ avec $m < d$. On nomme encodeur une fonction paramétrique $\mathcal{E}_{\theta_e} : X \rightarrow Z$ et on nomme décodeur une fonction paramétrique $\mathcal{D}_{\theta_d} : Z \rightarrow X$.

Définition 3.3.3 (Auto-encodeur). Soit (x_1, x_2, \dots, x_n) de \mathbb{R}^d . On nomme auto-encodeur un couple encodeur-décodeur solution du problème de minimisation :

$$\sum_{i=1}^n \|x_i - \mathcal{D}_{\theta_d}(\mathcal{E}_{\theta_e}(x_i))\|_2^2 + R(\theta_e, \theta_d) \quad (3.11)$$

avec \mathbf{R} un terme de régularisation.

L'idée générale est d'apprendre simultanément un opérateur de compression (encodeur) et un opérateur de reconstruction (décodeur) de sorte que leur composition soit la plus proche possible de l'identité, sous contrainte de réduction de dimension. Si la dimension de l'espace latent n'est pas inférieure à celle de l'entrée, l'auto-encodeur peut apprendre l'identité, mais dans le cas contraire, une perte d'information est inévitable. Dans le cas linéaire, la solution optimale correspond à la PCA, mais l'utilisation de réseaux de neurones permet d'approcher des transformations non linéaires plus complexes. Lorsque l'architecture comporte plusieurs couches cachées, on parle d'auto-encodeur empilé (Stacked Auto-Encoder, SAE)[34].

3.3.2 Bibliothèque EZyRB

EZyRB est une bibliothèque Python dédiée à la réduction de modèle (Model Order Reduction), basée sur la triangulation barycentrique pour la sélection des points de paramètres [35] et sur la Décomposition Orthogonale Propre (POD) pour la sélection des modes, ainsi que sur la méthode d'auto-encodeur (voir le diagramme de la bibliothèque 3.4). Elle est particulièrement adaptée aux problématiques industrielles réelles, car sa structure

permet d’interagir avec plusieurs logiciels de simulation en fournissant simplement les fichiers de sortie des simulations. Dans notre cas, puisque la bibliothèque supporte les fichiers au format VTK, nous avons effectué une conversion des fichiers de sortie du solveur haute fidélité (OpenFOAM) vers le format VTK pour la construction de la matrice de données.

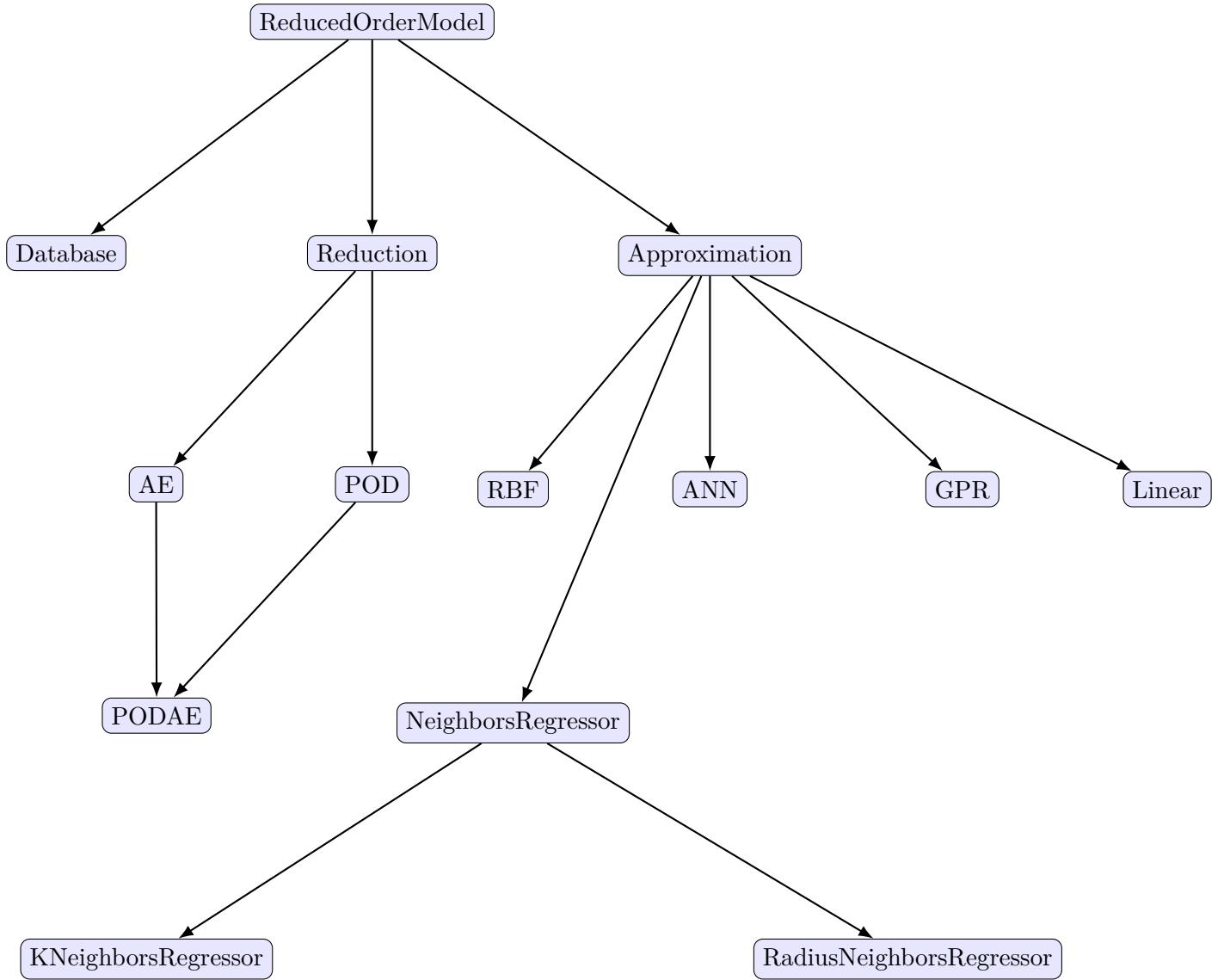


FIGURE 3.4 – Le diagramme de la structure complète de la bibliothèque EZyRB

Le logiciel utilise une approche d’interpolation POD, dans laquelle les solutions sont projetées dans l’espace de faible dimension engendré par les modes POD. La nouvelle solution est ensuite obtenue par interpolation des solutions de rang réduit dans l’espace des paramètres. Cette approche représente un avantage significatif dans la pratique par rapport aux méthodes intrusives tel que la méthode POD-Galerkin, car elle ne nécessite pas de modifier le solveur

haute fidélité. Cela permet une intégration facile dans les chaînes de simulation existantes, et la bibliothèque peut traiter aussi bien des fichiers au format vtk ([36], [37],[38]).

Chapitre 4

Construction de modèles réduits et résultats

Dans ce chapitre, nous abordons la construction de modèles réduits pour différentes géométries utilisées en simulation, ainsi que les résultats obtenus soit avec OpenFoam, soit par prédiction à l'aide des méthodes POD et AE toute en présentant les performances des modèles qui sont vérifiées en calculant les erreurs et le temps de calcul.

4.1 Simulation et résultats de la cavité 2D

Nous avons commencé par un cas test simple de géométrie (carré 2D) pour simuler notre problème avec la dépendance thermique. Ensuite, nous avons défini la géométrie associée à l'outillage de l'extrudeuse sous Gmsh, et nous avons abordé les conditions aux bords correspondantes.

4.1.1 Géométrie et conditions de bord de la Cavity 2D

Généralement les simulations sous OpenFoam sont réalisées en 3D, même lorsque l'on souhaite travailler en 2D, car il est nécessaire de construire le maillage en 3D. Cette dernière peut être créée directement dans OpenFOAM à l'aide de blockMesh, ou bien générée sous Gmsh puis convertie du format msh vers le format OpenFOAM. Pour la version OpenFOAM, voir [20]. Pour la version Gmsh, la géométrie est construite avec le code présenté dans la section A.3 de l'annexe. Concernant les conditions aux limites sous OpenFOAM pour un maillage 2D extrudé, on utilise la condition "empty" pour les faces avant et arrière du carré. Pour les autres faces, les conditions sont définies selon notre problème : ici, le fluide est

entrainé par la vitesse tangentielle de la paroi de haut. L'ensemble des conditions aux limites est résumé dans le tableau 4.1.

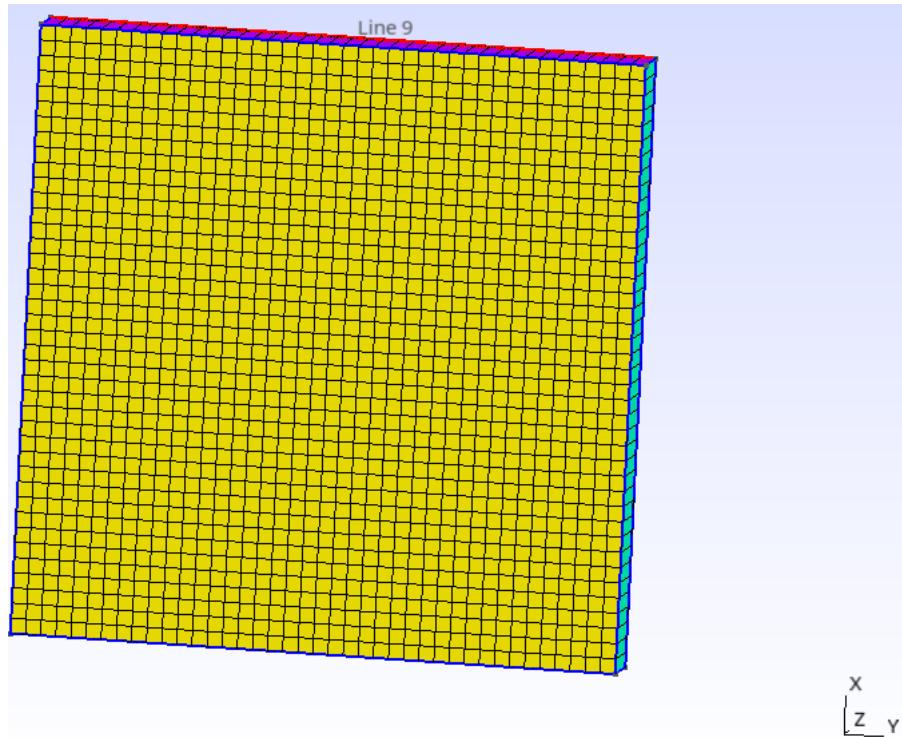


FIGURE 4.1 – Maillages utilisés pour tester la convergence de la méthode

TABLE 4.1 – Conditions aux bords associées aux différentes surfaces du domaine

Surface	P (Pression)	U (Vitesse)	T (Température)
front	empty	empty	empty
back	empty	empty	empty
left	zeroGradient	Dirichlet ($\mathbf{U} = 0$)	zeroGradient
right	zeroGradient	Dirichlet ($\mathbf{U} = 0$)	zeroGradient
top	zeroGradient	Dirichlet ($\mathbf{U} = (u \ 0 \ 0)$)	Dirichlet ($T = T_1$)
bottom	zeroGradient	Dirichlet ($\mathbf{U} = 0$)	Dirichlet ($T = T_2$)

4.1.2 Simulation de la cavité 2D

Pour la simulation de la cavité 2D, nous avons utilisé notre solveur développé dans OpenFOAM pour bien le tester. Nous présentons les résultats de la simulation de la cavité 2D

avec les conditions aux bords définies dans le tableau 4.1 pour les trois variables d'état avec $T_1 = 473.15K$, $T_2 = 433.15K$ et pour la vitesse $U = (1, 0, 0)$.

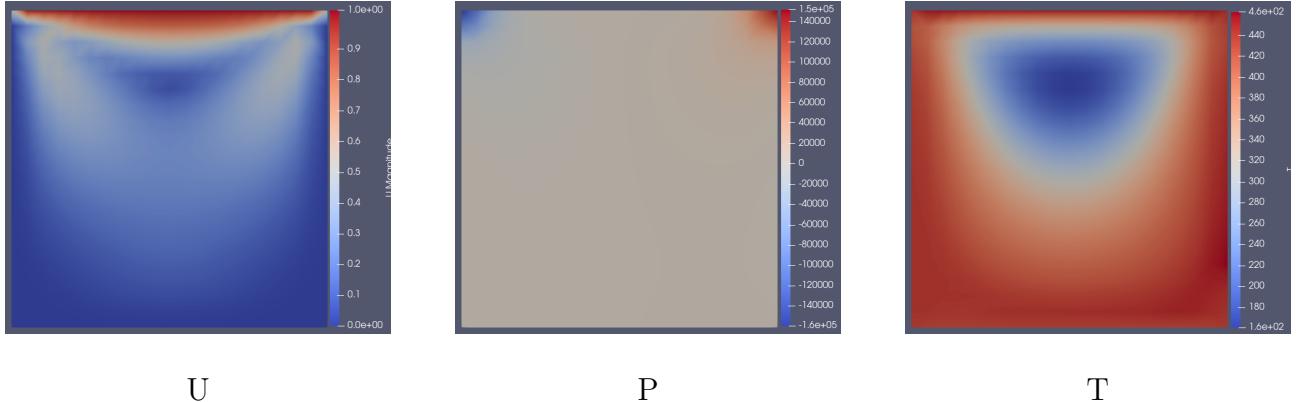


FIGURE 4.2 – Simulation de la cavité 2D pour les trois variables U en m/s , P en Pa et T en K.

4.1.3 Résultats de réduction de modèle POD et AE

Dans cette section, nous avons généré 50 snapshots de manière aléatoire en utilisant le Latin Hypercube Sampling (LHS) pour les paramètres de la solution de la cavité, afin de construire la base réduite avec les méthodes POD et AE avec les plages définies dans le tableau 4.2. Pour ce test, nous avons choisi $\mu = (U, T, A, k)$ comme vecteur de paramètres d'entrée, où U représente la vitesse imposée en haut, T est la température en haut, A et k sont des paramètres de la loi de Carreau Yassoda 1.2, avec $A = \mu_0$ et $k = \lambda$. Nous avons utilisé la méthode (SVD) pour la construction de la base POD avec 15 modes. Pour la méthode AE, nous avons utilisé un encodeur composé de trois couches (100, 50, 15) et un découpeur symétrique (15, 50, 100), avec également 15 pour la dimension de latent. Pour l'interpolation des solutions réduites, nous avons choisi la méthode de fonction à base radiale (RBF). Nous présentons les résultats de la prédiction de la cavité 2D avec les méthodes de réduction de modèle POD, AE, en les comparant à la solution référence pour les trois variables d'état, et en fournissant un tableau comparatif des erreurs pour les deux méthodes sur la vitesse, la pression et la température ces erreurs sont calculées entre la solution de référence et la solution prédite par ces formules pour L^2

$$\|u_r - u_p\|_2^2 \quad (4.1)$$

l'erreur relative

$$\frac{\|u_r - u_p\|_2^2}{\|u_r\|_2^2} \quad (4.2)$$

et l'erreur RMSE (Root Mean Square Error)

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (u_{r,i} - u_{p,i})^2} \quad (4.3)$$

où u_r est la solution de référence, u_p est la solution prédictée par le modèle réduit.

TABLE 4.2 – La plage des paramètres utilisés pour la construction de la base réduite.

Paramètre	Minimum	Maximum	Unité
U	0.9	1.1	m/s
T	443.15	463.15	K
A	14000	16000	Pa·s
k	0.10	0.14	s

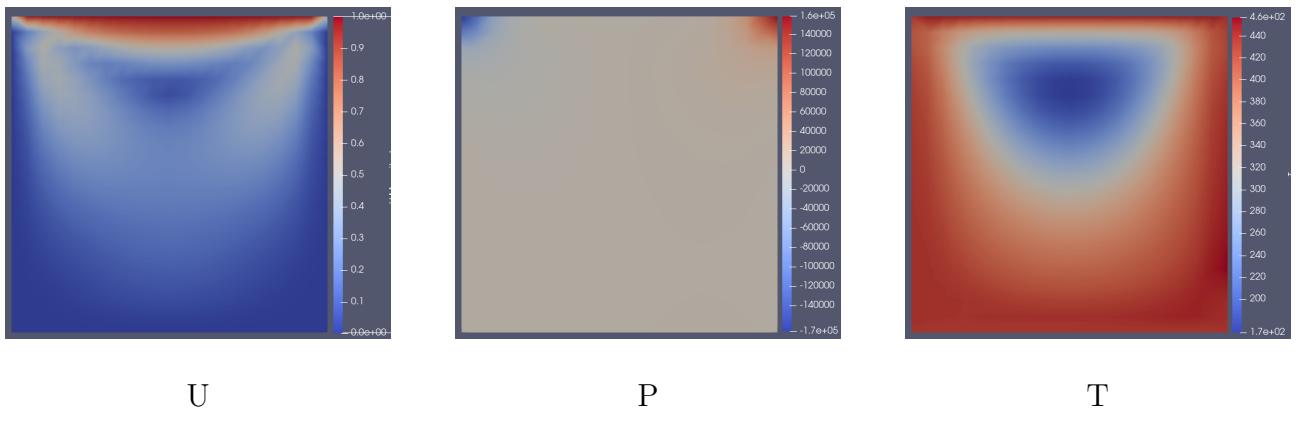


FIGURE 4.3 – Solution de Haut fidélité

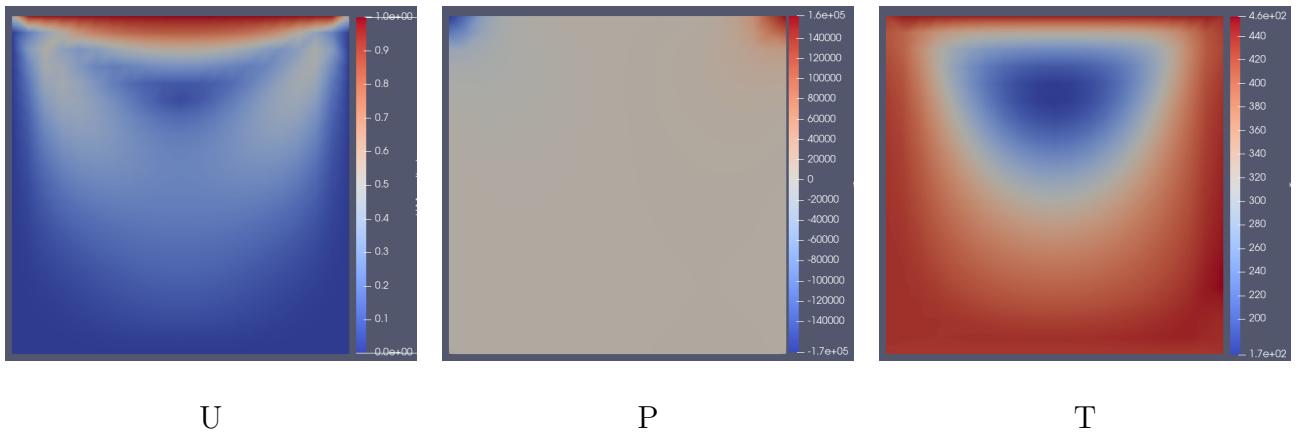


FIGURE 4.4 – Solution de POD

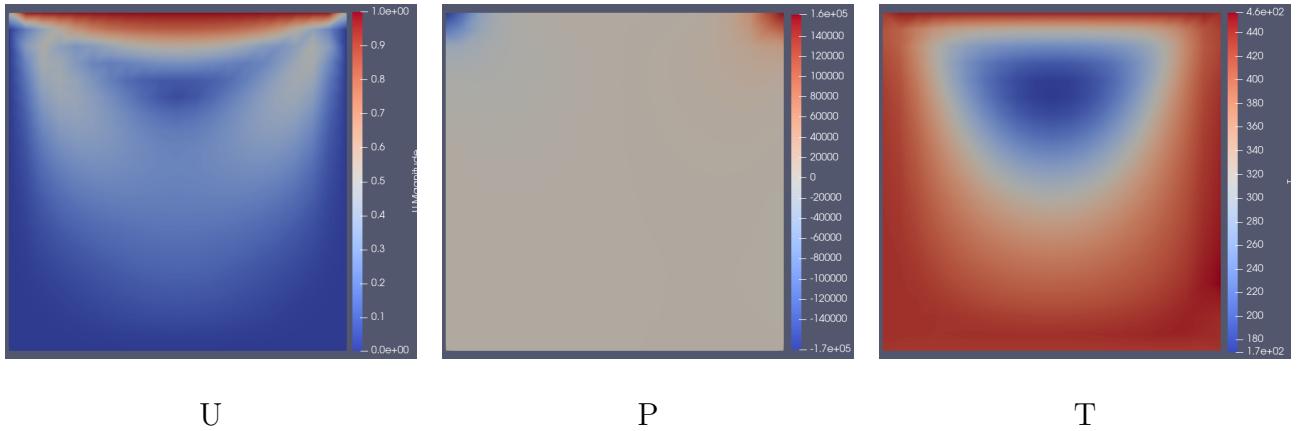


FIGURE 4.5 – Solution de AE

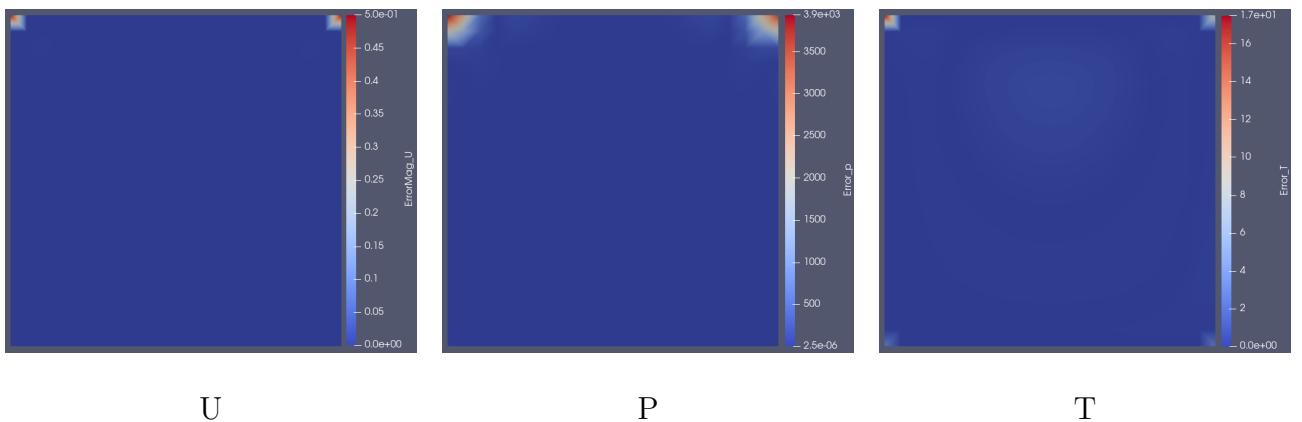


FIGURE 4.6 – Erreur entre la solution de référence et la solution prédite par les méthodes POD

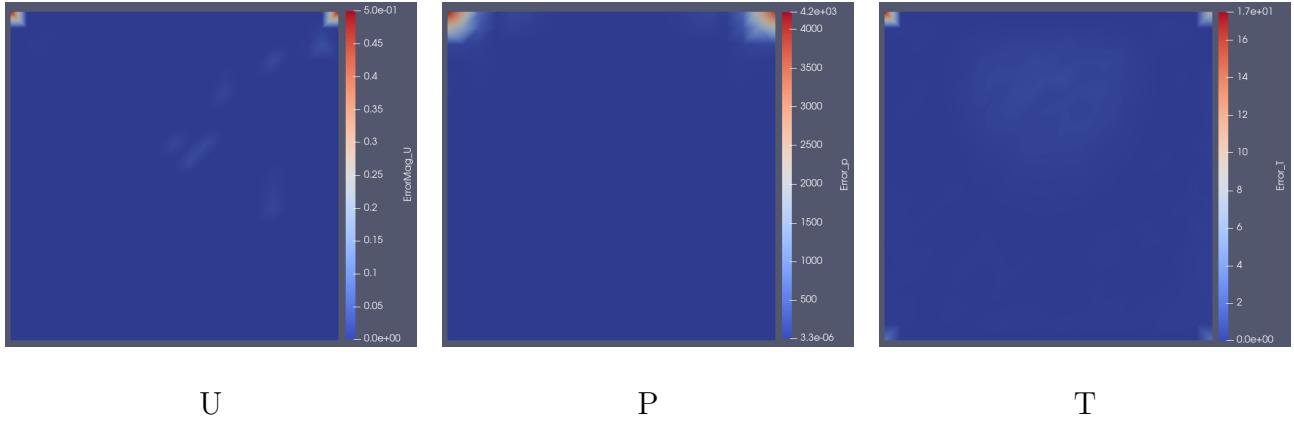


FIGURE 4.7 – Erreur entre la solution de référence et la solution prédictive par les méthodes AE

TABLE 4.3 – Comparaison des erreurs L2 entre POD et AE ($U = 1.0$, $T = 450.0$, $A = 15000$, $k = 0.12$), l'erreur est calculée entre la solution obtenue sous OpenFoam et la prédiction du modèle réduit avec EZyRB.

Variable	Erreur relative L2 POD	Erreur relative L2 AE
U (vitesse)	0.1060	0.1060
P (pression)	0.0198	0.0210
T (température)	0.0026	0.0393

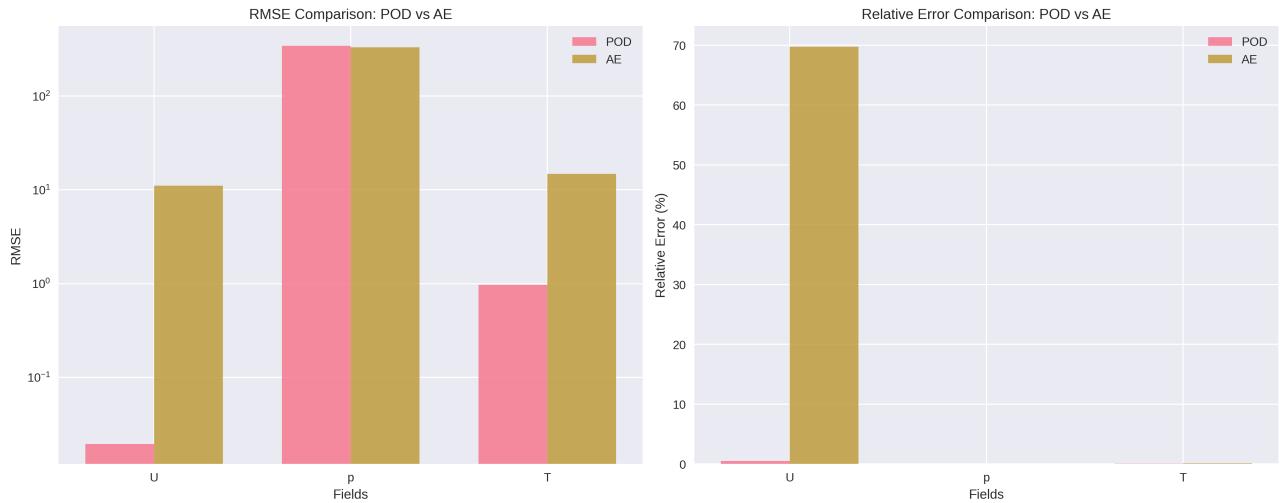


FIGURE 4.8 – Comparaison de la précision entre les méthodes POD et AE

4.2 Géométrie et conditions aux bords de l'extrusion de profilé

4.2.1 Cas simple de profilé

La géométrie dans ce cas est un profilé simple d'extrusion en 2D (Figure 4.9), qui est construit avec Gmsh. Cette géométrie est utilisée pour tester la convergence des méthodes de réduction de modèle POD et (AE) afin de vérifier si ces méthodes sont capables de capturer les variations de la solution avec un nombre réduit de points de maillage, ainsi que l'influence des conditions aux limites et des paramètres associés. Dans le tableau 4.4, nous avons résumé les conditions aux limites associées à cette géométrie.

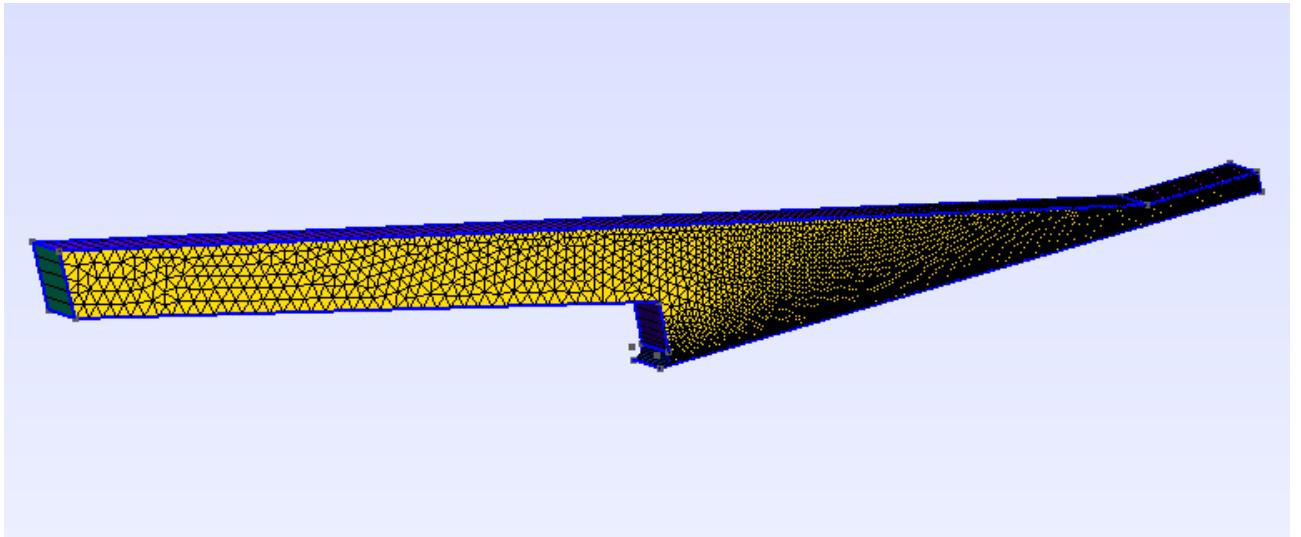


FIGURE 4.9 – Géométrie 3D du profilé d'extrusion

4.2.2 Résultats de réduction de modèle POD et AE

Dans cette section, comme précédemment, nous avons généré 50 snapshots de manière aléatoire en utilisant le Latin Hypercube Sampling (LHS) pour les paramètres de la solution du profilé, afin de construire la base réduite avec les méthodes POD et AE avec les plages définies dans le tableau 4.5. Pour ce test, nous avons pris μ de cette façon, $\mu = (U, T, A, K, a, n, b)$ comme vecteur de paramètres d'entrée, où U est la vitesse du câble, T est la température du câble, les autres composantes sont des paramètres de la loi de Carreau

TABLE 4.4 – Conditions aux bords associées aux différentes surfaces du domaine

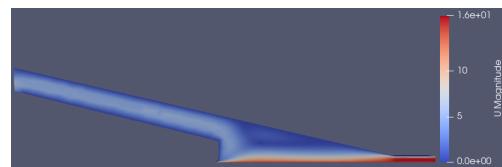
Surface	P (Pression)	U (Vitesse)	T (Température)
front	empty	empty	empty
back	empty	empty	empty
inlet	zeroGradient	flowRateInletVelocity	fixedValue($T = T_0$)
outlet	fixedValue ($P = 0$)	inletOutlet	zeroGradient
top	zeroGradient	noSlip	fixedValue ($T = T_0$)
bottom	zeroGradient	noSlip	fixedValue ($T = T_0$)
arc	zeroGradient	noSlip	fixedValue ($T = T_0$)
Moving	zeroGradient	fixedValue ($U = (u_{cable} \quad 0 \quad 0)$)	fixedValue ($T = T_{cable}$)

Yassoda. La base POD a été construire à l'aide de la méthode SVD avec 30 modes. Pour la méthode AE, nous avons utilisé un encodeur composé de trois couches (100, 50, 15) et un découpeur symétrique (15, 50, 100), avec également 15 pour la dimension de latent.

L'interpolation des solutions réduites a été réalisée à l'aide de la méthode des fonctions à base radiale (RBF). Nous présentons les résultats de la prédiction sur le profilé d'extrusion avec les méthodes de réduction de modèle POD, AE, en les comparant à la solution calculer avec OpenFoam pour les trois variables d'état, et en fournissant un tableau comparatif des erreurs pour les deux méthodes sur la vitesse, la pression et la température. Dans les figures 4.10, 4.11 et 4.12, nous avons illustré les solutions de haut fidélité, POD et AE respectivement, pour les trois variables d'état U , P et T pour ces paramètres la vitesse du câble $U = 15.002$, la température $T = 453.15$, les paramètres de la loi de Carreau Yassoda $A = 15219.6$, $k = 0.117575$, $a = 2.24$, $n = 0.153323$, et $b = 0.025$.

TABLE 4.5 – La plage des paramètres utilisés pour la construction de la base réduite.

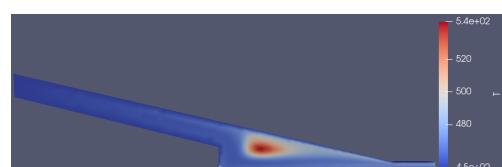
Paramètre	Minimum	Maximum	Unité
U	12	18	m/s
T	443.15	463.15	K
A	1687.88	17111.9	Pa·s
k	0.03003	0.18211	s
a	0.50188	3.9756	-
n	0.1	0.3323	-
b	0.01776	0.03370	k^{-1}



U

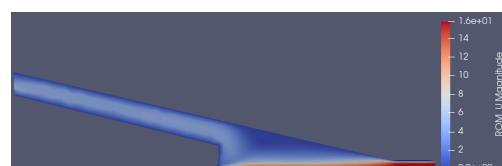


P



T

FIGURE 4.10 – Solution de Haut fidélité



U

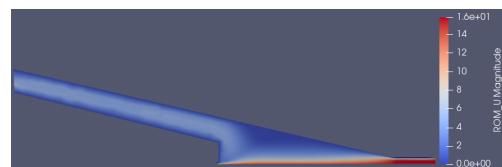


P

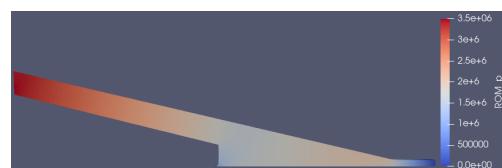


T

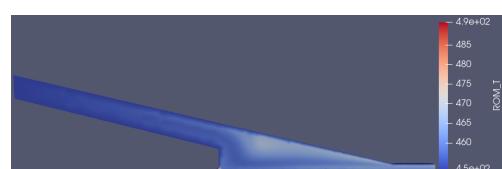
FIGURE 4.11 – Solution de POD



U

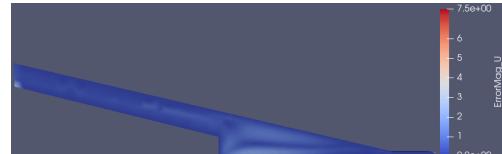


P



T

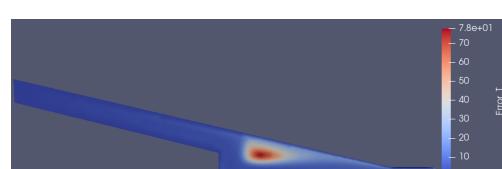
FIGURE 4.12 – Solution de AE



U



P



T

FIGURE 4.13 – Erreur entre la solution de référence et la solution prédite par les méthodes POD

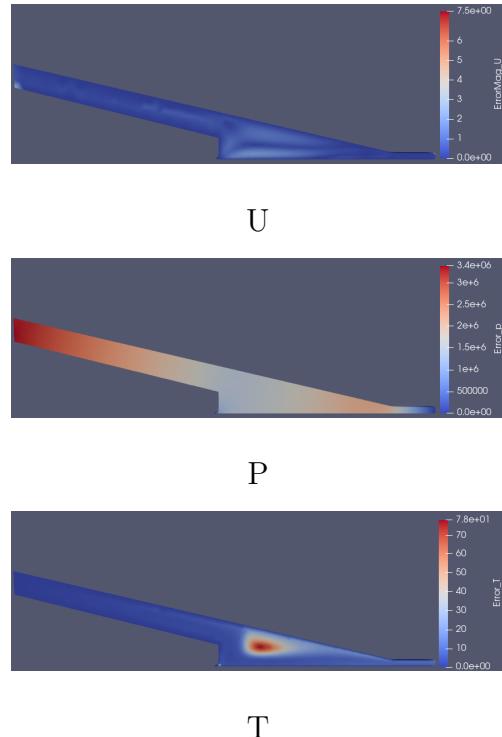


FIGURE 4.14 – Erreur entre la solution de référence et la solution prédictive par les méthodes AE

TABLE 4.6 – Comparaison des erreurs L2 entre POD et AE pour deux différents cas

Variable	Erreur relative L2 POD	Erreur relative L2 AE
U (vitesse)	0.083	0.0831
P (pression)	0.497	0.4978
T (température)	0.0350	0.03507
U (vitesse)	0.0837	0.08308
P (pression)	0.497	0.4978
T (température)	0.03506	0.03506

Dans le tableau 4.6, nous avons présenté les erreurs L^2 et les erreurs relatives pour deux différents cas de la vitesse du câble, $U = 15.002$ et $U = 15.006$, et différents cas de $A = 15219.6$ et $A = 15219.75$ pour les deux méthodes de réduction de modèle POD et AE. On peut constater que les erreurs sont relativement faibles pour les deux méthodes, mais la méthode POD semble donner des résultats plus précis que la méthode AE, en particulier pour la vitesse et la température.

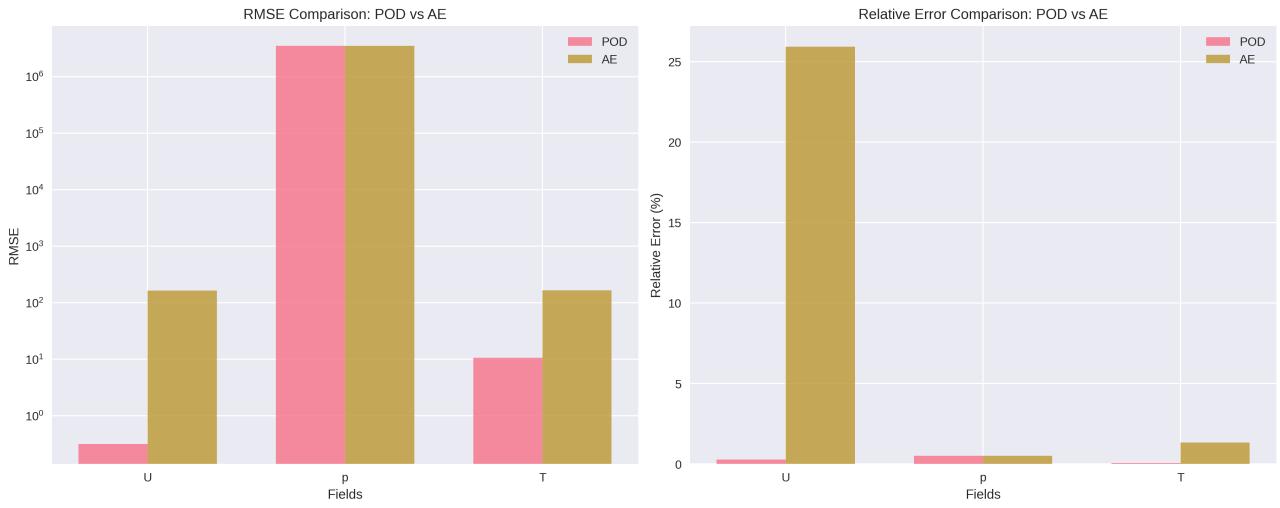


FIGURE 4.15 – Comparaison de la précision entre les méthodes POD et AE

4.3 Géométrie et conditions aux bords de l’extrusion de profilé cas axisymétrique

4.3.1 Géométrie en axisymétrie

La géométrie de l’extrusion de profilé est construite en 3D sous Gmsh (Figure 4.16). La Figure 4.17 présente une vue 2D de cette géométrie, avec les conditions aux bords associées. Pour les simulations, nous nous intéressons au cas axisymétrique de l’extrusion de profilé, ce qui permet de réduire le nombre de degré de liberté du maillage et par conséquent, le temps de calcul.

TABLE 4.7 – Conditions aux bords associées aux différentes surfaces du domaine

Surface	P (Pression)	U (Vitesse)	T (Température)
WedgeL	wedge	wedge	wedge
WedgeR	wedge	wedge	wedge
inlet	zeroGradient	fixedValue ($U = (u_{inlet} \ 0 \ 0)$)	fixedValue ($T = T_0$)
outlet	fixedValue ($P = 0$)	slip	zeroGradient
top	zeroGradient	noslip	fixedValue ($T = T_0$)
bottom	zeroGradient	noslip	fixedValue ($T = T_0$)
freeSurf	fixedValue ($P = 0$)	slip	zeroGradient
movingwall	zeroGradient	fixedValue ($U = (u_{cable} \ 0 \ 0)$)	fixedValue ($T = T_{cable}$)

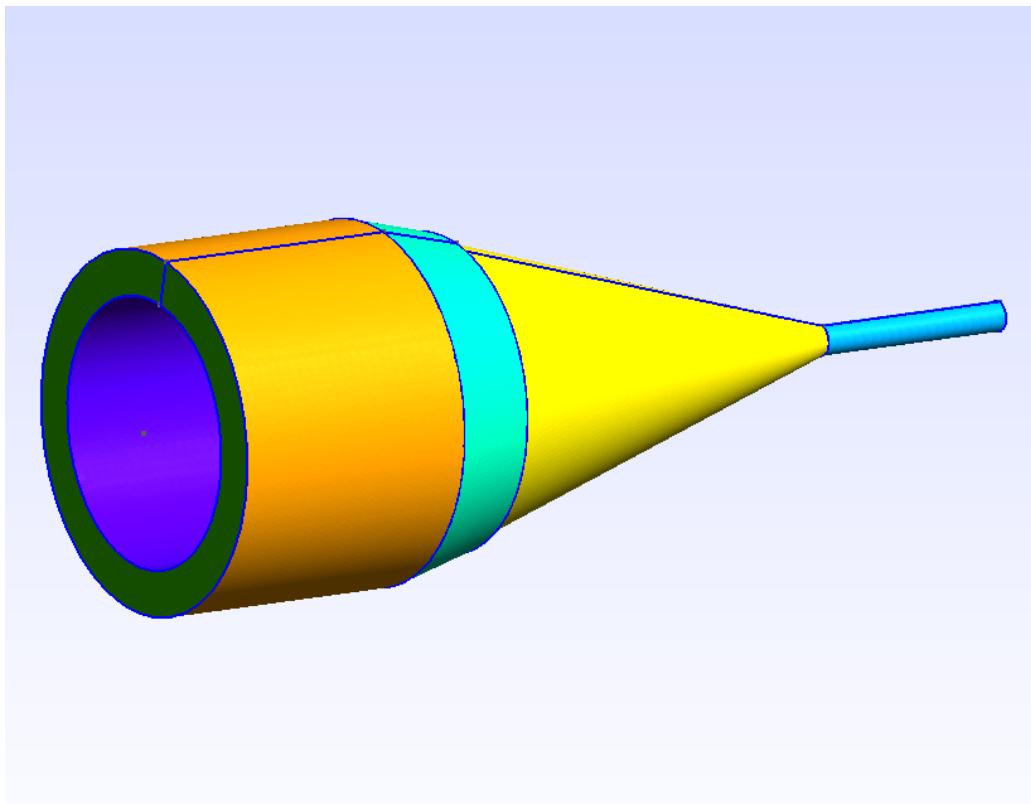


FIGURE 4.16 – Géométrie 3D du profilé d'extrusion

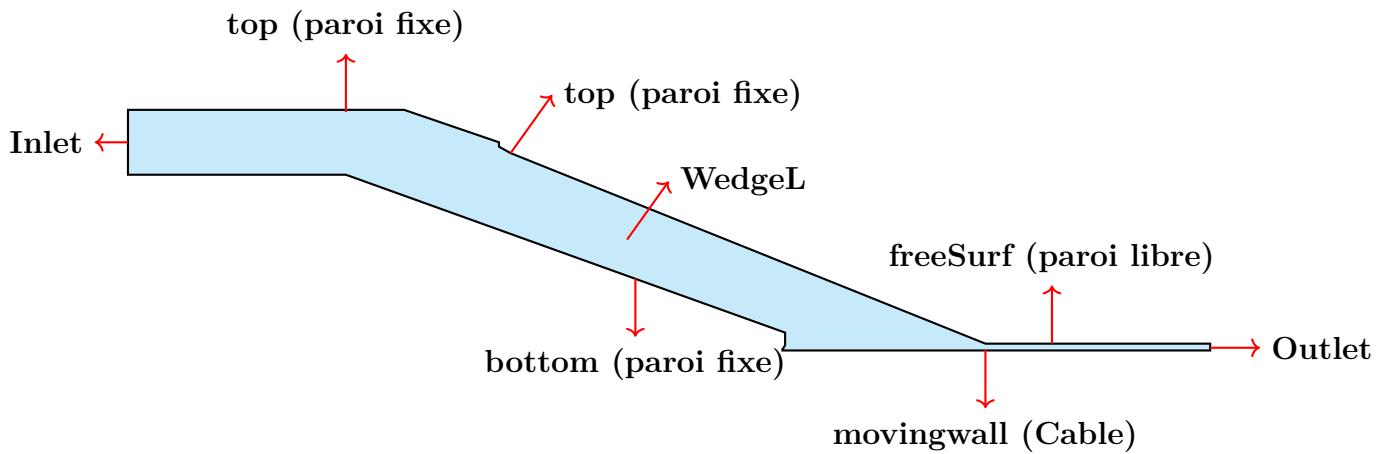


FIGURE 4.17 – Géométrie 2D du profilé d'extrusion, avec indication des conditions aux bords

Remarque 4.3.1. Pour les simulations d'axisymétries sous OpenFOAM, il est important de définir la géométrie comme un secteur de cercle avec un angle ne dépassant pas 5° , et d'appliquer la condition "wedge" sur les deux surfaces d'angle. Pour plus de détails, voir [Lien](#) le guide officiel OpenFOAM sur les conditions aux bords.

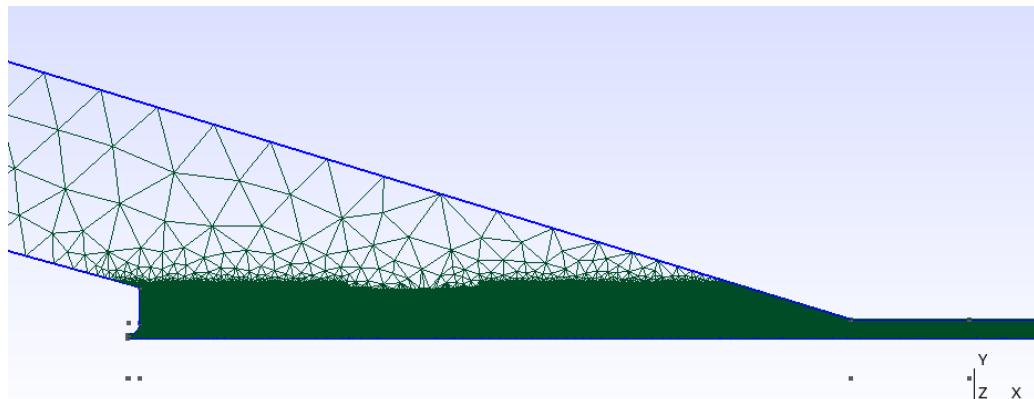
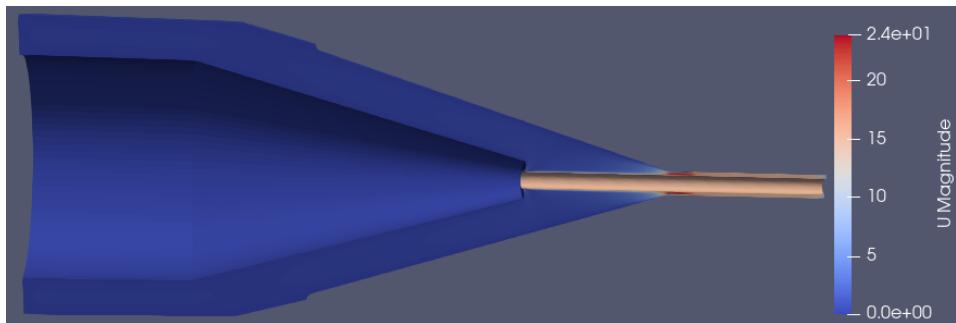


FIGURE 4.18 – Maillage plus fin autour de la sortie du profilé d’extrusion

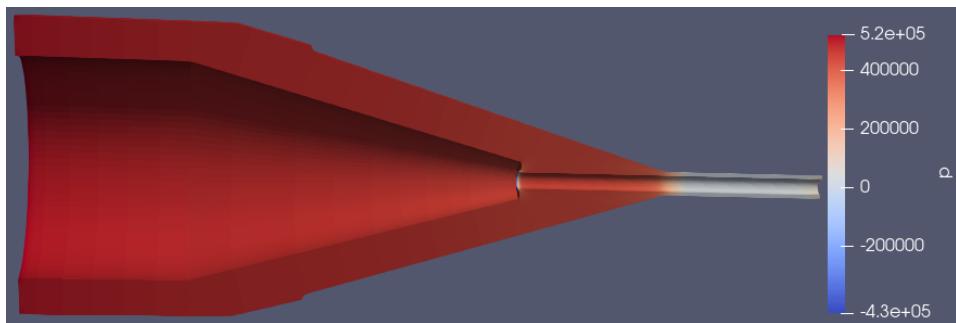
4.3.2 Simulation de l’extrusion de profilé

Dans cette section, nous présentons les résultats d’un cas de validation réalisé avec les mêmes conditions aux limites et les mêmes paramètres de la loi de Carreau-Yassoda que ceux utilisés dans la simulation sous CompuPlast, à l’exception de la partie solide de l’extrusion qui n’est pas prise en compte ici. Le cas de validation est construit avec les paramètres suivants :

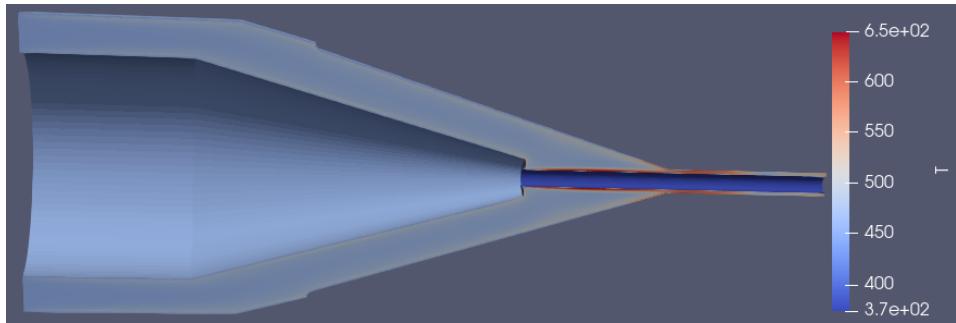
$U_{cable} = 15$, $U_{inlet} = 0.15$ (déterminé à l’aide du débit massique), $T_0 = 453.15$, $A = 15219.5$, $k = 0.11758$, $a = 0.53710$, $n = 0.15332$, $b = 0.03003$ et pour $T_{cable} = 373.15$ puisque la partie solide du câble qui affecte la température de l’extrusion n’a pas été prise en compte dans cette simulation.



U



P



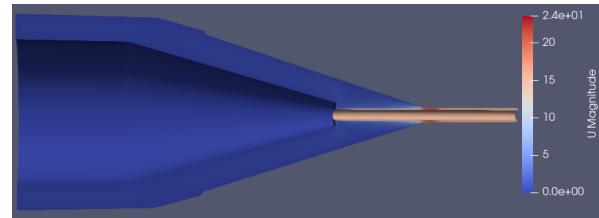
T

FIGURE 4.19 – Solution de validation de l’extrusion de profilé

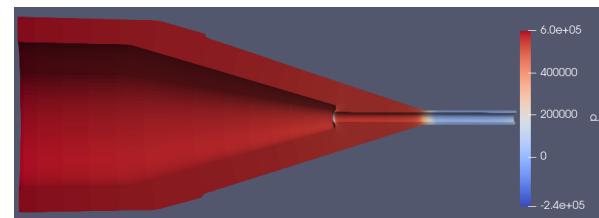
4.3.3 Résultats de réduction de modèle POD et AE pour l’axisymétrie

Dans cette section, nous nous intéressons à la réduction de modèle pour les quatres matériaux présentés dans le tableau 1.1 en utilisant la géométrie axisymétrique de l’extrusion de profilé. Nous avons généré 50 snapshots de manière aléatoire en utilisant une loi uniforme

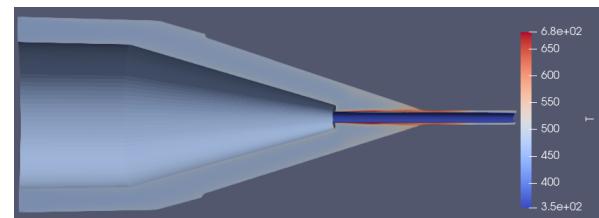
pour les paramètres de la solution du profilé, afin de construire la base réduite avec les méthodes POD et AE.



U



P



T

FIGURE 4.20 – Solution de Haut fidélité

Pour ce test, nous avons choisi le vecteur de paramètres d'entrée $\mu = (A, K, a, n, b)$, où tous ces paramètres sont issus de Carreau Yassoda. Les plages de valeurs utilisées pour chaque paramètre correspondent aux minimums et maximums indiqués dans le tableau 1.1, afin d'enrichir la base réduite et de capturer les variations de la solution pour différents matériaux, rendant ainsi l'espace paramétrique plus réaliste.

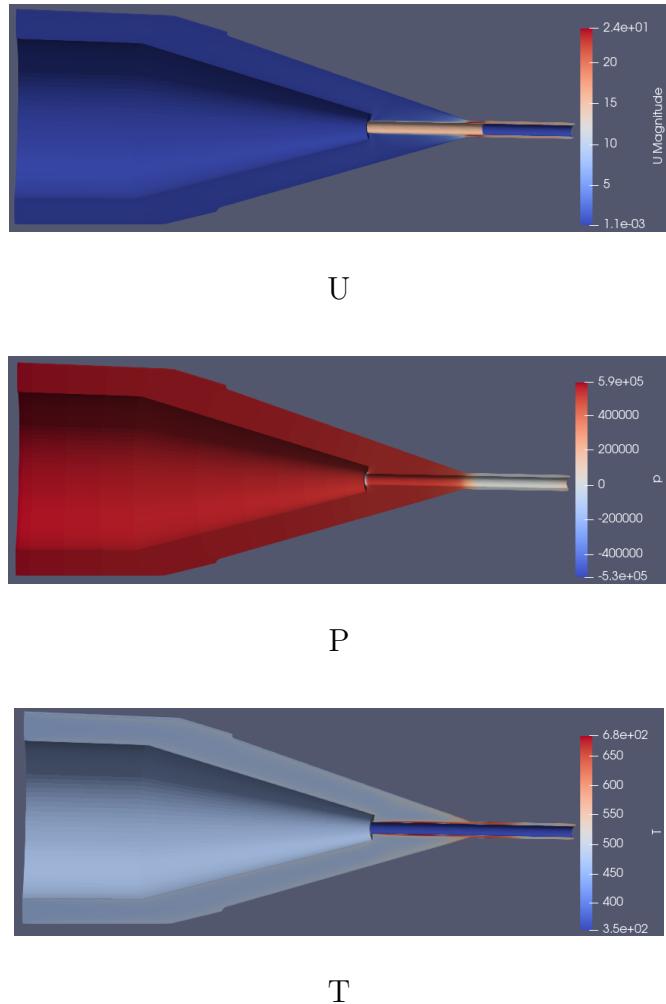
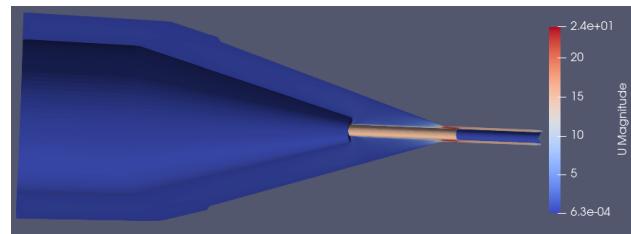


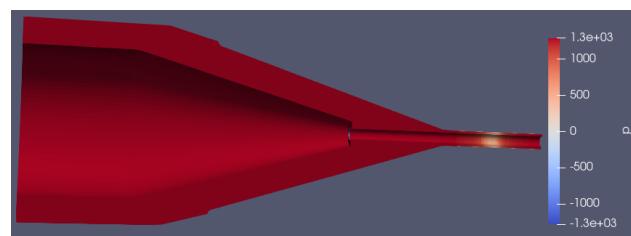
FIGURE 4.21 – Solution de POD

La base POD a été construite à l'aide de la méthode SVD avec 25 modes. Pour la méthode AE, nous avons utilisé un encodeur composé de trois couches (256, 128, 64) et un décodeur symétrique (64, 128, 256), avec également 64 pour la dimension de latent. L'interpolation des solutions réduites a été réalisée à l'aide de la méthode des fonctions à base radiale (RBF). Nous présentons les résultats de la prédiction sur la géométrie axisymétrique pour le premier matériau, obtenus avec les méthodes de réduction de modèle POD et AE, en les comparant à la solution de référence pour les trois variables d'état et pour les quatre matériaux. Un tableau comparatif des erreurs pour les deux méthodes, concernant la vitesse, la pression et la température, est également fourni. Dans les figures 4.20, 4.21 et 4.22, nous avons illustré les solutions de haut fidélité, POD et AE respectivement, pour les trois variables d'état U , P et T . Pour ce cas les paramètres utilisés pour la loi de Carreau Yassoda sont ceux du tableau,

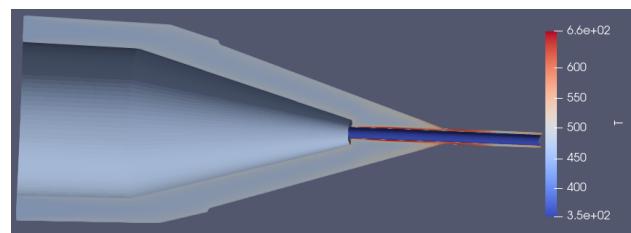
et pour les vitesses et température, nous avons pris $U_{cable} = 15$, $U_{inlet} = 0.15$, $T_0 = 463.15$ et $T_{cable} = 353.15$.



U



P

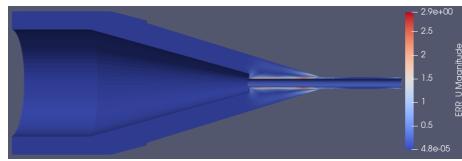


T

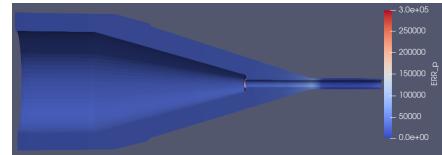
FIGURE 4.22 – Solution de AE

TABLE 4.8 – Comparaison des temps de calcul pour les quatre matériaux étudiés, et accélération obtenue par POD et AE

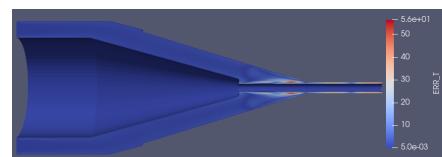
Cas	Haute Fidélité (s)	POD (s)	AE (s)	Acceleration POD	Acceleration AE
Cas 1	130	0.0463	0.0093	2808	13978
Cas 2	130	0.0265	0.0091	4906	14286
Cas 3	134	0.0351	0.0088	3818	15227
Cas 4	136	0.0206	0.0097	6602	14021



U

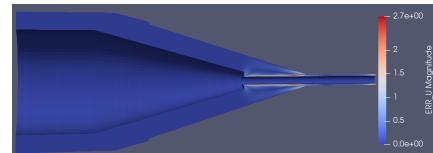


P

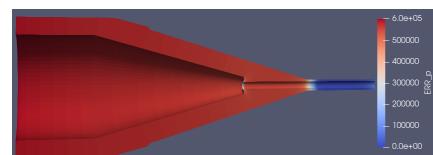


T

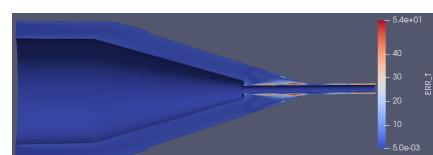
FIGURE 4.23 – Erreur entre la solution de référence et la solution prédite par les méthodes POD



U



P



T

FIGURE 4.24 – Erreur entre la solution de référence et la solution prédite par les méthodes AE

TABLE 4.9 – Comparaison des erreurs L2 et relatives pour POD et AE sur les quatre matériaux étudiés

Cas	Variable	Erreurs L2 POD	Erreurs relatives L2 POD (%)	Erreurs L2 AE	Erreurs relatives L2 AE (%)
1	U	0.657774	8.21	0.646653	8.08
	T	16.992360	3.24	16.523888	3.15
	P	26735.1749	5.35	498857.481	99.77
2	U	0.530335	6.71	0.320795	4.06
	T	23.237770	4.30	25.272972	4.68
	P	124534.4436	21.66	573805.267	99.80
3	U	0.642740	7.98	0.888022	11.03
	T	16.626969	3.38	42.995214	8.75
	P	147487.2626	59.88	245133.010	99.52
4	U	0.695092	8.82	0.303387	3.85
	T	39.909667	7.25	35.961548	6.53
	P	215460.6236	34.66	620481.148	99.81

Dans le tableau 4.9, nous avons présenté les erreurs L^2 et les erreurs relatives pour la prédiction des quatre matériaux étudiés, en utilisant les deux méthodes de réduction de modèle POD et AE, via la bibliothéque EZyRB. On observe que les deux méthodes fournissent des résultats au niveau d'erreur est significativement plus élevée pour l'AE que la POD et plus élevée pour la pression que la vitesse et la température.

Cependant, il est important de noter que les résultats sont encore préliminaires. Notamment, dans le cas d'équation non linéaire où la dépendance de la solution aux paramètres d'entrée est forte, comme pour les écoulements de polymères modélisés par la loi de Carreau-Yassoda, l'échantillon de 50 snapshots est relativement faible considérant le nombre élevé de paramètres du modèle réduit et la non-linéarité de la solution de chaque snapshot. Malgré ces limites, ces méthodes restent un bon compromis pour la réduction de modèle, permettant d'accélérer les simulations tout en conservant une précision acceptable. On constate que les erreurs sur la vitesse sont faibles, tandis que celles sur la pression sont relativement plus élevées. Cette différence s'explique par les gradients de pression importants dans l'écoulement, les maillages pas suffisamment fins et aussi le niveau de convergence de solveur itératif utilisé pour la résolution de l'équation de la conservation de la masse est insuffisant. Cependant, nous avons observé une réduction significative des temps de calcul, comme indiqué dans le tableau 4.8, où les temps de calcul pour la simulation haute fidélité sont très élevés par

rapport à ceux obtenus avec les méthodes de réduction de modéle.

Conclusion

En synthèse, nous avons réussi à développer et implémenter :

- Une loi de Carreau-Yassoda dépendante de la température dans OpenFoam, permettant ainsi de modéliser l'interaction entre l'équation de Navier-Stokes et l'équation de la chaleur pour modéliser les écoulements polymères fondus avec prise en compte de l'auto-échauffement.
- Une automatisation en Python des étapes nécessaires à la construction de modèles réduits POD et AE à l'aide des librairies EZyRB et OpenFOAM.
- Un calcul automatique de la norme L^2 entre la solution haute fidélité et la prédiction du modèle réduit obtenue par les méthodes de réduction de modèle POD et AE.

Dans le chapitre 4, nous avons présenté les résultats préliminaires pour valider le processus de construction de modèles réduits et analyser leurs performances. Nous avons appliqué les méthodes de réduction de modèle POD et AE pour les cas tests de la cavité, d'un profilé d'extrusion en 2d et de profilé en axisymétrique. En synthèse, nous pouvons conclure que :

- La méthode POD est plus précise que la méthode AE pour les configurations étudiées.
- La précision est plus élevée pour la vitesse et la température que pour la pression.
- Les temps de calcul sont considérablement réduits, passant de plusieurs minutes pour la simulation haute fidélité à quelques millisecondes pour les méthodes de réduction de modèle.

Perspectives

Les travaux réalisés étant préliminaires, plusieurs axes de progrès sont envisageables pour améliorer la précision des modèles réduits et leurs pertinences physiques pour les applications de polymères fondus chez ACOME. Notamment :

- Elargir le nombre de paramètres d'entrées pour la construction de la base réduite, afin de capturer l'ensemble de la physique utile pour anticiper l'extrudabilité de différents polymères fondus.
- Optimiser le nombre de snapshots pour la construction de la base réduite, en tenant compte de la non-linéarité de la solution et de la dépendance aux paramètres d'entrées, afin d'améliorer la précision des solutions prédites par les modèles réduits.
- Effectuer une études de convergences de maillages pour ajuster la finesse des maillages pour trouver un compromis entre la précision des solutions et le temps de calcul.
- Optimiser les paramètres de la POD et de l'AE pour améliorer la précision des solutions prédites, notamment en ajustant le nombre de modes pour la POD et la dimension latente pour l'AE.
- Remplacer les solveurs itératifs par des solveurs directs pour la résolution de l'équation de la conservation de la masse et de la chaleur, afin d'améliorer la précision des solutions.
- Enrichir la physique de la simulation en utilisant une méthode de transfert de chaleur conjuguée qui intègre les effets thermiques de la partie solide du câble ce qui modélisera la température du polymère fondu.

Annexe A

Annexes

A.1 Code de la loi de Carreau-Yasuda

Dans cette section, nous commençons par présenter la structure des fichiers sur OpenFoam pour l'implémentation de la loi de Carreau-Yasuda, puis nous détaillons le code C++ correspondant. Ce code utilisé pour modéliser la viscosité dépendante de la température dans les simulations OpenFOAM. Dans cette implémentation, le paramètre A représente le paramètre μ_0 de la loi de Carreau-Yassoda. Nous avons adopté cette notation pour rester cohérent avec les conventions utilisées chez ACOME.

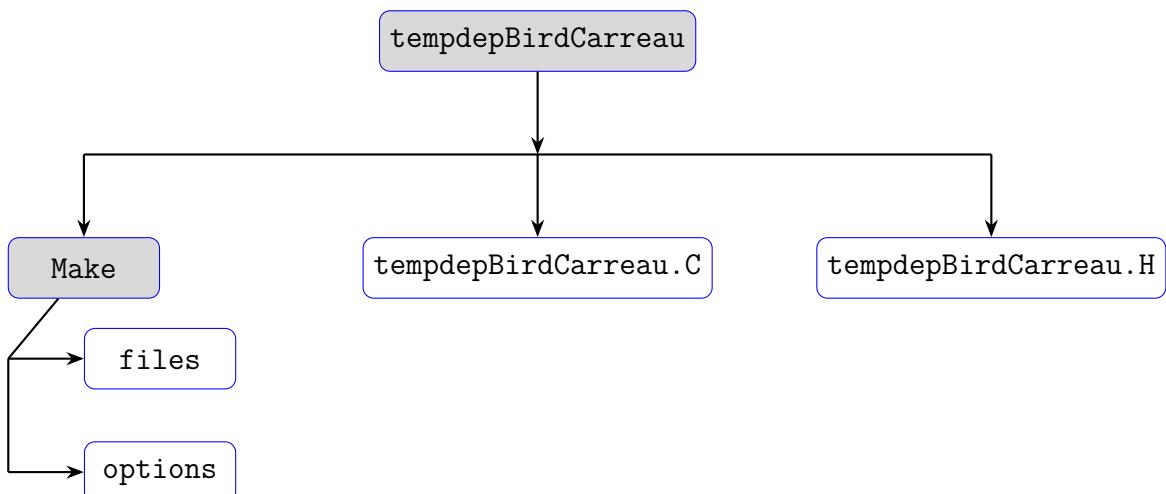


FIGURE A.1 – La structure des fichiers

tempdepBirdCarreau.C

```

// * * * * * * * * * * * * * * * * Protected Member Functions * * * * * * * * * *
* //

Foam::tmp<Foam::volScalarField>
Foam::viscosityModels::tempdepBirdCarreau::calcNu() const
{
    const volScalarField& T= U_.mesh().lookupObject<volScalarField>("T");

    return
        A_ * exp(-b_ * (T - T0_))
        * pow(scalar(1) + pow(k_ * exp(-b_ * (T - T0_)))
        * strainRate(), a_), (n_ - 1.0)/a_);
}

// * * * * * * * * * * * * * * * * Constructors * * * * * * * * * * * * * * * *
//



Foam::viscosityModels::tempdepBirdCarreau::tempdepBirdCarreau
(
    const word& name,
    const dictionary& viscosityProperties,
    const volVectorField& U,
    const surfaceScalarField& phi
)
:
    viscosityModel(name, viscosityProperties, U, phi),
    tempdepBirdCarreauCoeffs_
    (
        viscosityProperties.optionalSubDict(typeName + "Coeffs")
    ),
    A_("A", dimViscosity, tempdepBirdCarreauCoeffs_),
    //nuInf_("nuInf", dimViscosity, tempdepBirdCarreauCoeffs_),
    k_("k", dimTime, tempdepBirdCarreauCoeffs_),
    n_("n", dimless, tempdepBirdCarreauCoeffs_),
    a_

```

```

(
    tempdepBirdCarreauCoeffs_.getOrDefault
    (
        "a",
        dimensionedScalar("a", dimless, 2.0 )
    )
),

b_("b", dimensionSet(0, 0, 0, -1, 0, 0, 0), tempdepBirdCarreauCoeffs_),
T0_("T0", dimTemperature, tempdepBirdCarreauCoeffs_),
nu_
(
    IOobject
    (
        name,
        U_.time().timeName(),
        U_.db(),
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    calcNu()
)
{}

// * * * * * * * * * * * * * * * * Member Functions * * * * * * * * * * * * * *
//



bool Foam::viscosityModels::tempdepBirdCarreau::read
(
    const dictionary& viscosityProperties
)
{
    viscosityModel::read(viscosityProperties);

    tempdepBirdCarreauCoeffs_ =
        viscosityProperties.optionalSubDict(typeName + "Coeffs");
}

```

```

tempdepBirdCarreauCoeffs_.readEntry("A", A_);
//tempdepBirdCarreauCoeffs_.readEntry("nuInf", nuInf_);
tempdepBirdCarreauCoeffs_.readEntry("k", k_);
tempdepBirdCarreauCoeffs_.readEntry("n", n_);
a_ = tempdepBirdCarreauCoeffs_.getOrDefault
(
    "a",
    dimensionedScalar("a", dimless, 2.0)
);
tempdepBirdCarreauCoeffs_.readEntry("b", b_);
tempdepBirdCarreauCoeffs_.readEntry("T0", T0_);

return true;
}

// ****
//
```

tempdepBirdCarreau.H

```

// * * * * *
//



namespace Foam
{
namespace viscosityModels
{

/*
-----*/
```

Class tempdepBirdCarreau Declaration

```

/*-----*/
*/



class tempdepBirdCarreau
:
public viscosityModel
{
// Private data

dictionary tempdepBirdCarreauCoeffs_;


dimensionedScalar A_;
//dimensionedScalar nuInf_;
dimensionedScalar k_;
dimensionedScalar n_;
dimensionedScalar a_;
dimensionedScalar b_;
dimensionedScalar T0_;



protected:

// Protected Data

volScalarField nu_;



// Protected Member Functions

// Calculate and return the laminar viscosity
tmp<volScalarField> calcNu() const;





public:

// Runtime type information
TypeName("tempdepBirdCarreau");

```

```

// Constructors

//- Construct from components
tempdepBirdCarreau
(
    const word& name,
    const dictionary& viscosityProperties,
    const volVectorField& U,
    const surfaceScalarField& phi
);

// Destructor
virtual ~tempdepBirdCarreau() = default;

// Member Functions

//- Return the laminar viscosity
virtual tmp<volScalarField> nu() const
{
    return nu_;
}

//- Return the laminar viscosity for patch
virtual tmp<scalarField> nu(const label patchi) const
{
    return nu_.boundaryField()[patchi];
}

//- Correct the laminar viscosity
virtual void correct()
{
    nu_ = calcNu();
}

```

```

// - Read transportProperties dictionary
virtual bool read(const dictionary& viscosityProperties);

};

// * * * * *
// 

} // End namespace viscosityModels
} // End namespace Foam

// * * * * *
// 

#endif

// *****
// 

```

files

```

tempdepBirdCarreau.C
LIB = $(FOAM_USER_LIBBIN)/libusertempdepBirdCarreau

```

options

```

EXE_INC = \
-I$(LIB_SRC)/transportModels/incompressible/lnInclude/ \
-I$(LIB_SRC)/finiteVolume/lnInclude

LIB_LIBS = \
-lfiniteVolume

```

A.2 Code d'ajouter l'équation d'energie

Dans cette section, de même façon, nous avons présenté la structure des fichiers à adapter sur OpenFOAM pour le solveur tempSimpleFoam tout en donnant le code C++ correspondant. Nous avons présenté l'équation d'énergie qui est utilisé pour modéliser la dépendance thermique ainsi que les parties de code qu'il faut adapter pour prendre cette situation en compte.

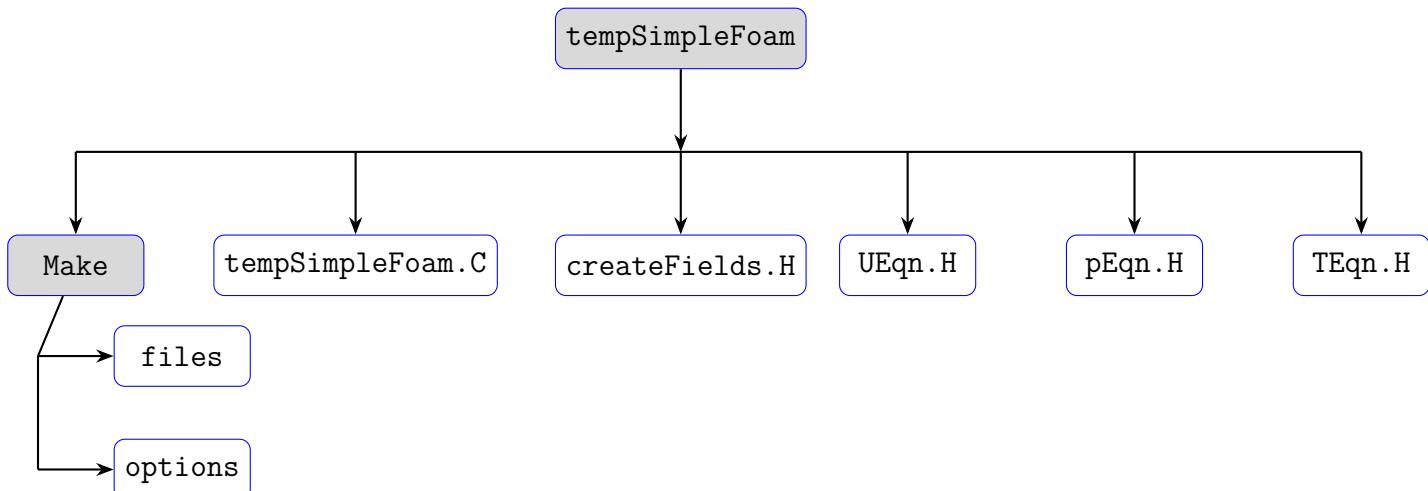


FIGURE A.2 – Structure du solveur tempSimpleFoam avec tous ses fichiers à adapter

tempSimpleFoam.C

```
{  
    #include "UEqn.H"  
    // --- Velocity equation  
    #include "pEqn.H"  
    // --- Pressure equation  
    #include "TEqn.H"  
    // --- Temperature equation  
}
```

TEqn.H

```
Info << "Solving thermal energy equation with viscous dissipation\n" << endl
;

// **** Solving thermal energy equation ****
tmp<fvScalarMatrix> TEqn
(
    fvm::div(phi, T)                                // convection
    - fvm::laplacian(TempD_, T) = viscousDissipation // conduction avec
    TempD_ = k / (rho Cp)
);

TEqn.ref().relax();
solve(TEqn);
```

createFields.H

```
#include "createRDeltaT.H"

Info<< "Reading field T\n" << endl;
volScalarField T
(
    IOobject
    (
        "T",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

```

#include "createPhi.H"

label pRefCell = 0;
scalar pRefValue = 0.0;
setRefCell(p, pimple.dict(), pRefCell, pRefValue);
mesh.setFluxRequired(p.name());

singlePhaseTransportModel laminarTransport(U, phi);

autoPtr<incompressible::turbulenceModel> turbulence
(
    incompressible::turbulenceModel::New(U, phi, laminarTransport)
);

// diffusivity [TempD]
dimensionedScalar TempD_("TempD", laminarTransport.lookup("TempD"));
// // density [rho]
// dimensionedScalar rho_("rho", laminarTransport.lookup("rho"));
// // specific heat [Cp]
// dimensionedScalar Cp_("Cp", laminarTransport.lookup("Cp"));

#include "createMRF.H"
#include "createFvOptions.H"

```

files

```

tempSimpleFoam.C

EXE = $(FOAM_APPBIN)/tempSimpleFoam

```

options

```
EXE_INC = \
-I$(LIB_SRC)/finiteVolume/lnInclude \
-I$(LIB_SRC)/meshTools/lnInclude \
-I$(LIB_SRC)/sampling/lnInclude \
-I$(LIB_SRC)/TurbulenceModels/turbulenceModels/lnInclude \
-I$(LIB_SRC)/TurbulenceModels/incompressible/lnInclude \
-I$(LIB_SRC)/transportModels \
-I$(LIB_SRC)/transportModels/incompressible/singlePhaseTransportModel \
-I$(LIB_SRC)/dynamicMesh/lnInclude \
-I$(LIB_SRC)/dynamicFvMesh/lnInclude \
-I$(LIB_SRC)/regionFaModels/lnInclude \
-I$(LIB_SRC)/transportModels/incompressible/lnInclude

EXE_LIBS = \
-L$(FOAM_USER_LIBBIN) \
-lfiniteVolume \
-lfvOptions \
-lmeshTools \
-lsampling \
-lturbulenceModels \
-lincompressibleTurbulenceModels \
-lincompressibleTransportModels \
-ldynamicMesh \
-ldynamicFvMesh \
-ltopoChangerFvMesh \
-latmosphericModels \
-lregionFaModels \
-lfiniteArea
```

A.3 Géometrie extruder sur GMSH

Dans cette section, nous présentons le code GMSH utilisé pour générer la géométrie de la cavité 2D extrudée en 3D. Ce code est utilisé pour un cas test de validation de notre

configuration et aussi pour comprendre comment générer la géométrie pour OpenFOAM.

Cavity extrudée

```
// Gmsh geometry for 2D cavity, extruded to 3D one-cell thick
L = 1.0;           // cavity length [m]
H = 1.0;           // cavity height [m]
thickness = 0.1;   // extrusion thickness [m] (one cell thick)

// Points (corner coordinates of the square, with mesh size)
Point(1) = {0, 0, 0, 0.1};
Point(2) = {L, 0, 0, 0.1};
Point(3) = {L, H, 0, 0.1};
Point(4) = {0, H, 0, 0.1};

// Lines connecting points (square edges)
Line(1) = {1, 2}; // bottom edge
Line(2) = {2, 3}; // right edge
Line(3) = {3, 4}; // top edge
Line(4) = {4, 1}; // left edge

Line Loop(5) = {1, 2, 3, 4};
Plane Surface(6) = {5}; // define the square surface

// Structured mesh: 50x50 cells on the surface
Transfinite Line {1, 2, 3, 4} = 51; // 11 points -> 10 divisions per edge
Transfinite Surface {6};
Recombine Surface {6}; // make quadrilateral cells on Surface 6

// Extrude the surface in z-direction to create a volume one cell thick
surfaceVector[] = Extrude {0, 0, thickness} {
    Surface{6};
    Layers{1};      // one layer in extrusion (2D behavior)
    Recombine;     // recombine to get hex elements (no division in z)
};
```

```
// Assign physical names to boundary surfaces (using indices from extrusion)
Physical Surface("front") = surfaceVector[0]; // front face (extruded
surface)
Physical Surface("bottom") = surfaceVector[2]; // bottom wall
Physical Surface("right") = surfaceVector[3]; // right wall
Physical Surface("top") = surfaceVector[4]; // top wall (moving lid)
Physical Surface("left") = surfaceVector[5]; // left wall
Physical Surface("back") = {6}; // back face (original surface)
Physical Volume("internal") = surfaceVector[1]; // volume of the cavity
```

Bibliographie

- [1] K. Ait Cheikh, A. (2020). Évaluation numérique de perte de charge dans une filière plane d'extrusion d'un écoulement viscoélastique. Université du Québec en Abitibi-Témiscamingue, École de Génie.
- [2] Sirovich, L. (1987). Turbulence and the dynamics of coherent structures. Part I-III. *Quarterly of Applied Mathematics*, 45(3), 561–590.
- [3] Holmes, P., Lumley, J. L., & Berkooz, G. (1996). *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press.
- [4] Moore, B. C. (1981). Principal Component Analysis in Linear Systems : Controllability, Observability, and Model Reduction. *IEEE Transactions on Automatic Control*, 26(1), 17–32.
- [5] Antoulas, A. C. (2005). *Approximation of Large-Scale Dynamical Systems*. SIAM.
- [6] Ito, K., & Ravindran, S. S. (1998). Reduced basis method for control problems governed by PDEs. *SIAM Journal on Control and Optimization*, 36(3), 1–27.
- [7] Chapitre V : Extrusion des Thermoplastiques. Tubes et profilés.
Site : <https://choucheneslim.wordpress.com/>
Article cours et TP : 10- Procédés de mise en forme des matières plastiques.
PLAYLIST YOUTUBE « Procédés - Extrusion des thermoplastiques » :
<https://www.youtube.com/playlist?list=PLVdWnPZXu-Oi835AX9dqXHLDd1KTo6WwK>
- [8] Vidal, J., & Nóbrega, J. M. (2023). An Enhanced Temperature Control Approach to Simulate Profile Extrusion. *Soprefa-Componentes Industriais SA, R. Alfredo Henriques*,

4520-909 Mosteiró, Portugal ; Institute for Polymers and Composites, University of Minho, Campus de Azurém, 4800-058 Guimarães, Portugal. Correspondence

- [9] Law of Carreau-Yasuda with dependence on temperature.

Site : [ref](#)

- [10] Wassgren, C. (2021). Notes on Thermodynamics, Fluid Mechanics, and Gas Dynamics. Lecture notes, Purdue University.

Site : [ref](#)

- [11] Magalhães, A. C. L. (2016). OpenFOAM® simulation of the injection moulding filling stage. Dissertação de Mestrado, Universidade do Minho, Escola de Engenharia. Trabalho efetuado sob a orientação du Doutor Luís Jorge Lima Ferrás et du Doutor Célio Bruno Pinto Fernandes.

- [12] McDonough, J. M. (2009). Lectures In Elementary Fluid Dynamics : Physics, Mathematics and Applications. University of Kentucky.

Disponible sur : [ref](#)

- [13] Bird, R. B., Armstrong, R. C., & Hassager, O. (1987). Dynamics of Polymer Liquids, Volume 1. Wiley-Interscience.

- [14] Un fluide newtonien, c'est quoi ? Navier Stokes et Euler.

Vidéo YouTube : [youtube](#)

- [15] Master 2 IMOI, Université de Lorraine. *Méthodes numériques avancées pour la résolution des EDP, Volumes Finis.* J.-F. Scheid, Année 2017-2018.

- [16] OpenFOAM, *The Open Source CFD Toolbox Programmer's Guide*, Version v2412, 24 décembre 2024. Copyright © 2004-2011, 2016-2023 OpenCFD Limited.

- [17] R. I. Issa, "Solution of the implicitly discretised fluid flow equations by operator-splitting," *Journal of Computational Physics*, vol. 62, no. 1, pp. 40-65, 1986.

- [18] Y. Zhai, *CFD with OpenSource software : Implementing the pimpleFoam to oscillating flow solver porousOsciPimpleFoam using volume-averaged kOmega turbulence model*. Chalmers University of Technology, cours enseigné par H. Nilsson, Technical University of Denmark, 2014. Licence CC-BY-NC-SA.

- [19] M. Payandeh, *Implementation of a Temperature Dependent Viscosity Model in OpenFOAM*. Projet du cours "CFD with OpenSource software", Chalmers University of

Technology, enseigné par Håkan Nilsson, développé pour OpenFOAM-2.1.0, 2013. Peer reviewed by Johannes Palm et Jelena Andric.

- [20] *OpenFOAM® Introductory Training, Online session – 2020 Edition. This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0). To view a copy of this license, visit*
- [21] Quarteroni, A., Manzoni, A., & Negri, F. (2016). *Reduced Basis Methods for Partial Differential Equations : An Introduction*. Springer.
- [22] Hesthaven, J. S., Rozza, G., & Stamm, B. (2015). *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer Briefs in Mathematics.
- [23] EDEL, P. (2022). *Reduced basis method for parameter-dependent linear equations. Application to time-harmonic problems in electromagnetism and in aeroacoustics*. Thèse de doctorat, Sorbonne Université, dirigée par Yvon Maday. Soutenue le 24 octobre 2022.
- [24] Haasdonk, B. (2016). *Reduced Basis Methods for Parametrized PDEs – A Tutorial Introduction for Stationary and Instationary Problems*. Institute of Applied Analysis and Numerical Simulation, University of Stuttgart.
<http://www.ians.uni-stuttgart.de/agb>Introductory Training, Online session – 2020 Edition. Licence Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0).
- [25] Lumley, J. (1967). The structure of inhomogeneous turbulent flows. In A.M. Yaglom and V.I. Tatarsky (Eds.), *Atmospheric Turbulence and Radio Wave Propagation* (pp. 166–178).
- [26] Liberge, E. (2007). *Réduction de modèles par POD-Galerkin pour les problèmes d'interaction fluide-structure*. Thèse de doctorat, Université de La Rochelle, UFR Sciences Fondamentales et Sciences pour l'ingénieur, sous la direction de Aziz Hamdouni.
- [27] Goudon, T. (2017). *La “Décomposition en Valeurs Singulières” (SVD : Singular Value Decomposition)*. Notes de cours, Université Côte d'Azur.
- [28] Da Silva, F. (2015). *Méthodologies de réduction de modèles multiphysiques pour la conception et la commande d'une chaîne de traction électrique*. Thèse de doctorat, Université Paris-Saclay. Français. <https://theses.hal.science/tel-01275878>
<NNT : 2015SACL022>

- [29] Girault, M., Launay, J., Allanic, N., Mousseau, P., & Deterre, R. (2018). Development of a thermal Reduced Order Model with explicit dependence on viscosity for a generalized Newtonian fluid. *Journal of Non-Newtonian Fluid Mechanics*, 260, 26–39.
 doi:[10.1016/j.jnnfm.2018.04.002](https://doi.org/10.1016/j.jnnfm.2018.04.002)
- [30] Podvin, B. (2001). *Introduction à la Décomposition Orthogonale aux Valeurs Propres ou P.O.D.* LIMSI-CNRS UPR 3251, BP133 Université Paris-Sud, 91403 Orsay Cedex, Novembre 2001.
- [31] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. (1986). Learning Internal Representations by Error Propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing : Explorations in the Microstructure of Cognition*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge, MA.
- [32] Pichi, F., Moya, B., & Hesthaven, J. S. (2023). A Graph Convolutional Autoencoder Approach to Model Order Reduction for Parametrized PDEs. *Journal of Computational Physics*, 486, 112120. doi:[10.1016/j.jcp.2023.112120](https://doi.org/10.1016/j.jcp.2023.112120)
- [33] Fu, R., Xiao, D., Navon, I. M., & Wang, C. (2023). A data driven reduced order model of fluid flow by Auto-Encoder and self-attention deep learning methods. *Journal of Computational Physics*, 486, 112123. doi:[10.1016/j.jcp.2023.112123](https://doi.org/10.1016/j.jcp.2023.112123)
- [34] Emmanuel Franck, *Apprentissage et calcul scientifique : Auto-encoder et réduction de dimension* <https://irma.math.unistra.fr/~franck/cours/SciML/output/html/chapAPsec5.html>
- [35] Gosea, I. V., Gugercin, S., & Werner, S. W. R. (2023). Structured barycentric forms for interpolation-based data-driven reduced modeling of second-order systems. Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany ; Department of Mathematics and Division of Computational Modeling and Data Analytics, Virginia Tech, USA ; Courant Institute of Mathematical Sciences, New York University, USA.
- [36] Salmoiraghi, F., Scardigli, A., Telib, H., & Rozza, G. (2018). Free Form Deformation, mesh morphing and reduced order methods : enablers for efficient aerodynamic shape optimization. *To appear in the International Journal of Computational Fluid Dynamics*, Vol. 00, No. 00, 1–18. SISSA, International School for Advanced Studies, mathLab, Trieste, Italy ; OPTIMAD Engineering srl, Torino, Italy ; Politecnico di Torino, DISMA, Italy.

- [37] Ivagnes, A., Demo, N., & Rozza, G. (2020). A shape optimization pipeline for marine propellers by means of reduced order modeling techniques. SISSA, International School for Advanced Studies, Mathematics Area, mathLab, Trieste, Italy.
- [38] EZyRB, *EZyRB : Easy Reduced Basis method in Python*. mathLab, SISSA, Trieste, Italy.
<https://mathlab.github.io/EZyRB/>
- [39] J.-F. Scheid, *Approximations numériques des équations de Navier-Stokes incompressibles*. Cours de Mastère de Recherche M2 « double diplôme » en Mathématique et Applications, Ecole Supérieure des Sciences et de la Technologie de Hammam Sousse, Université de Sousse, IECL, Université de Lorraine, Janvier 2020.
- [40] Miranda, D. A., Rauber, W. K., Vaz Jr., M., & Zdanski, P. S. B. (2024). Numerical Analysis of Thermophysical Conditions of Polymers Melt Flow During Injection Mold Filling. *Department of Mechanical Engineering, University of the Region of Joinville, UNIVILLE, São Bento do Sul, Santa Catarina, Brazil ; Department of Mechanical Engineering, Graduate Program of Materials Science and Engineering, State University of Santa Catarina, UDESC, Joinville, Santa Catarina, Brazil*.
- [41] Behdani, B., Senter, M., Mason, L., Leu, M., & Park, J. (2020). Numerical Study on the Temperature-Dependent Viscosity Effect on the Strand Shape in Extrusion-Based Additive Manufacturing. *Journal/Conference Name*, Received : 8 April 2020 ; Accepted : 13 May 2020 ; Published : 15 May 2020.
- [42] Galvan, L. (2023). Numerical simulations of two-phase non-Newtonian flows with OpenFOAM. Tesi di Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica, Politecnico di Milano. Student ID : 977175. Advisor : Prof. Nicola Parolini, Co-advisor : Emilia Capuano. Academic Year : 2022-23.
- [43] Yuan, J., Liu, Y., Wang, J., Qu, Y., Sun, H., Qin, Y., & Wang, N. (2023). Non-Isothermal Simulation and Safety Analysis of Twin-Screw Extrusion Process for Synthesizing Glycidyl Azide Polymer-Based Energetic Thermoplastic Elastomer. *Journal Name, Volume(Issue), pages*.
- [44] Reyes, R., Ruz, O., Bayona-Roa, C., Castillo, E., & Tello, A. (2023). Reduced order modeling for parametrized generalized Newtonian fluid flows. *Chair of Computational Mathematics and Simulation Science, École Polytechnique Fédérale de Lausanne (EPFL)*,

1015 Lausanne, Switzerland ; Universidad de Santiago de Chile, Departamento de Ingeniería Mecánica ; Departamento de Ingeniería Industrial, Facultad de Ingeniería, Pontificia Universidad Javeriana, Bogotá, Colombia ; Centro de Ingeniería Avanzada, Investigación y Desarrollo (CIAID), Bogotá, Colombia.