Nouvelles méthodes numériques pour la simulation de l'impression 3D métallique

Asmaâ Agouzoul

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR DE

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DES SCIENCES PHYSIQUES ET DE L'INGÉNIEUR

Spécialité: *mécanique*

Par **Asmaâ AGOUZOUL**

Nouvelles méthodes numériques pour la simulation de l'impression 3D métallique.

Sous la direction de Pierre JOYOT

Soutenue publiquement le *09 Janvier 2020*

Membres du Jury:

M.	Éric LACOSTE	Professeur, I2M-Université de Bordeaux	Président
M.	Francisco CHINESTA	Professeur, Arts et Métiers ParisTech, Paris	Rapporteur
M.	Pierre VILLON	Professeur des universités UTC, Compiègne	Rapporteur
Mme.	Marianne BERINGHIER	Maître de conférences, ISAE-ENSMA, Poitiers	Examinateur
M.	Patrick REUTER	Maître de conférences, HDR, LABRI-Université de Bordeaux	Examinateur
M.	Fabien POULHAON	Docteur, ESTIA-RECHERCHE, Bidart	Invité
M.	Pierre JOYOT	Enseignant-Chercheur, HDR, ESTIA-RECHERCHE, Bidart	Directeur

Résumé — Le procédé SLM offre de nouvelles perspectives en termes de conception de pièces. Cependant, les phénomènes thermo-mécaniques liés au procédé sont responsables des contraintes résiduelles et de la distorsion de la pièce fabriquée. La simulation numérique est un outil intéressant pour mieux cerner les phénomènes physiques à l'œuvre et leur impact sur la qualité de la pièce. Dans cette thèse, nous proposons différentes approches qui permettent de réaliser les simulations à moindre coût, en utilisant des algorithmes de réduction de modèles. Les résultats sont comparés à ceux obtenus par la méthode des éléments finis. Une méthode inverse d'identification rapide de la contrainte inhérente à partir d'abaques numériques est proposée. L'approche Proper Generalized Decomposition (PGD) est utilisée pour la construction de cet abaque. Nous explorons aussi les avantages qu'offre une implémentation de la PGD sur GPU.

Mots clés : Fabrication additive, Simulation numérique, Modèle réduit paramétrique, SLM, modèle inverse, Méthode des déformations inhérentes, PGD, POD, APR, HPC.

Abstract — Selective Laser Melting offers new perspectives in terms of part design and simplification of complex assemblies. However, severe thermo-mechanical conditions arise and are responsible for local plastic deformation, residual stresses and distortion of the manufactured component. Numerical simulation is an interesting tool for process understanding the physical phenomena and their impact on the quality of the part. In this thesis, we propose different approaches to perform simulations at a lower cost, by using model reduction algorithms. The results are compared with those obtained by the finite element method. A reverse analysis in order to identify the inherent strain responsible for the measured elastic springback makes possible to build offline numerical abacus. Therefore, we use a multi-parametric reduced order model using the so called Proper Generalized Decomposition (PGD) to construct this abacus. We also explore the benefits of an implementation of PGD on GPU.

Keywords : Additive manufacturing, Digital simulation, Parametric and reduced model, SLM, Reverse Identification, Inherent Strain Methode, PGD, POD, APR, HPC.

École Supérieure des Technologies Industrielles Avancées ESTIA-Recherche
Technopôle Izarbel, 64210 Bidart
Institut de mécanique et d'ingénierie - I2M CNRS UMR 5295
Université de Bordeaux - Campus Talence

Remerciements

Aucun remerciement ne sera témoin de mon immense gratitude à l'égard de mon directeur de thèse Pierre Joyot pour sa disponibilité, son encadrement, ses cruciaux conseils, ainsi que sa générosité en matière de formation et d'encadrement tout au long de ces trois années.

Je tiens aussi à remercier Fabien Poulhaon pour le suivi de la partie mécanique, et l'éclairage apporté. J'adresse aussi mes remerciements à l'ensemble des membres d'ESTIA.

Enfin, mes remerciements les plus chaleureux vont à mes parents pour leur amour inestimable, leur soutien et leurs encouragements, à mon frère et à ma sœur pour leur complicité, et leur présence malgré la distance qui nous sépare.

Table des matières

Table des figures

Liste des tableaux

Table des sigles et acronymes

FA *Fabrication additive.*

FEM *Méthode de résolution par éléments finis.*

SVD *Décomposition en valeurs singulières.*

PGD *Décomposition généralisée en modes propres.*

POD *Décomposition orthogonale en modes propres.*

RB *Bases réduites.*

ROM *Modèle d'ordre réduit.*

Block *C'est un regroupement de threads. Les threads d'un même block partagent une mémoire commune très rapide.*

Cluster *est un système informatique composé d'unités de calcul et de ressources (micro-processeurs, cœurs, unités centrales) indépendantes qui sont reliées entre elles à l'aide d'un réseau de communication et fonctionnant comme un seul système.*

CPU *voir Processeur.*

CUDA *C'est une technologie de type GP-GPU qui permet de programmer en C et en Fortran. Elle a été développée par NVIDIA pour exécuter les calculs sur les cartes graphiques, et utilise un pilote unifié utilisant une technique de streaming (flux continu).*

Cycle *Ou cycle d'horloge. C'est l'unité élémentaire de temps d'un ordinateur. Le nombre de cycles d'horloge d'un processeur est lié à sa fréquence. Un cycle d'horloge correspond à un battement du microprocesseur. Ainsi, un processeur cadencé à 300 MHz possède 300 millions de cycles d'horloge par seconde. Chaque instruction nécessite au moins un cycle d'horloge pour s'exécuter.*

GPU *Graphics Processing Unit est un microprocesseur présent sur les cartes graphiques au sein d'un ordinateur ou d'une console de jeux vidéo. Le processeur graphique se charge des opérations d'affichage et de manipulation de données graphiques. Les processeurs des cartes graphiques modernes (en 2009) ont une structure hautement parallèle (voir accélération matérielle) qui les rend efficaces pour une large palette de tâches graphiques.*

GP-GPU *General-Purpose Processing on Graphics Processing Units, calcul générique sur un processeur graphique.*

Latence *Aussi appelée lag, désigne le délai entre le moment où une information est envoyée et celui où elle est reçue. De façon plus générale, la latence peut aussi désigner l'intervalle entre la fin d'un événement et le début de la réaction à celui-ci.*

Grid *C'est l'ensemble des blocks de la carte graphique. En fait, du point de vu des processus parallèles, le grid, c'est la carte graphique. Il contient les blocks, qui contiennent les threads.*

Processeur *Ou CPU (de l'anglais Central Processing Unit, "Unité centrale de traitement"), est le composant de l'ordinateur qui exécute les programmes informatiques. Avec la mémoire notamment, c'est un des composants qui existent depuis les premiers ordinateurs et qui est présent dans tous les ordinateurs.*

Processus *En informatique, il est défini par : un ensemble d'instructions à exécuter (un programme) ; un espace mémoire pour les données de travail ; éventuellement, d'autres ressources, comme des descripteurs de fichier, des ports réseau, etc*

Thread *C'est la plus petite unité de traitement que l'on peut lancer sur la carte graphique. C'est elle qui exécute les programmes en parallèle.*

Warp *Un warp est un ensemble de 32 threads du GPU, envoyés ensemble à l'exécution, et exécutés simultanément sur deux cycles.*

Introduction

Au début des années 1980, différentes technologies de fabrication additive (FA) ont été au cœur de l'actualité avec le développement de la stéréolithographie permettant de fabriquer une pièce par ajout de matière. Aujourd'hui, la concurrence entre les industriels les pousse à adopter le procédé de FA le plus adapté au secteur d'activité de l'entreprise, afin de réduire considérablement les temps de fabrication d'une pièce et son impact environnemental tout en réalisant des géométries internes et externes complexes.

De nombreux secteurs d'activités s'intéressent aux procédés de FA. C'est le cas notamment en biomédical pour la réalisation de prothèses et d'implants, en architecture, en ameublement, joaillerie, automobile, aéronautique ...

Les entreprises du secteur aéronautique et spatial ont été parmi les premières à adopter la FA. L'enjeu majeur est de bénéficier des allégements de masse permis par l'utilisation des formes complexes, de réaliser sans assemblages (soudage, boulonnage...) des pièces finales.

Dans le cadre du projet TRANSFRON 3D, cette thèse est réalisée en partie au sein de la plateforme AddimAdour, qui est un centre de recherche et développement technologique dédiée à la fabrication de pièces métalliques de grande taille, et accompagnant les entreprises dans leurs projets et ce, à différents niveaux. Parmi les atouts de la plateforme on trouve le prototypage de pièces WAAM, l'utilisation du dépôt de poudre (LMD-P) grâce à la machine BeAM dont dispose la plateforme ainsi, que le développement du procédé LMD fil par chauffe laser dans le cadre du FUI ADDIMAFIL.

La fabrication additive métallique, met en jeu des phénomènes physiques complexes de nature essentiellement thermique, mécanique et métallurgique. On souhaite maîtriser ces propriétés afin de concevoir une pièce industrielle fonctionnelle. Ces phénomènes atteignent une complexité telle qu'une simulation numérique est indispensable à leur compréhension. Néanmoins, la complexité des phénomènes physiques ne permet pas de simuler une pièce industrielle complète dans un temps raisonnable. Il devient donc nécessaire de développer des méthodes de calculs plus légères soit en simplifiant le modèle soit en utilisant des approches par réduction de modèles.

Notre objectif n'est pas de proposer un nouveau modèle thermo-mécanique multi-échelle pour la simulation des procédés FA. Il existe une abondante littérature sur le sujet ainsi que des outils industriels pour la simulation (Sysweld, Virfac, Abaqus...). Le but n'est pas non plus de proposer des stratégies de réduction de modèles directement sur ces modèles thermo-mécaniques finis. Ce qui nous semble hors de portée. Par contre, de nombreux travaux ont montré l'intérêt d'une approche simplifiée comme la méthode des déformations inhérentes (*Inherent Strain Method (ISM)*) en mécanique ou l'approche des macros-couches en thermique. Notre objectif est de reprendre ces travaux et d'étudier comment les approches réduction de modèles peuvent les enrichir.

Le procédé SLM sera pris comme exemple car il semble plus simple et les approches proposées peuvent être étendues à des procédés comme le dépôt de poudre (LMD-P) ou de fil (LMD-W, WAAM).

La présente thèse est divisée en quatre parties :

La première partie brosse dans un premier temps un état de l'art de la FA, nous y présentons les différents procédés, ainsi qu'une présentation du projet TRANSFRON3D. Ensuite, un état de l'art des phénomènes physiques est donné et enfin, une vue d'ensemble sur les différents algorithmes de réduction de modèles utilisés dans la suite de ce manuscrit.

La deuxième partie traite de la problématique de la thermique d'un domaine multi-couche inspiré du procédé SLM. Dans un premier temps, on étudie comment le champ thermique résultant peut être réduit par une approche POD, on montre qu'une stratégie de changement de variable suivant la direction de construction est indispensable pour réduire efficacement le nombre de nœud. Nous réutilisons ce résultat pour résoudre ce même problème par APR puis par PGD. Dans cette partie, nous nous limitons au cas 1D et nous remarquons que le couplage entre l'espace et le temps ne se traduit pas par une explosion du nombre des opérateurs PGD, et ce, grâce au changement de variable proposé. Par contre, dans le cas 3D cette explosion ne peut être évitée ce qui nous amène à proposer une implémentation de la PGD sur une architecture GPU. Ceci fait l'objet de la troisième partie.

La troisième partie s'intéresse au problème de la distorsion de la pièce due à l'accumulation des contraintes résiduelles au cours du procédé. Nous reprenons l'approche ISM proposée par Ueda [103] pour le soudage, ainsi que les modèles analytiques adaptés à l'empilement de couches. Ce travail, nous permet de mieux comprendre les équations d'équilibre intégrant à la fois les déformations inhérentes et l'empilement de couches. Ces équations ne sont jamais explicitées dans la littérature. Ainsi, nous avons pu développer notre propre code que nous avons validé par rapport à l'outil industriel Virfac. Ensuite, nous proposons une résolution PGD multiparamétrique de ces équations, où la contrainte inhérente est un paramètre. Par conséquent, nous pouvons réaliser de manière rapide la phase de calibration des déformations inhérentes indispensable au calcul d'une pièce réelle.

Tout au long de cette thèse le rôle de l'informatique est primordial, afin de mettre en place les méthodes numériques les mieux adaptées pour la simulation des différents phénomènes physiques. Le choix d'une méthode nécessite une analyse particulière afin d'exploiter toutes les ressources informatiques mises à disposition. Par conséquent, la quatrième partie de ce manuscrit s'intéresse au développement et à la programmation de la PGD sur GPU. On s'intéressera dans un premier temps, à présenter les différents ingrédients nécessaires pour la parallélisation de la librairie. Ensuite, l'implémentation de la PGD sur CPU sera détaillée dans un premier temps, afin d'analyser le besoin, de déterminer les différentes pathologies, et repérer les fonctions chronophages. Pour des raisons de mémoire sur notre carte graphique, la librairie implémentée en Cuda C est ensuite découplée afin de réaliser des benchmarks sur les différents kernels les plus utilisés. On se focalisera dans cette partie principalement sur l'explosion du nombre d'opérateurs de la PGD, pour ce, nous développerons un kernel qui permet de calculer différents opérateurs dans un laps de temps courts. Ainsi, que la

résolution du système linéaire qui peut aussi devenir chronophage en fonction du problème à résoudre. L'objectif est donc d'implémenter et de mettre en place dans un premier temps la PGD développée en C, CUDA C et ensuite, de réaliser différents benchmarks des différentes méthodes numériques pour déterminer le choix le mieux adapté à notre application PGD. Et enfin, de développer les bonnes pratiques pour pérenniser et maintenir les codes, il reste toutefois utile d'adapter le programme à chaque architecture.

Enfin, nous terminerons ce manuscrit par un chapitre de conclusion donnant lieu à l'état de nos recherches effectuées ainsi qu'à une proposition des perspectives pour des recherches futures.

Première partie

Étude bibliographique

Chapitre 1

La Fabrication Additive

Ce chapitre est dédié à la présentation du contexte général de cette thèse. Après avoir introduit les différents procédés de FA, une présentation du projet est réalisée. Une attention particulière est ensuite portée sur l'intérêt de la simulation pour la compréhension des phénomènes agissant au cours de la fabrication.

Sommaire

1.1 Préambule

Avec la révolution industrielle et face à une situation économique complexe, les entreprises sont aujourd'hui appelées à faire face à un marché concurrentiel. Pour se différencier elles doivent recourir à de nouvelles techniques de fabrication plus personnalisées, plus performantes et nécessitant des délais plus courts avec un faible impact énergétique... La première technique de FA a été découverte simultanément par Jean-Claude André [1] et Chuck Hull [2] en 1984.

La FA est une technologie qui permet la fabrication d'une pièce directement par empilements successifs de couches de matière à partir d'un fichier numérique de conception, la matière est donc distribuée selon la géométrie de la pièce, ce qui permet d'entrevoir les avantages suivants :

L'optimisation topologique [96, 7, 6], est une technique qui permet d'optimiser la forme de la pièce en négligeant la matière là où les efforts ne transitent pas ce qui permet de diminuer leur masse (l'exemple du treillis), il devient aussi possible d'ajouter des fonctionnalités aux pièces [23]. Cette technique permet donc une grande liberté de conception ainsi qu'un important gain de matière [74].

Cependant, ce procédé présente aussi des inconvénients, notamment le coût élevé des machines et des matériaux. La taille des pièces reste limitée, ainsi que le faible recul technologique. En effet, durant plusieurs années la FA était principalement dédiée au prototypage rapide ; permettant la réalisation de maquettes pour les bureaux d'études pour les tester dans le but de raccourcir le temps de conception et de développement du produit [83]. Aujourd'hui, différents secteurs industriels illustrés par la figure 1.1 comprennent l'intérêt de ce procédé qui devient un outil de fabrication à part entière permettant ainsi la conception de pièces sur-mesure et de bonne qualité. Avec l'aérospatiale, l'automobile et le médical qui sont les secteurs précurseurs pour l'utilisation de la FA, on retrouve l'architecture, la bijouterie, le

1. chercheur à l'école nationale supérieure des industries chimiques de Nancy qui travaillait pour le compte de la société Cilas-Alcatel.
2. fondateur en 1986 de la première entreprise de procédé de stéréolithographie (3D systems).

militaire en pleine croissance. Ces secteurs exploitent généralement les procédés métalliques pour la production en petite série de composants afin de les personnaliser en s'affranchissant de la conception traditionnelle.

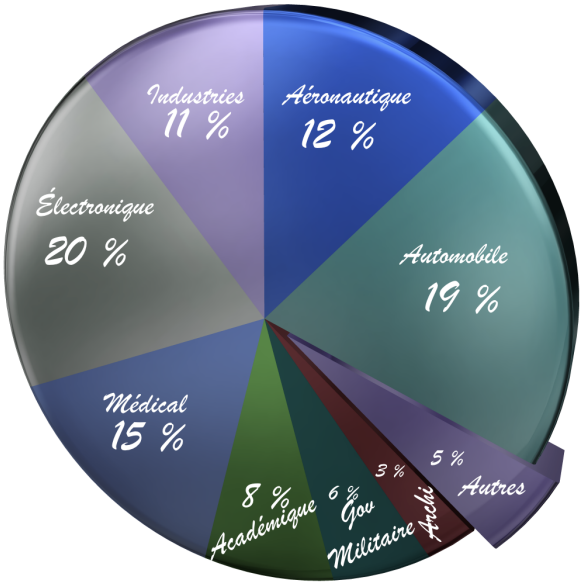


FIGURE 1.1 – Répartition relative des secteurs d'activité utilisant la FA [61, 106].

1.2 Contexte

1.2.1 Enjeux économiques

Suite à la grande récession, les acteurs ont dû repenser l'industrie manufacturière qui représente 16% de l'économie mondiale [1]. En offrant une alternative aux outils industriels soustractifs, les procédés additifs ouvrent la voie à de nouvelles marges de manœuvres. Les facteurs susceptibles de créer un avantage sont principalement les délais d'approvisionnement et le nombre d'étapes d'assemblage qui sont réduits, et des coûts réduits.

Le cabinet d'analyses américain Wohlers Associates [113] qui s'intéresse aux aspects économiques et environnementaux, a publié un état de marché basé sur la contribution d'une trentaine de pays. L'étude menée a pu souligner la croissance des matériaux d'impression 3D en mettant l'accent sur le développement des connaissances et compétences liées à la FA (connu en industrie sous la normalisation ASTM F2792 [3]). Le Wohlers Report 2019 [113] rassemble des informations d'utilisateurs finaux : Deutsche Bahn, Oerlikon, UPS, BMW, Jabil, FIT... Cette enquête enregistre la croissance de près d'un tiers que les ventes rapportées par les fabricants l'année dernière. Inversement, cette étude met l'accent sur la baisse significative des imprimantes 3D de bureau.

3. ASTM International Standard Terminology for Additive Manufacturing Technologies.

1.2.2 Impact environnemental

Sous un autre aspect, la rupture avec la fabrication soustractive intéresse particulièrement les éco-concepteurs. En effet, la réutilisation d'une grande partie de la poudre non fusionnée tout en fabriquant des pièces à la fois complexes et moins massives, ne nécessitant pas un long temps de post-traitement permet de réduire le gaspillage sur les phases de production. B. Vayre [106] précise l'impact non négligeable de la fabrication devant la phase d'extraction et de création de la matière première. D'autre part, il est à noter l'impact énergétique généré par l'allègement des pièces, ainsi que la flexibilité de conception ce qui permet de réduire les coûts de transports potentiellement sur les émissions de CO_2 et de stockage.

F. Le Bourhis [61] présente une analyse de l'impact environnemental d'une pièce réalisée par FA au cours de son cycle de vie. L'auteur développe une méthodologie qui permet d'évaluer l'impact environnemental du couple pièce-procédé au cours de la FA.

Par ailleurs, les différents rapports bibliographiques existants mettent en évidence un bilan énergétique plutôt positif, il existe toutefois des impacts négatifs comme la production de poudre qui est consommatrice d'énergie [4].

1.3 Procédés de fabrication

Les procédés de FA ont en commun le même processus de fabrication. Un fichier numérique est nécessaire et peut s'obtenir via un logiciel de CAO comme SolidWorks.... Ce fichier, souvent au format STL, est transmis à la machine, où un second logiciel (appelé slicer) réalise une découpe du modèle en fines couches d'épaisseur fixe et génère la trajectoire du laser pour chaque couche [5]. Ces instructions sont alors envoyées à l'imprimante dans laquelle seule la matière à imprimer est fondue. Selon le procédé la fusion est réalisée par un laser Nd :YAG (*Neodymium-Doped Yttrium Aluminium Garnet*), ou UV (*Ultraviolet*) dans le cas de la stéréolithographie. Dans ce qui suit, nous présentons les 7 grandes familles d'impression 3D.

1.3.1 La stéréolithographie (SLA)

La SLA (*StereoLithography Apparatus*) est considérée comme la première technique d'impression 3D brevetée en 1984, c'est une technologie basée sur la photo-polymérisation de résines photopolymères liquides. À chaque descente du plateau d'une valeur égale à l'épaisseur de la nouvelle couche, la lumière ultra-violet (fixé au niveau du bloc optique comme représenté dans la figure 1.2) balaye la résine liquide photosensible. Les propriétés de la lumière permettent à la couche de polymériser et de se solidifier [105]. Dès que la première couche est solidifiée, la plateforme mobile est replongée dans la résine à une hauteur qui cor-

4. http://www.cci-paris-idf.fr/sites/default/files/etudes/pdf/documents/impression-3d-etude-1509.pdf
5. 3darcwest.fr/component/k2/item/19-sla-stereolithographie-resines.html

respond à l'épaisseur de la couche. Le processus de construction se poursuit en suivant les instructions transmises par la CAO jusqu'à l'obtention du volume complet.

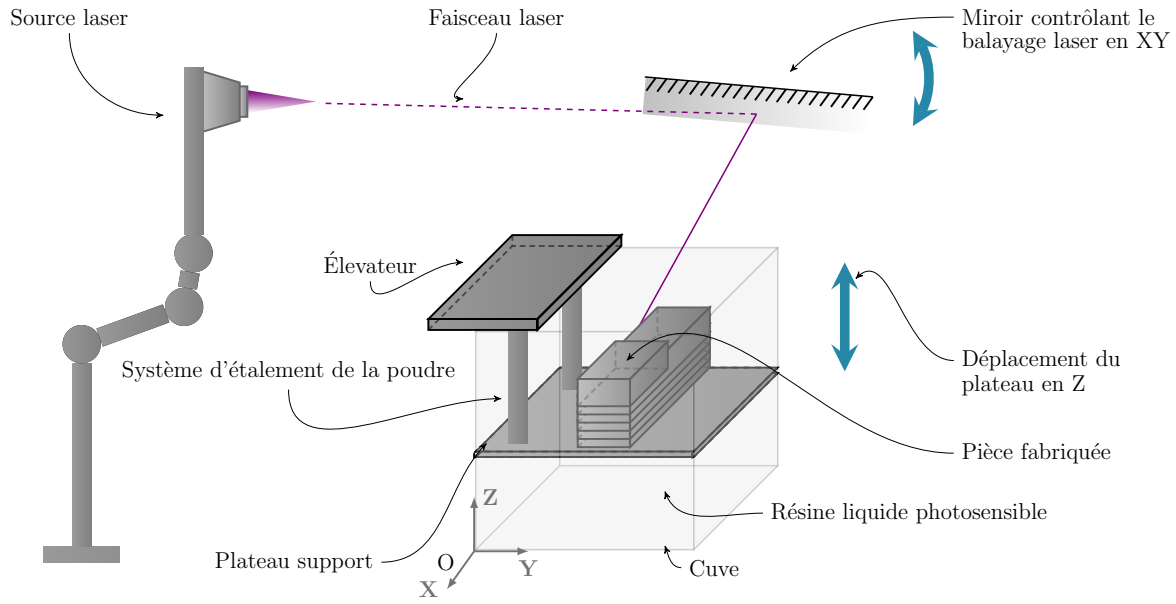


FIGURE 1.2 – Principe du procédé SLA

Ce procédé permet d'obtenir des pièces complexes avec de très bonnes finitions, néanmoins, il nécessite un long temps de post-traitement. En effet, les pièces imprimées doivent être nettoyées afin d'enlever le support nécessaire lors de la fabrication et supprimer le résidu de résine non polymérisé. La polymérisation doit ensuite être terminée dans un four.

1.3.2 Le dépôt de fil (FDM)

Le dépôt de fil (*FDM : Fused Deposition Modeling*) a été initialement développé par Scott Crump à la fin des années 80. Son fonctionnement consiste à faire fondre un filament thermoplastique de l'ordre du dixième de *mm* et à le déposer à l'aide d'une buse chauffée à haute température (cf. figure 1.3). Ce processus se répète jusqu'à l'obtention de la pièce complète.

L'intérêt de cette technologie réside dans la facilité du parachèvement dû à la possibilité de créer le support avec un matériau soluble. Elle offre également une certaine stabilité d'un point de vue mécanique et environnementale. En effet, plusieurs matériaux sont utilisés comme le PLA (*Polylactic Acid*) qui est biodégradable et non toxique. Cependant, c'est un procédé lent qui nécessite un traitement supplémentaire (comme le sablage, le polissage) selon les besoins de l'état de surfaces.

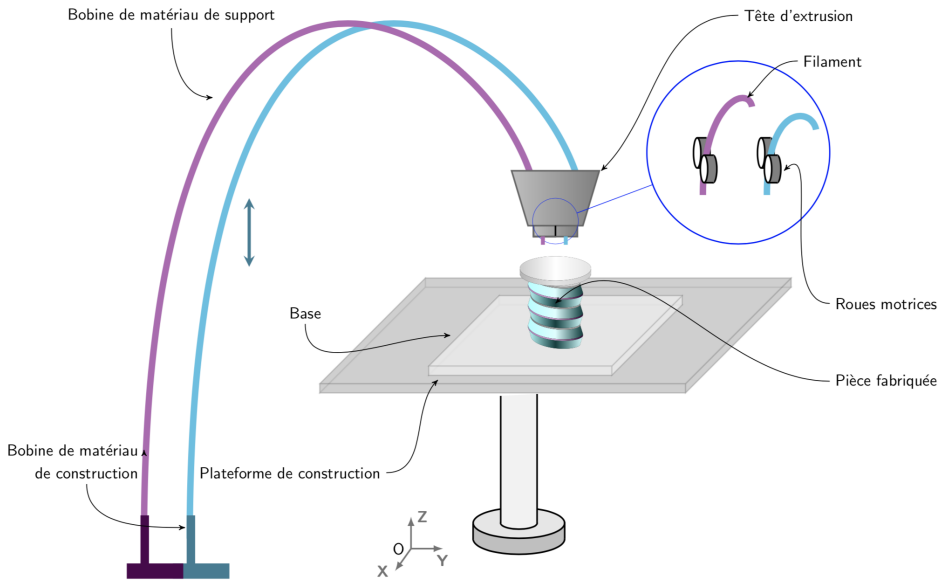


FIGURE 1.3 – Principe du procédé FDM

1.3.3 Dépôt direct de matériaux métalliques

Apparu en 1993, le procédé de dépôt de matériaux dispose de nombreuses techniques les plus connues sont le LENS(*Laser Engineered Net Shaping*), DMD (*Direct Metal Deposition*) et le CLAD (*Laser cladding*). Les matériaux utilisés étant métalliques, ce procédé permet la réalisation de pièces de grande taille allant jusqu'à $1.50m$, ainsi que la réparation et le prolongement de la durée de vie d'une pièce usée. En effet, ce procédé permet d'ajouter une quantité déterminée de métal afin de réparer une pièce endommagée.

Pour réaliser une pièce entière en utilisant ce procédé, la poudre métallique (généralement l'acier inoxydable 316L, l'inconel ou le Ti6Al4V) est directement propulsée dans le faisceau du laser. La poudre métallique est alors fondue par le laser. Le métal en fusion est alors appliqué directement sur le support couche par couche pour réaliser la pièce. Néanmoins, des supports peuvent être nécessaires afin de maintenir l'objet en cours de fabrication mais presque toute la poudre est alors solidifiée. L'épaisseur du dépôt peut varier entre 0.6mm et 2.4mm alors que l'épaisseur de couche est comprise entre 0.2mm et 0.8mm. Un gaz inerte (Argon, Hélium, Azote) est utilisé afin de limiter l'oxydation [105].

Dans le cadre d'un dépôt de fil métallique WAAM (*Wire and Arc Additive Manufacturing*), un arc électrique contrôlé par un bras robotisé fait fondre directement un fil métallique sur la surface. Ce procédé permet la fabrication rapide de grandes pièces, néanmoins, ces pièces nécessitent un post-traitement. Des études sont en cours principalement à l'échelle de la micro-couche afin de contrôler ce procédé et maîtriser l'état de la pièce.

Campatelli et al. [22] proposent une méthodologie tenant compte de l'ensemble du cycle de vie d'une lame en acier fabriquée en utilisant le procédé WAAM. À travers des études expérimentales, les auteurs mettent l'accent sur d'importantes économies d'énergie mesurées

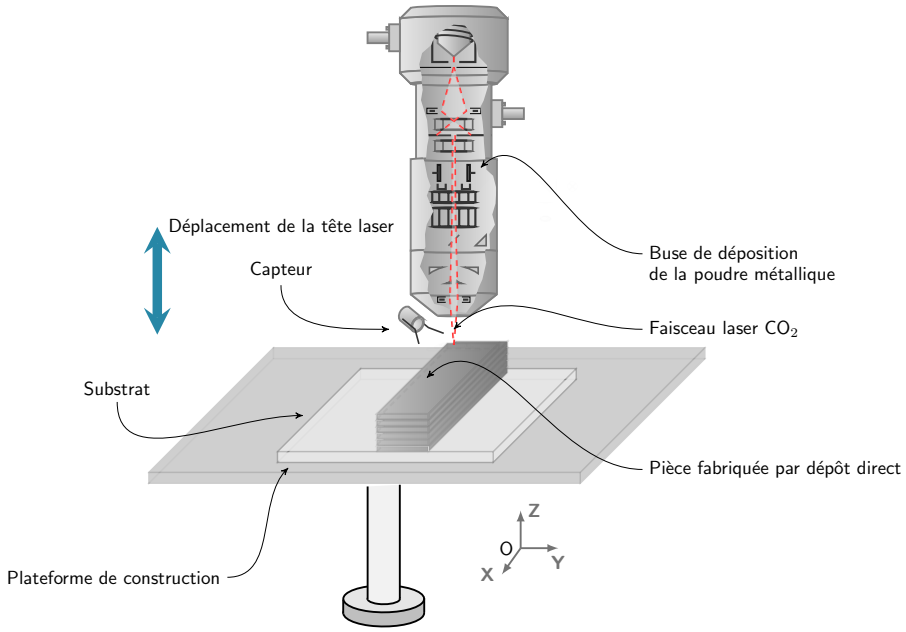


FIGURE 1.4 – Principe du procédé DMD

par rapport à la solution d'usinage. C. Priarone et al. [91] définissent le WAAM comme étant un procédé de dépôt d'énergie directe, économique et pratique, basé sur le soudage. Cet article propose une évaluation en acier doux structurel ER70S-6, en considérant les délais de fabrication, les coûts, l'empreinte carbone et le temps de post-traitement. Une autre variante est présentée par Guo et al. dans [41] afin de pallier le problème de l'apport excessif de chaleur. Dans ce travail, les auteurs étudient le comportement des arcs et le transfert des gouttelettes.

1.3.4 La fusion sur lit de poudre

Le procédé de lit de poudre dispose de trois techniques :

— Le **frittage sélectif par laser** (SLS : *Selective Laser Sintering*) : Cette technique consiste à chauffer de fines particules de poudre (préalablement déposées par couches avec un rouleau sur un plateau pour former le lit de poudre cf. figure 1.6) avec un laser en suivant les instructions transmises par le logiciel CAO, afin de réaliser une section de la pièce souhaitée.

Une fois la première couche solidifiée, le plateau descend d'une hauteur correspondant à l'épaisseur d'une couche, afin de permettre l'étalement de la couche de poudre sur la partie supérieure. Puis le processus précédent est répété jusqu'à l'obtention de la pièce finale.

Ce procédé se distingue également des procédés sus-mentionnés par la possibilité de construire des modèles aux géométries complexes et imbriquées.

— La **fusion par faisceau d'électron** (EBM : *Electron Beam Melting*) : procédé similaire au procédé SLS, mais à ceci près qu'elle utilise un faisceau d'électrons au lieu du

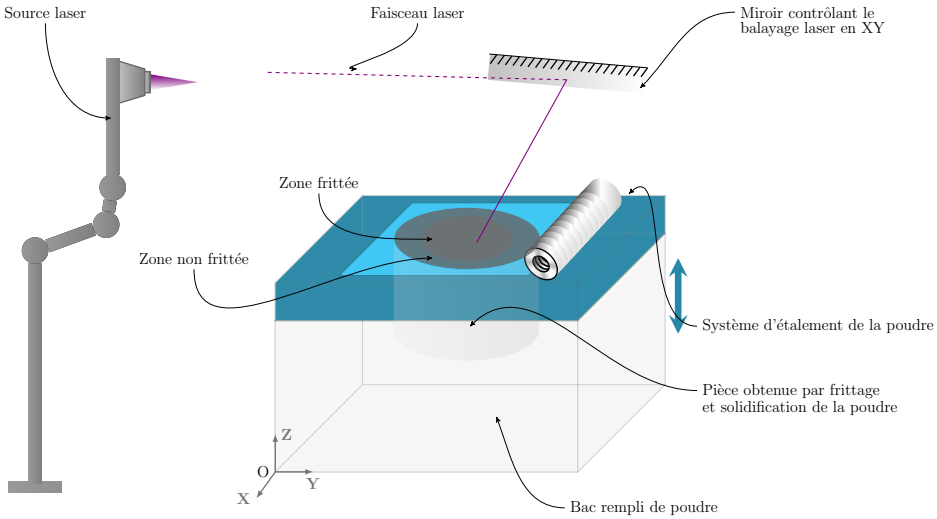


FIGURE 1.5 – Principe du procédé SLS

laser pour fondre couche par couche la poudre métallique.

Cette technologie chauffe la poudre de métal à des températures pouvant atteindre jusqu'à $1000°C$ avec un taux de vide élevé pour limiter l'oxydation. Ce procédé permet la production de pièces complexes.

— La **fusion laser sélective sur lit de poudres métalliques** (SLM : *Selective Laser Melting*) : basée sur le même principe que le procédé SLS, à une différence près le faisceau laser est de plus forte intensité. Ce procédé consiste à fusionner de manière sélective dans un réservoir sous gaz neutre les particules d'un lit de poudre à l'aide d'un faisceau laser qui balaie la surface et fait fondre la poudre métallique, formant ainsi un bain liquide sur une section déterminée par le fichier STL [6]. La géométrie de la pièce est réalisée par balayage du faisceau laser X et en Y et par le déplacement en Z du plateau. Les dimensions de ce plateau limitent en largeur et profondeur la taille maximale de la pièce à produire.

Après solidification de chaque couche, la plateforme est abaissée pour pouvoir étaler à l'aide du rouleau (figure 1.6), une nouvelle couche de poudre d'épaisseur prédéterminée. Ainsi, la construction de la pièce se fait selon une trajectoire dite de "balayage". Le laser balaie à nouveau la surface de la poudre avec une vitesse de lasage prédéterminée, ensuite un temps de refroidissement est introduit afin d'éviter un écroulement des bords de la nouvelle couche.

Au cours de ce procédé, différents paramètres doivent être pris en compte dans le but d'améliorer la précision de la pièce finale : la puissance du laser, le *spot size* (ie : le diamètre du faisceau laser), la vitesse du laser, le temps de refroidissement, l'épaisseur de la couche. Ce dernier est un paramètre très influent permettant ainsi de déterminer le temps de fabrication et l'état de surface obtenu avant finition. «Cette épaisseur de couche est à choisir lors du découpage du fichier STL. Plus l'épaisseur de couche est

6. https://www.techniques-ingenieur.fr/base-documentaire/electronique-photonique-th13/applications-des-lasers-et-instrumentation-laser-42661210/fusion-laser-selective-de-lit-de-poudres-metalliques-bm7900/principe-general-de-la-fusion-laser-selective-de-lit-de-poudres-metalliques-bm7900niv10001.html

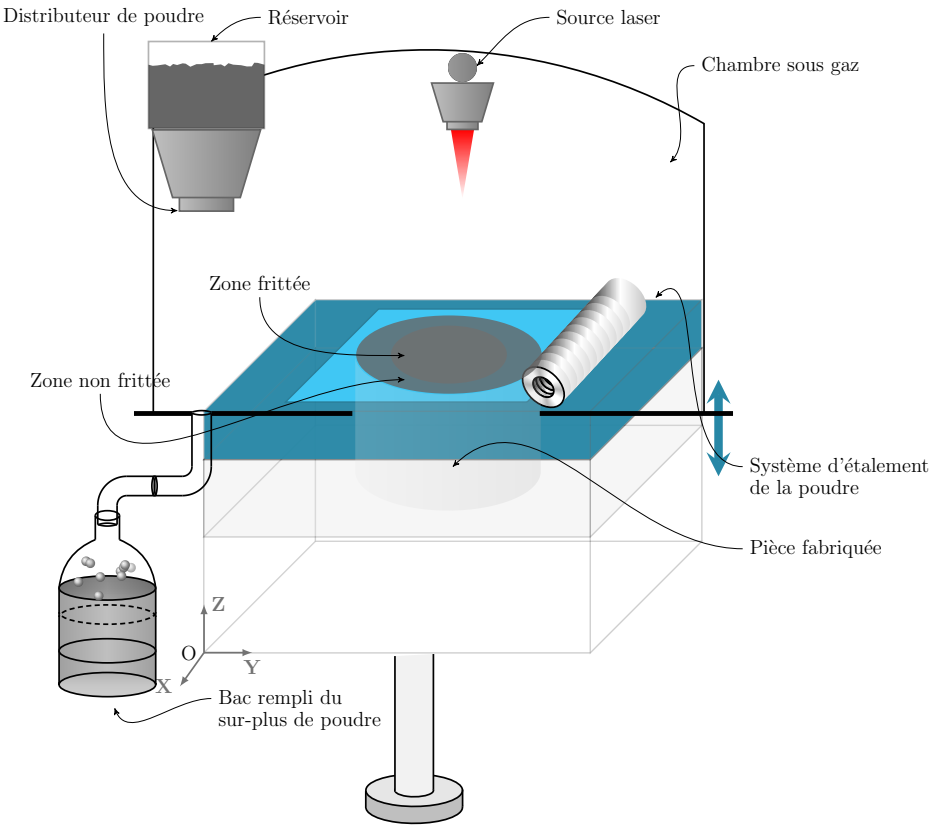


FIGURE 1.6 – Principe du procédé SLM

importante, plus la fabrication est rapide» [7].

Le SLM, a pour avantage de permettre la fabrication de pièces rapidement, tout en permettant de récupérer la poudre non fondue. Toutefois, ce processus nécessite beaucoup d'énergie (absorptivité de la poudre faible sur certains alliages). Les applications les plus fréquentes pour cette technologie sont dans l'industrie aérospatiale. C'est ce procédé qui est étudié dans la suite de ce travail.

1.4 Matériau considéré

Le SLM utilise de la poudre métallique comme matériau d'apport. Ces poudres doivent répondre à certaines caractéristiques à savoir une fine granulométrie et une distribution de type gaussienne. La morphologie des particules doit être sphérique afin d'assurer une bonne coulabilité.

Le $Ti6Al4V$ est un alliage largement étudié dans la littérature. C'est en effet, l'alliage de titane alpha-bêta le plus utilisé dans des applications de pointe comme l'aéronautique et le spatial. Cet alliage présente une excellente résistance à la corrosion, une température de fusion

7. https://www.cetim-certec.com/fabrication-additive-metallique/

élevée, ainsi qu'une bonne résistance en fatigue [31]. Cette poudre est de forme sphérique, avec une répartition granulométrique qui se situe entre $44\mu m$ et $82\mu m$, avec une taille moyenne de $61\mu m$.

À température ambiante, le $Ti6Al4V$ a une structure biphasée $\alpha+\beta$. La phase α est alors prédominante. Au cours d'un chauffage, la phase α va progressivement se transformer en phase β. Au-dessus de la température de transus β, une structure monophasée β est obtenue. Différentes transformations apparaissent au niveau de la microstructure lors du refroidissement [98].

R. Julien [53] aborde le comportement mécanique et l'évolution de la microstructure de l'alliage $Ti6Al4V$ en utilisant différentes condition mécaniques et thermiques. L'auteur propose une quantification des phases α et β lors de l'étape de refroidissement. Dans [68], les auteurs proposent une étude comparative des pièces $Ti6Al4V$ fabriquées en FA et celles conçues en utilisant la fabrication traditionnelle. Cette étude permet principalement d'examiner les propriétés mécaniques à savoir la traction, la fatigue, ou encore la température. Les auteurs abordent aussi l'influence des défauts sur les performances mécaniques de la pièce finale.

1.5 Projet TRANSFRON3D

Ce travail de recherche a été réalisé grâce au financement du projet POCTEFA TRANSFRON3D [8] qui a pour but le renforcement des collaborations économiques de l'espace frontalier Espagne-France-Andorre.

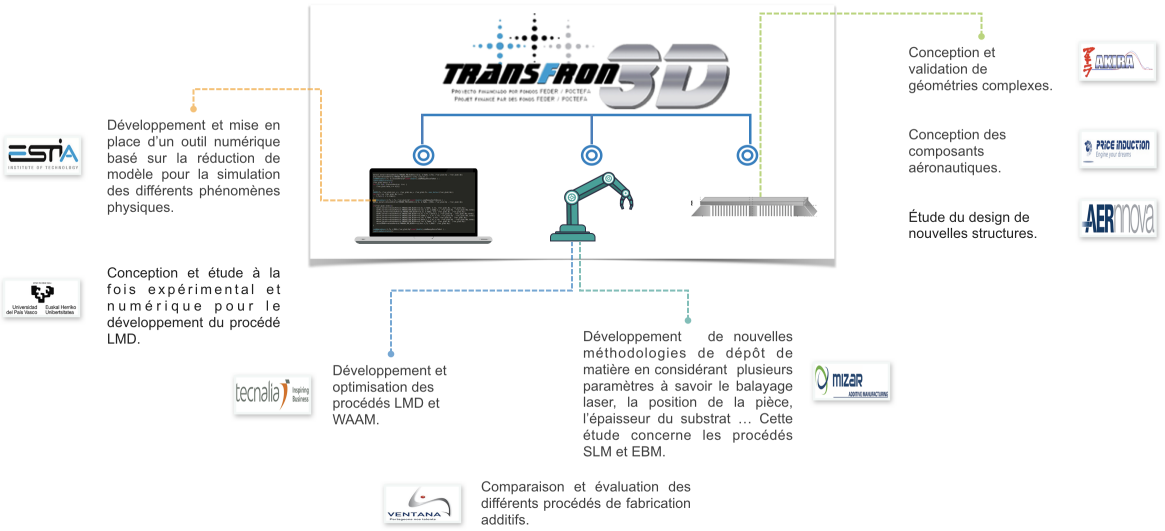


FIGURE 1.7 – Partenaires du projet TRANSFRON3D

8. https://www.transfron3d.eu

Ce projet regroupe comme l'illustre la figure 1.7 des centres technologiques : ESTIA [9], MIZAR [10], des industriels : TECNALIA [11], AKIRA [12], PRICE INDUCTION [13], VENTANA [14], AERNNOVA [15], et des universités : UPV/EHU [16].

La présente thèse a pour vocation de développer un modèle réduit du système décrivant les phénomènes multi-physiques engendrés par le processus SLM. La simulation thermo-mécanique complète ne peut en effet pas être utilisée pour la prédiction de la distorsion de la pièce finale. L'utilisation des modèles réduits permet de mettre en place un outil de simulation complet et précis, tout en considérant la solution en temps réel. Le diagramme de Gantt (figure 1.8) **non détaillé** permet d'avoir un aperçu des différentes options que l'on abordera dans ce manuscrit.

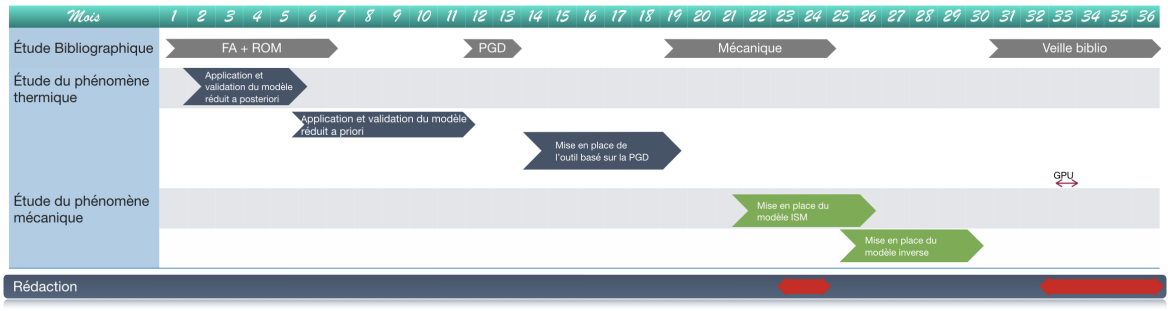


FIGURE 1.8 – Gantt condensé.

1.6 Pourquoi simuler ?

Le procédé de fusion laser sélective sur lit de poudres métalliques, procédé SLM, permet la fabrication de pièces de grandes dimensions, complexes, et sans porosité. Au cours du procédé, de nombreux cycles thermiques se produisent dans la pièce au cours de sa fabrication. Ces gradients de température induisent des déformations plastiques hétérogènes et de ce fait des contraintes résiduelles [105].... En conséquence, en fonction de la structure de la pièce, une distorsion non négligeable peut survenir au cours du processus et entraîner un dysfonctionnement grave ou un arrêt du processus en cours.

La maîtrise de la distorsion de la pièce présente un intérêt majeur pour assurer la conformité géométrique du composant fabriqué. D'une part, on peut essayer d'atténuer la distorsion induite en optimisant les stratégies de construction. D'autre part, on peut intégrer la distorsion au stade de la conception et envisager une forme théorique contre-déformée qui donnerait

9. http://www.estia.fr
10. http://www.mizaradditive.com/
11. http://www.tecnalia.com
12. http://www.akira-technologies.fr
13. http://www.price-induction.com/fr
14. http://www.ventana-group.eu/
15. http://www.aernnova.com/
16. http://www.ehu.eus/fr/

une pièce géométriquement conforme après la construction.

L'intérêt de la simulation numérique est primordial dans ce processus, étant donné les coûts de production. Cette dernière permet de prévoir le comportement mécanique utile pour extraire des données nécessaires à la compréhension et la visualisation de la déformation d'une pièce au cours de sa fabrication, et par la suite améliorer la qualité de la pièce finale. Tout en réduisant le recours à l'expérimentation ce qui permet d'amortir considérablement les différents coûts. En effet, avoir recours à la simulation permet d'appréhender l'impact des différents paramètres en jeu au cours du procédé de FA (a savoir la puissance du laser, le recouvrement, la stratégie de lasage, ...) sur le rendu de la pièce finale, l'apparition des contraintes résiduelles et la prédiction de la formation des porosités et leurs influences sur le comportement mécanique de la pièce [17].

La modélisation du procédé de fusion laser sélective sur lit de poudres métalliques, prend en compte «l'interaction entre les différents phénomènes physiques (thermique, mécanique, métallurgie) et leurs conséquences sur les propriétés mécaniques de la pièce finale et ce à différentes échelles» [17].

On s'intéresse ici (figure 1.9) à la modélisation de l'apport en énergie, au comportement thermo-mécanique, à la modélisation géométrique et à la déformation de la pièce finale. La plupart des modèles macroscopiques sont basés sur une étude utilisant la méthode des éléments finis. Étant donné qu'en FA la géométrie évolue au cours du temps, cette méthode présente donc un inconvénient en termes de temps de calcul qui est relativement élevé [17]. On a alors recours à des méthodes mathématiques afin d'accélérer les temps de calculs. La mise en place d'une stratégie de réduction de modèle (ROM) présente ainsi un grand intérêt.

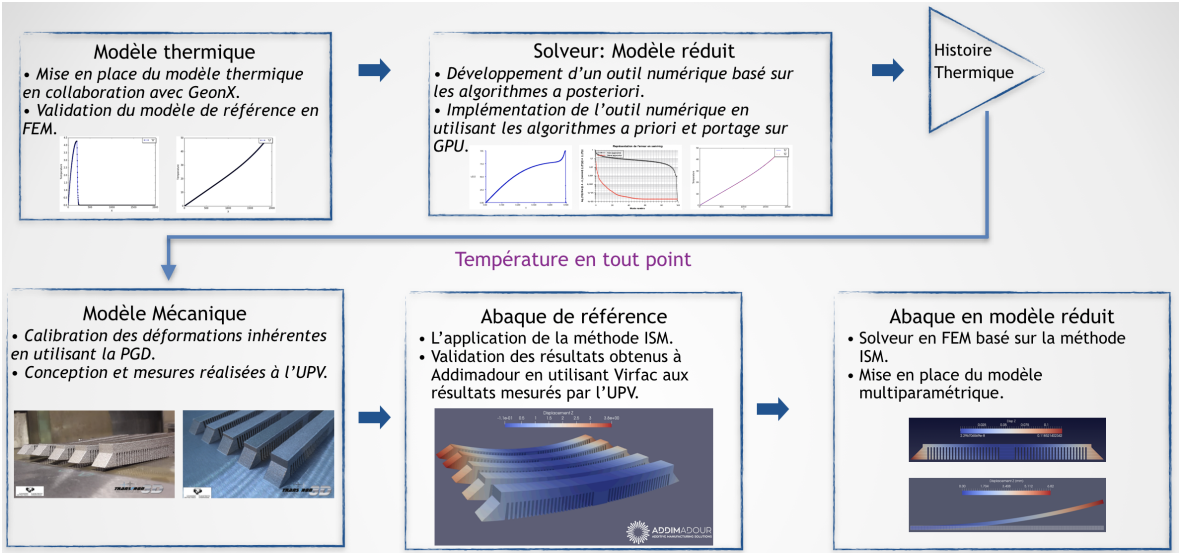


FIGURE 1.9 – Plan de thèse.

17. https://metalblog.ctif.com/2018/06/25/la-simulation-de-la-fabrication-additive-slm/

Chapitre 2

Modélisation des procédés en FA

Ce chapitre brosse dans un premier temps un état de l'art des différents phénomènes physiques qui impactent la qualité de la pièce fabriquée. Dans le cadre de cette thèse, les travaux de recherche sont axés autour du procédé SLM. Une introduction détaillée sera établie pour l'étude des phénomènes thermiques et mécaniques, ainsi que la mise en place des modèles mathématiques afin de décrire les problèmes que l'on cherche à simuler.

Sommaire

2.1 Phénomènes physiques

Dans le cadre des procédés de FA, de nombreux phénomènes physiques sont mis en jeu lors de la fabrication d'une pièce et ce à des échelles très différentes : bain de fusion, cordon, couche, pièce, etc. Dans le cadre de cette étude, le modèle numérique que l'on veut mettre en place vise à estimer la distorsion dans l'ensemble de la pièce fabriquée par SLM et se place à ce titre à une échelle dite macroscopique.

Les trois grandes physiques qui interagissent lors de la fabrication d'une pièce sont représentés dans la figure 2.1. Nous introduisons dans ce qui suit la mise en équations de chacune des physiques afin de construire le modèle numérique qui permet de représenter au mieux la réalité.

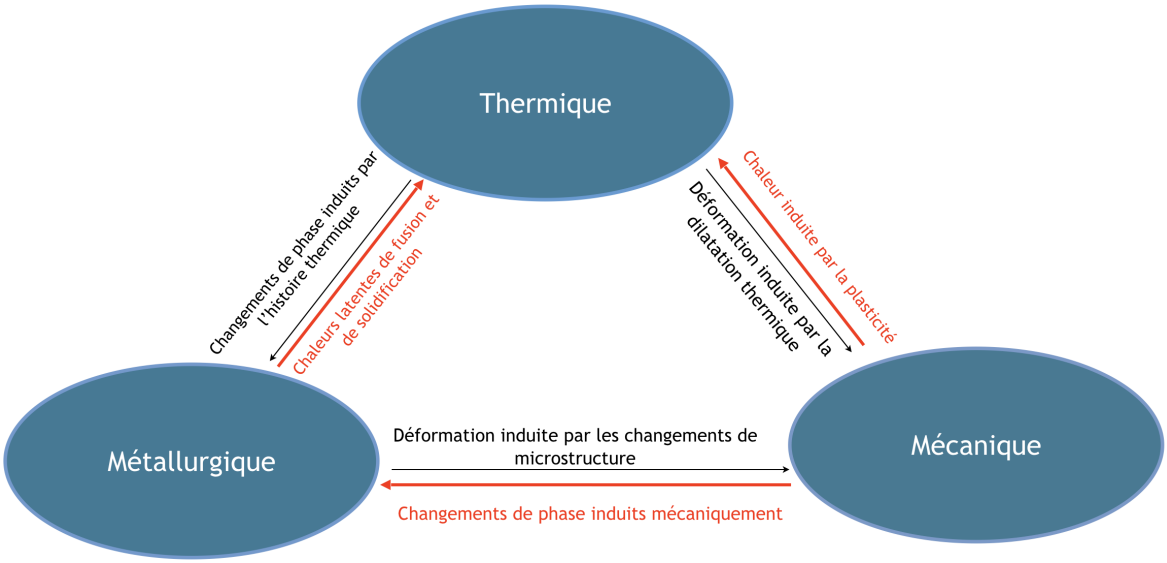


FIGURE 2.1 – Les différents phénomènes physiques qui interagissent au cours de la FA. Représentation en rouge des couplages forts entre ces phénomènes, et en noir les couplages faibles.

1. La **thermique** (fusion, conduction, convection, rayonnement) : au cours du procédé, un important apport en énergie et une lente évacuation thermique par la conduction de chaleur se produisent dans la pièce au cours de sa fabrication, ayant ainsi un impact significatif sur la qualité de la géométrie finale. En effet, ces nombreux cycles entraînent de fort gradients thermiques et provoquent des déformations plastiques hétérogènes, et de ce fait traduisent les modifications microstructurales et macrostructurales que subit le matériau. La modélisation thermique de ce procédé n'est plus vraiment une difficultée, en effet la simulation en FEM a été abordée par plusieurs auteurs [105, 72]. Compte tenu, la plage de température dans laquelle on travaille, on est obligé de prendre en compte la non-linéarité du matériau. La variation des coefficients thermiques ρ, C_p, κ en fonction de la température a un impact sur les temps de calculs.

   Cependant, l'intérêt de notre simulation est d'utiliser d'autres méthodes numériques (ROM) dans le but de récupérer uniquement l'histoire thermique **utile** pour les calculs

mécaniques et métallurgiques [72] en tout point de la pièce au cours de sa fabrication.

2. La **mécanique** (comportement, contraintes et déformations) : les différences de vitesses de chauffe et de refroidissement durant le processus de fabrication induisent des contraintes de traction dans la couche supérieure et des contraintes de compression dans les couches sous-jacentes [105]. En effet, lorsque le laser refond la couche de poudre en surface, le matériau se dilate subissant l'influence des couches inférieures, ce qui provoque une contrainte de compression [1]. Lorsque la limite élastique du matériau est atteinte, une déformation plastique de compression est générée. Au refroidissement, la contraction des couches fondues supérieures entraîne une contrainte résiduelle de traction. En conséquence, un traitement thermique de la pièce après fabrication à des fins de réduction des contraintes reste en général obligatoire. De plus, en fonction de la structure de la pièce, une distorsion non négligeable peut survenir au cours du processus et entraîner un dysfonctionnement grave ou un arrêt du processus en cours.

   Les modèles thermo-mécaniques transitoires fins sont limités dans leur utilisation à des domaines de petite dimension et à des géométries simples. Ils présentent un intérêt réel pour la compréhension du procédé et peuvent être calibrés grâce notamment à des relevés de température et de déplacements réalisés lors de la fabrication d'éprouvettes dédiées. Cependant, le coût de calcul associé rend de tels modèles non adaptés à l'estimation de distorsion à l'échelle d'une pièce complète.

   Des modèles plus légers ont récemment été mis en place afin de contourner l'obstacle lié au coût de calcul. Ils sont inspirés par les stratégies mises en œuvre en simulation numérique du soudage. Les méthodes les plus populaires sont les méthodes dites de "retrait thermique" [42] et de "déformation inhérente" introduite par Ueda et al. [104]. Les deux méthodes sont basées sur un découpage de la pièce en macro-couches, regroupant plusieurs micro-couches réelles. La méthode déformation inhérente (ISM) est détaillée dans la section suivante puisque c'est sur cette approche qu'est axée la partie mécanique de la présente thèse.

3. La **métallurgie** (composition chimique, cristallographie, changement de phase) : l'histoire thermique calculée sert aussi de données d'entrée au modèle métallurgique, visant entre autres à la prédiction des proportions de phases solides créées [71], au calcul d'un taux de porosité. Le couplage métallo-mécanique permet de déterminer le comportement mécanique des matériaux comme le changement de volume relatif lors du changement de phase intervenant au cours du procédé...[112].

Des couplages forts (voir figure 2.1) existent entre chaque physique. Néanmoins, des hypothèses communes concernant ces interactions sont généralement formulées, aboutissant finalement à des couplages faibles (voir les couplages en rouge sur la figure 2.1) et ce, dans le but d'optimiser la vitesse de calcul tout en prédisant de manière réaliste la géométrie et les caractéristiques attendues des pièces fabriquées.

Dans ce contexte, une approche séquentielle (illustrée par la figure 1.9) est généralement mise en œuvre, et consiste à évaluer dans un premier temps le champ de température dans la pièce et son évolution au cours du temps. L'histoire thermique calculée sert alors de données

1. https://metalblog.ctif.com/2018/06/25/la-simulation-de-la-fabrication-additive-slm/

d'entrée au modèle métallurgique. La métallurgie a un impact, qui peut être plus ou moins fort en fonction du matériau, sur la mécanique notamment lors des changements de phase et surtout lors des changements de volumes qui sont liés à ces changements de phase. Quand on passe d'une certaine organisation atomique à une autre, un changement relatif de volume a lieu et peut induire localement de la plasticité en plus de la plasticité de transformation ε_{pt}. Cette déformation est créée à une limite d'élasticité plus faible que la phase initiale. Le champ de température est aussi intégré dans le modèle thermo-mécanique, dont la finalité est le calcul du champ de déplacement, des déformations plastiques et élastiques et du champ de contraintes dans la pièce.

Les coûts de calculs de ce modèle thermo-mécanique transitoire rendent la simulation non adaptée à l'estimation de la distorsion à l'échelle d'une pièce complète, et ce, malgré la mise en place des modèles plus légers afin de contourner cet obstacle. Ces modèles légers sont détaillés dans le tableau [2] 2.2. En conséquence, les méthodes de calculs alternatives (ROM) deviennent donc attractives pour le calcul de problèmes de grande taille, et on peut espérer tirer parti de ces méthodes pour leur grande puissance, leur coût modique, ainsi que leur avantage en matière de compression de données à stocker.

Méthode		Type de résultats	Loi	Description
Thermo-Méchanique (TTMM)		Thermique + mécanique	Élasto-plastique	une densité thermique est calculée en fonction des paramètres du processus et appliquée uniformément sur des macro-couches épaisses
Shrinkage (SM)		Mécanique	Élasto-plastique	une contrainte thermique est calculée en tant que donnée d'entrée à partir des données thermiques du processus et appliquée aux macro-couches
Meso/Macro (LG)	Meso	Thermique + mécanique	Élasto-plastique	la stratégie laser est simulée avec précision sur une micro couche
	Macro	Mécanique	Élastique	les déformations inhérentes extraites du modèle méso sont appliquées à de fines macro couches

FIGURE 2.2 – Particularité des simulations en FA (source Virfac).

Pour fixer les idées, nous détaillerons dans la suite de ce chapitre les modèles physiques, leur mise en place, ainsi que la stratégie de résolution. Une des principales difficultés de la simulation des procédés de FA, est la modélisation de la géométrie, une attention particulière sera portée à la mise en couche de la pièce dans la section suivante.

2.2 Modélisation de la mise en couche

La FA et le soudage possèdent des caractéristiques physiques communes et du fait de ces analogies, certaines méthodes de simulation du procédé de soudage s'étendent également aux applications impliquant des procédés de FA métallique. L'ajout des couches de poudres en fonction du temps est donc reproduit par des méthodes classiquement utilisées dans la

2. http://www.geonx.com/index-2.html

simulation du soudage [73]. Au début de la simulation du modèle thermo-mécanique, comme du modèle thermique, les éléments représentant le cordon de soudure sont *désactivés*.

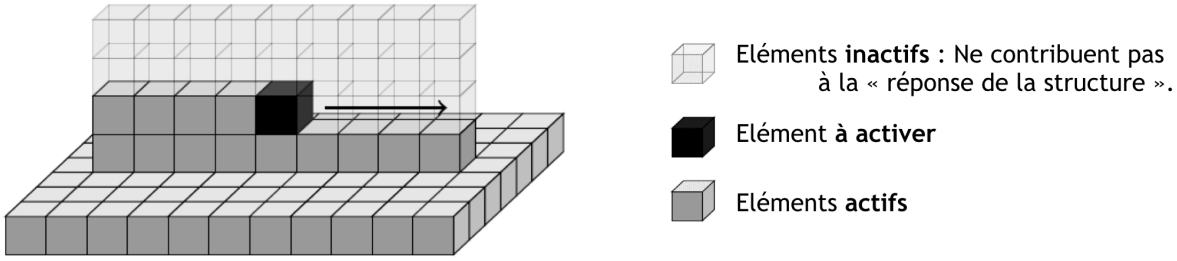


FIGURE 2.3 – Méthodologie d'activation des éléments.

Différentes méthodes permettent de modéliser l'ajout de couches multiples. Néanmoins, deux méthodes sont actuellement utilisées, à savoir les méthodes «*Element Birth and Death*» et «*Quiet Elements*».

L'approche *Element Birth and Death* est la méthode la plus utilisée, et celle qui permet de reproduire le plus fidèlement possible le dépôt de couches, car inclut uniquement les éléments actifs lors de la simulation. Au début de la simulation, les éléments d'une pièce préalablement connue sont tous *transparents vis-à-vis du calcul* bien que faisant partie du maillage. La construction de la pièce est représentée par l'activation progressive des éléments correspondant aux données CAO (voir figure 2.3). Néanmoins, l'étape d'activation peut-être coûteuse, car elle nécessite de recalculer les matrices de rigidité et d'assemblage à chaque changement de géométrie [105].

La prise en compte de l'apport de matière sera abordée dans cette thèse par la méthode *Quiet Elements*, qui suppose aussi une connaissance préalable de la géométrie avant d'effectuer le calcul. Contrairement à la méthode précédente, tous les éléments sont assemblés dans la matrice de rigidité globale. Les éléments constituant la pièce sont préalablement inactivés en définissant astucieusement des propriétés thermiques et mécaniques dégradées (faible conductivité thermique, faible module de Young, ...). Suivant la stratégie de balayage, un élément s'active (attribution de propriétés mécaniques et thermiques appropriées) s'il croise le faisceau du laser. Sinon, il reste inactif.

2.3 Modélisation numérique de la thermique

À partir des hypothèses émises précédemment, la simulation thermique à l'échelle macroscopique est d'une grande importance puisque la prédiction des transferts thermiques à chaque instant au cours de la fusion des poudres métalliques passe par la détermination de la distribution du champ de température provenant de l'interaction du laser sur le lit de poudre [105]. La mise en œuvre d'un modèle de référence nécessite de reproduire fidèlement, à chaque instant de la fabrication, les paramètres laser, tels que la quantité d'énergie ou le diamètre du faisceau laser, ou encore les paramètres de balayage, tels que la vitesse de déplacement de la source d'énergie tout au long de la trajectoire de dépôt, la stratégie de balayage... L'utilisation

de propriétés matériaux adaptées, avec prise en compte de leur évolution en fonction de la température est nécessaire.

2.3.1 Modèle mathématique

L'évolution du champ de température dans une couche de poudre en fonction de l'avancement du laser est définie en résolvant l'équation de la chaleur dans un premier temps via la méthode des éléments finis. Dans ce qui suit, nous écrivons mathématiquement les différents phénomènes thermiques en jeu, en utilisant une approche lagrangienne. De nombreux travaux existent pour la compréhension et la maîtrise du SLM [115, 15, 90].

On cherche à déterminer le champ de température $u(X,t)$ dans un domaine $\Omega = [0,L]^3$. On s'intéresse donc au problème transitoire (2.1) au cours de l'intervalle de temps $I = [0,T_f]$ ce qui permet de représenter fidèlement la fabrication en SLM.

$$\begin{cases} \rho\, C_p \dfrac{\partial\, u}{\partial\, t}\,(X,t) + \nabla\Big(\kappa(X,t)\cdot\nabla\, u(X,t)\Big) & = Q(X,t) \\ u(X,t=0) & = u_0(X) \end{cases} \tag{2.1}$$

avec :

Nom	Définition	Unité
ρ	Masse volumique	$kg.m^{-3}$
C_p	Chaleur spécifique	$J.kg^{-1}.^{\circ}C^{-1}$
$u(X,t)$	Champ de température en fonction de l'espace ($X = (\vec{x},\vec{y},\vec{z})$ dans le cas 3D) et en fonction du temps	$^{\circ}C$
κ	Conductivité thermique	$W.m^{-1}.^{\circ}C^{-1}$
Q	Source volumique de chaleur	$W.m^{-3}$,
u_0	température initiale imposée à tous les éléments inactifs afin de représenter le plus fidèlement possible l'activation des éléments	$^{\circ}C$
T_f	Temps finale nécessaire pour réaliser la simulation	s

TABLE 2.1 – Définition des paramètres.

2.3.2 Modélisation de la source de chaleur

La figure 2.4, représente de façon schématique le transfert de chaleur au cours de l'empilement des différentes couches représentant la pièce. Durant la phase de lasage, un faisceau laser focalisé fait fondre au point d'impact la poudre métallique suivant une trajectoire de balayage prescrite durant une très courte durée. L'énergie photonique est partiellement transformée en énergie thermique par absorption, et une partie est réfléchie. L'énergie thermique absorbée engendre une augmentation de la température de la poudre au delà de son point de fusion. La chaleur diffusée dans le reste de la pièce est évacuée au niveau des surfaces en contact

avec l'air par convection et radiation, mais aussi par conduction dans le substrat et le lit de poudre.

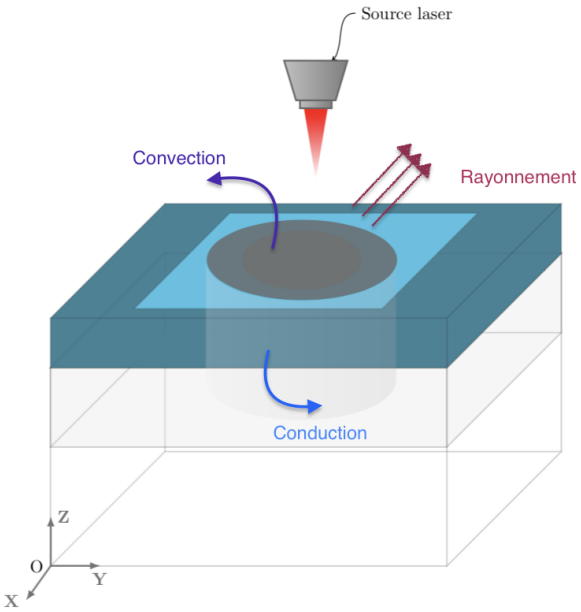


FIGURE 2.4 – Représentation schématique du transfert de chaleur.

Dans les différentes études numériques [105, 72, 55, 114], l'apport de chaleur est modélisé par une fonction volumique permettant de prendre en considération tous les paramètres qui peuvent influencer le résultat final. Comme sus-mentionné la simulation du procédé de FA est étroitement inspirée de la simulation du soudage avec quelques adaptations afin de répondre aux exigences de la FA. La source de chaleur la plus utilisée en soudage [100] est une source volumétrique à double ellipsoïde (2.5) développée par Goldak et al. [40]. Cette source est surtout utilisée en DED (*dépôt sous énergie concentrée*). La densité volumique de flux de chaleur est supposée de type Gaussienne

$$Q(\xi(x),t) = Q = \begin{cases} \dfrac{6\sqrt{3} f_f Q_s}{a_f b c \pi \sqrt{\pi}}\, e^{-3\left(\frac{\xi^2}{a_f^2}+\frac{\eta^2}{b^2}+\frac{\gamma^2}{c^2}\right)} & \text{si} \quad \xi \geqslant 0 \\ \dfrac{6\sqrt{3} f_r Q_s}{a_r b c \pi \sqrt{\pi}}\, e^{-3\left(\frac{\xi^2}{a_r^2}+\frac{\eta^2}{b^2}+\frac{\gamma^2}{c^2}\right)} & \text{si} \quad \xi < 0 \end{cases} \tag{2.2}$$

Où $\mathbf{a_f} = [a_f, b, c]$ et $\mathbf{a_r} = [a_r, b, c]$ représentent respectivement les deux géométries de l'ellipsoïde avant et arrière (voir figure 2.5). Dans le but de déterminer les différents paramètres de ce modèle (2.2) flexible, une phase de calibration sur des données expérimentales (la taille de la zone fondue, a la largeur de la zone fondue, la hauteur totale de la zone fondue, c la pénétration de la source dans la couche précédente) est nécessaire. Puis, Q_s est la puissance effective transmise au matériau par la source, f_f et f_r représentent respectivement la répartition de la chaleur dans les zones avant et arrière.

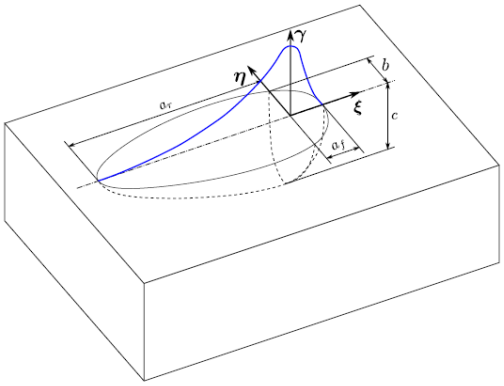


FIGURE 2.5 – Source de chaleur de Goldak

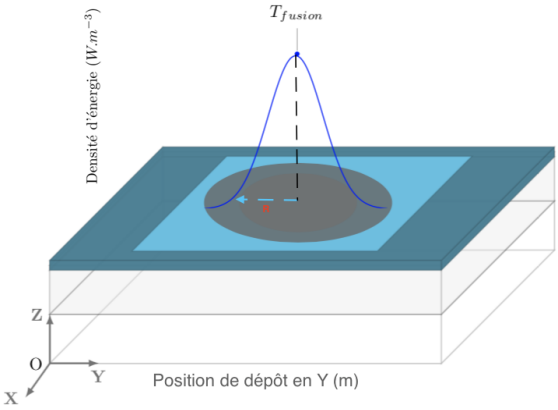


FIGURE 2.6 – Flux de chaleur Gaussien

Une alternative à cette fonction est l'utilisation d'une source surfacique (illustrée par la figure 2.6) basée sur un cercle de rayon (R) du faisceau laser pendant un temps déterminé en fonction de la taille de l'élément et de la vitesse de déplacement de la buse.

$$Q(r) = \frac{2P}{\pi R^2}\, e^{-2\left(\frac{r^2}{R^2}\right)} \tag{2.3}$$

Avec P la puissance du laser, et R le rayon par rapport au centre du faisceau laser.

En SLM, la source de chaleur peut être introduite comme une source mobile dépendant de l'espace, du temps, et d'un certain nombre de paramètres. «Les paramètres fonctionnels qui influencent l'intensité du faisceau laser sont la puissance, la vitesse de déplacement, l'espacement entre les différentes passes (*hatch distance*) et la longueur du scan laser» [3], cette source sera détaillée dans le chapitre 4.

2.3.3 Conditions aux limites

Pour s'assurer de simuler l'échange d'énergie avec le milieu extérieur, deux conditions aux limites sur toute la surface extérieure sont prises en compte :

- Convection sur faces extérieures

$$\kappa(T)\,\Delta T \cdot n = h\,(T_{ref} - T) \;\; sur\; \Gamma_{conv}$$

  avec h le coefficient d'échange par convection, T_{ref} la température courante, et T la température extérieure. Les pertes de chaleur par convection avec l'air peuvent ne pas être prises en compte.

- Radiation sur faces extérieures

$$\kappa(T)\,\Delta T \cdot n = \epsilon\,\sigma\,(T_{ref}^4 - T^4)\; sur\; \Gamma_{rad}$$

3. https://metalblog.ctif.com/2018/06/25/la-simulation-de-la-fabrication-additive-slm/

avec ϵ l'émissivité du matériau, σ la constante de Stefan-Boltzman.

La conduction dans le lit de poudre et le substrat peut être remplacée par de la convection en considérant un coefficient d'échange h élevé.

2.4 Modélisation numérique de la mécanique

L'analyse mécanique est basée sur une formulation élastoplastique avec un incrément mécanique non linéaire calculé après chaque incrément thermique. Les contraintes thermiques dues aux différents cycles thermiques, qui génèrent des dilatations et des contractions localisées du matériau, conduisent à la création de contraintes et de déformations.

Différents travaux ont analysé les phénomènes thermomécaniques des procédés additifs. L. Van Belle [105] a développé dans sa thèse un modèle numérique permettant de prédire l'apparition des contraintes résiduelles dans les pièces fabriquées en fusion laser de poudres métalliques. Le champ de température est obtenu à l'aide d'un modèle de balayage laser, puis les résultats sont utilisés pour réaliser le calcul mécanique et finalement prédire les champs de contraintes résiduelles.

Dans les travaux d'A. Longuet [72], un modèle mécanique a été validé en analysant la déflexion du substrat obtenue lors de la fabrication d'un mur. Ce modèle permet de prédire la distorsion à l'échelle macroscopique. Tous ces travaux sont basés sur un calcul séquentiel des différents phénomènes physiques (calcul de la thermique, puis la mécanique, puis la métallurgie).

C. Li et al. [62] ont évalué les sources de contraintes résiduelles, leurs caractéristiques et leurs mesures d'atténuation. Les auteurs ont mis l'accent sur les méthodes de mesure et les caractéristiques de la contrainte résiduelle à la fois dans les pièces métalliques en cours de construction et les pièces post-traitées.

2.4.1 Modèle mathématique

Le modèle mathématique, que nous considérons dans la suite, est le problème d'équilibre qui consiste à déterminer le champ de déplacement u. On adopte généralement l'hypothèse des petites perturbations (HPP)

$$\begin{cases} div(\sigma(u)) = 0 \qquad \text{sur} \quad \Omega, \\ \sigma = \mathcal{C} : (\varepsilon - \varepsilon_{th} - \varepsilon_p) \\ \varepsilon(u) = \dfrac{1}{2}\left(\nabla u + \nabla^T u\right). \end{cases} \tag{2.4}$$

Où $\Omega \subset \mathbb{R}^3$ est un domaine borné. σ le tenseur des contraintes de Cauchy générées par la composante élastique de la déformation $\mathcal{C}$ est le tenseur de rigidité, et ε le tenseur des déformations infinitésimales.

La déformation totale ε est une somme de différentes composantes

$$\varepsilon = \varepsilon_e + \varepsilon_{th} + \varepsilon_p \tag{2.5}$$

La déformation élastique ε_e augmente linéairement avec la contrainte. Dans le cadre de l'élasticité des matériaux isotropes. Le tenseur de déformations peut alors être exprimé en utilisant la Loi de Hooke.

$$\varepsilon_e = \frac{1+\nu}{E}\sigma - \frac{\nu}{E}tr(\sigma)\mathbb{I}_3 \tag{2.6}$$

avec ν et E respectivement le coefficient de poisson et le module de Young du matériau.

Les variations cycliques de température, conduisent à un rétraction de la pièce métallique par dilatation thermique. Le coefficient de dilatation thermique moyen α entre la température de référence T où la déformation thermique est nulle et $T + \Delta T$ la température de chauffe mesurée à un instant t donné, et N_{steps} le nombre d'incréments temporels.

$$\varepsilon_{th} = \sum_{t=1}^{N_{steps}} \Delta\varepsilon_{th}^{t} \text{ avec } \Delta\varepsilon_{th} = \int_{T}^{T+\Delta T} \alpha(T)\, dT \tag{2.7}$$

Lorsque le matériau a un comportement élastoplastique, la déformation plastique est constatée après application d'une sollicitation mécanique (contrainte de traction, contrainte de flexion). Les figures (2.7) et (2.8) illustrent deux essais d'éprouvettes soumises à un effort de traction avec décharge. Lorsque la sollicitation mécanique lors du chargement est inférieure à la limite d'élasticité $\sigma < \sigma_Y$, l'éprouvette reste élastique. La relation contrainte-déformation (cf. figure 2.7) est linéaire.

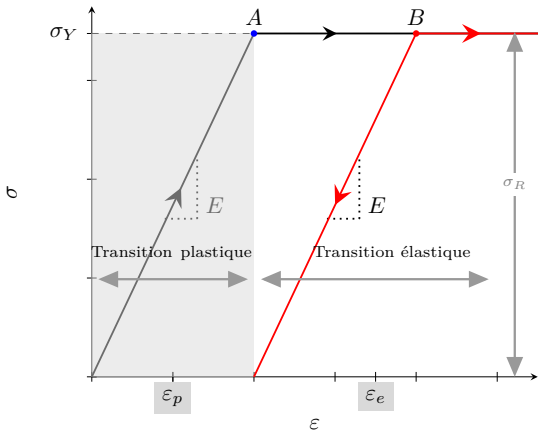


FIGURE 2.7 – Représentation de la relation entre contraintes et déformations.

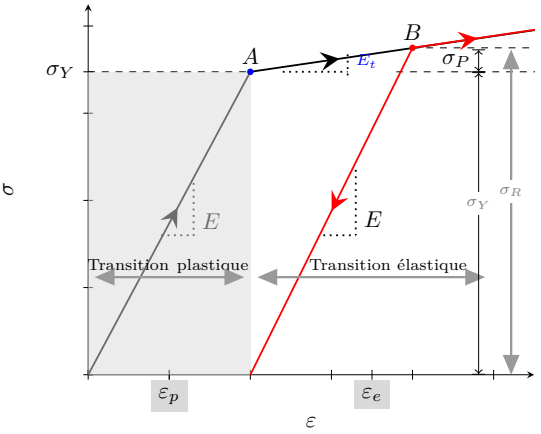


FIGURE 2.8 – Représentation de la courbe déformation-contrainte en déformation plastique.

La figure 2.7, illustre le comportement d'un matériau élasto-plastique en l'absence d'écrouissage. Au point A, la limite élastique initiale est atteinte. Une augmentation de la déformation n'entraîne ensuite aucune augmentation de la contrainte, qui demeure égale à la limite

d'élasticité ($\sigma_Y = \sigma_R$). Ce modèle ne décrit pas fidèlement la physique, pour ce, il nécessite d'incorporer des effets d'écrouissage.

Avec écrouissage, l'augmentation de la déformation au delà de ε_A entraîne une augmentation de la contrainte (et donc de la composante élastique de déformation). La limite élastique initiale évolue (cf. figure 2.8).

L'évolution de la limite d'élasticité dépend du mode d'écrouissage du matériau. Si l'écrouissage est linéaire, on définit par E_t le module tangent en A lorsque la contrainte devient plus grande que la limite élastique.

2.4.2 L'origine des contraintes résiduelles

Toute contrainte présente dans le corps en l'absence de chargement extérieur, dite résiduelle, peut être vue comme le résultat d'un état de déformation inhérente [47]. En effet, comme le montre Maitournam [76] (cf. équation 2.8), l'origine de l'apparition des contraintes résiduelles à la fin du déchargement peut être diverse, elles peuvent être générées à partir de gradients de température, résulter d'une déformation plastique locale, ou encore d'une transformation de phase...

$$\begin{cases} div(\sigma^r) = 0 & \text{sur} \quad \Omega, \\ [\![\sigma^r]\!] \cdot n = 0 & \text{sur} \quad \Sigma, \\ \sigma^r_{ij} \cdot n_j = 0 & \text{sur} \quad S_{T_i}, \quad i = 1,2,3 \\ u^r_i = 0 & \text{sur} \quad S_{u_i}, \quad i = 1,2,3 \\ \varepsilon^r = \dfrac{1}{2}\left(\nabla u^r + \nabla^T u^r\right) \\ \sigma^r = \mathcal{C} : (\varepsilon^r - \varepsilon_p). \end{cases} \tag{2.8}$$

Où σ^r représente la contrainte résiduelle du problème élastique avec comme champ de déformation initiale ε_p, Σ est la surface de discontinuité du champ de contrainte, $[\![\sigma^r]\!]$ représente le saut du tenseur de contrainte selon le vecteur unitaire n normal à la surface au point considéré, la structure est soumise à des forces surfaciques sur les parties $S_{T_i}, i = 1,2,3$ et à des déplacements sur les parties $S_{u_i}, i = 1,2,3$ du domaine.

Les contraintes résiduelles peuvent conduire à une distorsion importante post-fabrication, après découpe des supports et séparation de la pièce du substrat. En fonction de la structure de la pièce, une distorsion non négligeable peut aussi survenir au cours du procédé. La maîtrise de cette distorsion présente à ce titre un intérêt majeur pour assurer la conformité géométrique du composant fabriqué et aboutir à un procédé robuste.

2.4.3 Introduction à la méthode de déformation inhérente pour l'estimation de distorsion

Pour mieux appréhender le comportement de la pièce pendant les différents cycles de chauffage et de refroidissement, nous nous intéressons à la méthode de déformation inhérente.

La déformation inhérente générée par le procédé SLM dans la pièce fabriquée peut être obtenue de deux manières : par le calcul ou par la mesure. D'une part, le calcul de la déformation inhérente est basé sur une simulation thermo-mécanique transitoire réalisée sur un volume élémentaire [57].

D'autre part, la déformation inhérente peut être évaluée en mesurant le retour élastique généré lorsque la pièce est séparée du substrat. Une analyse inverse est ensuite effectuée pour identifier les déformations inhérentes responsables d'un tel retour élastique. Cette méthode sera abordée plus en détails dans le chapitre 6.

Chapitre 3

Vue d'ensemble sur les algorithmes de réduction de modèle

Ce chapitre propose un état de l'art des différents algorithmes de réduction de modèle, organisés en deux grandes familles : les algorithmes *a posteriori* et les algorithmes *a priori*. Cette dernière classe est la plus adaptée pour la résolution des problèmes multiparamétriques. Ces algorithmes seront étudiés en détail tout au long de cette thèse.

Sommaire

3.1 Motivations

«La résolution des équations mathématiques issues d'un modèle physique requiert de choisir entre différentes méthodes numériques. Une méthode bien connue est celle des éléments finis» [84]. Dans le cadre de ce travail, elle consiste à résoudre une équation aux dérivées partielles. Néanmoins, ces méthodes classiques conduisent à des temps de calcul prohibitif. Les limites physiques des machines poussent alors les chercheurs à exploiter les avantages qu'offrent les architectures parallèles, notamment la nouvelle pratique connue sous la terminologie General-Purpose computations on GPU (GPGPU) destinée au calcul scientifique. De nouveaux outils mathématiques permettent de tirer profit de la puissance de calcul phénoménale offerte par cette nouvelle lignée de processeurs graphiques (GPU). Les méthodes de calcul alternatives deviennent donc attractives pour le calcul de problèmes de grande taille. Le principe des algorithmes de réduction de modèle consiste à projeter le problème de départ et le résoudre dans un domaine de dimension plus petite que la dimension initiale.

De manière générale, un phénomène physique est décrit par l'équation différentielle suivante :

$$\begin{cases} \dfrac{\partial u}{\partial t} + \mathcal{F}(u) = B \\ + \textit{conditions initiales} \\ + \textit{conditions limites} \end{cases} \tag{3.1}$$

où u est un vecteur représentant les variables que l'on cherche à déterminer dans l'espace de Hilbert V de dimension infinie, $\mathcal{F}$ l'opérateur différentiel décrivant le phénomène étudié, et B représente la donnée du problème.

Définition 3.1
On définit l'espace de Sobolev $H^1(\Omega)$ *par*

$$H^1(\Omega) = \left\{ v \in L^2(\Omega),\ v \textit{ admet une dérivée } \frac{\partial v}{\partial x} \in L^2(\Omega) \textit{ au sens des distributions} \right\}$$

Théorème 3.1
L'espace $H^1(\Omega)$ *muni de la norme* $\|\cdot\|_{1,\Omega}$ *est un espace de Hilbert.*

L'idée de l'approximation variationnelle est de définir un sous-espace vectoriel $V_h \subset V$ de dimension finie. Le problème discret est alors donnée par

$$\begin{cases} \text{Trouver } u_h \in V_h \text{tel que} \\ \dfrac{\partial u_h}{\partial t} + \mathcal{F}_h(u_h) = B_h, \quad \forall v_h \in V_h \\ + \textit{conditions initiales} \\ + \textit{conditions limites} \end{cases} \tag{3.2}$$

La discrétisation spatiale du problème physique par éléments finis consiste alors à résoudre le problème (3.1) dans un espace V_h de dimension $dimV_h = N < \infty$. Le nouveau système est alors de dimension finie, néanmoins les temps de calculs dépendent étroitement de la taille du maillage. En effet, certains problèmes physiques nécessitent la réalisation d'un maillage fin ce qui engendre l'augmentation de la dimension du problème.

L'approximation par réduction de modèle du problème (3.1) est utilisée pour contourner cette limitation due aux contraintes géométriques sur le maillage (structure volumineuse ou maillage très fin), ainsi que les contraintes liées au temps de calcul. L'avantage de ce type d'approche, est de trouver une approximation (cf. figure 3.1) de la solution $u(\mu)$ et des résultats possibles dans un sous-espace vectoriel $\mathcal{V}_h$ de V_h de petites dimensions ($dim\mathcal{V}_h \ll dimV_h$) construit à partir de captures instantanées [43] tout en conservant les grandes hypothèses physiques (géométries, lois de comportement, conditions aux limites ...).

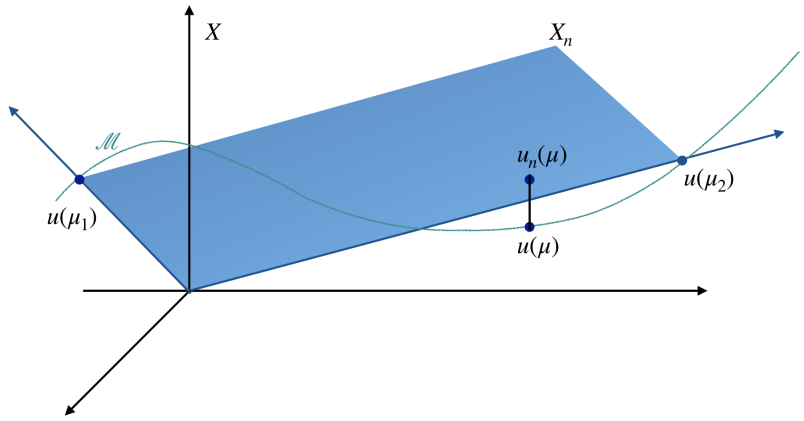


FIGURE 3.1 – Illustration de la solution par RB.

Se pose maintenant la question de la mise en place d'un domaine d'ordre réduit, et du choix de la méthode de résolution. On distingue les méthodes dites *a posteriori* [80, 89, 5, 67, 63], qui permettent d'approximer la solution dans une base réduite, contrairement aux méthodes dites *a priori* [37, 58, 39, 21] utilisées pour une simulation sans connaissances préalables de la base.

3.2 Décomposition Orthogonale aux valeurs Propres

Initialement adaptée en 1967 par Lumley [75] afin d'extraire les informations essentielles d'un écoulement turbulent, la Décomposition Orthogonale aux valeurs Propres (POD) consiste à effectuer le traitement d'un signal et à en capturer les informations caractéristiques qui permettront d'en prédire l'évolution temporelle [29]. Dans un grand nombre d'applications, la POD a aujourd'hui remplacé la simulation classique car elle permet une optimisation au sens énergétique [5]. En effet, un petit nombre d'informations extraites d'un ensemble de données constitue la base d'un système réduit dont le comportement permet une bonne approximation du système complet.

L'objectif de la POD, est de déterminer une base constituée de fonctions propres orthonormée dans un sous-espace prédéfini comme illustré par la figure 3.2. Cette méthode a été introduite en 1945 par M. Loève [70] sous le nom de Karhunen Loève Expansion (KLE), ou encore Principal Component Analysis (PCA) [54], et elle est surtout connue sous l'appellation Singular Value Decomposition (SVD) [64].

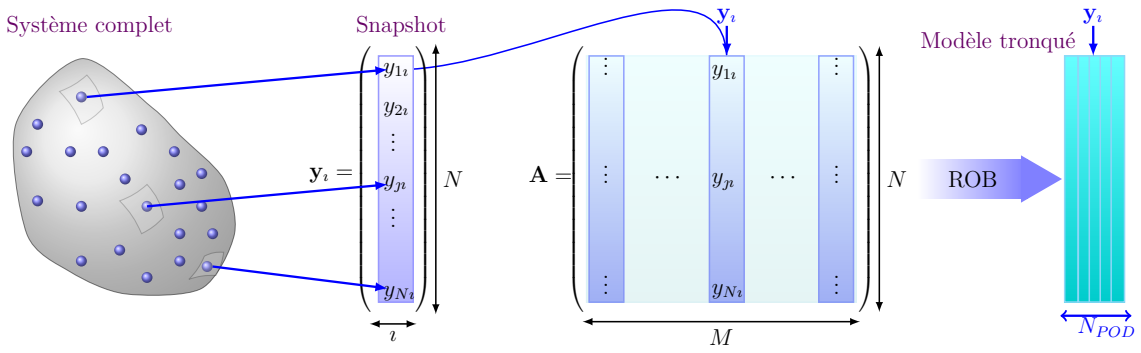


FIGURE 3.2 – Illustration de la solution par POD.

Les développements récents autour de la POD font que cette dernière n'est donc plus considérée uniquement comme une méthode de post-traitement, mais permet la construction d'une base de réduction de modèle, et est utilisée dans diverses disciplines : le traitement d'images, l'analyse des signaux, les statistiques, etc... [89]

Dans cette partie, nous réalisons une brève présentation de la POD en ne développant que les notions abordées dans la suite de cet exposé. Après une introduction de la POD dans le contexte général des méthodes d'approximation (cf. 3.2.1), nous présenterons la SVD (Singular Value Decomposition)(cf. 3.2.2) une méthode qui sera utilisée dans la suite de ce travail. Nous nous intéresserons plus particulièrement, à la méthode APR (cf. 3.3) qui permet de prédire le comportement d'un phénomène physique en se basant sur l'enrichissement d'une base de départ.

3.2.1 Méthodes d'approximation

On considère le champ u, comme une fonction dépendante des variables d'espace et de temps. La méthode de séparation de variables espace-temps consiste à chercher une approximation, notée u_M du champ u sous forme de somme de produits de fonctions de x et de fonctions de t [16] :

$$u(x,t) \simeq u_M(x,t) = \sum_{m=1}^{M} \omega_m(x)\,\varphi_m(t) \tag{3.3}$$

la fonction ω_m est appelée mode en espace, et la fonction φ_m est appelée mode en temps, et M représente le nombre de modes retenus. L'objectif de cette technique est de réduire la

quantité de données nécessaires à la reconstruction du champ u tout en obtenant une bonne approximation de ce dernier.

L'idée principale est donc d'extraire les modes les plus significatifs ω_m et de les utiliser comme fonctions de base dans la résolution d'autres problèmes [66, 8, 65], afin d'obtenir une bonne approximation. Il est par suite possible d'obtenir les modes φ_m à l'aide du produit scalaire suivant :

$$\varphi_m(t) \;=\; (u(x,t)\,,\,\omega_m(x)) \;=\; \int_\Omega u(x,t)\,\omega_m(x)\,dx$$

Pour résoudre le problème d'approximation (3.3), il suffit de déterminer les modes ω_m. Nous supposons donc connues les valeurs de la fonction $u(x,t)$ en N_x points de l'espace et en N_t instants, et par suite les modes ω_m sont obtenues par résolution du problème de minimisation :

$$\underset{\omega_m(x)}{\operatorname{argmin}} \sum_{m=1}^{N_t} \|u(x,t) \;-\; \varphi_m(t)\,\omega_m(x)\|^2 \tag{3.4}$$

À chaque instant $t_\imath$, l'ensemble des réalisations est stocké dans la colonne $\imath$ d'une matrice A de dimension $N_x \times N_t$ dite matrice des Snapshots [28] :

$$A \;=\; \begin{pmatrix} u(x_1,t_1) & u(x_1,t_2) & \cdots & u(x_1,t_{N_t}) \\ u(x_2,t_1) & u(x_2,t_2) & \cdots & u(x_2,t_{N_t}) \\ \vdots & \vdots & & \vdots \\ u(x_{N_x},t_1) & u(x_{N_x},t_2) & \cdots & u(x_{N_x},t_{N_t}) \end{pmatrix} \tag{3.5}$$

3.2.2 La Décomposition aux Valeurs Singulières (SVD)

Soit la matrice A de dimension $N_x \times N_t$ définie ci-dessus (3.5), une Décomposition aux Valeurs Singulières est une application linéaire composée de trois actions fondamentales illustrée par la figure 3.3. Cette figure spécifie que la matrice V représente une rotation, s'ensuit alors une dilatation via la matrice Σ, suivie d'une autre matrice de rotation U comme indiqué dans [99].

Mathématiquement, la matrice A peut être décrite par le produit de ces trois matrices :

$$A = U\Sigma V^T \tag{3.6}$$

avec $U\,U^T \;=\; I_{N_x}$, $V\,V^T \;=\; I_{N_t}$ et $\Sigma \;=\; diag(\sigma_1,\cdots,\sigma_r)$ une matrice diagonale de taille $N_x \times N_t$ contenant les valeurs singulières de A par ordre décroissant $\sigma_1 \geqslant \cdots \geqslant \sigma_r > 0$ qui correspondent aux longueurs des axes principaux de la matrice U (cf. figure 3.3.), et $r = \min(N_x, N_t)$.

En utilisant le théorème d'Eckart Young [32], on peut déterminer l'approximation de la

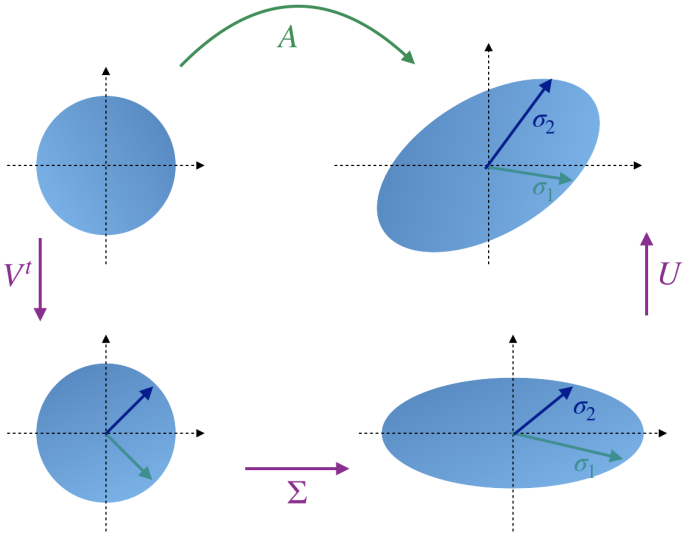


FIGURE 3.3 – Interprétation géométrique de la SVD.

matrice A en résolvant le problème de minimisation

$$\operatorname{argmin} \|A - A_k\|_F = \sqrt{\left(\sum_{i=k+1}^{r} \sigma_i^2(A) \right)}$$

Cette décomposition est une méthode de compression d'information. La simulation par POD est basée sur une information préalablement connue provenant de simulations numériques ou encore d'expériences. Dans la suite, nous considérons que la POD permet de fournir une base pour une méthode *a priori*.

3.2.3 Caractéristiques des modes POD

La base obtenue possède les propriétés suivantes

- Les modes temporels compatibles peuvent être obtenus par projection du champ solution sur la base ω_n
$$a_n(t) = (u, \omega_n)$$
- Les fonctions propres POD sont orthogonales deux à deux en utilisant le produit scalaire $\int_\Omega \nabla\omega_m \cdot \nabla\omega_n^\star \, d\Omega = 0$ si $m \neq n$, elles sont aussi orthonormales par le produit scalaire suivant $\int_\Omega \omega_m \cdot \omega_n^\star \, d\Omega = \delta_{m\,n}$
- Soit la forme bilinéaire $\int_\Omega \omega_m \cdot \omega_n^\star \, d\Omega$ symétrique, continue et définie positive sur V^2. Le quotient de Rayleigh est définie par
$$\mathcal{R}(\omega_m) = \lambda_m = \frac{\int_\Omega \nabla\omega_m \cdot \nabla\omega_n^\star \, d\Omega}{\delta_{m\,n}}$$

- L'erreur engendrée par la base correspond à la somme des valeurs propres rejetés

$$\langle \| u(x,t) - \sum_{m=1}^{M} \omega_m(x)\, \varphi_m(t) \|^2 \rangle \; = \sum_{m=M+1} \lambda_m$$

3.3 Réduction a Priori (APR - POD)

Dans le but d'accélérer les temps de calculs la méthode APR (*A Priori Reduction*) est utilisée. Cette méthode n'exploite aucune connaissance préalable de la solution à approcher u_M (3.3), mais se base plutôt sur l'enrichissement d'une base initialement déterminée. Pour ce, on utilise comme base initiale les informations compressées en utilisant la POD.

3.3.1 Méthodologie de résolution

Cette méthode itérative nécessite une première étape d'initialisation de base du problème étudié. L'influence du choix de cette base est étudié dans les travaux de thèse de N. Verdon [109]. Dans ce qui suit, la base initiale est déterminé à partir des snapshots préalablement calculé en utilisant la POD.

En considérant la solution du problème discret 3.2 sous la forme séparée 3.3, le problème discret devient simplement

$$\frac{\partial}{\partial\, t} \left[\sum_{m=1}^{M} \omega_m(x)\, \varphi_m(t) \right] + \mathcal{F}_h \left[\sum_{m=1}^{M} \omega_m(x)\, \varphi_m(t) \right] = B_h$$

La projection de cette expression sur une base APR, peut être exprimée comme suit

$$\sum_{m=1}^{M} (\omega_m(x), \omega_n(x)) \frac{\partial\, \varphi_m(t)}{\partial\, t} + \left(\mathcal{F}_h \left[\sum_{m=1}^{M} \omega_m(x)\, \varphi_m(t) \right], \omega_n(x) \right) = (B_h\,,\, \omega_n(x)) \qquad (3.7)$$

Où $(\cdot, \cdot)$ représente le produit scalaire dans l'espace $L^2(\Omega)$ [109].

Les résidus

$$\mathcal{R}(x,t) = \sum_{m=1}^{M} \omega_m(x) \frac{\partial\, \varphi_m(t)}{\partial\, t} + \mathcal{F}_h \left[\sum_{m=1}^{M} \omega_m(x)\, \varphi_m(t) \right] - B_h$$

sont évalués sur la totalité de l'intervalle de temps et leur norme L_2 comparée à un critère de convergence prédéfini par l'utilisateur. Si la norme du résidu dépasse le critère de convergence fixé à un instant t_i, cela signifie que la base de l'itération précédente n'a pas fournie une bonne reproduction du système, cette base doit donc être mise à jour pour l'itération suivante.

La mise à jour de la base s'effectue en deux phases

1. Une première phase d'amélioration qui a pour but de ne garder que les informations significatives sur la dynamique de la solution au cours de l'intervalle de temps obtenues grâce à la dernière itération APR.
2. La deuxième phase est la phase d'enrichissement qui nécessite le calcul de la partie inconnue de la solution.

3.3.2 Synthèse modale

Pour commencer, discrétisons le problème 3.7 avec un pas de temps Δt. Soit $\omega_m(x)$ les modes spatiaux stockés, la solution approchée reconstruite peut être déterminée suivant ces étapes

1. Déterminer la base initiale en utilisant la POD,
2. Résoudre l'équation différentielle 3.7 par un schéma numérique, ce qui permet de déterminer les modes temporels compatibles avec les modes spatiaux,
3. Reconstruire la solution approchée en utilisant

$$u_h(x,t) \;=\; \sum_{m=1}^{M} \omega_{m,h}(x)\, \varphi_{m,h}(t)$$

Cette méthode est avantageuse pour son faible coût de calcul si

- Le temps de calcul de la simulation est long.
- Plusieurs analyses d'une même configuration sont nécessaires, notamment en matière de calibration. En effet, la détermination des modes est l'étape la plus longue et n'est réalisée qu'une fois pour toutes.
- Un petit nombre de modes suffit pour obtenir une bonne approximation, ce qui permet aussi un gain en stockage de données.

3.4 La décomposition généralisée propre (PGD)

Tout comme la méthode vu précédemment, la PGD permet d'obtenir la meilleure approximation d'un système linéaire en alliant la représentation séparée à un algorithme qui construit cette dernière progressivement comme défini par N. Bur [21].

Initialement introduite par P. Ladevèze [59] sous le nom "Approximation radiale" pour résoudre efficacement des problèmes d'évolution. D. Néron et al. [82] passent en revue les développements de la méthode PGD, pour la résolution des problèmes non linéaires dépendant du temps en utilisant la méthode LATIN. L'inconnue u est représentée sous la forme

$$u(x,t) \approx \sum_{m=1}^{n} X_m(x)\, T_m(t)$$

Où $X_m(x)$ et $T_m(t)$ représentent respectivement les modes spatiaux et temporels. La PGD est utilisée pour la simulation de différents phénomènes physiques, comme les problèmes de dynamique transitoire [16], les problèmes multiparamétriques [34], les problèmes inverses [78].

L'avantage de cette méthode est de réaliser plusieurs simulations, dont l'approximation varie peu entre les différentes résolutions. En effet, il s'agit de développer en amont (*off-line*) un abaque numérique une fois pour toutes, en déterminant la solution la plus générale possible du problème 3.1. Cet abaque est ensuite utilisé dans des applications qui nécessitent de déterminer la solution en temps réel.

Afin de décrire l'algorithme de PGD progressive, on définit

$$u_n(x,t) \approx \sum_{m=1}^{n-1} X_m(x)\, T_m(t) + X_n(x)\, T_n(t) = u_{n-1}(x,t) + X_n(x)\, T_n(t)$$

On cherche donc à déterminer $(X_n, T_n) \in \Omega_X \times \Omega_t$ tel que :

$$\frac{\partial\, u_{n-1}}{\partial\, t} + X_n \frac{\partial\, T_n}{\partial\, t} + \mathcal{F}_h \left[u_{n-1} + X_n(x) \cdot T_n \right] = B_h \tag{3.8}$$

Une projection est réalisée sur chacune des inconnues

$$\begin{cases} \left(X_n \dfrac{\partial\, T_n}{\partial\, t} + \mathcal{F}_h \left[X_n(x) \cdot T_n \right], X_n \right)_{L^2(\Omega_X)} = \left(B_h - \dfrac{\partial\, u_{n-1}}{\partial\, t} - \mathcal{F}_h \left[u_{n-1} \right], X_n \right)_{L^2(\Omega_X)} \\ \left(X_n \dfrac{\partial\, T_n}{\partial\, t} + \mathcal{F}_h \left[X_n(x) \cdot T_n \right], T_n \right)_{L^2(\Omega_t)} = \left(B_h - \dfrac{\partial\, u_{n-1}}{\partial\, t} - \mathcal{F}_h \left[u_{n-1} \right], T_n \right)_{L^2(\Omega_t)} \end{cases}$$

Ce problème couplé est résolu par l'algorithme du point fixe détaillé dans le chapitre 5.

Deuxième partie

Simulation du phénomène thermique du procédé SLM

Chapitre 4

Calcul, réduction du champ thermique

La modélisation thermique du procédé SLM, est réalisée par l'approche macro-couche. Le domaine étudié, la pièce, évolue au cours du temps ce qui crée une dépendance entre l'espace et le temps. L'utilisation de méthode de réduction de modèle qui vise justement à séparer l'espace et le temps demande une approche particulière. C'est cette nouvelle approche que nous allons présenter ici, en se limitant à un domaine 1D. Ce qui suffit pour en montrer l'originalité.

Sommaire

4.1 Introduction

Comme nous l'avons exposé dans la section 2.3, la détermination de l'histoire thermique est une donnée importante pour prédire les contraintes et déformations d'une pièce fabriquée en SLM. Les travaux de R. Dayal [50] sont axés sur l'étude du transfert de chaleur, de la fusion, de la solidification de particules métalliques invoquées par chauffage au laser. Dans sa thèse, l'auteur présente les modèles thermiques développés pour décrire le transfert de chaleur, et ce, à différentes échelles. X. Liu [69] aborde aussi dans sa thèse l'utilisation de la méthode des éléments discrets pour la modélisation des phénomènes physiques et notamment thermiques à l'échelle mésoscopique. Irwin et al. [49] présentent un modèle thermo-mécanique pour la modélisation des processus en lit de poudre. Le modèle thermique décrit dans cet article utilise la source de chaleur de Goldak pour simuler avec précision le flux. Les auteurs mettent l'accent sur la relation entre le rayon de la source laser, sa vitesse, ainsi que les incréments de temps de la simulation. Une étude comparative des incréments temporels est discutée, les auteurs présentent la taille d'incrément de temps nécessaire pour réduire le temps de calcul requis pour toute simulation sur lit de poudre tout en maintenant la précision.

De nombreuses études [73, 105, 115] réalisées dans le but de simuler ce procédé sur une pièce industrielle complète portent sur la simulation à l'échelle de la couche ou encore de la macro-couche. Une attention particulière doit être portée à la discrétisation spatiale et temporelle afin de simuler des pièces complexes en utilisant l'approche *Quiet Elements* (cf. 2.2).

En effet, au cours du procédé SLM la modélisation des phénomènes thermiques générés au cours de la fabrication d'une pièce requiert de prendre en considération les discrétisations spatiales et temporelles à chaque dépôt de couche, la répétabilité, ce qui induit d'importants coûts de calculs. En effet, au fur et à mesure de la fabrication, une nouvelle couche est déposée, les calculs sont effectués sur la couche active ainsi que sur les couches du dessous, au bout d'un moment le domaine d'étude devient très grand. Ce problème nécessite à titre indicatif un million de degrés de liberté en espace et un million de pas de temps, ce qui demande beaucoup de grandeur à gérer et surtout à stocker. L'objectif ici est donc de pouvoir prédire l'histoire thermique en utilisant des algorithmes de réduction de modèles.

Étant donné qu'en SLM le phénomène thermique impacte principalement la hauteur de la pièce, on s'intéresse dans cette partie uniquement à la simulation du procédé dans un domaine Ω uni-dimensionnel (cf. figure 4.1). Nous nous intéressons d'abord à la meilleure manière de séparer en espace et en temps, le champ thermique par une approche POD. Une méthode de changement de variable est proposée. Nous abordons par la suite la résolution par une approche APR.

4.1.1 Équation de la chaleur pour un domaine évoluant au cours du temps

Le problème thermique est modélisé par une équation de la chaleur linéaire (4.1), dont la principale difficulté de résolution est liée au fait que le domaine de calcul évolue au cours du

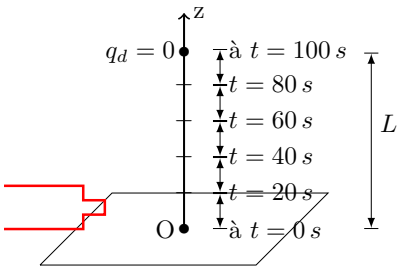


FIGURE 4.1 – Présentation du problème de la chaleur en 1D, en considérant 100 couches.

temps. Pour la suite, nous considérons $N = 100$ couches d'épaisseur $e = 5\,.10^{-3}$m chacune, en utilisant un maillage régulier. Ce modèle permet de déterminer le champ de température à tout instant, le temps nécessaire pour construire une couche est estimé à $20\,s$.

$$\begin{cases} \rho\, C_p \dfrac{\partial\, u}{\partial\, t}\,(x,t) + \kappa \Delta\, u(x,t) & = f(x,t) \\ u(x=0,t) & = 0 \\ u(x,t=0) & = 0 \end{cases} \tag{4.1}$$

Les pertes de chaleur lors de la fabrication sont modélisées dans un premier temps par la condition limite la plus simple, à savoir u nul en O. Avec $\Delta u(x,t) = \dfrac{\partial^2 u(x,t)}{\partial x^2}$ le Laplacien par rapport à la variable d'espace.

En jouant sur l'évolution des paramètres matériaux au cours du temps, nous mettons en évidence l'ajout de matière lors de la simulation. En effet, le maillage est réalisé sur la pièce globale, et afin de simuler les couches inactives comme illustré par la figure 4.2 on leur affecte des propriétés matériau dégradées. Par exemple, la conductivité thermique proche de zéro, garantit que la température des éléments inactifs ne change pas.

$$\kappa(x,t) = \begin{cases} \kappa_{acier} & si \quad x \leq \left(\dfrac{v\,t}{e_{couche}} + 1\right) \times e_{couche} \\ \kappa_{air} & sinon \end{cases}$$

Avec v représente la vitesse de la source laser, e_{couche} l'épaisseur de la couche active.

4.1.2 Modélisation de l'apport de chaleur

Afin de réduire la difficulté, l'apport de chaleur (cf. 2.3.2) doit être modélisée par un flux volumique qui reproduit fidèlement l'énergie transmise à la pièce. La source de chaleur est modélisée dans la suite par une fonction mobile dépendante de l'espace et du temps, elle est moins complexe que les sources de chaleur Gaussienne mentionnée au paragraphe 2.3.2.

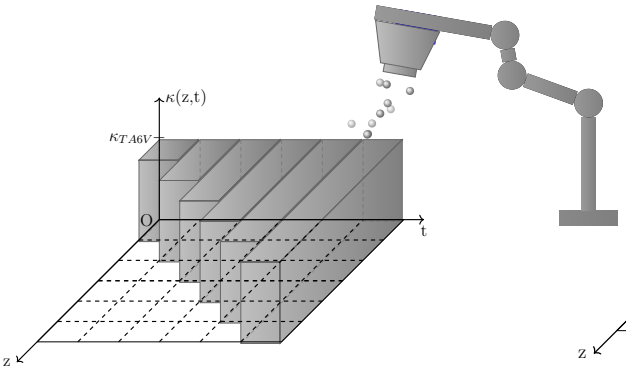


FIGURE 4.2 – Conductivité thermique caractérisant le comportement des matériaux en fonction du temps.

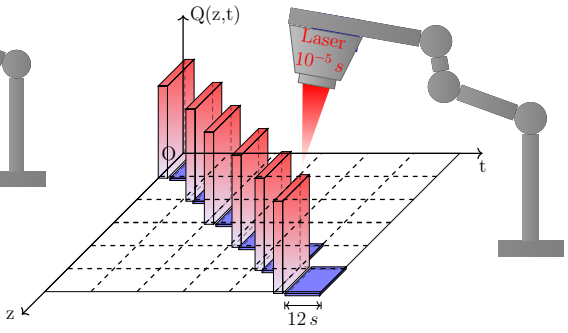


FIGURE 4.3 – Variation de l'apport de chaleur en fonction du temps.

Contrairement au diamètre du faisceau laser, la forme de la fonction représentant le flux de chaleur a peu d'influence sur la précision finale de la pièce, une étude approfondie est présentée par A. Longuet dans [72]. Parmi les paramètres qui influencent la précision on trouve : la puissance du laser Q, le nombre de couche n_{couche}, l'épaisseur d'une couche, la densité d'énergie, le temps de lasage et de refroidissement, et la vitesse du faisceau laser (figure 4.3). On chauffe une surface du lit de poudre équivalente à la taille du diamètre du faisceau laser, le temps de chauffe est de $10^{-5}s$ et le temps de refroidissement de $12s$ à chaque balayage du laser, le temps nécessaire pour parcourir chaque couche est $t_{couche} = 20s$.

$$f(x,t) = \begin{cases} \dfrac{Q}{n_{couche} \times t_{couche}} & si \quad \left(\dfrac{v\,t}{e_{couche}}\right) \times e_{couche} \leq x \leq \left(\dfrac{v\,t}{e_{couche}} + 1\right) \times e_{couche} \\ 0 \quad sinon \end{cases}$$

4.1.3 Formulation variationnelle du problème thermique

Soit $\Omega \subset \mathbb{R}$ un ouvert borné. $L^2(\Omega)$ l'espace de fonctions de carré intégrable. $\mathcal{D}(\Omega)$ est l'espace des fonctions indéfiniment dérivables à support compact. On introduit l'espace de Sobolev comme étant l'espace constitué des fonctions $v \in L^2(\Omega)$, dont la dérivée partielle au sens des distributions appartient à $L^2(\Omega)$. On note

$$H^1(\Omega) = W^{1,2}(\Omega) = \Big\{ v \in L^2(\Omega),\ \ \frac{\partial v}{\partial x} = v' \in L^2(\Omega) \Big\}.$$

On introduit l'espace des fonctions tests

$$V = H_0^1(\Omega) = \Big\{ v \in H^1(\Omega) \ \mid \ v_{|\Gamma_D} = 0 \Big\}.$$

La simulation d'un problème instationnaire nécessite en général de discrétiser la dérivée temporelle par une approximation de différence finies, ce qui donne un ensemble récursif de problèmes stationnaires puis transformer chaque problème stationnaire en une formulation variationnelle.

Soit Δt le pas de temps et notons u^n le vecteur inconnu au temps $t_n = n\,\Delta t$. On utilise une formule décentrée implicite

$$\frac{\partial u}{\partial t} = \frac{u^n - u^{n-1}}{\Delta t}$$

Pour écrire la formulation variationnelle du problème 4.1, on multiplie l'équation de la chaleur par une fonction test $v \in V$, et on intègre sur le domaine Ω :

$$\begin{cases} \text{Trouver } u \in H_0^1(\Omega) \text{ tel que :} \\ \rho\, C_p\, u\, v - \displaystyle\int_\Omega \kappa(x,t)\,\Delta t\,\Delta u\,.\,v = \int_\Omega (u^{n-1} + \Delta t\, f^n)\, v \qquad \forall v \in H_0^1(\Omega) \end{cases} \tag{4.2}$$

En appliquant la formule de Green :

$$\int_\Omega \Delta u\, v\, d\,x = \int_{\partial\,\Omega} \frac{\partial\, u}{\partial n}\, v\, \overrightarrow{n}\, d\sigma - \int_\Omega \nabla\, u \nabla\, v\, d\,x$$

On obtient alors :

$$\underbrace{\int_\Omega \rho\, C_p\, u\, v\, d\,x + \int_\Omega \kappa(x,t)\,\Delta t\, \nabla\, u\, :\, \nabla\, v\, d\,x}_{a(u,v)} = \underbrace{\int_\Omega (u^{n-1} + \Delta t\, f^n)\, v\, d\,x}_{l(v)} \tag{4.3}$$

Ce problème est bien défini s'il vérifie les conditions du théorème de Lax [1]-Milgram [2] :

Théorème 4.1
Soit V un espace de Hilbert, on considère le problème suivant :

$$\textit{Trouver } u \in V \;/\; a(u,v) = l(v)\ \forall v \in V. \tag{4.4}$$

Si a est une forme bilinéaire, symétrique, continue sur V :

$$\exists \beta > 0 \;/\; |a(u,v)| \leqslant \beta \|u\|_V \|v\|_V\ \forall (u,v) \in V \times V,$$

coercive sur V

$$\exists \alpha > 0 \;/\; a(u,u) \geqslant \alpha \|u\|_V^2\ \forall u \in V,$$

et l une forme linéaire, continue sur V,
alors il existe une solution et une seule au problème 4.4 qui dépend continûment des

1. Peter David Lax, né en Hongrie en 1926
2. Arthur Norton Milgram, américain, 1912-1961

données :

$$\exists C > 0 \ / \ \|u\|_V \leqslant C\|l\|_V, \ \forall l \in V'$$

Pour plus de détails, nous renvoyons le lecteur intéressé à [18, 26, 2].

4.1.4 Approximation de Galerkin

Soit $V_h \subset V$ un sous-espace de dimension finie tel que $V_h \underset{h\to 0}{\longrightarrow} V$. Soit $n = dim(V_h)$ et $[\phi_1, \cdots, \phi_n]$ une famille libre de V_h. La solution discrète u_h de V_h se décompose sur la base $[\phi_J]_{1\leqslant J\leqslant n}$ comme suit :

$$u_h(x) = \sum_{J=1}^{n} \mu_J \phi_J(x)$$

où les inconnues numériques du problèmes sont les coefficients $(\mu_1, \cdots, \mu_n)$ de u_h. Or, étant donné que $a(.,.)$ est bilinéaire et par linéarité en v_h, le problème discret devient :

$$\begin{cases} \text{Trouver } (\mu_1, \cdots, \mu_n) \in \mathbb{R}^n \text{ tel que :} \\ \qquad \sum_{J=1}^{n} \mu_J a(\phi_J, \phi_I) = l(\phi_I) \qquad \forall I \in [\![1, n]\!] \end{cases} \tag{4.5}$$

Notre problème se ramène donc à un problème d'algèbre linéaire à n équations à n inconnues. Étant donné $\mathbb{F}$, trouver le vecteur U tel que

$$\mathbb{A}U = \mathbb{F} \tag{4.6}$$

avec $\mathbb{A}$ une matrice telle que $\mathbb{A}_{IJ} = a(\phi_J, \phi_I) \in \mathcal{M}(n)$, U un vecteur de longueur n et dont la $i^{\text{ème}}$ composante est μ_i, et $\mathbb{F}$ le second membre déterminé par $l(\phi_I)$.

La simulation de ce problème par FEM nécessite la résolution du système linéaire 4.6. Pour diminuer le coût de cette résolution, on utilise les méthodes de réduction de modèle *a priori* ou *a posteriori* (cf. chapitre 3).

4.1.5 Problème de référence

À ce stade, on considère le problème composé de 5 macro-couches. On considère 100 couches d'épaisseur $5\,.10^{-3}$m, et le temps nécessaire pour parcourir chaque couche est de 20s. Dans ce premier exemple, on ne considère pas le temps de refroidissement. Le flux de chaleur est considéré par morceaux comme $Qmoy = \dfrac{3.42E14 \times 10^{-4}}{100 \times t_{couche}}$. On présente par la figure 4.4 les résultats FEM, qui serviront de références tout au long de ce chapitre.

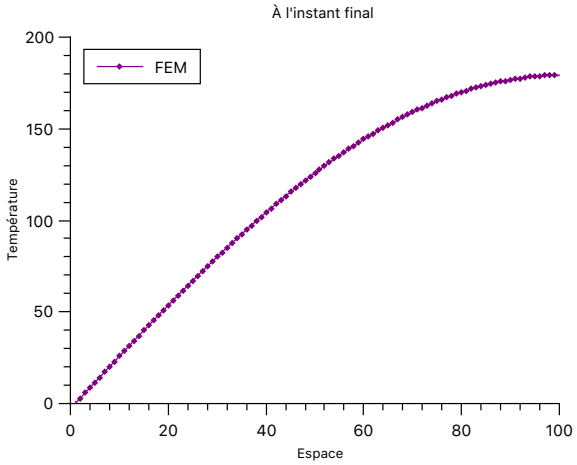


FIGURE 4.4 – Champ thermique illustré après construction des 100 couches.

4.2 Compression des données par séparation de variables

Avant de s'intéresser à la résolution en utilisant des modèles réduits, nous allons tout d'abord voir si le champ de température peut se réduire efficacement.

On considère les champs thermiques $u(x,t)$ supposés connus à l'aide de la simulation numérique par éléments finis. Comme sus-mentionné dans la section 3.2.1, on cherche à approximer le champ thermique sous forme d'une représentation à variables séparées appartenant à un espace de dimension finie (cf. 4.7). Soit la base spatiale $\omega = [\omega_1, \cdots, \omega_{N_x}]$ et temporelle $\varphi = [\varphi_1, \cdots, \varphi_{N_T}]$.

$$u(x,t) \simeq u_M(x,t) = \sum_{m=1}^{M} \omega_m(x)\,\varphi_m(t) \tag{4.7}$$

Cette représentation n'étant pas unique, on cherche donc à obtenir la meilleure approximation du champ thermique, tout en utilisant peu de modes. Théoriquement, on s'attend à ce que cette approximation devienne exacte lorsque le nombre de modes retenus "M" soit égal à la dimension de l'espace global.

Les champs représentés par l'équation 4.7, sont stockés dans une matrice de snapshots A en tout point de l'espace et du temps comme cité dans la section 3.5. Dans cette première simulation, nous allons évaluer l'application de la POD pour la simulation de ce procédé. Dans ce cas d'étude, on a deux variables à séparer à savoir le temps et l'espace, la POD est donc identique à l'application d'une SVD.

La décomposition en valeurs singulière introduite par l'équation 3.6, est représentée par la figure 4.5. La matrice diagonale Σ (*illustrée par la couleur cyan*) permet d'obtenir les valeurs singulières. On s'aperçoit que ces valeurs ont tendance à décroître, le nombre de

valeurs conséquentes correspond donc à la troncature de la POD. L'objectif est de garder uniquement les termes dont les valeurs sont plus grandes que la valeur singulière maximale $|\sigma_i| \geq \max_k |\sigma_k|$.

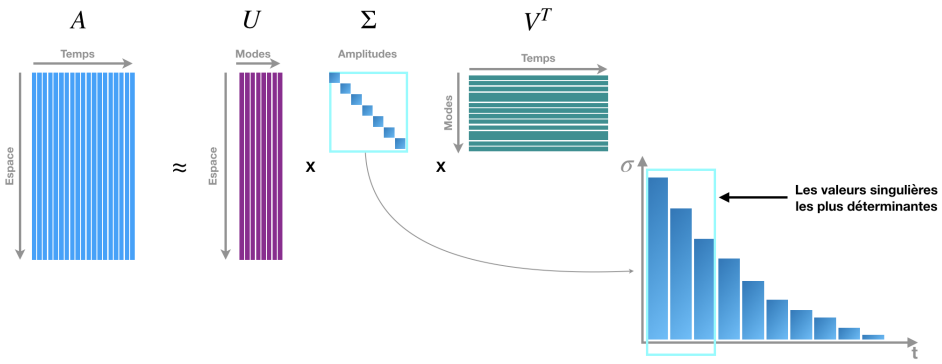


FIGURE 4.5 – Décomposition en valeurs singulières.

Soit $\Lambda = \Sigma V^T \in \mathbb{R}^{d \times N_T}$, avec $d < min(N_X, N_T)$. La solution du problème de minimisation 3.4 peut être considéré sous la forme suivante

$$u_h(x, t_\imath) = \sum_{j=1}^{d} U[:, j] \cdot \Lambda_{j\imath}^{d} \tag{4.8}$$

La matrice $U_{POD} \in \mathbb{R}^{N_X \times N_{POD}}, (N_{POD} \leq d)$ représente une base réduite pour le modèle 4.7, constituée des différentes valeurs singulières choisies. À partir de cette base, on cherche à déterminer la solution réduite Γ_{POD} avec $A_{reconst} \approx U_{POD} \cdot \Gamma_{POD}$. L'erreur entre la matrice initiale et sa reconstruction est calculée en utilisant la norme de Frobenius.

$$\varepsilon(k) = \frac{\|A - A_{reconst}\|_F}{\|A\|_F} = \left(\frac{\sum_{k=1}^{N_{POD}} \lambda_k}{\sum_{i=1}^{d} \lambda_i} \right)^{\frac{1}{2}} \quad \text{avec } \lambda_i = \sigma_i^2 \ , \forall i$$

L'algorithme POD utilisé dans la suite de ce travail est le suivant

Algorithme 1 : Séparation espace-temps en utilisant la SVD

fonction POD (x, t);
Données : La matrice de snapshots $A \in \mathbb{R}^{N_X \times N_T}$
Sorties : $U \in \mathbb{R}^{N_X \times N_{POD}}$ et $\Gamma \in \mathbb{R}^{N_{POD} \times N_T}$.

Initialisation Définir la matrice de snapshots A à partir du calcul FEM ; $A = U\Sigma V^T$
for $i = 1$ *to* $rank(A)$ **do**
  if $\lambda_i < \lambda_{max}$ **then**
    $N_{POD} = i - 1$;
    break;
  end
end
$U \longleftarrow U[:, : N_{POD}]$;
$\Gamma \longleftarrow (U^T U)^{-1} U^T A$;

4.2.1 Stratégies de compression

Nous allons tester plusieurs manières pour construire la matrice des snpashots dans le but de déterminer une méthode de réduction optimale. Dans un premier temps, une approche directe sera étudiée. Ensuite, et dans le but de tirer parti de la POD, l'idée est de partitionner l'espace grâce à un changement de variable en espace. Deux types de changement de variables seront comparés.

4.2.1.1 Approche directe

En reprenons la problème thermique décrit dans le paragraphe 4.1.3. À chaque dépôt d'une nouvelle couche, on stocke uniquement la partie acier (macro-couches construites) dans la matrice des snapshots comme illustré par la figure 4.6. Puis, on applique l'algorithme 1.

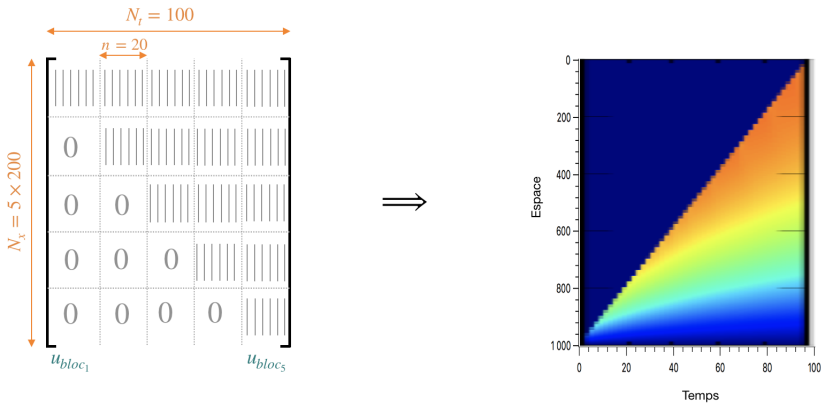


FIGURE 4.6 – Représentation schématique de la matrice de snapshot.

La figure 4.7(a), représente les valeurs du champ thermique à l'extrémité du domaine après

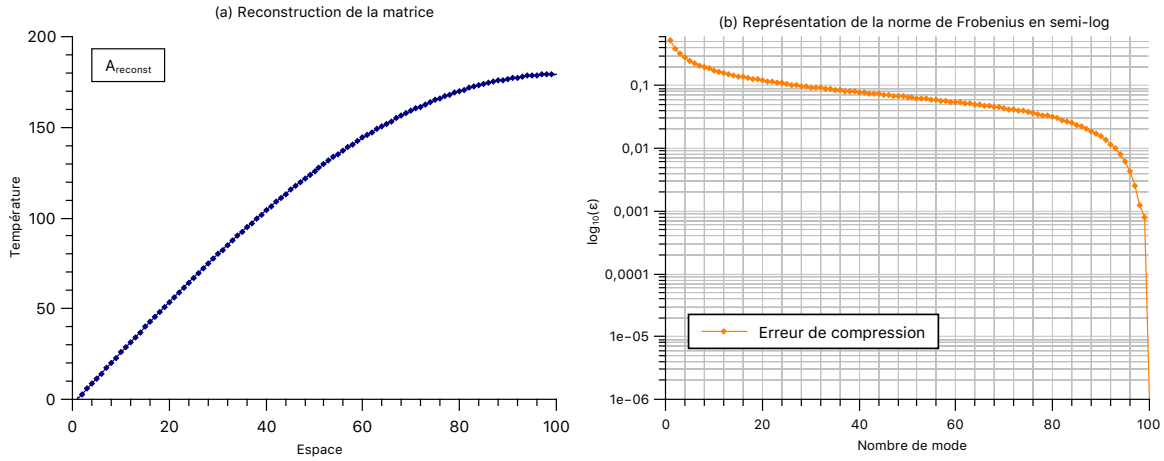


FIGURE 4.7 – Calcul de l'erreur entre la matrice initiale et sa reconstruction en utilisant la norme de Frobenius.

reconstruction. L'erreur entre la matrice initiale et sa reconstruction est calculée en utilisant la norme de Frobenius et représenté par la figure 4.7(b).

L'application de l'approche directe nécessite de recalculer presque la totalité du problème afin de reconstruire une solution précise. Cette méthode ne fournit donc pas de bons résultats en matière de compression.

4.2.1.2 Compression en utilisant un changement de variables en espace

L'efficacité d'un modèle réduit dépend du nombre de modes dans la base réduite, et de la méthodologie utilisée pour déterminer ces modes. La figure 4.8 représente les changements de variables en espace que nous allons réaliser. Dans un premier temps, nous allons utiliser pour la résolution du problème FEM un maillage uniforme variant entre 0 et 1 (cf. 4.8 (a)). Sur la figure 4.8 (b) on représente une deuxième méthode qui sera discutée dans le paragraphe suivant.

On peut maintenant construire un maillage dans l'intervalle $[0, 1]$, en définissant une subdivision :

$$0 = z_0 < z_1 < z_2 < \cdots < z_{nn} = 1$$

On note $h = \dfrac{1}{nn-1}$ le pas du maillage et on définit l'espace V_h, sous-espace de dimension finie et on a $V_h \subset H_0^1([0,1])$ défini par :

$$V_h = \left\{ v_h \in \mathcal{C}^0([0,1]),\ \ v_h|_{[z_\imath, z_{\imath+1}]} = a_\imath + b_\imath z \quad (a_\imath, b_\imath) \in \mathbb{R}^2 \quad \forall 0 \leqslant \imath \leqslant nn,\ \ v_h(0) = 0 \right\}$$

Remarquons que V_h est entièrement déterminée par ses valeurs $z_1, \cdots, z_{nn}$, on établit que la dimension de V_h est nn, et qu'une base de V_h est $[\phi_1, \cdots, \phi_{nn}]$, où les fonctions de base sont

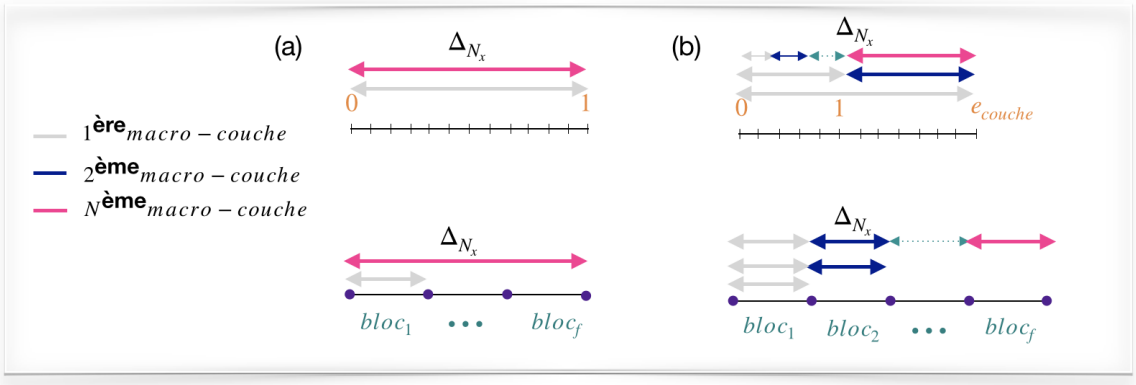


FIGURE 4.8 – Application d'un changement de variable pour la projection du modèle sur un nouveau maillage, (a) projection des macro-couches actives sur un maillage variant de 0 à 1, (b) projection des macro-couches antérieures sur un maillage variant de 0 à 1 et de la macro-couche active sur un domaine variant entre 1 et e_{couche}.

définies si le maillage est uniforme par $z = \phi_i(\zeta) = ie\,\zeta$. La dérivée dans l'élément $[0,1]$ n'est pas égale à la dérivée dans l'élément de référence $[z_{\imath-1}, z_\imath]$. Il faut tenir compte du changement de variable, on pose $dz = ie\,d\zeta$. On passe donc d'un calcul de $z \in [0, i \times e]$ à une intégration sur l'intervalle $[0,1]$:

$$\int_{\Omega_{z_1}\times\cdots\times\Omega_{z_i}} f(z)dz \longrightarrow \int_{\phi_i(0)=0}^{\phi_i(1)=i\times e} f(z)dz = \int_0^1 f(\phi_i(\zeta))\frac{d\phi_i(\zeta)}{d\zeta}d\zeta$$

$$\Longrightarrow \int_0^{i\times e} f(z)dz = ie\int_0^1 f(\phi_i(\zeta))d\zeta$$

Tout l'intérêt de cette méthode réside dans le fait que chaque fonction de base $\phi_\imath(\zeta)$ a un support très réduit, c'est-à-dire l'ensemble ζ tels que $\phi_\imath(\zeta) \neq 0$ est petit devant le domaine de résolution. La variable z est approximée au sens des éléments finis par :

$$ie\int_0^1 N_I(\phi_i(\zeta))N_J(\phi_i(\zeta))d\zeta$$

Nous insérons $v = \hat{\phi}_\imath$ et $u^n = \sum_{\jmath=1}^{M} U_\jmath^n\,\phi_\jmath$ dans l'équation (4.3), on obtient :

$$\mathbb{A} = \rho\,C_p\sum_{\jmath=1}^{M}\left(\left(\int_\Omega \hat{\phi}_\imath\,\phi_\jmath\,d\,x\right)U_j^n + \kappa(x,t)\,\Delta\,t\left(\int_\Omega \nabla\,\hat{\phi}_\imath\,\nabla\,\phi_\jmath\,d\,x\right)U_j^n\right)$$

On définit les fonctions de base en fonction des fonctions de forme par

$$N_I^-(z) = \frac{1}{h_i}z - (I-2) \qquad avec \quad h_i = \frac{ie}{nn-1}$$

Où h_i représente le pas du maillage de la couche active, $\imath$ définit la macro-couche active, e l'épaisseur de la couche, et nn discrétisation spatiale.

$$N_I^-(\phi_i(\zeta)) = \frac{nn-1}{ie} ie\zeta - (I-2) = (nn-1)\zeta - (I-2)$$

En effectuant le changement de variable dans l'intégrale (4.3), on obtient :

$$\rho\, C_p \sum_{\jmath=1}^{M} \left(\left(ie \int_0^1 N_I(\phi_i(\zeta)) N_J(\phi_i(\zeta)) d\zeta \right) U_j^n + \kappa\, \Delta\, t \left(ie \int_0^1 \nabla_z\, N_I(\phi_i(\zeta))\, \nabla_z\, N_J(\phi_i(\zeta))\, d\zeta \right) U_j^n \right)$$

On cherche à représenter cette formulation uniquement en fonction de ζ. On note $\overline{h} = \dfrac{1}{nn-1}$, les fonctions de formes sont définies comme suit

$$\bar{N}_I^-(\zeta) = \frac{1}{\overline{h}}\zeta - (I-2), \qquad \bar{N}_I^+(\zeta) = \frac{-1}{\overline{h}}\zeta + I$$

On définit alors

$$\rho\, C_p \sum_{\jmath=1}^{M} \left(\left(ie \int_0^1 \bar{N}_I(\zeta) \bar{N}_J(\zeta) d\zeta \right) U_j^n + \kappa\, \Delta\, t \left(\frac{1}{ie} \int_0^1 \bar{N}_{I,\zeta}(\zeta)\, \bar{N}_{J,\zeta}(\zeta)\, d\zeta \right) U_j^n \right) \tag{4.9}$$

De la même manière on détermine le second membre, ensuite on compare les résultats. La résolution de ce nouveau système est réalisé sur le nouveau domaine d'étude. Dans ce cas, l'allure de la matrice de snapshot est représentée par la figure 4.9.

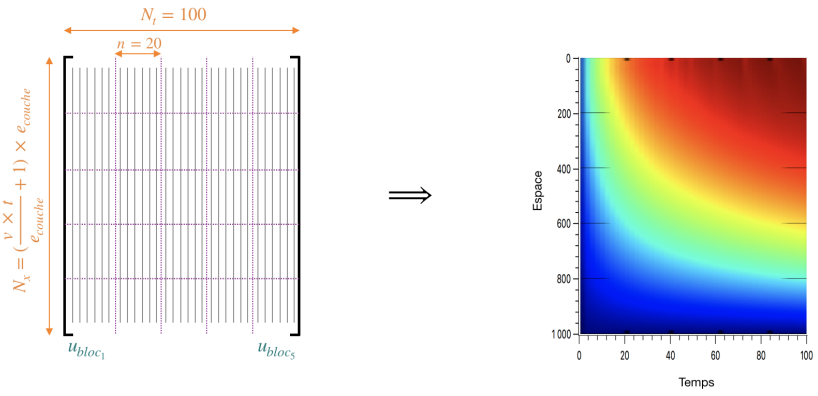


FIGURE 4.9 – Représentation schématique de la matrice de snapshot.

L'erreur calculée entre la matrice initiale des champs solution de la simulation numérique et sa reconstruction est représentée en échelle logarithmique par la courbe bleue (cf. figure 4.10). Cette approche nous permet d'obtenir une reconstruction de la matrice initiale en utilisant moins de mode. En effet, comme l'illustre les graphes de la figure 4.10, la courbe bleue converge bien plus rapidement.

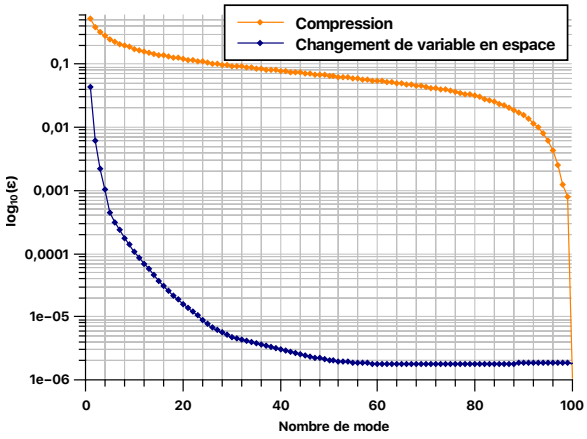


FIGURE 4.10 – Représentation de l'erreur calculée après compresion en orange et celle calculée en utilisant la deuxième approche en bleu lors du même mode.

4.2.1.3 Compression en utilisant un changement de variables à deux domaines

Dans cette deuxième méthode (cf. figure 4.8 (b)), le problème est résolu sur un maillage variant en fonction de la hauteur de la couche fabriquée. Cette méthode est basée sur le calcul introduit dans la méthode précédente avec comme particularité le calcul de la couche active à un instant t donné s'effectue sur l'intervalle $[1, e_{couche}]$, et le calcul des couches sous-jacentes au même instant t est calculé sur l'intervalle $[0, 1]$.

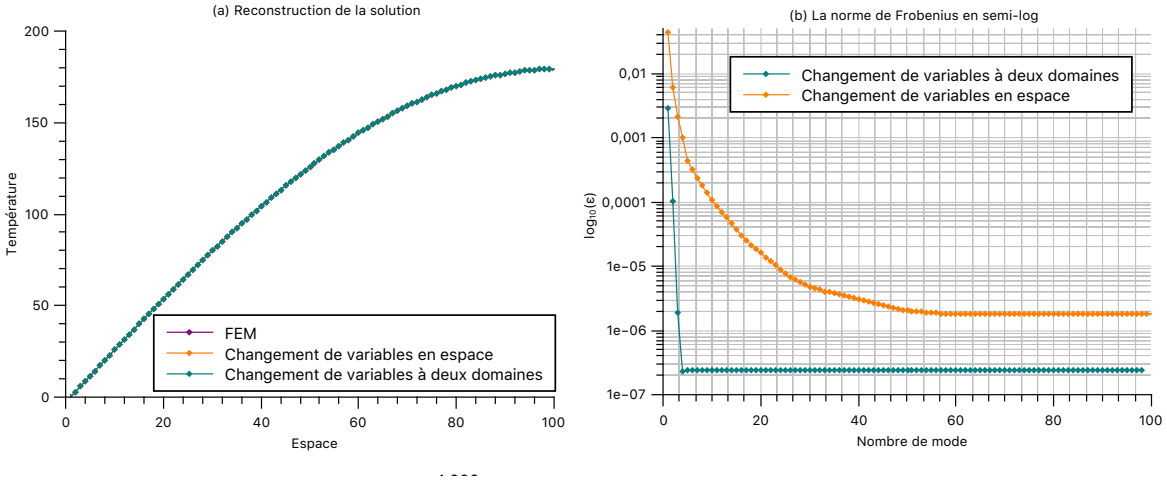


FIGURE 4.11 – Résultats en utilisant le changement de variables.

Cette nouvelle approche donne de très bons résultats comme illustré figure 4.11. En effet, seulement 6 modes sont nécessaires pour représenter la solution.

Nous avons vu qu'un changement de variable a une grande influence sur la compression espace-temps du champ. Nous allons nous intéresser notamment à la résolution du même

problème par une approche APR.

4.3 Résolution du problème par APR

Dans le but d'accélérer le temps de calcul en utilisant la POD, on propose d'utiliser la méthode APR qui se décompose en un calcul *off-line* et un calcul *on-line*. Cette méthode est proposée par de nombreux auteurs pour la simulation de différents phénomènes physiques en raison du fait qu'elle permet de capter l'information utile à partir des résultats du calcul précédent.

Lors de la simulation des problèmes non-linéaires dépendant du temps D. Ryckelynck présente dans [93] une approche *a priori* basée sur l'exploitation des sous-espaces de Krylov. La méthode de réduction proposée utilise la décomposition Karhunen-Loève 3.2 afin d'extraire les modes les plus significatifs. Dans cet article, l'auteur propose une approche non-incrémentale basée sur la construction d'une base initiale à partir du résidu des équations du problème en FEM. Des étapes d'enrichissement permettent d'obtenir une représentation du modèle réduit. Dans [92], D. Ryckelynck propose une deuxième approche connue sous le nom APHR (*A Priori Hyper Reduction*) obtenue à partir de points d'intégration du modèle FEM, implémentée en utilisant un algorithme incrémental. Cette approche s'est avérée être un avantage considérable en termes de temps de calcul.

Verdon et al. [108] utilisent cette méthode afin d'évaluer les champs de températures pour la simulation d'une équation de convection-diffusion en 2D, ainsi que la simulation de l'équation 1D de Burgers. Dans cet article, la méthode APR est comparée à la décomposition Karhunen-Loève 3.2. Les auteurs abordent les temps de calcul nécessaires pour la résolution des problèmes réduits que ce soit en 1D ou 2D, tout en gardant la précision recherchée. Les auteurs indiquent d'ailleurs que l'approche APR représente un moyen efficace de corriger le comportement à long terme des systèmes dynamiques d'ordre inférieur. C. Allery et al. [9] détaillent la méthode APR tel que nous l'utiliserons dans ce qui suit, et l'appliquent pour déterminer la solution approchée des équations bidimensionnelles de Burgers. Les auteurs présentent une comparaison entre l'approche APR, les résultats obtenus en utilisant l'algorithme Newton-Raphson, et les résultats exposés dans la littérature. Dans ce travail, l'approche APR permet selon les auteurs d'importantes économies en temps de calcul, tout en conservant la précision souhaitée.

L'utilisation de la méthode APR nécessite une initialisation de la base, la solution converge grâce aux améliorations effectuées à chaque apport de matière. Le choix de cette base détermine le nombre d'itérations du calcul APR pour faire converger la solution [110]. Pour plus de détails sur cette méthode, nous envoyons le lecteur intéressé à [111].

Le nouveau problème projeté dans la méthode APR s'écrit sous forme

$$\begin{cases} \text{Trouver } U_h(x,t) \text{ tel que :} \\ \rho\, C_p \dfrac{\partial U_h}{\partial t}(x,t) + \nabla\Big(\kappa(x,t)\cdot\nabla U_h(x,t)\Big) = f(x,t) \\ \qquad\qquad\qquad\qquad \text{Avec } U_h = \displaystyle\sum_{m=1}^{M} \omega_{m,h}(x)\,\varphi_{m,h}(t). \end{cases} \tag{4.10}$$

Où l'exposant m détermine l'indice du mode.

En pratique, la POD permet de conserver les informations les plus importantes du modèle complet, ce qui permet de déterminer une base spatiale qui dans notre cas représente l'initialisation pour le calcul des modes temporels compatibles avec les modes spatiaux.

4.3.1 APR appliquée à un domaine défini par ajout de couches

La méthode APR est une méthode de réduction de modèle itérative qui n'exploite pas les champs résultant de la FEM, mais qui est plutôt basée sur l'enrichissement d'une base initialement déterminée en utilisant la POD.

On cherche à résoudre notre problème thermique en divisant le domaine d'étude en 5 blocs. Chaque bloc est composé de 20 micro-couches (layer). On applique la POD à 20% de chaque sous-domaine (figure 4.12) et on essaye de reconstruire le reste (les autres layer du même bloc) de la simulation à partir des modes résultants de la POD. Des améliorations itératives sont effectuées au niveau des interfaces.

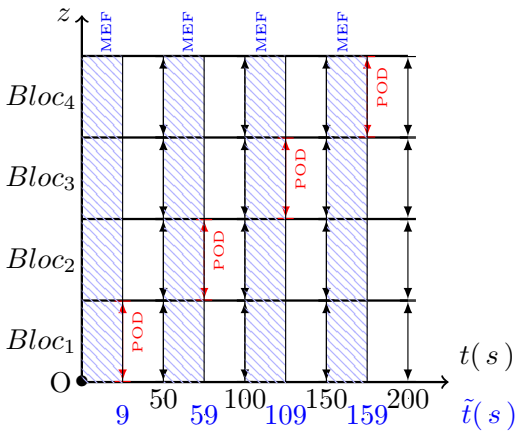


FIGURE 4.12 – Utilisation de la POD pour accélérer les calculs. Les champs bleus sont calculés par éléments finis, et les autres reconstruits à l'aide de la POD.

Afin de raccorder les différentes couches les unes aux autres nous appliquons la méthode de Nitsche.

La méthode de Nitsche [14] est donc une méthode qui permet d'assurer la continuité au niveau des interfaces, et d'une certaine façon transférer les champs de température entre les différentes couches.

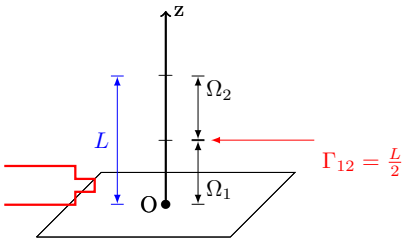


FIGURE 4.13 – Présentation de l'interface entre deux sous-domaines.

En considérant un cas simple dans lequel on simule la conception de deux couches, représenté par la figure 4.13 par les domaines Ω_1 et Ω_2, ces domaines sont raccordés par l'interface Γ_{12} qui permet d'assurer la continuité de la température.

On commence par mailler chaque $\Omega_i, i = 1, 2$. Soit $\mathcal{T}_h$ ce maillage, K^i un élément courant du maillage et V_h^i un espace conforme dans $H^1(\Omega^i)$. On choisit pour chaque V_h^i l'espace des fonctions continues sur $\mathcal{T}_h^i$ polynomiales par morceaux de degré inférieur ou égale à 1 sur chaque élément K^i.

$$V_h^i := \left\{ v_h \in \mathcal{C}^0(\Omega^i), \;\; v_{h|K^i} \in \mathbb{P}_1(K^i) \qquad \forall K^i \in \mathcal{T}_h^i \right\}$$

On cherche $u_h^i = (u_h^1, u_h^2)$ dans $V_h^i = (V_h^1, V_h^2)$

$$\begin{cases} \rho\, C_p \dfrac{\partial\, u^i}{\partial\, t}\,(x,t) + \kappa(x,t)\,\Delta\, u^i(x,t) & = f(x,t) \qquad\qquad \forall x \in [0,L], \quad i = 1,2 \\ u^1(0) & = u_0 \\ u^2(L) & = 0 \end{cases} \tag{4.11}$$

À l'interface Γ_{12} on a $u^1 = u^2$.

Les conditions aux limites de Dirichlet sont par la suite traitées de la même manière que dans le cas classique (cf. 4.11).

On considère une fonction test $v^i \in V_h^i$, et on intègre sur le domaine Ω^i ($\Omega^1 = [0, \dfrac{L}{2}]$ et $\Omega^2 = [\dfrac{L}{2}, L]$). Puis par intégration par parties, on diminue la régularité demandée sur la fonction solution. Ensuite, on symétrise, et on stabilise pour s'assurer de la coercivité discrète uniforme [13, 33]

$$\begin{aligned}\int_\Omega \rho\, C_p \frac{\partial\, u}{\partial\, t}\, d\, x \,+ \int_\Omega \kappa\, \nabla\, u\, \nabla\, v\, d\, x - \int_{\partial\Omega} \frac{\partial u}{\partial n} v ds - \int_{\partial\Omega} \frac{\partial v}{\partial n} u ds + \gamma \int_{\partial\Omega} uv ds = \\ = \int_\Omega f\, v\, dx - \int_{\partial\Omega} \frac{\partial u}{\partial n} v ds - \int_{\partial\Omega} \frac{\partial v}{\partial n} u ds + \gamma \int_{\partial\Omega} uv ds\end{aligned} \tag{4.12}$$

En utilisant un schéma décentré implicite, la forme faible 4.12 s'écrit pour chaque sous-domaine sous la forme :

$$\begin{aligned}\int_{\Omega^1} \rho\, C_p \frac{u^{1\,n} - u^{1\,n-1}}{\Delta\, t}\, d\, x \quad &+ \quad \int_{\Omega^1} \kappa(x,t)\, \nabla\, u^1 \,:\, \nabla\, v^1\, d\, x = \\ &= \quad \int_{\Gamma_0} \Big[(\delta_n\, u^1)\, v^1 \,+ (u^1 - u_0)\, (\delta_n\, v^1) \,+ \frac{\gamma}{h}\, (u^1 \,-\, u_0)\, v^1 \Big] \\ &+ \quad \int_{\Gamma_{12}} \Big[(\delta_n\, u^1)\, v^1 \,+ (u^1 - u^2)\, (\delta_n\, v^1) \,+ \frac{\gamma}{h}\, (u^1 \,-\, u^2)\, v^1 \Big] \\ &+ \quad \int_{\Omega^1} f^{1\,n}\, v^1\, d\, x\end{aligned}$$

$$\begin{aligned}\int_{\Omega^2} \rho\, C_p \frac{u^{2\,n} - u^{2\,n-1}}{\Delta\, t}\, d\, x \quad &+ \quad \int_{\Omega^2} \kappa(x,t)\, \nabla\, u^2 \,:\, \nabla\, v^2\, d\, x = \\ &= \quad \int_{\Gamma_{12}} \Big[(\delta_n\, u^2)\, v^2 \,+ (u^2 - u^1)\, (\delta_n\, v^2) \,+ \frac{\gamma}{h}\, (u^2 \,-\, u^1)\, v^2 \Big] \\ &+ \quad \int_{\Omega^2} f^{2\,n}\, v^2\, d\, x\end{aligned}$$

avec $\gamma = O(10)$ paramètre de stabilisation.

La première étape est de vérifier la consistance 4.1, la stabilité 4.2 et la coercivité 4.1 du modèle sus-mentionnés, Hansbo et al. [44, 45] ont détaillé les preuves de ces principales propriétés.

Définition 4.1
Le schéma numérique $u^{i\,n}$ est consistant à l'ordre p en temps et à l'ordre q en espace si la différence entre le schéma numérique et la solution exacte pris en chacun des termes qu'il approche tend vers 0.

Définition 4.2
Un schéma est dit stable (au sens L^2) s'il existe un réel strictement positif C, indépendant de Δx et de Δt, tel que

$$\|u^{n\,i}\| \leq C \|u^0\|$$

avec $u^{n\,i}$ définie sur Ω^i par $u_j^{n\,i}$, $x_j \leq x \leq x_{j+1}$ et u^0 la fonction constante définie à partir de la condition initiale, le rapport $\dfrac{\Delta\, t}{\Delta\, x}$ restant fixe.

Lemme 4.1
Soit V_h un espace discret de Hilbert, on considère le problème suivant

$$Trouver \;\; u_h \in V_h \;\; / \;\; a(u_h, v_h) = l(v_h) \quad \forall v_h \in V_h$$

La forme discrète $a(\cdot,\cdot)$ est coercive sur V_h

$$\exists \beta > 0 \;\; / \;\; a(u_h, u_h) \geq \beta |||u_h|||^2 \quad \forall u_h \in V_h$$

à condition que γ soit pris suffisamment grand. On a aussi continuité

$$\exists \beta > 0 \;\; / \;\; a(u_h, v_h) \leq \beta |||u_h||| \cdot |||v_h||| \quad \forall (u_h, v_h) \in V_h \times V_h$$

Pour plus de détails nous renvoyons le lecteur intéressé à [13].

La formulation faible peut être représentée sous la forme matricielle suivante [20]

$$\begin{pmatrix} M_1 \dfrac{U_1 - U_1^{n-1}}{\Delta t} \\ M_2 \dfrac{U_2 - U_2^{n-1}}{\Delta t} \end{pmatrix} + \left\{ \begin{pmatrix} K_1 & 0 \\ 0 & K_2 \end{pmatrix} + N_l \right\} \times \begin{pmatrix} U_1 \\ U_2 \end{pmatrix} = \begin{pmatrix} M_1 F_1 \\ M_2 F_2 \end{pmatrix} + N_r \tag{4.13}$$

avec $N_l =$

$$\left(\begin{array}{cccc|cccc} \gamma - \frac{2}{h} & \frac{1}{h} & & & & & \frac{-1}{h} & -\gamma + \frac{2}{h} \\ \frac{1}{h} & & & & & & & \frac{-1}{h} \\ & & 0 & & & 0 & & \\ & & & 0 & 0 & & & \\ \hline & & & 0 & 0 & & & \\ & & 0 & & & 0 & & \\ \frac{-1}{h} & & & & & & & \frac{1}{h} \\ -\gamma + \frac{2}{h} & \frac{-1}{h} & & & & & \frac{1}{h} & \gamma - \frac{2}{h} \end{array}\right)$$

$$N_r = \begin{pmatrix} u_0 \left(\gamma - \frac{1}{h}\right) \\ \frac{u_0}{h} \\ \vdots \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

Le système linéaire 4.6, peut être introduit en utilisant la matrice de masse M, et la matrice de raideur K.

$$(M \; + \; \Delta t \, K) \, U^n = M \, U^{n-1} + \Delta t \, M \, F^n$$

Donc le nouveau système linéaire à simuler s'écrit indépendamment des différents domaines sous la forme

$$P^T \, (\, M \; + \; \Delta t \, K \,) \, P \, \Lambda^n = P^T \, M \, P \, \Lambda^{n-1} \; + \; \Delta t \, P^T \, M \, F^n \tag{4.14}$$

avec

$$\mathbf{U^{nT}} \, U = \mathbf{U^{nT}} \, \mathbf{U^n} \, . \, \Lambda \tag{4.15}$$

La reconstruction de la solution s'écrit sous forme

$$u^n = \mathbf{U^n}\, \Lambda^n \tag{4.16}$$

$$\Rightarrow \underbrace{\begin{pmatrix} u^0 \\ u^1 \\ \vdots \\ u^{n_l-1} \\ u^{n_l} \end{pmatrix}}_{u} = \underbrace{\begin{pmatrix} \mathbf{U^0} & & & & \\ & \mathbf{U^1} & & & \\ & & \ddots & & \\ & & & \mathbf{U^{n_l-1}} & \\ & & & & \mathbf{U^{n_l}} \end{pmatrix}}_{P} \underbrace{\begin{pmatrix} \Lambda^0 \\ \Lambda^1 \\ \vdots \\ \Lambda^{n_l-1} \\ \Lambda^{n_l} \end{pmatrix}}_{\Lambda} \tag{4.17}$$

La matrice P est supposée connue à partir des modes spatiaux récupérés du calcul POD, la méthode APR consiste donc à calculer les modes temporels Λ compatibles avec ces modes spatiaux.

4.3.2 Vérification par simulation numérique

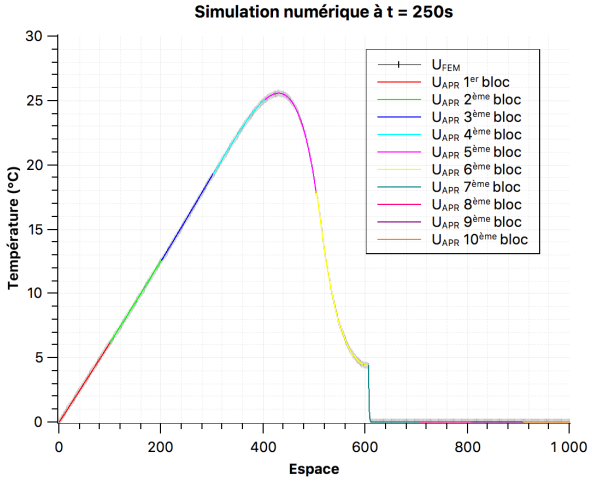
On représente par la figure 4.13, les valeurs du champ thermique en FEM et en APR à différents instants de la simulation, dans l'exemple sont représentés 10 domaines. L'espace est discrétisé en 100 intervalles. Le temps nécessaire pour parcourir chaque domaine est de $50\,s$, et le pas de temps utilisé lors de cette simulation est de $10^{-1}\,s$. La densité d'énergie est plus faible que dans le problème précédent. Néanmoins, on garde les mêmes temps de chauffe $10^{-5}\,s$ et de refroidissement $12\,s$.

Les figures 4.14a et 4.13b représentent la simulation du problème thermique respectivement à l'instant $t = 250$ et $t = 500$, la théorie prévoit une convergence des valeurs obtenues en FEM (tracé noir) et les approximations par la méthode APR.

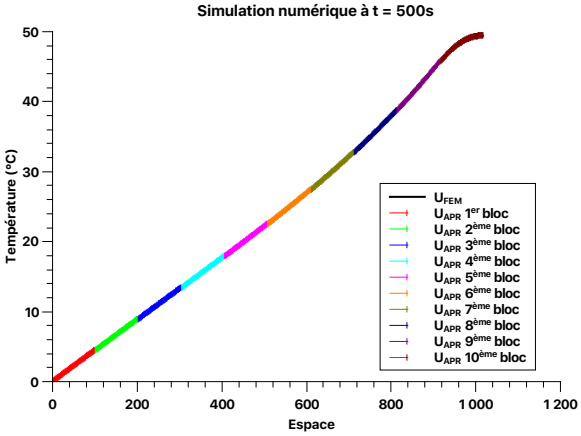
La différence entre ces deux solutions à l'instant final est représentée par la figure 4.13c en échelle semi-logarithmique. Le tableau 4.1 représente un récapitulatif du nombre de mode retenu afin de construire la base initiale de l'APR. Pour chaque macro-couche (bloc) on calcule 9 micro-couches en utilisant la POD, le nombre de mode retenu varie selon le critère de convergence prédéfini par l'utilisateur. En moyenne, 4 modes suffisent pour obtenir une bonne base néanmoins, comme le montre la figure 4.13c l'approximation est meilleure quand le nombre de mode est élevé.

En reprenant les équations 4.16 et 4.15, on détermine les modes temporels en résolvant le problème $\Lambda^n = \mathbf{U^{nT}}\, U \,/\, \mathbf{U^{nT}}\, \mathbf{U^n}$.

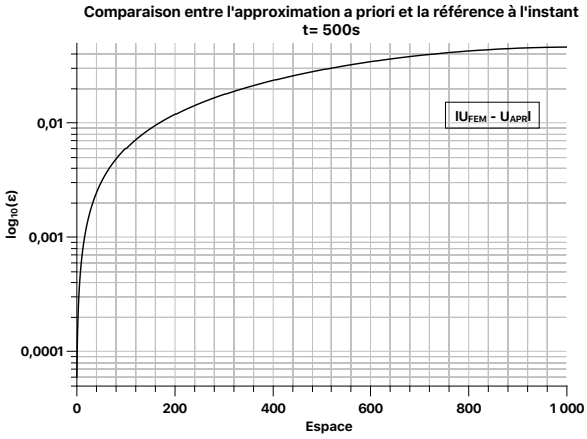
En considérant le résidu défini comme suit $\mathcal{R} = \mathbf{U^n}\, \Lambda^n - U$, la figure 4.14 représente le calcul de la norme du résidu en échelle semi-logarithmique et ce, pour les différentes macro-couches calculées. Les différentes figures permettent d'évaluer la précision de la reconstruction en fonction du nombre d'incrément spatial calculé.



(a) Simulation à t = 250



(b) Simulation à t = 500



(c) Calcul de la différence à t = 500

FIGURE 4.13 – Utilisation de la méthode de Nitsche.

	Nombre de modes									
	$Bloc_1$	$Bloc_2$	$Bloc_3$	$Bloc_4$	$Bloc_5$	$Bloc_6$	$Bloc_7$	$Bloc_8$	$Bloc_9$	$Bloc_{10}$
$Layer_1$	6	6	4	4	4	3	3	3	3	3
$Layer_2$	9	7	4	4	4	4	4	4	4	3
$Layer_3$	3	9	4	4	4	4	4	4	4	4
$Layer_4$	3	10	4	4	4	4	4	4	4	4
$Layer_5$	2	10	4	4	4	4	4	4	4	4
$Layer_6$	1	10	4	4	4	4	4	4	4	4
$Layer_7$	1	10	4	4	4	4	4	4	4	4
$Layer_8$	1	10	4	4	4	4	4	4	4	4
$Layer_9$	1	3	4	4	4	4	4	4	4	4

TABLE 4.1 – Récapitulatif du nombre de mode retenu pour chaque macro-couche.

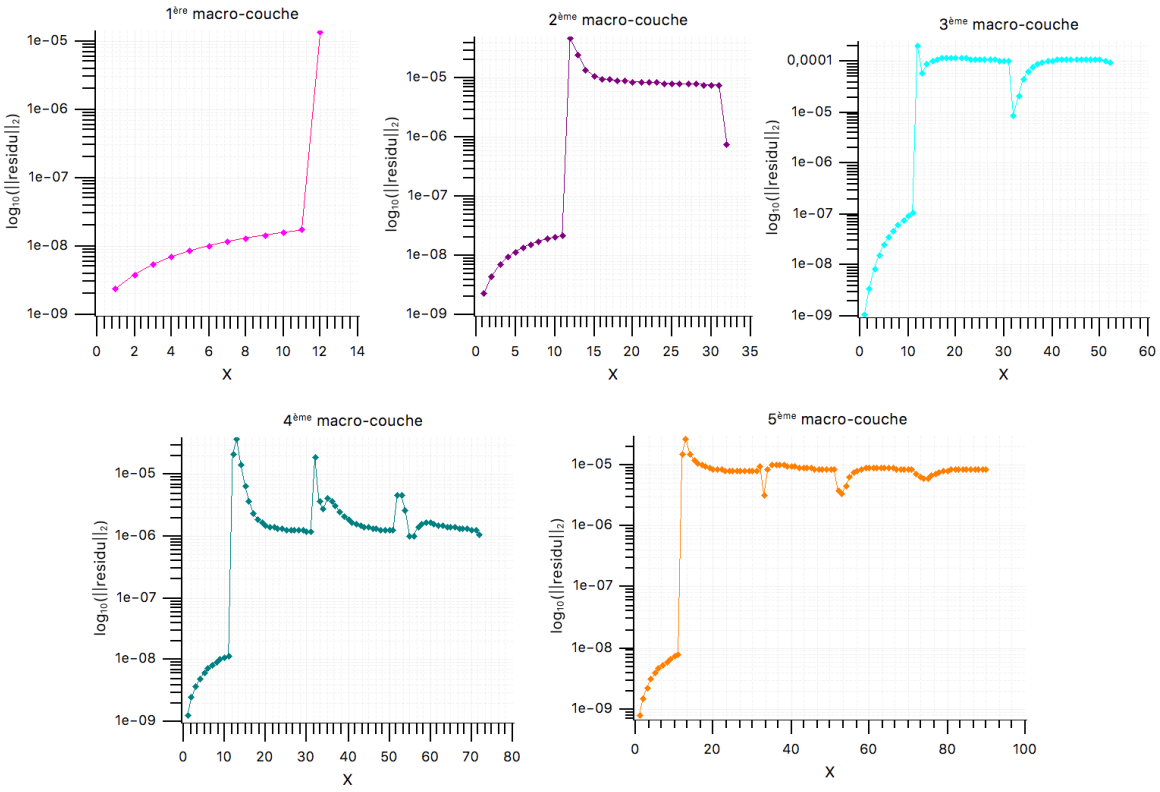


FIGURE 4.14 – Erreur de projection en fonction de la macro-couche active.

4.3.3 Résolution sur un domaine spatial réduit

Dans cette section, on projette le nouveau système linéaire défini par la méthode APR (cf. eq.4.14) sur un domaine réduit en utilisant un changement de variables en espace. Lors de la construction d'une nouvelle macro-couche couche, le système linéaire défini par l'équation 4.14 est projeté et résolu dans un intervalle variant entre 0 et 1 (cf. 4.2.1.2).

La figure 4.15(a) représente les champs de températures pour un domaine avec 5 couches en FEM et en APR résolus dans l'intervalle $[0, 1]$. Le nombre de modes nécessaires pour obtenir convergence entre les deux méthodes est représenté dans la figure 4.15(b). On remarque donc que très peu de données sont nécessaires dans ce cas de figure. Ce qui confirme l'importance du changement de variable.

4.4 Conclusion

Dans ce chapitre, nous nous sommes intéressés dans un premier temps à la mise en place du problème thermique de référence donné en 4.1.1. Afin de garder uniquement les modes énergétiquement optimaux, l'efficacité de la POD est étudiée et améliorée en utilisant un changement de variables (cf. 4.2.1.2, 4.2.1.3) sur un nouveau domaine plutôt qu'une approche

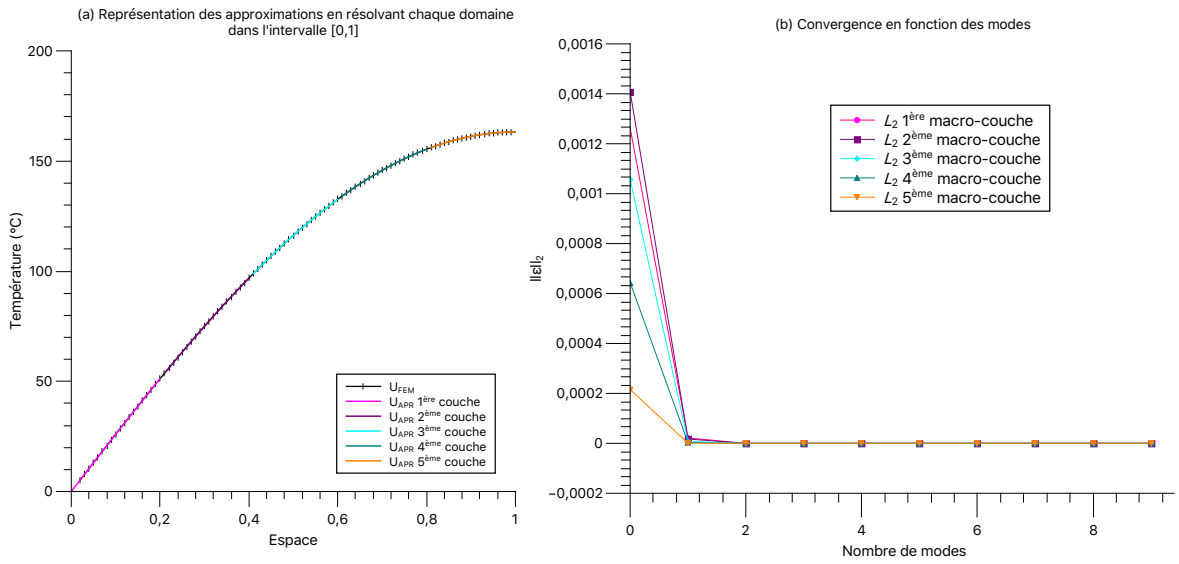


FIGURE 4.15 – Résultats en utilisant le changement de variables.

directe 4.2.1.1.

Dans le cas de la résolution, on introduit la méthode APR dont l'objectif est de calculer une approximation du champ thermique sans un calcul complet préalable. On s'aperçoit que très peu de modes sont nécessaires pour reproduire les résultats en FEM. Comme le montre la figure 4.15, la solution obtenue en traitant la méthode APR sur l'intervalle $[0, 1]$ converge plus rapidement vers la solution FEM du problème équivalent.

La méthode APR demande de calculer une partie de la solution complète en FEM. Dans le chapitre suivant, nous allons voir comment la méthode PGD se comporte pour la résolution de ce problème.

Chapitre 5

Calcul du champ thermique par PGD

Ce chapitre a pour but d'étendre la formulation mathématique du problème thermique vue au chapitre précédent, et l'adapter à la PGD. On montre ici l'intérêt de la PGD multiparamétrique en considérant la conductivité thermique comme nouvelle coordonnée du problème à résoudre. En première approche, l'avantage du changement de coordonnée est présenté sur un modèle 1D. Ensuite, nous mettons en évidence les limites de cette approche dans le cas 3D.

Sommaire

5.1 PGD sur un domaine mono-couche

La simulation des phénomènes thermiques au cours des procédés de FA et notamment à l'échelle du bain de fusion ou de la micro-couche, exige de prendre des pas de temps très petit à cause de la forme de la source de chaleur comme présenté dans le paragraphe 4.1.2, et du grand nombre de couche ce qui nécessite d'utiliser un intervalle de simulation suffisamment grand. On propose ici de mettre en place la méthode PGD (*Proper Generalized Decomposition*) pour la simulation d'un problème dont la taille croît au cours du temps.

La méthode PGD repose sur la recherche a priori du champ solution sous la forme d'une représentation séparée. Au cours des vingt dernières années, de telles décompositions sont utilisées pour divers problèmes variés.

Dans le cadre d'une source mobile, N. Bur présente dans sa thèse [20] de déterminer la meilleure puissance de chauffe pour la mise en œuvre des composites. L'auteur propose donc un modèle réduit permettant la paramétrisation du nombre de couches en PFR (Placement de Fibre Robotisé). L'optimisation et le contrôle en temps réel sont aussi proposés par C. Ghnatios dans [38], qui met en avant le calcul "off-line" des solutions paramétriques, et leurs exploitations "on-line" sur des smartphones ou des tablettes [30]. Joyot et al. [52] soulèvent la question de l'efficacité de cette méthode pour la résolution de l'équation de la chaleur avec une conduction non linéaire. Les auteurs ont présenté plusieurs approches permettant de limiter la reconstruction des champs recherchés au cours des itérations.

5.1.1 Mise en équation de la PGD appliquée à une seule couche

On s'intéresse dans cette partie au comportement de la PGD lors du dépôt de couches, et par conséquent l'évolution du domaine. Pour ce, on reprend le milieu unidimensionnel dans lequel on étudie le problème thermique 5.1. Afin de valider l'approche PGD. Nous nous limitons au cas des conditions aux limites de Dirichlet.

$$\begin{cases} \rho\, C_p\, \dfrac{\partial\, u}{\partial\, t}\,(z,t) + \kappa\, \Delta\, u(z,t) & = f(z,t) \\ \quad u(z=0,t) & = 0 \\ \quad u(z,t=0) & = 0 \end{cases} \tag{5.1}$$

Par ailleurs, nous supposons le solide composé d'une seule couche. On suppose qu'il est possible d'évaluer la température de ce milieu en espace et en temps, et on cherche une approximation du champ thermique sous forme d'une somme de produits de modes spatiaux et d'amplitudes temporelles :

$$u(z,t) \;=\; \sum_{m=1}^{n} Z_m(z)\; T_m(t) \tag{5.2}$$

Dans notre application, le terme source se sépare naturellement, car il n'a qu'une composante

temporelle

$$f(z,t) = \begin{cases} \dfrac{Q}{n_{couche} \times t_{couche}} & si \quad \left(\dfrac{v\,t}{e_{couche}}\right) \times e_{couche} \leq z \leq \left(\dfrac{v\,t}{e_{couche}} + 1\right) \times e_{couche} \\ 0 \quad sinon \end{cases}$$

Il convient de maintenir l'extrémité Γ de Ω à une température nulle pour chaque mode

$$\begin{cases} Z^m(0) = & 0, \\ T^m(0) = & 0. \end{cases} \quad ; \forall m \in [1, n]$$

L'idée principale consiste à trouver l'approximation u_M de u sans aucune connaissance à priori du tenseur u. L'espace d'approximation incluant tous les modes espace-temps de rang fini n est représenté par

$$Z_h = \left\{ u \in \mathbb{R}^{n_Z} \otimes \mathbb{R}^{n_T} \quad | \quad u(z,t) = \sum_{m=1}^{n} Z_m(z) \otimes T_m(t) \quad \text{avec} \quad Z_m \in \mathbb{R}^{n_Z}, \;\; T_m \in \mathbb{R}^{n_T} \right\}$$

Nous allons résoudre le problème 5.1 en cherchant une approximation u^n de u sous forme

$$\begin{aligned} u^n(z,t) &= u^{n-1}(z,t) + Z_n(z)\, T_n(t) \\ &= \sum_{m=1}^{n-1} Z_m(z)\, T_m(t) + R(z)\, P(t) \end{aligned}$$

À chaque nouvel enrichissement, un produit $R(z)\,P(t)$ est ajouté à la somme. La meilleure approximation u^n de u est définie par L. Boucinha dans [16] comme étant la solution du problème de minimisation

$$u^n = \underset{u^\star \in Z_h}{\operatorname{argmin}} \|u - u^\star\|_2$$

Il existe différentes formulations de la PGD, la première est basée sur le critère d'orthogonalité de Galerkin [60, 79, 81] pour la résolution des problèmes symétriques. La deuxième approche est basée sur la minimisation de résidu [11, 20], cette approche permet d'assurer la convergence de la solution approchée dans le cas des opérateurs non symétriques vers la solution exacte. Cette définition est plus robuste [21] en particulier pour les problèmes non symétrique (terme de convection).

Ces différentes formulations sont passées en revue par A. Nouy dans [80], qui a aussi proposé une nouvelle définition appelée Minimax, dans laquelle les fonctions de bases réduites testées sont reliées par un problème adjoint. L'auteur propose aussi un algorithme pour chacune des définitions. Cependant, toutes ces définitions sont fondées sur le même principe de calcul, à savoir une phase d'enrichissement et une phase de normalisation, que l'on détaillera dans ce qui suit.

5.1.2 Formulation variationnelle

En séparant les variables, nous écrivons la formulation variationnelle de ce problème, pour cela on introduit la fonction test $u^* = R^*(z)\,P(t)$ admissible à 0 en espace et en temps et on intègre sur le domaine $\Omega = \Omega_z \times \Omega_t$. Prendre $R(z)$ et $P(t)$ dans $H_0^1(\Omega)$ permet d'assurer d'abord, l'existence de toutes les intégrales qui interviennent, et ensuite que la condition limite est bien vérifiée pour chaque mode. Nous obtenons la formulation suivante

$$\left\{\begin{array}{l} \text{Trouver } R(z) \in H_0^1(\Omega_z) \text{ tel que :} \\ \displaystyle\int_{\Omega_Z\times\Omega_T} R^*\,P\Big(\rho\,C_p\,R\,\frac{dP}{dt} - \kappa\,\Delta\,RP\Big)\,dz\,dt = -\int_{\Omega_Z\times\Omega_T} R^*\,P\sum_{i=1}^{n-1}\Big(Z_i\,\frac{dT_i}{dt} - \kappa\Delta Z_i\,T_i\Big)\,dz\,dt + \\ \qquad\qquad + \displaystyle\int_{\Omega_Z\times\Omega_T} R^*\,P\,f\,dz\,dt \qquad\qquad \forall R^*(z) \in H_0^1(\Omega_z) \end{array}\right. \tag{5.3}$$

La deuxième équation du problème couplé est identifiée de la même manière en prenant pour fonction test $u^* = R(z)\,P^*(t)$.

$$\left\{\begin{array}{l} \text{Trouver } P(t) \in H_0^1(\Omega_t) \text{ tel que :} \\ \displaystyle\int_{\Omega_Z\times\Omega_T} R\,P^*\Big(\rho\,C_p\,R\,\frac{dP}{dt} - \kappa\,\Delta\,RP\Big)\,dz\,dt = -\int_{\Omega_Z\times\Omega_T} R\,P^*\sum_{i=1}^{n-1}\Big(Z_i\,\frac{dT_i}{dt} - \kappa\Delta Z_i\,T_i\Big)\,dz\,dt + \\ \qquad\qquad + \displaystyle\int_{\Omega_Z\times\Omega_T} R\,P^*\,f\,dz\,dt \qquad\qquad \forall P^*(t) \in H_0^1(\Omega_t) \end{array}\right. \tag{5.4}$$

Le schéma le plus simple pour la résolution de ce problème couplé est l'algorithme du point fixe. Cette méthode consiste à déterminer dans un premier temps $R(z)$ à partir de la fonction $P(t)$ initialement connue pour démarrer le processus d'enrichissement. Le résultat obtenu est ensuite injecté dans l'équation 5.4 afin de déterminer une mise à jour de $P(t)$. On répète alors successivement ces deux étapes jusqu'à l'obtention de l'estimation de l'erreur relative proposée par F. Chinesta dans [24]

$$\frac{\|R(z)P(t) - Z_{n-1}(z)T_{n-1}(t)\|}{\|Z_{n-1}(z)T_{n-1}(t)\|} < \varepsilon \tag{5.5}$$

On présente l'algorithme 2 utilisé pour l'implémentation de la PGD qui permet d'obtenir une approximation *a priori* à variables séparées espace-temps. Afin d'assurer la stabilité de l'algorithme du point fixe, il est nécessaire de normaliser les fonctions séparées après chaque calcul (ligne 13 de l'algorithme 2) [10]. Le critère d'arrêt de l'enrichissement est défini comme étant l'écart entre l'ancien couple (mode, amplitude) et le nouveau qui doit être inférieur à

un certain critère prédéfini par l'utilisateur.

Algorithme 2 : PGD espace-temps

fonction PGD (z,t);
Données : $A_z^1, A_z^2, \cdots$, matrices sur Ω_Z,
$A_t^1, A_t^2, \cdots$, matrices sur Ω_T
Sorties : $u(z,t) = \sum_{m=1}^{n} Z_m(z)\, T_m(t)$ solution de $\mathbb{A}U = \mathbb{F}$.
Paramètres : $\varepsilon,\ \widetilde{\varepsilon},\ M_{Max},\ k_{fp}$

for $i = 1$ *to* M_{Max} **do**
  Initialiser la base $[T_1, \cdots, T_n]$;
  Initialiser la base $[Z_1, \cdots, Z_n]$;
  for $j = 1$ *to* k_{fp} **do**
    Boucle sur l'algorithme du point fixe;
    Trouver Z_n en utilisant la formulation variationnelle et connaissant T_{n-1};
      Résoudre le système $\mathcal{A}_n^1(T_{n-1})Z_n = \mathcal{B}_n^1$;
    Mise à jour des valeurs de la base $[Z_1, \cdots, Z_n]$;
    Calcul des valeurs $[T_1, \cdots, T_n]$ en connaissant Z_n;
      Résoudre le système $\mathcal{A}_n^2(T_n)Z_n = \mathcal{B}_n^2$;
    Normaliser;
    Vérifier la convergence 5.5 du point fixe;
    if $\dfrac{\|Z_n(z)T_n(t) - Z_{n-1}(z)T_{n-1}(t)\|}{\|Z_{n-1}(z)T_{n-1}(t)\|} < \varepsilon$ **then**
      break;
    end
  end
  /*Vérifier le critère d'arrêt de l'enrichissement*;
  if $\dfrac{\|Z_n(z)T_n(t)\|}{\|Z_{n-1}(z)T_{n-1}(t)\|} < \widetilde{\varepsilon}$ **then**
    break;
  end
end

5.1.3 Résultats et validation pour une macro-couche

Afin de mettre en place la PGD, on reprend le modèle présenté dans 5.1, en considérant une conductivité thermique constante égale à la conductivité thermique de l'acier, on obtient alors la solution reconstruite représentée par la courbe rouge dans la figure 5.1 (a).

Dans cet exemple, nous analysons brièvement l'utilisation de la PGD pour la simulation du problème thermique. Dans ces tests numériques, nous choisissons un critère de convergence $\varepsilon = 10^{-8}$ pour les itérations, le nombre d'itérations maximum est fixé à 20. Néanmoins, seules 4 itérations suffisent pour capturer avec précision le couple dominant (mode, amplitude).

La figure 5.1 (b) illustre l'estimateur d'erreur en norme infinie entre les valeurs obtenues en utilisant le calcul purement FEM et le calcul PGD. Cette courbe indique que les résultats

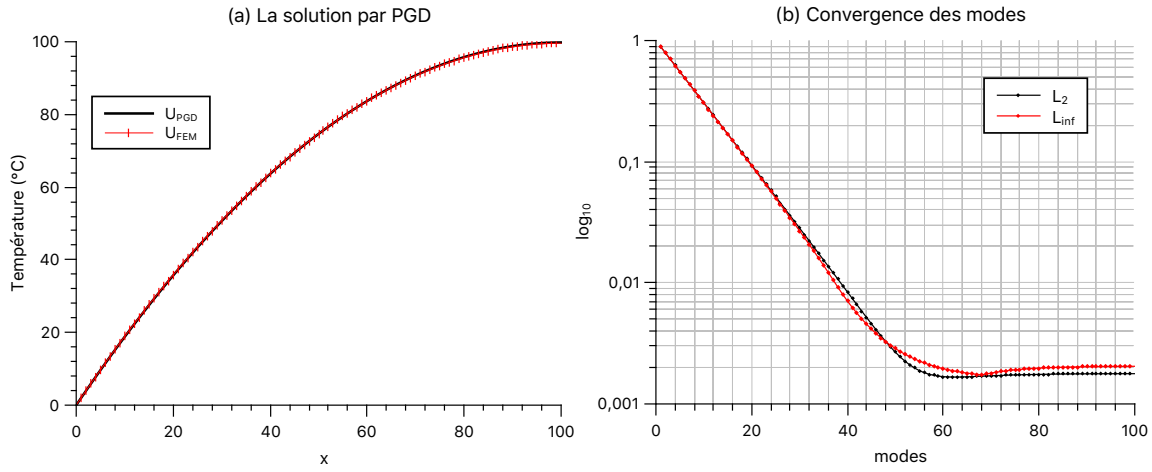
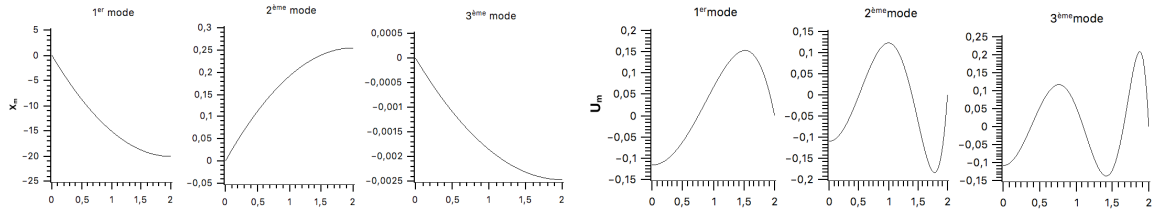


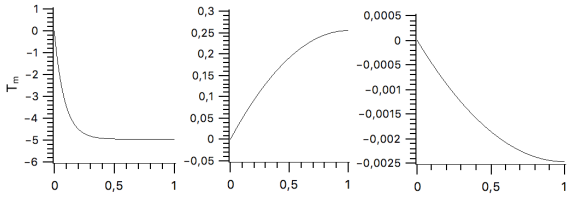
FIGURE 5.1 – La solution par PGD.

de la PGD ne sont pas affectés par le caractère aléatoire de l'initialisation.

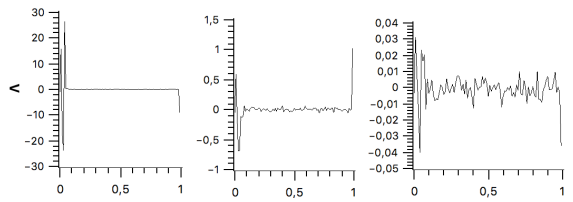


(a) Mode Z_m.

(b) Mode U_m.



(c) Mode T_m.

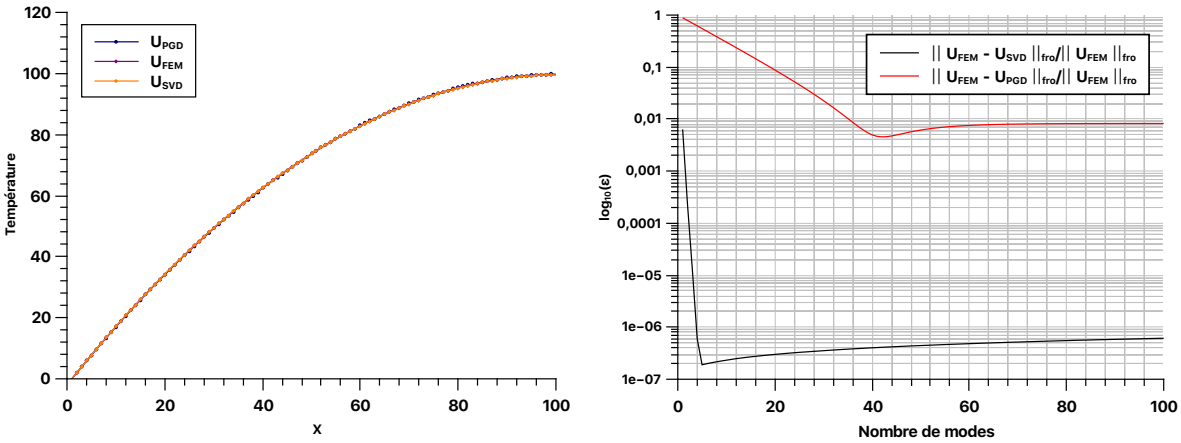


(d) Mode $\Lambda_m = \Sigma \cdot V_m$.

FIGURE 5.2 – Comparaison des 3 premiers modes obtenus par PGD (5.2a , 5.2c) et SVD (5.2b , 5.2d).

Sur la figure 5.2, on trace les 3 premiers modes déterminés par la PGD et les modes obtenus en utilisant la SVD. On peut observer que l'allure des modes est différente selon la stratégie utilisée (PGD/SVD). Afin de comparer ces deux méthodes, il est donc utile de comparer la reconstruction des champs thermiques comme illustré par la figure 5.3a. On observe que les deux algorithmes permettent d'obtenir des résultats quasi-similaires.

La figure 5.3b représente l'erreur de Frobenius des différentes solutions obtenues en utilisant l'algorithme *a priori*, et l'algorithme *a posteriori*. Dans les deux cas une erreur de codage



(a) Reconstruction des champs thermiques.

(b) Représentation des erreurs relatives calculées en norme de Frobenius.

FIGURE 5.3 – Application de la PGD et la SVD au même problème.

est observée au niveau de l'allure des courbes. La courbe représentant l'erreur de la SVD décroît plus rapidement, ce qui est attendu étant donnée que dans ce cadre c'est une utilisation de l'algorithme *a posteriori*.

5.1.4 Avantage du modèle paramétrique

C. Paillet [85] présente une variante de la méthode PGD qui permet de contourner la limite actuelle liée au nombre de paramètre élevé (autour d'une vingtaine) à prendre en considération pour le traitement de certains problèmes. L'auteur développe des algorithmes qui permettent d'aborder des problèmes allant jusqu'à mille paramètres. P-E. Allier [10] décrit l'algorithme PGD paramétrique appliqué aux problèmes thermiques et élastiques, en présentant un estimateur d'erreur afin de mesurer la qualité des solutions obtenues.

À présent, on reprend le problème thermique 5.1 mais sans une connaissance préalable de la valeur de la conductivité thermique.

On cherche à approximer le champ de température sous la forme séparée suivante

$$u^n(z,t,k) = \sum_{i=1}^{n-1} Z_i(z)T_i(t)K_i(\kappa) + R(z)P(t)W(\kappa)$$

À chaque enrichissement, on suppose les $n-1$ premiers modes connus, et on cherche à déterminer alternativement l'ensemble des modes spatiaux $R(z)$, modes temporels $P(t)$, et N

fonctions paramétriques $W^i(\kappa)$ solution du problème de minimisation suivant

$$\min_{\{W\}^n} \|\mathcal{R}^n - R(x)\,P(t)\prod_{i=1}^{N} W^i(\kappa)\|$$

On introduit dans ce qui suit l'algorithme PGD illustré par la figure 5.4 où la conductivité thermique est considérée comme paramètre supplémentaire du problème à résoudre. La technique utilisée est très proche de celle précédemment introduite et ne représente pas de difficulté particulière étant donné le nombre de paramètre.

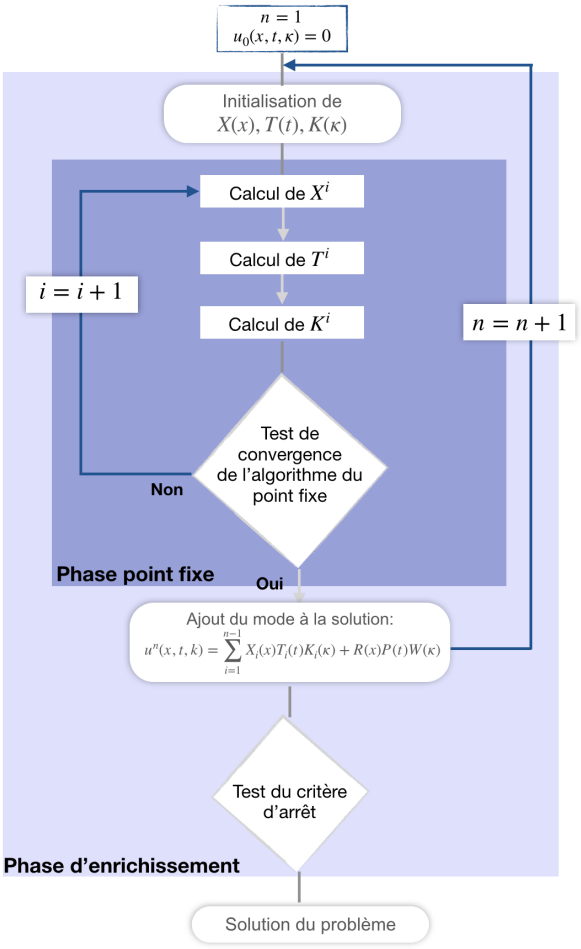


FIGURE 5.4 – L'algorithme de résolution d'un problème paramétrique.

5.1.5 Notation matricielle

La formulation variationnelle est représentée de manière condensée en considérant la fonction test $u^\star(z,t,k) = R^\star(z)P(t)W(\kappa) + R(z)P^\star(t)W(\kappa) + R(z)P(t)W^\star(\kappa)$.

$$\begin{cases} \displaystyle\int_{\Omega_Z\times\Omega_T\times\Omega_\kappa} R^*\,P^\star\,W^\star\Big(\rho\,C_p\,R\,\frac{dP}{dt}W - \kappa\,\Delta\,RPW\Big)\,dz\,dt\,d\kappa = -\int_{\Omega_Z\times\Omega_T\times\Omega_\kappa} R^*\,P^\star\,W^\star\sum_{i=1}^{n-1}\Big(\rho\,C_p\cdot \\ \qquad\qquad \cdot Z_i\,\frac{dT_i}{dt}\,K_i - \kappa\Delta Z_i\,T_i\,K_i\Big)\,dz\,dt\,d\kappa + \displaystyle\int_{\Omega_Z\times\Omega_T\times\Omega_\kappa} R^*\,P^\star\,W^\star\,f\,dz\,dt\,d\kappa \end{cases} \tag{5.6}$$

Soient N_Z et N_T les vecteurs des fonctions de forme. On définit les matrices de masse et de raideur comme suit

$$\mathbb{M} = \int_{\Omega_Z} NN^T d\Omega_Z \quad \mathbb{K} = \int_{\Omega_Z} \frac{dN}{dz}\frac{dN^T}{dz} d\Omega_Z \quad \mathbb{C} = \int_{\Omega_t} Nc^T d\Omega_t$$

avec

$$c_i(t) = \begin{cases} \dfrac{1}{\Delta t} & z \in [z_{i-1}, z_i] \\ 0 & sinon. \end{cases}$$

La formulation matricielle est donc représentée sous la forme discrète suivante

$$\begin{cases} \rho\,C_p\cdot\Big(R^{*T}\,\mathbb{M}\,R\Big)\cdot\Big(P^{*T}\,\mathbb{C}\,P\Big)\cdot\Big(W^{*T}\,\mathbb{M}\,W\Big) - \Big(R^{*T}\,\mathbb{K}\,R\Big)\cdot\Big(P^{*T}\,\mathbb{M}\,P\Big)\cdot\Big(W^{*T}\,\mathbb{M}\,W\Big) = \\ \qquad = -\displaystyle\sum_{i=1}^{n-1}\Big[\rho\,C_p\cdot\Big(R^{*T}\,\mathbb{M}\,Z_i\Big)\cdot\Big(P^{*T}\,\mathbb{C}\,T_i\Big)\cdot\Big(W^{*T}\,\mathbb{M}\,K_i\Big) - \\ \qquad - \Big(R^{*T}\,\mathbb{K}\,Z_i\Big)\cdot\Big(P^{*T}\,\mathbb{M}\,T_i\Big)\cdot\Big(W^{*T}\,\mathbb{M}\,K_i\Big)\Big] \end{cases} \tag{5.7}$$

5.1.6 Résultats

En utilisant les mêmes paramètres de vitesse et de puissance de la source laser, nous pouvons comparer avec les résultats obtenus par FEM. Ainsi, la figure 5.5 illustre les deux solutions approchées du problème considéré. Pour comparer ces solutions, nous calculons les normes $\mathcal{L}_2$ et $\mathcal{L}_\infty$ que l'on représente sur la figure 5.6. Cette figure permet de visualiser la convergence des modes et de déterminer le nombre de modes nécessaires pour approcher au mieux la solution. Dans cet exemple, 100 modes ont été calculés, néanmoins, seulement 30 modes permettent de reconstruire le champ de température. Comme l'illustrent les courbes la PGD espace-temps permet une meilleure précision lorsqu'on augmente le nombre de modes retenus.

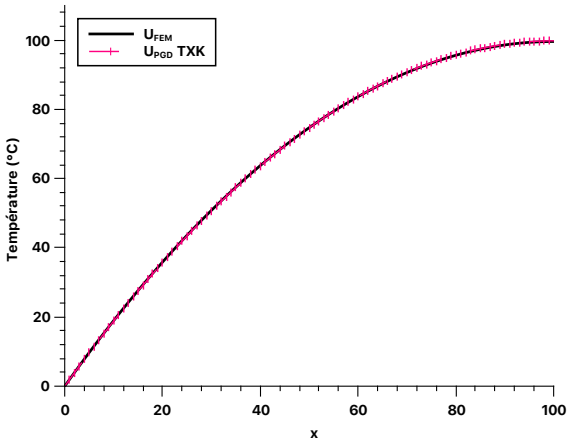


FIGURE 5.5 – La solution par PGD.

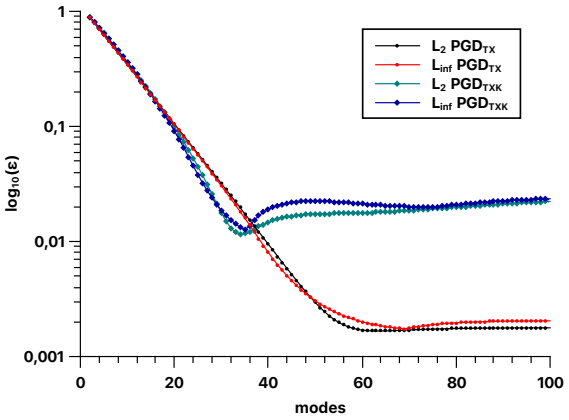


FIGURE 5.6 – Convergence.

5.2 PGD sur un domaine multicouche évoluant au cours du temps

Nous reprenons ici le modèle initial décrit dans la section 4.1.1. Comme vu précédemment dans la section 4.2.1.2, le redimensionnement du domaine suivant l'axe z permet de réduire considérablement le nombre de modes.

Dans le cadre de la PGD, en plus de devoir travailler dans le même domaine suivant l'axe z afin de réduire le nombre de modes, il existe une autre pathologie liée à l'explosion du nombre d'opérateurs. Pour ce, dans un premier temps on aborde uniquement le cas 1D. L'objectif est de pouvoir effectuer une transformation du domaine spatial $\Omega_{z_1}, \cdots, \Omega_{z_i}$ à un intervalle qui varie entre 0 et 1. On introduit les fonctions de base définies sur un maillage uniforme par $z = \phi_i(\zeta) = ie\,\zeta$. Pour passer d'un calcul de $z \in [0, i \times e]$ à une intégration sur l'intervalle $[0, 1]$, on pose $dz = ie\,d\zeta$. On présente la formulation faible de la PGD en utilisant les mêmes fonctions de formes présentées dans la section 4.2.1.2.

En considérant le terme de gauche de l'équation 5.3, on obtient alors dans le cas d'un domaine évoluant au cours du temps

$$\sum_{i=1}^{nn} \rho\, C_p \int_{\Omega_{T_i}} P(t)\, \frac{dP}{dt}\, dt \int_{\Omega_{Z_1}+\cdots+\Omega_{Z_i}} R^*\, R(z) dz - \sum_{i=1}^{nn} \kappa \int_{\Omega_{T_i}} P(t)\, P\, dt \int_{\Omega_{Z_1}+\cdots+\Omega_{Z_i}} \nabla_z R^*\, \nabla_z R(z) dz = \cdots \quad (5.8)$$

En effectuant un changement de variable et à partir de l'équation 4.9, on peut représenter la formulation faible sous la forme suivante

$$\rho\, C_p \left(\sum_{i=1}^{nn} ie \int_{\Omega_{T_i}} P\, \frac{dP}{dt}\, dt \right) \left(\int_0^1 \bar{R}^*(\zeta)\, \bar{R}(\zeta) d(\zeta) \right) - \frac{\kappa}{e} \left(\sum_{i=1}^{nn} \frac{1}{i} \int_{\Omega_{T_i}} P\, P\, dt \right) \left(\int_0^1 \nabla_\zeta \bar{R}^*(\zeta)\, \nabla_\zeta \bar{R}(\zeta) d\zeta \right) = \cdots \quad (5.9)$$

Il est à noter que cette méthode permet non seulement de réduire le nombre de modes, mais aussi de réduire le nombre d'opérateurs. En effet, comme indiqué par l'équation 5.10, on ne

retrouve plus un opérateur en temps et un opérateur en espace.

$$\Longrightarrow \left(\mathbb{C}_t \otimes \mathbb{M}_\zeta + \mathbb{M}_t \otimes \mathbb{K}_\zeta\right) U = V_t \otimes V_\zeta \tag{5.10}$$

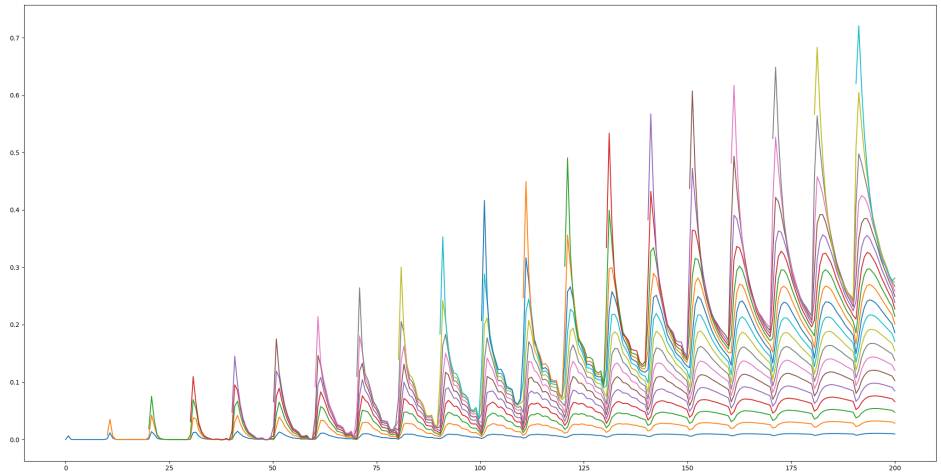


FIGURE 5.7 – La température en utilisant la PGD sur un domaine réduit.

La figure 5.7 illustre l'évolution du domaine d'étude en superposant les champs thermiques au niveau des différentes couches en un seul graphe.

5.3 Difficultés liées au cas 3D

La figure 5.8 illustre une décomposition espace (z) plan (xy). Un bloc représente une macro-couche composée de plusieurs couches différentes selon le plan xy. En se basant sur les mêmes équations précédemment introduites dans le cas 1D (cf. 4.2.1.2 et 5.2), on remarque que la résolution du problème en appliquant un changement de variable (cf. 5.2) sur le domaine Ω_z ne permet pas de factorisation à cause du nombre d'opérateur qui est lié au nombre de couche. Étant donné qu'on ne peut contourner la pathologie liée à l'explosion du nombre d'opérateurs, on n'étend donc pas la méthode de changement de variable au cas 2D au cours de ce travail. Cependant, l'utilisation du GPU pourrait réduire les temps de calculs.

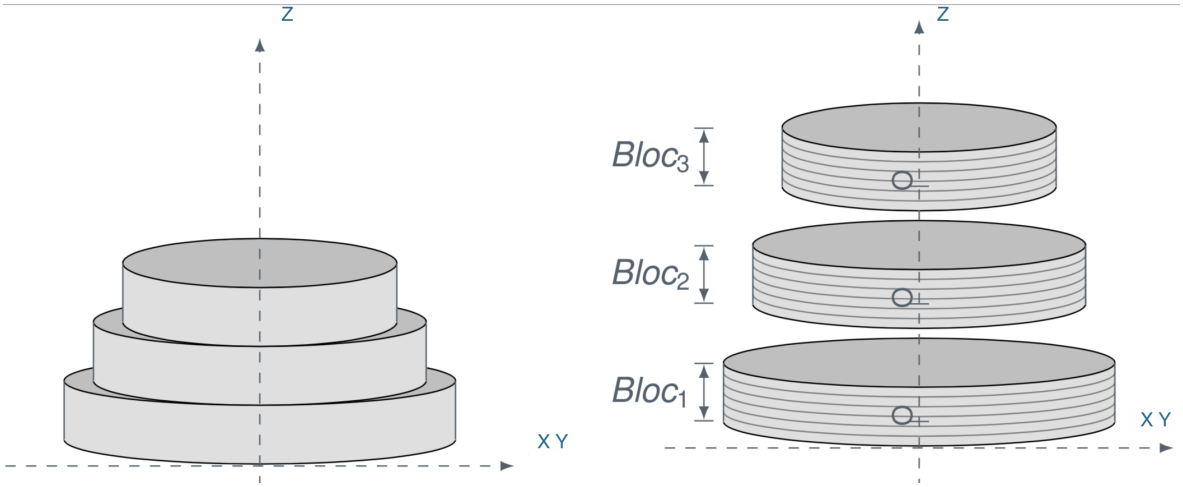


FIGURE 5.8 – Séparation espace-plan pour un domaine 3D.

5.4 Conclusions

Dans le but de déterminer le comportement de la PGD pour la résolution du problème thermique sur un domaine qui évolue au cours du temps, nous avions émis l'hypothèse dans un premier temps que l'étude du cas 1D pouvait suffire. La mise en place de la PGD est étudiée et validée sur un modèle composé d'une seule macro-couche et de 100 micro-couches.

L'utilisation de la méthode de changement de variable a permis d'améliorer la prédiction de la solution, tout en réduisant à la fois le nombre de modes nécessaires ainsi que le nombre d'opérateurs.

Cependant, dans certains cas, notamment au niveau de la géométrie plane (cas 2D), le nombre d'opérateurs ne peut être réduit, ce qui implique l'utilisation d'un grand nombre d'opérateur. Afin de contourner ce problème, on peut tirer parti des architectures hybrides que nous offrent nos machines actuelles ce que nous étudierons plus en détail dans le chapitre 7.

Troisième partie

Simulation mécanique du procédé SLM

Chapitre 6

Déformations inhérentes et analyse inverse

L'objectif de ce chapitre est de présenter plusieurs modèles analytiques qui permettent de comprendre la notion de déformation inhérente, et ce, à travers l'exploration des travaux existants appliqués à la simulation du soudage et adaptée par la suite au SLM. Dans un second temps, nous aborderons la notion d'analyse inverse pour le calcul des déformations inhérentes. Nous détaillerons la méthodologie, et illustrerons son utilisation sur une éprouvette de référence. Enfin, nous détaillerons un modèle paramétrique, PGD, permettant de réaliser plusieurs simulations afin de déterminer les composantes de déformations inhérentes inconnues tout en réduisant les coûts de calcul.

Sommaire

6.1 Modèles mécaniques simplifiés

6.1.1 Déformation inhérente et modèle d'Ueda

Avant d'aborder l'analyse inverse décrite dans la section 2.4, nous présentons brièvement le modèle simplifié proposé par [103, 102, 104] et utilisé dans [107], permettant d'introduire la notion de déformations inhérentes inspiré des stratégies mises en œuvre en simulation numérique du soudage.

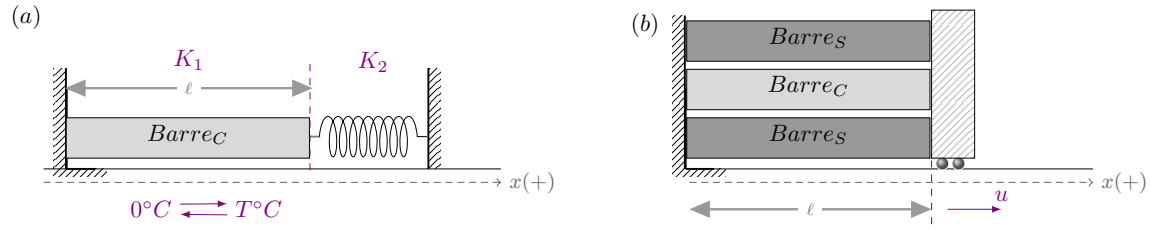


FIGURE 6.1 – Représentation du modèle simplifié.

Le soudage est modélisé en utilisant un modèle à 3 barres. Une barre C (figure 6.1 (a)) représente la zone à haute température où la déformation inhérente se produit, tandis que le voisinage est représenté par le ressort ou des barres latérales (figure 6.1 (b)). Le voisinage ne subit pas de changements de température et agit comme une contrainte sur la barre centrale. En effet, le modèle est encastré à son extrémité gauche et relié à un corps rigide mobile à l'autre extrémité.

Lorsque la barre C est chauffée à une température T_{max} à partir d'une température ambiante ($T_{amb} = 0°C$) puis refroidie à T_{amb}, une contrainte thermique de compression apparaît comme illustrée par la figure 2.8. Le comportement élasto-plastique quant à lui varie en fonction de la température maximale T_{max} par rapport à T_{amb} et T_Y (figure 2.8). Avec

$$T_Y = \frac{A + A_0}{A_0} \frac{\sigma_Y}{E\alpha}$$

avec A la section (constante) de la barre C, $A_0/2$ la section transverse de la barre S, E le module de Young, et ℓ la longueur de la barre.

Quand le processus de chauffage appliqué est à basse température ($0°C \mapsto T_{max} \leq T_Y$) aucune déformation plastique ni contrainte résiduelle ne peut apparaître. Étant donné que la barre C est limitée par les barres S via le corps rigide mobile, elle est donc soumise à une déformation $\alpha T \ell$ en raison du coefficient de dilatation thermique α.

Au cours de ce processus, la barre C se retrouve donc en contrainte de compression, et les barres S en contrainte de traction. Toutes les barres sont soumises au même déplacement u. Néanmoins, comme cité auparavant, le changement de température s'applique uniquement au niveau de la barre C, c'est pourquoi la déformation thermique est observée uniquement

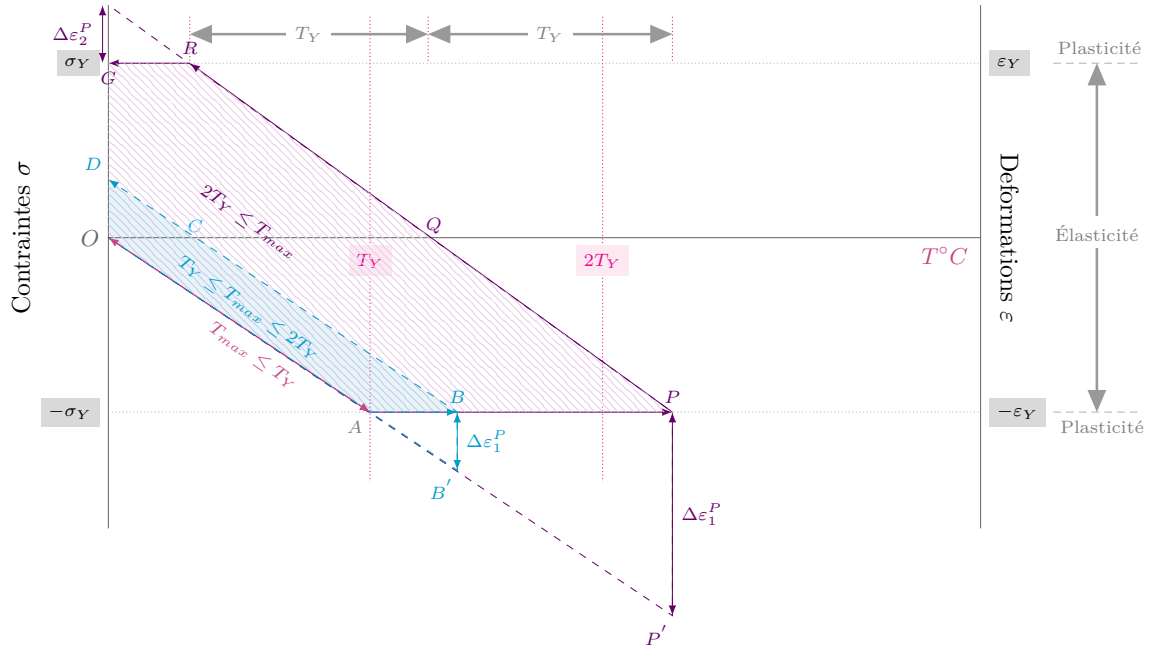


FIGURE 6.2 – Représentation de l'histoire thermique en utilisant différentes configurations.

sur ce barreau. On obtient alors les déformations et contraintes de ce système comme suit

$$\begin{cases} \text{Barre C :} & \varepsilon^e = \varepsilon - \varepsilon^{th} = \dfrac{u}{\ell} - \alpha T \\ \text{Barre S :} & \varepsilon = \varepsilon^e = \dfrac{u}{\ell} \end{cases} \Rightarrow \begin{cases} \text{Barre C :} & \sigma = E(\dfrac{u}{\ell} - \alpha T) \\ \text{Barre S :} & \sigma = E\dfrac{u}{\ell} \end{cases} \tag{6.1}$$

Le déplacement des barres est alors défini par l'équation 6.3 et ce, à partir de l'équation d'équilibre suivante

$$\frac{A_0}{2}E\frac{u}{\ell} + AE(\frac{u}{\ell} - \alpha T) + \frac{A_0}{2}E\frac{u}{\ell} = 0 \tag{6.2}$$

$$\Rightarrow u = (\frac{A}{A_0 + A})\alpha T\ell \tag{6.3}$$

Lorsque la température diminue, le point de contrainte A (cf. figure 6.2) revient à l'origine. la barre C se comporte comme un solide élastique et donc aucune contrainte résiduelle ne peut apparaître.

La zone en bleu sur la figure 6.2 illustre le processus de chauffe lorsque T_{max} dépasse la limite d'élasticité T_Y, ce qui provoque une déformation plastique de compression $\Delta\varepsilon_1^P$ et produit une déformation thermique supplémentaire $\alpha\Delta T$ avec $\Delta T = (T_{max} - T_Y)$.

Lorsque la température diminue, le système subit une décharge élastique. En conséquence, la contrainte totale dans chaque barre devient nulle au point C de la figure 6.2. Ensuite, la barre subit une décharge complètement élastique et atteint le point D.

Cependant, lorsque la température de chauffe appliquée est élevée ($0°C \mapsto 2T_Y \leq T_{max}$), la phase de refroidissement se déroule en 3 étapes : la première phase est la décharge du point P au point Q (cf. 6.2), ce qui produit une déformation $= -\alpha T_Y$. Lors de la seconde étape, le refroidissement de la barre se poursuit, sa température chute à nouveau de T_Y, conduisant à une contraction thermique $\Delta\varepsilon^{th} = -\alpha T_Y$. Lorsque le point de contrainte atteint le point R, les déformations thermiques $\varepsilon^{th} = \alpha(T_{max} - 2T_Y)$ et plastiques sont observées $\varepsilon^P = -\alpha(T_{max} - T_Y)$. Lors de la dernière étape, une déformation plastique de traction $\Delta\varepsilon_2^P$ lorsque le point de contrainte atteint le point G. La déformation inhérente est alors la somme des déformations plastiques

$$\begin{aligned} \varepsilon^\star &= \Delta\varepsilon_1^P + \Delta\varepsilon_2^P \\ &= -\alpha(T_{max} - T_Y) + \alpha(T_{max} - 2T_Y) \quad = -\alpha T_Y \end{aligned} \tag{6.4}$$

T_Y dépend de $\sigma_Y, E, \alpha, A, A_0$, qui sont connus. Ainsi, Ueda montre que la déformation inhérente $\varepsilon^\star$ est directement reliée à T_Y. On peut donc facilement la calculer. Pour des géométries soudées plus complexes, la méthode fait l'hypothèse que les déformations plastiques restent identiques au cours du soudage et peuvent être assimilées à cette déformation inhérente. Son calcul demande de réaliser une simulation fine d'une partie limitée du cordon de soudure afin d'en extraire sa valeur. Au niveau de la structure soudée, la partie plastique sera remplacée par $\varepsilon^\star$. Il ne restera donc que la partie élastique à calculer. Cette approche est valable pour le soudage, elle est également utilisée pour la modélisation du SLM [55], même si le fait de déposer des couches change la physique du problème.

Analysons maintenant les modèles analytiques adapté au dépôt de couches.

6.1.2 Modèle de Safronov et al.

Le modèle de Safronov et al. est un modèle simplifié adapté au dépôt de couches. Au cours de ce procédé, une source laser focalisée, fait fondre localement un lit de poudre selon une trajectoire prédéfinie. Des couches d'une épaisseur de l'ordre de $50\mu m$ sont ainsi superposées (cf. 6.3) et l'adhésion entre ces couches est garantie par la re-fusion partielle de la couche inférieure lors du lasage d'une nouvelle couche.

Dans la figure 6.3, les couches rouges représentent les couches actives et donc les régions à température élevée. Chaque couche à une longueur initiale ℓ, et est encastrée à son extrémité gauche et est en appui à l'extrémité droite (pour l'empêcher uniquement de se plier mais libre de se déplacer).

Si les déformations subies restent faibles, le comportement reste élastique, et donc à peu près semblables à des ressorts. De ce fait, et afin de mettre en évidence les contraintes de traction qui s'appliquent aux couches, on adopte une représentation simplifiée illustrée par les schémas (a.1, b.1 et c.1) de la figure 6.3. En FA, contrairement au soudage, une température initiale différente de zéro est imposée.

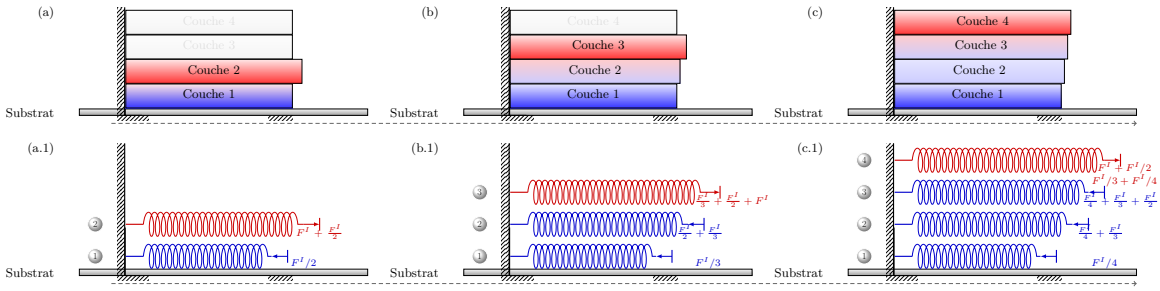


FIGURE 6.3 – Représentation du modèle simplifié en FA.

Lors du dépôt d'une couche, les couches sous-jacentes ont un rôle important car elles limitent la dilatation. Le calcul des contraintes générées dans une couche au cours du procédé doit donc tenir compte de ces interactions. Au cours du calcul, les couches sous-jacentes limitent la dilatation thermique et la contraction des couches [88]. Au cours du calcul, l'étude au niveau des interfaces ainsi que la partie de la couche exposée au laser sont à prendre en considération.

Les différentes couches possèdent les mêmes propriétés élastiques, la déformation longitudinale $\varepsilon = \dfrac{\Delta\ell}{\ell}$ augmente en fonction de la force appliquée F. Soit S la section transverse, en phase élastique, ε est proportionnel à $\sigma = \dfrac{F}{S}$.

En utilisant la loi de Hooke, la proportionnalité entre la contrainte longitudinale σ et ε peut être exprimée par la relation

$$\sigma = E\varepsilon = \frac{F}{S} = E\frac{\Delta\ell}{\ell}$$

On obtient donc,

$$\frac{F}{F_0} = \frac{\ell - \ell_0}{\ell_0} \implies \frac{\ell}{\ell_0} = 1 + \frac{F}{F_0}$$

Avec ℓ_0 la longueur de la couche non chargée, et $F_0 = E \cdot S_0$. Lorsqu'une nouvelle couche est déposée, sa longueur est linéairement allongée par dilatation thermique [94]

$$\varepsilon = \frac{\ell}{\ell_0} - 1 = \alpha\Delta T \implies \frac{\ell}{\ell_0} = \varepsilon + 1 \tag{6.5}$$

Étant donné que la première couche est construite sur un support, le changement de température s'applique donc uniquement sur cette couche, elle est donc d'une longueur $\ell_1 = \ell_0$. Au cours du processus de construction de la deuxième couche, l'amplitude de la dilatation thermique est contrôlée par la couche précédente. En conséquence, la couche active se retrouve en contrainte de traction. Des allongements relatifs $\varepsilon_1 < 0$ et $\varepsilon_2 > 0$ apparaissent lorsque la température de cette couche diminue et s'approche de la température de la première couche.

La différence de retrait entre les deux couches est égale à :

$$\Delta\varepsilon = \varepsilon_2 - \varepsilon_1 = \frac{\ell_1}{\ell_0} - \frac{\ell_2}{\ell_0} = \varepsilon$$

Étant donné que les deux couches ont la même section lorsque celles-ci sont indépendantes les unes des autres, alors la déformation moyenne assurant l'équilibre [87]

$$\sigma_1 + \sigma_2 = 0 \implies \varepsilon_1 + \varepsilon_2 = 0$$

correspond à

$$\varepsilon_2 = -\varepsilon_1 = \frac{\alpha\Delta T}{2}$$

En généralisant la formulation pour N couches, cet exercice a été effectué dans [95, 94]. L'équilibre des forces est représenté par

$$\sum_{i=1}^{N} \varepsilon_i = 0$$

La relation entre les déformations découlent du caractère consécutif du dépôt de couche [27, 94]

$$\varepsilon_{i+1} - \varepsilon_i = \frac{\varepsilon}{i} \tag{6.6}$$

Comme on remarque sur la figure 6.3, la première couche est contractée alors que la dernière est en traction. Ce qui permet d'introduire la déformation résiduelle au niveau de ces couches comme suit [95]

$$\varepsilon_1 = \sum_{j=2}^{N} \frac{-\varepsilon}{j} \quad \text{et} \qquad \varepsilon_N = \frac{\varepsilon(N-1)}{N}$$

La déformation résiduelle qui apparait au niveau des autres couches est déterminée par

$$\varepsilon_i = \varepsilon\Big(1 - \sum_{j=1}^{N} \frac{1}{j}\Big), \forall i = 1, \cdots, N$$

6.1.3 Modèle analytique appliqué à la fabrication additive

Nous proposons dans cette section un modèle analytique (cf. figure 6.4) adapté au SLM en s'inspirant du modèle de Ueda et al. [102] décrit dans la section 6.1.1 et où les couches déposées jouent le rôle de la barre S (cf. figure 6.1).

En utilisant la loi de Hooke, la proportionnalité entre la contrainte longitudinale σ et ε peut être exprimée pour la première couche par la relation

$$\sigma^{(0)} = E\varepsilon^{e(0)} = E\frac{u^{(1;0)}}{\ell_0} \qquad \text{Avec } u^{(1;0)} = \ell_1 - \ell_0 \text{ et } \varepsilon^{(1;0)} = \varepsilon^{e(0)} \tag{6.7}$$

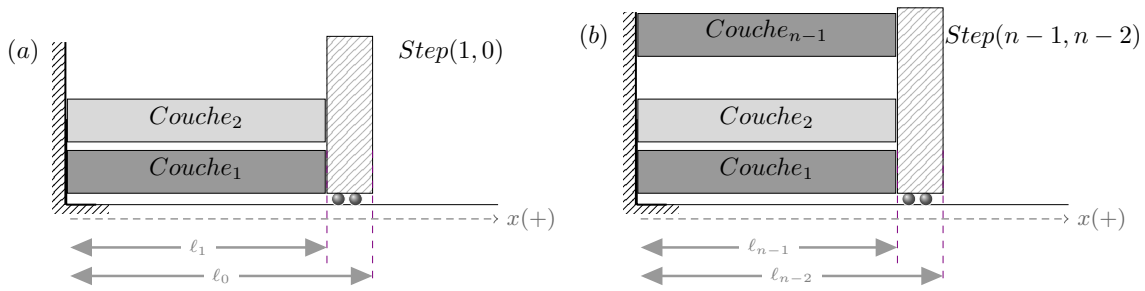


FIGURE 6.4 – Représentation du modèle analytique.

Où $\varepsilon^{e(0)}$ représente la contrainte élastique de la couche active. Au cours du processus de construction de la deuxième couche, une déformation thermique est observée comme introduit dans les sections précédentes

$$\varepsilon^{th(1)} = \alpha(0 - T) = -\alpha T$$

On obtient alors

$$\varepsilon^{e(1)} = \varepsilon^{(1;0)} - \varepsilon^{th(1)} \implies \sigma^{(1)} = E\Big(\frac{u^{(1;0)}}{\ell_0} + \alpha T\Big) \tag{6.8}$$

Le déplacement des couches est alors défini par l'équation 6.10 et ce, à partir de l'équation d'équilibre suivante

$$\begin{gathered} A_0\sigma^{(0)} + A\sigma^{(1)} = 0 \\ A_0 E\frac{u^{(1;0)}}{\ell_0} + AE(\frac{u^{(1;0)}}{\ell_0} + \alpha T) = 0 \end{gathered} \tag{6.9}$$

$$\Rightarrow u^{(1;0)} = -\beta_1\alpha T\ell_0 \qquad \text{Avec } \beta_i = (\frac{A}{A_0 + iA});\ \ i = 1 \tag{6.10}$$

Les contraintes peuvent être définies comme suit

$$\begin{cases} \text{Couche}_0 : & \sigma^{(0)} = -E\beta_1\alpha T \\ \text{Couche}_1 : & \sigma^{(1)} = -E(\beta_1 - 1)\alpha T \end{cases} \tag{6.11}$$

En généralisant la formulation pour n couches, la relation de déformation se découle du caractère consécutif du dépôt de couche

$$\varepsilon^{(n-1;0)} = \frac{u^{(n-1;0)}}{\ell_0} \approx -\left(\sum_{i=1}^{n-1}\beta_i\right)\alpha T \tag{6.12}$$

Les contraintes peuvent être généralisées comme suit

$$\begin{cases} \text{Couche}_0 : & \sigma^{(0)} = -E\left(\sum_{i=1}^{n-1} \beta_i\right)\alpha T \\ \text{Couche}_j : & \sigma^{(j)} = -E\left(\sum_{i=j}^{n-1} \beta_i - 1\right)\alpha T \\ \text{Couche}_{n-1} : & \sigma^{(n-1)} = -E(\beta_{n-1} - 1)\alpha T \end{cases} \tag{6.13}$$

Lorsque $A_0 = A$, le modèle que nous proposons, est identique au modèle de Safronov 6.1.2. Il est possible d'étendre cette approche au cas plastique, c'est un travail en cours.

Ces modèles simplifiés ne donnent pas de résultats fiables, mais permettent de mieux comprendre l'origine de l'approche des déformations inhérentes $\varepsilon^\star$, adapté au SLM que l'on va traduire par la suite par les équations 6.17 et 6.18.

6.2 Estimation de la distorsion en utilisant la méthode des déformations inhérentes

6.2.1 Méthodologie

Ploshikhin et al. [55] ont suggéré d'utiliser la théorie des déformations inhérentes, jusqu'alors utilisée en soudage [103], pour estimer la distorsion dans le cadre du procédé SLM. La déformation inhérente (2.4.3) est qualifiée de "déformation incompatible", au sens où elle ne peut exister sans contrainte. Par conséquent, toute contrainte présente dans un corps non chargé peut être vue comme le résultat d'une déformation inhérente. Déterminer cette déformation permet de reproduire, les contraintes résiduelles qui peuvent conduire à une distorsion importante post-fabrication, après découpe des supports et séparation de la pièce du substrat.

La distorsion peut être déterminée à partir d'un calcul purement élastique, en imposant une déformation inhérente dans chacune des couches déposées (cf. figure 6.5).

Après découpe du support, un retour élastique est généré. Une analyse inverse est ensuite effectuée pour identifier la déformation inhérente à l'origine d'un tel retour élastique comme décrite ci-dessous.

6.2.2 Analyse inverse

L'analyse inverse de la déformation inhérente implique dans un premier temps la mise en œuvre du modèle direct. Dans ce modèle, l'éprouvette est divisée en macro-couches (cf. figure 6.5). L'activation d'une macro-couche s'accompagne d'un calcul élastique visant à déterminer un nouvel état d'équilibre mécanique suite à la perturbation introduite au travers de la

déformation inhérente considérée.

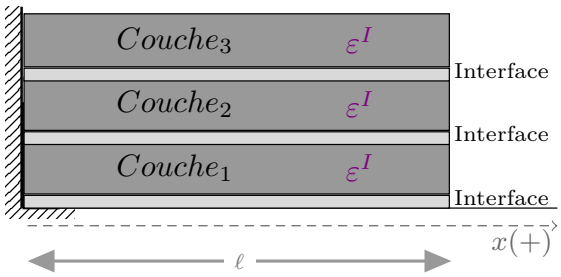


FIGURE 6.5 – Représentation schématique de l'état initial de déformation.

Après activation de la totalité des macro-couches, une dernière étape reproduisant la découpe partielle de l'échantillon (voir la figure 6.8) permet de quantifier le retour élastique.

Comme abordé dans [3, 4], pour un déplacement vertical mesuré $\tilde{U}_z$, on cherche la fonction $\varepsilon^I \equiv \varepsilon^\star \in \Omega_p$, avec Ω_p l'espace paramétrique. Considérons une fonction coût qui mesure l'erreur commise par le modèle direct lors de la prévision du retour élastique obtenue expérimentalement. En considérant une mesure de l'erreur en norme L_2, cette fonction s'écrit :

$$\mathcal{J}\left(\varepsilon^I\right) = \left(\sum_{i=1}^{N}\left(\tilde{U}_z(x_i) - U_z(x_i, \varepsilon^I)\right)^2\right)^{1/2} \tag{6.14}$$

où $\mathcal{J}$ est la fonction coût, N le nombre de points de mesure, U_z est le déplacement vertical calculé et x_i représente la position du $i^{\text{ème}}$ point de mesure. La déformation inhérente recherchée, notée $\tilde{\varepsilon}^I$, est définie dans l'espace paramétrique Ω_p, et minimise la fonction coût $\mathcal{J}$,

$$\tilde{\varepsilon}^I = \underset{\varepsilon^I \in \Omega_p}{\operatorname{argmin}}\, \mathcal{J}\left(\varepsilon^I\right) \tag{6.15}$$

Certaines hypothèses peuvent être émises concernant la déformation inhérente :
— La direction de déformation principale considérée au niveau de la macro-couche coïncide avec la direction principale de construction de l'échantillon, notée **x**, **y** et **z**.
— La déformation inhérente est considérée comme identique pour toutes les macro-couches. Un historique thermo-mécanique similaire ou comparable est présent dans chaque macro-couche.

Dans ce cadre là, le tenseur de déformation inhérente exprimé dans la base principale, noté ε^I, prend la forme

$$\varepsilon^I = \begin{pmatrix} \varepsilon_x^I & 0 & 0 \\ 0 & \varepsilon_y^I & 0 \\ 0 & 0 & 0 \end{pmatrix}_{(\vec{e}_x, \vec{e}_y, \vec{e}_z)} \tag{6.16}$$

Deux composantes sont donc à priori inconnues et doivent être identifiées par analyse inverse. Cela nécessite la définition d'un plan d'expérience numérique qui permet une explo-

ration de l'espace paramétrique dans lequel sont définies ces composantes. La fonction coût $\mathcal{J}$ est évaluée pour chaque couple de composantes de déformation inhérente défini dans le plan d'expérience numérique, et une surface de réponse est construite. Un minimum global correspondant aux composantes de déformation inhérente recherchées est finalement déterminé.

6.3 Calibration expérimentale

6.3.1 Fabrication et contrôle des éprouvettes

Mesurer expérimentalement les contraintes résiduelles présentes dans les pièces obtenues par fabrication additive repose généralement sur une approche indirecte consistant à fabriquer une éprouvette dédiée puis à la découper et à quantifier la distorsion.

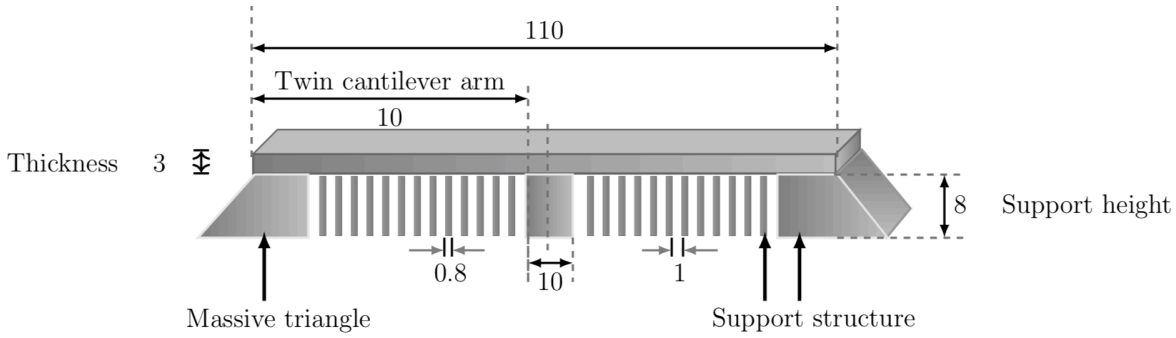


FIGURE 6.6 – Géométrie utilisée pour la réalisation du "Twin Cantilever" [19].

L'éprouvette de référence en SLM est l'éprouvette peigne dont les caractéristiques géométriques sont données sur la figure 6.6. Plusieurs travaux ont été réalisés pour étudier la distorsion de ces éprouvettes. Setien et al. [97] utilisent cette éprouvette fabriquée par différentes stratégies de balayage afin d'évaluer l'impact de la trajectoire de lasage sur la distorsion obtenue. Pour ce faire, une méthodologie empirique est présentée pour déterminer les déformations inhérentes. Ces déformations sont calculées en tenant compte des stratégies de regroupement de couches qui pourraient également être adoptées dans le modèle numérique. Kruth et al. [56] ont utilisé la méthode de courbure de pont afin d'identifier la contrainte résiduelle, tout en comparant plusieurs paramètres y compris les différents modèles de balayage laser, la longueur de ces derniers ainsi que leur angle de rotation. Les résultats des expériences présentés illustrent l'amplitude de la contrainte résiduelle thermique générée au sein du composant.

Buchbinder et al. [19] ont exploré la méthode de réduction de la distorsion en utilisant un substrat préchauffé au cours du processus de fabrication. Ce travail vise à étudier la structure du support la plus adaptée pour minimiser la déformation du Twin Cantilever. Cette déformation a été mesurée après retrait du substrat.

En règle générale, plusieurs éprouvettes sont construites avec différentes épaisseurs de poutre (figure 6.7) et, dans certains cas, des orientations différentes de dépôt. En procédant ainsi, l'analyse inverse repose sur une information plus riche et est plus susceptible de produire une "solution unique".

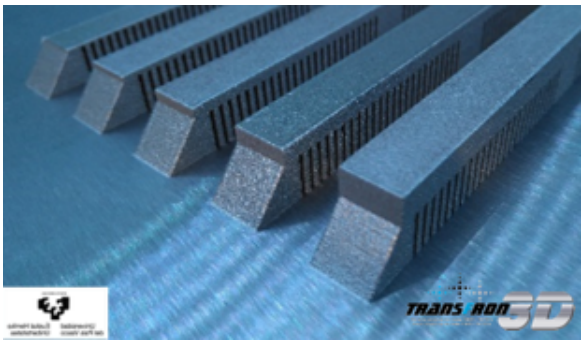


FIGURE 6.7 – "Twin cantilever" sur le support [19]

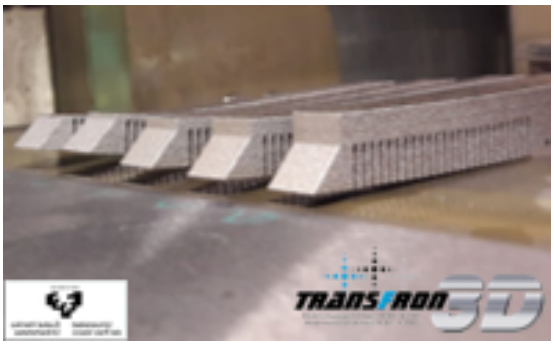


FIGURE 6.8 – Retour élastique résultant après découpe [48]

Après la fabrication, les pièces sont partiellement séparées du support par électroérosion (EDM). Ce procédé a pour principal avantage de ne pas introduire de déformation plastique supplémentaire dans la partie coupée.

La découpe des supports se traduit par une flexion de l'éprouvette (voir la figure 6.8). Ce retour élastique est dû à la relaxation des contraintes générées dans la pièce lors de sa fabrication. La contrainte résiduelle est elle-même une conséquence de la déformation inhérente générée dans le matériau. Donc, finalement, le retour élastique peut être directement relié à la déformation inhérente.

Le déplacement vertical de l'éprouvette après découpe est mesuré par une méthode optique (scan laser) ou sur une machine à mesurer tridimensionnelle (MMT) équipée d'un palpeur. Les données collectées servent de référence pour l'identification inverse de la déformation inhérente par simulation numérique.

6.4 Équations d'équilibre de l'approche de déformation inhérente

6.4.1 Formulation du problème

En s'inspirant du modèle introduit dans 6.1.3, nous proposons la formulation faible du problème d'équilibre pour la première couche. Soit une fonction test notée v, on a

$$\int_{\Omega_0} \varepsilon^\star : \mathcal{C} : \varepsilon\left(u_{(0)}\right)\ d\Omega_0 = \int_{\omega_0} \varepsilon^\star : \mathcal{C} : \varepsilon^I\ d\omega_0 \qquad \text{avec } \Omega_0 = \omega_0 \tag{6.17}$$

En généralisant la formulation pour l couches déposées. Soit Ω_i le domaine qui détermine les couches fabriquées, ω_i représente la couche active à l'instant d'observation. On établit la formulation suivante

$$\sum_{i=0}^{l} \int_{\Omega_i} \varepsilon^\star : \mathcal{C} : \left(\varepsilon \left(u_{(l)} \right) - \varepsilon \left(u_{(l-1)} \right) \right) \, d\Omega_i = \int_{\omega_l} \varepsilon^\star : \mathcal{C} : \varepsilon^I \, d\omega_l \tag{6.18}$$

avec $u_{(l)}$ le champ de déplacement, ε^I le vecteur de déformation inhérente, $\varepsilon(u) = \dfrac{1}{2}\left(\nabla u + \nabla^T u\right)$, $\varepsilon(u) = \varepsilon^e + \varepsilon^I$, et $\varepsilon^\star = \varepsilon(v)$.

6.4.2 Discrétisation

Le champ de déplacement doit vérifier les conditions cinématiques au niveau des interfaces. La méthode de Nitsche permet d'assurer la continuité en impliquant en plus des termes de pénalités [12] sur le maillage des dérivées normales à travers l'interface [77]. Chouly et al. [25] proposent une méthode inspirée de la méthode de Nitsche et qui traite les conditions aux limites ou aux interfaces au sens faible en utilisant un terme de pénalité cohérent pour la résolution des problèmes de contact sans frottement en élasticité. La méthode introduite dans cet article ne nécessite aucun autre inconnu à savoir le multiplicateur de Lagrange contrairement à la méthode introduite par Fritz et al. dans [35].

Dans cette partie, nous utilisons une approximation discontinue [101] en utilisant des termes de stabilisation [46].

Définition 6.1
Soit $\mathcal{T}_h$ un maillage de Ω tel que

M1. $\overline{\Omega} = \underset{K \in \mathcal{T}_h}{\cup} \overline{K}$, *l'intersection de deux éléments distincts est vide* $\cap K = \emptyset$.

M2. *Le rapport entre le diamètre d'un élément K et le diamètre du cercle inscrit dans K doit respecter la relation [26]*

$$\exists \sigma > 0, \, \forall \mathcal{T} \in \mathcal{U}_\sigma, \, \forall K \in \mathcal{E}_\mathcal{T} \qquad \frac{h_K}{\rho_K} \leq \sigma; \quad h = \max_{K \in \mathcal{T}_h} h_K$$

Avec $\mathcal{U}_\sigma$ représente une famille régulière de triangulations[1]*. Cette condition interdit les triangles/tétraêdres dégénérés (ie :3 sommets alignés en 2D), car ce serait de mesure nulle. Pour chaque élément K, on introduit $h_{|_K}$ comme suit [46]*

$$h_{|_K} = \begin{cases} \min\{h_{K^+}, h_{K^-}\} & E \subset \partial K^+ \cup \partial K^- \\ h_K & E \subset \partial K \cup \partial \Omega \end{cases} \tag{6.19}$$

M3. *L'intersection distincte entre Γ et la limite de K (∂K), divise chaque élément en 2*

1. Une famille régulière de maillages ie : il n'existe pas de triangle trop aplati

parties, et coupe d'autres arêtes de K dans un point chacun.

M4. *Soit $\Gamma_{K,h}$ l'intersection entre l'élément K et le plan. Le lien entre la fonction Γ_K et la longueur Γ_h est donné par [13]*

$$\Gamma_K := \Big\{(\xi,\eta);\ 0<\xi<|\Gamma_h|,\ \eta=\delta(\xi)\Big\}$$

On considère l'espace discret

$$V_h = \{u \ \mid \ v_{|_K} \in \mathbb{P}^k(K) \quad \forall K \in \mathcal{T}_h\} \subset V$$

On écrit la formulation faible du problème d'équilibre en considérant le saut entre les différentes couches, on multiplie par une fonction test v dans un espace V et on intègre sur une somme d'éléments du maillage. On représente le terme de droite pour une seule couche par l'équation suivante

$$\begin{aligned}\sum_{K\in\mathcal{T}_h}\int_K \sigma(u):\varepsilon(v)dx &- \frac{1}{2}\sum_{E\in\mathcal{E}^{int}}\int_E\Big(\{\sigma(u)\}:[\![v]\!]+\{\sigma(v)\}:[\![u]\!]\Big)ds\\ &+\frac{\mu}{2}\sum_{E\in\mathcal{E}^{int}}\int_E\Big(\frac{\gamma_\mu}{h}[\![u]\!][\![v]\!]\Big)ds+\frac{\lambda}{2}\sum_{E\in\mathcal{E}^{int}}\int_E\Big(\frac{\gamma_\lambda}{h}[\![u.n]\!][\![v.n]\!]\Big)ds\\ &-\int_{\partial\Gamma_D}\Big(\sigma(u).n.v.n+\sigma(v).n.u.n\Big)ds\\ &+\mu\int_{\partial\Gamma_D}\frac{\gamma_\mu}{h}u.vds+\lambda\int_{\partial\Gamma_D}\frac{\gamma_\lambda}{h}u.n.v.nds=0\end{aligned} \tag{6.20}$$

Avec $\{\sigma(u)\} = \frac{1}{2}\Big(\sigma(u)^+ + \sigma(u)^-\Big)$ et $\mathcal{E}^{int}$ le bord intérieur, et $[\![\sigma]\!].n = \sigma^+n^+ - \sigma^-n^-$ représente l'écart entre les contraintes de chaque côté de l'interface.

Les termes en bleu dans l'équation 6.20 correspondent aux termes de saut et au flux numérique au niveau des interfaces. Les fonctions de stabilisation γ_μ et γ_λ permettent de pénaliser les sauts de u et v sur les interfaces de $\mathcal{T}_h$. Les trois derniers termes sont les conditions aux limites définies sur $\partial\Gamma_D$.

6.5 Exemple d'application et validation

L'objectif ici est de valider l'approche précédente sur une éprouvette de référence 6.6 programmée dans FreeFem++ en la comparant aux résultats obtenus par Virfac. Un exemple de validation plus simple a été publié dans [4]. Pour estimer la déformation plastique correcte ou la déformation inhérente induite dans un composant complet lors de sa construction par le procédé SLM, une procédure de calibration est réalisée. Dans la méthode de déformation inhérente, les micro-couches réelles sont remplacées par des macro-couches dont l'épaisseur peut atteindre 2 mm comme illustré par la figure 6.9.

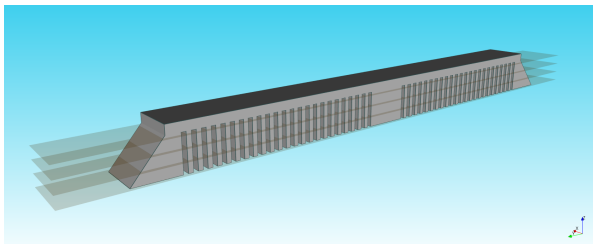
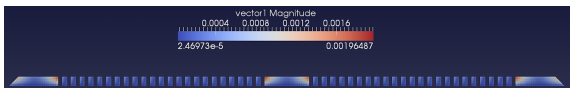
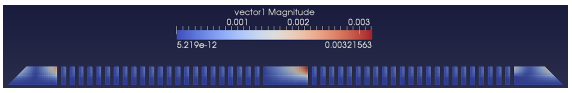


FIGURE 6.9 – La mise en place de la méthode des déformations inhérentes.

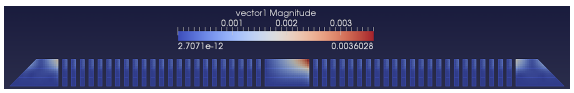
La déformation inhérente est considérée comme identique pour toutes les macro-couches, et isotrope. La déformation inhérente utilisée dans le premier test est $\varepsilon = -0.0005$. La figure 6.10 représente les champs de déplacement au niveau des différentes macro-couches. Le tableau 6.1 représente une comparaison des résultats obtenus et attendus en utilisant le logiciel Virfac. On remarque que les résultats sont similaires.



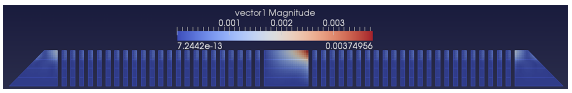
(a) Déplacement u_x 1ère couche.



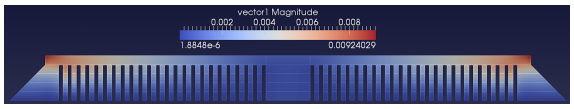
(b) Déplacement u_x 2ème couche.



(c) Déplacement u_x 3ème couche.



(d) Déplacement u_x 4ème couche.



(e) Déplacement u_x 5ème couche.

FIGURE 6.10 – Résolution du problème d'élasticité en utilisant l'éprouvette de référence.

Le déplacement vertical maximum atteint 0.118 mm dans cette première simulation comme l'illustre la figure 6.11 alors que la valeur cible, issue d'un calcul en utilisant logiciel de VirFac est d'environ 0.111 mm (cf. table 6.1).

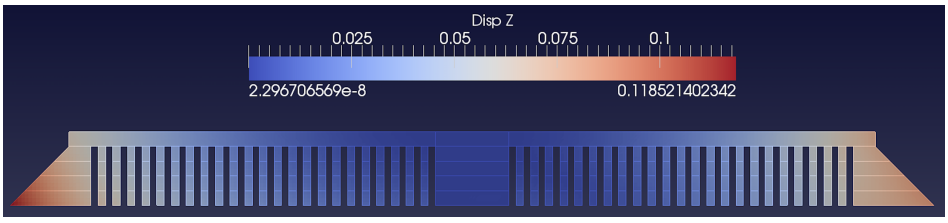


FIGURE 6.11 – Retour élastique U_z pour la simulation en FEM, avec $\epsilon_x^I = -0.0005$, $\epsilon_y^I = 0$.

	$\varepsilon_x^I = -0.0025$		$\varepsilon_x^I = -0.0005$	
	réf	FF++	réf	FF++
max $u_{(0)}$	-0.0094	-0.0098	-0.002151	-0.00196487
max $u_{(1)}$	-0.0153	-0.0160	-0.00362	-0.00321563
max $u_{(2)}$	-0.0196	-0.0180	-0.00389	-0.0036028
max $u_{(3)}$	-0.021	-0.0187	-0.004022	-0.00374956
max $u_{(4)}$	-0.04	-0.0462	-0.01047	-0.00924029
max springback	0.5	0.55	0.111	0.118

TABLE 6.1 – Récapitulatif des valeurs clés entre la simulation de référence (Virfac) et la simulation FEM pour différentes valeurs de ε_x^I. $\epsilon_y^I = 0$.

Dans le cadre de cette thèse, nous nous intéressons à la méthode de déformation inhérente introduite par Ueda (cf. 6.1.1) pour la simulation du soudage, que nous adaptons à la simulation du SLM. Le modèle analytique que nous proposons dans la section 6.1.3 nous permet d'estimer la distorsion d'une pièce après sa découpe du substrat à partir d'un calcul purement élastique. On programme cette méthode en FreeFem++ que l'on valide en comparant les résultats obtenus aux champs de déplacements récupérés d'une simulation en utilisant Virfac. On s'aperçoit que notre méthode de calcul s'avère prometteuse pour déterminer les déformations inhérentes.

L'approche décrite jusqu'à présent permet de mettre en place un plan d'expérience utilisable pour trouver ε^I optimal. Cependant, la phase de calibration nécessite alors d'effectuer plusieurs simulations. Pour ce, nous introduirons dans la partie un modèle élastique paramétrique permettant une évaluation instantanée de la déformation inhérente. Les composantes inconnues de la déformation inhérente sont traitées comme des dimensions supplémentaires du problème résolu.

6.6 Modèle paramétrique

6.6.1 Mise en place du modèle réduit

Afin de contrôler la distorsion de la pièce fabriquée, tout en s'affranchissant du verrou lié au temp de calcul, on propose un modèle numérique où on considère les composantes de déformation inhérente recherchées comme des paramètres du problème 6.18. Le problème à résoudre est ainsi défini en dimension $N + 2$ (où N représente la dimension de l'espace physique), et sa résolution repose sur une approche PGD [39].

En PGD, la solution du problème est calculée en temps réel par enrichissement progressif (ajout successif de modes), et exprimée sous forme séparée. Cette méthode comme l'illustre la figure 6.12 est réalisée en deux temps. Dans un premier temps, une phase de calcul *offline* en amont de l'utilisation du modèle réduit permet d'effectuer des simulations en utilisant un jeu de paramètres. Les résultats obtenus sont stockés afin de construire la base initiale. Après ce

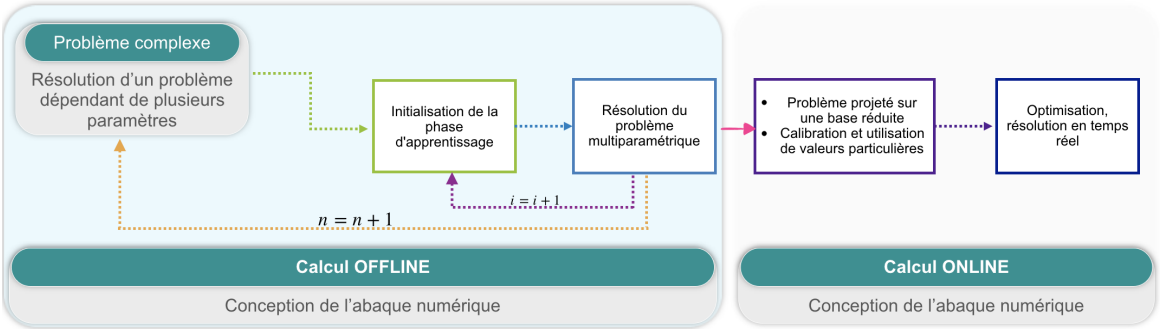


FIGURE 6.12 – Illustration de l'approche en deux temps du modèle multiparamétrique.

calcul chronophage, on espère effectuer des simulations en temps réel pendant la phase *online* en lançant plusieurs fois le modèle réduit lors des procédures itératives. Ainsi, le champ de déplacement solution du problème 6.18, est défini à l'itération n comme suit

$$u^n\left(X, \varepsilon_x^I, \varepsilon_y^I\right) = \sum_{i=1}^{n-1} X^i\left(x, z\right) \mathcal{E}_x^i\left(\varepsilon_x^I\right) + R(x, z) S(\varepsilon_x^I) \tag{6.21}$$

où X^i représente l'ensemble des fonctions défini dans l'espace physique, $\mathcal{E}_x^i$ sont les fonctions définies dans l'espace paramétrique et sont associées aux composantes ε_x^I du tenseur de déformation inhérente.

6.6.2 Solution par PGD

L'accroissement virtuel cinématiquement admissible des déplacements s'exprime sous la forme séparée en considérant un seul paramètre $\varepsilon(u) = \varepsilon(u^n)$. En injectant la forme séparée 6.21 dans la formulation variationnelle 6.17 on obtient

$$\begin{cases} \displaystyle\int_\Omega \varepsilon(R) \cdot S : \mathcal{C} : \varepsilon(R^\star) \cdot S d\Omega = -\int_\Omega \sum_{i=1}^{n-1} \varepsilon(X^i) \cdot \mathcal{E}_x^i : \mathcal{C} : \varepsilon(R^\star) \cdot S d\Omega + \int_\omega \varepsilon(R^\star) \cdot S : \mathcal{C} : \varepsilon^I \; d\omega \\ \displaystyle\int_\Omega \varepsilon(R) \cdot S : \mathcal{C} : \varepsilon(R) \cdot S^\star d\Omega = -\int_\Omega \sum_{i=1}^{n-1} \varepsilon(X^i) \cdot \mathcal{E}_x^i : \mathcal{C} : \varepsilon(R) \cdot S^\star d\Omega + \int_\omega \varepsilon(R) \cdot S^\star : \mathcal{C} : \varepsilon^I \; d\omega \end{cases}$$

En considérant toutes les fonctions en ε_x^I connues, on obtient

$$R^{\star T}\left[\int_{\Omega_X} \nabla N : \mathcal{C} : \nabla N^T d\Omega_X\right] R \cdot S^T\left[\int_{\Omega_{\varepsilon_x^I}} \mathbb{M}\mathbb{M}^T d\Omega_{\varepsilon_x^I}\right] S = R^{\star T} \int_{\Omega_X} \nabla N : \mathcal{C} : \varepsilon^I - \mathcal{R}^{n-1}$$

avec $\mathcal{R}^{n-1} = \sum_{i=1}^{n-1} R^{\star T}\left[\int_{\Omega_X} \nabla N : \mathcal{C} : \nabla N^T d\Omega_X\right] X^i \cdot S^T \left[\int_{\Omega_{\varepsilon_x^I}} \mathbb{M}\mathbb{M}^T d\Omega_{\varepsilon_x^I}\right] \mathcal{E}_x^i$ Ensuite on suppose les fonction en (x, y) connues et on utilise l'algorithme 2 du point fixe. On détermine le critère d'arrêt du point fixe comme suit

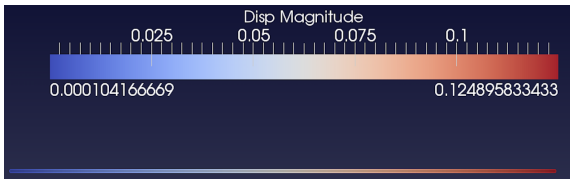
$$\int_\Omega (R_{new}S_{new} - R_{old}S_{old})^2 \, d\Omega \; < \; \epsilon_{fp}$$

L'étape d'enrichissement continue jusqu'à l'obtention de la convergence suivante

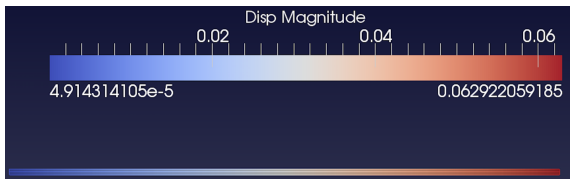
$$\frac{\int_\Omega (\varepsilon(u^\star) : \mathcal{C} : \varepsilon(u^n) - \varepsilon(u^\star) : \mathcal{C} : \varepsilon(u^{n-1}))^2 \, d\Omega}{\int_\Omega (\varepsilon(u^\star) : \mathcal{C} : \varepsilon(u^{n-1}))^2 \, d\Omega} \; < \; \epsilon_{cv}$$

6.7 Résultats

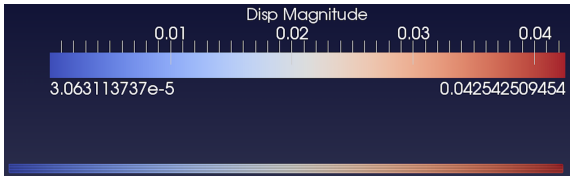
En première approche, cette stratégie est validée sur une géométrie 2D simplifiée, à savoir un domaine rectangulaire $[0, 50\,mm] \times [0, 1\,mm]$ composé de 4 macro-couches d'épaisseur $0.25\,mm$ déposées dans la direction Z. Le matériau considéré est l'acier dont les propriétés élastiques sont : $E = 210\,GPa$, $\nu = 0.3$.



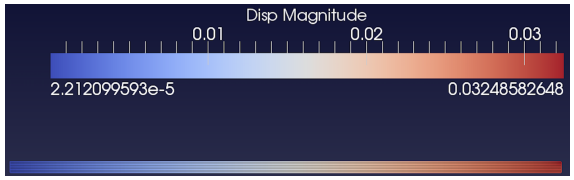
(a) Déplacement u_x 1ère couche.



(b) Déplacement u_x 2ème couche.



(c) Déplacement u_x 3ème couche.



(d) Déplacement u_x 4ème couche.

FIGURE 6.13 – Résolution par PGD du problème.

La figure 6.13 présente les champs de déplacement obtenus pour 1 à 4 couches en utilisant la méthode sus-mentionnée. Ces résultats peuvent être analytiquement vérifiés comme cité dans la section 6.1.2. Le déplacement de la première couche est donné en utilisant la formule 6.5, on a donc

$$\varepsilon^I = \frac{\ell_1 - \ell_0}{\ell_0} \implies u_1 = \ell_1 - \ell_0 = \varepsilon^I \ell_0 = -0.0025 \times 50 = -0.125$$

De même pour la deuxième couche, en considérant l'équation 6.6

$$\frac{\varepsilon^I}{2} = \frac{\ell_2 - \ell_1}{\ell_1} \implies u_2 = u_1 + \ell_2 - \ell_1 = \varepsilon^I \ell_0 \left(1 + \frac{1}{2}(\varepsilon^I + 1)\right)$$
$$= -0.0025 \times 50 \left(1 + \frac{1}{2}(-0.0025 + 1)\right) = -0.187$$

De la même manière, on peut généraliser pour n'importe quel nombre de couche

$$u_N = \varepsilon^I \ell_0 \left(1 + \frac{1}{2}(\varepsilon^I + 1) + \cdots + \frac{1}{N}(\varepsilon^I + 1)\right)$$

La figure 6.14 illustre le retour élastique U_z calculé à partir de l'algorithme PGD susmentionné.

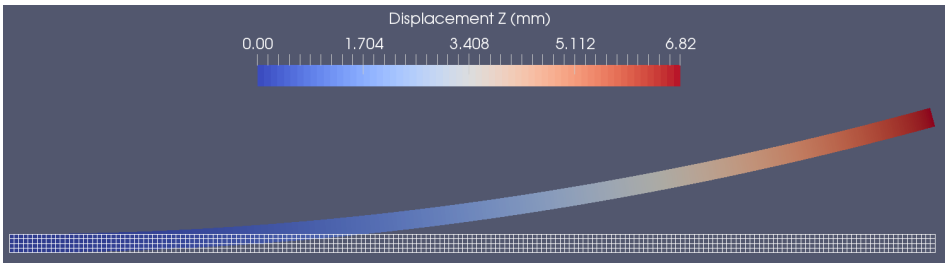


FIGURE 6.14 – Retour élastique U_z pour la simulation en PGD, avec $\epsilon_x^I = -0.0025$, $\epsilon_y^I = 0$.

La Table 6.2 récapitule pour 3 cas d'étude distincts, le déplacement horizontal maximal U_x en cours de procédé et le retour élastique siuvant z après découpe partielle, obtenus en utilisant le logiciel de GeonX VirFac d'une part et l'approche paramétrique PGD d'autre part. Les valeurs sont exprimées en millimètres.

	$\varepsilon_x^I = -0.0025$		$\varepsilon_x^I = -0.001$		$\varepsilon_x^I = -0.0005$	
	réf	pgd	réf	pgd	réf	pgd
max u_0	-0.125	-0.125	-0.050	-0.050	-0.025	-0.025
max u_1	-0.187	-0.187	-0.075	-0.075	-0.037	-0.037
max u_2	-0.229	-0.229	-0.091	-0.091	-0.046	-0.046
max u_3	-0.259	-0.260	-0.104	-0.104	-0.052	-0.052
max springback	6.82	6.81	2.75	2.74	1.37	1.38

TABLE 6.2 – Récapitulatif des valeurs clés pour différentes valeurs de ε_x^I. $\varepsilon_y^I = 0$.

Les résultats obtenus par les 2 approches sont très similaires. Le modèle paramétrique semble à même de reproduire de façon fidèle le comportement prédit par le modèle direct "standard". Néanmoins, une validation plus avancée sur la géométrie réelle de l'éprouvette peigne est indispensable.

6.8 Conclusions et perspectives

L'utilisation de la PGD afin de calculer un champ de déplacement paramétrique dans l'échantillon fabriqué, en considérant la composante de contrainte inhérente inconnue comme une dimension supplémentaire du problème à résoudre, permet de mettre en place un abaque numérique durant la phase *off-line*.

La méthode a été validée sur un modèle simplifié. L'objectif par la suite est de considérer chacune des composantes de déformations inhérentes comme étant une fonction dépendante de l'épaisseur de la couche. Puis implémenter la partie enrichissement sur GPU ce qui permet éventuellement d'augmenter le nombre de paramètres.

Quatrième partie

Calcul haute performance

Chapitre 7

Développement d'une librairie scientifique

L'application de la PGD sur la simulation des procédés SLM entraîne une multiplication du nombre d'opérateur. Il devient important d'avoir une implémentation efficace. L'objectif principal ici est d'explorer l'avantage qu'engendre l'implémentation de la PGD sur GPU.

Sommaire

7.1 Motivations

Comme nous l'avons vu respectivement aux chapitres 4 et 5, les algorithmes de réduction de modèles permettent la simulation de la thermique tout en réduisant la complexité à la fois en terme de calcul et en terme de quantité de données. Néanmoins, au cours de la simulation du procédé SLM, le nombre d'opérateurs croît linéairement avec le nombre de couches, ce qui a un impact sur le coût des différentes opérations de notre logiciel principalement le produit matrice-vecteur et le produit vecteur-vecteur. Dans le cas d'une décomposition sur des domaines 1D, le coût des différentes opérations numériques est plus élevé que le coût de l'inversion du système. Cependant, pour accroître encore plus la vitesse des simulations en temps réel tout en considérant des problèmes plus proches de la réalité, et en obtenant le meilleur compromis entre précision et temps de calcul, on utilise alors le calcul intensif.

«Les architectures parallèles longtemps délaissées au détriment des architectures séquentielles à cause de la complexité de leur programmation émergent enfin» [17]. «Les limites physiques des machines poussent les fabricants de processeurs à orienter leur production vers une multiplication du nombre de cœurs. La course aux performances menée par cette nouvelle lignée de processeurs multi-cœurs a été rejointe par les processeurs graphiques (GPU)» [17] et leur puissance de calcul phénoménale. Mais, la réelle révolution n'a eu lieu qu'en 2004 lorsque NVIDIA a élargi le domaine d'application des cartes graphiques en permettant leur exploitation au calcul scientifique.

Mais ce n'est qu'à partir de 2007 que la première version du langage de programmation CUDA (*Compute Unified Device Architecture*) a été mise à disposition des programmeurs afin de permettre un accès direct à l'architecture du périphérique en utilisant le langage C.

Les GPUs deviennent donc attractifs pour le calcul intensif, et on peut espérer tirer parti de ces accélérateurs intéressants pour leur grande puissance de calcul, leur prix modeste, ainsi que pour leur faible consommation électrique.

Le calcul haute performance nécessite de déterminer d'une part le nombre de ressources à adapter au problème traité, de comparer les différentes implémentations et d'en mesurer les performances, pour ce, on introduit dans ce qui suit la loi d'Amdahl et la loi de Gustafson-Barsis. Ensuite, une bonne compréhension de l'architecture et spécification propre à chaque carte graphique notamment au niveau des performances en double précision est proposée.

7.1.1 Scalabilité

7.1.1.1 Loi d'Amdahl

Le comportement d'un logiciel est modélisé par la loi d'Amdahl [1]. Soit $\alpha \in [0, 1]$ la portion de code parallélisable, le facteur d'accélération (*Speedup*) est le rapport entre le temps de calcul du programme séquentiel et le temps de calcul du programme sur p processeurs

1. Gene Myron Amdahl(1922-2015), informaticien et entrepreneur américain.

$$S_\alpha(p) = \frac{T_S}{T_p} = \frac{T_S}{(1-\alpha)T_S + \frac{\alpha T_S}{p}} = \frac{1}{1-\alpha+\frac{\alpha}{p}} \underset{p\to\infty}{\longrightarrow} \frac{1}{1-\alpha}$$

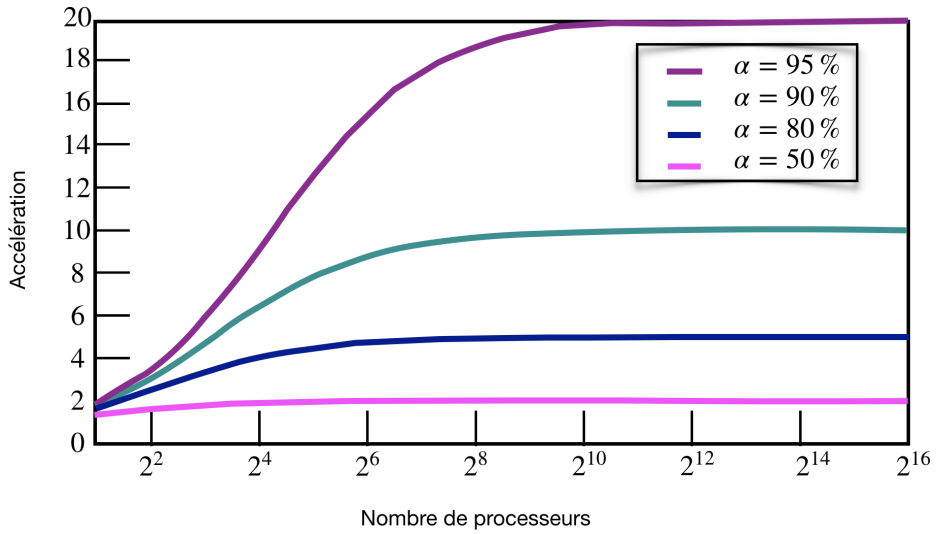


FIGURE 7.1 – Évolution selon la loi d'Amdahl du facteur d'accélération en fonction du nombre de processeurs.

$(1-\alpha)T_S$ représente la tâche d'exécution en séquentielle, à laquelle est ajoutée la partie $(\alpha T_S/p)$ puisque cette partie est accélérée à l'aide de p processeurs. En augmentant le nombre de processeurs, le rapport tend vers $\frac{1}{1-\alpha}$, cela signifie que lorsque la partie parallélisable devient négligeable, l'accélération maximale est bornée par la partie non parallélisable du code. La figure 7.1, représente le *speedup* en fonction d'un certain nombre de processeurs variant entre 1 et 65000. La courbe magenta représente un programme dont la partie parallélisable est fixée à 50%, dans ce cas, augmenter les ressources n'est pas utile puisque l'évolution du facteur d'accélération stagne à partir de 2 processeurs. En effet, ici le programme perd en efficacité en augmentant le nombre de ressources en plus de la consommation énergétique qu'engendre l'utilisation de plusieurs processeurs.

7.1.1.2 Loi de Gustafson-Barsis

Cependant, la loi de Gustafson-Barsis permet de contourner cet obstacle en fixant la taille du problème par cœur de calcul ou par ressource utilisé. Ici, l'évolution du facteur d'accélération augmente à la fois avec l'augmentation de la taille du problème et du nombre de processeurs. Dans ce cas-là, on suppose que le temps d'exécution de la partie non parallélisable du code ne dépend pas de la taille du problème, et donc

$$S_\alpha(p) = \frac{T_S}{T_p} = \frac{\alpha p T_p + (1-\alpha)T_p}{T_p} = \alpha p + (1-\alpha) \underset{p\to\infty}{\longrightarrow} \infty$$

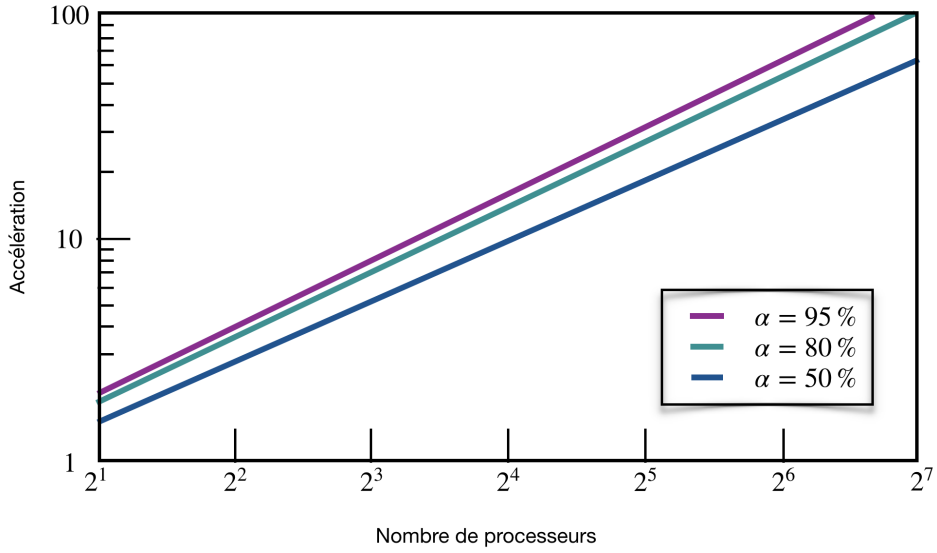


FIGURE 7.2 – Évolution selon la loi de Gustafson-Barsis du facteur d'accélération en fonction du nombre de processeurs.

Contrairement à la formule précédente, ici c'est le T_S qui est modélisé en fonction de T_p, la partie parallélisable est donc $\alpha p T_p$. C'est plus intéressant, puisque le fait d'augmenter le nombre de processeurs permet d'avoir toujours de plus en plus d'accélération. La figure 7.2, permet de mettre en évidence l'évolution du facteur d'accélération pour un code parallélisable à seulement $\alpha = 50\%$.

7.1.1.3 Conséquences

Il est à noter que la performance d'un code doit être mesurée en déterminant la métrique la plus adaptée à l'application. Pour ce, il va falloir dans un premier temps déterminer dans quel cas de figure on se place afin d'adapter le nombre de ressources au problème à résoudre. En effet, il n'est pas utile d'augmenter toujours plus le nombre de ressources si çà ne réduis pas le temps de calcul. Dans la loi d'Amdhal, l'accélération maximale est bornée par la taille du problème et la partie non parallélisable du code, il peut toutefois y avoir d'autres limitations engendrées par d'autres facteurs comme des ressources non scalables utilisés dans l'application. La sauvegarde des résultats dans un fichier sur le disque est séquentiel, d'autre part, l'utilisation d'un câble ethernet limite les performances.

Dans les années 80-90, la technologie de fabrication de CPU permettait de fabriquer de plus en plus petit et par conséquent gagner en fréquence et du coup en performances. Il y a 20 ans, il existait des processeurs en MHz, alors qu'il y a 10 ans on avait des processeurs à plusieurs GHz, il suffisait donc de changer de processeur et de passer d'un à 2 GHz, ce qui permettait de diviser en 2 les temps de calcul. Aujourd'hui, il n'est plus possible d'obtenir un *speedup* gratuit et ce, à cause de la fréquence des processeurs qui n'augmentent plus dans les nouvelles technologies.

D'autre part, étant donné qu'un programme parallèle peut être limité par des ressources séquentielles ou qui vont saturer en augmentant le nombre de ressources, il va donc falloir regarder l'architecture du matériel afin de déterminer ce qui peut limiter la performance parallèle et comment contourner cet obstacle.

7.1.2 Architecture hétérogène

Les machines actuelles disposent au moins d'un processeur et d'un accélérateur coprocesseur ou carte graphique, ce qui présente déjà deux architectures différentes. Le nombre de CPU, les branchements internes, le nombre de composante, le type de processeur,... tous ces paramètres varient d'une machine à l'autre, ce qui permet d'exploiter au mieux chaque architecture en distribuant les différentes parties du programme sur les différents processeurs ce qui permet de bénéficier des capacités spécifiques à chaque architecture. Cette implémentation sera détaillée dans ce qui suit.

La principale difficulté de l'utilisation de ces machines hétérogènes réside principalement dans le développement du code. On ne développe pas de la même manière sur CPU que sur GPU ou sur des processeurs Intel Xeon phi par exemple. Donc un code développé sur mesure pour une architecture précise risque de devenir obsolète dans un futur proche, il faut donc prévoir une stratégie pour pérenniser et maintenir ces codes.

Il existe deux grandes familles d'architectures de processeurs ; les processeurs à mémoire partagée et les processeurs à mémoire distribuée. Un processeur à mémoire partagée est composé de 3 processeurs qui ont chacun leur cache qui accèdent via un bus à une mémoire partagée. Cette famille de processeurs ne possède donc qu'une seule mémoire partagée ce qui est considéré comme un facteur limitant. En effet, si une centaine de processeurs essayent de lire des données à des endroits différents de la mémoire et si cette mémoire ne supporte pas les accès multiples ou pas suffisamment ceci conduit à un ralentissement de la mémoire.

À l'opposé, la mémoire distribuée n'est pas commune à tous les processeurs, chaque processeur possède une quantité de RAM qui lui est propre. Avec cette architecture le facteur limitant n'est plus la mémoire, puisque chaque processeur possède la sienne et peut y accéder aussi rapidement. Cependant, l'accès à la mémoire du voisin nécessite l'utilisation d'un réseau d'interconnexion qui unifie la mémoire et c'est ce qui peut causer problème en utilisant ce type d'architecture. La mémoire est distribuée mais d'un point de vue logiciel elle est unifiée c'est-à-dire qu'il n'existe qu'un seul endroit mémoire pour chaque adresse

7.1.3 Le choix de la technologie

Le choix de la technologie dépend essentiellement de l'adéquation entre les méthodes numériques utilisées pour la simulation du problème physique à résoudre, les caractéristiques et la précision numérique attendues pour les solutions ainsi que les spécificités de la machine mise à disposition.

L'idée principale est de combiner plusieurs technologies en une seule application, ce qui permet de décomposer la charge de travail. En effet, l'utilisation d'au moins deux technologies différentes implique deux niveaux de programmation ce qui conduit à découper les tâches et permet par la suite d'obtenir un logiciel plus souple. Les perspectives qu'offre une programmation hybride sont détaillées dans l'Annexe A. Cependant, dans la suite de ce travail sera uniquement présentée la programmation sur GPU, ainsi qu'une parallélisation CPU en utilisant le BLAS [2] (*Basic Linear Algebra Subprograms*).

7.2 Présentation du matériel

La série Pascal basée sur l'architecture d'un GP107, composée de 768 unités de calcul élémentaires (les cœurs CUDA), organisées en 6 Streaming Multiprocessors [3](SMX) de 48 cœurs chacun comme représentée par la figure 7.3, offre une performance en double précision allant jusqu'à 66.82 GFlops [4]. Les unités d'exécution du GPU bénéficient d'un accès à plusieurs niveaux de mémoire comme détaillé dans l'Annexe A. Les fonctionnalités de CUDA permettent de tirer parti de n'importe quel type de carte graphique (GeForce : cartes graphiques pour usage public, Quadro : cartes graphiques professionnels, Tesla : cartes dédiées au calcul scientifique...) et l'adapter au calcul scientifique.

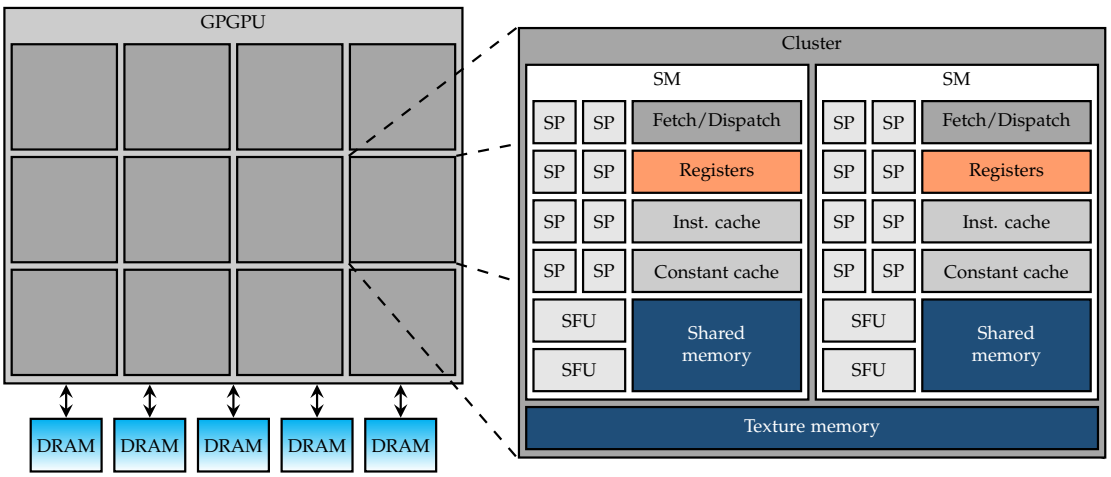
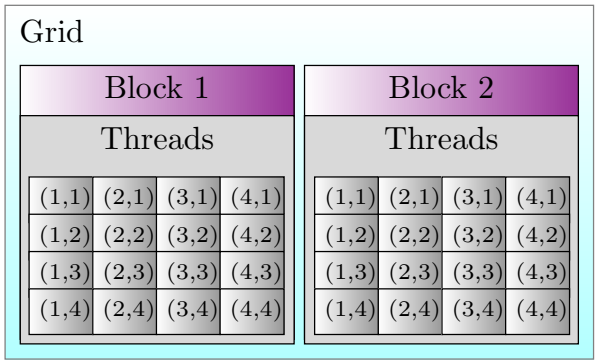


FIGURE 7.3 – Représentation visuelle d'une mémoire partagée.

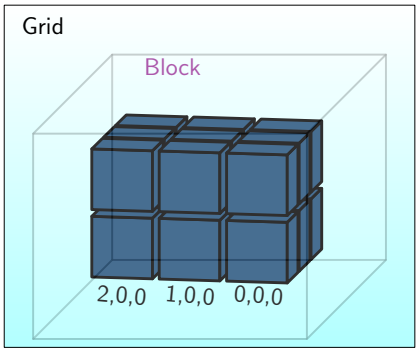
Les GPUs NVIDIA, sont tous munis de plusieurs processeurs et chacun d'entre eux possède plusieurs cœurs, chacun de ces cœurs peut faire tourner des threads de manière indépendante. Un thread est donc une tâche exécutée sur un cœur CUDA. Pour que la programmation soit

2. http://www.netlib.org/blas/
3. Streaming Multiprocessors est une entité au sein d'un processeur qui rassemble plusieurs cœurs CUDA
4. Une grandeur de mesure de la performance d'un système, elle exprimée par le nombre d'opérations en utilisant des nombres réels à virgules flottantes par secondes d'exécution.

générique quelque soit la carte graphique, ces threads sont regroupés au sein de blocs en 1D, 2D (fig. 7.4a) ou bien 3D (fig. 7.4b).



(a) Grille en 2D.



(b) Grille en 3D.

FIGURE 7.4 – Représentation d'un bloc respectivement en 2D et 3D.

Ces blocs sont eux-mêmes regroupés au sein d'une grille qui peut, elle aussi, être en 1D, 2D ou 3D, les figures (7.4a) et (7.4b) représentent ces grilles. Le nombre de threads utilisé est donc égal au nombre de threads par blocs, multiplié par le nombre de blocs par grille.

7.2.1 Organisation des threads

Les performances de CUDA résident dans le modèle logique qui partitionne les calculs en blocs de threads et les répartit sur l'architecture hautement parallèle du GPU. La programmation parallèle ne revêt pas de caractère complexe, cependant la répartition des threads qui exécutent les kernels en parallèle pose quelques difficultés [5].

Les threads CUDA sont ensuite répartis sur les SMX libres et se divisent alors par groupes de 32 en sous-groupes appelés warps (fils de chaînes). Ce sont ces warps qui jouent un rôle significatif sur l'efficacité de l'exécution lors de l'utilisation d'une structure de contrôle conditionnel. En effet, chaque thread exécute la même opération au même moment. L'organisation des threads est limitée par certains facteurs à savoir la taille maximale d'un bloc qui est de 1024 threads, la taille maximale de la grille : $(2^{32} - 1)$ blocs, et le nombre de blocs (16 blocs) qui peuvent passer en même temps sur un même SMX. En respectant ces conditions, l'identifiant unique d'un thread est défini comme une combinaison de l'indice de son bloc au sein de la grille et de l'indice du thread au sein du bloc.

5. http://info.univ-angers.fr/~richer/cuda_crs3.php

7.2.2 Streams

Par défaut, les étapes de chargement en mémoire, de calcul et de récupération du résultat se font de manière synchrone. Ce qui signifie, que le kernel ne peut pas calculer tant que l'ensemble des données n'est pas disponible sur le périphérique et le résultat n'est pas disponible avant la fin du calcul (cf. figure 7.5).

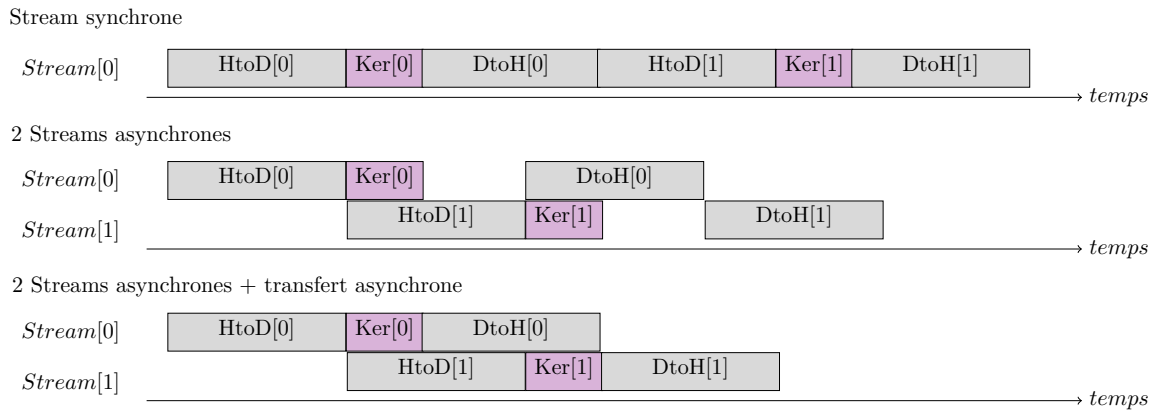


FIGURE 7.5 – Éxécution asynchrone de stream.

Il est toutefois, possible de découper ce flux de données et de calcul, en commençant le chargement d'une partie des données tout en calculant une autre partie. Lors du portage sur GPU on utilise les fonctions de copie en mode asynchrone pour ne pas rendre ce chargement bloquant pour le périphérique. Ce qui permet aussi de choisir dans quel stream réaliser le chargement. Pour la récupération du résultat l'opération est la même. Quant au lancement du kernel, il est par défaut, asynchrone mais s'exécute sur le stream default. Dans la version asynchrone on voit que le chargement des données est toujours séquentiel mais que le kernel peut s'exécuter en parallèle d'une opération mémoire.

7.3 Algorithme PGD pour la thermique stationnaire

7.3.1 Mises en équation

Soit Ω de $\mathbb{R}^d$ un domaine borné, on considère ici une membrane tenue sur le bord de ce domaine, et on cherche le déplacement vertical u vérifiant l'équation aux dérivées partielles suivante :

$$-\Delta u(x_1, x_2) = f(x_1, x_2); \qquad (x_1, x_2) \in \Omega = \Omega_{X_1} \times \Omega_{X_2} \tag{7.1}$$

La condition limite qui traduit le fait que le déplacement est nul sur le bord s'écrit

$$u = 0 \quad \text{sur} \quad \partial\Omega$$

La méthode PGD utilisée dans ce calcul est similaire à la méthode précédemment introduite au chapitre 5. La solution est recherchée sous forme séparée

$$u(x_1,x_2)=\sum_{j=1}^{n}\sigma_j(u)\prod_{k=1}^{2}\phi_k^j(x_k),\qquad u^\star(x_1,x_2)=\sum_{j=1}^{n}\sigma_j^\star(u)\prod_{k=1}^{2}\phi_k^j(x_k).$$

Où $\sigma_j(u)$ est le $j^{\text{ème}}$ degré de liberté de u. On injecte alors cette écriture dans la formulation variationnelle du problème 7.1, et on obtient :

$$\int_\Omega \nabla\Big(\sum_{j=1}^{n}\sigma_j^\star\prod_{k=1}^{2}\phi_k^j(x_k)\Big)\nabla\Big(\sum_{j=1}^{n}\sigma_j\prod_{k=1}^{2}\phi_k^j(x_k)\Big)d\Omega=\int_\Omega\Big(\sum_{j=1}^{n}\sigma_j^\star\prod_{k=1}^{2}\phi_k^j(x_k)\Big)f \tag{7.2}$$

Le terme source peut aussi être sous forme séparée

$$f(x_1,x_2)=\sum_{j=1}^{q}\prod_{k=1}^{2}f_k^j(x_k)$$

On considère

$$f_1(x_1)=-2x_1(x_1-1)\qquad f_2(x_2)=-2x_2(x_2-1)$$

On suppose connus les "$n-1$" premiers modes, et on cherche à déterminer le $n^{\text{ème}}$ mode, comme suit

$$\begin{aligned}u^n(x_1,x_2)=&\sum_{j=1}^{n-1}\sigma_j(u)\prod_{k=1}^{2}\phi_k^j(x_k)+\prod_{k=1}^{2}W_k(x_k)\\=&\quad u^{n-1}+\prod_{k=1}^{2}W_k(x_k)\end{aligned} \tag{7.3}$$

Ce qui se traduit sous forme matricielle en posant N_{X_1} et N_{X_2} les vecteurs de fonctions de forme associées à chaque espace. Les matrices de masse et de raideur sont donc définies comme suit

$$\begin{aligned}A_1^2=\ M_{X_1}=&\int_{\Omega_{X_1}}N_{X_1}N_{X_1}^T dx_1 & A_1^1=\ K_{X_1}=&\int_{\Omega_{X_1}}dN_{X_1}dN_{X_1}^T dx_1;\\ A_2^1=\ M_{X_2}=&\int_{\Omega_{X_2}}N_{X_2}N_{X_2}^T dx_2 & A_2^2=\ K_{X_2}=&\int_{\Omega_{X_2}}dN_{X_2}dN_{X_2}^T dx_2.\end{aligned} \tag{7.4}$$

Ce qui implique

$$\begin{aligned}\sum_{i=1}^{n_A}\sum_{j=1}^{n-1}\Big(\sigma_j W_k^{\star T}A_k^i\phi_k^j\prod_{\substack{l=1\\l\neq k}}^{2}W_l^T A_l^i\phi_l^j\Big)+&\sum_{i=1}^{n_A}\Big(W_k^{\star T}A_k^i W_k\prod_{\substack{l=1\\l\neq k}}^{2}W_l^T A_l^i W_j\Big)=\\=&\sum_{i=1}^{n_B}\Big(W_k^{\star T}B_k^i W_k\prod_{\substack{l=1\\l\neq k}}^{2}W_l^T B_l^i\Big)\end{aligned} \tag{7.5}$$

avec

$$B_1^i=M_{X_1}f_{X_1}^j;\qquad B_2^i=M_{X_2}f_{X_2}^j.$$

7.3.2 Développement d'un logiciel de calcul scientifique

L'objectif ici est de mettre en place un logiciel de simulation fonctionnel aussi bien sur CPU que sur GPU, et ce quelque soit le problème physique à simuler en utilisant à la fois FEM pour comparer les résultats et la PGD pour mettre en évidence l'intérêt de l'implémentation des modèles réduits sur GPU.

7.3.2.1 Analyse du besoin

La résolution d'un problème d'équations aux dérivées partielles nécessite la détermination de certains concepts. Premièrement, la définition du problème à résoudre, dans quel cadre on se place ainsi, que la définition de l'opérateur mathématique continu. Dans le cadre de la réduction de modèle, la notion d'opérateur implique la notion de séparation de variables. L'entité variable correspond aux données, conditions initiales ou inconnues du problème. Enfin, la spécification de la géométrie et des conditions aux limites.

Les différentes phases de résolution sont décrites dans le section 3.1. Il est maintenant question de mettre en place une conceptualisation des phases de résolution en utilisant un diagramme de séquence.

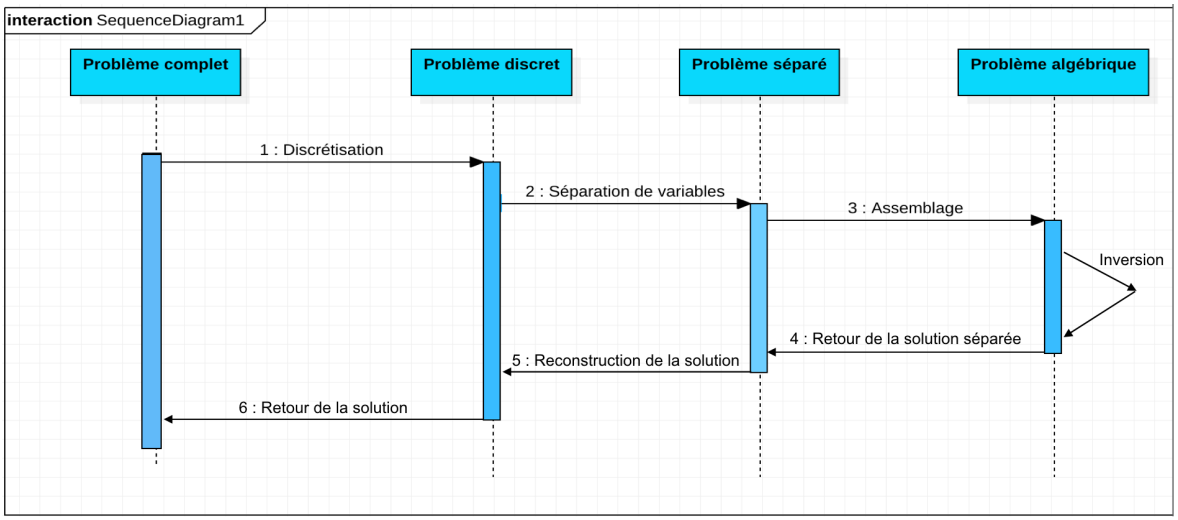


FIGURE 7.6 – Diagramme de séquence pour la simulation d'un problème en utilisant la PGD.

On cherche à mettre en évidence l'intérêt des modèles réduits pour la résolution du problème 7.1, en comparant différentes méthodes de stockage de la matrice ainsi que différentes méthodes de résolution.

7.3.2.2 Phase de conception

Malgré l'utilisation d'un langage procédurale, on utilise le diagramme de classe 7.7 afin de déterminer les différentes entités en jeu pour la modélisation du problème 7.1. Le paquetage problème pilote la totalité de la simulation numérique, et discrétise l'ensemble des autres entités. Le paquetage variables comprend la classe variable discrète et les sous-classes Matrice et Vecteur, et est définie sur un domaine spatial. L'entité domaine permet de gérer les données et le problème à résoudre. L'opérateur traduit le modèle physique à résoudre et est défini sur un domaine prédéfini, l'opérateur discret est déterminé en fonction de la méthode de discrétisation choisie. Les entités de bas niveau : méthodes numériques représentent les différentes méthodes de résolution et le choix de la méthode appropriée.

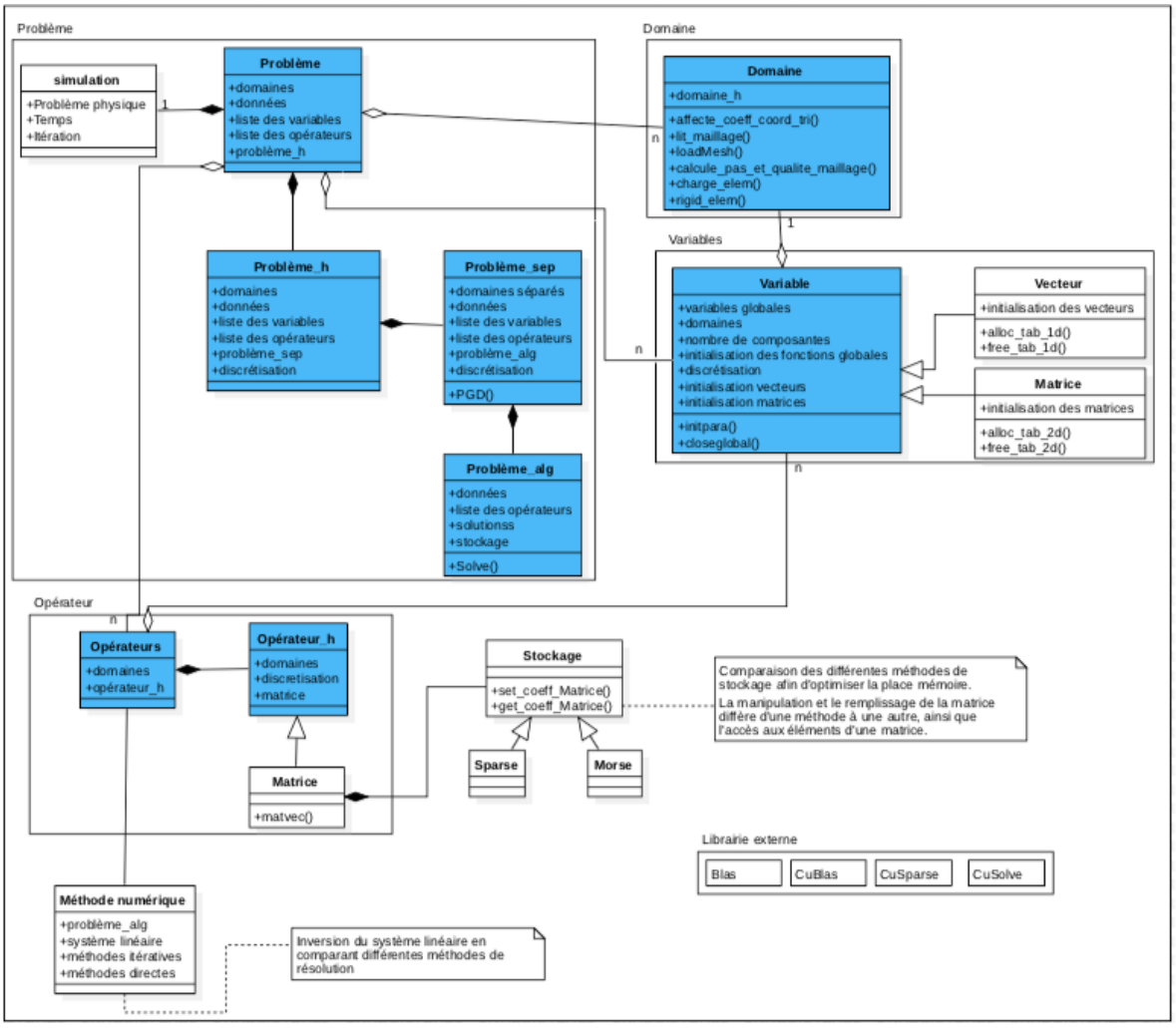


FIGURE 7.7 – Diagramme de classe pour la simulation d'un problème en utilisant la PGD.

7.4 Mise en place des différentes entités sur CPU

7.4.1 Stockage des matrice [6]

Les propriétés des fonctions de base, conduisent à un système linéaire avec une matrice d'assemblage $\mathbb{A}$ symétrique et définie positive. Dans le cadre de la PGD avec décomposition sur des domaines 1D, cette matrice est tri-diagonale. Pour illustrer les différentes entités de stockage, on donne la matrice

$$\mathbb{A} = \begin{pmatrix} 1.1 & 1.2 & 0 & 0 \\ 1.2 & 2.2 & 2.3 & 0 \\ 0 & 2.3 & 3.3 & 3.4 \\ 0 & 0 & 3.4 & 4.4 \end{pmatrix}$$

7.4.1.1 Stockage Sparse

Le stockage Sparse consiste à chaîner pour chaque ligne (ou colonne) une liste de coefficients et leur indice colonne (ou ligne) dans l'ordre croissant des colonnes (ou lignes). Ce stockage est un stockage creux optimal dans le sens où il ne stocke que les coefficients non nuls. Il a la particularité d'être complètement dynamique, et souple étant donné qu'il n'utilise pas de tableaux, ce qui facilite l'insertion de nouveaux coefficients lors de l'assemblage comme l'illustre la figure 7.8.

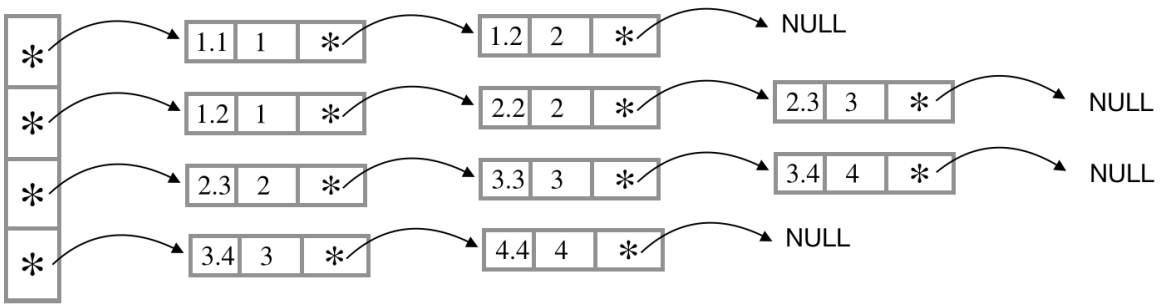


FIGURE 7.8 – Stockage Sparse en utilisant les listes chaînées.

L'inconvénient de ce type de stockage est la taille de structure qui a tendance à augmenter lors du remplissage de la matrice par de nouveaux coefficients non-nuls, générant des allocations dynamiques répétées ce qui implique un temps perdu en matière de gestion de mémoire. Ce stockage est bon pour l'assemblage du système linéaire, il reste toutefois peu efficace pour la résolution par des méthodes itératives puisqu'il ne permet pas d'exploiter les mémoires caches, car il n'y a pas de continuité en mémoire.

6. F. Lefèvre, Calcul scientifique : systèmes linéaires creux

7.4.1.2 Stockage Morse

Au même titre que le stockage Sparse, le stockage Morse est un stockage creux optimal adapté aux méthodes itératives. Dans notre cas, la matrice $\mathbb{A}$ est symétrique, stockée dans un tableau condensé. Dans la table 7.1, les coefficients de la $i^{\text{ème}}$ ligne de la matrice sont

indexe	1	2	3	4	5	6	7
ATAB	1.1	1.2	2.2	2.3	3.3	3.4	4.4
JTAB	1	1	2	2	3	3	4
IFIN	1	3	5	7			

TABLE 7.1 – Stockage Morse.

rangés dans la structure entre les indices $\text{IFIN}(i-1)+1$ et $\text{IFIN}(i); \forall i \in [\![2, n]\!]$. La première ligne est stockée à la position $\text{IFIN}(1) = 1$. Cette méthode de stockage permet l'utilisation des mémoires caches sur CPU lors des produits matrice-vecteur.

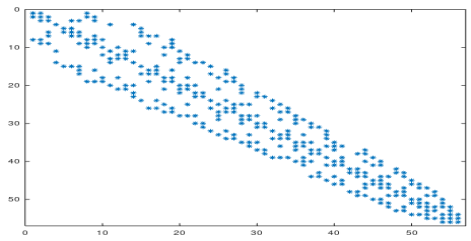
7.4.1.3 Stockage Profil

Ce type de stockage (cf. figure 7.9a) est une des méthodes les plus adaptées aux matrices utilisées lors d'une implémentation PGD. En effet, cette méthode permet de stocker les coefficients entre le premier non-nul et le dernier non-nul rangés par lignes

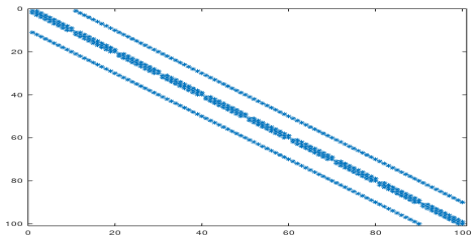
indexe	1	2	3	4	5	6	7
ATAB	1.1	1.2	2.2	2.3	3.3	3.4	4.4
IDIAG	1	3	5	7			

TABLE 7.2 – Stockage Profil.

Le coefficient $a_{i\,j}$ de la matrice $\mathbb{A}$ se trouve dans la structure Profil à la position $I = j - i + \text{IDIAG}(i)$.



(a) Stockage profil d'une matrice de taille $N = 56 \times 56$.



(b) Stockage bande d'une matrice de taille $N = 100 \times 100$.

FIGURE 7.9 – Illustration des différents types de stockage adaptés aux matrices tri-diagonales.

7.4.1.4 Stockage Bande

Le deuxième type de stockage (cf. figure 7.9b) adapté aux matrices tri-diagonales est le stockage Bande. En effet, cette méthode permet de stocker les diagonales non-nulles rangées en colonnes. En considérant la matrice $\mathbb{A}$ sus-mentionnée, on a en plus de la diagonale principale, $\omega = 1$ sous-diagonales non-nulles. $\forall(i,j)/max(1, i-\omega) \leq j \leq i, a_{i\,j}$ se retrouve en position $(i, j-i+\omega+1)$ d'un tableau de taille $n \times (\omega+1)$.

7.4.2 Méthodes numériques

Pour la résolution du système algébrique, plusieurs choix se présentent à la fois en utilisant des méthodes directes (factorisations LU, LDL^T, QR, Cholesky) et les méthodes itératives (Jacobi, Gauss-Seidel, gradient-conjugué). La méthode choisie doit respecter un compromis entre nombre de nœuds, capacité mémoire et optimisation d'algorithme de résolution.

7.4.2.1 Méthodes directes

Ces méthodes prennent en compte le type de la matrice car l'implémentation est en effet différente selon le type. La matrice $\mathbb{A}$ étant tri-diagonale, on définit une structure *Vecteur* composée d'un entier qui correspond à la taille du vecteur, et d'un pointeur *tab* contenant les coefficients du vecteur. Nous définissons donc notre matrice $\mathbb{A}$ avec trois vecteurs : dA, uA, lA correspondant respectivement aux coefficients diagonaux, sur-diagonaux et sous-diagonaux de la matrice $\mathbb{A}$. Pour une matrice symétrique nous allouons uA et dA et mettons lA à NULL alors que pour une matrice diagonale nous allouons seulement dA et les deux autres sont NULL. Afin de résoudre le système linéaire $\mathbb{A}X = \mathbb{B}$, en utilisant la décomposition LU, la

Algorithme 3 : Décomposition LU

Solveur_LU_TriDiag ;
Données : $lA \in \mathbb{R}^{n-1}, uA \in \mathbb{R}^{n-1}, dA \in \mathbb{R}^n$
Sorties : $l \in \mathbb{R}^{n-1}, d \in \mathbb{R}^n$.

$d_1 \leftarrow dA_1$ **for** $i = 2$ *to* n **do**
  for $j = 2$ *to* $i-1$ **do**
    $l_{i-1} \leftarrow \dfrac{lA_{i-1}}{d_{i-1}}$
    $u_i \leftarrow uA_i$
    $d_i \leftarrow dA_i - l_{i-1}u_i$
  end
end

résolution s'effectue en deux étapes

Un algorithme de descente qui consiste à résoudre le problème triangulaire inférieur $Ly = \mathbb{B}$

$y_1 \leftarrow \mathbb{B}_1$
for $i = 1$ to n
$y_i \leftarrow \mathbb{B}_i - l_{i-1}y_{i-1}$
end

Un algorithme de remontée qui consiste à résoudre le problème triangulaire supérieur $Ux = y$

$x_n \leftarrow \dfrac{y_n}{d_n}$
for $i = n - 1$ to 2
$x_i \leftarrow \dfrac{y_i - uA_{i+1}x_{i+1}}{d_i}$
end

7.4.2.2 Méthodes itératives

En utilisant la méthode des itérations successives décrite ci-dessous, on cherche à décomposer la matrice $\mathbb{A}$ sous la forme $\mathbb{A} = M - N$, où M est une matrice facilement inversible.

$$\begin{aligned} Mx_{k+1} = Nx_k + b &\iff Mx_{k+1} = (M - A)x_k + b \\ &\iff M(x_{k+1} - x_k) = -Ax_k + b = r_k \\ &\iff x_{k+1} = x_k + M^{-1}r_k \end{aligned}$$

On cherche à décomposer la matrice $\mathbb{A}$ sous la forme $\mathbb{A} = D - L - U$, avec D la diagonale principale, $(-U)$ et $(-L)$ respectivement les parties triangulaires supérieures et inférieures strictes de la matrice $\mathbb{A}$. On résume les différentes méthodes itératives dans le tableau ci-

Jacobi	$M = D$
Gauss-Seidel	$M = (D - L)$
Relaxation	$M = (D/\omega - L)$ avec $0 < \omega < 2$

dessus. L'algorithme général est représenté par

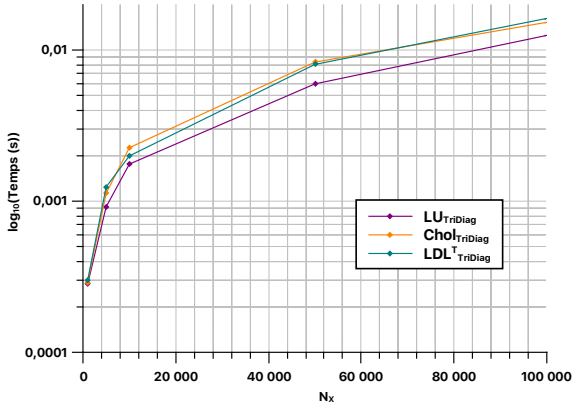
Algorithme 4 : Algorithme itérative

Solveur_iter ;
Données : $A \in \mathbb{R}^n$, $b \in \mathbb{R}^n$
Sorties : $x \in \mathbb{R}^n$.

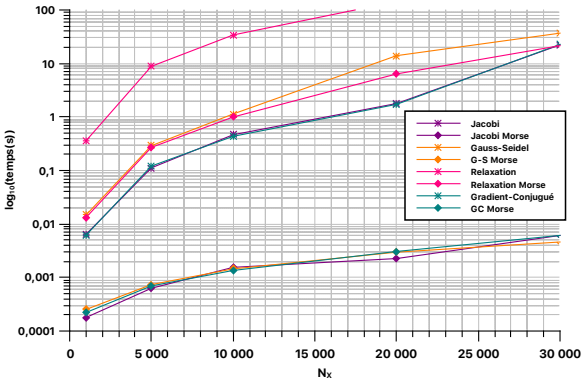
$k = 0; \quad r = b - Ax$
while $\dfrac{\|r\|_\infty}{\|b\|_\infty} > \epsilon \quad \& \quad k < k_{max}$ **do**
  $x = x + d$
  $r = b - Ax$
  $k = k + 1$
end

7.4.2.3 Le choix de la stratégie de programmation

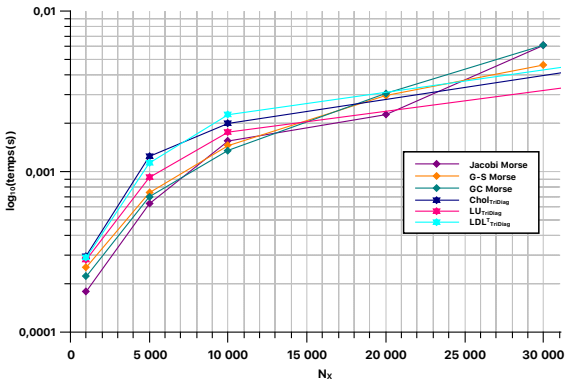
La figure 7.10a compare 3 méthodes numériques directes pour la résolution de problème en utilisant différentes tailles. Comme sus-mentionnée, notre matrice est tri-diagonale elle est donc implémentée dans une structure composée de 3 vecteurs où sont stockées la diagonale principale et les 2 sous-diagonales. La figure 7.10 permet de constater que l'algorithme LU est le moins coûteux en terme de temps de calcul.



(a) Méthodes directes en utilisant un stockage bande.



(b) Méthodes itératives en utilisant un stockage Morse et un stockage plein.



(c) Les temps de calcul des méthodes itératives et directes en utilisant la même matrice

FIGURE 7.10 – Comparaison des temps de calcul des méthodes de résolution en utilisant différentes tailles de problème et différents types de stockage.

D'autre part, on compare en utilisant la même matrice les méthodes de Jacobi, de Gauss-Seidel, de relaxation et du gradient conjugué avec et sans stockage Morse. Pour chaque méthode, nous traçons le temps de calcul en échelle logarithmique en fonction de N_X pour

les deux types de stockage (cf. 7.10b). Les temps de calcul sont plus intéressants lorsque le stockage est de type Morse. Ce type de stockage est avantageux lorsque les matrices sont tri-diagonales. En effet, plus la taille de la matrice devient importante, plus le gain en temps en utilisant le stockage Morse devient considérable comme l'illustrent les courbes (cf. 7.10b).

On peut par ailleurs comparer les temps de calcul entre les méthodes directes et les méthodes itératives. Comme nous pouvons l'observer sur le graphe 7.10c, les temps de calcul avec le stockage Morse augmentent très faiblement lorsque la taille de la matrice augmente. Cependant, à partir de $N_X = 30000$ les méthodes directes deviennent plus intéressantes notamment la décomposition en LU.

7.5 Du séquentiel au parallèle

Papadrakakis et al. [86] ont abordé une configuration hybride CPU-GPU pour l'implémentation des méthodes de décomposition de domaine, qui sont des méthodes permettant la résolution des problèmes exigeants. Plus de détails sont apportés par Jolivet et al. [51]. Ces techniques ne sont pas des algorithmes de réduction de modèles, cependant ils permettent d'avoir une idée sur les ressources qu'offre une implémentation des domaines décomposés sur GPU.

Fritzen et al. [36] ont proposé une implémentation parallèle de l'algorithme de réduction de modèles pour la résolution des problèmes visco-plastiques sur GPU. Les auteurs ont aussi présenté en détail l'implémentation des différentes opérations algébriques ainsi que leur impact sur les performances. Cet article détaille l'implémentation sur GPU des subroutines calculant le produit matrice-vecteur, et le produit matriciel. Les gains de performances sont illustrés à l'aide d'exemples numériques en utilisant un algorithme hétérogène *Reduced-Basis Model-Order-Reduction* (RB-ROM). Une accélération significative est obtenue tout en préservant une bonne précision. Les auteurs réalisent un *benchmark* en utilisant 3 cartes graphiques dont 2 architectures différentes la Fermi et la Kepler.

7.5.1 Principe de l'implémentation

Le GPU est un processeur à mémoire partagée (96 ko), relié à la carte mère via PCI Express. Toutes les fonctions CUDA sont développées autour du même modèle de traitement illustré par la figure 7.11.

Afin de minimiser les latences mémoire, il est nécessaire d'effectuer le maximum de calculs possible sur la carte graphique, tout en minimisant les transferts de données entre les deux architectures. Il est vivement conseillé de récupérer uniquement les valeurs finales et non la totalité des données.

L'objectif est donc de :
— Effectuer le minimum de transfert de données entre le CPU et le GPU, car cela consti-

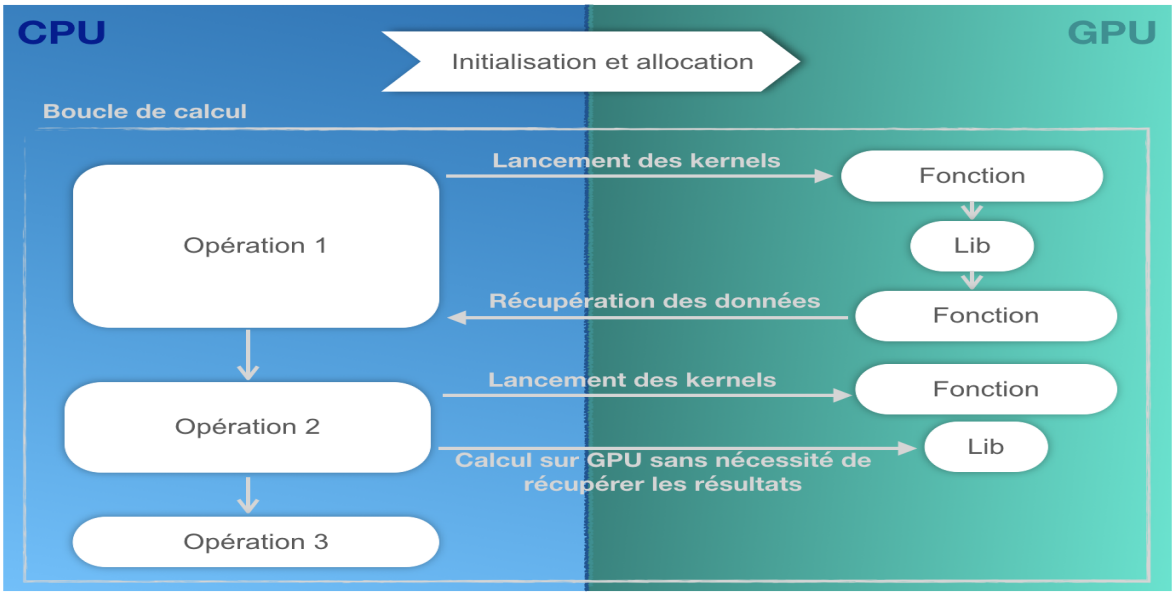


FIGURE 7.11 – Diagramme représentant l'exécution du code à la fois sur CPU et GPU.

tue un goulet d'étranglement étant donné que le temps de transfert est souvent plus long que le calcul effectué sur GPU.

— Réaliser le maximum de calculs sur le GPU.

— Minimiser l'accès que ce soit en lecture ou en écriture à la mémoire globale (comme détaillé dans l'Annexe A).

7.5.2 Repérer les portions de code les plus chronophages

L'algorithme PGD introduit dans la section 7.3.1, aborde la séparation du domaine 2D en 2 dimensions comme illustré par le tableau suivant dans cet exemple on se contente aussi de deux opérateurs pour chaque dimension. Les fonctions qui calculent les différents produits vecteurs-vecteurs ou matrices-vecteurs sont les plus appelées dans le programme, la méthode de résolution utilisée est le solveur LU tri-diagonale.

Nombre de nœuds dans chaque direction	Nombre de modes	Nombre d'opérateurs	Dimensions séparées	Appels au prod vec-vec	Appels au prod mat-vec
1 000	9	2	2	$\sim 56 \times 10^5$	$\sim 56 \times 10^2$
5 000	11	2	2	$\sim 28 \times 10^6$	$\sim 56 \times 10^2$
10 000	11	2	2	$\sim 57 \times 10^6$	$\sim 57 \times 10^2$

TABLE 7.3 – Les variables de la PGD.

L'objectif ici est de déterminer les fonctions les plus chronophages en profilant le code séquentiel, ensuite de déterminer les processus qui sont indépendants les uns des autres. Une première analyse de l'exécution du code séquentiel est effectué à l'aide de l'outil gprof. Une partie du profilage est présentée par la figure 7.12. Dans un premier temps, les parties qui se

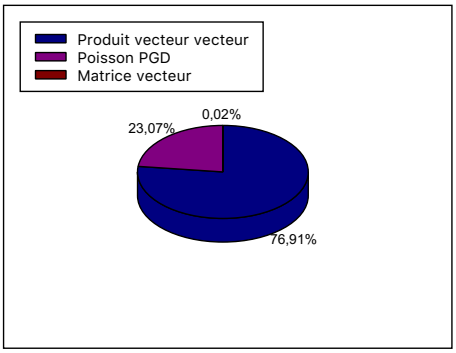


FIGURE 7.12 – Part du temps de calcul des fonctions les plus chronophages.

prêtent bien à l'utilisation parallèle intensive au cours de l'enrichissement de la PGD sont les nombreuses multiplications vecteur-vecteur, et matrice-vecteur.

On peut donc constater sur la figure 7.12 que le programme passe près de 77% du temps d'exécution à calculer le produit de deux vecteurs. En effet, cette fonction est appelée environ 49×10^6 fois dans le cas de la simulation du problème 7.1 en utilisant un modèle avec $N_x = N_y = 10^3$ discrétisation, et est utilisée à plusieurs moments de la phase d'enrichissement. En se référant à la loi d'Amdahl 7.1.1.1, une telle fonction se prête à un portage GPU très favorable. Nous réaliserons un benchmark des performances GPU en comparaison des bibliothèques CPU préalablement optimisées (PGI BLAS, ATLAS...)

7.5.3 Mise en œuvre sur cartes graphiques

Afin d'alléger la notation de la forme séparée 7.3, on note

$$u^n(x_1, x_2) = \sum_{j=1}^{n-1} X_j(x) \cdot Y_j(y) + R(x) \cdot P(y) \tag{7.6}$$

Il s'agit ici de détailler l'implémentation sur GPU de la fonction $R(x)$ à partir de la fonction $P(y)$ initialement déterminée pour démarrer le processus d'enrichissement. On détermine alors les intégrations numériques suivantes

$$\alpha_x = \int_{\Omega_y} (P(y))^2 \, dy \qquad \gamma_x^j = \int_{\Omega_y} P(y) \cdot Y_j(y) \, dy$$

$$\xi_x = \int_{\Omega_y} P(y) \cdot f_y \, dy \tag{7.7}$$

$$\beta_x = \int_{\Omega_y} P(y) \cdot \frac{d^2 P(y)}{dy^2} \, dy \qquad \delta_x^j = \int_{\Omega_y} P(y) \cdot \frac{d^2 Y_j(y)}{dy^2} \, dy$$

Afin de mettre en évidence les performances d'une implémentation sur GPU, et pour des raisons de mémoire nous découplons les différentes fonctions afin de réaliser des benchmarks conséquents, l'implémentation complète est présentée dans l'Annexe B, dans ce qui suit nous

détaillerons uniquement certains kernels.

7.5.3.1 Produit vecteur-vecteur

Les algorithmes de réduction de modèles comportent un nombre élevé de produit scalaire à la fois au niveau de chaque itération et au niveau des algorithmes de résolution. En fonction de la taille du problème, ces produits peuvent alourdir le programme à la fois en terme de temps de calcul et en terme de gestion de mémoire. Ce produit peut se séparer en deux tâches distinctes. La première consiste à multiplier une paire de données correspondant à chaque thread comme l'illustre la figure 7.13, ensuite le thread passe à la paire suivante. La seconde consiste à calculer la somme de tous ces produits par paires. Cette somme partielle est stockée temporairement dans la mémoire partagée du block.

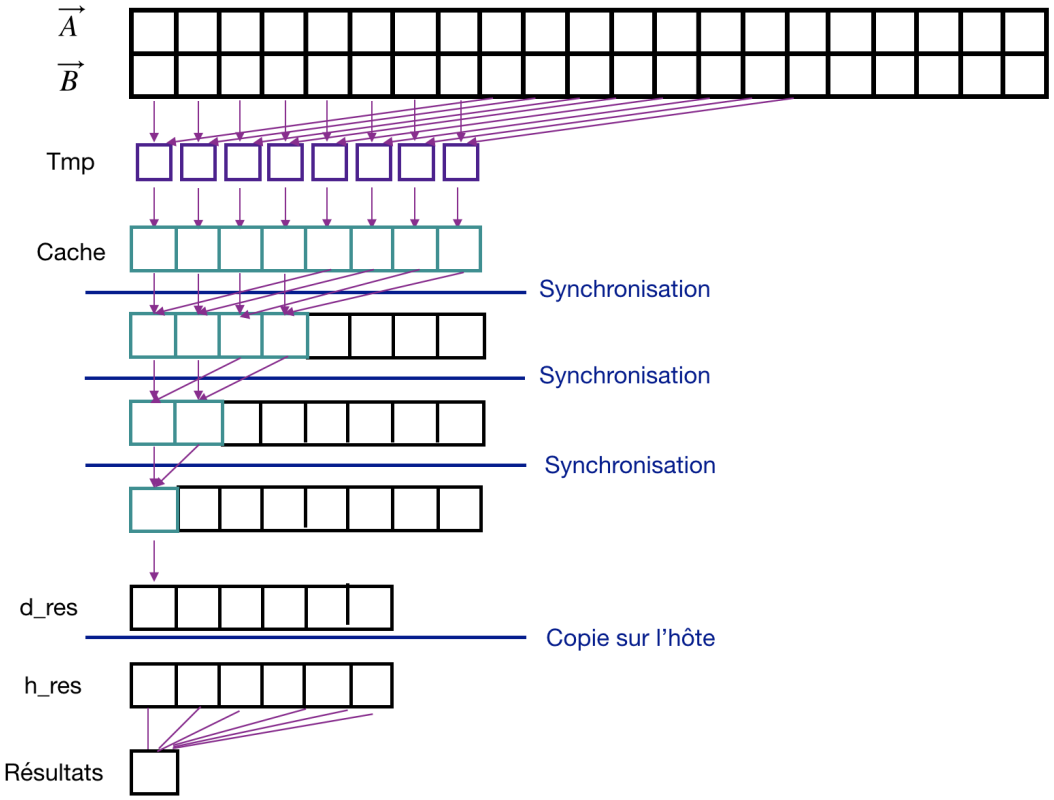


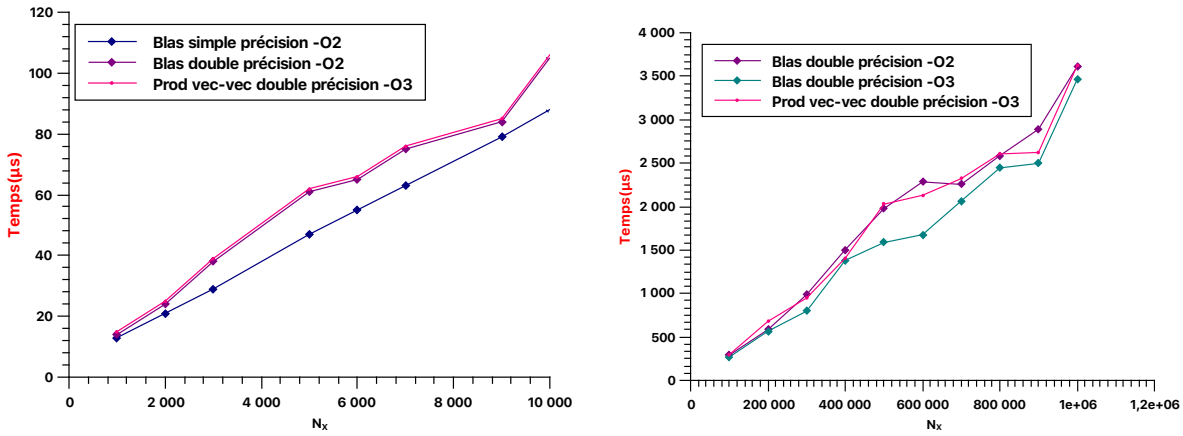
FIGURE 7.13 – Algorithme du produit.

L'opération de sommation est tout aussi délicate, en effet, le fait de parcourir la mémoire partagée et de calculer la somme en cours d'exécution nécessite un temps de calcul proportionnel à la longueur du tableau. Cependant, on peut utiliser les instructions *shuffle down*[7] (Illustrée par l'algorithme 7.13), qui consiste à calculer chaque somme partielle et la stocker

7. Fonctionne avec les types int, float et half. Pour des types plus grands il est nécessaire de séparer l'instruction en plusieurs appels.

dans la mémoire partagée du block. Un thread de ce block va ensuite parcourir ce tableau temporaire pour ensuite calculer une seconde somme partielle qui est stockée cette fois au niveau de la mémoire globale (les détails sur l'architecture des mémoires sont présentés dans l'Annexe A). On récupère suite à l'exécution de la totalité des warps qui parcourent les différents blocks, un nouveau tableau qui contient l'ensemble des sommes partielles de tous les blocks. La répétition de ce processus permet d'obtenir la somme totale du tableau initial. Ce processus est décrit par le diagramme représenté par la figure 7.13.

Cependant, il faut choisir le meilleur compromis entre l'utilisation efficace du GPU et la réduction du nombre d'exécution lors de l'exploitation des tableaux de grande taille. Il est à noter qu'un block exécute au maximum 1024 tâches, pour cette raison et afin de pouvoir réaliser une réduction sur un seul block, le tableau intermédiaire (qui se trouve au niveau de la mémoire globale) doit être d'une taille inférieure ou égale à 1024, ce qui implique de nombreux appels de fonctions et de lancements de tâches sur le GPU (cf. figure 7.11). D'autre part, l'utilisation d'un seul block ne permet donc d'utiliser qu'un seul SMX. Cependant, avec des GPUs qui possèdent une dizaine de SMX, cette approche ne permet d'exploiter qu'un dixième de la puissance du périphérique. La manière la plus optimale est d'utiliser des tableaux intermédiaires d'une taille maximale égale à 1024, ce qui permet d'appliquer une réduction sur au maximum 1024 block en utilisant uniquement 1024 threads. Après synchronisation, il suffit d'un seul block de 1024 threads pour calculer la somme totale. Cette valeur est recopiée sur l'hôte pour pouvoir l'exploiter, parce qu'il y a que le CPU qui a accès via l'OS au disque dur.



(a) Blas en simple (32 bits) et double précision (64 bits).

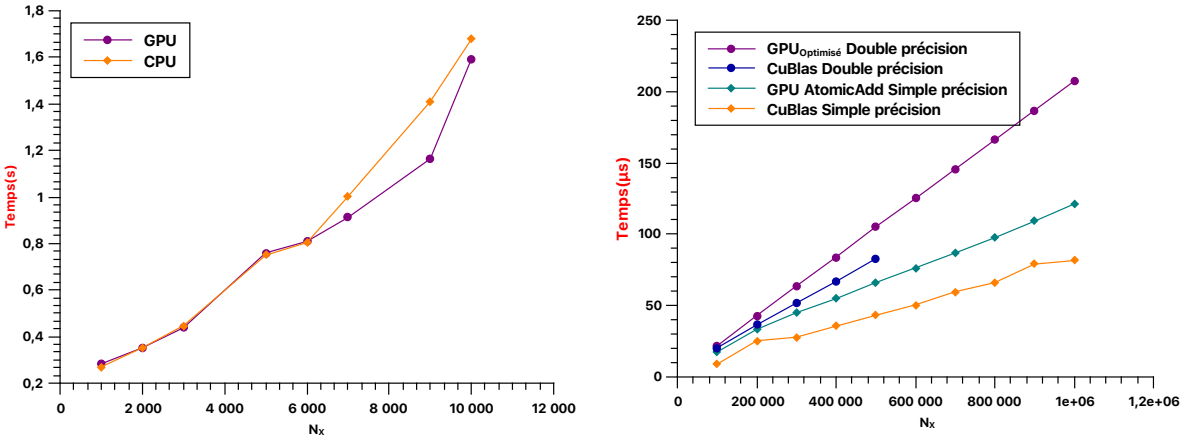
(b) Blas en double précision (64 bits)

FIGURE 7.14 – Benchmark produit vecteur-vecteur en utilisant la librairie Blas en simple et double précision sur la figure 7.14a et comme option de compilation "-O2", et une comparaison avec la fonction développée et compilée grâce à l'option "-O3". La figure 7.14b représente le benchmark en utilisant une grande échelle en double précision et avec comme option d'optimisation "-O2" et "-O3", et une comparaison avec notre fonction produit vecteur-vecteur en utilisant "-O3".

Un premier benchmark est réalisé sur la partie CPU en utilisant la librairie GSL, il est à noter que les deux benchmarks de la figure 7.14 n'ont pas les mêmes échelles. La figure 7.14a représente un benchmark de l'utilisation du Blas en simple et double précision en utilisant l'option d'optimisation "-O2" ainsi que les temps de calcul obtenus en utilisant notre fonction optimisée couplée à l'option d'optimisation "-O3", sur cette figure sont représentées différentes tailles de vecteurs où N_X représente la longueur d'un vecteur. On remarque dans ce cas que le temps de calcul est réduit en utilisant la simple précision. Dans la figure 7.14b, on représente une comparaison du Blas en double précision sur une plus grande échelle et en utilisant deux options d'optimisations différentes "-O2" et "-O3".

Un premier benchmark de la version naïve à la fois sur CPU et sur GPU sans aucune option d'optimisation est illustré par la figure 7.15 afin de mettre en évidence l'intérêt d'une parallélisation sur GPU. Le nombre d'accès en mémoire globale à chaque vecteur en utilisant une version naïve (cf. figure 7.15a) est égale en lecture à $2 \times N_X \times$ NB_ELEM (avec NB_ELEM représente le nombre d'éléments d'un tableau) et en écriture à $(N_X + 1)$NB_ELEM. L'objectif est de réduire ce nombre d'accès en réutilisant les données des threads voisins grâce à la mémoire partagée, ce qui permet de réduire le nombre d'accès en lecture à NB_ELEM $+2 \times N_X \times gridDim.x$ et en écriture à NB_ELEM (cf. figure 7.15b). Afin d'optimiser cette version deux étapes sont essentielles à savoir

— Le chargement et utilisation de la mémoire partagée
— La synchronisation de tous les threads du block



(a) Version naïve sur CPU et sur GPU.

(b) Version optimisée sur GPU.

FIGURE 7.15 – Benchmark des différentes implémentations du produit vecteur-vecteur sur GPU sans l'utilisation d'une option d'optimisation.

```
__global__ void prod_tab( double *a, double *b, double *c ) {
        __shared__ double cache[THREADS_PER_BLOCK];
        int index = threadIdx.x + blockIdx.x * blockDim.x;
        int cacheIndex = threadIdx.x;
        double tmp = 0;
        while (index < N) {
                tmp += a[index] * b[index];
                index += blockDim.x * gridDim.x;
        }
        cache[cacheIndex] = tmp;
        __syncthreads();
        int i = blockDim.x/2;
        while (i != 0) {
                if (cacheIndex < i)
                        cache[cacheIndex] += cache[cacheIndex + i];
                __syncthreads();
                i /= 2;
        }
        if (cacheIndex == 0)
                c[blockIdx.x] = cache[0];
}
```

Le benchmark de la figure 7.15b permet de comparer les différentes implémentations en utilisant différentes tailles de vecteurs. Pour cette comparaison, CuBlas est implémenté en simple et double précision, on remarque que pour des raisons de mémoires sur notre carte graphique le CuBlas en double précision ne peut effectuer de calcul au-delà de $N_X = 5 \times 10^5$ éléments. Toutefois, cet obstacle est pallié en simple précision. Il est aussi possible de garder notre kernel optimisé et d'effectuer un changement au niveau des sommes partielles et d'utiliser la fonctionnalité *AtomicAdd* comme détaillée ci-dessous.

Les opérations atomiques permettent d'exécuter des opérations en lecture-modification-écriture en mémoire globale ou partagée. L'avantage de cette méthode réside dans la sommation des différents produits partiels. En effet, l'opération atomique lit une valeur à une adresse donnée dans la mémoire globale ou partagée, calcule ensuite la somme de cette ancienne valeur et d'une nouvelle et stocke le résultat dans la mémoire à la même adresse. En d'autres termes, les fonctions atomiques garantissent la non concurrence avec d'autres threads, car la zone mémoire concernée par la sommation est inaccessible par les autres threads, ce qui n'implique pas de contraintes de synchronisation ou d'ordre pour les opérations de mémoire, ce qui pourrait représenter un inconvénient étant donnée la sérialisation de l'exécution. Cependant, un benchmarck est réalisé en utilisant la fonction *AtomicAdd()* et illustré par la courbe turquoise (cf. figure 7.15b).

Le graphe 7.16 illustre la hiérarchie de la mémoire du kernel produit vecteur-vecteur en utilisant le modèle optimisé (cf. figure 7.16a) et le modèle utilisant l'opération *AtomicAdd* (cf. figure 7.16b). En vert est représenté l'espace mémoire logique, tandis qu'en bleu c'est l'unité matérielle de la puce qui est représentée. Le rapport entre le nombre de demandes pouvant être traitées et les données disponibles localement dans le cache pour toutes les demandes effectuées est indiqué pour chaque cas par le pourcentage indiqué sous les différentes mémoires caches.

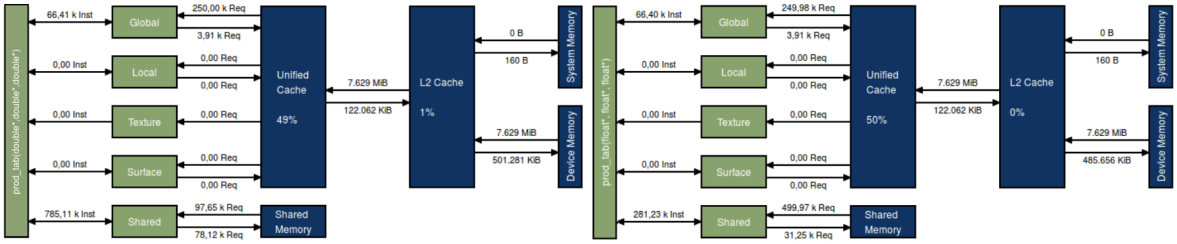


FIGURE 7.16 – Statistiques de l'utilisation des différentes mémoires.

La quantité de mémoire utilisée entre les opérations de lecture et d'écriture sont représentées par les différents chemins de données entre les espaces mémoires et la mémoire cache ou encore la mémoire partagée, et c'est au niveau de cette dernière qu'on remarque une réduction de la quantité totale de mémoire transférée en octets en utilisant les opérations atomiques.

7.5.3.2 Produit matrice-vecteur

L'algorithme de PGD fait intervenir des produits matrice-vecteur, notamment lors du calcul du second membre dans la formulation variationnelle 7.5. Ce sont ces produits qui sont les plus coûteux dans ces algorithmes, toutefois, étant donnée le type de la matrice, on peut se ramener parfois à des produits vecteur-vecteur. Ce produit est illustré par le schéma 7.17

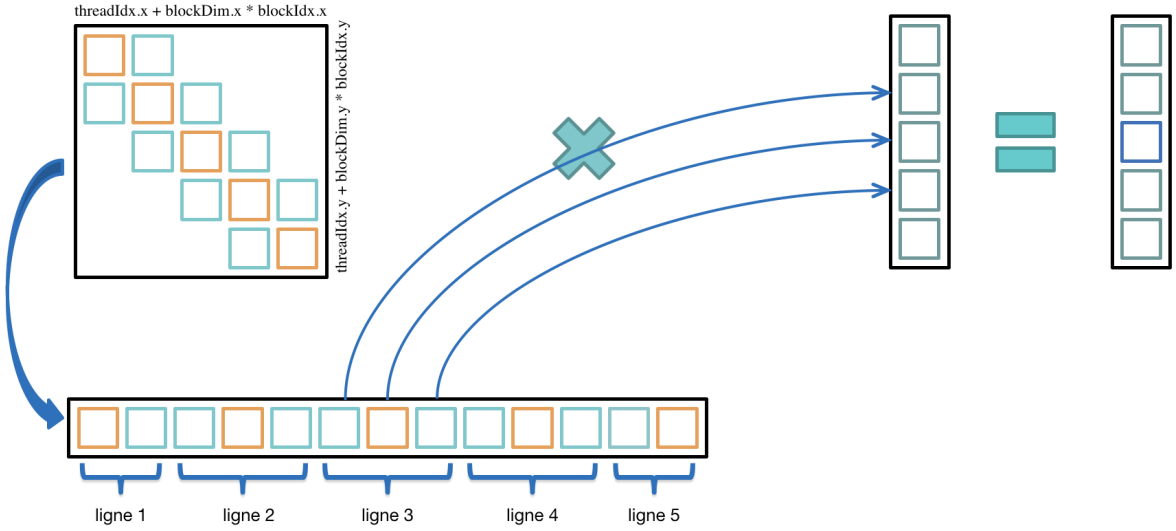


FIGURE 7.17 – Produit matrice-vecteur.

Étant donnée l'importance de la multiplication matrice-vecteur, plusieurs chercheurs ont étudié cette multiplication connue dans la littérature sous la terminologie SpMV (*Sparse matrix-vector multiplication*) afin de réaliser des tests à grande échelle.

L'objectif de l'implémentation de l'algorithme PGD sur GPU réside principalement dans le traitement et la résolution de très grands problèmes. Pour ce, nous cherchons à traiter

à travers ces différents kernels de très grandes matrices tri-diagonales en utilisant plusieurs threads en parallèle ce qui permet de réduire la charge de travail pour chaque thread et par suite accélère le calcul. Un autre point essentiel est l'utilisation de la mémoire partagée, comme nous l'avons précédemment vue l'utilisation de ce niveau de mémoire permet d'améliorer les performances des kernels.

De la même façon que pour le calcul du produit vecteur-vecteur, la manière la plus optimale d'optimiser le produit matrice-vecteur est de diviser la matrice en différents blocs d'une taille maximale égale à 1024 threads. Nous utilisons un vecteur uni-dimensionnel comme l'illustre la figure 7.17.

Le pseudo-code qui permet de calculer le produit matrice-vecteur est assez simple et peut se ramener à un produit vecteur-vecteur, il est toutefois possible d'effectuer un certain nombre d'optimisations afin d'améliorer les performances du produit notamment en ce qui concerne le réordonnancement de la matrice, et l'utilisation de la mémoire partagée par block. Dans le cadre d'un produit matrice-vecteur il est possible d'utiliser la mémoire partagée en tant que cache pour stocker le vecteur qui est fréquemment appelé.

```
__global__ void matvec_kernel( double *a, Matrix b, double *c , int n, int M) {
        __shared__ double a1[nx][ny], b1[nx];
        int idx = blockIdx.x * blockDim.x + threadIdx.x;
        int idy = blockIdx.y * blockDim.y + threadIdx.y;
        double sum=0;
        b1[threadIdx.x][threadIdx.y] = a.tab[idx*nx+threadIdx.x];
        a1[threadIdx.x] = a[threadIdx.x];
        __syncthreads();
        for(int i=0; i<nx; i++)
           sum += b1[threadIdx.x][i]*a1[threadIdx.x];
        atomicAdd(c + blockIdx.y * idx + threadIdx.x, sum);
}
```

Pour calculer le produit $Ax = b$, chaque élément de la matrice A est copié une seule fois et de manière synchrone dans la mémoire partagée afin d'éviter un goulot d'étranglement au niveau de la bande passante. Au sein de chaque itération de la boucle for, une valeur de la mémoire partagée est diffusée à tous les threads d'un warp. Dans la présente thèse, le kernel que nous traitons peut utiliser directement les coefficients non-nuls des matrices tri-diagonales ce qui représente une économie en espace de stockage et en temps de traitement.

Dans un premier temps, on compare dans la figure 7.18 les temps de calcul du produit matrice-vecteur en utilisant les fonctions *cblas_dgemv* et *cblas_ddot* de GSL et notre fonction CPU optimisée tous trois en double précision.

La figure 7.19a représente un benchmark pour comparer le kernel matvec et le CuBlas et ce, en utilisant différentes tailles de problèmes. On remarque que pour une matrice de taille $N = 10000 \times 10000$, l'utilisation du CuBlas permet de diviser par 4 les temps de calculs. Les performances $(flops_{kenrel} \times 10^{-9})/(ms_{kenrel}/10^4)$ des deux méthodes sont illustrées par le graphe 7.19b. L'utilisation du CuBlas permet d'obtenir de meilleurs performances, cependant

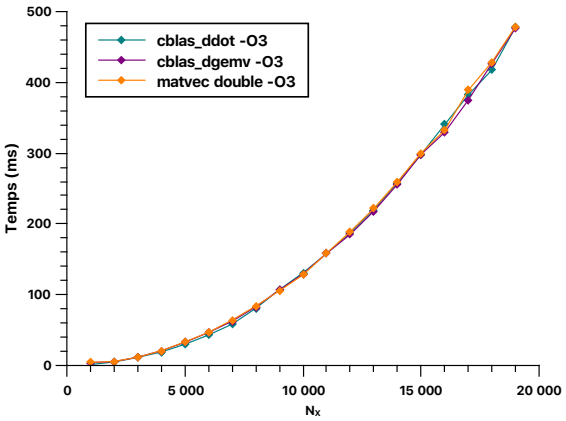
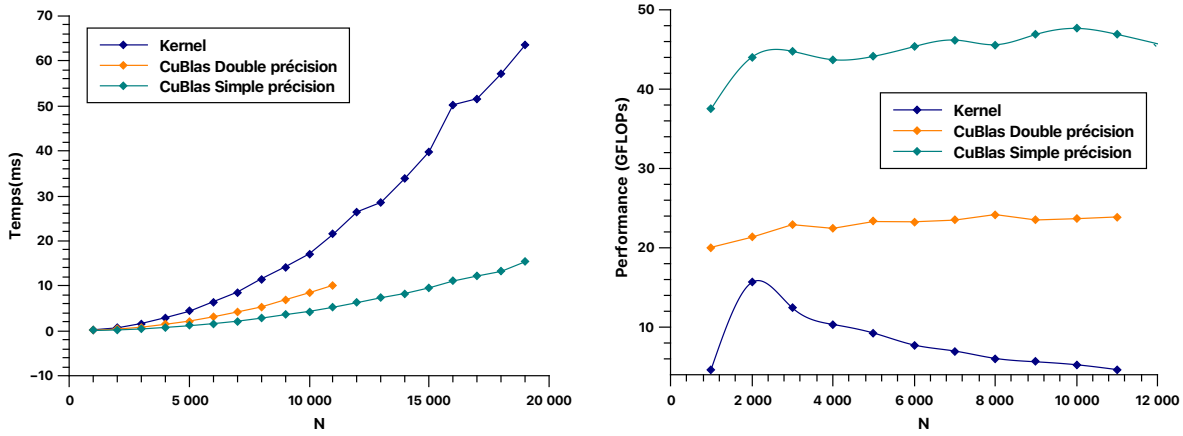


FIGURE 7.18 – Benchmark du kernel matvec et de Blas avec N_X nombre de lignes de la matrice.



(a) Benchmark du kernel matvec et de CuBlas avec N nombre de lignes de la matrice.

(b) Performances des deux méthodes.

FIGURE 7.19 – Benchmark produit matrice-vecteur.

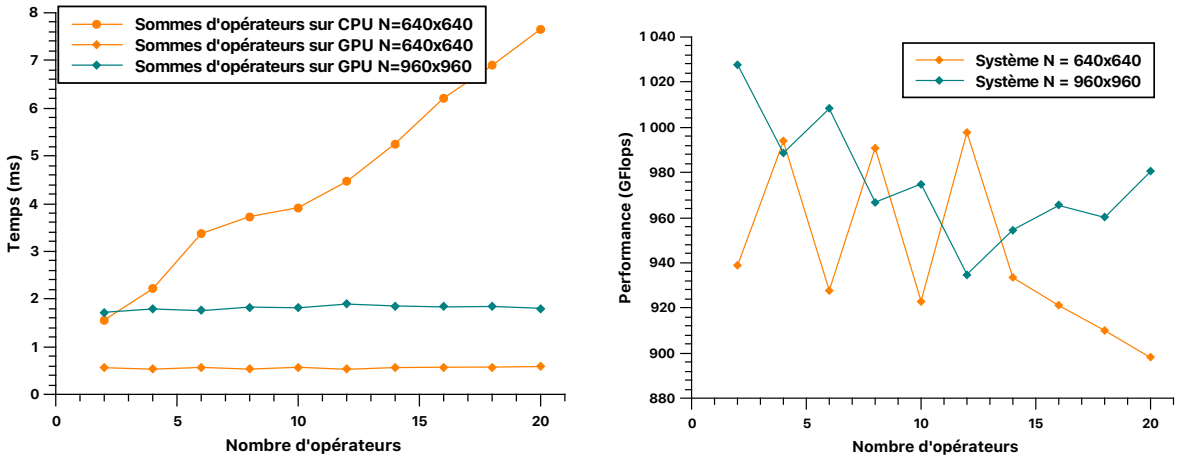
il n'est pas possible pour nous de l'utiliser à cause des contraintes liées à la mémoire de notre carte graphique. Le CuBlas en simple précision permet de contourner cet obstacle de mémoire, mais c'est un compromis entre précision et accélération des temps de calculs. Le kernel développé peut être encore plus optimisé, cependant le nombre de fois que ce kernel est appelé est relativement peu élevé, donc pour cette application on a pas besoin d'optimiser encore plus.

7.5.3.3 Les opérateurs de la PGD

Un des principaux goulots d'étranglements de la PGD est la sommation des différents opérateurs en jeu (cf. FV 7.5). Dans le cadre de cet exemple il existe uniquement deux opérateurs, il est toutefois possible d'étudier l'efficacité de ce kernel en augmentant le nombre d'opérateur comme l'illustre la figure 7.20a.

```
__global__ void matOp_kernel( Matrix a, Matrix b, Matrix c , double alpha,
    double beta, int n, int M){
    int idx = threadIdx.x;
    for (int k = 0; k < b.width; ++k){
        float belement = b.tab[idx * b.width + k];
        float celement = c.tab[idx * c.width + k];
        a.tab[idx * a.width + k]= alpha*belement + beta*celement;
    }
}
```

On effectue un benchmark avec un nombre d'opérateurs variant entre 2 et 20. Les courbes oranges du graphe 7.20a permettent d'illustrer une comparaison CPU/GPU d'un même système avec le même nombre d'opérateurs. Le temps de calcul d'une telle opération sur CPU est proportionnelle au nombre d'opérateurs, alors que les temps de calculs sont plutôt linéaires sur GPU. Les performances d'une telle optimisation sont représentées dans la figure 7.20b pour deux systèmes différents.



(a) Utilisation de deux systèmes différents.

(b) Performances en utilisant plusieurs opérateurs avec deux systèmes.

FIGURE 7.20 – Benchmark en faisant varier le nombre d'opérateurs de la PGD.

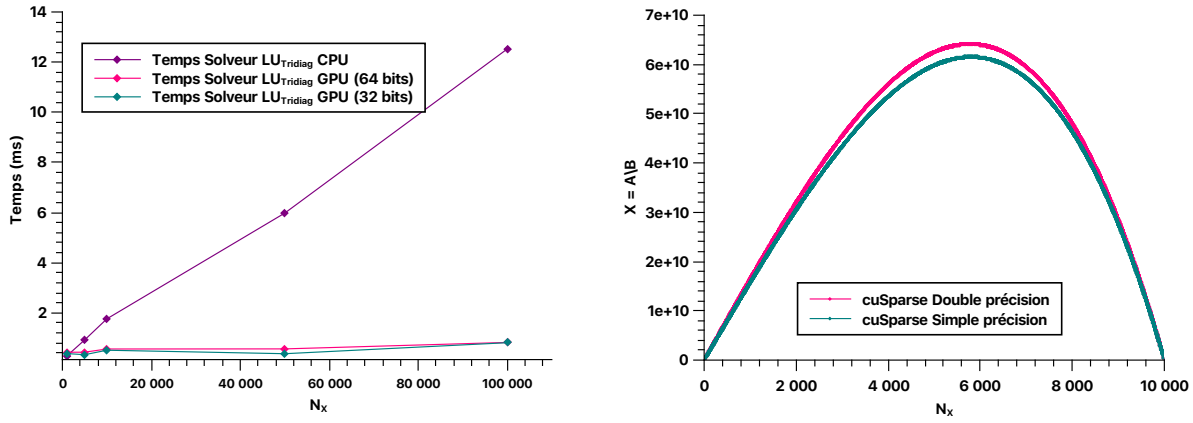
Ce résultat est très important car il montre que l'architecture GPU est bien adaptée au cas où le nombre d'opérateur est grand. Ce qui est le cas de la simulation du procédé SLM.

7.6 Précision numérique

Les cartes graphiques Nvidia et l'environnement de développement CUDA supportent la mémoire ECC [8](précision et fiabilité), ainsi que la possibilité d'exécuter des instructions en simple et double précision.

7.6.1 La résolution du système linéaire

L'objectif ici est de pouvoir mettre en place la résolution du système linéaire en utilisant les bibliothèques cuBlas et cuSparse. Chacune de ces bibliothèques met à disposition plusieurs fonctionnalités utiles similaires à LAPACK. Dans le cadre de notre application, le système que l'on cherche à résoudre est composé d'une matrice $\mathbb{A}$ tri-diagonale. Dans cette partie, nous ne cherchons pas à paralléliser notre solveur LU, mais plutôt effectuer une comparaison avec les bibliothèques CUDA existantes afin de mettre en évidence l'intérêt du portage GPU de cette partie ainsi qu'une étude de la précision attendue. Néanmoins, étant donné les capacités mémoires de notre carte graphique on n'adoptera pas ces bibliothèques.



(a) Solveur LU TriDiag en double précision. (b) Précision du cuSparse en simple et double précision.

FIGURE 7.21 – Benchmark des différents solveurs.

La phase de résolution du système peut être coûteuse dans certains cas de figure, comme par exemple la résolution de la PGD pour le problème élastique (cf. 6.6.2) ou encore le problème thermique sur des dimensions 2D ou 3D (cf. 5.3). En effet, ces problèmes demandent d'inverser des matrices denses ou des matrices creuses pas forcément tri-diagonales.

8. (Error Correction Coding ou Error Checking and Correcting) sont des mémoires possédant plusieurs bits dédiés à la correction d'erreur(on les appelle ainsi bits de contrôle).

```
for((*var_glob).i=N0; (*var_glob).i<= (*var_glob).Max_terms; (*var_glob).i++){
    for((*var_glob).j=1; (*var_glob).j<=(*var_glob).k_fp; (*var_glob).j++){

        // ....
        /* ----------------------------------------
             RESOLUTION DU SYSTEME LINEAIRE
        ---------------------------------------- */
        // CPU
        Solveur_LU_TriDiag((*var_glob).Mx,RHStx,Rxt);
        // GPU
        cusolverSpHandle_t solver_handle;
        cusolverSpCreate(&solver_handle);
        // --- LU
        cusparseSafeCall(cusparseDgtsv(handle, (*var_glob).Nx, 1, d_lA, d_dA,
            d_uA, d_RHStx, (*var_glob).Nx));

    } /* FIN phase de point fixe */
} /* FIN phase d'enrichissement '*/
```

Le choix des méthodes directes n'est pas anodin dans le cadre de cette application. En effet, ces méthodes sont d'une part robustes et assurent une convergence pour n'importe quel problème, et d'autre part, le benchmark (cf. figure 7.10) réalisé en utilisant différentes tailles de matrices tri-diagonales permet de confirmer ce choix.

7.7 Comparaison et validation

La résolution de l'équation de Poisson en PGD est lancée sur CPU et sur GPU (cf. 7.2) avec les mêmes paramètres afin de comparer les résultats obtenus dans les deux cas. Pour reconstruire la solution 5 modes sont retenus comme illustrés par la figure 7.22.

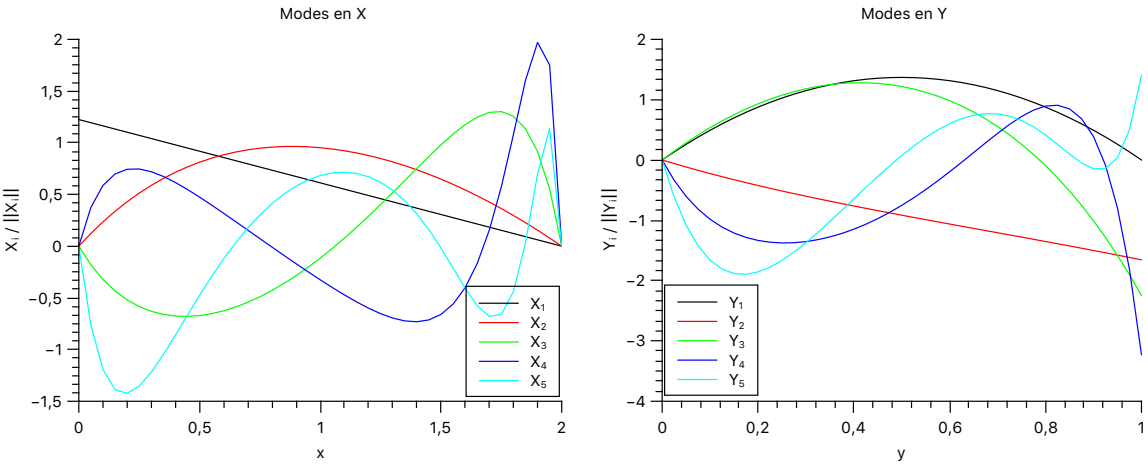


FIGURE 7.22 – Tracé des modes pour l'équation de Poisson sur GPU.

En considérant 10 modes dans chaque direction, on illustre par la figure 7.23 la différence entre les résultats obtenues sur GPU et sur CPU en échelle semi-logarithmique.

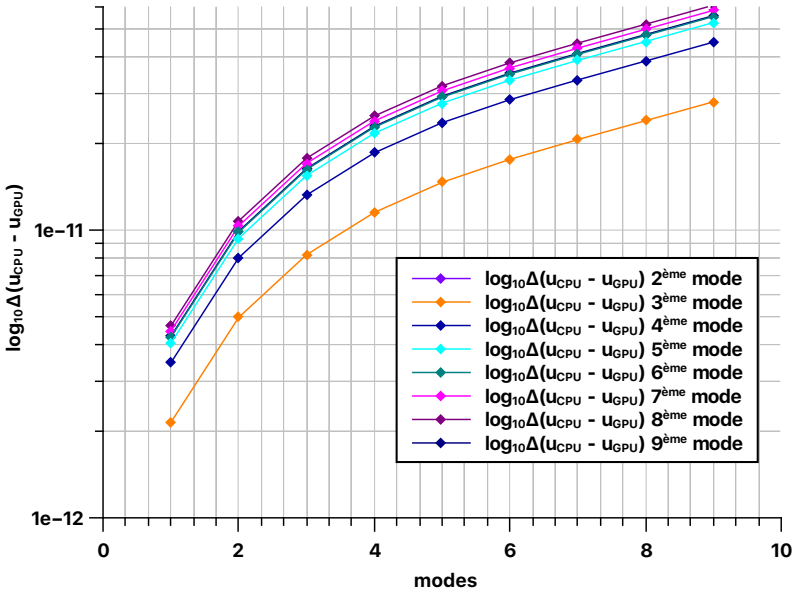
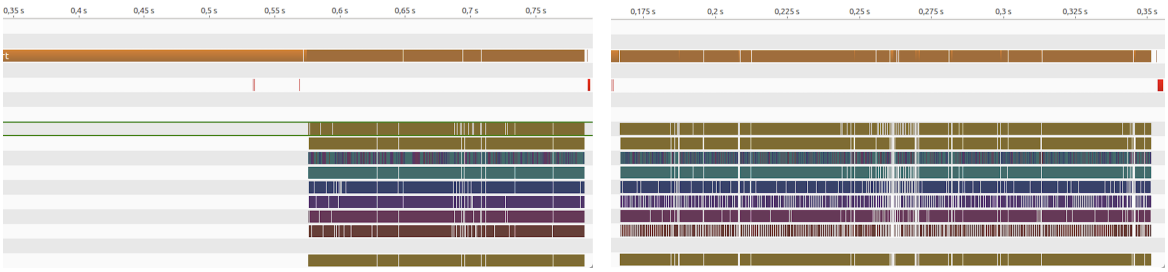


FIGURE 7.23 – Comparaison d'une première approche de l'implémentation GPU en échelle semi-logarithmique.

7.8 Analyse des performances

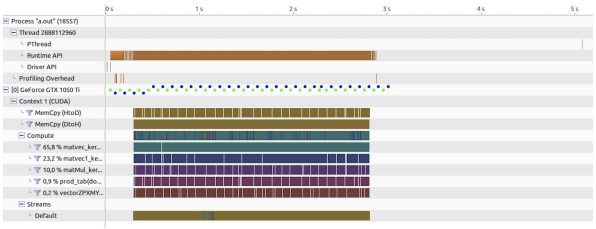
La figure 7.24a et 7.24b représentent le profilage du code GPU en utilisant respectivement le mode synchrone et asynchrone, on remarque que l'on gagne un facteur 2, nous regarderons donc de plus près ce profil sur un modèle de plus grande taille ($N_{X_1} = 500$, $N_{X_2} = 500$) représenté par les figures 7.25a et 7.25b.



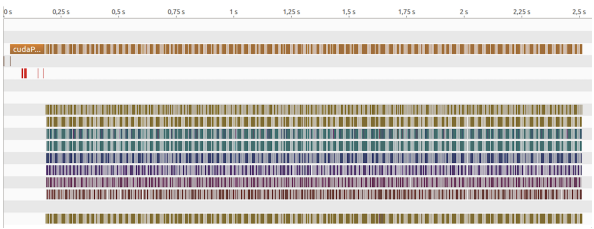
(a) Profiler en utilisant le mode synchrone (b) Profiler en utilisant le mode asynchrone

FIGURE 7.24 – Profiler le code GPU en utilisant NVIDIA Visual Profiler.

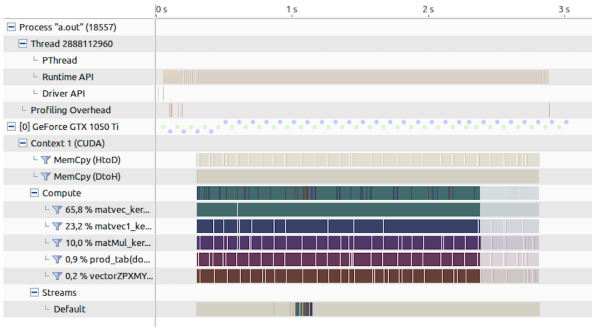
La figure 7.25c représente le taux d'occupation qui est défini comme étant le rapport entre le nombre de warps en cours d'exécution (celui-ci est considéré comme actif à partir du moment où tous ses threads commencent à s'exécuter jusqu'à leur sortie du kernel) et le nombre maximum de warps qui peuvent simultanément fonctionner sur un SM. Une faible occupation entraîne une faible efficacité du logiciel parallèle à cause du faible nombre de warps éligibles pour masquer la latence entre les différentes instructions dépendantes. Une importante utilisation des mémoires partagées/registres dans chaque thread peuvent entraîner des répercussions sur le nombre de warps qui peuvent simultanément être lancés.



(a) Profiler en utilisant le mode synchrone



(b) Profiler en utilisant le mode asynchrone



(c) Profiler en utilisant le mode synchrone

(d) Profiler en utilisant le mode asynchrone

FIGURE 7.25 – Profiler le code GPU en utilisant NVIDIA Visual Profiler.

7.9 Bilan et perspectives

Sur la base des tests numériques effectués dans cette étude, un programme PGD fonctionnel aussi bien sur le CPU que sur le GPU est mis en place. Une démarche de validation systématique est effectuée, les résultats obtenus jusqu'à présent sont encourageants et montrent qu'on peut obtenir de très bonnes performances, néanmoins, faute de temps, on n'a pas eu l'occasion de creuser encore plus de ce côté.

De nombreuses perspectives se dégagent, en effet, la compute capability actuelle permet l'utilisation de plusieurs librairies notamment le cuBLAS, le cuSolver, ... Au delà de cet aspect, il est aussi possible de porter plusieurs autres parties du code sur GPU, ce qui permet de

réaliser moins d'aller/retour [9] entre *host* et *device* possible, ceci permet d'augmenter le taux d'occupation et par suite éviter les latences mémoires. Les différentes mémoires n'ont pas été exploitées au cours de ce travail, c'est une des perspectives à aborder par la suite. En effet, il est nécessaire de faire des modifications de l'algorithme et du code du kernel afin d'utiliser les mémoires partagées.

Le concept de mémoire partagée spécifie les caractéristiques de la mémoire et du système de communication entre les processus et permet à plusieurs processus d'accéder au même segment de mémoire. Ces données sont disponibles à partir des différents processus. Pour plus de détails, on envoie le lecteur intéressé à l'Annexe A.

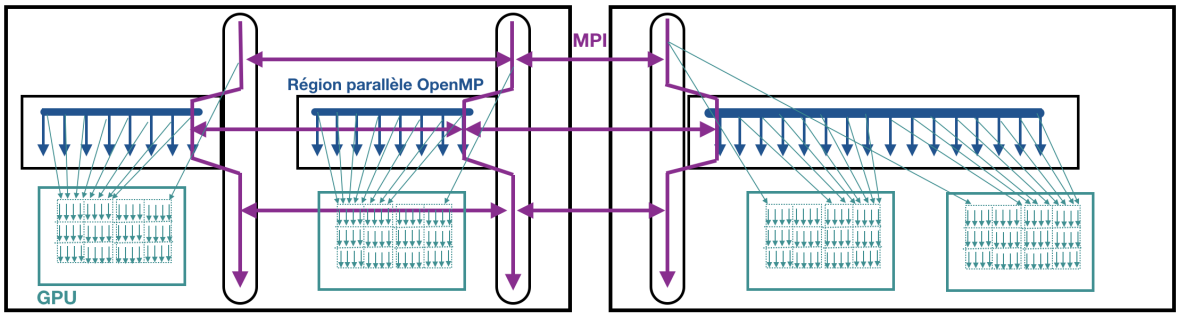


FIGURE 7.26 – Architecture hybride.

Cette première partie a mis en évidence l'intérêt du portage GPU, il est toutefois possible d'améliorer encore plus les performances d'une part parce que CUDA est en plein développement, et d'autre part, par le passage à la parallélisation dynamique, le multi-GPU (CUDA, OpenACC), en plus de la prallélisation CPU (MPI, OpenMP) afin d'exploiter la notion de parallélisation hybride comme illustré par la figure 7.26.

Le principe de la programmation hybride est de combiner plusieurs technologies en une seule application. Plusieurs questions se posent, quelle technologie utiliser et à quel endroit ? MPI est un paradigme de parallélisation développé initialement pour les machines à mémoire distribuée, les mécanismes de la mémoire partagée ne sont donc pas bien gérés par MPI. Il existe d'autres technologies spécifiques à la mémoire partagée comme OpenMP qui peut être appliquée dans les nœuds à mémoire partagée puis utilisée MPI entre les nœuds. De plus amples détails sont apportés dans l'Annexe A.

9. L'utilisation des directives de compilations ; le calcul est effectué sur le device, néanmoins il n'est guère nécessaire de recopier la totalité des résultats sur CPU, dans ce cas de figure le compilateur génère le code le plus efficace possible.

Annexe A

CUDA

A.1 Niveaux de mémoire

La figure A.1 illustre les niveaux de mémoire, la mémoire globale qui est de l'ordre du GB est différente de la mémoire de l'hôte. Chaque block a une mémoire cache dans laquelle sont stockées les données, ensuite chaque thread détient une variable "registers" qui lui est propre, ces différents threads communiquent entre eux à travers la "shared memory", néanmoins, une attention particulière doit être porter aux différentes lectures/écritures des différents threads afin d'éviter les conflits.

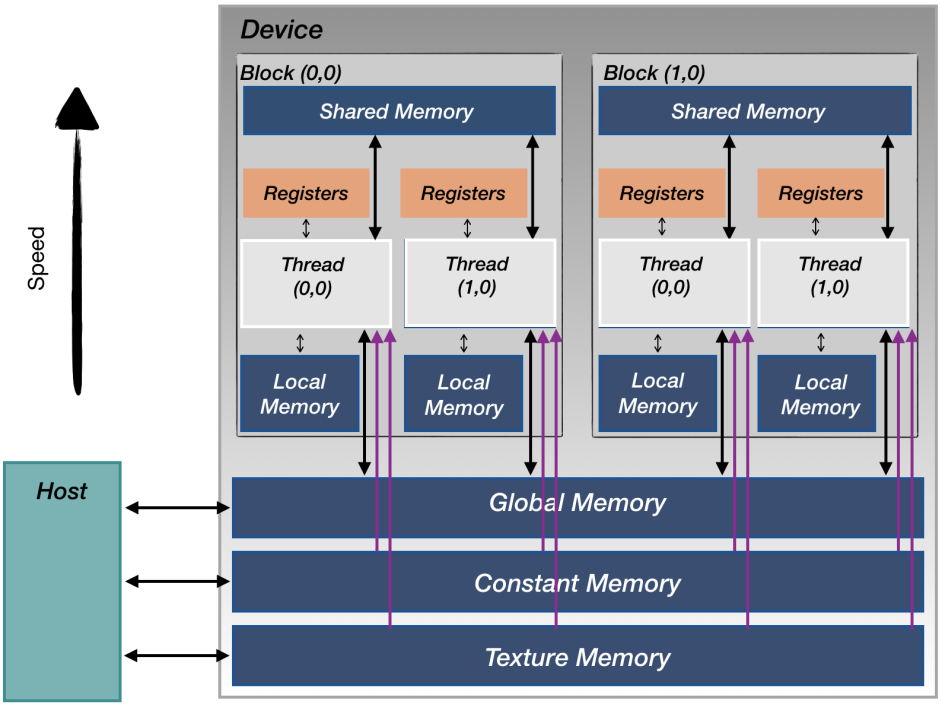


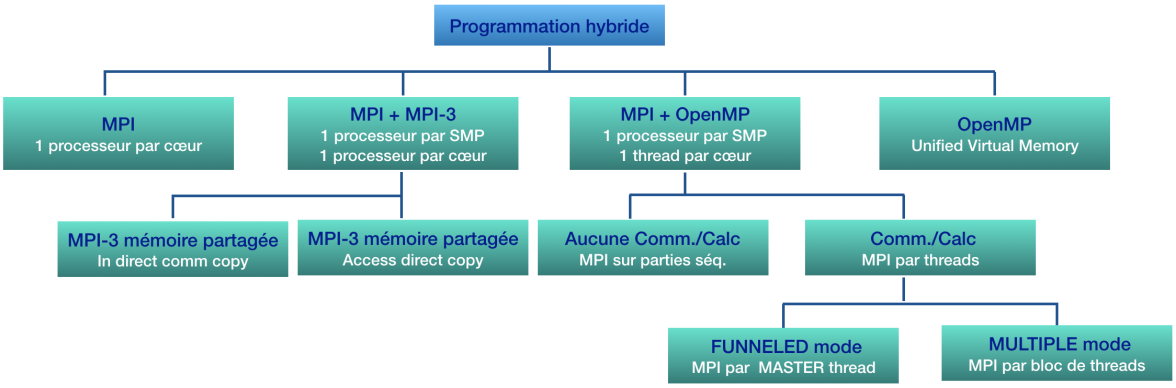
Figure A.1 – Le modèle de mémoire GPU.

Les différents threads accèdent aux différentes données dont ils ont besoin pour écrire leurs résultats, ce qui implique plusieurs accès à la mémoire globale. Afin de diminuer ce nombre d'accès, on accède à la mémoire globale une fois pour copier la donnée dans la mémoire partagée, ensuite les threads stockent dans les registres les résultats intermédiaires, les différents

appels se font à partir de cette mémoire pour finalement stocker le résultat finale dans la mémoire globale, ce qui permet d'aller plus vite. Il est à noter que **plus on s'éloigne d'un thread, plus longs sont les calculs**.

A.2 Programmation hybride

Le diagramme ci-dessous résume le panel de configurations possibles en programmation hybride. Il est tout à fait possible de réaliser un calcul MPI pur ou OpenMP pur, néanmoins ce n'est pas la partie que l'on souhaite approfondir. Ensuite, on trouve la configuration MPI+OpenMP, caractérisée par un processus MPI par multi-processeur en plus d'un thread OpenMP par cœur de calcul, cette configuration permet deux cas de figure. Le premier, il n'existe pas de recouvrement de comm./calcul dans ce cas MPI est utilisé uniquement dans les régions séquentiels du programme. Des appels MPI par threads OpenMP sont lancés au niveau des régions parallèles, lorsqu'il faut communiquer pendant les calculs. Selon le niveau de support par thread deux modes sont possibles ; le mode FUNNELED où les appels MPI sont impérativement lancés par le thread Master, sinon lorsque le support MULTIPLE est utilisé les threads sont libres.



Après, il existe un hybride MPI pur + MPI-3 qui permet de réaliser des communications en mémoire partagée. En effet, le niveau 3 du support standard propose un équivalent à OpenMP au niveau de la mémoire partagée. Il est toutefois relativement récent et toutes les librairies ne le supportent pas ou juste à moitié.

Annexe B

Développement de la PGD sur GPU

```
for((*var_glob).i=N0; (*var_glob).i<= (*var_glob).Max_terms; (*var_glob).i++){
    for((*var_glob).j=1; (*var_glob).j<=(*var_glob).k_fp; (*var_glob).j++){

        // ....

        /* N+1 */
        if((*var_glob).i>N0){
            matvec_kernel<<<blocksPerGrid,THREADS_PER_BLOCK>>>(d_Py, d_My, d_res
                , (*var_glob).Ny , (*var_glob).Ny);
            matMul_kernel<<<blocksPerGrid,THREADS_PER_BLOCK>>>(d_res, d_Yt1.
                elements, d_Gamma_x_i, (*var_glob).Ny , (*var_glob).Max_terms);
            matvec1_kernel<<<blocksPerGrid,THREADS_PER_BLOCK>>>(d_Py, d_Ky, d_Ty
                , (*var_glob).Ny , (*var_glob).Ny);
            matMul_kernel<<<blocksPerGrid,THREADS_PER_BLOCK>>>(d_Ty, d_Yt1.
                elements, d_Delta_x_i, (*var_glob).Ny , (*var_glob).Max_terms);
            matvec_kernel<<<blocksPerGrid,THREADS_PER_BLOCK>>>(d_Gamma_x_i,
                d_Xt1, d_res, (*var_glob).Ny , (*var_glob).Max_terms);
            matvec_kernel<<<blocksPerGrid,THREADS_PER_BLOCK>>>(d_res, d_Kx,
                d_resx, (*var_glob).Ny , (*var_glob).Ny);
            matvec_kernel<<<blocksPerGrid,THREADS_PER_BLOCK>>>(d_Delta_x_i,
                d_Xt1, d_res, (*var_glob).Ny , (*var_glob).Max_terms);
            matvec_kernel<<<blocksPerGrid,THREADS_PER_BLOCK>>>(d_res, d_Mx, d_Ty
                , (*var_glob).Ny , (*var_glob).Ny);
            vectorZPXMY<<<blocksPerGrid,THREADS_PER_BLOCK>>>(d_resx, d_Ty,
                d_RHSx , (*var_glob).Ny);
        }/* Fin N+1 */

        cudaMemcpyAsync((*var_glob).RHSx->tab, d_RHSx,(*var_glob).Ny*sizeof(
            double),cudaMemcpyDeviceToHost ) ;
        cudaThreadSynchronize();

        // ....

    } /* FIN phase de point fixe */
} /* FIN phase d'enrichissement '*/
```

```
for((*var_glob).i=N0; (*var_glob).i<= (*var_glob).Max_terms; (*var_glob).i++){
    for((*var_glob).j=1; (*var_glob).j<=(*var_glob).k_fp; (*var_glob).j++){

       // ....

        cudaMemcpyAsync( d_Py, (*var_glob).Py->tab, (*var_glob).Ny*sizeof(double
            ),cudaMemcpyHostToDevice ) ;
        matvec_kernel<<<blocksPerGrid,THREADS_PER_BLOCK>>>(d_Py, d_My, d_Ty, (*
            var_glob).Ny , (*var_glob).Ny);
        prod_tab<<<blocksPerGrid,THREADS_PER_BLOCK>>>( d_Ty, d_Py,d_h );
        cudaMemcpyAsync( h, d_h,blocksPerGrid*sizeof(double),
            cudaMemcpyDeviceToHost ) ;
        cudaThreadSynchronize();

        (*var_glob).alpha_x = 0;
        for (int il=0; il<blocksPerGrid; il++) {
            (*var_glob).alpha_x += h[il];

        }

        prod_tab<<<blocksPerGrid,THREADS_PER_BLOCK>>>( d_Ty, d_Fy,d_h );
        cudaMemcpyAsync( h, d_h,blocksPerGrid*sizeof(double),
            cudaMemcpyDeviceToHost ) ;
        cudaThreadSynchronize();


        (*var_glob).ksi_x_i = 0;
        for (int il=0; il<blocksPerGrid; il++) {
            (*var_glob).ksi_x_i += h[il];
        }
        matvec1_kernel<<<blocksPerGrid,THREADS_PER_BLOCK>>>(d_Py, d_Ky, d_Tky,
            (*var_glob).Ny , (*var_glob).Ny);
        prod_tab<<<blocksPerGrid,THREADS_PER_BLOCK>>>( d_Tky, d_Py,d_h );
        cudaMemcpyAsync( h, d_h,blocksPerGrid*sizeof(double),
            cudaMemcpyDeviceToHost ) ;
        cudaThreadSynchronize();

        (*var_glob).beta_x = 0;
        for (int il=0; il<blocksPerGrid; il++) {
            (*var_glob).beta_x += h[il];
        }

        SAXPBY(Ty, (*var_glob).ksi_x_i, 1, (*var_glob).Fx, ones_Vecteur((*
            var_glob).Nx));
        cudaMemcpyAsync( d_Ty, Ty->tab, (*var_glob).Ny*sizeof(double),
            cudaMemcpyHostToDevice ) ;
        cudaThreadSynchronize();

        matvec_kernel<<<blocksPerGrid,THREADS_PER_BLOCK>>>(d_Ty, d_Mx, d_RHSx,
            (*var_glob).Ny , (*var_glob).Ny);

       // ....

    } /* FIN phase de point fixe */
} /* FIN phase d'enrichissement */
```

Bibliographie

[1] *3D Printing : A Manufacturing Revolution*. AT Kearney 2015 (cf. p. 9).

[2] R. A Adams et John J. F Fournier. *Sobolev spaces*. English. OCLC : 180703211. Amsterdam ; Boston : Academic Press, 2003 (cf. p. 48).

[3] A Agouzoul, F Poulhaon et P Joyot. « Analyse inverse des déformations inhérentes à l'aide d'un modèle réduit paramétrique. Application au procédé SLM. » fr. In : (), p. 7 (cf. p. 87).

[4] Asmaâ Agouzoul, Fabien Poulhaon et Pierre Joyot. « Model reduction method for the simulation of the selective laser melting process ». In : Vitoria-Gasteiz, Spain, 2019, p. 150008 (cf. p. 87, 91).

[5] Nissrine Akkari. « Mathematical study of the sensitivity of the POD method (Proper orthogonal decomposition) ». Theses. Université de La Rochelle, déc. 2012 (cf. p. 33).

[6] Grégoire Allaire et Lukas Jakabčin. « Taking into account thermal residual stresses in topology optimization of structures built by additive manufacturing ». en. In : *Mathematical Models and Methods in Applied Sciences* 28.12 (nov. 2018), p. 2313-2366 (cf. p. 8).

[7] Grégoire Allaire et al. « Shape optimization of a layer by layer mechanical constraint for additive manufacturing ». en. In : *Comptes Rendus Mathematique* 355.6 (juin 2017), p. 699-717 (cf. p. 8).

[8] C. Allery, C. Béghein et A. Hamdouni. « Applying proper orthogonal decomposition to the computation of particle dispersion in a two-dimensional ventilated cavity ». en. In : *Communications in Nonlinear Science and Numerical Simulation* 10.8 (déc. 2005), p. 907-920 (cf. p. 35).

[9] C. Allery et al. « A priori reduction method for solving the two-dimensional Burgers' equations ». en. In : *Applied Mathematics and Computation* 217.15 (avr. 2011), p. 6671-6679 (cf. p. 56).

[10] Pierre-Eric Allier. « Error control for and with PGD reduced models ». Theses. Université Paris-Saclay, nov. 2017 (cf. p. 68, 71).

[11] A. Ammar, F. Chinesta et A. Falcó. « On the Convergence of a Greedy Rank-One Update Algorithm for a Class of Linear Systems ». en. In : *Archives of Computational Methods in Engineering* 17.4 (déc. 2010), p. 473-486 (cf. p. 67).

[12] Douglas N. Arnold. « An Interior Penalty Finite Element Method with Discontinuous Elements ». en. In : *SIAM Journal on Numerical Analysis* 19.4 (août 1982), p. 742-760 (cf. p. 90).

[13] Nelly Barrau. « Généralisation de la méthode Nitsche XFEM pour la discrétisation de problèmess d'interface elliptiques ». Thèse de doctorat dirigée par Luce, Robert Mathématiques appliquées Pau 2013. Thèse de doct. 2013 (cf. p. 58, 60, 91).

[14] Roland Becker, Peter Hansbo et Rolf Stenberg. « A finite element method for domain decomposition with non-matching grids ». en. In : *ESAIM : Mathematical Modelling and Numerical Analysis* 37.2 (mar. 2003), p. 209-225 (cf. p. 58).

[15] Michel Bellet. « Thermomechanical modelling of solidification and plastic deformation processes ». Habilitation à diriger des recherches. Université Nice Sophia Antipolis, juil. 2005 (cf. p. 24).

[16] Luca Boucinha. « Réduction de modèle a priori par séparation de variables espace-temps : Application en dynamique transitoire ». Thèse de doctorat dirigée par Gravouil, Anthony Mécanique Lyon, INSA 2013. Thèse de doct. 2013 (cf. p. 34, 39, 67).

[17] Vincent Boulos. « Programming model for the implementation of 2D-3D image processing applications on a hybrid CPU-GPU cluster ». Theses. Université de Grenoble, déc. 2012 (cf. p. 102).

[18] Haïm Brézis. *Analyse fonctionnelle : théorie et applications ; [cours ; master, agrégation]*. fre. Nouv. présentation. Mathématiques appliquées pour le master. OCLC : 934125232. Paris : Dunod, 2005 (cf. p. 48).

[19] Damien Buchbinder et al. « Investigation on reducing distortion by preheating during manufacture of aluminum components using selective laser melting ». en. In : *Journal of Laser Applications* 26.1 (fév. 2014), p. 012004 (cf. p. 88, 89).

[20] Nicolas Bur. « Reduced order model algorithms for Automated Tape Placement optimisation ». Theses. Université de Technologie de Compiègne, avr. 2015 (cf. p. 60, 66, 67).

[21] Nicolas Bur et al. « On the use of model order reduction for simulating automated fibre placement processes ». In : *Advanced Modeling and Simulation in Engineering Sciences* 3 (2016), p. 4 (cf. p. 33, 38, 67).

[22] Gianni Campatelli et al. « Integrated WAAM-Subtractive Versus Pure Subtractive Manufacturing Approaches : An Energy Efficiency Comparison ». en. In : *International Journal of Precision Engineering and Manufacturing-Green Technology* (mar. 2019) (cf. p. 12).

[23] Victor Chastand. « Studying the mechanical behaviour and the damaging mechanisms of metallic parts produced by additive manufacturing ». Theses. Ecole Centrale de Lille, nov. 2016 (cf. p. 8).

[24] Francisco Chinesta, Roland Keunings et Adrien Leygue. *The Proper Generalized Decomposition for Advanced Numerical Simulations*. en. SpringerBriefs in Applied Sciences and Technology. Cham : Springer International Publishing, 2014 (cf. p. 68).

[25] Franz Chouly, Patrick Hild et Yves Renard. « Symmetric and non-symmetric variants of Nitsche's method for contact problems in elasticity : theory and numerical experiments ». en. In : *Mathematics of Computation* 84.293 (oct. 2014), p. 1089-1112 (cf. p. 90).

[26] Philippe G. Ciarlet. *The finite element method for elliptic problems*. Studies in mathematics and its applications v. 4. Amsterdam ; New York : New York : North-Holland Pub. Co. ; sole distributors for the U.S.A. et Canada, Elsevier North-Holland, 1978 (cf. p. 48, 90).

[27] T.W. Clyne. « Residual Stresses in Surface Coatings and Their Effects on Interfacial Debonding ». en. In : *Key Engineering Materials* 116-117 (déc. 1995), p. 307-330 (cf. p. 84).

[28] L. Cordier et M. Bergmann. « Proper Orthogonal Decomposition : an overview ». In : *Lecture series 2002-04, 2003-03 and 2008-01 on post-processing of experimental and numerical data, Von Karman Institute for Fluid Dynamics, 2008*. VKI, 2008, 46 pages (cf. p. 35).

[29] L Cordier et M Bergmann. *Réduction de dynamique par Décomposition Orthogonale aux Valeurs Propres (POD 1)*. Rapp. tech. (cf. p. 33).

[30] Elías Cueto, David González et Icíar Alfaro. *Proper Generalized Decompositions*. en. SpringerBriefs in Applied Sciences and Technology. Cham : Springer International Publishing, 2016 (cf. p. 66).

[31] Philippe Dufrenoy et al. « ÉTUDE EXPÉRIMENTALE ET MODÉLISATION DU COMPORTEMENT THERMOMÉCANIQUE ». fr. In : (), p. 250 (cf. p. 16).

[32] Carl Eckart et Gale Young. « The approximation of one matrix by another of lower rank ». en. In : *Psychometrika* 1.3 (sept. 1936), p. 211-218 (cf. p. 35).

[33] Mathieu Fabre. « Fictitious domain methods for finite element methods, application to structural mechanics ». Theses. INSA de Lyon, juil. 2015 (cf. p. 58).

[34] Gilberto Fontecha Dulcey. « Parametric, reduced and multiscale model for the interactive optimization of laminated composite structures ». Theses. Université de Bordeaux, déc. 2018 (cf. p. 39).

[35] A. Fritz, S. Hueber et B.I. Wohlmuth. « A comparison of mortar and Nitsche techniques for linear elasticity ». en. In : *Calcolo* 41.3 (oct. 2004), p. 115-137 (cf. p. 90).

[36] Felix Fritzen, Max Hodapp et Matthias Leuschner. « GPU accelerated computational homogenization based on a variational approach in a reduced basis framework ». en. In : *Computer Methods in Applied Mechanics and Engineering* 278 (août 2014), p. 186-217 (cf. p. 117).

[37] C. Ghnatios et al. « Proper generalized decomposition based dynamic data-driven control of material forming processes ». In : *AIP Conference Proceedings*. 2011 (cf. p. 33).

[38] Chady Ghnatios. « Advanced Simulation of Thermal Problems Encountered in Composite Forming Processes ». Theses. Ecole Centrale de Nantes (ECN), oct. 2012 (cf. p. 66).

[39] Eugenio Giner et al. « The Proper Generalized Decomposition (PGD) as a numerical procedure to solve 3D cracked plates in linear elastic fracture mechanics ». en. In : *International Journal of Solids and Structures* 50.10 (mai 2013), p. 1710-1720 (cf. p. 33, 93).

[40] John Goldak, Aditya Chakravarti et Malcolm Bibby. « A new finite element model for welding heat sources ». en. In : *Metallurgical Transactions B* 15.2 (juin 1984), p. 299-305 (cf. p. 25).

[41] Meng Guo et al. « Investigating the generation process of molten droplets and arc plasma in the confined space during compulsively constricted WAAM ». en. In : *Journal of Materials Processing Technology* 275 (jan. 2019), p. 116355 (cf. p. 13).

[42] D.B. Go H. Peng et R. Billo. « Part-Scale Model for fast prediction of thermal distortion in DMLS Additive Manufacturing. Part 2 : A Quasi-Static Thermomechanical Model ». In : Solid Freeform Fabrication Symposium. 2016, p. 382-397 (cf. p. 21).

[43] Bernard Haasdonk. « Reduced Basis Methods for Parametrized PDEs – A Tutorial Introduction for Stationary and Instationary Problems ». en. In : (), p. 66 (cf. p. 33).

[44] Anita Hansbo et Peter Hansbo. « An unfitted finite element method, based on Nitsche's method, for elliptic interface problems ». en. In : *Comput. Methods Appl. Mech. Engrg.* (2002), p. 16 (cf. p. 59).

[45] Peter Hansbo. « Nitsche's method for interface problems in computational mechanics : Nitsche's method for interface problems in computational mechanics ». en. In : *GAMM-Mitteilungen* 28.2 (nov. 2005), p. 183-206 (cf. p. 59).

[46] Peter Hansbo et Mats G. Larson. « Discontinuous Galerkin methods for incompressible and nearly incompressible elasticity by Nitsche's method ». en. In : *Computer Methods in Applied Mechanics and Engineering* 191.17-18 (fév. 2002), p. 1895-1908 (cf. p. 90).

[47] M.R. Hill et D.V. Nelson. « The inherent strain method for residual stresses determination and its application to a long welded joint ». In : t. 318. 1995, p. 343-352 (cf. p. 29).

[48] *Interreg Poctefa TRANSFRON3D project*. https://www.transfron3d.eu/fr/. 2018 (cf. p. 89).

[49] Jeff Irwin et P Michaleris. « A Line Heat Input Model for Additive Manufacturing ». en. In : (2015), p. 11 (cf. p. 44).

[50] Ram Dayal Jodhpur. « Numerical modeling of process governing selective laser sintering ». en. In : (2014) (cf. p. 44).

[51] P. Jolivet et al. « High performance domain decomposition methods on massively parallel architectures with freefem++ ». en. In : *Journal of Numerical Mathematics* 20.3-4 (jan. 2012) (cf. p. 117).

[52] Pierre Joyot et al. « La méthode pgd appliquée à l'équation de la chaleur non-linéaire : vers une formulation performante ». fr. In : (), p. 9 (cf. p. 66).

[53] Renaud Julien. « Thermomechanical behaviour and microstructural evolution of a forged $\alpha+\beta$ – Ti-6Al-4V alloy during quenching : experiments, analysis and modelling ». Theses. Ecole des Mines d'Albi-Carmaux, fév. 2017 (cf. p. 16).

[54] Sasan Karamizadeh et al. « An Overview of Principal Component Analysis ». In : *Journal of Signal and Information Processing* 04.03 (2013), p. 173-175 (cf. p. 34).

[55] Nils Keller et Vasily Ploshikhin. « New method for fast predictions of residual stress and distorsion of AM parts ». en. In : (), p. 10 (cf. p. 25, 82, 86).

[56] Jean-Pierre Kruth et al. « Assessing and comparing influencing factors of residual stresses in selective laser melting using a novel analysis method ». en. In : *Proceedings of the Institution of Mechanical Engineers, Part B : Journal of Engineering Manufacture* 226.6 (juin 2012), p. 980-991 (cf. p. 88).

[57] E. Komi L. D'Alvise, P. Kokkonen et A. Majumda. « Numerical Simulation of the Selective Laser Melting Process to Support Defect Tolerant Design ». In : nov. 2016 (cf. p. 30).

[58] P. Ladevèze, J. C. Passieux et D. Néron. « The LATIN multiscale computational method and the Proper Generalized Decomposition ». In : *Computer Methods in Applied Mechanics and Engineering* (2010) (cf. p. 33).

[59] Pierre Ladevèze. « On a family of algorithms for structural mechanics ». In : 1985 (cf. p. 38).

[60] P. Ladevèze, J.-C. Passieux et D. Néron. « The LATIN multiscale computational method and the Proper Generalized Decomposition ». en. In : *Computer Methods in Applied Mechanics and Engineering* 199.21-22 (avr. 2010), p. 1287-1296 (cf. p. 67).

[61] Florent Le Bourhis. « Predictive model for environmental impact assessment in additive manufacturing processes, metallic powder projection application ». Theses. Ecole Centrale de Nantes (ECN), juil. 2014 (cf. p. 9, 10).

[62] C. Li et al. « Residual Stress in Metal Additive Manufacturing ». en. In : *Procedia CIRP* 71 (2018), p. 348-353 (cf. p. 27).

[63] Y. C. Liang et al. « Proper orthogonal decomposition and its applications - Part I : Theory ». In : *Journal of Sound and Vibration* (2002) (cf. p. 33).

[64] Y.C. Liang et al. « PROPER ORTHOGONAL DECOMPOSITION AND ITS APPLICATIONS—PART I : THEORY ». en. In : *Journal of Sound and Vibration* 252.3 (mai 2002), p. 527-544 (cf. p. 34).

[65] E. Liberge et A. Hamdouni. « Reduced order modelling method via proper orthogonal decomposition (POD) for flow around an oscillating cylinder ». en. In : *Journal of Fluids and Structures* 26.2 (fév. 2010), p. 292-311 (cf. p. 35).

[66] Erwan Liberge. « Modèles réduits obtenus par la méthode de POD-Galerkin pour les problèmes d'interaction fluide structure ». en. In : (), p. 153 (cf. p. 35).

[67] Erwan Liberge, Mustapha Benaouicha et Aziz Hamdouni. *Application de la réduction de modèle à l'Interaction Fluide Structure*. Rapp. tech. 2007 (cf. p. 33).

[68] Shunyu Liu et Yung C. Shin. « Additive manufacturing of Ti6Al4V alloy : A review ». en. In : *Materials & Design* 164 (fév. 2019), p. 107552 (cf. p. 16).

[69] Xin Liu. « Numerical modeling and simulation of selective laser sintering in polymer powder bed ». en. In : (), p. 139 (cf. p. 44).

[70] Michel Loève. « Fonctions aleatoires de seconde ordre ». In : 1945 (cf. p. 34).

[71] A. Longuet et al. « A multiphase mechanical model for Ti–6Al–4V : Application to the modeling of laser assisted processing ». en. In : *Computational Materials Science* 46.3 (sept. 2009), p. 761-766 (cf. p. 21).

[72] Arnaud Longuet. « Modélisation du procédé de projection laser : Application au Ti-6Al-4V ». Thèse de doctorat dirigée par Cailletaud, Georges et Colin, Christophe Sciences et génie des matériaux Paris, ENMP 2010. Thèse de doct. 2010, 1 vol. (231 p.) (Cf. p. 20, 21, 25, 27, 46).

[73] Arnaud Longuet et al. « Modélisation de la fabrication directe de pièces par projection laser : application au Ti-6Al-4V ». In : *Matériaux 2006*. Dijon, France : xx, 2006, 11 p. (Cf. p. 23, 44).

[74] Yannick Louvigny. « L'optimisation topologique et la fabrication additive, amélioration de la chaine de conception ». fr. In : (2015), p. 5 (cf. p. 8).

[75] J. L Lumley. « The Structure of Inhomogeneous Turbulence ». en. In : *A. M. Yaglom and V. I. Tatarski, Eds., Atmospheric Turbulence and Wave Propagation, Nauka, Moscow* (1967), p. 166-178 (cf. p. 33).

[76] Habibou Maitournam. « Mécanique des structures anélastiques ». fr. In : (), p. 260 (cf. p. 29).

[77] Arif Masud, Timothy J. Truster et Lawrence A. Bergman. « A unified formulation for interface coupling and frictional contact modeling with embedded error estimation ». en. In : *International Journal for Numerical Methods in Engineering* 92.2 (oct. 2012), p. 141-177 (cf. p. 90).

[78] Santiago Montagud. « Real time simulation in non linear dynamics : application in soft robots ». Theses. Université de Bordeaux, déc. 2018 (cf. p. 39).

[79] Anthony Nouy. « A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations ». en. In : *Computer Methods in Applied Mechanics and Engineering* 196.45-48 (sept. 2007), p. 4521-4537 (cf. p. 67).

[80] Anthony Nouy. « A priori model reduction through Proper Generalized Decomposition for solving time-dependent partial differential equations ». en. In : *Computer Methods in Applied Mechanics and Engineering* 199.23-24 (avr. 2010), p. 1603-1626 (cf. p. 33, 67).

[81] Anthony Nouy. « Generalized spectral decomposition method for solving stochastic finite element equations : Invariant subspace problem and dedicated algorithms ». en. In : *Computer Methods in Applied Mechanics and Engineering* 197.51-52 (oct. 2008), p. 4718-4736 (cf. p. 67).

[82] David Néron et Pierre Ladevèze. « Proper Generalized Decomposition for Multiscale and Multiphysics Problems ». en. In : *Archives of Computational Methods in Engineering* 17.4 (déc. 2010), p. 351-372 (cf. p. 38).

[83] A.-F. Obaton et al. « Fabrication additive : état de l'art et besoins métrologiques engendrés ». fr. In : *Revue française de métrologie* 37 (mar. 2015), p. 21-36 (cf. p. 8).

[84] Matthieu Ospici. « Programming models and execution models for parallel and hybrid architectures. Application to physics simulations. » Theses. Université de Grenoble, juil. 2013 (cf. p. 32).

[85] Charles Paillet. « New model order reduction methods for problems with a high number of parameters ». Theses. Université Paris-Saclay, juin 2019 (cf. p. 71).

[86] M. Papadrakakis, G. Stavroulakis et A. Karatarakis. « A new era in scientific computing : Domain decomposition methods in hybrid CPU–GPU architectures ». en. In : *Computer Methods in Applied Mechanics and Engineering* 200.13-16 (mar. 2011), p. 1490-1508 (cf. p. 117).

[87] Olivier Parant. « Experimental study and calculation of residual stresses in extruded polyethylene tubes ». Theses. École Nationale Supérieure des Mines de Paris, déc. 2002 (cf. p. 84).

[88] Albert E. Patterson, Sherri L. Messimer et Phillip A. Farrington. « Overhanging Features and the SLM/DMLS Residual Stresses Problem : Review and Future Research Need ». en. In : *Technologies* 5.2 (avr. 2017), p. 15 (cf. p. 83).

[89] Berengere Podvin. « Introduction à la Décomposition Orthogonale aux Valeurs Propres ou P.O.D ». fr. In : (), p. 39 (cf. p. 33, 34).

[90] Rémi Ponche. « Design for additive manufacturing methodology, applied to the laser cladding process ». Theses. Ecole Centrale de Nantes (ECN), oct. 2013 (cf. p. 24).

[91] Paolo C. Priarone et al. « A modelling framework for comparing the environmental and economic performance of WAAM-based integrated manufacturing and machining ». en. In : *CIRP Annals* 68.1 (2019), p. 37-40 (cf. p. 13).

[92] D. Ryckelynck. « A priori hyperreduction method : an adaptive approach ». en. In : *Journal of Computational Physics* 202.1 (jan. 2005), p. 346-366 (cf. p. 56).

[93] David Ryckelynck. « Réduction a priori de modèles thermomécaniques ». en. In : *Comptes Rendus Mécanique* 330.7 (jan. 2002), p. 499-505 (cf. p. 56).

[94] V. A. Safronov et al. « Distortions and Residual Stresses at Layer-by-Layer Additive Manufacturing by Fusion ». en. In : *Journal of Manufacturing Science and Engineering* 139.3 (oct. 2016), p. 031017 (cf. p. 83, 84).

[95] V Saphronov, R S Khmyrov et A V Gusarov. « Experimental and theoretical study of residual deformations and stresses at additive manufacturing by fusion ». en. In : (), p. 5 (cf. p. 84).

[96] A Schneider, J Gardan et N Gardan. « Optimisation numérique en prototypage rapide ». fr. In : (2012), p. 18 (cf. p. 8).

[97] Iñaki Setien et al. « Empirical methodology to determine inherent strains in additive manufacturing ». en. In : *Computers & Mathematics with Applications* (juin 2018) (cf. p. 88).

[98] E Soylemez. « MODELING THE MELT POOL OF THE LASER SINTERED TI6AL4V LAYERS WITH GOLDAK'S DOUBLE-ELLIPSOIDAL HEAT SOURCE ». en. In : (), p. 16 (cf. p. 16).

[99] Gilbert Strang. « The Fundamental Theorem of Linear Algebra ». In : *The American Mathematical Monthly* 100.9 (nov. 1993), p. 848 (cf. p. 35).

[100] Ngoc Thuy Trinh. « Sur la modélisation du comportement thermomécanique et métallurgique des aciers. Application au procédé de soudage et de traitements thermiques ». Theses. Ecole Polytechnique X, juin 2008 (cf. p. 25).

[101] Timothy J. Truster et Arif Masud. « A Discontinuous/continuous Galerkin method for modeling of interphase damage in fibrous composite systems ». en. In : *Computational Mechanics* 52.3 (sept. 2013), p. 499-514 (cf. p. 90).

[102] Yukio Ueda et Keiji Fukuda. « New Measuring Method of Three-Dimensional Residual Stresses in Long Welded Joints Using Inherent Strains as Parameters—Lz Method ». en. In : *Journal of Engineering Materials and Technology* 111.1 (1989), p. 1 (cf. p. 80, 84).

[103] Yukio Ueda, Hidekazu Murakawa et Ninshu Ma. *Welding deformation and residual stress prevention*. en. 1st ed. Amsterdam ; Boston : Butterworth-Heinemann, 2012 (cf. p. 2, 80, 86).

[104] Yukio Ueda et al. « A New Measuring Method of Residual Stresses with the Aid of Finite Element Method and Reliability of Estimated Values ». In : *Journal of the Society of Naval Architects of Japan* 1975.138 (1975), p. 499-507 (cf. p. 21, 80).

[105] Laurent Van Belle. « Analysis, modeling and simulation of residual stresses during the SLM process of metallic powders ». Theses. INSA de Lyon, nov. 2013 (cf. p. 10, 12, 17, 20, 21, 23, 25, 27, 44).

[106] Benjamin Vayre. « Design for Additive Manufacturing, focus on EBM technology ». Theses. Université de Grenoble, juil. 2014 (cf. p. 9, 10).

[107] Adán Vega Sáenz et al. « Analysis and prediction of welding distortion in complex structures using elastic finite element method ». en. In : *Ciencia y tecnología de buques* 6.11 (juil. 2012), p. 35 (cf. p. 80).

[108] N. Verdon et al. « Reduced-order modelling for solving linear and non-linear equations ». en. In : *International Journal for Numerical Methods in Biomedical Engineering* 27.1 (jan. 2011), p. 43-58 (cf. p. 56).

[109] Nicolas Verdon. « A low-order dynamical system based on a APR-POD approach for studying turbulent flow-particles interaction ». Theses. Université de La Rochelle, déc. 2007 (cf. p. 37).

[110] Nicolas Verdon, Cyrille Allery et Aziz Hamdouni. « Résolution numérique des équations de transfert par une méthode de réduction a priori ». fr. In : (2007), p. 6 (cf. p. 56).

[111] Nicolas Verdon et al. « An adaptive ROM approach for solving transfer equations ». fr. In : *European Journal of Computational Mechanics* 15.5 (jan. 2006), p. 589-605 (cf. p. 56).

[112] Liang Wang et Sergio Felicelli. « Process Modeling in Laser Deposition of Multilayer SS410 Steel ». en. In : *Journal of Manufacturing Science and Engineering* 129.6 (2007), p. 1028 (cf. p. 21).

[113] *Wohlers report 2019 : 3D printing and additive manufacturing state of the industry.* English. OCLC : 1100022043. 2019 (cf. p. 9).

[114] Wei Xing et al. « Estimation of Residual Stress in Selective Laser Melting of a Zr-Based Amorphous Alloy ». en. In : *Materials* 11.8 (août 2018), p. 1480 (cf. p. 25).

[115] Kai Zeng, Deepankar Pal et Brent Stucker. « A review of thermal analysis methods in Laser Sintering and Selective Laser Melting ». en. In : (), p. 19 (cf. p. 24, 44).